An Effective Classification method with RNN and Grand Boosting for Retention Rate of Baidu Hao Kan APP Challenge

UltimateFUN

Bo Wang
student/Postgraduate
Xidian University of Electronic Technology
Shanxi, China
000wangbo@163.com

Weilong Chen
student/Postgraduate
University of Electronic Science and Technology
Sichuan, China
837461957@qq.com

Ailin Sheng
student/Postgraduate
Sichuan University
Sichuan, China
2693757672@qq.com

ABSTRACT

WSDM Cup's Retention Rate Prediction Challenge refers to a binary classification task to predict whether a user will reuse the app after their first day's Experience of Baidu Hao Kan APP. On one hand, some new users may download the app to browse and play the video for hours. Some new users can use the app again to watch videos the next day (call as retained users); on the other hand, some would no longer use the app after they download it for a while. In this paper, we designed a practical machine learning ways to tackle such a challenge, including feature engineering, Decision Trees like Lightgbm [1], CatBoost[2], RNN's ManyToMany, and stacking of learning models. We want to find the methods to increase the percentage of user retention and the elements which affect user retention. In the competition, we eventually received the second place with an evaluation score of 0.7671.

KEYWORDS

Feature engineering, Lightgbm, CatBoost, RNN, ManyToMany, SWA

1 INTRODUCTION

Baidu Hao Kan App is an aggregation platform that provides users with massive high-quality short video content. It provides a full coverage of high-quality videos such as fun, music, film, entertainment, games, life, and essays. Baidu Hao Kan App uses intelligent algorithms to understand users' interests and preferences and to recommend tailored video content to users. In the process of rapid growth, Baidu Hao Kan App faces new challenges. New users may download the app to browse and play the video for a while. Some new users will use the app again to watch the video the next day (we call them retained users); however, others no longer use the app. We want to know how to increase the percentage of user retention and the reasons that affect user retention.

Based on user's portrait data (gender, age, education, geography, interests, etc.), user viewing behavior, usage time of the day, installation source, etc., the task is to estimate whether the new user will use Baidu Hao Kan App the next day.

For this heterogeneous data, the Grand Boosting model will perform very well, and support for category data is especially important here. Therefore, we chose LightGBM and CatBoost, which can support the category features, as our Grand Boosting's basic model, which can test the validity of our features and the difference of results.

Each input of the RNN ManyToMany structure corresponds to whether user is active after the output, making full use of the supervision information, reducing the gradient to the burden, making training easier. Compared to Lightgbm and CatBoost, with historical statistic being a feature, RNN does not need to deal with the input sequence too much, which can directly input various behavior sequences. With cosine-annealed snapshot integration, a large number of different local optimums can be obtained at very low cost.

The training set contains a large number of user and video interaction information, including likes, reposts, comments, etc., and some of the videos that are not displayed to the user and recommended in the background. We use the fusion of the RNN ManyToMany model and the Grand Boosting model to enhance the robustness of the model.

2 PROBLEM DEFINITION

WSDM Cup's Retention Rate Prediction Challenge refers to a binary classification task to predict whether a user will reuse the app after their first day's Experience of Baidu Hao Kan APP. The usable raw information includes user's portrait data, video's portrait data, and interaction information data. The setting of the challenge is using the first day interaction information as training data and predicting whether users reuse tomorrow.

For justifying the performance of the prediction, we use Log Loss as the evaluation metric, which is defined as

$$\text{logloss} = -\frac{1}{N}\sum_{i=1}^{N}(y_i log(p_i) + (1-y_i) log(1-p_i))$$

where N is the number of users, log is the natural logarithm, y_i is the binary target, and p_i is the predicted probability that y_i equals 1.

3 APPROACH

With the huge amount of data, we have to change the way, or reduce the scale of the data into training our model.

3.1 Preprocessing Data

The region code and the comment are all the same value in the data, so the two columns can be removed. The time when the user last played the video is s. The time point of the last video play plus the time point of the video playback time is t. If s and t are not on the same day, the label will be 1, and all such users in the training set are removed.

Since gender, age, education level, installation channel is not unique in the same users, these features are selected to use the most frequently of one person, and use the LabelEncoder as the final feature.

