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 student bodies 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.

Organization. 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.

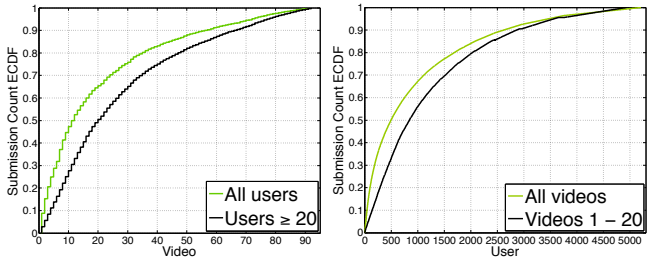
II. Course Description and Basic Statistics

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, it is not required, as we will discuss in Sec. V-C.

[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.



(a) All users (top) and all with at least 20 submissions (bottom).

(b) All videos (top) and only the first 20 videos (bottom).

Fig. 1: ECDFs of submission count (a) over videos, where counting only serious users makes the distribution more uniform, and (b) over users, where counting only the first part of the course does the same.

B. Quiz and Clickstream Data

1) Quiz submissions: An export of the quiz responses, including the user ID, quiz ID, time of submission, and answer selected, was obtained from Coursera. From this, we were able to determine all of the (first-attempt) user-quiz submission pairs for the course. There were 40,464 such pairs, from a total of 5,205 users and 92 quizzes (an error prevented one quiz from user display). This means that less than 12% of the entries were known, underscoring the sparsity of the data.

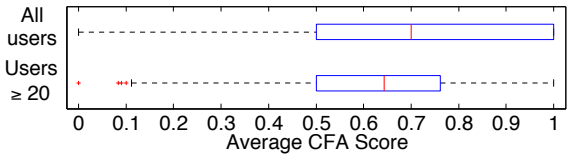
Basic statistics. We first present some statistics on how the submission counts and CFA scores vary across videos and users. This will be useful when we consider different subsets of the data for analysis and evaluation. To this end, we consider a video to be in the "beginning" of our course if it is within the first 20, or roughly the first two weeks of the 12-week course. Further, we consider a user to be "active" if she answered at least 20 questions, or two weeks worth of the quizzes.[2]

Submission counts: In Fig. 1(a), we show the empirical CDF (ECDF) of the number of submissions made for each video. The first trace includes all users. The earlier videos had more submissions; the first 11 accounted for 50%, showing the drop-off in participation over time. The mean number of submissions per video is 440, with a high SD (608). The second trace only includes submissions from the 794 active users who took at least 20 quizzes, where there is less variation; the mean is 266 with a lower SD (210).

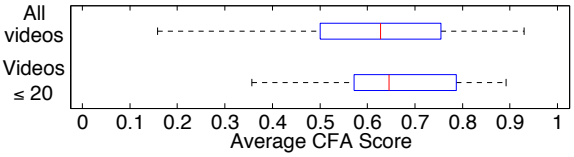
Fig. 1(b) shows the ECDF of submissions over users. Considering all videos, 50% of the submissions were made by the most active 497 users; in fact, only 18 submitted all questions, while 2246 submitted only one or two. The mean questions submitted was 7.8, with a high SD (13.3). The mean over the first 20 videos is 5.08 per user with a lower SD (5.60).

Average CFA scores: The mean score over all 40K pairs is 0.658 with an SD of 0.474. Fig. 2(a) shows the average score over users, which is the mean of the CFA scores (0 or 1) over all quizzes a user submitted. The top shows all users; the mean is 0.646, with a high SD (0.342). The large variance is due in part to the fact that many users submit few quizzes. To see

[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.



(a) All users (top) and all with ≥ 20 quizzes (bottom).



(b) All videos (top) and the first 20 videos (bottom).

Fig. 2: Boxplots of average CFA scores (a) over users, where counting only serious users reduces the variance, and (b) over videos, where the average for those in the first two weeks is higher.

this, at the bottom we plot the average score across users with at least 20 quizzes, and the SD drops substantially (0.181), with the mean only lowering slightly (0.624).

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 played, with repetition, relative to its length.

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 (p_W) from this test, and when it is low enough (below 0.05), we can 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 $\hat{p}$ is significant.[3] For these cases, we report the p-value, $\hat{p}$, and a 95% Confidence Interval (CI) around $\hat{p}$, 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 and non-CFA samples.

