Modeling MOOC Dropouts

Degao Peng (degao@stanford.edu), Gaurav Aggarwal (gaggarwa@stanford.edu)

Abstract

In this project, we model MOOC dropouts using user activity data. We have several rounds of feature engineering and generate features like activity counts, percentage of visited course objects, and session counts to model this problem. We apply logistic regression, support vector machine, gradient boosting decision trees, AdaBoost, and random forest to this classification problem. Our best model is GBDT, achieving AUC of 0.8763, about 3% off the KDD winner.

1. Introduction

Massive Open Online Course (MOOC) revolutionizes education by providing easy access to course materials through Internet. MOOC can achieve a very large scale compared to traditional schools thanks to the availability of the materials. However, MOOC also faces an embarrassing issue: because there is almost no cost to register to a course, the dropout rate is very high. The completion rate on Coursera is only 7%-9% [1]. Studying MOOC dropouts can help reducing dropout rate and boost the values of MOOC. Better understanding dropout behaviors can also help improving MOOC content/websites to retain more students.

In this project, we are modeling MOOC dropouts using the 2015 KDD cup data [2]. The task is to predict an enrollment (an enrollment is a (user, course) tuple which means user registers a course) will drop out or not, based on given data. The term 'dropout' is refer to 'all who failed to complete' a course. Input is data sets with majority of the data being user activity log data, and some course structure data, but no user profile data. It is challenging to extract useful features from log data to training. Meanwhile, this problem is also very similar to other problems many websites like LinkedIn and Facebook are facing with massive event log data. Techniques in modeling MOOC dropouts can also be beneficial to solve these problems.

2. Related Work

In In this section we will briefly discuss related works on MOOC dropouts prediction. We will go over major approaches explored in MOOC dropouts literature.

A large number of prior research has focused on contextual information like discussion posts. Yang et al., 2013 [11] tried to assess posting behaviour in discussion forum and analyze student participation by model a concept of social positioning. Aggregated weekly actions (posts & replies) in discussion forum were considered as a measure of social positioning. This approach is very restrictive in application because of its reliance on a small stream of input. Discussion forum is just one of the input among problem set, video and many other events. Similarly, some research (Ramesh et al, 2014 [13]) depends on discussion forum events including viewing a post, upvote/downvotes, "cognitive engagement", and "sentiment".

As per Huang et al. 2014 [12], not all students engage in peer-to-peer discussion forum which reduces our coverage for social positioning.

Meanwhile, some prior research do try to model behavioural aspect of students. Kloft, et al. 2014 [4] tries to identify most active time for a user and how that relates with drop outs. For instance features like hours of activity facilitates identification of night users and day users. Similarly, Sinha, et al. 2014 [10] use a sequence of events. Features such as clickstream n-gram looks promising. Both these work are very restricted in modelling algorithm.

Kloft, et al. 2014 [4] explores only SVM classification. Another set of prior works explore student interaction with lecture videos. Kim, Juho, et al. 2014 [14] analyzes click events from student activities for a given MOOC video, namely skipping, zooming , playing, panning, pausing, and quitting.

After all, not all features mentioned above are available in our dataset. We take a more generic approach by working with student events and course structure data sets. We explore multiple learning methods by leveraging features from [4][10] and creating some of our own.

3. Data Description

Our data sets are sourced from KDD Cup 2015 [3]. All the data sets are event and relationship based with no contextual information. There is no text based information available. Overall dataset is divided into four types:-

1. Course and Module Information: This data set provides hierarchical information about modules and courses. A course can have multiple modules and a module also have multiple modules within itself. Data set schema is as follows:-
   a. Module: <course_id,module_id,category,children,start>
   b. Course: <course_id, start_time, end_time>
   Module level start_time is only available for modules of category "chapter", "course" or "sequential". Majority of modules are of "problem", "sequential", "vertical" and "video" type.

