Technical Report of Shanda Innovations Team for Music Data Science Hackathon

Yunwen Chen, Zuotao Liu, Daqi Ji, Yingwei Xin, Lu Yao, Wenguang Wang

Shanda Innovation Institute, Shanghai, China
{kddchen, zuotao.liu, ducy2001, xinyingwei, luyao.2013, svd.wang}
@gmail.com

Abstract. This paper describes the solution of Shanda Innovations team to Music Data Science Hackathon. Factorization Machine and Linear Regression are employed to incorporate a great variety of features. User age, gender, working status, region and some other user profiles are utilized for modeling user interests. Users' descriptions to EMI artists are also integrated to improve the prediction accuracy. In addition, a simple ensemble method and postprocessing strategy are applied to combine 2 different predictors and produce the final results. The proposed approach obtained 13.19638 (public score)/13.24598(private score) on the testing dataset, which achieved the 1st place globally in the competition.

1 Introduction

Recommendation systems aim to predict users' interest to products, news, TV shows, music, etc, based on their historical profile of previous purchases, views and clicks. Locating users' interest is critical for many consumer-oriented websites, such as Amazon and Netflix. However, design and optimization proper algorithms are not an easy task, and for different practical applications, it need deep analysis of both valid domain-based data and effective models. Therefore, recommendation system draws increasing attentions from research and industry communities in the last few years.

Music Recommendation or prediction is a traditional recommendation task. Recently, KDD-Cup 2011 provides to access to yahoo music data set. Thanks to these widely publicized competitions and various academic papers from research areas, there have been many promising technologies for recommendation systems, among which, Collaborative filtering (CF) and Factorization Machine are most famous ones.

The task [1] of EMI Music Data Science Hackathon is to develop such a system to capture users' interests to new songs. The dataset [2] provided by EMI, is one of the largest music preference datasets in the world today that connects data about people–who they are, where they live, how they engage with music in their daily lives–with their opinions about EMI's artists. For users in the dataset, we

[1] http://www.kaggle.com/c/MusicHackathon
[2] http://musicdatascience.com/emi-million-interview-dataset/

need to design an algorithm to understand what it is about people and artists, according to users, (a) demographics, (b) artist and track ratings, (c) answers to questions about their preferences for music, and (d) words that they use to describe EMI artists, in order to predict how much they like tracks they have just heard. Root Mean Square Error (RMSE) is used to evaluate the results provided by 138 teams around the world.

Compared to traditional recommender problems, e.g. the Netflix Prize[1], the settings of Music Data Science Hackathon appears more complex. Firstly, there are much richer features between users and artists. In words.csv, some questions are answered by users to locate the different degrees of familiarity to EMI artists. Besides, a list of words from a given set are selected by users for best describing the artist after listening to tracks from a particular artist. Secondly, user profiles contain rich information, such as gender, age, region of UK, working status, listening habits which are investigated by answers to some questions. So models that are capable to integrate various features are required, and it is demanding to have an effective preprocessing of these features to cope with this challenge.

In this report, we describe our solution that combines Factorization Machine and Linear Regression. Various features are extracted from the training data and integrated into the proposed models. After preprocessing of various features, a stochastic gradient descent (SGD) based training framework [3] are employed to optimize the model. We built 2 models that pay particular emphasis on different features and are under different settings. We also used a linear combination to ensemble these 2 models and a postprocessing strategy to gain a higher accuracy.

The rest of the report is organized as follows. Section 2 presents the preprocessing method we used, and how to generate the validation sets. In Section 3, we will propose our models, which are adopted in the final solution. Ensemble and postprocessing methods are proposed in Section 4 and Section 5 respectively. We also give the github url of our codes for this competition.

2 Preprocessing

This section introduces the procedure of data preprocessing and validation sets generation.

2.1 Words

words.csv is about how users describe EMI artists.

Firstly, we merged the two "Good lyrics" columns.

The values of the "LIKE_ARTIST" column are mapped to new ones, divided by 100 simply. All the missing values are abandoned.

The answers of "HEARD_OF" and "OWN_ARTIST_MUSIC" columns are converted to discrete keys. The value of a user to a key is 1, only if the user gives the corresponding answer.

[3] http://apex.sjtu.edu.cn/apex_wiki/svdfeature/

For the list of 82 different words, we keep the values of positive scores, and abandon the rest.

