Predicting Student Retention in Massive Open Online Courses using Hidden Markov Models

Girish Balakrishnan

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2013-109
http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-109.html

May 17, 2013

Acknowledgement

I would like to thank Derrick Coetzee, for being an extremely thought-provoking research partner and for guiding me with the formulation of this report. I would also like to thank Professor Armando Fox for his guidance during my Master's program, and Professor John Canny for all his help in discussing and refining our research methodologies. Last, but never the least, I would like to thank my friends and family for their never-ending encouragement without which this would not have been nearly as rewarding.

Predicting Student Retention in Massive Open Online Courses using Hidden Markov Models

Girish K. Balakrishnan, Derrick Coetzee

1 Introduction

A Massive Open Online Course (MOOC) is a large-scale web-based course that is offered to a very large number of participants. In the last few years, MOOCs have seen an increase in popularity due to the emergence of several well-designed online-education websites, such as edX and Coursera, and rising interest from top universities, such as MIT, Stanford and UC Berkeley, to open a variety of their courses to the wider public. The appeal for end consumers lies in the accessibility of high-quality education from any location regardless of personal background. MOOCs typically contain short lecture videos (10-15 minutes), as well as quizzes and homework assignments to assess students' understanding of subject matter.

Despite their advantages and popularity, their open nature means that student retention remains a significant problem, since virtually anyone can register for the course and the consequences for failing a course are minimal. This results in a large number of students enrolling in the course without ever participating once it begins, as well as students continuing to drop out at virtually every point during the course (illustrated in figure 1 for two different MOOCs offered from different universities on different websites). While the problem of the never-participating student may be due to factors that are external from the course itself, the latter phenomenon suggests that for whatever reason, students are losing the will to complete the course. This behavior has also been observed at physical colleges and universities, albeit on a smaller scale [1], and attempts have been made to understand this behavior using event-history modeling [2]. Such models proved very useful for inferring the reasons for students leaving, and for suggesting interventions for institutions to mitigate the problem.

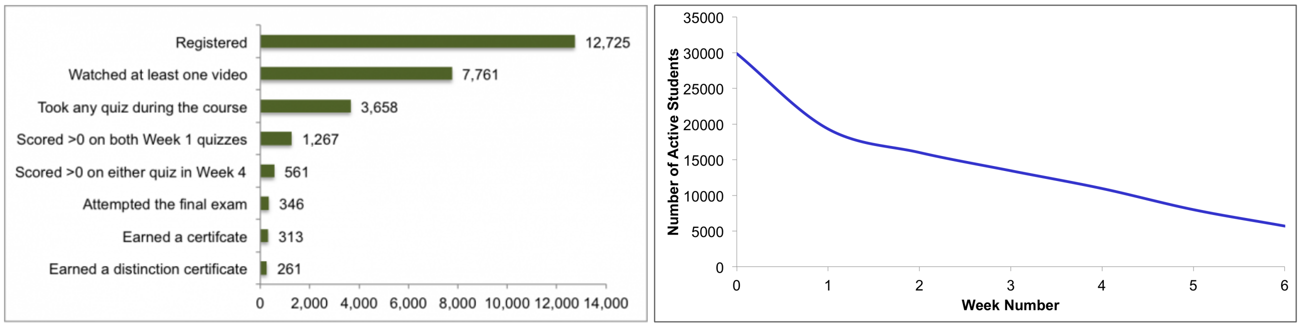


Figure 1: Student Persistence in (left) Bioelectricity, Fall 2012 (Duke University MOOC) [3], and in (right) Software as a Service, Fall 2012 (Berkeley edXs MOOC). In both cases, students drop the course every week.

Here we attempt to do the same for students in an online setting at a much larger scale, using Hidden Markov Models (HMMs, [4]) as the means to understand student behavior over time. HMMs prove a suitable choice since the hidden state can model latent characteristics of the student that influence their will to persevere, and we can then infer these from their observable interactions with the MOOC. Furthermore, an HMM for MOOCs allows us to infer a student's behavior in the next time step based on their previous state and their currently observable actions. Since there are many different types of observable actions, we explore two alternatives for creating a composite model that incorporates multiple features. In our first method, we build a multidimensional, continuous-valued feature matrix for students across the time slices of the course, and quantize this feature space, using either k-means clustering or cross-product discretization, into a discrete number of observable states that are integral to a Discrete Single Stream HMM, as studied in [5]. Using this technique, we can then apply the Baum-Welch algorithm [6][7] to train our HMM on a chosen number of hidden states. In our second method, we use a stacking ensemble approach, where we train several individual HMMs that each consider a single observable feature, and pass their results to a logistic regressor which we train to predict whether a student will be retained in the next time step of the course.