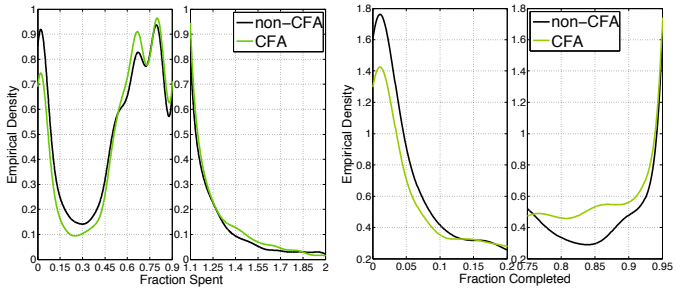
A. Statistical Analysis

Playing behavior. This corresponds to Quantities 1 to 3.

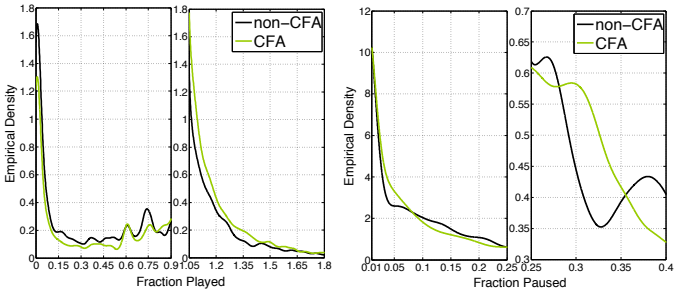
fracSpent: Much of the density is in $[0.9, 1.1]$ (40% of CFA, 42% of non-CFA). The mean for CFA is 0.82 (SD = 0.36), compared to 0.78 (SD = 0.39) for non-CFA. This indicates that, as expected, a user submitting a correct answer tends to have spent more time with the video. The difference between the distributions is significant (p_W = 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 $\hat{p}$ 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

[3]In other words, a significant p-value tells us we can trust $\hat{p}$. Then, if $\hat{p}$ is above 0.5, $\hat{p} - 0.5$ tells us how much more likely CFA is in that interval; if below 0.5, then $0.5 - \hat{p}$ tells us how much more likely non-CFA is.



(a) fracSpent, 3 intervals. $\eta = 0.02$. (b) fracComp, 2 intervals. $\eta = 0.02$.



(c) fracPlayed, 2 intervals. $\eta = 0.02$. (d) fracPaused, 3 intervals. $\eta = 0.015$.

Fig. 3: Depiction of intervals of significance for continuous clickstream quantities, considering all videos and all users taking at least 20 quizzes. We can visually identify the difference between the CFA and non-CFA classes, particularly in (b) and (d).

distributions not significant ($p_W = 0.443$). Still, we identified two intervals of significance (Fig. 3(b)): $[0, 0.13]$, where there is more non-CFA density, and $[0.76, 0.95]$, with more CFA, giving $\hat{p}$ of 0.47 and 0.53.

fracPlayed: For this, much of the density is in $[0.9, 1.1]$ (64% of CFA, 62% of non-CFA). The mean for CFA is 0.91 (SD = 0.36), as opposed to 0.85 (SD = 0.39) for non-CFA. Similar to fraction spent, this indicates that a student submitting CFA tends to watch more of the video, and the difference between the distributions is significant ($p_W =$ 4.2e-24). We found two intervals (Fig. 3(c)): $[0, 0.8]$, with more non-CFA density, and $[1.05, 1.65]$ with more CFA, giving $\hat{p}$ of 0.44 and 0.55.

Pausing Behavior. This corresponds to Quantities 4 and 5.

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 tend to pause more (*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 $\{2, ..., 10\}$ with more CFA, giving $\hat{p}$ of 0.42 and 0.57.

fracPaused: Here, much density lies in $[0, 0.01]$ (32% of CFA, 35% of non-CFA). The mean of each class is roughly 0.13 (SD = 0.20), and there is no significant difference between the distributions ($p_W = 0.253$). Even so, we identify three intervals (Fig. 3(d)): $[0.01, 0.082]$ and $[0.28, 0.356]$, where there is more non-CFA density, and $[0.082, 0.25]$, where there is more CFA, giving $\hat{p}$ of 0.54, 0.51, and 0.46.