Then, for each pair of users and artists, there is at most one value of [0, 1] to describe the degree of preference, and a filtrated list of words to describe user's opinion.

2.2 Users

users.csv gives data about the respondents themselves, including their attitude towards music.

For music habit questions, we keep all valid rates, while rescale them to [0, 1].

For the "AGE" column, we keep all the valid values.

For the rest columns, such as such as "GENDER", "WORKING", "REGION", "MUSIC","LIST_OWN", "LIST_BACK", we transform their values to discrete keys.

2.3 Validation Generation

We randomly split train.csv to training set and validation set with probabilities 0.75 and 0.25 respectively. Models are trained on training set and tested on validation set.

3 Models

In this section, we will first give an explanation that how our models worked.

Before formally introducing the models, we first define our notational conventions. We denote U as the user set and I as the track set, which will be replaced as item set in the rest. Then we have the number of $|U|$ users and $|I|$ items, where function $|\cdot|$ indicates the quantity of members in a set. Special indexing letters are reserved for distinguishing users from items: for user u and for item i. r_{ui} indicates the preference by user u for item i. The ratings are arranged in a $|U| * |I|$ matrix $R = r_{ui}$. We denote K as all user-item pairs in training data set.

3.1 Factorization Machine

Recently, latent factor models comprise an alternative approach to collaborative filtering with the more holistic goal to uncover latent features that explain the relation between users and items[2]. Because of its attractive performance, availability and scalability[3][4], Singular Value Decomposition(SVD) is one of the most popular models. In this paper, we denote $\hat{r}_{ui}$ as the prediction of user u's rate to item i. The baseline predictor of SVD with user bias and item bias is defined as:

$$\hat{r}_{ui} = b_{ui} + \boldsymbol{q}_i^T \boldsymbol{p}_u \tag{1}$$

where $\boldsymbol{p}_u$ is the d-dimensional user feature vector and $\boldsymbol{p}_u$ is the item feature vector. d is a parameter that is set beforehand. Bias factor b_{ui} is defined as:

$$b_{ui} = \mu + b_u + b_i \tag{2}$$

In this equation, parameter μ denotes the overall average rating. b_u and b_i indicate the deviations of user u and item i. These parameters can be learnt by solving the following regularized least squares problem:

$$\min_{\boldsymbol{p}_*,\boldsymbol{q}_*,b_*} \sum_{(u,i)\in K} \left(r_{ui} - b_{ui} - \boldsymbol{q}_i^T \boldsymbol{p}_u\right)^2 + \lambda_1 \left\|\boldsymbol{p}_u\right\|^2 + \lambda_2 \left\|\boldsymbol{q}_i\right\|^2 + \lambda_3 b_u^2 + \lambda_4 b_i^2 \tag{3}$$

The first term in the above equation strives to find b_u, b_i, $\boldsymbol{p}_u$, and $\boldsymbol{q}_i$ that fit the given ratings. The others are regularization terms with parameters λ_1, λ_2, λ_3 and λ_4 to avoid over-fitting.

3.2 Model Learning Approach

Stochastic Gradient Descent algorithm (SGD)[3] is a very effective approach for solving the optimization problems(Eq.3). It randomly loops through known ratings randomly in set K and takes a small gradient descent step on the relevant parameters along the direction that minimizes the error on each rating. Let us denote the prediction error as e_{ui}. The SGD updates the parameters based on the following equations:

$$\begin{aligned} b_u &\leftarrow b_u + \eta\left(e_{ui} - \lambda_1 b_u\right) \\ b_i &\leftarrow b_i + \eta\left(e_{ui} - \lambda_2 b_i\right) \\ \boldsymbol{q}_i &\leftarrow \boldsymbol{q}_i + \eta\left(e_{ui}\boldsymbol{p}_u - \lambda_3 \boldsymbol{q}_i\right) \\ \boldsymbol{p}_u &\leftarrow \boldsymbol{p}_u + \eta\left(e_{ui}\boldsymbol{p}_u - \lambda_4 \boldsymbol{p}_u\right) \end{aligned}$$

where parameter η indicates the learning rate, λ_1, λ_2, λ_3, λ_4 are parameters that define the strength of regularization. The complexity of each iteration is linear in the number of ratings.

3.3 Linear Regression

In statistics, linear regression is an approach to modeling the relationship between a scalar dependent variable y and one or more explanatory variables denoted X. In linear regression, data are modeled using linear predictor functions, and unknown model parameters are estimated from the data. Most commonly, linear regression refers to a model in which the conditional mean of y given the value of X is an affine function of X.

