proposal

May 21, 2017

1 Machine Learning Engineer Nanodegree

1.1 Capstone Proposal

Michael Feng
May 16th, 2017

1.2 Proposal

1.2.1 Education Data Mining Challenge

A lot of universities are currently using Artificial Intelligent Tutoring systems. These systems are used to tutoring students on some courses. Students can acquired knowledge from remote and anytime they want. The time and hours students spend on the system are valuable for both students and universities. Analysis of these behaviors can be a good practice, so that Universities can use students' behaviors on tutoring system to optimize their tutoring systems and exercises. Two specific systems include the the Carnegie Learning Algebra system, deployed 2005-2006 and 2006-2007.

Meanwhile, improved models could be saving millions of hours of students' time (and effort) in learning.

From a scientific viewpoint, the ability to achieve low prediction error on unseen data is evidence that the learner has accurately discovered the underlying factors which make items easier or harder for students. Knowing these factors is essential for the design of high-quality curricula and lesson plans (both for human instructors and for automated tutoring software). So we have the potential to influence lesson design, improving retention, increasing student engagement, reducing wasted time, and increasing transfer to future lessons.

1.3 Domain Background

A model which accurately predicts long-term future performance as a byproduct of day-to-day tutoring could augment or replace some of the current standardized tests from the goal of assessing performance while simultaneously assisting learning. Previous work has suggested that these analysis and improvements is indeed possible: e.g., an appropriate analysis of 8th-grade tutoring logs can predict 10th-grade standardized test performance as well as 8th-grade standardized test results can predict 10th-grade standardized test performance (Feng, Heffernan, & Koedinger, 2009). Our algorithms may provide insights that allow important improvements in optimization Artificial Intelligent Tutoring Systems.

We are gonna to predict student Student First Correct Attempt probabilities on problems from logs of student interaction with Intelligent Tutoring Systems. Currently, I'm applying a MCS of University of Illinois at Urbana-Champaign. Some of the university courses may later be taken online with some tutoring systems. In personally, i think it's really important and valuable for all kinds of students and universities to know some interesting insights of these tutoring systems.

Problem Statement A step is an observable part of the solution to a problem. Because steps are observable, they are partly determined by the user interface available to the student for solving the problem. Behavior steps are including: find the radius of the end of the can (a circle) find the length of the square ABCD find the area of the end of the can find the area of the square ABCD find the area of the left-over scrap etc.

This whole collection of steps comprises the solution. A step record is summary of all of a given student's attempts for a given step. Training data are step records of different students on different exercises. These data sets come from multiple schools over multiple school years. We gonna use thousands of student-step (a record of a student working on a step) to predict Correct First Attempt. Correct First Attempt: our prediction value, a decimal number between 0 and 1, that indicates the probability of a correct first attempt for a student-step. The target data is 0 or 1 makes the problem likely to be a classification problem, so we gonna use some ensemble machine learning model to make prediction based on training data. But in order to give a RMSE measure result, we also need to output CFA's probability so that we can get RMSE for predicted result. In this section, clearly describe the problem that is to be solved. The problem described should be well defined and should have at least one relevant potential solution. Additionally, describe the problem thoroughly such that it is clear that the problem is quantifiable (the problem can be expressed in mathematical or logical terms) , measurable (the problem can be measured by some metric and clearly observed), and replicable (the problem can be reproduced and occurs more than once).

1.4 Datasets and Inputs

Stamper, J., Niculescu-Mizil, A., Ritter, S., Gordon, G.J., & Koedinger, K.R. (2010). [Data set name]. [Challenge/Development] data set from KDD Cup 2010 Educational Data Mining Challenge. Find it at http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp

The data sets are provided for familiarizing ourselves with the format and developing our learning model.

For a description of the format of the data, we can see the official Data Format page.

We will split the train data into 3 part: training set, validation set and test data set. Use training set to train our model, validation set to validate model, then test set to know model's performance on unidentified data.