2. Event Log: This data set provides events log for user actions. Event schema is <event_time, source, event_type, module_id, enrollment_id> . Event types are evenly distributed except "wiki" (1%) and "access" (30%).
   a. Some events have "event_time" before associated module "start_time". That is, student interacted with

module even before module was posted. This seems like an anomaly in the data with about ~1% of enrollment_ids in our train set.

b. "page_close" events are corrupted as all page_close events are associated with same module which do not exist in course information data set.
c. We find that many events (15%) are associated with
d. modules which are not present in module information dataset.

3. Enrollment: This data set is a mapping of "enrollment_id", "student_id" and "course_id". Each student can opt for multiple courses.
4. Completion: This is the label set with "enrollment_id" and 0/1 "label" (1= drop out). Class distribution is skewed towards dropout labels (79%) against non-dropouts (21%) labels.

4. Featurization

1.1 Round 1: Basic count based features

Since there are no directly available basic user profile data (e.g. user age, gender, occupation and etc) and course profile data (e.g. course category, prerequisites, difficulty, and etc), the log data would be the most powerful source of knowledge. Our first batch of features are some basic counts of activities for each enrollment (enrollment is one (user, course) entry). These features are frequently used in many literatures of MOOC dropout study (e.g. [4][5]). It is in this exploration that we find KDD cup data set is somehow corrupted, with visited items in the event log absent in course material catalog. Due to the defect of the source data, some event type has zero counts. In Round 1, we generate 17 nonzero features.

1.2 Round 2: number of visits discounted by number of items

Our second bet on features is the ratio of unique items visited discounted by the total items given in the course, in the category. The original idea was to get the coverage of the items visited by a user, since this gives a hint of how the user make use of the course material and what is the proportion of material visited by the user. We validated our findings by exploring feature matrix and label vector using Weka[6]. Figure 1 is an example of our observation which showcase that students who have lower coverage on video and problems are likely to dropout. We observed appreciable F1 score improvements over Round 1. Table 2 lists all features used in Round 2 in addition to Round 1. We added one new features for percentage of course modules covered by a students from each course. In Round 2, we generate 7 additional features.

1.3 Round 3: number of activity at day X

We also observe that all courses are open for exactly 30 days. Therefore, we use 30 new features, one for number of activities[4] at day x (the first day of the class is day 1) and added them as features. We generate 30 additional features in this round.

1.4 Round 4: Activity at date of week/hour

We acknowledge that people prefer to study at a particular hour of day or a day of the week. For instance, a full-time working professional may visit MOOC only during the weekend or late at night. We hypothesised that may be a particular time of students are more susceptible to drop out. We created 24 features, one for an hour each and 7 features, one for a day of the week.

1.5 Round 5: Study session counts of week X

We observe that students study in sessions which contain multiple events in quick succession. We consider two events to be in same session if they are separated by less than 30 min gap. We created 5 features, one for total count of study session in a given week. Please note that no course extends beyond 5 weeks (5 weeks > 30 days).

1.6 Failed Features

Here we also list some failed trials on our feature explorations. For example,

1. **Total enrollments per course** - Idea was that large class size will have a negative effect on student experience and hence will affect dropout rate. This feature was not fruitful and had a negative impact on F1 score and auc score. Probable reasons could have been the fact that we have only 40 courses and number of students per course is in similar range.
2. **Maximum parallel enrollments** - Idea was to create a signal to identify students who sign up for a course just out of curiosity. These kind of students are likely to sign up for too many courses. We added this feature but we did not observed any appreciable positive impact. Probable explanation for this effect can be attributed to the fact that we have only 40 courses data and that the data was spread over 2 year (2013-14) which gives very few number of concurrent courses for students.
3. **Average lag in viewing videos** - Idea was to identify student lagging in course and hence are likely to drop out. But we do not have data about when a video was posted.
4. **Average lag in problems** - Idea was to identify student lagging in course and hence are likely to drop out. But we do not have data about when a problem set was posted.
5. **Average lag in chapter** - Similar to #3 and #4 but we had data for some of the chapter posting dates. This feature was learned by logistic regression algorithm and a non-zero coefficient was assigned, but there was no appreciable metric impact
6. **Amount of time spent in watching videos** - We need to identify whether a student completed a video or not. Sadly we do not have data to infer if a student completely watched a video or not.
7. **N-gram events**: We thought that there are some behaviour pattern[4] in event log. An ideal pattern could be student visiting "video" for learning, "problem" and then "discussion" for asking any doubts. We created features using a sequence of events for 2-gram and 3-grams. We tried 7 (number of events) X 2 (source) combination which creates 14^2 2-grams and 14^3 3-grams feature. This

approach really bloated out feature count and drastically slowed down our training step for all the models. We believe event log data from KDD is not continuous and probably a sampled set of data which removed sequential pattern and affected this N-gram feature