Playback rate behavior. This is for Quantities 6 and 7.

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 $\mathbb{R}_{\geq 0} \backslash 1$, with more CFA, giving $\hat{p}$ of 0.48 and 0.53.

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 $\mathbb{R}_{>0}$, with more CFA density, giving $\hat{p}$ of 0.46 and 0.53.

Jumping behavior. Finally, this is for Quantities 8 and 9.

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 $\{1, ..., 5\}$, with a higher concentration of non-CFA and CFA, respectively, giving $\hat{p}$ of 0.43 and 0.53.

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.

B. 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:

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).

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

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

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.

IV. 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.

Definitions. In general, let $n \in \Omega$ denote entry/instance n in the set of all entries Ω 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

Feat	Int/Set	$\hat{p}$	95% CI	p-value
fracSpent	$[0, 0.54]$	0.452	$(0.438, 0.466)$	5.27e-12
	$[0.54, 0.90]$	0.519	$(0.504, 0.534)$	0.017
	$[1.1, 2.0]$	0.511	$(0.500, 0.521)$	0.049
fracComp	$[0, 0.13]$	0.471	$(0.459, 0.483)$	1.63e-6
	$[0.76, 0.95]$	0.527	$(0.517, 0.537)$	3.59e-7
fracPlayed	$[0, 0.80]$	0.437	$(0.422, 0.451)$	2.2e-16
	$[1.05, 1.65]$	0.554	$(0.541, 0.567)$	6.9e-15
numPaused	$\{0, 1\}$	0.420	$(0.403, 0.437)$	2.2e-16
	$\{2, ..., 10\}$	0.567	$(0.550, 0.584)$	1.35e-14

Feat	Int/Set	$\hat{p}$	95% CI	p-value
fracPaused	$[0.01, 0.082]$	0.540	$(0.527, 0.553)$	3.97e-9
	$[0.082, 0.25]$	0.464	$(0.450, 0.477)$	6.06e-8
	$[0.28, 0.356]$	0.507	$(0.501, 0.513)$	0.022
avgPBR	$\{1\}$	0.481	$(0.463, 0.498)$	0.032
	$\mathbb{R}_{\geq 0} \setminus \{1\}$	0.529	$(0.512, 0.546)$	8.2e-4
stdPBR	$\{0\}$	0.463	$(0.448, 0.477)$	1.11e-6
	$\mathbb{R}_{\geq 0} \setminus \{0\}$	0.530	$(0.523, 0.552)$	1.11e-6
numRW	$\{0\}$	0.429	$(0.413, 0.446)$	< 2.2e-16
	$\{1, ..., 5\}$	0.533	$(0.519, 0.548)$	1.07e-5

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.

Ω through a procedure described in Sec. V-A; the training set Ω_T and test set Ω_E are always chosen such that $\Omega_T \cap \Omega_E = \emptyset$.

A. Metrics

Accuracy: Let $\tilde{y}_n \in \{0, 1\}$ denote the rounded output of the prediction $\hat{y}_n$ for entry n. The accuracy is the fraction of these in the test set that are correct:

$$\frac{1}{|\Omega_E|} \sum_{n \in \Omega_E} \mathbb{I}_{\tilde{y}_n = y_n},$$

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 instance in the positive class (*i.e.,* $y_n = 1$) 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]

B. Standard Algorithms in Big Data

Naive: This predicts the global average over Ω_T for all n:

$$\hat{y}_n = \frac{1}{|\Omega_T|} \sum_{n' \in \Omega_T} y_{n'}.$$

This most naive predictor will only serve as a benchmark for measuring incremental improvement for all other algorithms.