3.2 Features Engineering

Since the raw data gives us a collection of three datasets, our characteristics can be derived from three dimensions. Through any combination of these three dimensions, we can create hundreds of features.

The number of the user action, including the number of actions after removing displaying and clicking missing values, number of videos viewed by users and number of user following etc. making the feature of user action.

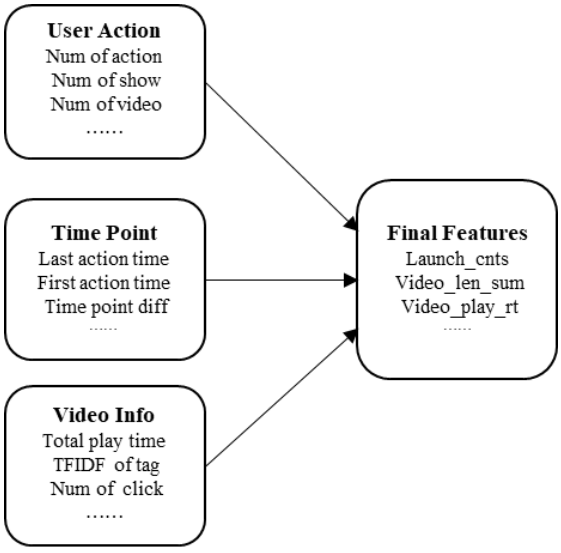

Figure 1: The feature build with three data dimensions.

Since the training users are all new users and their time is limited to one day, the time sliding window feature will lose its statistical significance. But the time point features, like on the last time or the first time what does the user do, mean a lot. We can do some timing operations on these points in time, like time difference, the average of time difference, and so on.

Also, the features of the video are also taken in our consideration. We do a lot of video-based statistical features, such as the total duration of the play, the number of clicks, the sum of the tag's TFIDF values, etc.

With some brain storm, we create some uncommon features. The time decay features magnify the effect of time on user operations. Since there is only one day of action records, the user's operation at certain times will become especially important. We can aggregate multiple dimensions for nesting and then stack. We can choose to aggregate on one of the other dimensions and then aggregate at the user level. For example, we aggregate the characteristics of the video type, and then aggregate them into a series of aggregation features such as click, like, and forward, and then merge to the user level and then perform aggregation operations at the user level. We call it as Multi-layer stacking feature.

Through our model selection and sorting, we get the top five features: 1) the time when the user last opened the app; 2) the difference between the time when the user first opened the app and the last time; 3) install ways; 4) the time when the user first opened the app; 5) the total duration of the user. It shows that the idea of our feature engineering is correct.

3.3 Learning Algorithms

3.3.1 Grand Boosting（LightGBM,CatBoost）

LightGBM and CatBoost are two Grand Boosting frameworks and are decision tree-based learning algorithms. CatBoost uses a symmetric tree. XGboost[3] builds nodes layer by layer, lightGBM builds nodes one by one, and CatBoost always uses a full binary tree. Its nodes are mirrored. CatBoost calls symmetry trees useful for avoiding overfit, increasing reliability, and greatly speeding up predictions. Lightgbm, on the other hand, is based on the Histogram-based decision tree algorithm, the leaf-wise leaf growth strategy with depth limitation, and the use of histograms for differential acceleration. Therefore, combining these two GB methods can achieve very good results.

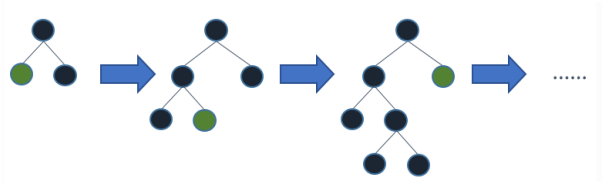

Figure 2: LightGBM's leaf growth strategy

We train based on a large number of historical data features, and through the parameters to control the over-fitting strategy, so that the improvement of the effect can be more stable. We also used two frameworks for training and got the following effects.

We can see that under this data set, the effect of CatBoost is better than that of LightGBM, and the features we construct are also better reflected in the specific scores.

