MOOC Performance Prediction via Clickstream Data and Social Learning Networks

Christopher G. Brinton and Mung Chiang
Department of Electrical Engineering, Princeton University
{cbrinton, chiangm}@princeton.edu

Abstract—We study student performance prediction in Massive Open Online Courses (MOOCs), where the objective is to predict whether a user will be Correct on First Attempt (CFA) in answering a question. In doing so, we develop novel techniques that leverage behavioral data collected by MOOC platforms. Using video-watching clickstream data from one of our MOOCs, we first extract summary quantities (e.g., fraction played, number of pauses) for each user-video pair, and show how certain intervals/sets of values for these behaviors quantify that a pair is more likely to be CFA or not for the corresponding question. Motivated by these findings, our methods are designed to determine suitable intervals from training data and to use the corresponding success estimates as learning features in prediction algorithms. Tested against a large set of empirical data, we find that our schemes outperform standard algorithms (i.e., without behavioral data) for all datasets and metrics tested. Moreover, the improvement is particularly pronounced when considering the first few course weeks, demonstrating the “early detection” capability of such clickstream data. We also discuss how CFA prediction can be used to depict graphs of the Social Learning Network (SLN) of students, which can help instructors manage courses more effectively.

I. INTRODUCTION

In the past few years, Massive Open Online Courses (MOOCs) have drastically risen in popularity, creating global connectivity among students of unprecedented size and diversity. Platforms such as Coursera, edX, and Udacity have offered courses with enrollments reaching hundreds of thousands, and have become subjects of intensive debate [1].

Policy and business issues aside, with MOOC teacher-to-student ratios at fractions of one percent [2], there is one technology advance that will be critical to the efficacy of learning at this scale: automated mechanisms to assist an instructor in enhancing the learning experience [3]. To this end, our recent work [4] pointed out a number of research avenues pertaining to the Social Learning Network (SLN) of MOOC, which is a type of social network between students, instructors, and modules of learning. These include recommendation, personalization, and prediction such as of performance or participation dropoff rates. They arise in part due to the various learning modes available to students on these platforms: video lectures, assessments, and discussion forums. Data about student behavior with each of these modes can be analyzed to help investigate the research areas.

We investigate two research questions for MOOC:

• Q1: Is it possible to correlate student performance on assessments with their video-watching behavior?

• Q2: Can we use student behavior to predict their performance better than without it?

To investigate these questions, we will use data from one of our own MOOC offerings on Coursera [5]. We focus on user (student) video-watching behavior and in-video quiz performance, which were the most abundant types of data collected, consisting of over 1.3M clickstream logs of user interaction with the video player and over 40K quiz submissions.

Organizational. In Sec. II and III, we will focus on Q1 through statistical analysis of the video-watching and performance data. In doing so, we will identify how certain watching characteristics are indicative of whether a user is more likely to be Correct on First Attempt (CFA) or not at answering a question. Then, in Sec. IV, we will turn to Q2 and design a scheme that estimates CFA probabilities from these characteristics and uses them as learning features for prediction. For comparison, we also present some standard algorithms that have been employed for CFA prediction using only performance data. Finally, in Sec. V, we evaluate these methods, where we will see that our scheme consistently outperforms the standard ones in different scenarios, and that the incremental gain is particularly high early in the course when there is little information about each user. We will also highlight how our findings can be useful in defining SLN graphs that can assist a course instructor in clustering similar students together, in recommending study partners, and in early detection of students who may benefit from remedial help.

Related work. Various works have studied performance prediction in traditional education settings. There is a long line of work on Item Response Theory (IRT), which seeks to probabilistically estimate the response that a particular examinee will provide to a particular item [6]. More recently, research has focused on developing predictors for whether a user will be CFA or not on a question. Collaborative filtering (CoF) algorithms have been applied as classification models in this setting; memory-based CoF, such as neighborhood methods, have been used [7], but model-based CoF, such as latent semantic analysis and matrix factorization, are perhaps the most widespread (e.g., [8]–[10]). One reason for the popularity of CoF techniques in this context is its inspiration from the Netflix Prize competition [7], where CoF methods were seen to perform quite well. For Netflix Prize, the dataset consisted of users, movies, ratings between 1 and 5 for some (about 1%) of the user-movie pairs, and timestamps on the rating submissions [11]. For CFA prediction, there are also unknown entries but in
practice many less since assignments are typically compulsory, and the target is binary rather than discrete.