Biases: This includes a bias for each user and quiz, and is equivalent to the Rasch model in IRT [9] with a global bias term. Letting $b_{u(n)}$ and $b_{q(n)}$ be the biases for $u(n)$ and $q(n)$, $\mathbf{b}$ be the vector of biases, and μ be a global term, we solve

$$\underset{\mu, \mathbf{b}}{\operatorname{argmin}} \sum_{n \in \Omega_T} -\ln \Phi\left(\hat{y}'_n \cdot y'_n\right) + \frac{\lambda_B}{2} ||\mathbf{b}||^2,$$

where $\hat{y}'_n = 2(\hat{y}_n - 0.5)$, $y'_n = 2(y_n - 0.5)$, $\hat{y}_n = \mu + b_{u(n)} + b_{q(n)}$, λ_B is the regularization parameter, and $\Phi(\cdot)$ is the sigmoid function $\Phi(z) = 1/(1 + e^{-z})$. Here, λ_B must be tuned through cross validation (described in Sec. V).

Matrix Factorization (MF): This includes a latent factor vector of dimension K_M for each user, $\mathbf{u}_i \in \mathbb{R}^{K_M}$, and quiz, $\mathbf{q}_j \in \mathbb{R}^{K_M}$, in addition to the biases from the previous case. It is a common CoF method that modifies conventional Singular Value Decomposition (SVD) to learn only over known training instances [9]–[11]. Letting $\mathbf{U} = [\mathbf{u}_i]$ and $\mathbf{Q} = [\mathbf{q}_j]$ be the matrix of user and quiz factors, respectively, we solve

$$\underset{\mu, \mathbf{b}, \mathbf{U}, \mathbf{Q}}{\operatorname{argmin}} \sum_{n \in \Omega_T} -\ln \Phi\left(\hat{y}'_n \cdot y'_n\right) + \frac{\lambda_B ||\mathbf{b}||^2 + \lambda_M (||\mathbf{U}||^2 + ||\mathbf{Q}||^2)}{2},$$

where $\hat{y}'_n$ and y'_n are defined as in Biases, and $\hat{y}_n = \mu + b_{u(n)} + b_{q(n)} + \mathbf{u}^T_{u(n)} \mathbf{q}_{q(n)}$. λ_B, λ_M, and K_M must be tuned.

K Nearest Neighbor (KNN): For user i, this uses the CFA scores of the set of K_N users $\mathcal{U}_i$ who have the most similar quiz results to i for prediction. We determine $\mathcal{U}_i$ for all i as in [7]: (i) compute the Pearson correlation coefficient $\rho_{i,i'}$ between i and all other users i' using the performances on the set of quizzes $\mathcal{J}_{i,i'}$ common to the pair in Ω_T, (ii) shrink the correlations according to $\bar{\rho}_{i,i'} = \frac{|\mathcal{J}_{i,i'}| \rho_{i,i'}}{|\mathcal{J}_{i,i'}| + \alpha}$, (iii) apply the sigmoid mapping $\tilde{\rho}_{i,i'} = \Phi\left(\delta \bar{\rho}_{i,i'} + \gamma\right)$, and (iv) define $\mathcal{U}_i$ as the set of K_N users $i' \neq i$ with maximum $|\tilde{\rho}_{i,i'}|$. Then,

$$\hat{y}_n = \frac{\sum_{i' \in \mathcal{U}'_{u(n)}} \tilde{\rho}_{u(n),i'} \cdot y_{e(i',q(n))} + m_{u(n)} \beta}{\sum_{i' \in \mathcal{U}'_{u(n)}} |\tilde{\rho}_{u(n),i'}| + \beta},$$

where $\mathcal{U}'_{u(n)}$ is the set of only those users $i' \in \mathcal{U}_{u(n)}$ with $e(i', q(n)) \in \Omega_T$, and m_i is the mean score of user i over Ω_T. Here, α, δ, γ, β, and K_N are parameters to be tuned.

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, they determine suitable intervals/sets of values (referred to generally as intervals) for each feature by analyzing the densities over Ω_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 Ω^C_T be the subset of Ω_T belonging to class $C \in \{0, 1\}$, *i.e.,* $\Omega^C_T = \{n \in \Omega_T : y_n = C\}$. Also, let $f \in \mathcal{V}$ denote clickstream quantity f in the set of behaviors $\mathcal{V} = \{1, ..., 8\}$, indexed as in Sec. II (we do not use 9 because