We focus our attention on the Fall 2012 offering of edX's *"CS169.1x - Software as a Service"*, which is a relatively stable course that has matured over several offerings and contains most of the archetypal features of a MOOC. With our model built, we would then like to answer the questions:

- Can we accurately predict whether a student is likely to drop the MOOC in the near future?
- Can we identify patterns in the behavior of students who eventually drop the course, and thus suggest interventions to prevent this from occurring?

2 Dataset

As previously mentioned, we focused on edX's Fall 2012 offering of UC Berkeley's *"CS169.1x - Software as a Service"* course. This was a six-week course with 29,882 enrolled students, and had the following format:

- 11 lectures, each broken up into several 10-20 minute videos and containing ungraded multiple-choice practice problems
- 4 homework assignments, each containing a differing amount of programming problems
- 4 graded multiple-choice quizzes of differing lengths

Its important to note that apart from the first week, where no graded assignments were due, the course materials were spread out pretty evenly across the six weeks of the course. The course also had an accompanying forum with basic features such as thread-following and upvoting.

Data from this course was obtained directly from edX, who generate complete data dumps for every course and distribute data for UC Berkeley courses to Berkeley researchers. The data was split across three different entities as follows:

1. Click-stream data for the entire duration of the course in JSON format, including server and browser side events. For instance, a students interaction with a lecture video (such as clicking Pause) was recorded as a browser-side JSON event, and every page visited was stored as a server-side JSON event.
2. Assignment scores stored in a MySQL database for every enrolled student.
3. Forum threads and comments stored in a MongoDB collection, along with metadata regarding number of replies, number of edits etc. Note that passive forum data, such as how many views a thread received was not stored here and had to be inferred from the clickstream data.

3 Featureset

When defining our HMM, we first split the course into six time slices, where a time slice lasts a week. This is a practical choice since the course is evenly distributed across the weeks in terms of material, and it is reasonable to assume that active students visit the course website at least once a week, as new lectures and assignments are released on a weekly basis. Thus, for each feature defined, every student gets a value for that feature for every week of the course. In other words, if S was the set of students, F the set of features and W the number of weeks, then the feature-matrix would have dimensions $|S|W \times |F|$.

Even though there are many possible features to consider, we only selected a few of them that spanned several aspects of a MOOC and seemed independent from one another since dependent observation states impair the performance of a HMM. In the following subsections, we present each of the selected features. Some other features we would like to see in future implementations are mentioned in section 7.

3.1 Student "in"/"out" State

Our primary feature is whether a student has dropped the course. This is the feature we eventually want to be able to predict, and is encoded as a sticky binary value, i.e. a student can be an active participant (the "in" state), or have dropped (the "out" state), but once the student drops the course they can no longer rejoin the course. This is a fair definition as our purpose is to try and predict when a student finally gives up on the course - a student who resumes the course later may have been temporarily unable to participate, rather than have abandoned the course.

For a given student, this feature is easily computed by examining the clickstream data to check the last date they accessed any element of the course. This definition is simple and thorough, capturing any type of interaction with the course, but fails to capture some nuances in behavior, such as the student who has dropped the course, but decides to visit the course website at a later date with no intention of catching up on the material. Under our definition, such a student would be considered to be active for a much longer period of time than they actually were. Such students are expected to be rare.

3.2 Cumulative Percentage of Available Lecture Videos Watched

Lecture videos are broken into 10-20 minute segments and are the primary means of transferring knowledge to the students. These lectures are released on a weekly basis, so a student cannot access future material in the course, motivating regular interaction with videos. Further, the provided clickstream data records the number of seconds of a particular video that was watched by a student. Thus, instead of simply asking whether lecture videos were watched, we can pose more thorough questions regarding the exact completion of lecture videos.

One possible feature is the seconds of lecture videos watched weekly, but this has the limitation that some weeks have much longer lecture segments than others, owing to an uneven distribution of lecture seconds over the weeks. Instead, we consider the cumulative percentage of available lectures watched from the beginning of class until the end of the current week. Not only does this avoid the issue of uneven distribution, due to the use of percentages, but this metric also indirectly measures how far behind a student is in the course. For instance, a cumulative percentage that significantly decreases from week to week highlights a student that is falling behind.

3.3 Number of Threads Viewed on the Forum

The course forum is the primary means of student support during the course, and the most basic interaction that a student can exercise is to view threads of questions and responses. This is an important gauge of student participation in the course, since it is a passive indicator that most students undertake. However, the click-stream data collects every single time a student visits any thread, meaning that if we simply look at the number of times a student visits any thread, then students who visit the same threads multiple times a day appear to have visited many more threads than a student that visits several different threads in a day. To overcome this issue, we impose the limit that a given thread can only be uniquely viewed once by a student in a given day. Thus, we compute for every student the number of daily-unique threads viewed in the week.

