UNCOVER THE FACTORS TO HELP MEASURE HOW YOUNG CHILDREN LEARN

Yu, Shenglan

December 14, 2019

1 Introduction

Engagement with high-quality educational media and games is an emerging approach for young children learning. However, the relationship between the media and learning outcomes is still somewhat unclear. It is of great interest to discover the factors influencing the learning processes. In the project, we aim to use anonymous gameplay data, including knowledge of videos watched and games played, from a game-based learning app, to predict the performance on in-game assessments. In the app, children navigate a map and complete various levels, which may be activities, video clips, games, or assessments. Each assessment is designed to test a child's comprehension of a certain set of measurement-related skills.

The task belongs to the supervised multi-class classification family. The input to our model is gameplay statistics of the player until the moment of the assessment. We then use a Gradient boosting based regression and rounding mechanism to output a predicted assessment class from totally four categories.

Link to the code: https://drive.google.com/open?id=12Gum62-1SBUsiaHOvehXnrrUC69p9a9j

2 Related Work

Educational learning aims to improve education quality by mining educational data. There are attempts to predict student score of distant learning using regression based method [Kotsiantis2012]. Here our goal is similar: predicting young children's performance in assessment. Educational app appears in just few years. Since this project focus on a quite new practical problem, we have not found a lot of similar tasks on mining educational app data.

Boosting algorithm [Friedman2002] trains a couple of weak learners which gradually focus more on samples which are hard to predict. It has been the wining solution of many kaggle competitions [SRK2019]. Our final system also uses boosting algorithms.

3 Dataset and Features

3.1 Dataset

The dataset comes from 2019 Kaggle Data Science Bowl [Kaggle2019], consisting of 11341042 training samples and 1156414 test samples. Each raw training/test sample describes one gameplay event information, which records the installation id, game session id, timestamp, game type, event id and the specific gameplay data of the event. Some important facts of the raw data:

Installation Id The installation id is an unique identification of an installation on a mobile device. It can be roughly considered as player id, despite the fact that some players may share one device.

Game Type One of the four types: Clip, Assessment, Activities and Game. Assessment event are the ones we want to predict the performance. Basically young children will use the app, play its clip, activities and games, and participate the assessments.

Event Id and Event Data Event id is the unique identification of the game event. There totally 386 different types of events. The events come from the log of app playing.

For example, the event: *"This event occurs when the player clicks on something that is not covered elsewhere. It can be useful in determining if there are attractive distractions (things the player think should do something, but don't) in the game, or diagnosing players who are having mechanical difficulties (near misses)."*

Thus, events give a quite comprehensive description of player's behavior in one game session. The event gameplay data then records the specific attributes of the event. For the example above, the event data records values like event occurring count, game time, coordinates of the mouse click and etc.

The event data is informative but difficult to use since different event has different attribute set. Tacking the semi-structured data is one of the challenges in the feature engineering.

The performance on in-game assessments are grouped into four categories, which are our target classes of classification:

3: the assessment was solved on the first attempt.

2: the assessment was solved on the second attempt.

1: the assessment was solved after 3 or more attempts.

0: the assessment was never solved.

3.2 Feature Engineering

The goal is to predict the player's performance in assessments. There are totally 1000 assessment sessions needed for prediction in the test set. All other samples including clips, games, activities and some assessments are all used to provide information of the player in the assessments for prediction.

The first step we do is to aggregate samples by the installation id. Thus, we can know what the players have done in previous game sessions. And we need that information for predicting the player's performance in the upcoming assessments.

Then, based on the complete game play list of each player(installation id), we extract features for making prediction. Since the raw data has timestamp, we can not include the game sessions which happen after the assessment sessions for prediction. Like stock price prediction, we can not use the future data to predict past performance.

Based on the above principle, we engineer two groups of features(totally 912 features):

Assessment Features This group of feature describes the information of assessment for prediction. There are totally five types of assessment: Bird Measurer, Cart Balancer, Cauldron Filler, Chest Sorter and Mushroom Sorter. Different types of assessment aim to assess different aspects of the player's STEM concept, like width and weight. Using the training dataset, we find that some assessments are inherently more difficult than others. We record the assessment type, the time spent in the assessment and each events count occurring in the assessment. Also, we extract the assessment event data like the coordinates of mouse click.