5. Methods

In this section we will briefly discuss learning algorithms used. They are commonly used, thus we do not introduce the details here.

1.1 Logistic Regression

Logistic Regression was the first training algorithm we tried. We tuned lambda (L2 regularization parameter) from [1.0E-4, 1.0E-3, 1.0E-2, 1.0E-1, 1.0, 1.0E2, 1.0E3]. The best lambda for the final feature sets is 1.0E-3. Logistic regression is based on logistic function with parameter θ for input x_i as follows:-

$$h(x) = \frac{1}{1 + e^{-\theta x_i}}$$

1.2 SVM

We observe that our data set might have non-linear relationships which compelled us to try SVM. We first used linear SVM and then extended to Gaussian Kernel. SVM is large margin classifier which try to separate positive and negative samples with a separating hyperplane W. We need to solve following optimization problem to find the concerned separating hyperplane:-

Minimize (in w , b) $\|\mathbf{w}\|$ subject to (for any i = 1, ..., n)

$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1$$

We tuned SVM hyperparameters using sckit-learn grid search to arrive at following results:-

- Linear SVM C= 0.001
- Gaussian SVM c = 0.10, gamma = 0.001

1.3 Ensemble

Ensemble method is a type of learning algorithm which rely on a set of models for the prediction and final output is determined on the vote of all the individual models. Generally, the ensemble model utilize bagging or boosting. The idea behind bagging is to train multiple models and each model is trained by uniformly sampling training data to a smaller set and output is determined as a weighted combination of individual models. Decision Trees are popular choice for individual models. On the other hand, Boosting [17][19] is an alternative to bagging. Boosting rely on training multiple weak learners and combine them to obtain a strong learner [20].

1.3.1 AdaBoost

AdaBoost[17] is type of boosting ensemble learning algorithm, where output is defined in terms of multiple weak learners as follows:-

$$F_T(x) = \sum_{t=1}^{T} f_t(x)$$

where each f_t is a weak learner taking an input x and returning a real valued output. The sign determine predicted class, while absolute value determined the confidence in the result. Labels are generally positive if the sample is in positive class, otherwise negative.

We tuned AdaBoost using scikit-learn grid search and found n_estimators=150 with all other default hyperparameter to be the best choice for validation set.

Individual weak learners [20] produces a response given by $h(x_i)$, for each sample in the training data. We select a weak learner at each iteration t and assign a coefficient α_t, which results in minimized E_t i.e. training error at stage t.

$$E_t = \sum_i E[F_{t-1}(x_i) + \alpha_t h(x_i)]$$

Where $F_{t-1}(x)$ is the classifier trained as part of previous stage, E(F) is also an error function and $f_t(x) = \alpha_t h(x)$ is the final weak learner.

We weight each sample in the the training data set, according to the the current error E(F_{t-1}(x_i)) on the concerned sample.This enable next set of learners to favour currently misclassified samples.

1.3.2 Gradient Boosted Decision Trees

Gradient boosted decision trees (GBDT)[15][16] is an ensemble tree learner. At each iteration, GBDT tries to generate a tree to minimizes the error. Direct minimization is usually very difficult, therefore GBDT uses a gradient descent approach to train the new trees to approximate the gradient of the total loss function.

To use GBDT, one should specify the loss function to be used, the number of trees, number of features, the learning rate, and the maximal depth of the trees. In this study, we find that using deviance loss with learning rate 0.05, 150 trees of maximal depth 5 and maximal features 0.6 X total number of features optimal to our dataset.