3.4 Number of Posts Made on the Forum

Another basic interaction with the course forum is making posts, whether it be asking questions, responding to them, or commenting on answers. While it is expected that far fewer students make posts, compared to the number of students that view posts, we still posit it an essential indicator of student engagement and their sense of community since it is the most active interaction a student can voluntarily have with the course. Further, it may prove interesting to compare this most active forum interaction with the most passive forum interaction - viewing threads - with respects to student retention.

3.5 Number of Times the Course Progress Page was Checked

Every edX course gives student's access to a course progress page that contains information about their completion of lecture modules and assignments. It is reasonable to hypothesize that an active and engaged student would monitor their progress a few times every week since material is released and due on a weekly basis. In this case, the frequency with which a student checks their progress in a given week indicates their engagement with all course materials, and particularly with the assignments, owing it to be a potentially useful feature.

4 Modeling Methods

All our models are discrete single stream HMMs [5], whose characteristic property is that they have a single discrete-valued observation variable at every time slice. Our choice of employing discrete single stream HMMs, as opposed to their multiple stream counterpart or continuous HMMs, was based on the fact that this is the simplest kind of HMM to train and use with plenty of well-establish tool support. In addition, it has been found [5] that when comparing single vs. multiple stream discrete HMMs, the most important factor is the independence of the features chosen, as opposed to the actual kind of model employed. As a result, we came up with the best possible model with the simplest possible method. Having said that, the simplistic parameters of a discrete single stream HMM mean that several transformations of the data had to be made to fit the model description, as outlined in the subsections to follow.

Furthermore, we use ergodic HMMs, in which every state can transition to every other state, as opposed to linear HMMs where states correspond to individual weeks. This provides two desirable properties: translation of state sequences (e.g. using the same state to model two students dropping out in weeks 2 and 4, respectively, for similar reasons), and the ability to extrapolate predictions beyond the end of the course to a hypothetical longer course.

Such a model is fully described by the following parameters[4]: N for the number of hidden states, M for the number of observable states that the observation variable can undertake, and the probability parameters $\lambda = (A, B, \pi)$, where A is the state transition probability matrix, B is the observation state probability matrix, and π is the initial state distribution vector of size N. Further, the set of hidden states are denoted as $H = \{H_1, H_2, ..., H_N\}$ with q_t being the hidden state at time t, and the set of observable states are denoted as $V = \{V_1, V_2, ..., V_M\}$ with o_t being the observable state at time t. With this definition, we can express the probability of a sequence of length T, with the observation state sequence $o = (o_1, o_2, ..., o_T)$, and hidden state sequence $q = (q_1, q_2, ..., q_T)$ as:

$$p(o, q) = p(q_1)p(o_1|q_1)\prod_{t=2}^{T} p(q_t|q_{t-1})p(o_t|q_t) \tag{1}$$

In (1), the probabilities $p(q_t|q_{t-1})$ and $p(o_t|q_t)$ are directly obtained from A and B respectively.

In our situation, however, all our models incorporate multiple features since we always consider the primary feature of student "in"/"out" state in conjunction with one or more other features. Further, while the "in"/"out" state is discrete, all our other features are continuous. Thus, we explore several options to encode a multidimensional continuous-valued featureset in a single, discrete observation variable. For the purposes of the following discussion, we denote our featureset as $F = \{F_0, F_1, F_2, ...\}$, where we label F_0 as the student's "in"/"out" state.

4.1 Comparing Different Models

In designing the procedure for training our models, we were required to tune several model parameters and select certain modeling approaches. Thus, before defining different models, we needed a fair and efficient means of comparing these choices. Ultimately we evaluate our models using binary classification metrics (see section 4.4 for exact details) that gauge the model's usefulness in predicting whether a student will drop the course or not in the next time step. However, relying solely on this method proves time consuming in situations where the final model is made up of many parts (e.g. several HMMs followed by a logistic regressor as in section 4.3), since we must wait for the entire model to be trained. Thus, we also define a score, L, that can be computed just for the current HMM being trained. L is the sum of negative log-likelihoods over all the observed sequences of “in”/“out” states for students in the training dataset. In other words, if the training dataset was made up of a set of observed “in”/“out” state sequences, O, where each observed sequence is $o \in O$, then for each o, we sum over all possible state transition sequences $q \in Q$, giving us:

$$L = -\sum_{o \in O} \sum_{q \in Q} \log(p(o, q)) \tag{2}$$

where $p(o, q)$ is computed as in equation (1). The higher the value of L, the higher the probability assigned by our model in predicting those training occurrences that actually occurred. Thus, we plot L against the choices of the decision we are to make, and select the simplest choice that yields a reasonable score. For instance, figure 2 shows an example of this procedure for a HMM that uses only 1 additional feature to the “in”/“out” state of a student, where we are trying to decide a good number of hidden states, N, for the model. Here, since it is simple enough to generate the full model, we also compare the binary classifcation metric of Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) plot.