History Features Here, we make the connection between player's playing history and the assessment performance. Intuitively, the past performance on assessments and the total time spent on the app are strong indicators of the future performance. We also extract the each event counts in the history.

4 Methods

Baseline I use a simple heuristic as the baseline: make prediction purely based on history performance. Concretely, I take the median of history assessment performance as prediction.

Best Performance System My first observation is that the task is similar to a regression problem, despite its multi-classification nature. Since prediction classes are based on the number of attempts used to pass an assessment and the classification boundaries are continuous integers, the classes 0, 1, 2, 3 has a natural ordering implicitly. The higher classes number, the less attempts used. Thus, it makes sense if we use regression and rounding instead of classification to make prediction. The benefit of using regression in this task is that we can explicitly find the optimized rounding boundaries, making the classification more robust.

We use tree-based boosting algorithm in our model. Boosting algorithm trains a couple of weak learners which gradually focus more on samples which are hard to predict. Thus, subsequent trees help us to classify observations

that are not well classified by the previous trees. Gradient boosting algorithm archives the effect by using the gradients in the loss function. One huge advantage for gradient boosting is that we can specify any loss functions for our training purpose. We implement the algorithm with LightGBM library [Ke et al.2017].

The subsequent step after obtaining real values using regression is rounding the outputs into target classes. The rounding boundary is a three-element tuple (a_0, a_1, a_2). Value smaller than a_0 will be rounded into class 0; value between a_0 and a_1 will be rounded into class 1, and so on. I obtain the optimal rounding boundaries by using Nelder-Mead method implemented in *scipy* package which uses simplex algorithm [Nelder and Mead1965], to maximize the quadratic-weighted kappa over the training set. We will describe the quadratic-weighted kappa metric in subsequent section.

Overall, the system first obtain real-value prediction of samples and then round the prediction using optimised rounding boundaries. This model is hugely inspired by one Kaggle discussion [Lukyanenko2019]. Based on their work, we do more feature engineering and alter the rounding mechanism.

Other Learners We also experiment SVM, random forest and boosting classification for the task. SVM is a max-margin classifier. We try both rbf kernel and linear kernel in the experiment with "One-Versus-All" approach. Random forest is another tree-based algorithm which uses majority voting of a couple of weak learners. Each weak learner uses a subset of training samples and features. We implement the models above using scikit-learn library [Pedregosa et al.2011]. We have not tried neural networks since our training set after aggregation is not large enough for neural networks.

5 Experiments and Results

5.1 Evaluation Matrices

Predictions are scored based on the quadratic weighted kappa, which measures the agreement between actual output and expected output.

The quadratic weighted kappa for the task is calculated as follows [Kaggle2019]. First, an $N \times N$ histogram matrix O is constructed, such that $O_{i,j}$ corresponds to the number of installation ids i (actual) that received a predicted value j. An $N \times N$ matrix of weights, w, is calculated based on the difference between actual and predicted values:

$$w_{i,j} = \frac{(i-j)^2}{(N-1)^2}$$

An $N \times N$ histogram matrix of expected outcomes, E, is calculated assuming that there is no correlation between values. This is calculated as the outer product between the actual histogram vector of outcomes and the predicted histogram vector, normalized such that E and O have the same sum.

From these three matrices, the quadratic weighted kappa is calculated as:

$$\kappa = 1 - \frac{\sum_{i,j} w_{i,j} O_{i,j}}{\sum_{i,j} w_{i,j} E_{i,j}}$$

5.2 Hyper-parameters

We use a 5-fold cross-validation for tuning hyper-parameters. The used parameters of the boost regression are: learning rate: 0.05, number of estimator: 2000, loss function: rmse.

We use early stopping to prevent overfiting. Early stopping stops the training procedure once the performance on test set stops improving by a number of iterations.

5.3 Results

Approach	Kappa
Median voting	0.39
Random forest	0.42
Gradient boost classification	0.46
SVM rbf kernel	0.23
SVM linear kernel	0.12
Gradient boost regression + rounding	0.53

Table 1: Results

The baseline median voting approach archives 0.39 kappa score, inferring a strong correlation between history and future assessment performance. SVM models perform badly. The reason may be the applied kernel not matching the task. Boosting algorithm boost the performance to near 0.5. Using regression and rounding trick improves the performance further.