Table 1: The score of Grand Boosting make

Features Enginering	Learning Algorithms	Log Loss on LB
base features	LightGBM	0.7459
base features	CatBoost	0.7463
with Multi-layer stacking feature	CatBoost	0.7612
with Time-Decay feature	CatBoost	0.7634
All feature	Hyper Parameters Optimized CatBoost	0.7658

3.3.2 RNN（many-to-many）

A solution used RNN is generally conceived. The user behavior sequence is input, and the sequence is labeled with whether the user is active in the next day. This is a solution called ManyToOne.

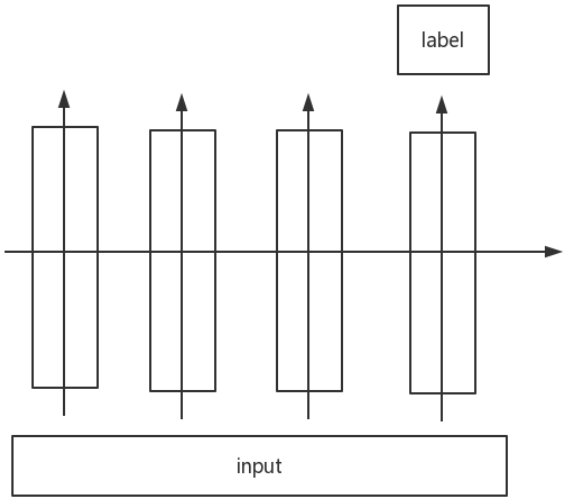

Figure 3: ManyToOne

In order to make full use of the data, a large number of sliding windows are needed for the training data to achieve data augmentation, but it costs high computational. In addition, each sequence has only one label, and the gradient is difficult to transmit, leading to the difficulty in training. On the contrary, we can consider the ManyToMany structure, that is, each input is corresponding to an output whether the user is active the next day. It will make full use of the supervision information, reduce the burden of transmitting gradient, and make the training easier.

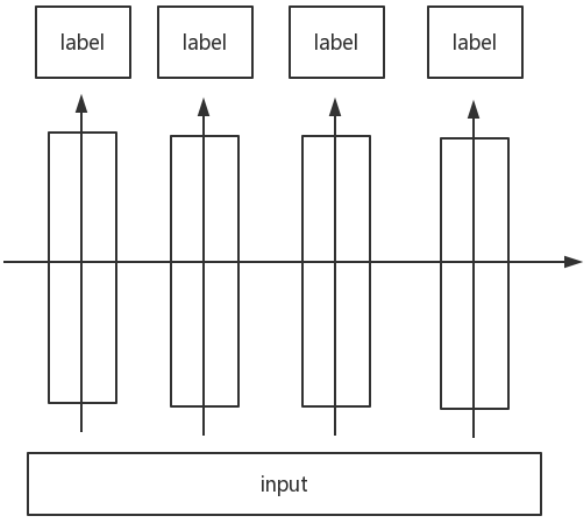

Figure 4: ManyToMany

Compared to the solution of Lightgbm, CatBoost used the historical statistic as a feature. RNN does not need to deal with the input sequence too much, and can directly input various behavior sequences. The simple list is as follows:

- Whether to log in on the day (0/1)
- The times (add one to take log) of watching videos
- The times (add one to take log) of action type (whether to display, whether to click, whether to favorite, etc.)

In addition, make an Intercept stitching directly on the output layer, and enter the date and installation channel. Low frequency categories can be grouped into one category. With cosine annealing and snapshot ensemble (see 5.1), a large number of different local optimal solutions can be obtained at very low cost, and finally fusion can achieve significant improvements.

3.3.3 Stochastic Weight Averaging（SWA）

3.3.3.1 Snapshot Ensemble

At the beginning of the training, SGD will make a big jump in the weight space. Subsequently, due to the cosine annealing strategy, the learning rate is gradually reduced. SGD will converge to a local optimal solution, and the model is added to the set by snapshot ensemble to achieve the fusion of the model. Then, the learning rate will be reset to a larger value, and the SGD will jump significantly again before the model converges to some different local optimal solutions.

