Prediction of repeated listening by means of GBDT-based approach

WSDM – KKBox's Music Recommendation Challenge 2018

Vasiliy Rubtsov
National Research University Higher School of Economics
vrubcov@hse.ru

Dmitry I. Ignatov
National Research University Higher School of Economics
dignatov@hse.ru

Anvar Kurmukov
National Research University Higher School of Economics
kurmukovai@gmail.com

ABSTRACT

The proposed problem at WSDM Recommendation Challenge 2018 was to predict whether or not a user will listen a song repetitively after the first observable listening event. This paper presents a model which achieves ROC AUC 0.74688 (third place in the competition). The reproduced code solution is available at github[1].

ACM Reference Format:
Vasiliy Rubtsov, Dmitry I. Ignatov, and Anvar Kurmukov. 2018. **Prediction of repeated listening by means of GBDT-based approach**: WSDM – KKBox's Music Recommendation Challenge 2018. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The challenge was to build a music recommendation system using a dataset from KKBOX music streaming service. The training data set consists of information about the first observable listening event for each unique user-song pair within a specific time duration. Meta data of each unique user and song pair is also provided. The train and the test data are selected from users listening history in a given time period.

The paper is organized as follows: in section 2, we describe the challenge task and the data provided by organizers along with the evaluation metric used; in section 3, we describe validation methodology, feature engineering process preparation, and introduce the proposed recommender model along with its components. Section 4 concludes and discusses additional possibilities to fine-tune the model and some peculiarities of the gained competition experience.

2 DATA

2.1 Data description

The provided data contains information about the first listening of a musical track by a particular user. In addition to user's and track's indices, the contextual information is provided: the name of the tab where the event was triggered, the name of the layout a user sees, and the entry point a user first plays music on the mobile application.

Moreover, user profile includes the following information: city, age, gender, registration method, registration date, and expiration date. The track's related data are provided as well: song length, genre, artist name, composer, lyricist, and language.

[1]https://github.com/VasiliyRubtsov/wsdm_music_recommendations

Conference'17, July 2017, Washington, DC, USA
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2.2 Competition details

The competition problem was formulated as follows: we need to predict the chances of a user listening to a song repetitively after the first observable listening event within a time window.

The evaluation metric was ROC AUC.

3 METHODS

In this section, we present our approach to the challenge in details. We describe the validation scheme used in our experiments, and talk about the features that we extracted from the challenge dataset as well as the models built on these features.

All the experiments were conducted on Debian machine with Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz (28 cores, 56 threads) and 256 GB of RAM memory. Our method was implemented in Python.

3.1 Training procedure

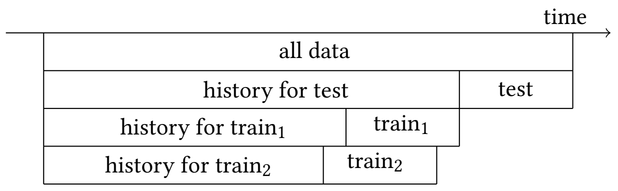

Figure 1: Training procedure.

The whole training procedure is shown in Fig. 1. This scheme relies on three important ideas. First, we found out that observations are sorted in historical order (thus the row index is a time-related feature), we make use of this property and split our data based on the row index. Second, we split a training set into three parts: *history*, $train_i$, and *validation* (around 20% of all labeled data), we extract features for $train_i$ from *history* part (this will be described later in section 3.2), next we train a model on $train_i$ and choose its parameters using *validation* set. The final idea is data augmentation: extracting features from historical data left us with a few observations for the *train* set, thus we use $train_1$ and $train_2$. In order to avoid overfitting, we do not combine them but train models independently on both sets and blend their predictions after (see Section 3.4 on the recommender model for details).

For final training we take train set of the same size as test set, and the remaining labeled data were used as history (to generate features for training).

3.2 Feature engineering

Since almost all the features are categorical, the main idea here lies in grouping by those features, their pairs, and triples followed by some aggregation function.

Supervised features. Since these features are calculated with the presence of the target variable, we use only historical information.

- Mean. We group data by categorical features, their pairs, and triples and then calculate the average by the target value within the groups.
- Linear regression. We perform linear regression by time for different groups: user, user-artist, user-genre, user-source_type, user-composer, user-language, and song_id. For example, for user i - genre j group (all observations which contain both user i and genre j) we perform one dimensional linear regression with time as X and original target variable as y. This allows us to catch simple intra group trends.
- Matrix factorization. We also use LightFM [3] implementation of matrix factorization with logistic loss and consider a particular user within different contexts as separate users, but use user id as a feature.

$$r_{uic} = b_{uc} + b_u + b_i + \langle p_{uc} + p_u, p_i \rangle, \text{ where}$$

  r_{uic} is the rating given by the user u in the context c for the item i, $b \in \mathbb{R}$ is the bias, and $p_{uc}, p_u, p_i \in \mathbb{R}^n$ are the latent vectors for the user-context, the user and the item, respectively.