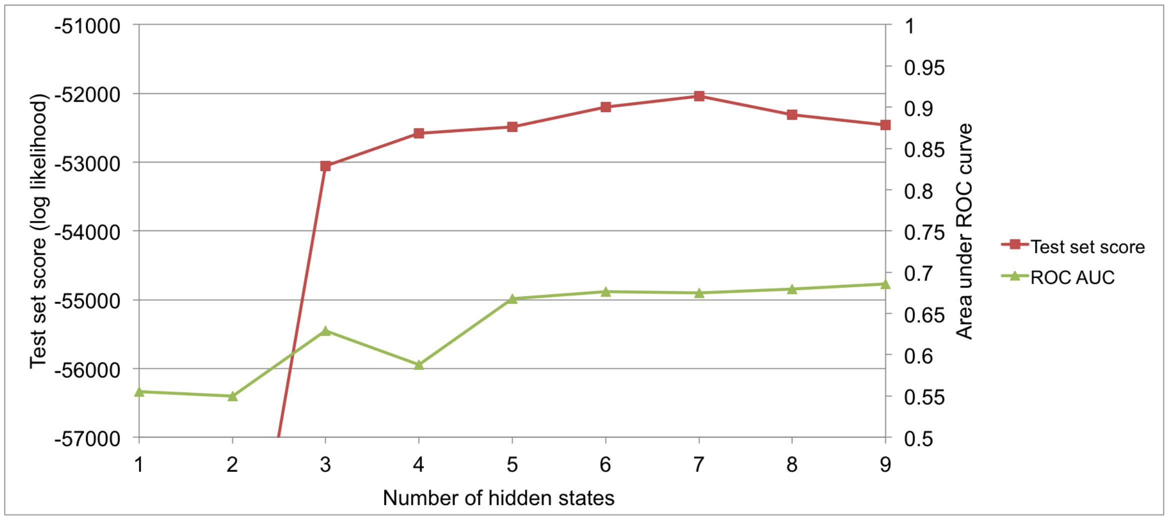


Figure 2: L score and AUC against the number of hidden states, N, for a HMM that uses one additional feature to the “in”/“out” state of a student. For this model, 7 hidden states were chosen.

4.2 HMMs with a Single Feature and the “in”/“out” State

The simplest of our models are HMMs that incorporate F_0, the “in”/“out” state, with a single other feature, F_i. These models allow us to understand the relationship between individual features and student retention. If F_i is continuous, then it is first discretized using one of the following approaches:

- k-means clustering - useful for clustering students into groups of similar behavior
- Splitting the values into quartiles - useful for features encoded as percentages
- Making the feature binary, i.e. 0 if the feature’s value is 0 and 1 otherwise - useful for those features where simply performing an action is a good indicator of the student’s interaction with the feature

The exact choice of approach depends on what is most apt for each feature, and we compare choices by computing scores as explained in section 4.1. Once we discretize F_i, we obtain a new variable $F_{i,D}$ that can undertake one of the values from a set of discrete values $D = \{D_0, D_1, ...\}$. Then, we combine F_0 with $F_{i,D}$ by computing the cross-product of D with the possible values that F_0 can take, $\{$“in”, “out”$\}$. This yields a single observation variable with states $\{(D_i, \text{“in”}), (D_i, \text{“out”})\}$ for all $i \in [0, |D|)$.

Now, we can use the Baum-Welch algorithm [6][7] to train the probability parameters in λ. The data is divided evenly into a training and test set, and only training data is used. Training also requires us to to specify N, the number of hidden states. Since we are not interested in attaching meaning to these hidden states (although some meaning can be inferred), this becomes a tunable parameter that we select using the approach in section 4.1.

The model is then evaluated using the test set, S_T, which contains students and their associated sequence of feature vectors. In the case of individual HMMs, the feature vectors only contain two elements, one for F_0 and the other for F_i, but this is extended to multiple features for our composite models. Thus, every student $s \in S_T$ would have the feature vector sequence $(f_{s1}, f_{s2}, ..., f_{s6})$, where f_{sw} is the feature vector for student s for week w. There are at maximum 6 entries because there are 6 weeks in the course. First, we compute all the subsequences of these feature vector sequences that start at the first week and end at the week that the student drops the course. We dont consider the weeks after they drop the course since the student cant rejoin the course so their value for F_0 cannot change. Call this set of subsequences Z. We then remove feature F_0 from every $f_{sw} \in z \in Z$ and keep them separate, since this is the feature we are trying to predict. Then, we discretize all feature vectors using the approach selected during training, and combine the discretized feature vectors with the “in”/“out” state of the student at that week to obtain all the observation states. Let this final set of observation sequences be Z_O, then a subsequence $z \in Z_O$ would contain observation states of the form $(*, F_0)$ where * denotes the discretized value of the feature vector and $F_0 = \{\text{“in”}, \text{“out”}\}$. With this setup, we can question what the likelihood of the student staying in the course in the next time slice, $t+1$, is given their observation subsequence, z, and hidden state sequence, q, until time t. This probability can be expressed as:

