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

Girish Balakrishnan {guru_g@berkeley.edu}
Derrick Coetzee {dcoetzee@eecs.berkeley.edu}

Abstract

Massive Open Online Courses (MOOCs) have a high attrition rate: most students who register for a course do not complete it. By examining a student's history of actions during a course, we can predict whether or not they will drop out in the next week, facilitating interventions to improve retention. We compare predictions resulting from several modeling techniques and several features based on different student behaviors. Our best predictor uses a Hidden Markov Model (HMM) to model sequences of student actions over time, and encodes several continuous features into a single discrete observable state using a simple cross-product method. It yielded an ROC AUC (Receiver Operating Characteristic Area Under the Curve score) of 0.710, considerably better than a random predictor. We also use simpler HMM models to derive information about which student behaviors are most salient in determining student retention.

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.

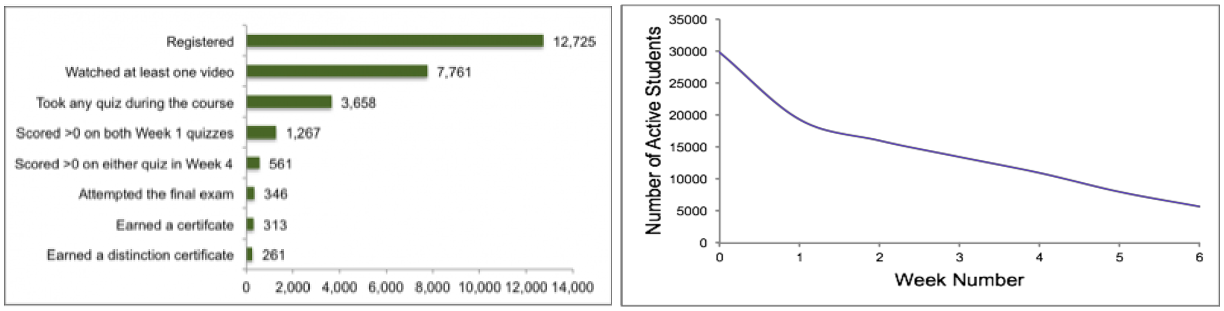

Figure 1 Student Persistence in (left) *Bioelectricity, Fall 2012* (Duke University MOOC) [7], and in (right) *Software as a Service, Fall 2012* (Berkeley edX's MOOC). In both cases, students drop the course every week.

Here we attempt to do the same for students in an online setting at a much larger scale, using Hidden Markov Models (HMMs, [3]) 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 [4]. Using this technique, we can then apply the *Baum-Welch* algorithm [6] to train our HMM on a chosen number of hidden states. In our second method, we use an 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 can we suggest interventions to prevent this from occurring?

2. Methods

2.1 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

It's 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 student's 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.

2.2 Feature set

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 more than 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, the entire feature-matrix would have dimensions $|S|W \times |F|$.

Next, we present the features we extracted from the available data, grouped by type.

2.2.1 Student "In"/"Out" State

Our primary feature is the one we eventually want to be able to predict, which is whether a student has dropped the course. This 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. 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.

2.2.2 Lecture Interaction Features

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 in the current week, 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, we consider the actual amount of time that was spent on watching lectures in a given week. The features extracted are:

1. Total time (in seconds) spent on watching lectures this week, including any of the available lectures in the course so far. This is the most coarse feature for lecture videos and indicates the amount of effort that a student put into the course that week on learning the material.
2. Percentage of lecture minutes released this week that were watched this week by the student. While this feature is a poor indicator of a student's coverage of material, it is a good measure of how well the student is keeping up with the pace of the course, since a student who watches this week's lectures presumably has understanding of the course so far.
3. Cumulative percentage of available lectures watched from the beginning of class until the end of the current week. This metric indirectly measures how far behind a student is in the course, as a cumulative percentage that significantly decreases from week to week highlights a student that is falling behind.

2.2.3 Forum Interaction Features

The forum is the primary means of student support and interaction during the course. The forum's basic software mechanisms allow us to observe the following useful features:

1. Number of threads viewed this week, where a thread can only be viewed once a day. Since most active students undertake this passive interaction it is an important metric of engagement.
2. Number of threads followed this week, which is a slightly more active sign of engagement than 1.
3. Number of upvotes given this week, indicating posts students found to be useful, which is also a more active sign of engagement.
4. Number of posts made this week. Although most students aren't active on the forum, for those who are this feature is a strong indicator of engagement and sense of community.
5. Number of replies received this week to any post previously made. This is very important as it directly correlates with how much belonging a student feels in the course.
6. Number of upvotes received this week to any post previously made. This is important for the same reasons as 5.

