Proposal

May 29, 2017

1 Machine Learning Engineer Nanodegree

1.1 Capstone Proposal

- Michael Feng

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.

Carnegie Learning systems[1] are mainly focusing on transforming math classroom to both students and teachers. A course on Carnegie Learning system tutoring algebra deployed 2005-2006 and 2006-2007. It's used to tutoring Algebra knowledge.

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).

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

These tutoring systems design to guide students and teachers the way to better learning mathematics. They are now in use in more than 2,500 schools across the US for some 500,000 students per year.

A model which accurately predicts long-term future performance as a byproduct of day-to-day tutoring could augment or replace some current standardized tests from the goal of assessing performance while simultaneously assisting learning. Previous work has suggested that these analyses 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)[2].

Our experiment and data mining may provide insights that allow important improvements in optimization Artificial Intelligent Tutoring Systems. We are gonna to predict student first correct attempt probabilities on problems from logs of student interaction with the Carnegie Learning Algebra System data. Currently, I'm applying a MCS of University of Illinois at Urbana-Champaign. Some 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.

1.4 Problem Statement

Terminologies referred in our proposal are as follows:

- Step: an observable part of the solution to a problem.
- Transaction: students' interaction with tutoring system. For e.g., hint request, incorrect attempt, or correct attempt. Every transaction is referred to as an attempt for a step.
- Solution: All steps comprises to figure problem out.
- Problem: A task for a student to perform that typically involves multiple steps.
- Answer: A final step to one problem
- Knowledge Component(KC): A piece of knowledge or information could be used to accomplish a problem. A KC is associated with one or more steps. Also one or more KC can be associated with one step.
- Opportunity: A chance for student, representing he or she has acquired of the KC that are needed to accomplish a problem.
- Opportunity count: Steps(contains KC) count for student that are needed to solve a problem. So, student can have multi chances to solve a problem.

The students solve problems in the tutor and each interaction between the student and computer is logged as a transaction. Our data are records of transactions(which we've mention before).

A record is a step summary of all of a given student's attempts info for given exercise problem. Training data are records of different students's transactions on different exercises problems.

First, We are gonna use thousands of student-record (a record of a student working on a step) to predict students' correct first attempt on problems. CFA(Correct First Attempt): a binary number, which is 0 or 1. A CFA value indicates the result of a student's correct first attempt on one problem.

Then, the probability values of different students' first correct attempt on different problems are going to be predicted as our final prediction result.

The CFA is 0 or 1 makes the problem likely to be a classification problem. In order to calculate log loss, we need to output the probability of CFA equals to 1 for students on problems.

Probability can be calculated by sklearn. We can find some models or methods at Sklearn Home and sklearn.svm.libsvm.predict_proba.

So, we can get model's performance metric like log loss on unseen data for measurement.

1.5 Datasets and Inputs

Data sets	Students	Steps	File
Algebra I 2008-2009	3,310	9,426,966	algebra_2008_2009.zip

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 evaluate 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 id for a student
- Problem Hierarchy: the hierarchy of curriculum levels containing the problem.
- Problem Name: unique identifier for one 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 the duration 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 student's first attempt on a 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): steps(contains KC) count for student that are needed to solve a problem. So, student can have multi chances to solve a problem. 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, we can use to calculate log loss of model's prediction performance on probability.

Noticing: The CFAR method calculation method comes from the paper of KDD CUP10 Winner[3].

```
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
```

1.6 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 numeric measurable 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: - CFA: Student's correct first attempt - N: Total number of one student's all records(CFA = 1) - T: Total number of one student's all records(both CFA = 0 and CFA = 1)

CFAR = $\frac{N}{T}$

This CFAR directly connects a feature and CFA, which is now the target for prediction. CFAR is numeric between 0 and 1. So the classification problem can be transformed to numeric measurable problem. Our model on test data can be measure with log loss.

Which can be represented as follows:

- y′: Predicted target probability of Correct First Attempt(CFA=1)
- y: Target's original CFAR

$-\log P(y' \mid y) = -(y' \log(y) + (1 - y') \log(1 - y))$

Our goal is to minimize the log loss on test data set which coming 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.

Since it's a classification problem, and a lot of data records involved in, sklearn's SGD classifier would be a good choice rather than linear SVM. Cause SVM takes a lot of time in large data set.

But I would consider LightGBM as my best choice. LightGBM has a very good performance not only in time dimension but also in accuracy. We can find performance experiement by LightGBM team from here.

LightGBM is a gradient boosting framework that uses tree based learning algorithms. Based on my personal experience on some side-projects I've done myself, LightGBM is 10 times faster than xgboost, may be 20 ~ 50 times faster than sklearn's alogorithm. Besides, it's distribution supported on multi devices. If we don't have a cuDNN good supported GPU, i would say it's a good choice.

Recently, LightGBM seems already support AMD GPU and tested on a lot of data sets and hardware. I will give it a try using my laptop's AMD GPU.

```
In [ ]: # Use lightGBM algorithm, Code example

        from pylightgbm.models import GBMClassifier
        from sklearn import cross_validation, metrics

        test_features_scale = scaler1.fit_transform(test_features)
        test_labels_scale = scaler2.fit_transform(test_labels)

        clf = GBMClassifier(exec_path=exec, min_data_in_leaf=1)
        clf.fit(x_train, y_train, test_data=[(x_test, y_test)])
        y_pred = clf.predict(x_test)
        print("Accuracy: ", metrics.accuracy_score(y_test, y_pred))
        print("log_loss: ", metrics.log_loss(y_test, y_pred))
```

1.7 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 the nearest K students for one student. So these neighbors' average probability of first correct attempts on problems is thought to be the student's probability on that problem.

Their average probability of first correct attempt will be calculated by the number(K) of students' whose first correct attempt is 1 divide by the total number of students in K.

- N: Number of Students(CFA = 1).
- K: Number of Students in K.
- Py: Probability of student's correct first attempt on one problem. P = $\frac{N}{K}$
- CFARy': find this term and represent formula in Solution Statement section.

Log loss on test portion of training data could be represented as follows:
-log P(y' | y) = -(y' log(y) + (1 - y') log(1 - y))

The probability calculated by benchmark model will also calculate log loss. To compare the result to our design solution result. We should optimize our solution result in better performance than Benchmark Model result. So, we can achieve a good performance.

1.8 Evaluation Metrics

Our 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 log loss.

- y': Predicted target probability of Correct First Attempt(CFA=1)
- y: Target's original CFAR

-log P(y' | y) = -(y'log(y) + (1 - y') log(1 - y))

The use of log loss is very common and it makes an excellent general purpose error metric for numerical predictions.

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

1.9 Project Design

- Firstly, we are gonna to load data and prepare data.
- Secondly, do one-hot encoding and regularization on numeric features for train data.
- Thirdly, we will visualize data into chart, so that we can see trends and scatters. Remove scatters and add potential feature to data. We are 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 log loss.

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. Computation could be accelerated by using 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.

1.10 Reference

[1] Carnegie Learning system, http://www.carnegielearning.com/

[2] Addressing the assessment challenge with an online system that tutors as it assesses, http://repository.cmu.edu/cgi/viewcontent.cgi?article=1303&context=hcii

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

[4] Collaborative Filtering Applied to Educational Data Mining, http://pslcdatashop.org/KDDCup/workshop/papers/KDDCup2010_Toescher_Jahrer.pdf