Mid-report for Advanced Machine Learning project

Xiaoxiao CHEN XIAOXIAO.CHEN@TELECOM-PARISTECH.FR
Yuxiang WANG YUXIANG.WANG@U-PSUD.FR
Honglin LI HONG-LIN.LI@U-PSUD.FR

Abstract

The purpose of this middle report of the project is to present the problem we plan to solve and the progress of the project and also the problems we meet. The whole report will be divided into 5 parts: (1) Introduction: present the problem and our plan; (2) Recommendation system: present the data set and the specific methods we are using; (3) Evaluation: the evaluation metrics of this challenge; (4) Discussion and perspectives: all the problems we meet so far and their potential solutions; (5) Conclusion: the work need to be done and the future plan of the project.

1. Introduction

The Million Song Dataset Challenge aims at being the best possible offline evaluation of a music recommendation system. Any type of algorithm can be used: collaborative filtering, content-based methods, web crawling,etc. By relying on the Million Song Dataset, the data for the competition is completely open: almost everything is known and possibly available.

The specific task can be described briefly. The data we have is: 1) the full listening history for 1M users which is the original data set, 2) half of the listening history for 110K users, and another missing half should be predicted and compared to the results.

The most straightforward approach to this task is pure collaborative filtering, but there is a wealth of information available to the data scientist through the Million Song Dataset. In other words, the content-based methods have various possibilities and we have plenty of things to experiment which might bring us some surprising results.

For so far, we have met several problems in both theoretical aspect and engineering aspect. We will discuss the baseline method we have set and the potentials of other better methods. The different forms of evaluation metrics will also be discussed.

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

2. Recommendation system

Recommender systems or recommendation systems are a subclass of information filtering system that seek to predict the "rating" or "preference" that a user would give to an item. Recommender systems have become increasingly popular in recent years, and are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general. There are also recommender systems for experts, collaborators, jokes, restaurants, garments, financial services, life insurance, romantic partners (online dating), and Twitter pages.

2.1. Challenge problem

In our context, a music recommendation system need to be developed to predict the preferences of the existing users based on their listening history. The original data set consists of the listening history of 1M users, the challenge data is about 110K users. The data set is manually divided into train set and test set, each consists of half of the listening history.

Our mission is to use the train set to develop a model to do the prediction of the test set. The kaggle site used *Root mean square error* as their evaluation metric, but we are able to set our own metrics since the kaggle competition is already closed. According to the evaluation metric, the problem can be also regarded as classification problem or regression problem. The details will be discussed in section 3.

2.2. Data Analysis

There are four files available at kaggle site:

- kaggle_visible_evaluation_triplets.txt - this is the core data: triplets(*user,song,play count*). The play count

are generally considered equal as the rating of different songs from the user. The more a song played by a user, the more preference showed by the user on this song.

- kaggle_users.txt - this is the canonical list of user identifiers, which must be used to sort the predictions.
- kaggle_songs.txt - this is the canonical list of songs, which contains all song IDs from the training data and the test data.
- taste_profile_song_to_tracks.txt - this file includes the information of the corresponding track for each song, by this information we can extract more information(artist, release year) from the external sources.

Since the original data comes from kaggle site and the submission channel is already closed, we are not able to evaluate our results on-line. In order to solve this problem, which means find a reference values for our prediction results, we managed to find the evaluation results from the site http://labrosa.ee.columbia.edu/millionsong/challenge#data1. The visible part can be reconstructed as train data, and the hidden part can be used directly as evaluation data.

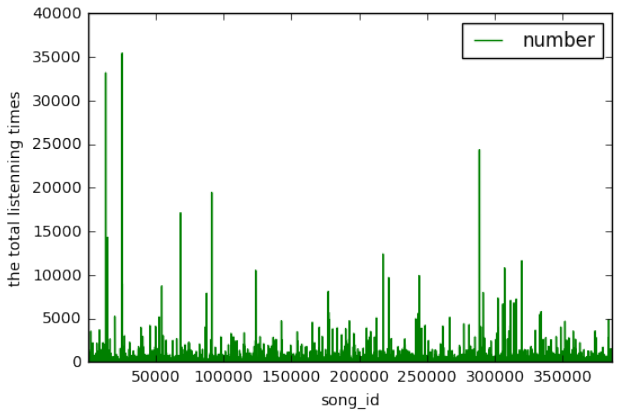

Figure 1. the distribution of the total number of each song in the list

The figure 1 show the total listening number of each song in the list, it can be seen from the figure, the disparity between different song is huge. The listening number of some songs is very little even zero, but some of them are more than 30k listening times. In fact, the song more popular will be more probable to be recommended. So, we can recommend directly the most popular song to all users.