Beyond CoF, other works for performance prediction have applied probabilistic graphical models (PGMs) such as Hidden Markov Models (HMMs) and Bayesian networks [12, 13], decision tree classifiers [14], and factorization machines (FM) [15], typically when there is additional, coarse-granular information collected (e.g., course difficulty, time spent answering questions, age range) about users and/or courses over multiple sessions. In answering Q1 and Q2, we focus instead on relating a type of behavioral data – video-watching behavior – to performance, for users within a single course.

There has been a lack of work studying performance prediction for MOOC. The problem can be more difficult in this setting because though there are many more users than in a classroom, the fraction of assessments a user completes can be much less due to participation dropoff over time [2]. Beyond performance prediction, some recent studies on MOOC have analyzed behavioral data. For example, in terms of discussion forums, in another work [2] we analyzed the decline of participation in 73 MOOC courses over time. As for video-watching data, [16] looked at which characteristics of lecture videos contribute to peaks in watching behavior and dropoff rates.

**Contribution.** We discover video-watching behavioral quantities that are correlated with student performance, and show that they can be used to enhance CFA prediction. Additionally, we identify the “early detection” capability of clickstream data, showing that the incremental improvement is higher in the first few course weeks. Moreover, this work is the first to study CFA prediction in the context of MOOC. Each of these are important steps in studying the SLN of MOOC users.

## II. Course Description and Basic Statistics

### A. Course Format

We have instructed two MOOCs on Coursera over multiple offerings. The first offering of *Networks: Friends, Money, and Bytes* (N:FMB) [5] in fall 2012 is well-suited for exploration of our research questions because of its structure, as follows.

There were a total of 20 lectures, roughly two per week over 12 weeks. Each lecture was composed of a number of videos; the majority (12/20) had 5, while most others (7) had 4 and one had 6, for a total of 93 videos. Over these 93, the average length was 16.89 min (standard deviation (SD) = 5.96). To supplement the videos, we created in-video quizzes to test student understanding with the material, each in machine-graded, radio-button response format with four choices each. In doing so, we asked exactly one question at the end of each video, in such a way that each question was testing material limited to, and encapsulating the majority of, the video. As a result, we effectively had a 1:1 correspondence between videos and quizzes. \(^1\) But though this 1:1 relationship is convenient,

---

1. Besides in-video quizzes, there were other forms of assessment for our course: exams and homeworks. We do not focus on those here, because the in-video quizzes received a much larger number of submissions than these other assessments, since no certificate was allowed online by our institution.

---

\(^2\)The choice of 20 questions is somewhat arbitrary, serving to show that there are qualitative differences between different subsets of the data here and in Sec. V. Varying it was not seen to affect our conclusions.
this, at the bottom we plot the average score across users with at least 20 quizzes, and the SD drops substantially (0.24).

In Fig. 2(b), we plot the average score over quizzes. The top shows all 92 videos; the mean is 0.615 (SD = 0.173). The bottom considers only the beginning 20 videos, for which the distribution has a higher mean of 0.657 (SD = 0.141).

2) Clickstream logs: We also obtained an export of the video-watching clickstream data collected by Coursera, which log user interaction with the video player. Each time an event – play, pause, rate change, or seek – is fired, a data entry is recorded that specifies the user and video IDs, event type, playback position, playback speed, and UNIX time.

In total, there were 1,322,243 clickstream logs, with 122,533 user-video pairs. But we removed all pairs which did not have both a quiz submission and at least one clickstream log recorded, bringing the total to 38,703. Then, we removed 9,134 pairs that had at least one null entry. From the remainder, we discounted all 3,346 entries that were either stall or error events. In the end, we were left with 26,223 pairs.