[4] https://pslcdatashop.web.cmu.edu/KDDCup/LeaderBoard

it was not significant). In determining suitable intervals over Ω_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 Ω_T^C with a Kernel Density Estimator (KDE):

$$p_f^C(v) = \frac{1}{|\Omega_T^C|\eta} \sum_{n \in \Omega_T^C} \kappa\left(\frac{v - v_n^f}{\eta}\right),$$

where v_n^f is the value that quantity f takes for entry n, η 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 $p_f^0(v)$ and $p_f^1(v)$ as the boundaries between the intervals for f. More formally, define the ordered set $\mathcal{I}_f = \{0\} \cup \{v : p_f^0(v) = p_f^1(v)\} \cup \{u_f\}$; then, there are $|\mathcal{I}_f| - 1$ intervals, where interval $h \in \{1, ..., |\mathcal{I}_f| - 1\}$ spans the range $\mathcal{B}_h^f = [\mathcal{I}_f(h), \mathcal{I}_f(h+1)]$.

Discrete (4 & 8): For discrete f, we compute the empirical probability mass function of each C, $p_f^C(v)$, over Ω_T^C for values $v \in \{0, ..., u_f\}$. 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 $\mathcal{I}_f = \{0\} \cup \{v : p_f^0(v) \lesseqgtr p_f^1(v) \ \wedge \ p_f^0(v+1) \gtreqless p_f^1(v+1)\} \cup \{u_f\}$.[5] The interval boundaries are defined by these changes, *i.e.,* $\mathcal{B}_h^f = \{\mathcal{I}_f(h), ..., \mathcal{I}_f(h+1) - 1\}$.

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: $\mathcal{B}_0^f = \{G_f\}$ and $\mathcal{B}_1^f = \mathbb{R}_{\geq 0} \setminus \mathcal{B}_0^f$, where $G_6 = 1$ and $G_7 = 0$.

Success estimates. We now compute the CFA estimates for each $\mathcal{B}_h^f$. First, the total occurrences of C in h over Ω_T is

$$O_f^C[h] = \sum_{n \in \Omega_T} \mathbb{I}_{y_n = C} \cdot \mathbb{I}_{v_n^f \in \mathcal{B}_h^f},$$

and the corresponding fraction is $d_f^C[h] = O_f^C[h] / |\Omega_T^C|$. Then, letting $O_f[h] = O_f^0[h] + O_f^1[h]$, we apply Laplace's rule of succession [18] to compute the estimated probability of a new element n (*i.e.,* those in Ω_E) with $v_n^f \in \mathcal{B}_h$ having $y_n = C$:

$$\hat{p}_f[h] = \frac{r_f[h] \cdot O_f[h] + 1}{O_f[h] + 2},$$

where $r_f[h] = d_f^1[h] / (d_f^0[h] + d_f^1[h])$ is the fraction of density in h that is of the positive class.[6] For the examples with $v_n^f \notin [0, u_f]$, 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:

$$\tilde{p}_f[h] = \frac{\hat{p}_f[h] \cdot O_f[h] + 0.5\sigma|\Omega_T|}{O_f[h] + |\Omega_T|},$$

where σ is a parameter controlling the weight of the population average. In this way, the success estimates are adjusted based on the sample sizes of the corresponding interval to reduce the possibility of overfitting based on a small number of samples. This is particularly useful for the continuous features, where η trades off the bias and variance of the KDE [18]; as we will see in Sec. V, including σ allows the estimators to risk choosing a lower η (higher variance), thereby generating a larger number of smaller-width intervals, since the $\hat{p}_f[h]$ are adjusted accordingly. To evaluate this tradeoff, we will consider two separate instances: **VID-A**, which uses $\tilde{p}_f[h]$ for the success estimates, and **VID-N**, which uses $\hat{p}_f[h]$ instead.

[5]The $\gtreqless$ and $\lesseqgtr$ symbols used in this way imply $>$ and $<$, or $<$ and $>$.

[6]The terms of 1 and 2 in the numerator and denominator of $\hat{p}_f[h]$ are required in theory to generate the correct estimate over a Bayesian prior.

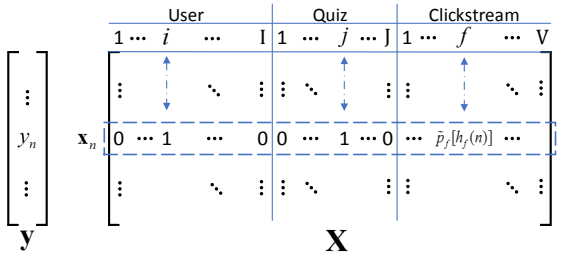


Fig. 5: Visualization of the design matrix $\mathbf{X}$ and target vector $\mathbf{y}$ for the SVM classification algorithms using clickstream features.

