Final Report: Feature Engineering and Machine Learning On Predicting Students' Mathematical Performance

Ge Zhang and Peihao Wang

[1] School of Information Science and Technology, ShanghaiTech University
zhangge3@shanghaitech.edu.cn
[2] School of Information Science and Technology, ShanghaiTech University
wangph@shanghaitech.edu.cn

1 Introduction

This project asks us to predict student performance on mathematical problems from logs of student interaction with Intelligent Tutoring Systems. While doing this project, we referenced some Ideas from KDD Cup 2010, especially the paper "Feature Engineering and Classifier Ensemble for KDD Cup" from National Taiwan University. Their methods has inspired us a lot.

2 Data set Separation

Since the data set is already separated into train data and test data, we don't need to separate them for training and testing. However, the data is very large in size, and the feature engineering needs complex operations. so we took the first 10 percent of the train data for experiment. Such separation has saved us many time.

3 Feature engineering

This section describes how we generate and choose features. We observed that there are both classification data and numerical data in the given data set. The classification data includes Anon Student Id, Problem Name, Step Name, Hierarchy, and KC(Default), while the numerical data includes Opportunity , Problem View, and Hints. Considering the variety of the data, We tried to divide features into condensed features and sparse features and train them separately.

4 Condensed Features

4.1 Correct First Attempt Rate

We first considered the direct relation between the result of Correct First Attempt and other columns of data. A simple way to represent this relation is to

calculate the Correct First Attempt Rate, we note if as `CFAR` for short. Use `Anon Student Id` for example. The `CFAR` of a particular student with Sid is :

$$CFAR|Sid = \frac{\#\,of\,rows(AnonStudentId == Sid\,\&\,CFA = 1)}{\#\,of\,rows(AnonStudentId == Sid)} \tag{1}$$

`CFAR` can be calculated for many columns, such as KC(Default), Problem Name, Step Name and their combinations. For the test set, we use the `CFAR` in the train set to complete the `CFAR` columns because there is no `CFA` given in the test set. For example, a student has same `CFAR|Anon Student Id` in train set and in test set. We calculated 7 `CFARs` in total, they are:

`CFAR|Anon Student Id`, `CFAR|Problem Name`, `CFAR|Step Name`, `CFAR|KC(Default)`, `CFAR|(Anon Student Id, KC(Default))`, `CFAR|(Anon Student Id,Problem Name)`, `CFAR|(Anon Student Id,Step Name)`.

The `CFAR` feature describes the overall performance for a student and

4.2 Historical Features

Observed that there are time related columns in train data set, and considering the fact that one can remember better when it comes to knowledge which has been recently accessed, we tried to add historical features related to KCs with which a student has encountered recently. We didn't calculate history for a specific step or problem because there are too many steps and problems,there distributions are too sparse, therefore a student can hardly encounter same step in 2 adjacent days. the history can be calculated as below, here we use the history in 3 days for example.

$$KC\,History = \#(KC(Default) == x[\text{"}KC(Default)\text{"}]\,\&\,|Date - x[\text{"}Date\text{"}]| < 3days) \tag{2}$$

where x is a specific row, and Date columns is generated from the start time for that step. In the test data set, there is no time related columns, but we are aware that in each row in the test data set is selected from the origin data set, and it's always from the problem next to the last question in the train data set. Therefore, we can use the History of the last occurred same KC in the same unit in train data set to complete the test data set. There shouldn't be very large difference in time because most student finish one unit in a rather small time range.

4.3 Ability Features

Ability feature is another high level feature, it estimates a student's ability based on number of `Corrects`,`Incorrects`,Hints and `KC History`. This feature is based on the idea that if student uses more hints,have less rate of corrects on a KC,he is less likely to master it. the calculation for an ability feature is as below:

$$Ability = \frac{\#\,of\,steps(Anon\,Student\,Id == Sid, KC = kc)}{\#\,of\,Hints(Anon\,Student\,Id == Sid, KC = kc)} \tag{3}$$

5 Sparse Features

In our approach, sparse features turn out to be the preferable solution to this estimation challenge. Here, we extract feature of logs of students interaction by one-hot key transformation, where we test various category combinations and learning models.

5.1 Basic Features

Intuitively, lower-level features are considered as the basic yet important features in our method. It consists of categorical features including `student ID`, `problem hierarchy`, `problem sections`, `problem name`, step name and `KC`, as well as numerical features including `problem view` and `opportunity`.

For categorical features, we expand each field in the data frame into multiple columns, as many as the number of categories, then each record sets the corresponding bits in these columns to indicate the value it holds. For example, if there are 1130 students in our dataset, then 1130 columns named by their IDs will be added. Afterward, each row will set the column with the consistent student ID. In this stage, all fields have no preprocessing, it keeps the raw data, and match columns lexicographically.