Video-watching quantities. For each of these pairs, we computed 9 summary quantities (behaviors) of interest:

1. Fraction spent (fracSpent): The fraction of (real) time the user spent playing the video, relative to its length.
2. Fraction completed (fracComp): The percentage of the video that the user played, not counting repeated play position intervals; hence, it must be between 0 and 1.
3. Fraction played (fracPlayed): The amount of the video that the user paused the video.
4. Number of pauses (numPaused): The number of times the user paused the video.
5. Fraction paused (fracPaused): The fraction of time the user spent paused on the video, relative to its length.
6. Average playback rate (avgPBR): The time-average of the playback rates selected by the user. The player on Coursera allows rates between 0.75x and 2.0x the default speed.
7. Standard deviation of playback rate (stdPBR): The standard deviation of the playback rates selected over time.
8. Number of rewinds (numRWS): The number of times the user jumped backward in the video.
9. Number of fast forwards (numFFs): The number of times the user jumped forward in the video.

We will now study how these quantities vary between CFA and non-CFA instances. Then, we will use those findings to motivate the design of CFA prediction scheme (Sec. IV-C). Note that these quantities are not independent of each other, but each will tell us something different about user behavior.

III. CLICKSTREAM DATA ANALYSIS

For each quantity, we perform two groups of analysis. First, we examine where the probability density lies, and determine whether there is an overall difference in the distributions for the CFA and non-CFA classes. Since Shapiro-Wilk tests detected significant departures from normality for the distributions, we ran the non-parametric Wilcoxon Rank Sum test [17] for the null hypothesis that there is no difference between the classes overall. We will report the p-value (pw) from this test, and when it is low enough (below 0.05), we may reject the null hypothesis and assume the difference is significant.

Second, we consider whether there are certain intervals or sets of values that indicate a higher likelihood of being in one of the classes. We identified the potential intervals by visually analyzing the probability density of the two classes; for continuous quantities 1-3 and 5 shown in Fig. 3, we used Gaussian Kernel Density Estimation (formalized in Sec. IV-C) [18] with a bandwidth parameter η stated in each case. For each of the intervals, we run a two-sample test for proportions [17] for the null hypothesis that there is no difference between the fraction of CFA and non-CFA samples occurring there, relative to the totals for each class. If the p-value from this test is low, then there is a large enough difference between the fractions and a large enough sample size in the interval to assume that the CFA probability estimate ρ is significant. For these cases, we report the p-value, ρ̂, and a 95% Confidence Interval (CI) around ρ̂, all of which are tabulated in Fig. 4.

For this analysis, we consider all videos, but only the active users who answered at least 20 questions, for which there are roughly 9.2K CFA and 4.7K non-CFA samples.

A. Statistical Analysis

Playing behavior. This corresponds to Quantities 1 to 3. fracSpent: Much of the density is in [0.9, 1.1] (40% of CFA, 42% of non-CFA). The mean for CFA is 0.82 (SD = 0.36), compared to 0.78 (SD = 0.39) for non-CFA. This indicates that, as expected, a user submitting a correct answer tends to have spent more time with the video. The difference between the distributions is significant (pW = 4.7e-8). As shown in Fig. 3(a), we identified three intervals of interest: [0, 0.54] for which there is more non-CFA density, and [0.54, 0.90] and [1.1, 2.0] with more CFA, giving ρ̂ of 0.45, 0.52, and 0.51.

fracComp: Here, much density lies in [0.95, 1] (57% of both classes). The mean for CFA is 0.76 (SD = 0.35), as opposed to 0.74 (SD = 0.37) for non-CFA. The difference between the

3In other words, a significant p-value tells us we can trust ρ̂. Then, if ρ̂ is above 0.5, ρ̂ - 0.5 tells us how much more likely CFA is in that interval; if below 0.5, then 0.5 - ρ̂ tells us how much more likely non-CFA is.
CFA, giving significant (that students who get questions correct tend to pause more as opposed to 67% of non-CFA). The mean for CFA is numPauses, giving \( \hat{p} \) of 0.47 and 0.55.

\textbf{Pauing Behavior.} This corresponds to Quantities 4 and 5. \textit{numPauses:} Much of the density is in \([0, 1]\) (60% of CFA, 67% of non-CFA). The mean for CFA is 1.77 (SD = 2.05), as opposed to 1.48 (SD = 1.84) for non-CFA. This indicates that students who get questions correct more often (i.e., to reflect on the material), and the overall difference is significant (\( p_W = 2.7e-16 \)). We identified two sets of interest: \{0, 1\}, with more non-CFA density, and \{2, 7\}, with more CFA density, giving \( \hat{p} \) of 0.42 and 0.57.