2.2.4 Assignment Features

Students are exposed to ungraded lecture problems that are intertwined with lecture videos, as well as graded quizzes and homeworks that assess their understanding of the material. Graded assignments are conveniently due at the end of a week. Since these types of problems carry very different weights, we define them individually as follows:

1. Cumulative percentage score on homework problems that are due at the end of this week, or have been due in previous weeks. When monitoring this value from week to week, we again get a good gauge on how far up-to-date a student is on the course.
2. Cumulative percentage score on quiz problems that are due at the end of this week, or have been due in previous weeks.
3. Cumulative percentage score on lecture problems that are available from the start of the course until this week. The difference between this and features 1 and 2 is that there is no due date for lecture problems, so a student actually has the possibility to catch up on them at any point in the course.
4. Percentage score on homework problems that are only due this week. The score that a student

receives in the current week could be a good indicator of how motivated the student is to continue into the next week. For the weeks where there are no homeworks due, students get 100%.

5. Percentage score on quiz problems that are only due this week, for the same reasons as 4.
6. Number of times a student checks their course progress, which contains information about their status on successfully completing the course. It is reasonable to hypothesize that an active and engaged student would monitor their progress a few times every week, particularly as they complete lectures and assignments.

In features 1-3, we look for the actual number of problems gotten correct by the student, but it is also instructive to simply check how many of the problems were attempted, since that is a better indicator of the effort put in by the student. Consequently, we define three other features that are identical to features 1-3, except that they are cumulative percentages of problems attempted as opposed to problems gotten correct. In total, we have 9 features related to course assignments.

Despite extracting many features, we only selected a few of these features to build a composite model. We ensured that the selected features span the different aspects of the MOOC to limit dependence between features. We chose only the following four features in addition to our primary feature of students' "in"/"out" state:

- Number of time a student checks their course progress
- Cumulative percentage of available lectures watched from the beginning of class
- Number of threads viewed this week
- Number of posts made this week

2.3 Modeling Technique

A discrete single stream HMM [4] has a single discrete-valued observation variable at every time slice. Such a model is fully described by the following parameters [3]: 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 $H = \{H_1, H_2, \ldots, H_N\}$ with q_t being the hidden state at time t, and the set of observable states are denoted as $V = \{V_1, V_2, \ldots, V_M\}$ with o_t being the observable state at time t.

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.

With such a HMM, the probability of a sequence of length T, with the observation state sequence $o = (o_1, o_2, \ldots, o_T)$, and hidden state sequence $q = (q_1, q_2, \ldots, q_T)$ is given by:

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

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, we have several features, $F = \{F_0, F_1, F_2, ...\}$, and all of them, save the "in"/"out" feature which we label as F_0, are continuous. The above model admits only a single, discrete observable state.
Thus, we explore two alternative methods to incorporate these multiple continuous-valued features into a single model that allows us to predict student retention in the next time step of the course.

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

In this approach, 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, which is the "in"/"out" feature, 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 to achieve as effective a model as possible, as defined by the testing procedure outlined later in this section.

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 (based on ranges), and then combine these individual discretizations using a simple cross-product, yielding a total number of observed states equal to the product of the number of 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.

Having defined the model parameters, we use the Baum-Welch algorithm [5][6] to train the probability parameters, 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. To select a good value for N we train HMMs with different values of N and compute a score, L, per model where 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:

$$L = -\sum_{o \in O} \sum_{q \in Q} log \left[p(q_1)\, p(o_1|q_1) \cdot \prod_{t=2}^{T} p(q_t|q_{t-1})\, p(o_t|q_t) \right] \qquad (2)$$

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 for different values of N and decide on the smallest number of states, for simplicity, that yields a relatively reasonable score. 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.