The figure 2 shows us some special information different from figure 1. This figure shows the mean frequency of each song. In our natural mind, we think the song more popular will be listened more times for each person. In fact, it is not. Some popular song, it is just listened by many people, but each people will not take much time to repeat it, maybe it is not special but just popular. But for some special songs, maybe it is not so popular as the other, but it is special for some listener, maybe because its special style or its singer. So we can find that, for some special song, the fans of these song repeat more than 400 times. It is incredible that, if this song take 3 minutes, its fans take more than 20 hours to listen it. As the result, we can recommend these special songs to those special listener, maybe based on the artist, maybe based on the style of the song etc.

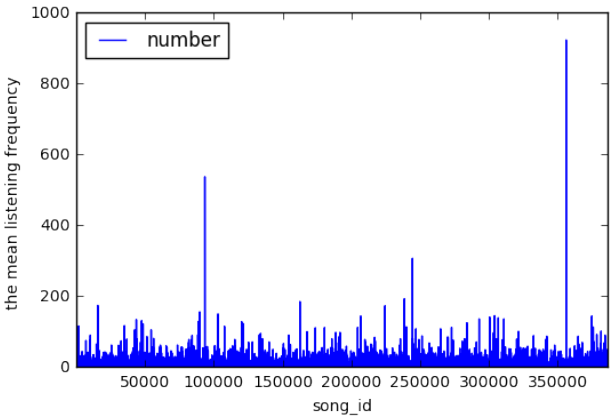

Figure 2. The mean listening frequency for each song in the list

2.3. Methods

Broadly speaking, recommender systems are based on one of two strategies. The **content filtering approach** 3 creates a profile for each user or product to characterize its nature. For example, a song profile could include attributes regarding its genre, the artist, and so forth. User profiles might include demographic information or answers provided on a suitable questionnaire. The profiles allow programs to associate users with matching products. Of course, content-based strategies require gathering external information that might not be available or easy to collect.But in this case all the attributes of the songs are available on the web site http://labrosa.ee.columbia.edu/millionsong/pages/example-track-description. A trained music analyst finally scores each song based on hundreds of distinct musical characteristics. These attributes, or genes, capture not only a songs musical identity but also many significant qualities that are relevant to understanding listeners musical preferences.

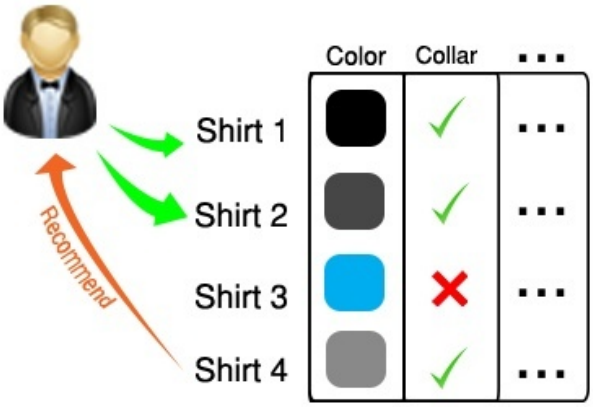

Figure 3. Content Filtering approach

An alternative to content filtering relies only on past user behavior. For example, previous transactions or product ratings without requiring the creation of explicit profiles. This approach is known as **collaborative filtering** 5. Collaborative filtering analyzes relationships between users and interdependencies among products to identify new user-item associations. A major appeal of collaborative filtering is that it is domain free, yet it can address data aspects that are often elusive and difficult to profile using content filtering. While generally more accurate than content-based techniques, collaborative filtering suffers from what is called the cold start problem, due to its inability to address the systems new products and users. In this aspect,content filtering is superior.

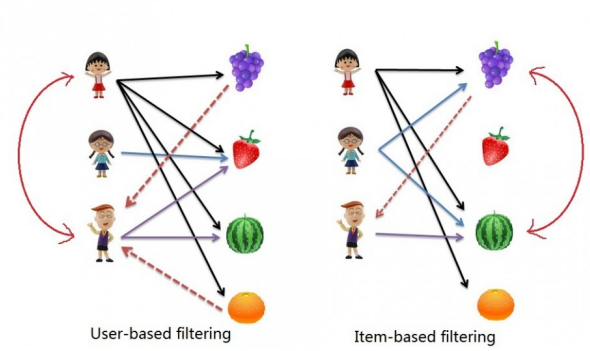

Figure 4. Collaborative Filtering

The two primary areas of collaborative filtering are the neighborhood methods and latent factor models. Neighborhood methods are centered on computing the relationships between items or, alternatively, between users. The item-oriented approach evaluates a users preference for an item based on ratings of neighboring items by the same user. A products neighbors are other products that tend to get similar ratings when rated by the same user.