Unsupervised features. This group of features does not use the target variable to compute, thus we can use the whole data set.

- Count. This type of feature contains the number of occurrences for the categorical features, the co-occurrences of their pairs and triples.
- Time from the previous listening answers how much *time* (rows) has passed since the last listening event for a given category (pair or triple).
- Time before the next listening is the *time* before next listening event inside a category (pair of categories or their triples).
- Count from future shows how many times category/pair/triplet occurred before the end of the whole time period of the data (history).
- Count from past shows how many times category occurred since the end of a *history*.
- Last time difference is the *time* to the last listening event within a category.
- Share of unique songs is the fraction of songs of a given artist listened by a particular user.
- Time from test period is the *time* passed since the end of the available *history*.
- Part of a song listened is the song length divided by the *time* to the next listening event.
- Song length.
- User age.
- Registration date.
- Expiration date.

In total 1350 features were generated. Table 1 shows the top 20 features according to the XGBoost [1] feature importance with the gain importance type, where model is fit on train$_1$ with matrix factorization feature.

Table 1: Top 20 features

Feature	Gain
Matrix factorization	4116.54
Mean (genre_ids, source_screen_name, source_type)	1897.92
Last time difference (artist_name, language, msno)	1502.82
Last time difference (artist_name, genre_ids, msno)	1187.89
Mean (genre_ids, lyricist)	1071.63
Time to the next listening (msno, source_type)	636.76
Mean (source_system_tab, source_type)	588.12
Mean (gender, song_id)	581.72
Mean (city, song_id)	572.86
Mean (artist_name, composer, lyricist)	570.04
Mean (genre_ids, language, source_screen_name)	512.88
Count from future (registered_via, song_id)	495.69
Time to the next listening (msno, source_system_tab)	487.80
Count from future (song_id, source_type)	462.39
Mean (gender, registered_via, song_id)	458.42
Count from future (song_id)	454.10
Time from the previous listening (msno)	412.73
Mean (gender, source_screen_name, source_type)	381.84
Mean (language, msno, source_screen_name)	380.28
Mean (registered_via, source_system_tab)	361.06

3.3 Recommendation model

We train Catboost [2] and XGBoost on train1 and train2 datasets with and without matrix factorization features. This training results in 8 models, which are blended [4] then as follows:

$$\alpha_1 p_1 + \alpha_2 p_2 + \ldots + \alpha_8 p_8, \text{ where}$$

p_i is the probability of listening returned by i-th model. The coefficients are fit based on validation sets.

In Table 2 the performance of each model is shown on a private leaderboard.

The model blending result in 0.74688 of ROC AUC.

4 DISCUSSION

The main task of this competition was to predict repeated song listening by users. It was assumed that model trained for this purpose should be used to recommend a track for a user that will be listened again. However, it seems that listeners use recommender service of streaming audio to find new music compositions that are similar to those they listened to before; that is they want to discover something new. However, the model does not fit that purpose.

The quality metric was ROC AUC, which is used for classification. However, usually a recommender should be able to rank items for

Table 2: Single model's score

Model	Train set	Matrix factorization feature	Score
XGBoost	1	Yes	0.74421
XGBoost	1	No	0.74377
XGBoost	2	Yes	0.74380
XGBoost	2	No	0.74331
Catboost	1	Yes	0.74496
Catboost	1	No	0.74441
Catboost	2	Yes	0.74451
Catboost	2	No	0.74384

a target user; thus, the used metric may not be suitable for such a system. As an alternative, one can use ROC AUC that is averaged by users.

It is worth noting that in this competition there were no strong shake-up between the public and the private leaderboards and a good correlation between the validation set and the leaderboard was present; that effects allowed us to test all the hypotheses on the validation set.

REFERENCES

[1] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 785–794. https://doi.org/10.1145/2939672.2939785

[2] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. 2017. CatBoost: gradient boosting with categorical features support. In *NIPS Workshop on ML Systems*.

[3] Maciej Kula. 2015. Metadata Embeddings for User and Item Cold-start Recommendations. In *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015. (CEUR Workshop Proceedings)*, Toine Bogers and Marijn Koolen (Eds.), Vol. 1448. CEUR-WS.org, 14–21. http://ceur-ws.org/Vol-1448/paper4.pdf

[4] Andreas Töscher, Michael Jahrer, and Robert M. Bell. 2009. The BigChaos Solution to the Netflix Grand Prize.