$$p(F_{0,t+1} = \text{“in”} | z, q) = \frac{p(F = \text{“in”} \cap z \cap q)}{p(z, q)} \tag{3}$$

The joint probability $p(z, q)$ is identical to that expressed in (1), and is easily computed. Defining the set of all possible state transitions as $Q = H \times H$ such that a $q \in Q$ is a state transition of the form (H_i, H_j), we can express the probability of the student staying “in” for the next time slice as:

$$p(F_{0,t+1} = \text{“in”}) = \sum_{(H_i, H_j) \in Q} \sum_{k=1}^{K} p(o_{t+1} = (k, \text{“in”}) \mid q_{t+1} = H_i) \cdot p(q_{t+1} = H_i \mid q_t = H_j) \tag{4}$$

where the two probabilities come directly from the transition and emission matrices A and B. In this way, we come up with percentages for the student remaining “in” the course, given our observations thus far. Drawing a threshold at 50%, we can then classify whether our model predicts that the student will stay or drop in the next time slice, and compare the models answer with the actual answer, which we have beforehand, to evaluate the model’s effectiveness.

4.3 HMM with Multiple Features Using Stacking

While the HMMs that incorporate a single feature help understand patterns in student behavior with respect to a single feature, we can improve on the predictions made by these models by instead building a composite model that incorporates multiple features. One simple approach to achieving this is to separately train individual HMMs (as in section 4.2) for each feature we would like to consider, and employ a stacking ensemble method, where the prediction probabilities from individual HMMs are fed into an off-the-shelf logistic regressor, which assigns weights to each of the feature models and computes an overall prediction probability for the student’s rention in the next time step.

Specifically, since we are trying to predict student retention in the next time step, we compute $P_{F_0,i} = p(F_{0,t+1} = \text{“in”} | z_i, q_i)$ for each HMM, where $1 \leq i \leq |F|$. This value is the ith HMMs likelihood value for a student staying in the course in the next time slice, $t+1$, given their observation subsequence for the ith HMM, z_i, and hidden state sequence for the ith HMM, q_i, until time t. The equations used to compute these values are identical to (3) and (4). The logistic regressor has $|F|$ features, one for each HMM, and the $P_{F_0,i}$ values are the feature values for the logistic regressor. This allows us to train the regressor to classify the student as being “in” or “out” in the next time step.

To evaluate this model, for every student in the test set, each individual HMM will produce values for $P_{F_0,i} = p(F_{0,t+1} = \text{“in”} | z_i, q_i)$. Then, we simply feed all these $P_{F_0,i}$ to the trained logistic regressor, which classifies whether the student will be retained in the next time step. The stacking approach tends to yield better predictions than individual HMMs, as shown in section 5 while also being very scalable with the number of features incorporated.

4.4 HMM with Multiple Features Using K-Means Clustering and Cross-Product Discretization

In this approach to building a composite model, we define a feature matrix, C, with dimensions $|S|W \times |F|$, where S is the set of students in the training set, W is the number of weeks in the course, and F is the featureset. Further, $C = \{c_{swj}\}$ where c_{swj} is the value of feature F_j for student s during week w. We first remove feature F_0, since that is the feature we are trying to predict and will need to be able to control which observation states correspond to the "in" state vs. the "out" state. Then, we transform this sub-matrix, $C' \subset C$, into a single variable using k-means clustering. Inputting C' into the clustering algorithm returns K clusters as well as a mapping from any feature vector $f = \{f_1, f_2, ...\}$ to one of the K clusters. These K clusters are combined with each of the two outcomes of F_0, resulting in our M observable states, i.e. $M = 2K$. Also, the mapping from the clustering is used in the testing phase to obtain the HMM's observation states from actual recorded sequence of features as exhibited by the students in our test set. K is a parameter we tune using the procedure in section 4.1.