Algorithm 1 User based with cosine similarity

Input: training_matrix $M(m,n)$, prediction_vector $y(1,n)$
Calculate the similarity between training_matrix and prediction_vector
for $row = 1$ **to** m **do**
$sim(m_{row}, y) = \frac{\sum_{i=1}^{n} m_i * yi}{\|m_{row}\|_2 * \|y\|_2}$
end for
Recombine the similarity
for $column = 1$ **to** n **do**
$r_{col} = \frac{\sum_{i=1}^{m} sim(m_i, y) * m_{i,col}}{\sum_{i=1}^{m} \|sim_i\|_2}$
end for

2.3.1. USER-BASED WITH COSINE SIMILARITY

We have tried the user-based with cosine similarity. known as vector-based similarity, we can use cosine similarity to present the user action similarity. For example, if the user A like song 1,2,3, and user B also like song 1,2,3, it is reasonable that the user A is similar to user B. Naturally, if user A like song 4 we can recommend it to user B. In this algorithm, we calculate firstly the similarities between the candidate and all the users, if the user is similar to the candidate, the rating of the candidate will be more similar to this user. In this algorithm, the similarity is treated as a weight to each user, and the combination of all the user and their similarity will be the rating of the candidate. After implementing the algorithm, We basically achieved a simple recommendation system with user-based. And we encountered some problems. Firstly, the size of the data is too big. When we translated the original data into a matrix, our memory was overloaded. In fact, we can not create a matrix with (110000,380000) which means the number of users and the number of songs. So for making the unit test for our algorithm, we selected the first 10 percents of data, which contains 840 users and 8087 songs. So we think we should find some other methods, because this algorithm is not suitable for lager number of users. Secondly, it is too bad for new user, because for a new user, we can not calculate the similarity . In other words, the similarity can not present the relation between old user and new user. However, this algorithm give us some ideas for recommendation system and return us an acceptable result.

2.4. Baseline method

As mentioned in the previous section, we can no longer evaluate our prediction result on-line, so we need to find a result as a standard baseline for reference. The challenge provides a *getting started file* to help us get familiar with

```
for user 0 recommend ['song_id2189', 'song_id6588', 'song_id6043', 'song_id3176', 'song_id7182', 'song_id5596', 'song
_id3354', 'song_id5795', 'song_id7251', 'song_id4024']

for user 1 recommend ['song_id2550', 'song_id1900', 'song_id4539', 'song_id5833', 'song_id2639', 'song_id3513', 'song
_id5959', 'song_id535', 'song_id6652', 'song_id6903']

for user 2 recommend ['song_id4843', 'song_id1790', 'song_id6999', 'song_id2333', 'song_id1083', 'song_id4129', 'song
_id3367', 'song_id1928', 'song_id7990', 'song_id2481']

for user 3 recommend ['song_id235', 'song_id3647', 'song_id5596', 'song_id2912', 'song_id6819', 'song_id533', 'song_i
d607', 'song_id998', 'song_id7745', 'song_id3779']

for user 4 recommend ['song_id8086', 'song_id2699', 'song_id2686', 'song_id2687', 'song_id2688', 'song_id2689', 'song
_id2690', 'song_id2691', 'song_id2692', 'song_id2693']

for user 5 recommend ['song_id5310', 'song_id3663', 'song_id3255', 'song_id1282', 'song_id6849', 'song_id7506', 'song
_id4149', 'song_id4185', 'song_id1688', 'song_id3524']

for user 6 recommend ['song_id6287', 'song_id7001', 'song_id5310', 'song_id3663', 'song_id6043', 'song_id1615', 'song
_id4015', 'song_id4028', 'song_id6849', 'song_id8023']

for user 7 recommend ['song_id1788', 'song_id8085', 'song_id4921', 'song_id3828', 'song_id3714', 'song_id1613', 'song
_id7724', 'song_id2295', 'song_id6867', 'song_id3182']

for user 8 recommend ['song_id7203', 'song_id238', 'song_id5781', 'song_id6043', 'song_id5300', 'song_id1111', 'song_
id959', 'song_id311', 'song_id6191', 'song_id1142']

for user 9 recommend ['song_id3393', 'song_id6947', 'song_id422', 'song_id1489', 'song_id6234', 'song_id2003', 'song_
id6313', 'song_id634', 'song_id6457', 'song_id2225']
```

Figure 5. The result of user-based with cosine similarity

Algorithm 2 Recommendation based on raw popularity

Input: triplets $(u_i, song_i, play\_discout_i)$, user_id u,triplet size m,limited size k
Initialize $x = dict(song, count), r = dict(user, songlist)$.
Count the number of each song played in the train set and sort as the
for $i = 1$ **to** m **do**
  $x_i = x_i + 1$
end for
x ordered descending order
recommend the most popular songs to all the users
for all users **do**
  if s from x and size $r < k$ **then**
    $r.append(s)$
  end if
end for

the data set and also its result can serve as a baseline to our model. The algorithm details are showed in Algorithm 2.