Input data and features:

- Data size: 8918055 records (algebra_2008_2009_train.txt)
- Train portion of records: 0.7 x 8918055 or 0.6 x 8918055
- Validation portion of records: 0.15 x 8918055 or 0.2 x 8918055
- Test portion of records: 0.15 x 8918055 or 0.2 x 8918055

We will try different portion for performance. For all training data sets, each record will be a step that contains the following attributes: - Row: row number for record. - Anon Student Id: unique, anonymous identifier for a student - Problem Hierarchy: the hierarchy of curriculum levels containing the problem. - Problem Name: unique identifier for a problem - Problem View:

the total number of times the student encountered the problem so far. - Step Name: each problem consists of one or more steps (e.g., "find the area of rectangle ABCD" or "divide both sides of the equation by x"). The step name is unique within each problem, but there may be collisions between different problems, so the only unique identifier for a step is the pair of problem_name and step_name. - Step Start Time: the starting time of the step. Can be null. - First Transaction Time: the time of the first transaction toward the step. - Correct Transaction Time: the time of the correct attempt toward the step, if there was one. - Step End Time: the time of the last transaction toward the step. - Step Duration (sec): the elapsed time of the step in seconds, calculated by adding all of the durations for transactions that were attributed to the step. Can be null (if step start time is null). - Correct Step Duration (sec): the step duration if the first attempt for the step was correct. - Error Step Duration (sec): the step duration if the first attempt for the step was an error (incorrect attempt or hint request). - Correct First Attempt: the tutor's evaluation of the student's first attempt on the step—1 if correct, 0 if an error. - Incorrects: total number of incorrect attempts by the student on the step. - Hints: total number of hints requested by the student for the step. - Corrects: total correct attempts by the student for the step. (Only increases if the step is encountered more than once.) - KC(KC Model Name): the identified skills that are used in a problem, where available. A step can have multiple KCs assigned to it. Multiple KCs for a step are separated by ~~ (two tildes). Since opportunity describes practice by knowledge component, the corresponding opportunities are similarly separated by '~~'.

- Opportunity(KC Model Name): a count that increases by one each time the student encounters a step with the listed knowledge component. Steps with multiple KCs will have multiple opportunity numbers separated by ~~.

Output 1: Correct First Attempt will be used as predicting target feature. Other feature gonna used to train model.

Output 2: Then We will calculate every student's CFA rate and output prediction on target feature's probability. So that we can use to calculate RMSE of model's prediction performance on probability.

Noticing: The CFAR method calculation method comes from the paper of KDD CUP10 Winner. As follow,