Because k-means clustering can make poor choices for certain distributions of feature vectors, we also explored a simpler discretization in which we select an ad hoc discretization for each feature individually (as in the discretization in section 4.2), and then combine these individual discrete set of values using a simple cross-product, yielding a total number of observed states equal to the product of the number of discrete states for each feature. This tends to yield better predictions, but has problems with scaling due to the resulting very large number of observed states, most of which are not encountered in training. Because several ad hoc discretizations of the same feature are possible, we select the best one by building various HMMs using each discretization and comparing their prediction performance.

The evaluation of these models is akin to the evaluation procedure oulined in section 4.2.

5 Results

Our objective, as stated in the introduction, is twofold: to identify defining patterns in student behavior with regards to their interaction with the MOOC and their propensity to stay in the course, and to predict whether a student is likely to stay in the course for the next week. Thus, we present the results to both objectives as separate sections.

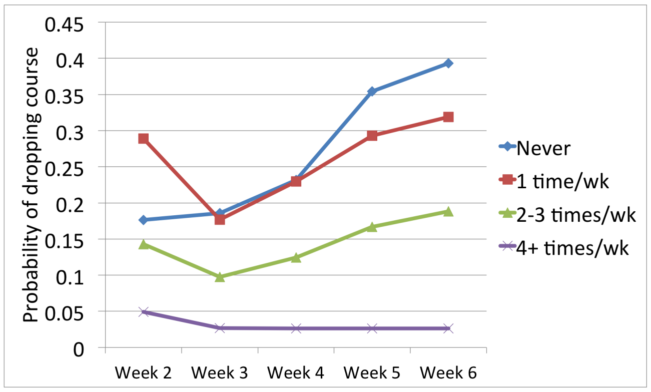
5.1 Patterns in Student Behavior

First, we examine some of the interesting individual HMMs that use single features in combination with the "in"/"out" states. To generate these predictions, we predict the likelihood that a student will reach a certain week in the course without dropping while exhibiting a certain behavior pattern, and then compare it to the probability that they will reach that point while exhibiting that behavior and then drop the next week. This allows us to examine the impact of particular behaviors across the duration of the course.

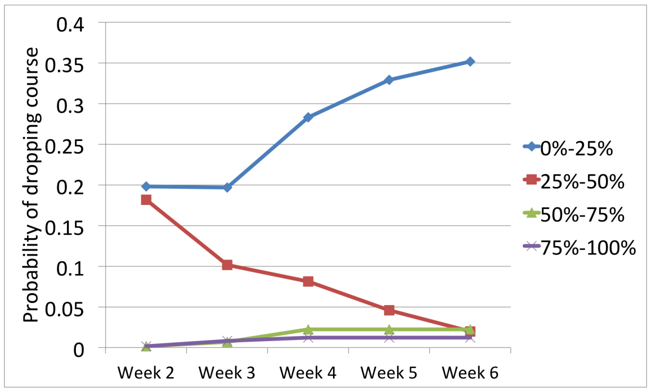
5.1.1 Patterns with regards to the frequency with which a student checked their course progress

Figure 3a shows the relationship between students who consistently check their progress a certain number of times a week, and their likelihood to drop the course in the next week. Students who rarely or never check their progress drop the most frequently in early weeks, whereas in late weeks students who never check their progress drop far more frequently, with drop rates approaching 40%. This can be explained by the fact that if a student has not checked their progress by weeks 4 and 5, by which time a couple of graded assignments are due, it is more likely that they havent even attempted those assignments, and thus are inclined to drop the course.

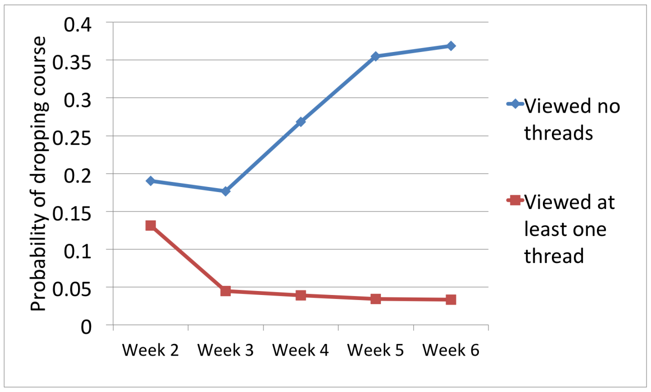
Unsurprisingly, those students who consistently check their progress 4 or more times a week have very low likelihoods of dropping the course ($\sim$2%).