The system recommends the most popular songs among all the users to each user. They delete the song a specific user has already listened and recommend 500 new songs for each user. We can take a reference from the result, which shows a very basic way to recommend songs.

3. Evaluation

Evaluation is important in assessing the effectiveness of recommendation algorithms. The commonly used metrics are the mean squared error and root mean squared error.
Mean square error:
$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i^* - y_i)^2$
Root mean square error:
$RMSD = \sqrt{\frac{\sum_{i=1}^{n}(y_i^* - y_i)^2}{n}}$

Another way to evaluate is to transform the final problem to a classification problem. According to the play count, we can divide the value into subsection as a class. But this form requires enormous workload of encoding. We will study its viability.

4. Discussion and perspectives

4.1. problems

we have encountered some problems in our project.

- the size of data
  the size of data is too large in this challenge. There are 110k users and more than 380k songs in the data. And the training set contain more than 1500k rows of records. We can not calculate directly by our computer, even just calculate the mean of each row, our computer was overloaded.

- sparse data
  The data is too sparse. There are many zeros in each row. In general, each user listens less than 1000 songs, but this size of song is 380K. The sparse date lead to large error of prediction. For example, for the cosine similarity, more zeros means the similarity is less creditable.

- lack of information
  From the data set, we just know the user listened a song. But we have not enough information about the user and the song. It makes us too hard to improve our performance. So we think we need to find more information from other resource or other ways. For example, the lyrics of the song, the artist of the song etc.

- evaluation of performance
  For this challenge, it is difficult to evaluate the performance. The recommendation system, it is different from the classification or regression problem, because there are so many ways to evaluate the performance. In general, we can use MSE and RMSD to evaluate the loss. But some times, we do not even know whether our prediction is suitable for the user, because we can not get the current feed-beck. We assume that we have a new method which is total different to the others, it make a new recommendation to a user, maybe it is a new and interesting subject to this user, but it is different type from user's habit. But in fact, we can not judge whether it is good for this user and we just know it is different to this user's habit.

we would like to try some other methods in the following work.

4.2. potential methods

- Singular Value Decomposition:
  For a matrix M, its SVD is the factorization of M into three constituent matrices such that $M = U \sum T^T$, $\sum$ is a diagonal matrix whose values θ_i are the singular values of the composition, and both U and T are orthogonal. After reading some papers and document, we find that Singular value decomposition can reduce the dimensionality of a dataset, and it can improve the accuracy of prediction.

- Content-based method:
  Basically, the use an item profile (i.e., a set of discrete attributes and features) characterizing the item within the system. The system creates a content-based profile of users based on a weighted vector of item features. The weights denote the importance of each feature to the user and can be computed from individually rated content vectors using a variety of techniques.

  In our case, most of the fields about a specific track could be taken directly from the Echo Nest Analyze API. Some interesting attributes are listed below:

  1. artist - many fans like to listen especially the music of certain artists
  2. genre - rock and country has their own listener
  3. tempo & beats - rhythm is a significant character of certain genre of songs
  4. year - old music or new music

  Based on all these extra information, we have various possibilities to characterize each song and try to make a more predictive model.

- Hybrid recommender systems:
  Hybrid approaches can be implemented in several ways: by making content-based and collaborative-based predictions separately and then combining them; by adding content-based capabilities to a collaborative-based approach (and vice versa); or by unifying the approaches into one model. These methods can also be used to overcome some of the common problems in recommender systems such as cold start and the sparsity problem. We would like to study carefully the possibilities to integrate several models to reach a better performance.

5. Conclusion

In conclusion, we have finished our plan that we implemented our first version of recommendation system, and we got the result. Then, after reading some document and discussing with each other, we found our problems and tried to find some ways to solve them. In the following work, we need to implement the some new algorithms and try to recombine them.

6. References

[1] wikipedia about recommendation system
$https://en.wikipedia.org/wiki/Recommender_system$

[2] M.D.Ekstrand, J.T.Riedl and J.A.Konstan
Collaborative Filtering Recommender Systems, P89-P101

[3] Website of Mr.Antoine Cornujols
$www.agroparistech.fr/ufr-info/membres/cornuejols/Teaching/Master-AIC/$

[4] Slides of course of stanford
$http://infolab.stanford.edu/\ ullman/mmds/ch9.pdf$

[5] Slides of course of Mr. Xavier Amtriain
$http://fr.slideshare.net/xamat/recommender-systems-machine-learning-summer-school-2014-cmu$