Feature Engineering and Classifier Ensemble for KDD Cup 2010, http://pslcdatashop.org/KDDCup/workshop/papers/kdd2010ntu.pdf

```
In [19]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt

         # Load data: Algebra 2008-2009
         train_file = 'algebra_2008_2009_train.txt'
         traindata = pd.read_table(train_file)
```

```
In [14]: # Show data example
         traindata.head()
```

```
Out[14]:    Row Anon Student Id                Problem Hierarchy Problem Name  \
         0    1  stu_de2777346f  Unit CTA1_01, Section CTA1_01-3      REAL20B
         1    2  stu_de2777346f  Unit CTA1_01, Section CTA1_01-3      REAL20B
         2    3  stu_de2777346f  Unit CTA1_01, Section CTA1_01-3      REAL20B
```

```
3    4  stu_de2777346f  Unit CTA1_01, Section CTA1_01-3      REAL20B
4    5  stu_de2777346f  Unit CTA1_01, Section CTA1_01-3      REAL20B

   Problem View Step Name        Step Start Time First Transaction Time  \
0             1      R2C1  2008-09-19 13:30:46.0  2008-09-19 13:30:46.0
1             1      R3C1  2008-09-19 13:30:46.0  2008-09-19 13:30:46.0
2             1      R3C2  2008-09-19 13:30:46.0  2008-09-19 13:30:46.0
3             1      R4C1  2008-09-19 13:30:46.0  2008-09-19 13:30:46.0
4             1      R4C2  2008-09-19 13:30:46.0  2008-09-19 13:30:46.0

  Correct Transaction Time          Step End Time        ...          \
0   2008-09-19 13:30:46.0  2008-09-19 13:30:46.0        ...
1   2008-09-19 13:30:46.0  2008-09-19 13:30:46.0        ...
2   2008-09-19 13:30:46.0  2008-09-19 13:30:46.0        ...
3   2008-09-19 13:30:46.0  2008-09-19 13:30:46.0        ...
4   2008-09-19 13:30:46.0  2008-09-19 13:30:46.0        ...

   Correct First Attempt  Incorrects  Hints  Corrects  \
0                      0           3      1         1
1                      1           0      0         1
2                      1           0      0         1
3                      1           1      0         1
4                      1           0      0         1

                                    KC(SubSkills)  Opportunity(SubSkills)  \
0                                 Identifying units                       1
1                                   Define Variable                       1
2  Write expression, any form~~Using simple numbe...        1~~1~~1~~1~~1~~1
3  Entering a given~~Enter given, reading words~~...                 1~~1~~1
4  Using simple numbers~~Find Y, any form~~Using ...             2~~1~~2~~1

                                  KC(KTracedSkills)  \
0                                               NaN
1                                               NaN
2  Using simple numbers-1~~Using large numbers-1~...
3                                Entering a given-1
4  Using simple numbers-1~~Using large numbers-1~...

  Opportunity(KTracedSkills)                                KC(Rules)  \
0                        NaN                                UNIT-HELP
1                        NaN                           VARIABLE-HELP
2                    1~~1~~1                STANDARD-MX+B-FORMULA-HELP
3                          1             GIVEN-HELP-NON-NUMERIC-PHRASE
4                    2~~2~~1  CALCULATED-VALUE-HELP-MX+B-GIVEN-X-ZERO

  Opportunity(Rules)
0                  1
1                  1
```

```
2                   1
3                   1
4                   1

[5 rows x 23 columns]
```

```
In [17]: # Show data columns
         traindata.columns
```

```
Out[17]: Index([u'Row', u'Anon Student Id', u'Problem Hierarchy', u'Problem Name',
                u'Problem View', u'Step Name', u'Step Start Time',
                u'First Transaction Time', u'Correct Transaction Time',
                u'Step End Time', u'Step Duration (sec)',
                u'Correct Step Duration (sec)', u'Error Step Duration (sec)',
                u'Correct First Attempt', u'Incorrects', u'Hints', u'Corrects',
                u'KC(SubSkills)', u'Opportunity(SubSkills)', u'KC(KTracedSkills)',
                u'Opportunity(KTracedSkills)', u'KC(Rules)', u'Opportunity(Rules)'],
               dtype='object')
```

```
In [18]: # Print the number of unique students' id
         print 'Number of unique students: ', len(np.unique(traindata['Anon Student Id']))

         # Print the number of unique problems
         print 'Number of unique problems: ', len(np.unique(traindata['Problem Name']))

         # Print the number of KC(SubSkills)
         print 'Number of unique KC(SubSkills): ', len(np.unique(traindata['KC(SubSkills)']))
         # Print the number of KC(KTracedSkills)
         print 'Number of unique KC(KTracedSkills): ', len(np.unique(traindata['KC(KTracedSkills
         # Print the number of KC(Rules)
         print 'Number of unique KC(Rules): ', len(np.unique(traindata['KC(Rules)']))

         # Print the number of total records
         print 'Number of total records: ', len(traindata)
```

```
Number of unique students:  3310
Number of unique problems:  188368
Number of unique KC(SubSkills):

/Users/michaelfeng/code/tf/venv/lib/python2.7/site-packages/numpy/lib/arraysetops.py:216: Future
  flag = np.concatenate(([True], aux[1:] != aux[:-1]))


 1829
Number of unique KC(KTracedSkills):  922
Number of unique KC(Rules):  2979
Number of total records:  8918054
```

1.5 Solution Statement

Based on existing data given by the competition, we only have target feature Correct First Attempt. Since the CFA only contains value 1 and 0. So we can use classifier to predict the target result. We will train a classifier on training data set with validation data set together.