1.3.3 Random Forest

Random forest [19] is bagging based ensemble approach which leverage random subspaces. We tuned Random Forest hyperparameters with grid search to arrive at n_estimators=400, max_depth=7 and max_features=0.8 using sckit-learn.

The idea behind Random Forest is that we train multiple models as decision trees by randomly sampling training data as well as randomly selecting features. We combine results of individual models by taking a majority vote. Overall, individual trees in the forest is trained with three step procedure: We start with randomly sampling (with replacement) to create a new data set from entire data set. Then, we move forward by randomly selecting features. And then we start to build tree with this subset of data and features.

6. Metrics

6.1.1 Accuracy

The accuracy is the simplest metrics in a classification problem which defines as the percentage of instances correctly classified in the validation set. However, since the labels are skew with 79% positives and 21% negatives, accuracy is not a good metric since a dummy model predicting all positive will achieve accuracy of 79%.

6.1.2 CONFUSION MATRIX, PRECISION AND RECALL

The confusion matrix is a detailed numbers of instances of a class classified to a certain class. In this two class classification problem, the confusion matrix is a 2X2 matrix with 4 entries, true positive, false negative, false positive, and true negative, shown in Table 1.

Table 1 Confusion matrix

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

With the confusion matrix, other metrics can also be defined. For example,
Precision=TP/(TP+FP)
Recall=True Positive Rate=TP/(TP+FN)
False Positive Rate=FP/(FP+TN)

6.1.3 F1 SCORES

F1 score is the harmonic mean of precision and recall:

$$F1=2*Precision*Recall/(Precision+Recall)$$

6.1.4 ROC AUC

Area under receiver operating characteristic curve (ROC AUC) is the main metric we use to do parameter tuning and model selection. For classifiers with probability output, a moving threshold can change the confusion matrix and so precision and recall. The ROC curve is defined as the curve of True Positive Rate v.s. False Positive Rate varying the threshold. Figure 3 shows some example of ROC curves in this study. The area under the ROC curve, i.e. ROC AUC, is a number between 0 and 1. Also there is a similar concept of PR AUC, i.e. area under precision-recall curve. AUC is a comprehensive metric that accounts information of probability so we choose ROC AUC as our main metric for hyper parameter tuning and model selections. Note that AUC is the metric used in KDD cup to judge the winner.

