Modelling Student Perfomance in Massive Online Courses.

Okonda Joseph
Stanford University
jlikhuva@stanford.edu

Abstract

This project applys supervised machine learning to Massive Online Open Courseware (MOOC) data. We develop and analyze models to predict learner performance in large online classes. First, we fit regression models with parameters derived from a learner's learning-platform interaction patterns to predict their total normalized quiz score in a given class. We also fit classifiers to the same features but with the discretized response variable.

1. Introduction

Online learning promises to revolutionize the education industry by delivering high quality educational content to a large number of remote learners at an affordable price. However, data from some large, open online classes suggests that only a handful of the students who enroll in these classes manage to complete them[4]. Many believe that one way of solving the problem of high attrition rate in these course is by adopting a personalized-learning approach, where content is tailored to individual learners or groups of learners based on their needs and by *automatically* offering 'just in time' personalized assistance. This project posits that, a function that relates a learner's final score in a class, to his/her interaction with the learning platform can be used a trigger for such intervention.

The project's research questions are:

- Can we use data on how a learner interacts with the learning platform (Videos, Forums, and other course content) to predict the learner's score in the course?
- Can we generalize a model trained on one course to other courses. That is, How good are parameters trained on learner interaction data from course A in predicting the scores of learners in course B where $A \neq B$?

2. Data Sets and Training Features.

The Project used clickstream data collected from 4 MOOCs offered through Stanfords OpenEdx Learning Platform. The data was acquired from Stanford's Center for Advanced Research through online learning IRiSS.
Table 1 Summarizes the available data. The data catalogues each students interaction with the learning platform. Of the students enrolled in each course, we only consider those who attempted at least one quiz. This explains the difference between the values for 'Total Enrollment' and 'Training Data' in 1 below. The platform logs various events for each student enrolled in these classes. In this project we will focus on three types of events

Course	Enrollment	Training Data
Mining Massive Datasets	13 818	430
Algorithms (I) & (II)	18 358	333
Introduction to Compilers	30 485	2200
Introduction to Statistical Learning	125 105	39616

Table 1. The 4 MOOCs from which the training data comes.

- *Video interaction events*: Actions the learner does while interacting with teh video component of a course, e.g playing a video.
- *Platform interaction events*: Actions the learner perfoms on components of the learning platform that are not videos. For instance downloading transcripts.
- *Quiz interaction events*: Actions involving the lecture quizzes.

From these event types, we derive the following training features for each student.

1. Video
   (a) The number of times the learner paused videos.
   (b) The number of times the learner played videos
   (c) The number of times the learner changed the video speed
   (d) The number of times the learner stopped videos
   (e) The number of video seeks, i.e using the transcript to go to a specific part of the video.

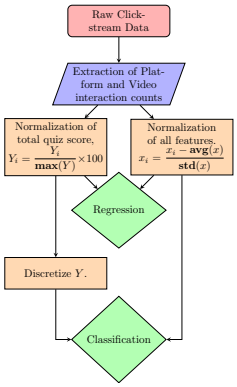

Figure 1. The training data creation process.

2. Platform
   (a) The number of times the learner downloaded content,
   (b) The number of interactions with the discussion forum
   (c) The number of interactions with lecture problems
   (d) The number of other interactions not captured by the first three.
3. Quizzes
   (a) The average number of attempts the learner spent on each quiz problem
   (b) The total number of points the learner garnered from the quizzes. This is our response variable, Y.

3. Model Building and Analysis.

In this section we describe various machine learning models that we fit to the training data. Additionally, we discuss our error analysis methods and how we measured the perfomance of our models. The algorithms were implemented with the aid of the *scikit-learn*[3] machine learning library.

3.1. Models.

- *L_2 Regularized Linear Regression.*
  Here, we model the response variable as a linear function of our training features. More specifically, our model finds the parameters $\theta \in \mathbb{R}^p$ that minimize the function below:

$$\ell(\theta) = \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)}) + \lambda \sum_{i=1}^{p} \theta_i^2$$

  The first term is the mean-squared error loss function and the second is the L_2 penalty used to regularize the model[2]. The optimal parameters are found using Stochastic Gradient Descent.[3]

- *$K-$Nearest Neigbor Regression*
  Here, we fit a non-parametric model to the data, which predicts the $y^{(i)}$ value of an unseen data-point by looking at the K points that are closest to it for whom $y^{(i)}$ is known. In order to find the points to use in the estimation, the model measures the Euclidean distance,

$$\mathbf{d}(q,p) = \left(\sum_{i=1}^{p} q_i - p_i \right)^{\frac{1}{2}}$$

  between all our training datapoints and the query point.

