Music Recommendation System on Digital Music Dataset

Bowei Feng
Cong Deng
Department of Computer Science, Rutgers University
{bowei.feng,cong.deng}@rutgers.edu

1 INTRODUCTION

Recommendation system has been widely used in many fields. It can be seen as a platform or engine which predicts user preferences and can recommend items to a user based on them. recommendation system can be used as playlist generators for music or videos in YouTube or Netflix. It can achieve product recommendation in Amazon or eBay. And it can also provide news recommendation in Bloomberg.

The problem this project tries to solve is a music rating prediction problem. The dataset we have has some ratings of some users to some music, However, there are also many vacancies. Our goal is to predict these vacancies and recommend items to users based on these predictions.

Many models can be used in recommendation system. We decide to use user-based collaborative filtering.

The experimental results will be described below in details.

2 RELATED WORK

Here are three types of recommendation system that are frequently used.

2.1 Content-Based Filtering

The Content-Based Recommendation relies on the similarity of the items being recommended. The basic idea is that if you like an item, then you will also like a similar item.The concepts of Term Frequency (TF) and Inverse Document Frequency (IDF) are used to pick important features such as author/actor/director to create an item profile for each music. Then, the user profile vectors are also created based on his actions on previous attributes of items and the similarity between an item and a user is also determined in a similar way.

2.2 Collaborative Filtering

The Collaborative Filtering Recommendation[1] is entirely based on the past behavior and not on the context. More specifically, it is based on the similarity in preferences, tastes and choices of two users. It analyzes how the tastes of one user are similar to another and makes recommendations based on that. In User-User Collaborative Filtering, we find look-alike users based on similarity and recommend musics which the first user has chosen in the past. Item-Item Collaborative Filtering is quite similar to previous algorithm, but instead of finding similar users, we try to find similar music.

2.3 Matrix Factorization

In the previous attempt, rating matrices may be overfitted to noisy representations of user tastes and preferences. In addition to this, it doesn't fit scale well to extremely large datasets because it needs a lot of computations to make recommendations. So, we can apply Dimensionality Reduction techniques to derive the tastes and preferences from the raw data, which is also known as low-rank matrix factorization.

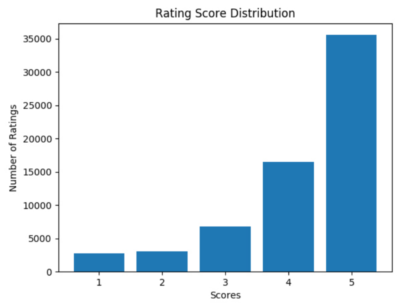
3 PRELIMINARIES

The dataset we use is from Amazon product data. The datasets have dataset for different type of product. Based on our hardware infrastructure and computing resources, we choose the 5-core dataset. This dataset contains product reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 - July 2014. We choose a digital music data with over 64000 ratings, which is applied to 3568 music by 5541 users. There are several parameters for each record for the dataset, e.g., reviewerID, asin, reviewerName, reviewText, overall, summary and reviewTime, and we only focus on reviewerID, asin and overall, they are userID, productID and rating respectively.

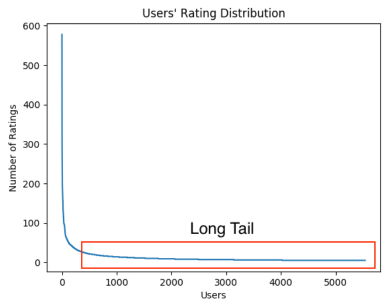
The dataset needs to be separated into two parts — training set and test set. The training set contains 80% of the data. The test set contains 20% of the data. Two ways are used in our project to split the data. We write splitData() to directly split the data into 2 parts. We also choose some of users and some of their ratings to get train/test dataset. Training set is used to train the model and test set is used to assess whether the model is good or not.

Through research and revealing the data set from figure 1, we found that the data has a very obvious long tail phenomenon. When we perform user based collaborative filtering, the long-tail phenomenon may mean two things. One is that users in the long tail are less likely to have same interests with each other, and second, the users in the long tail may be easier to show similarity interests with user outsides long tails. It's easy to understand that if a user's interests are broad enough and there are sufficient ratings, then his interests and ratings are more likely to match those users with only a few ratings. But is this enough to show that the two users have similar interests? In a situation where one of the users has broad interests? It is easy to see that there are actually some biases. So, we need a function to balance the dominance of users with many ratings over users with few ratings which we will talk about later.