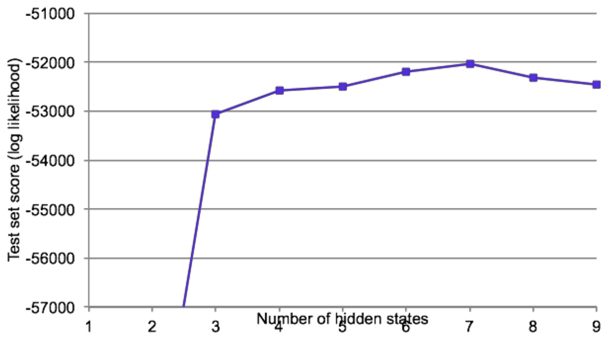

Figure 2 L score against the number of hidden states, N, for a HMM that uses one additional feature to the “in”/”out state of a student. For this model, 7 hidden states were chosen.

Then we use the test set to evaluate our model. The test set, S_T, contains students with an associated sequence of feature vectors. Thus, every student $s \in S_T$ would have the feature vector sequence $(f_{s1}, f_{s2}, \ldots, f_{s6})$, where f_{sw} is a 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 don’t consider the weeks after they drop the course since the student can’t come back in the course so their “in”/”out” state cannot change. Call this set of subsequences Z. We then remove the “in”/”out” state feature from every $f_{sw} \in z \in Z$ and keep them separate as we did during training. Then, we use the mapping from our k-means clustering to map every feature vector to a cluster label. Lastly, we form the actual observation states by combining the cluster label with the “in”/”out” state of the student at that week. Let this final set of observation sequences be Z_O, then a subsequence $z \in Z_O$ would contain observation states of the form (k, F_0) where $k \in \{1, \ldots, K\}$ and $F_0 = \{"in", "out"\}$. 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:

$$p(F_{0,t+1} = "in" \mid z, q) = \frac{p(F_{0,t+1} = "in" \cap z, q)}{p(z, q)} \tag{3}$$

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:

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

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 model’s answer with the actual answer, which we already have beforehand, to evaluate the model’s effectiveness.

2.3.2 HMM with Multiple Features Using Stacking

In this approach, we use the Baum-Welch algorithm to train multiple HMMs, where each HMM incorporates a single discretized feature in combination with F_0 , the “in”/”out” state, as the observation variable. As a specific example, if the model should consider feature F_1 , then we first discretize all the values of F_1 for every student for every week. The method of discretization is specific to what makes sense for the feature itself; we can employ *k*-means clustering, or split the feature-values into quartiles (as in the case for features that take percentage values), or simply treat the feature as a binary value (0 if the value is 0, and 1 otherwise). Then, the cross product of these quantizations with the possible outcomes of F_0 , i.e. “in” or “out”, produce all the possible observation states of the composite observation variable, F^* . Now, we can train an HMM using just F^* , and in such a way, we train separate HMMs for every feature we would like to consider.

A stacking ensemble method is then used, where we stack an off-the-shelf logistic regressor to the results of these HMMs. Specifically, since we are trying to predict student retention in the next time step, we compute $P_{F_{0,i}} = p(F_{0,t+1} = "in" \mid z_i, q_i)$ for each HMM, where $1 \leq i \leq |F|$. This value is the i ’th HMM’s likelihood value for a student staying in the course in the next time slice, $t+1$, given their observation subsequence for the i ’th HMM, z_i , and hidden state sequence for the i ’th HMM, q , 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 slice.

To evaluate this model, for every student in the test set, each individual HMM will produce values for $P_{F_{0,i}} = p(F_{0,t+1} = "in" \mid z_i, q_i)$. 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 a single HMM using k-means clustering, while also avoiding the scalability issues of cross-product discretization. We compare prediction quality of the stacking approach with these single-HMM approaches in the results section.

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

The procedures described above create models that allows us to effectively predict whether the student will drop the course in the next time step by considering multiple features. We refer to this as the composite HMM. However, to understand the impact of individual features on student retention, we also train HMMs for individual features, where a single feature, F_i , is selected and discretized in combination with the student “in”/”out” state to produce our observation variable and states. The rest of the procedure, including finding a reasonable N and predicting the student’s next “in”/”out” state remain the same as in the case of the composite HMM using k-means clustering. We refer to these HMMs as individual HMMs using a particular feature, and we use these to understand patterns in student behavior.

3. Results

In this section, we present the results obtained through the methods described in the previous section. 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.

3.1 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.

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

Figure 3 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 only check their progress occasionally drop the most frequently in early weeks, whereas in late week students who never check their progress drop 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 haven't 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 (~2%).