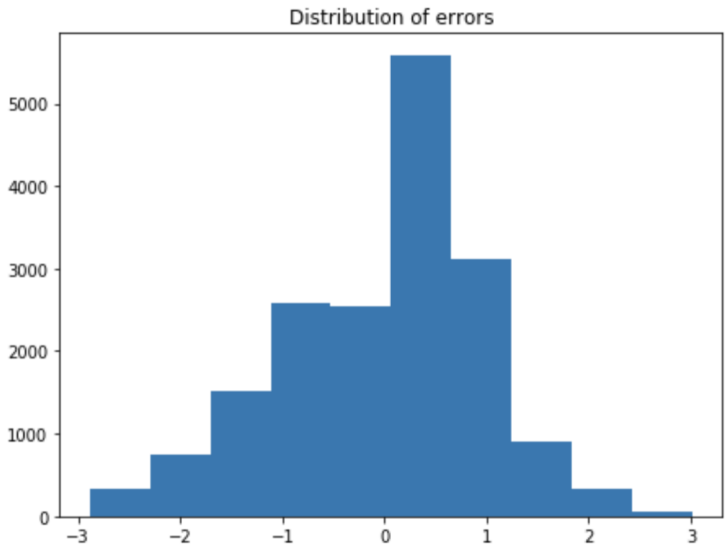

Figure 1: error distribution

We plot the error distribution which equals to prediction of training sample minus true label. From the figure, we find that most of the error has absolute value 1. So the most common errors are classifying one class 0 sample into class 1, one class 1 sample into class 2, and so on. It is unusual for our model to predict class 2 for a class 0 sample. That means it is hard to distinguish the number of attempts precisely. But our model can tell whether one player will use a large number of attempts or just few attempts quite accurately.

With ablation analysis, we find that the assessment type and player game total duration have the highest impact on performance, which meets our intuition. Also, some events which describe a distraction situation are strong indicators of large number of attempts. For example, children click some background object which they think should work but indeed not.

Our model gets around 0.73 kappa score on training set, which still is not high enough. This implies that our model suffers from high bias problem. The main reason is that still a large number of game play event data such as the cloud size and the dinosaur color are not used in our model. These data sounds silly, but contains important information for how children interact with the application. For example, some children may be more likely to click a colorful dinosaur than a black one. But these data are hard to deal with properly since different event has different feature set.

6 Conclusion and Future Works

In this project, we implement a system that takes game play data and output prediction on assessment performance. The system can be used to analyze the important factors contributing to young children's learning in education mobile apps. Our best performing system first outputs a real value and then uses optimized rounding mechanism to get prediction class. It outperforms other models by using the power of boosting algorithm which adapts to solve hard samples, and by using explicitly found optimized discrete boundary.

Given time allowed, we wish to exploit more event data to improve the high bias problem. Treating different assessment type separately and engineering independent feature sets may deserve a try.

7 Contributions

As an individual team, I do all the stuffs and take all responsibilities.

References

[Friedman2002] Jerome H. Friedman. 2002. Stochastic gradient boosting. *Computational Statistics Data Analysis*, 38(4):367 – 378. Nonlinear Methods and Data Mining.

[Kaggle2019] Kaggle. 2019. 2019 Data Science Bowl. [Online; accessed 30-October-2019].

[Ke et al.2017] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc.

[Kotsiantis2012] S. B. Kotsiantis. 2012. Use of machine learning techniques for educational proposes: a decision support system for forecasting students' grades. *Artificial Intelligence Review*, 37(4):331–344, Apr.

[Lukyanenko2019] Andrew Lukyanenko. 2019. Quick and dirty regression. [Online; accessed 30-October-2019].

[Nelder and Mead1965] John A. Nelder and Ronald Mead. 1965. A simplex method for function minimization. *Comput. J.*, 7:308–313.

[Pedregosa et al.2011] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[SRK2019] SRK. 2019. Winning solutions of kaggle competitions. [Online; accessed 30-October-2019].