- *Neural Network Regression*
  Here, we create a 4-Layer Fully connected neural network with ReLU $f(x) = \mathbf{max}(0, x)$ nonlinearities and a linear output layer. The model schematic is shown below.

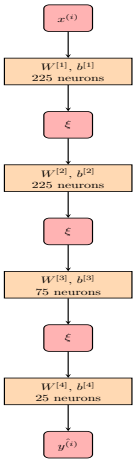

Figure 2. Neural Network.

  We fit the parameters of the model using Gradient Descent using mean squared error as the loss function with an L_2 penalty on the parameters for regularization, just like we did with Ridge regression.

- *Multinomial Logistic Regression*
  First, we turned the problem into a classification task by discretizing the target variable $y^{(i)}$ into quartiles. Since we normalized the response variable to be a percent, we obtain 4 quartiles.

  On the discretized data, we first fit a softmax regression model. This model tries to estimate the probability of a sample belonging to one of the four classes, by using the softmax function shown below:

$$P(y^{(i)} = j | x^{(i)}) = \frac{e^{x^{(i)}}}{\sum_{j=1}^{4} e^{x_j^{(i)}}}$$

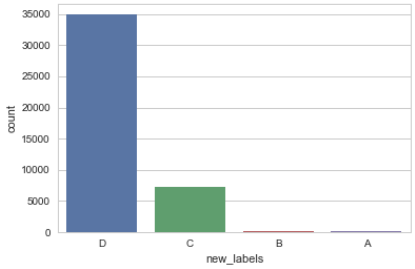

Figure 3. The 4 Quartiles on all training data combined.

	MMDS	Compilers	Algorithms	Stats.	All.
KNN	23.50	20.02	20.03	13.22	13.76
Ridge Reg.	23.60	19.50	**19.33**	13.77	**13.44**
NN Reg.	**23.42**	**19.32**	19.54	**12.79**	13.45

Figure 4. RMS Errors of the Regression models on the 4 courses.

To find the parameters of the model, we minimize the cost function shown below:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{4} 1\{y^{(i)} = j\} \log \frac{e^{z^{(i)}}}{\sum_{l=1}^{4} e^{z_l^{(i)}}}\right]$$

- *Support Vector Classification*
  As our final model, we fit an SVM to the dicretized data. The SVM tries to find the optimal separating hyperplane and uses that to classify unseen datapoints. One method of finding the parameters that define the optimal hyperplane is by minimizing the regularized average hinge loss[1]

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[1 - y^{(i)}K^{(i)T}\theta\right]_+ + \frac{\lambda}{2}\theta^T K\theta$$

Here K is the Kernel matrix. In our specific case, we use the Radial Basis Function Kernel(RBF) [1]

$$K(x,z) = \mathbf{exp}\left(\frac{||x-z||^2}{2\sigma^2}\right)$$

3.2. Analysis

To answer our first research question, i.e whether a student's interaction profile can be used as a predictor for overall perfomance, we evaluated our models on a hold-out cross-validation set for each of the four courses. The results for the regression models are shown in the table below.

We observe that on average, the Neural Network performs better that the other two regression models, although not by a large margin. In fact, on one of the courses (Algorithms) and on the combined data (All) Ridge regression exhibits a smaller Root Mean Squared Error.

To evaluate our classification models, we measure the F-1 score of the two classifiers. We use this instead of accuracy because our discretized data is highly unbalanced, as seen in Figure 3. The chart below summarises the results. We observe Both classifiers have roughly the same F-1 scores on each of the four courses. To better understand

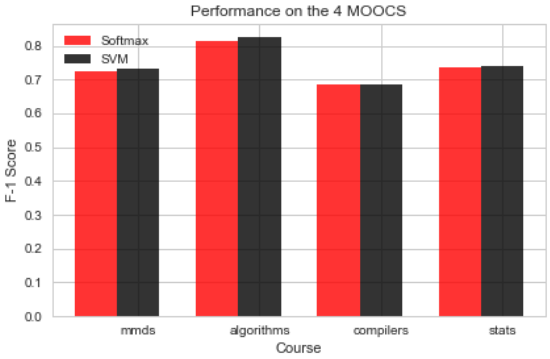

Figure 5. F-1 Scores for our classifiers.