\textbf{Playback rate behavior.} This is for Quantities 6 and 7. \textit{avgPBR:} Much density is at 1 (63% for non-CFA, 60% for CFA), indicating that many keep the default rate. The mean for both classes is roughly the same at 1.17 (SD = 0.28), but the difference between them is significant (\( p_W = 0.018 \)). We identified two sets: 1, with more non-CFA density, and \( R_{\geq 0} \backslash 1 \), with more CFA, giving \( \hat{p} \) of 0.48 and 0.53.

\textit{stdPBR:} For this quantity, much of the density is at 0 (79% for non-CFA, 75% for CFA), meaning that many hold the playback rate constant. The mean for non-CFA is 0.011 (SD = 0.043), while that for CFA is 0.015 (SD = 0.049). The difference between the distributions is significant (\( p_W = 1.3e-7 \)), indicating that CFA tends to change the playback rate more.

We identified two sets: 0, with more non-CFA density, and \( R_{>0} \), with more CFA density, giving \( \hat{p} \) of 0.46 and 0.53.

\textbf{Jumping behavior.} Finally, this is for Quantities 8 and 9. \textit{numRWs:} Here, much density is at 0 (78% for non-CFA, 73% for CFA). The mean for non-CFA is 0.46 (SD = 1.04), compared to 0.61 (SD = 1.21) for CFA. There is a significant difference between the distributions (\( p_W = 3.5e-10 \)), indicating that CFA tends to rewind more (i.e., revisit material). We consider two sets: 0 and \{1, \ldots, 5\}, with a higher concentration of non-CFA and CFA, respectively, giving \( \hat{p} \) of 0.43 and 0.53.

\textit{numFFs:} The density is largest at 0 (79% for both classes), and the means for both classes are roughly 0.42 (SD = 0.99). There is no significant difference between the classes (\( p_W = 0.768 \)), and we found no sets of interest.

\section{Key Messages}

We conclude from our dataset that satisfying at least one of the following characteristics is an indication that a user has a higher chance of CFA than not on a quiz:

\textit{Playing behavior:} Playing more of the video than its length, spending more time on a video than its length, or completing more than 3/4ths of a video (but not its entirety).

\textit{Pausing behavior:} Pausing more than once, or pausing either for a very short or very long time relative to the video length.

\textit{Playback rate behavior:} Having an average playback rate different from the default speed, or varying the playback rate.

\textit{Jumping behavior:} Rewinding at least once.

These give an instructor indication as to which characteristics serve as signals for success. Turning such indication to prediction is our next step.

\section{Prediction Algorithms}

Now that we have investigated Q1, we move to Q2, which seeks to use our findings to enhance performance prediction for MOOC. Our approach will be formalized in Sec. IV-C. We begin here by describing a number of standard algorithms (Sec. IV-B) that have been applied for prediction in traditional education settings and leverage only performance data, as well as standard metrics (Sec. IV-A) that will be used for evaluation.

\textbf{Definitions.} In general, let \( n \in \Omega \) denote entry/instance \( n \) in the set of all entries \( \Omega \) that form the full dataset. We index users (students) by \( i \) and quizzes (videos) by \( j \); each entry is associated with a particular user \( u(n) \), quiz \( q(n) \), CFA score \( y_n \in \{0, 1\} \) (1 is CFA, 0 is non-CFA), and algorithm prediction \( \hat{y}_n \in \{0, 1\} \). We also write \( n = e(i, j) \) to denote the entry \( n \) associated with user \( i \) and quiz \( j \), where \( e : (i, j) \to \Omega \). For evaluation, we generate training and test sets as subsets of
Fig. 4: Identified intervals/sets with difference between CFA and non-CFA classes. The estimated \( \hat{p} \), 95% CI, and p-value are given for each.

A. Metrics

Accuracy: Let \( \hat{y}_n \in \{0, 1\} \) denote the rounded output of the prediction \( \hat{y}_n \) for entry \( n \). The accuracy is the fraction of the test set in which the true class and \( \hat{y}_n \) agree:

\[
\frac{1}{|\Omega_E|} \sum_{n \in \Omega_E} \mathbb{I}_{\hat{y}_n = y_n},
\]

where \( \mathbb{I} \) is the indicator function.