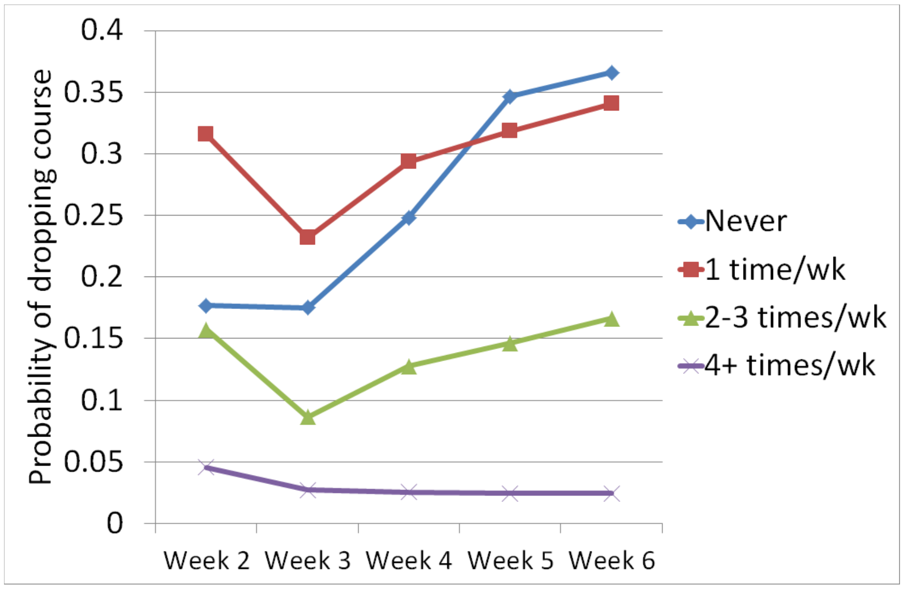

Figure 3 Attrition with time for students who check their course progress with a consistent frequency every week. As an example, if a student is active in the course up until week 4, and checks their progress 2-3 times per week during that period, their likelihood of dropping the course in week 4 is about 13%, as shown by the green line.

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

Figure 4 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.

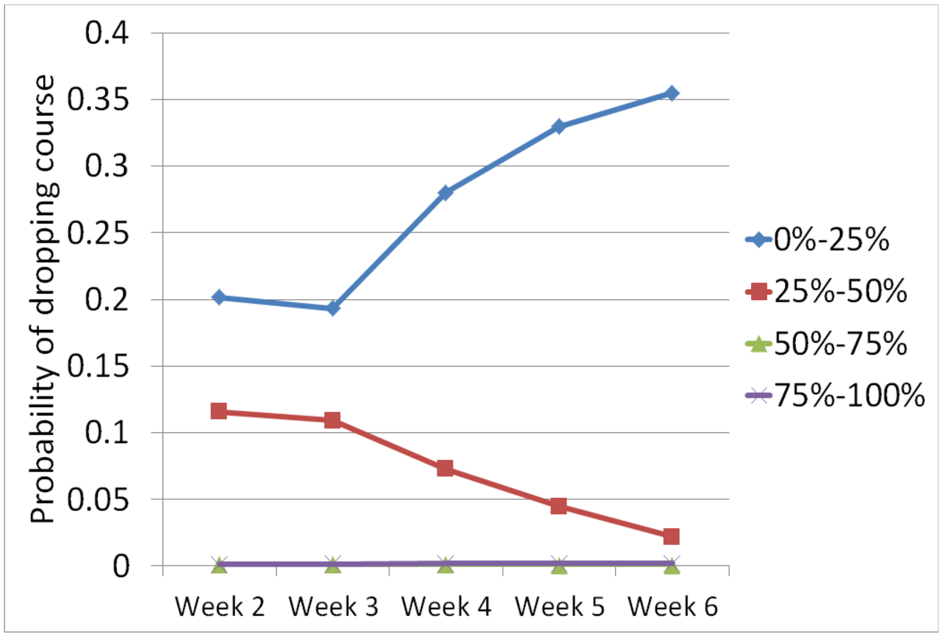

Figure 4 Attrition with time for students who view a consistent percentage of lecture videos each week. As an example, if a student is active in the course up until week 4, and views 25%-50% of lecture minutes each week during that period, their likelihood of dropping the course in week 4 is about 8%, as shown by the red line.

3.1.3 Patterns with regards to the number of threads viewed

Figure 5 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 (41%), whereas those who view at least one thread a week are very unlikely to drop (3.4%). This suggests even minimal interaction with the forum can be a crucial factor for retention.