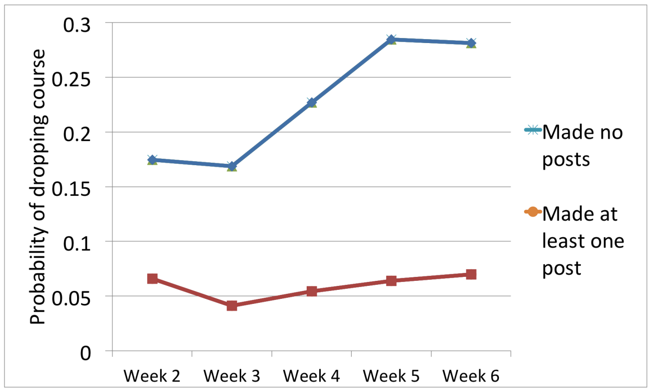
(a)



(b)



(c)



(d)

Figure 3: Attrition with time for students who:
(a) check their course progress with a consistent frequency every week
(b) view a consistent percentage of lecture videos each week, view a consistent percentage of lecture videos each week
(c) consistently view or do not view forum threads each week
(d) consistently post once per week or consistently do not post on the forum
As an example, the green line in (a) shows the probability that a student who checks their course progress 2-3 times a week will dropout of the course each week. So, if a student maintained such consistent behavior for the first 4 weeks, their likelihood of dropping the course in week 5 is about 17%.

5.1.2 Patterns with regards to the cumulative percentage of lecture videos watched

Figure 3b shows the relationship between students who consistently view a certain percentage of lecture minutes each week, and their likelihood to drop the course in the next week. As would be expected, students who watch more of the lectures are less likely to drop - students who watch at least 50% of lecture minutes are extremely unlikely to drop (<0.1% throughout). Interestingly, as the course continues, students who watch no lectures become more likely to drop, exceeding a 35% drop rate, and students who watch only 25%-50% of lectures become very unlikely to drop, reaching 2.2%. This suggests that watching lectures is important, but watching them in their entirety becomes less important toward the end of the course.

5.1.3 Patterns with regards to the number of forum threads viewed

Figure 3c shows the relationship between whether students view forum threads each week, and their likelihood to drop the course in the next week. In the first week the difference is relatively small, but towards the end of the course students who never view the forum become very likely to drop (37%), whereas those who view at least one thread a week are very unlikely to drop (4%). This suggests even minimal interaction with the forum can be a crucial factor for retention.

5.1.4 Patterns with regards to the number of forum posts made

Figure 3d shows the relationship between whether students post forum threads each week, and their likelihood to drop the course in the next week. Students who post on a weekly basis are very unlikely to drop (consistently in the rage of 4-7%). The probability of dropping for students who do not post is lower for this feature than for others considered above, even in week 6 (about 28% as compared to 35-38%), consistent with the notion that students can participate actively in the course without actively contributing to the forum.

5.2 Immediate Student Retention Predictions

We explored individual HMMs that incorporate a single feature as well as two alternatives for composite HMMs, all of which allowed us to classify students as being in or out of the course at every time step subsequent to the first week of the course. For every student in the test set, we examine all subsequences of actions leading up to the point at which they dropped, and predict whether they will drop in the next timestep (we do not examine subsequences following the drop point because these are trivial to predict - students who are dropped remain dropped).

We evaluate our predictions using standard binary classification metrics, such as precision, recall, and F1 scores, as well as the Area Under the Curve (AUC) score for the Receiver Operating Characteristic (ROC) plot, and the Matthews correlation coefficient. In figure 4, we show the ROC plots for the best composite model, in table 1 we summarize the results for both composite models, and in table 2 we summarize the AUC scores for the individual HMMs.

	Composite model using cross-product discretization	Composite model using stacking
Accuracy	0.801	0.805
Matthews Correlation Coefficient	0.137	0.119
ROC AUC Score	0.710	0.696
Precision	Positive: 0.807 Negative: 0.558	Positive: 0.807 Negative: 0.647
Recall	Positive: 0.987 Negative: 0.064	Positive: 0.995 Negative: 0.036
F_1 Score	Positive: 0.888 Negative: 0.115	Positive: 0.891 Negative: 0.068

Table 1: Summary of binary classification evaluation metrics for the composite HMMs. For precision, recall and F1 scores, positive queries are where we ask if a student is staying in the course, and negative queries are for students dropping out.

Feature considered by individual HMM	AUC Score
Number of times the course progress page was visited	0.692
Cumulative percentage of lecture videos watched	0.656
Number of threads viewed	0.657
Number of posts made	0.609

Table 2: Summary of AUC Scores for the different individual HMMs. Each individual HMM is identified by the single feature, in addition to the "in"/"out" state, that is incorporated into the model.