RMSE: Unlike accuracy, the Root Mean Squared Error (RMSE) uses \( \hat{y}_n \) directly, and is evaluated as follows:

\[
\sqrt{\frac{1}{|\Omega_E|} \sum_{n \in \Omega_E} (y_n - \hat{y}_n)^2}.
\]

AUC: This measures the Area Under the Receiver Operating Characteristic (AUROC) curve of the classifier, where the ROC plots the tradeoff between the true and false positive rates [18]. AUC can also be seen as the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance.

Even one percent improvement in some of these metrics can be substantial. As a reference, for CFA prediction in KDD Cup 2010 there was only 1% improvement in RMSE from the 132nd to the best score on the leaderboard.\(^4\)

\(^4\)https://pslcdatashop.web.cmu.edu/KDDCup/LeaderBoard

C. Our Algorithms Using Clickstream Data

We now present our methods for enhancing performance prediction with video-watching data. Motivated by the findings in Sec. IV, we determine suitable intervals/sets of values (referred to generally as intervals) for each feature by analyzing the densities over \( \Omega_T \), estimate the CFA probabilities within each interval, and use them as learning features. In Sec. V, this will be seen to improve performance relative to the standard algorithms for all metrics and dataset partitions tested.

Interval extraction. Let \( \Omega_T^C \) be the subset of \( \Omega_T \) belonging to class \( C \in \{0, 1\} \), i.e., \( \Omega_T^C = \{ n \in \Omega_T : y_n = C \} \). Also, let \( f \in \mathcal{V} \) denote clickstream quantity \( f \) in the set of behaviors \( \mathcal{V} = \{1, \ldots , 8\} \), indexed as in Sec. II (we do not use 9 because
it was not significant). In determining suitable intervals over \( \Omega_T \), we group each quantity into one of three types:

- **Continuous** (1 – 3, 5): For each continuous \( f \), we approximate the probability density function of each class \( C \) over \( \Omega_T^C \) with a Kernel Density Estimator (KDE):
  \[
  p^C_f(v) = \frac{1}{|\Omega_T^C|} \sum_{v_n \in \Omega_T^C} \kappa \left( \frac{v - v_n}{\eta} \right),
  \]
  where \( v_n \) is the value that quantity \( f \) takes for entry \( n \), \( \eta \) is the bandwidth of the estimator, and \( \kappa(\cdot) \) is the kernel function [18]. Here, we use the standard Gaussian Kernel, and fit the estimator for values \( v \in [0, u_f) \) (the upper bound controls for outliers). Then, we find the intersection points between \( p^C_f(v) \) and \( p^f(v) \) as the boundaries between the intervals for \( f \). More formally, define the ordered set \( \mathcal{I}_f = \{0\} \cup \{v : p^C_f(v) = p^f(v)\} \cup \{u_f\}; \) there, then, are \( |\mathcal{I}_f| - 1 \) intervals, where interval \( h = 1, \ldots, |\mathcal{I}_f| - 1 \) spans the range \( B_h^f = [\mathcal{I}_f(h), \mathcal{I}_f(h+1)] \).

- **Discrete** (4 & 8): For discrete \( f \), we define the empirical probability mass function of each \( C \), \( p^C_f(v) \), over \( \Omega_T^C \) for values \( v \in \{0, \ldots, u_f\} \). Then, we find the values \( v \) at which a change occurs in the class that has more density between \( v \) and \( v + 1 \). More formally, we let \( \mathcal{I}_f = \{0\} \cup \{v : p^C_f(v) \leq p^f(v) \land p^f(v+1) \geq p^f(v + 1)\} \cup \{u_f\}. \)

The interval boundaries are defined by these changes, i.e., \( B_h^f = (\mathcal{I}_f(h), \ldots, \mathcal{I}_f(h+1) - 1) \).

- **Binary** (6 & 7): Though these two features take on continuous values, we saw in Sec. IV that it is more informative to group each of them into two sets: \( B_0^f = \{G_f\} \) and \( B_1^f = \mathbb{R}_{\geq 0} \setminus B_0^f \), where \( G_0 = 1 \) and \( G_T = 0 \).