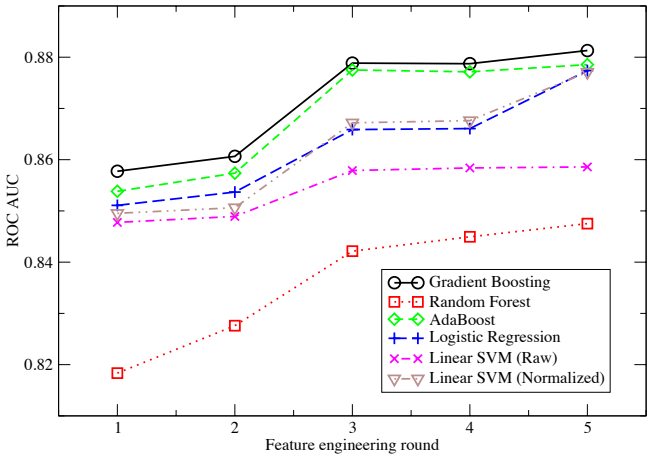

Figure 1 ROC AUC at different feature engineering round

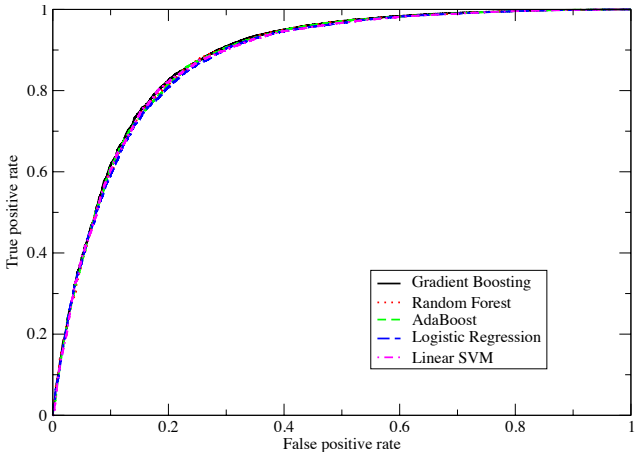

Figure 2 ROC curve for different models

7. Results

Wit With the features generated as stated in Featurization Section, we do training using logistic regression, SVM, AdaBoost, GBDT and Random Forest. All feature processing code is in Python.[Link to repo] We use SVMLight for SVM training[T. Joachims, Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.], and Scikit Learn[8] Python to do other training tasks. In order to do cross validation, we split the data according to 0.6, 0.2, 0.2 partitioning to TRAIN, VALIDATION and TEST sets. We do holdout validation for parameters tuning using ROC AUC[Citation]. Data are standardized to have zero mean and unit variance before training.

Figure 1 shows the VALIDATION AUC for different round of feature engineering, using all default parameters. At Round 5, we have in total 90 features. Models are not tuned in this figure and we just want to confirm that these feature engineering actually have positive impact on the metric. We also observe that standardization is very important for SVM. We include nonlinear SVM and ensemble methods because we think there may be nonlinear effects in the dataset. Figure 1 shows that nonlinear effects are significant in some rounds. The rest of the paper will show AUCs of all features we generated.

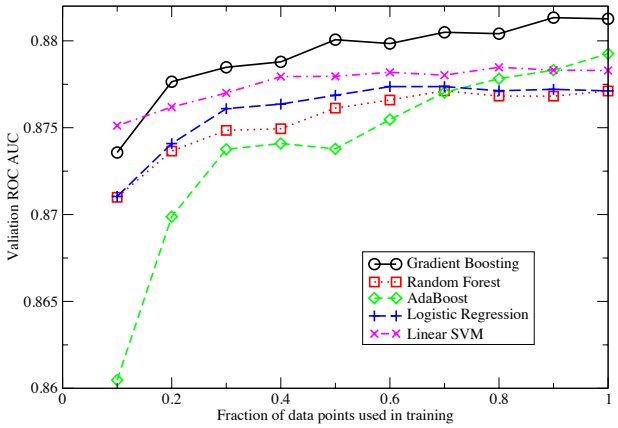

Figure 3 Learning curve of different models

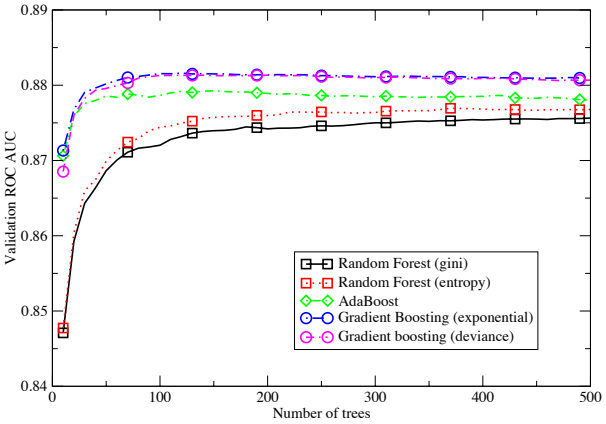

Figure 4 Convergence of ensemble tree models

In order to get best performance, we do hyper parameter tuning for each models. The hyper parameters we tuned are listed in Models Section. We do exhaustive grid search and select the best hyper parameters according to AUC on VALIDATION data. For VALIDATION and TEST sets, GBDT gave the best performance with TEST AUC 0.8763. AdaBoost and Random Forest are the second tier, with slightly better metrics than logistic regression and SVMs. The ROC curves for these models are shown in Figure 2.

To gauge the goodness of the fitting, we also draw the learning curve in Figure 3, as the curve of VALIDATION AUC v.s. fraction of training data used in the training.