The snapshot ensemble method has a period of 20 to 40 iterations. Long-term cyclic learning rates can find models that are as different as possible in the weight space. If the models are similar, the predictions of the individual networks in the fusion model will

be too close, which will make the advantages of the integrated model insignificant.

The snapshot ensemble works very well, which greatly improves the performance of the model, but the Fast Geometric Ensemble method works better.

3.3.3.2 Stochastic Weight Averaging（SWA）

The random weighted average (SWA) is very close to the FGE method, but its computational losses are small. SWA can be applied to any model structure and dataset and shows good results in these datasets. The SWA will tend to a global minimum. SWA is not an integrated approach that we understand in the traditional sense. At the end of the training, you will get a model that will perform better than snapshot Ensemble and FGE.

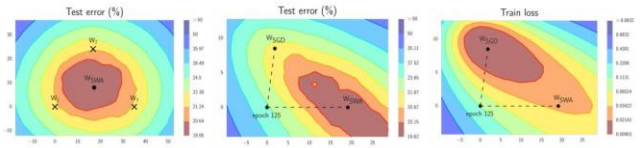

Figure 5: SWA

As shown in the figure above, W1, W2 and W3 represent three independently trained networks, and w_{SWA} is their average. Compared to SGD, w_{SWA} showed better performance on the test set.

The perspective of SWA is from empirical observation that the local minima at the end of each learning rate period tends to accumulate at the boundary of the region where the loss value is low on the loss plane. By averaging the loss values of such points, a global optimal solution with lower loss values, generalization, and versatility can be obtained.

Only two separate models are needed, without the need to integrate with many models. The principle of operation is as follows:

The first model is used to store the average of the model weights (such as w_{SWA} in the formula). This will get the final model after the training and use it for prediction.

The second model is used to cross the weight space (such as w in the formula) and explore using the cyclic learning rate.

$$w_{SWA} \leftarrow \frac{w_{SWA} \cdot n_{models} + w}{n_{models} + 1}$$

At the end of each learning rate period, the current weight of the second model will be used, and the average weight of the model will be updated by weighting the average between the old average weight and the new weight set of the second model. In this way, you only need to train one model and store both models in memory during training. In the forecasting stage, only the model with the average weight is needed and predicted, which is much faster than using the above-mentioned integrated methods that require multiple models for prediction.

3.4 EVALUATION RESULTS

The proposed two-stage learning process contains a few technique components, such as feature engineering and parameter optimization. We applied different settings to obtain the final results. We used the best score CatBoost and the two RNN model to ensemble, and got the highest score. Our strategy for ensemble is stacking and averaging. We used the RNN ManyToOne's result (stacking) as a feature to train the CatBoost, and applied the average of CatBoost's result and RNN ManyToMany's result. We submitted the offline best score and got 0.7671 at the leaderboard online.

Table 2: Final Score of ensemble

Features Engineering	Learning Algorithms	Log Loss on LB
All feature	Hyper Parameters Optimized CatBoost	0.7658
All feature	RNN ManyToMany	0.7632
All feature	2 model ensemble(CatBoost, RNN ManyToMany)	0.7668
All feature	3 model ensemble (CatBoost, RNN ManyToOne, RNN ManyToMany)	0.7671

4 CONCLUSION

In this paper, we introduced a practical machine learning pipeline for the Retention Rate of Baidu Hao Kan APP Challenge of the WSDM Cup 2019. Our final model is ranked the second place on the leaderboard. The most important thing is the understanding of the data. In the feature engineering part, we enrich the feature extraction by employing various types of data segmentation, and use Multi-layer stacking feature and time decay feature to enhance our model. Finally, in the model learning stage, we use three ways to build our final score, and it surely perform very well on both offline leaderboard and online leaderboard. Our further work includes to use CNN and DeepFFM to enhance our model performance.

REFERENCE

[1] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In Advances in Neural Information Processing Systems, pages 3149–3157, 2017.

[2] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. and Gulin, A. (2019). CatBoost: unbiased boosting with categorical features.Papers.nips.cc. Available at: http://papers.nips.cc/paper/7898-catboost-unbiased-boosting-with-categorical-features

[3] 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. ACM, 785–794.