Since original competition KDD CUP 2010 ask for probability of students' correct first attempt on problems. We will transform the problem from classification to regression problem.

We proposed replacing each categorical feature with a numerical one by using the "correct first attempt rate" (CFAR) .

The CFAR can be expressed by:

CFAR = #steps with student id = sid and CFA = 1 / #steps with student id = sid

This CFAR directly connects a feature and CFA, which is now the target for prediction.

So that the classification problem will be transformed to regression problem.

Therefore, we can measure our model on test data with RMSE.

Which can be represented as follow:

y': Predicted target probability of First Correct Attempt(numpy ndarray)

y: Target's original CFAR(numpy ndarray)

n: Number of data records

RMSE = np.sqrt(np.sum(np.square(y' - y)) / n)

Our goal is to minimize the RMSE on test data set which comes from portion of training data set.

The test data set in original data is used to predict and submit to competition's leader board so that attendants can achieve their ranks. But in our experiments, there is no leader board. So we only use portion of training data as test data. That's important to know.

1.6 Benchmark Model

KNN is one of the most popular algorithms for find the nearest neighbors in data. For our KDD CUP 2010 competition problem, we suppose to find K nearest student for one student. So these neighbors' average probability of first correct attempt on problems is thought to be the students' probability on that problem.

Their average probability of first correct attempt will be caculated by the number of students' in K whose First Correct Attempt is 1 divide to the total number of students in K.

- N: Number of Students(CFA = 1)
- K: Nubmer of Students in K
- P: Probability of student's first correct attempt on problem

$P = \frac{N}{K}$

The probability calculated by Benchmark Model will also caculate RMSE with CFAR on test portion of training data. To compare the result to our design solution result. We should optimze our solution result in better performance than Benchmark Model result. So that we can achive a convinciable performance.

1.7 Evaluation Metrics

Since the data comes from KDD CUP 2010 competition. Therefore, we don't have an unidentified portion to validate model's generalize performance. So we will only train on the training portion of each data set, meanwhile, use part of training portion data as validation set, and will then be

evaluated on our performance at providing Correct First Attempt values for the test portion(part of training data).

We will compare the predictions we provided against test portion(part of training data) true values and calculate the difference as Root Mean Squared Error (RMSE).

The square root of the mean/average of the square of all of the error.The use of RMSE is very common and it makes an excellent general purpose error metric for numerical predictions. Compared to the similar Mean Absolute Error, RMSE amplifies and severely punishes large errors.

We will dedicate our best to acquire the lowest RMSE as possible.

1.8 Project Design

- Firstly, we gonna to load data and prepare data.
- Secondly, do one-hot encoding and regularization on numeric features for train data.
- Thirdly, we gonna visualize data into chart, so that we can see trends and scatters and etc. Removes scatters and add potential feature to data. We gonna use GMM to cluster data into different clusters so that we can add additional features.
- Finally, we will use ensemble algorithms model to train data(train set and validation set). For e.g., XGBoost, LightGBM, GBDT or etc. And tuning hyper parameters. Then predict result on test data set. Compute RMSE.

Noticing: Using notebook to present details would be a good practice.

LightGBM is a fast, distributed, high performance gradient boosting (GBDT, GBRT, GBM or MART) framework based on decision tree algorithms, used for ranking, classification and many other machine learning tasks. It supports multi CPU cores' computation. My laptop a MBP(2015 Mid) only contains an AMD GPU which is not supported well by NVIDIA's cuDNN. So I'd like to use some algorithms such as LightGBM or xGBoost so that computation could be accelerated by multi cpu cores.

Some data features are string type. But they may have highly dependency with prediction result. Such as: unit name, section name, problem name, step name and knowledge components. These features may be considered using some clustering tricks or feature combining skills to process.