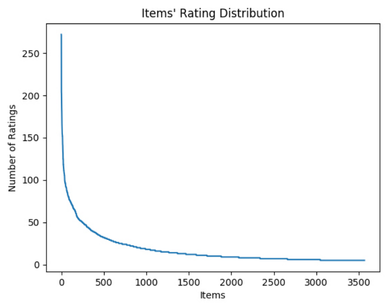
Conference'17, Washington, DC, USA
© 2016 ACM. 978-x-xxxx-xxxx-x/YY/MM...$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn



a



b



c

Figure 1: Data Reveal:(a) is the distribution of rating scores. Almost half of ratings are 5-score. (b) is the distribution of number of ratings from each user. There exists long tail phenomenon which indicates there may be a bias when we compare different users' interests. (c) is the distribution of number of ratings of different musics.

4 PROBLEM FORMALIZATION

We can formalize the problem as the following way.

$$U : set\ of\ users$$
$$M : set\ of\ music$$
$$Utility\ function\ u : X * I \longrightarrow R$$
$$R : set\ of\ ratings$$
$$A\ sample\ Utility\ Matrix\ can\ be\ presented\ as :$$

	user1	user2	user3
music1	1		5
music2		3	5
music3	4		

There are several steps to solve the problem:

- Create training and validation dataset by splitting the original dataset (the rating in the validation dataset are treated as unknown).
- Predict unknown ratings from the known ones - develop user- based CF model to conduct rating prediction.
- Evaluate predictions by calculating the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).
- Predict the ratings on all the items that user did not buy before, then rank the items in descending order of the predicted rating, and finally take the top 10 items as the recommendation list.
- Evaluate the quality of the recommendation list by calculating precision, recall, F-measure and NDCG.

5 THE PROPOSED MODEL

Our model is based on user-based collaborative filtering.

The main idea is to find set N of other users whose ratings are similar to x's ratings and create the recommendation list based on them.

Firstly, we weight the similarity between each two users. Here we combine Jaccard similarity measure and Euclidean distance to calculate the similarity.

$$d(x,y) = \sqrt{\sum (x_i - yI)^2}$$
$$W_{x,y} = \frac{|N(x) \bigcap N(y)|}{|N(x) \bigcup N(y)|}$$
$$sim(x,y) = d(x,y) * W_{x,y}$$

Next, we move forward to implement prediction for music i of user x. Let N be the set of K users most similar to x who have rated music i.

$$r_{xi} = \frac{\sum_{y \in N} sim(x,y) * r_{yi}}{\sum_{y \in N} sim(x,y)}$$

At this time, we could recommend music for each user.

Algorithm 1: User-based Collaborative Filtering

```
res = {}
for each $u \in users$ do
    Poss_music = set();
    rank = {};
    N = the set of top K similar users of u;
    for each $v \in N$ do
        v_bought = music that u has bought;
        for each $music \in v\_bought$ do
            if $u$ has not bought music then
                Poss_music.add(music);
            end
        end
    end
    for each $music \in Poss\_music$ do
        rate = predict(u.music);
        rank.setdefault(music,0);
        rank[music] = rate
    end
    res[u] = sorted(rank.items())[0:10]
end
```

6 EXPERIMENTS

6.1 Environment Setting

The original dataset contains 64,706 reviews given by 5,541 users. As we described, we randomly split each user's reviews into 80% part and 20% part as training set and testing set. We use testing set as our ground truth to evaluate and compare our prediction results. According to the predictions, we can make a music recommendation list for users. Also, we evaluate whether these recommendations are good or not and robust or not. We run these experiments using Python on macOS platform with 2.6 GHz Intel Core i5 processor and 8 GB 1600 MHz DDR3 SDRAM.

6.2 Metrics

To evaluate these results and compare them to the ground truth, we apply several metrics to calculate the error and how close the predictions are to the ground truth.

- Mean absolute error (MAE) is a measure of errors between paired observations expressing the same phenomenon.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Actual_i - Prediction_i|$$

- Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Actual_i - Prediction_i)^2}$$

Also, we will make a recommendation list according to these prediction results. So, we apply a few metrics to evaluate whether the recommendations are good or not and robust or not.