In general, the precision, recall and F-1 values show that our model is relatively poor at predicting negative queries, although this may be mostly due to the fact that there are many more positive queries than negative queries, since a sequence of "in"/"out" states for a given student contains several "in" states (positive queries), but only one "out" state (negative

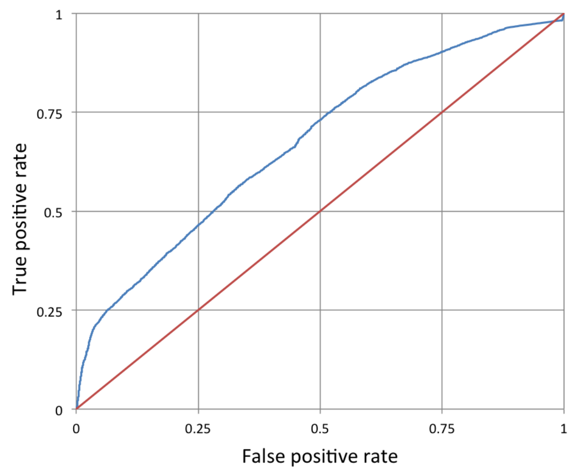


Figure 4: ROC Plot for composite HMM that employs the cross-product discretization method. The line through the center of the plot has area under the curve of 0.5, and denotes a classifier that is no better than random.

queries). As a result, it may not be too instructive to pay attention to these figures. A much better metric is the ROC curve, which indicates that our best classifier is quite a bit better than random, with an AUC score of 0.710.

Both composite models exhibit higher ROC AUCs than the HMMs using single features (comparing table 1 and table 2), suggesting that both techniques somewhat effectively incorporate information from multiple features. The relatively small gain suggests considerable dependency between features.

The cross-product discretization method yielded higher overall ROC AUC than the stacking method. This can be explained in part by the HMMs ability to perform state transitions based on combinations of multiple feature values. However, this advantage is offset by a much longer training time due to the large size of the observed state space.

6 Conclusion

Overall, we were able to design effective Hidden Markov Models to help predict student retention as well as to infer general patterns of behavior between those students that complete the course, and those that drop at different points in time. Our composite HMM that incorporated multiple features produced a reasonable ROC curve with an AUC value of 0.710. Lastly, our individual HMMs offered insight into certain patterns of student behavior, such as the fact that a student who never checks their course progress dramatically increases their probability of dropping out of the class only after the fourth week of the course. While this is purely correlational, it does offer some interesting insight into how students interact with the MOOC and can be used to suggest behavior changes to students that are headed towards dropping the course.

7 Future Work

The most obvious extension to our current methods is the inclusion of more features from the MOOC to enrich our composite model, as well as bring to light more insightful patterns in student behavior. Some of these features include:

- Scores received on homeworks/quizzes
- Number of posts followed and/or upvoted on the forum
- Number of replies or upvotes received on posts made
- Percentage of only this week's lecture videos watched

In addition, it would be instructive to model the course using Kalman filters, as opposed to our current approach of quantizing a multidimensional feature matrix of continuous values. This would preserve some of nuances in the data that is lost in the current process, and may yield better predictions.

Finally, one could explore alternative definition of what it means for a student to be an active participant of the course. Currently, our model does not take into account complex patterns of behavior, such as those students who leave the course for one or two weeks but come back to finish the course. A definition that incorporates these subtleties would enable us to gauge how good the adaptable the course is to serve such individuals.

Tools Used

- **Python** - The primary language used for feature extraction, and model creation as well as inference.
- **GHMM Library** - General Hidden Markov Model LIbrary implemented in C with a Python interface, used for creating our models (http://ghmm.org/).
- **Scikit-Learn** - Python machine learning library, used for k-means clustering (http://scikit-learn.org/). [8]

References

[1] V. Tinto, "Dropout from Higher Education: A Theoretical Synthesis of Recent Research". *Review of Educational Research*, vol. 45, no. 1, pp. 89-125, 1975.

[2] S. L. DesJardines, D. A. Ahlburg, B. P. McCali "An event history model of student departure". *Economics of Education Review*, vol. 18, issue 3, pp. 375-390, 1999.

[3] Y. Belanger, J. Thorton, "Bioelectricity: A Quantitative Approach - Duke Universitys First MOOC", *Duke University Report*, February 2003.

[4] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", in *Proceedings of the IEEE*, 77(2):257-285, February 1989.

[5] J. Schenk, S. Schwarzler, G. Rigoll, "Discrete Single Vs. Multiple Stream HMMs: A Comparative Evaluation of Their Use in On-Line Handwriting Recognition of Whiteboard Notes", in *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, August 2008.

[6] J. A. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models". *Technical Report TR-97-021*, International Computer Science Institute, Berkeley, CA., 1998.

[7] L. Baum, T. Petrie, "Statistical Inference for Probabilistic Functions of Finite State Markov Chains". *Annals of Mathematical Statistics*, 37:1554-1563, 1966.

[8] Pedregosa et al., "Scikit-learn: Machine Learning in Python", JMLR 12, pp. 2825-2830, 2011.