**Success estimates.** We now compute the CFA estimates for each \( B_h^f \). First, the total occurrences of \( C \) in \( h \) over \( \Omega_T \) is
\[
O^C_f[h] = \sum_{n \in \Omega_T} \mathbb{I}_{y_n = C} \cdot \mathbb{I}_{v_n^f \in B_h^f},
\]
and the corresponding fraction is \( d^C_f[h] = O^C_f[h]/|\Omega_T^C| \). Then, letting \( O_f[h] = O_f^C[h] + O_f^1[h] \), we apply Laplace’s rule of succession [18] to compute the estimated probability of a new element \( n \) (i.e., those in \( \Omega_E \)) with \( v_n^f \in B_h^f \) having \( y_n = C \):
\[
\hat{p}_f[h] = \frac{r_f[h] \cdot O_f[h] + 1}{O_f[h] + 2},
\]
where \( r_f[h] = d^C_f[h]/(d^C_f[h] + d^1_f[h]) \) is the fraction of density in \( h \) that is of the positive class.\(^5\) For the examples with \( v_n^f \not\in [0, u_f] \), we set \( \hat{p}_f[h] = 0.5 \) (i.e., the population average).

Finally, to account for the fact that there can be high variation in the number of samples for each interval, we apply Bayesian adjustment [19] to the estimates as follows:
\[
\tilde{p}_f[h] = \frac{\hat{p}_f[h] \cdot O_f[h] + 0.5\sigma|\Omega_T^C|}{O_f[h] + |\Omega_T^C|},
\]
where \( \sigma \) is a parameter controlling the weight of the population average. In this way, the success estimates are adjusted based

\(^5\)The \( \geq \) and \( \leq \) symbols used in this way imply > and <, or < and >.
\(^6\)The terms of 1 and 2 in the numerator and denominator of \( \tilde{p}_f[h] \) are required in theory to generate the correct estimate over a Bayesian prior.

**SVM classification.** These \( \hat{p}_f[h] \) (or \( \tilde{p}_f[h] \)) are used as features in a Support Vector Machine (SVM) classifier. We choose SVM because it can readily generate complex decision boundaries through application of different kernel functions [20]. We visualize the design matrix \( X \) for the SVM scheme in Fig. 5: with \( I \) users, \( J \) quizzes, and \( |V| = 8 \) clickstream quantities, each instance \( n \) is described by an \( I + J + |V| \) dimensional feature vector \( x_n \), with indicator features for user and quiz, and the CFA probability estimates for each \( f \). For the estimates, we find interval \( h_f \) such that \( v_n^f \in B_h^f \) and then use \( \hat{p}_f[h_f(n)] \) (or \( \tilde{p}_f[h_f(n)] \)) as the corresponding feature value. Then, the following optimization problem [18] is solved:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} ||w||^2 + C \sum_{n \in \Omega_T} \epsilon_n \\
\text{subject to} & \quad y_n^c (\kappa(w, x_n) + w_0) \geq 1 - \epsilon_n, \forall n \in \Omega_T \\
& \quad \epsilon_n \geq 0, \forall n \in \Omega_T
\end{align*}
\]

Here, we use a polynomial kernel of the form \( \kappa(z, z') = (\pi \cdot z^T z' + c_0)^d \), because (i) it readily generates product features of degree \( d \) to help capture interaction terms between the features and (ii) it was seen to give better performance than other kernel choices (such as Gaussian). With the resulting \( w \), \( w_0 \), and \( \epsilon \), the individual probabilities \( \hat{y}_n^c \in [0, 1] \) are generated using the standard Platt scaling procedure implemented in [21].

For VID-A, \( d, C, \pi, c_0, \sigma, \) and \( \epsilon \) are parameters to tune. For VID-N, all except \( \sigma \) must be tuned. We set the \( u_f \) based on our intuition for each clickstream quantity: \( u_1 = u_3 = 2, u_2 = u_5 = 1, u_4 = 10, \) and \( u_8 = 5 \).

**V. PREDICTION PERFORMANCE EVALUATION**

In this section, we will compare the performance of the algorithms presented in Sec. IV on our course data. In doing so, we will find it informative to consider different subsets of the dataset, to see how the performance varies under different conditions (see Sec. II-B). In general, let \( \Omega_{u_0, v_0} \subset \Omega \) denote
the set consisting of all instances \( n \) such that the user \( u(n) \) has \( \geq u_0 \) instances over \( \Omega \) (i.e., having answered at least \( u_0 \) questions) and the video \( q(n) \leq v_0 \) (i.e., within the first \( v_0 \) videos of the course).