- Precision is the fraction of relevant instances among the retrieved instances.

$$Precision = \frac{prediction\_truth}{all\_prediction}$$

- Recall is the fraction of the total amount of relevant instances that were actually retrieved.

$$Recall = \frac{prediction\_truth}{all\_truth}$$

- F-Measure provides a way to combine both precision and recall into a single measure that captures both properties. Alone, neither precision or recall tells the whole story. We can have excellent precision with terrible recall, or alternately, terrible precision with excellent recall. F-measure provides a way to express both concerns with a single score.

$$F\_Measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

- Normalized Discounted Cumulative Gain (NDCG) provides normalized cumulative gain at each position for a chosen value across queries. This will allow us to compare a search engine's performance from one query to the next consistently.

$$NDCG[2] = \frac{DCG}{IDCG}$$

6.3 Results and evaluations

We predict the ratings in the testing set and compare the true ratings with the prediction. The result is shown below:

$$MAE = 0.7763$$
$$RMSE = 1.1331$$
$$Precision = 0.0512$$
$$Recall = 0.0315$$
$$F\_Measure = 0.0386$$
$$NDCG = 0.2143$$

What's more, we calculate the MAE and RMSE on different situation. We consider ignoring some user similarity if their common products are less a number. The first row of table 1 and table 2 means we ignore users' similarity with common products less than that number. And we also use two different similarity method to calculate the MAE and RMSE, we can see the results from figure 2 and figure 3.

	2	3	4	5	8	10	15
MAE	0.87913	0.83647	0.81088	0.82934	0.89411	0.9125	1.01712
RMSE	1.26365	1.20323	1.18373	1.19021	1.28525	1.31018	1.43527

Table 1: Euclidean & Jaccard

	2	3	4	5	8	10	15
MAE	0.83202	0.84555	0.8243	0.85623	0.90148	0.95971	1.04592
RMSE	1.17778	1.20411	1.17657	1.22239	1.28246	1.34904	1.47162

Table 2: Cosine

From the comparison, we can see the MAE and RMSE share the same trend at first. And then, we can see the number of MAE decrease at first and then increase, it will have best performance when threshold is 4, which means if users only have 2 or 3 common products, we would better ignore these users, and only consider users with more than 4 common products. Furthermore, if we only consider users with too many common users, it also cause the decrease of the performance.

7 CONCLUSIONS AND FUTURE WORK

In this project, we use user-based Collaborative Filtering to do the prediction. We calculate many criteria to assess whether the model is good or not. We think the most appropriate criterion is MAE and RMSE. Purchasing a certain digital music is a random event. One user can have a certain flavour, but he can't but all music under this genre. It is quite common that one user never buy music A even if he bought music B which is similar to A. There are often dozens of music similar to music B. The user may buy only one or two of them if he is not a music fanatics. That is why precision and recall are very small in the experiments. Making a recommendation list for each user and assessing each model based on it is not a good idea. There is too much randomness in it. MAE and RMSE is more appropriate and stable to use as a criterion.

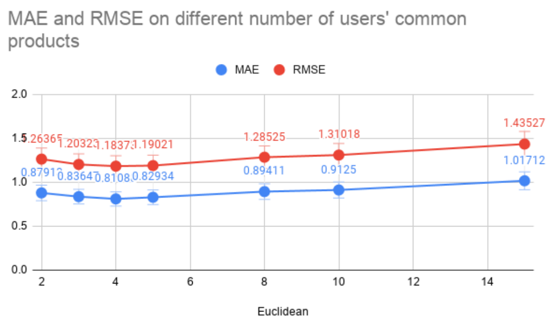


Figure 2

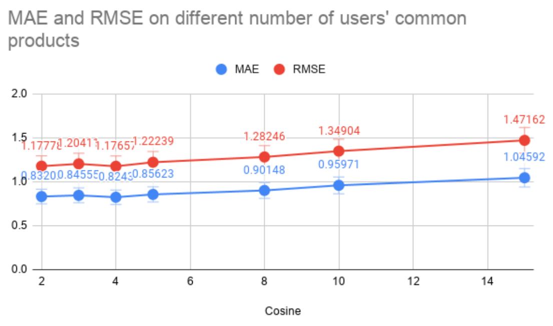


Figure 3

Here are some pros and cons for our model. Actually, it is easy to implement and content independent. However, the percentage of people who rate items is really low, new users will have no to little information about them to be compared with other users, and new items will lack of ratings to create a solid ranking. Furthermore, Better recommendation requires more users as well as more computing power and time.

We use several hours to train the model on our laptop. Due to the limitation of our computing resources, we can only choose small dataset. I think this is why our precision and recall are very small.

If we have access to high performance computing clusters in the future, we can use larger dataset(tens of million) and more complex model above to build our music recommendation system.

ACKNOWLEDGEMENT

Thanks to Bowei Feng and Cong Deng.

REFERENCES

[1] Zhi-Dan Zhao and Ming-Sheng Shang. User-based collaborative-filtering recommendation algorithms on hadoop. In *2010 third international conference on knowledge discovery and data mining*, pages 478–481. IEEE, 2010.

[2] Consistent Distinguishability. A theoretical analysis of normalized discounted cumulative gain (ndcg) ranking measures. 2013.