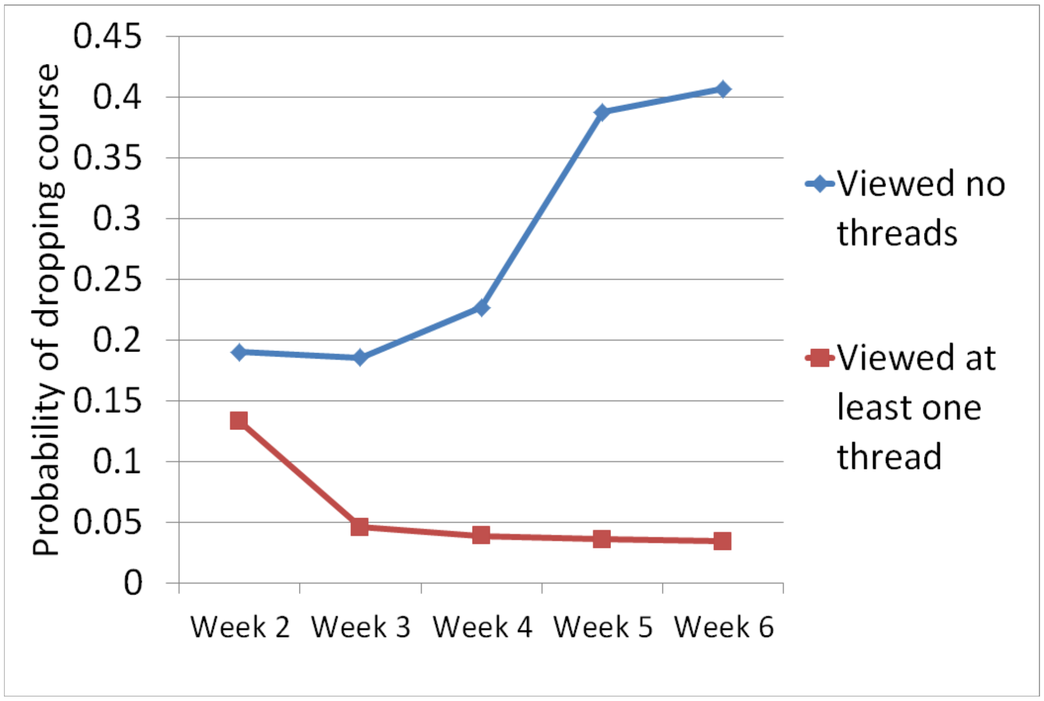

Figure 5 Attrition with time for students who consistently view or do not view forum threads each week. As an example, if a student is active in the course up until week 4, and views at least one forum thread each week, their likelihood of dropping the course in week 4 is about 4%, as shown by the red line.

3.1.4 Patterns with regards to the number of forum posts made

Figure 6 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 (about 2% from Week 3 onwards). 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 30% as compared to 35-40%), consistent with the notion that students can participate actively in the course without actively contributing to the forum.