A. Implementation Details

Software. The Biases and MF algorithms were each implemented with libFM [22] using stochastic gradient descent (SGD) with a small enough step size (0.01) and a large enough number of iterations (8000) for convergence in all cases. For VID-A and VID-N, the KDE method is implemented through Python’s scikit-learn [21], and the SVM classifier through libSVM [23] with the SMO algorithm. The naive and KNN algorithms were programmed de novo in Python.

Training and cross validation. In evaluating the algorithms over \( \Omega^{u_0,v_0} = \Omega' \), we use k-fold cross validation (CV) [18] to consider multiple training/test set partitions. We partition \( \Omega' \) into \( k \) disjoint subsets \( \Omega_1, \ldots, \Omega_k \) such that \( \Omega_1 \cup \cdots \cup \Omega_k = \Omega' \). These subsets are formed as follows: letting \( \mathcal{N}_i' = \{ n \in \Omega : u(n) = i \} \) (i.e., all instances of user \( i \) in \( \Omega' \)), we randomly permute \( \mathcal{N}_i' \) and allocate the \( i \)th set of \( \lfloor |\mathcal{N}_i'|/k \rfloor \) instances to \( \Omega_i \); the remainder is allocated across the subsets randomly. This process is repeated over all users, and is done to ensure that the entries for each user are spread evenly across the sets, since the \( |\mathcal{N}_i'| \) tend to be small. With the \( k \) subsets in hand, for \( z = 1, \ldots, k \) each algorithm is trained on \( \Omega_T = \Omega \setminus \Omega_z \) and tested on \( \Omega_E = \Omega_z \), and each of the metrics are averaged over the \( k \) trials. We set \( k = 5 \) in our evaluation.

Parameter tuning. We handle tuning of continuous and discrete parameters differently in a procedure which we followed closely for each algorithm. The continuous ones were tuned over \( \Omega^{20,92} \) using a multi-dimensional grid search procedure [20]. To do this, we first randomly selected 15% of the instances from each subset \( \Omega_i \). Then, the following is performed for each algorithm: (1) choose initial center points \( c_p \in \mathbb{R} \), ranges \( r_p \in \mathbb{N} \), and step sizes \( s_p \in \mathbb{R}_{>0} \) for each parameter \( p \); (2) run 5-fold CV over all combinations in the set \( \mathcal{G} = \{ c_1-r_1 s_1, \ldots, c_1+r_1 s_1 \} \times \cdots \times \{ c_p-r_p s_p, \ldots, c_p+r_p s_p \} \) \(^8\) where \( P \) is the total number of parameters; (3) set \( (c_1, \ldots, c_P) = \mathcal{G}_g \), where \( g \) is the index of the combination with the highest accuracy, and set \( s_p \) to \( s_p/\zeta \), \( \zeta > 1 \) \( \forall p \); (4) repeat until \( (c_1, \ldots, c_P) \) does not change between three successive iterations. The final \( (c_1, \ldots, c_P) \) are taken as the tuned parameters. Due to space constraints we do not report the initial \( c_p \), \( r_p \), and \( s_p \) for each algorithm.

Since the behavior of the discrete parameters is not easy to capture in the above procedure, we simply repeat the search over the continuous parameters for each discrete choice, choosing the best overall combination. The final values tested for \( K_M \) and \( d \) were in \{1,...,10\}, and for \( K_N \) were in \{20,...,40\}. \(^9\) In Fig. 6, we give the tuned parameters for each algorithm.

| Biases | \( \lambda_B = 0.105 \) |
| MF | \( K_M = 3, \lambda_B = 0.115, \lambda_M = 0.181 \) |
| KNN | \( K_T = 30, \alpha = 18.4, \beta = 1.9, \delta = 14.1, \gamma = -5.0 \) |
| VID-N | \( d = 6, C = 0.011, \pi = 0.008, c_0 = 1.88, \eta = 0.183 \) |
| VID-A | \( d = 6, C = 0.0062, \pi = 2.38, c_0 = -1.44, \sigma = 0.130, \eta = 0.0186 \) |

\(^{8}\)The performance of MF in educational settings is known to saturate after the first few factor dimensions [9]. For KNN, the performance did not vary much within \{20,...,40\}.
and which is substantial but not as high as for the standards is 1.55%, 1.91%, and 4.60% for each metric, those from hand, the RMSE improvements are roughly consistent with performance than the percent improvements. Compared with the previous two evaluation over reasonable to conclude that video-watching data is particularly best achieved in each case.