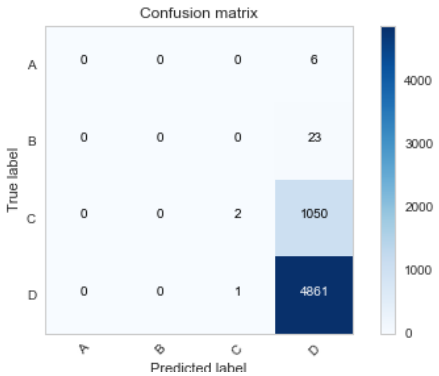

Figure 6. Confusion matrix of Softmax on stats.

our classifiers, we plot their confusion matrices for one of the courses.

We see that our clasifiers are good at identifying students in the lowest quartile, but do a poor job when assigning labels to samples from the other classes. This can be attributed to the fact that, our training data is highly unbalanced, with samples from the first and second quartile making up only a tiny fraction of the training data. One way of solving this problem would have been to down-sample the first and second quartiles. However, doing so would have greatly reduced the amount of training data used to fit the Neural network.

Finally, to answer our second research question, we fo-

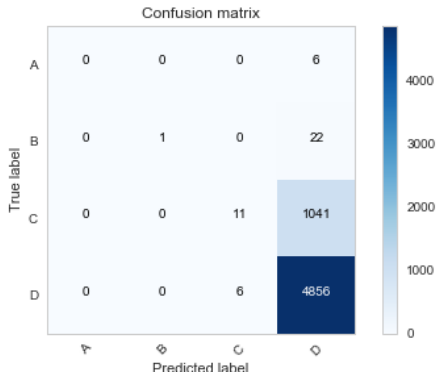

Figure 7. Confusion Matrix of the SVM on stats.

cus on neural-network regression. We train the network on each of the four courses, and for each course, we evaluate the models generated on every other course by measuring the RMS errors. The table below summarises the results. The columns indicate the course on which the model was trained and the rows indicate the course on which the model was evaluated. The diagonal entries are the values from Figure 4 above.

	MMDS	Compilers	Algorithms	Stats.
MMDS	23.42	24.66	24.92	24.97
Compilers	17.74	19.32	17.34	17.36
Algorithms	17.74	16.89	19.54	16.82
Stats.	14.28	13.12	13.01	12.79

Figure 8. RMS Errors when training and testing courses are different.

We observe that, we indeed generalize to 'unseen courses'. To see this, focus on the rows of the table above. We see that having the training and testing courses be different does not negatively affect the perfomance. In fact, it improves it in some cases. For instance, the RMSE for 'Algorithms' is lower when evaluated models trained on other courses. An explanation for this could be that, the other courses have more samples and thus models trained on them are able to generelize better.

4. Conclusion and Future Work.

This project demonstrated the ability to use a learners click behavior to predict their performance in an online class as measured by the number of points they accrue from the courses quizzes. This was done in the context of creating a system that can automatically flag when a student needs assistance. However, the project does not tell us in what areas the flagged student needs assistance in. Additionally, instead of using the features used in the project to predict a students score at the end of the course, one could include a temporal component in the analysis and fit models such as Hidden Markov Models.

5. Acknowledgements.

We'd like to thank Dr. Andreas Paepcke of the InfoLab for help in obtaining the training data. We also acknowledge the aid of Stanford's Center for Advanced Research through Online Learning for collecting and making the data available to researchers. We also thank the CS 229 staff for organizing a fun and informative class.

References

[1] J. Duchi. General loss functions. *Machine Learning*, pages 1–12, 2016.

[2] A. Ng. Regularization and model selection. *Machine Learning*, pages 1–8, 2012.

[3] 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. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] D. Yang, R. Kraut, and C. Rose. Exploring the Effect of Student Confusion in Massive Open Online Courses. *Journal of Educational Data Mining*, 8(1):52–83, 2016.