For numerical features, we first normalize `problem view` and `opportunity` by the formula 4 as two features separately, and then try them with one-hot key transformation. The former methods give the better results in RMSE, while the latter one eliminates the sense of numerical magnitude. Note that, for multiple `opportunity` field, we only consider the minimum one in this stage, providing the intuition of Barrel effect. Another experiment tells that raw numerical values never improve the results.

$$\log(1+x) \tag{4}$$

5.2 Feature Combination

The problem hierarchy column actually contains two-dimension features: problem unit and `problem section`. Thus, the parsing becomes so essential to increase the degree of freedom for these two variables, since in test set, not all problem unit and `problem section` are combined in the same pattern with the train set.

Besides splitting problem hierarchy, we consider adding more combined features to figure out the dependency correlation. As mentioned in following section, we prefer linear classifiers as our learning model. However, they are not capable to explore the dependency among different features. We manually combine features with some meaningful implication in this stage. We have tired some pairs, triples and even higher-level features as follows. This method increases the dimensionality sharply but some basic pairs and triples are effective enough to reduce the RMSE.

Pairs	(student id, problem unit) (problem unit, problem section) (problem section, problem name) (problem name, step name) (student id, problem section) (student id, problem name) (student id, step name)
Triplets	(student id, problem unit, problem section) (problem unit, problem section, problem name) (problem section, problem name, step name)
Higher-Level	(student id, problem unit, problem section, problem name) (problem unit, problem section, problem name, step name) (student id, problem unit, problem section, problem name , step name)

5.3 KC Tokenization

Due to the string-list representation of KCs, we propose a method to tokenize the KCs field and apply one-hot key transformation to obtain the multi-occurrence series for the KC features. We find this method provides lower feature dimensionality, but glorifies the results effectively. Besides, we have considered more complicated tokenization on KC, including semantic parsing. However, some experiments only give the overfitting results.

Additionally, considering the connection between KCs and opportunity, we set normalized opportunity value to corresponding KC column instead of 0-1 representation. This reveals the weights on each KC, and brings slight improvement.

6 Model Training

Because we separated features into condensed ones and sparse ones, it is hard to train them in one simple model. therefore, we selected different models for sparse features and condensed features, and tried to ensemble them to get use of all features.

6.1 Model for condensed features

After feature generation, we have 12 features(7 `CFAR` , 3 History, 2 `Abilities`).The aim of the task is to predict CFA on the test data set, which only has 2 values. Though we can give any value between 0 and 1 as output, the task is similar to a classification question.In the condensed Features, the number of dimension not very large, but we don't know which dimension of data is more important, so we chose random forest because it can show importance of different features. After first few test we find that the importance from `CFAR` features related with Step Name and Problem Name are very high, but we we observed that there

are not many repentance of Step Name and Problem Name, also, they are relatively sparse features and is causing serious overfitting. so the result was not very good(the RMSE was around 0.42). After test We removed them and only remains CFAR — Anon Student Id and CFAR — Anon Student Id, Unit in CFARS.

For the hyper-parameters, we only chose to adjust the Number of Trees and the max-depth, and let others set default. We tested the hyper parameter by comparing their results on the test data set, and chose the best one. The best RMSE result using only condensed model is 0.375.

6.2 Model for sparse features

Sparse features composes a very large matrix with high sparsity. We consider using sparse matrix container and solver in our case to reduce the memory usage. Due to the large training size, nonlinear classifier may not be a ideal solution, since they are so time consuming. We consider several linear model alternatively. Among all of them, logistic classifier beats others in RMSE which is its direct optimization target. AdaBoost significantly reduces the overfitting level, while gives high RMSE.

7 Conclusion

After training 2 methods separately, the RMSE for the condensed model is around 0.375 and the sparse one is around 0.345, which is far better. We thought it would enhance the performance if we ensemble 2 models. we tried 2 methods. The First one is to first generate the output of the sparse model, and put that output into condensed model as one more dimension of feature, but the result is no better than the single sparse model. We thought it may because the condensed feature cannot provide more information than the sparse features, and it brings some under-fitting factors at same time.

Then, we tried put all condensed features to sparse features directly, and the result is easy to guess. 7 features didn't make any difference to the sparse data with 658691 columns.

In conclusion, sparse features are the preferable solution to this educational datamining challenge. High-level features are conclusive but reduce the capacity of information, while sparse features tend to cause overfitting but strong yet to predict better results. Further trials can be focused on the combination of condensed features and sparse features, using both raw information and conclusive information.