**C. Discussion, Intuition, and Extensions**

**Benefit of behavioral data to early prediction.** The evaluations consider one type of student behavior – video-watching data – to see how it can improve performance prediction. We see that incorporating it can achieve gains over standard algorithms, but that the highest benefit comes from applying it early in the course (Fig. 8(b)) for “quickest detection,” discussed shortly. The reason for the high differential in this case is that there is relatively few entries for each individual user, which puts the Biases and MF algorithms at a clear disadvantage since they rely on explicit user models (i.e., $b_i$ and $u_i$). On the other hand, while the clickstream algorithms incorporate user/quiz biases (i.e., indicator features in Fig. 5), they also leverage the video-watching data aggregated over all users to assist in classifying CFA or not. The sensitivity of the standards to per-user information is further emphasized when considering all users over the entire course (Fig. 8(c)): the performance improves, because there are more entries for each user, but is not near what is possible when only considering active users, because there are many who only take a few quizzes (see Fig. 1). In future work, we plan to investigate how other forms of behavioral data can be leveraged to enhance clustering students: One is a graph of students, where a given pair is linked if they share similar predicted CFA scores across all questions currently available. By varying the threshold similarity required for a link, this could lead to different
clusters of students with enough correlation for the instructor to address them collectively.

User and quiz detection: Another example is a bipartite graph between user and assessment nodes, where a user is linked to an assessment with a weight equal to the predicted CFA score. The interface could then aggregate the link weights across user nodes so that the instructor can identify those who are struggling or need additional challenge and assign supplementary material accordingly. It could similarly do this for quiz nodes, to identify those which may need to be explained more thoroughly or should be more challenging.

Study buddies: Another example is a graph of users in which pairs who may work well together as study partners are linked, by connecting those who tend to have opposing skills so they can provide mutual aid to each other. One way of determining this would be to find the correlation coefficient between each pair’s predicted CFA scores on different quizzes, determine each user’s preferences by ranking potential partners based on these coefficients, and then apply a stable matching algorithm.

In each of these cases, predictions are particularly helpful in the early stages of the course, since many of the CFA scores are not yet known. We are in the process of developing an instructor interface for our own MOOC platform, 3ND, and with it we plan to investigate these applications further.

Other courses and assessments. As stated in Sec. II, our MOOC setup is convenient for performance prediction using clickstream data because of the 1:1 correspondence between videos and quizzes. But other courses may not have this property. The way to handle it in the general case is to have the instructor add tags to specific lengths in videos that are pertinent to a given assessment question. Then, by considering the lengths between these tags as separate “videos” paired with the corresponding questions, the methods developed here are directly applicable.

VI. CONCLUSION

Student performance prediction is an intriguing research area, and especially so for MOOC because of its potential benefits, such as the definition of different SLN graph structures that can help an instructor manage her course more effectively. In this paper, using data from one of our own MOOC offerings, we applied some standard algorithms to CFA prediction in this setting, and showed how one type of behavioral data collected about students – video-watching clickstream events – can be used as learning features to improve prediction quality. Through evaluation, we saw that our scheme outperformed the standards under each dataset partition and metric considered, and that the improvement was particularly pronounced in the beginning of the course. Also, we saw that it is useful to parse the clickstream data into summary quantities for each user-video pair, because in doing so is possible to identify intervals for these quantities that indicate a higher likelihood of a user being CFA or not in answering the corresponding question. In the future, we plan to consider other types of behavioral data, test over other course offerings, and incorporate our schemes into our own instructor interface for the continuous evolution of large-scale social learning networks.

ACKNOWLEDGMENT

This work was in part supported by ARO grants W911NF-14-1-0190 and W911NF-11-1-0036. Additionally, we thank the reviewers for their valuable comments.

REFERENCES