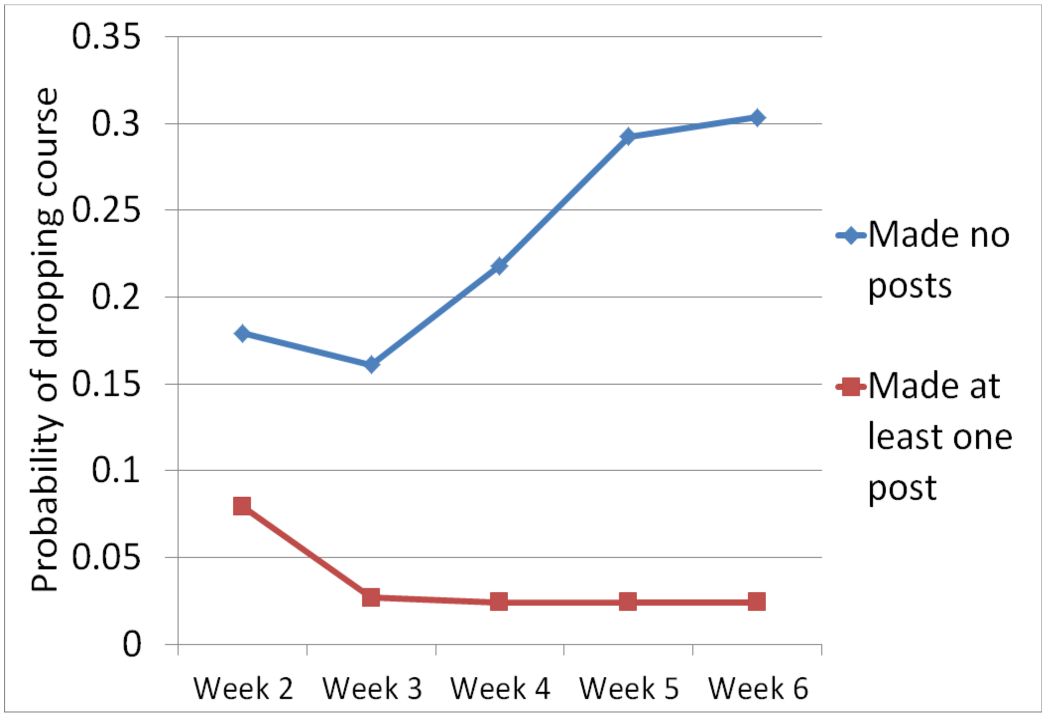

Figure 6 Attrition with time for students who consistently post once per week or consistently do not post on the forum. As an example, if a student is active in the course up until week 4, and posts at least one forum post each week, their likelihood of dropping the course in week 4 is about 2%, as shown by the red line.

3.2 Immediate Student Retention Predictions

We explored two alternatives for composite HMMs, each 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 F_1 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 composite stacking model, and in Table 1 we summarize the results for both models.

	Composite model using cross-product discretization	**Composite model using stacking**
Accuracy	0.801	0.805
Mathews Correlation Coefficient	0.137	0.119
ROC AUC Score	0.710	0.698
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 F_1 scores, positive queries are those where we ask if a student is staying "in" the course, and negative queries are for students dropping "out".

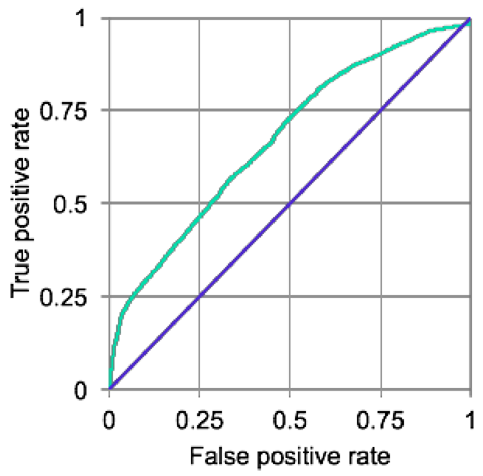

Figure 7 ROC Plot for composite HMM that employs the cross-product discretization method. The line through the center of the plot has area under the curve of 0.5, and denotes a classifier that is no better than random.

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 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 (which have ROC AUCs of 0.686, 0.668, 0.660, 0.609), suggesting that both techniques 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 HMM's 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.

4. 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 those students who seem like they are headed towards dropping the course.

5. Future Work

There are a few extensions that can be explored to improve our current methods. Firstly, 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 clustering process, and may yield better predictions.

Next, our composite model currently incorporates 4 features, but there are still many features left to incorporate, particularly with relation to students' interaction with course assignments. Incorporating these features would further enrich the composite model, and could improve the predictions made.

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.

References

[1] Tinto, V (1975) Dropout from Higher Education: A Theoretical Synthesis of Recent Research. *Review of Educational Research* Vol.45, No1, pp.89-125.
[2] DesJardines, S.L., Ahlburg, D.A., McCali, B.P. (1999) An event history model of student departure. *Economics of Education Review* Vol. 18, Issue 3, pp.375-390.
[3] Rabiner, L. (1989) A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, *Proc. of the IEEE*, 77(2):257-285.
[4] Schenk, J., Schwarzler, S., Rigoll, G. (2008) Discrete Single Vs. Multiple Stream HMMs: A Comparative Evaluation of Their Use in On-Line Handwriting Recognition of Whiteboard Notes, *Proc. of International Conference on Frontiers in Handwriting Recognition*
[5] Bilmes, J. A. (1998): 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.
[6] Baum, L., Petrie, T. (1966) Statistical Inference for Probabilistic Functions of Finite State Markov Chains, *Annals of Mathematical Statistics*, 37:1554-1563
[7] Belanger, Y., Thornton, J. (2013) Bioelectricity: A Quantitative Approach - Duke University's First MOOC, *Duke University Report*
[8] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

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]