Table 2 VALIDATION and TEST metrics on all models

Model	VALIDATION AUC	TEST metrics				
		AUC	F1 Score	Accuracy	Precision	Recall
Logistic regression	0.8775	0.8705	0.924	0.872	0.881	0.971
Linear SVM	0.8783	0.8719	0.924	0.873	0.882	0.970
SVM with Gaussian kernel	0.8769	0.8709	0.923	0.871	0.880	0.970
GDBT	0.8823	0.8763	0.926	0.877	0.892	0.961
AdaBoost	0.8793	0.8739	0.925	0.875	0.888	0.964
Random forest	0.8799	0.8730	0.925	0.877	0.892	0.962

For ensemble models, we also plot the VALIDATION AUC v.s. number of trees in Figure 4. These models will add a tree per iteration. We notice that Random Forest models are generally monotonically improved with more trees, while GBDT and AdaBoost have a desired number of trees to reach the best AUC. More trees require more computation time. In this case, for run time perspective, GBDT and AdaBoost is favorable compared to Random Forest.

8. Conclusion

In this section we will briefly discuss related works on MOOC dropou In this study, we apply various machine learning methods to model MOOC dropouts. We achieve a TEST AUC of 0.8763 from GBDT, which is 3% off from the KDD winner. Note that they are using a different test data sets for the judge. We explore a couple ensemble tree models not covered in the class, and they all performed better than logistic regression and SVMs after tuning. GBDT is a very powerful learner, and achieves very good results even at Round 3 compare to logistic regression.

References

1. Massive open online course. (2015, December 8). In *Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Massive_open_online_course&oldid=694372484
2. KDD cup 2015. The website may be down already. https://kddcup2015.com/
3. KDD Cup 2015 data set description: https://goo.gl/xYZZq6
4. Kloft, Marius, et al. "Predicting MOOC dropout over weeks using machine learning methods." *EMNLP 2014* (2014): 60.
5. Bussaba Amnueypornsakul, Bussaba, Suma Bhat, and Phakpoom Chinprutthiwong. "Predicting Attrition Along the Way: The UIUC Model." *EMNLP 2014* (2014): 55.
6. Weka: http://www.cs.waikato.ac.nz/ml/weka/
7. Agarwal, Deepak, et al. "Personalizing LinkedIn Feed." *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015.

8. Scikit learn package: http://scikit-learn.org/stable/index.html
9. Implementation Code: https://bitbucket.org/lics229/mooc-dropout-prediction/
10. Sinha, Tanmay, Patrick Jermann, Nan Li, and Pierre Dillenbourg. "Your click decides your fate: Inferring information processing and attrition behavior from mooc video clickstream interactions." In *2014 Empirical Methods in Natural Language Processing Workshop on Modeling Large Scale Social Interaction in Massively Open Online Courses*, no. EPFL-TALK-202095. 2014.
11. Yang, Diyi, et al. "Turn on, tune in, drop out: Anticipating student dropouts in massive open online courses." *Proceedings of the 2013 NIPS Data-Driven Education Workshop*. Vol. 11. 2013.
12. Huang, Jonathan, et al. "Superposter behavior in MOOC forums."*Proceedings of the first ACM conference on Learning@ scale conference*. ACM, 2014.
13. Ramesh, Arti, et al. "Learning latent engagement patterns of students in online courses." *Twenty-Eighth AAAI Conference on Artificial Intelligence*. 2014.
14. Kim, Juho, et al. "Understanding in-video dropouts and interaction peaks inonline lecture videos." *Proceedings of the first ACM conference on Learning@ scale conference*. ACM, 2014.
15. MLA Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." Annals of statistics (2001): 1189-1232.
16. Friedman, Jerome H. "Stochastic gradient boosting." *Computational Statistics & Data Analysis* 38.4 (2002): 367-378.
17. Freund, Yoav, and Robert E. Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting." *Journal of computer and system sciences* 55.1 (1997): 119-139.
18. Breiman, Leo. "Random forests." *Machine learning* 45.1 (2001): 5-32.
19. Schapire, Robert E., et al. "Boosting the margin: A new explanation for the effectiveness of voting methods." *Annals of statistics* (1998): 1651-1686.
20. Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer, Berlin: Springer series in statistics, 2001