We employ linear regression to integrate user information and users' descriptions to artists. Predicted values of factorization machine and linear regression are linearly combined for better modeling.

$$\hat{r}_{ui} = b_{ui} + y_{ui} + \boldsymbol{q}_i^T \boldsymbol{p}_u \tag{4}$$

y_{ui} is defined in the following equation:

$$y_{ui} = \sum_k \beta_k * X_{ui}^k \tag{5}$$

$\{X_{ui}^k\}$ are values for each key of user u and item i in the converted users.csv and words.csv.

For discrete keys, X_{ui}^k will be 1 or 0, such as "GENDER", "WORKING", and the list of words in words.csv. For continuous keys, such as "LIKE_ARTIST" and music habit questions, X_{ui}^k will be converted values in range of [0, 1]. Especially, we map "AGE" x to a new value

$$age(u) = \begin{cases} 36, & \text{if } x >= 72 \\ floor(x/2), & \text{if } 0 \leq x < 72, \end{cases} \tag{6}$$

and X_{ui}^k corresponding to $age(u)$ will be set to 1.

The parameters β_i are learned by modifying the SGD update procedure (3.2). For each step, we add

$$\beta_k \leftarrow \beta_k + \eta \left(e_{ui} X_{ui}^k - \lambda_5 \beta_k \right), \text{ for all } \beta_k.$$

Where λ_5 is the regularization term.

3.4 Parameter of Models

Parameter settings of the stated model are shown in Table 1.

Table 1. Model settings

ID	parameters	values
1	d	250
2	η	0.0002
3	λ_1	0.004
4	λ_2	0.04
5	λ_3	0
6	λ_4	0
6	λ_5	0.004

3.5 Backup Model

We also train a backup model to prevent the result being trapped in locally optimum. The difference between the backup model and the formal one is as follows.

Firstly, the way to utilize users.csv is kind of different.

Secondly, the parameters for training are also different. Table 2 shows the settings.

Table 2. Settings for back model

ID	parameters	values
1	d	50
2	η	0.001
3	λ_1	0.005
4	λ_2	0.05
5	λ_3	0
6	λ_4	0
6	λ_5	0.0001

4 Ensemble

We simply average the prediction results of the 2 models stated before. For each row of test.csv, the predicted rating $\hat{r}_{ui}$ is $(\hat{r_1}_{ui} + \hat{r_2}_{ui})/2$, where $\hat{r_1}_{ui}$ and $\hat{r_2}_{ui}$ are results by the 2 models.

5 Postprocessing

The goal of this task is to predict how much a user would like a song. The rating should in range of [0, 100], so the rating predicted by our models should be reassigned to 0 or 100 if less than 0 or larger than 100 respectively.

For each user, we try to reduce the biases of predicted ratings against to the rating distributions of training data. For user u, we find the minimal rating in training data is s_u, and the maximal rating is l_u. Then a predicted rating $\hat{r}_{ui}$ will be converted to

$$\hat{r'}_{ui} = \begin{cases} \hat{r}_{ui} * 0.7 + s_u * 0.3, & \text{if } \hat{r}_{ui} < s_u - 5 \\ \hat{r}_{ui} * 0.7 + l_u * 0.3, & \text{if } \hat{r}_{ui} > l_u + 5 \\ \hat{r}_{ui}, & \text{otherwise.} \end{cases} \quad (7)$$

6 Codes

The source codes are now available on GitHub [4].

It's the refined version of actual codes we wrote during the competition. The procedures of preprocessing, training, testing and postprocessing can be started by several scripts. The results that we submitted can be reproduced pretty similarly.

7 Acknowledgment

We would like to acknowledge the organizers for holding this exciting and inspiring competition. We also thank the colleagues at Shanda Innovation Institute for fruitful discussions.

References

1. Bennet, J., Lanning, S.: The netflix prize. In: KDD Cup and Workshop. (2007)
2. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: Advances in Neural Information Processing Systems 20 (NIPS'07). (2008) 1257–1264
3. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. IEEE Computer **42** (2009) 30–37
4. Paterek, A.: Improving regularized singular value decomposition for collaborative filtering. In: Proc. KDD Cup and Workshop. (2007) 39–42

[4] https://github.com/fancyspeed/codes_of_innovations_for_emi/tree/master/code_of_innovations