SVM classification. These $\tilde{p}_f[h]$ (or $\hat{p}_f[h]$) 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 $\mathbf{X}$ for the SVM scheme in Fig. 5: with I users, J quizzes, and $|\mathcal{V}| = 8$ clickstream quantities, each instance n is described by an $I + J + |\mathcal{V}|$ dimensional feature vector $\mathbf{x}_n$, 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 $v_n^f \in \mathcal{B}_h^f$ and then use $\tilde{p}_f[h_f(n)]$ (or $\hat{p}_f[h_f(n)]$) as the corresponding feature value. Then, the following optimization problem [18] is solved:

$$\begin{aligned} \underset{\mathbf{w}, w_0, \epsilon}{\text{minimize}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C \sum_{n \in \Omega_T} \epsilon_n \\ \text{subject to} \quad & y_n' \left(\kappa(\mathbf{w}, \mathbf{x}_n) + w_0\right) \geq 1 - \epsilon_n, \ \forall n \in \Omega_T \\ & \epsilon_n \geq 0, \ \forall n \in \Omega_T \end{aligned}$$

Here, we use a polynomial kernel of the form $\kappa(\mathbf{z}, \mathbf{z}') = \left(\pi \cdot \mathbf{z}^T \mathbf{z}' + c_0\right)^d$, 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 $\mathbf{w}$, w_0, and ϵ, the individual probabilities $\hat{y}_n \in [0, 1]$ are generated using the standard Platt scaling procedure implemented in [21].

For VID-A, d, C, π, c_0, σ, and η are parameters to tune. For VID-N, all except σ 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 Ω (*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 Ω' into k disjoint subsets $\Omega_1, ..., \Omega_k$ 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 Ω'), we randomly permute $\mathcal{N}_{i'}$ and allocate the lth set of $\lfloor |\mathcal{N}_{i'}|/k \rfloor$ instances to Ω_l; 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 $z = 1, ..., k$ 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 Ω_l.[7] 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 $\mathcal{G} = \{2^{c_1 - r_1 s_1}, ..., 2^{c_1 + r_1 s_1}\} \times \cdots \times \{2^{c_P - r_P s_P}, ..., 2^{c_P + r_P s_P}\}$,[8] where P is the total number of parameters; (3) set $(c_1, ..., c_P) = \mathcal{G}_g$, where g is the index of the combination with the highest accuracy, and set s_p to s_p/ζ, $\zeta > 1$ $\forall p$; (4) repeat until $(c_1, ..., c_P)$ does not change between three successive iterations. The final $(c_1, ..., c_P)$ 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 that are used to generate the results in Sec. V-B. Notice that as expected, VID-A has a lower η than VID-N.

[7] In the end, for KNN we used the entirety of $\Omega^{20,92}$ because of its sensitivity to selecting the specific neighbors for each user. For the other algorithms, 15% was not seen to significantly affect the parameter choices.

[8] For the parameters that can take positive and negative values, $\mathcal{G} = \{c_1 - r_1 s_1, ..., c_1 + r_1 s_1\} \times \cdots \times \{c_P - r_P s_P, ..., c_P + r_P s_P\}$ instead.

Biases	$\lambda_B = 0.105$
MF	$K_M = 3$, $\lambda_B = 0.115$, $\lambda_M = 0.181$
KNN	$K_N = 30$, $\alpha = 18.4$, $\beta = 1.9$, $\delta = 14.1$, $\gamma = -5.0$
VID-N	$d = 6$, $C = 0.011$, $\pi = 0.608$, $c_0 = 1.88$, $\eta = 0.183$
VID-A	$d = 6$, $C = 0.0062$, $\pi = 2.38$, $c_0 = -1.44$, $\sigma = 0.030$, $\eta = 0.0186$

Fig. 6: Tuned parameters for each of the algorithms.

B. Results

We now present an evaluation of the algorithms under different scenarios, followed by additional discussion in Sec. V-C. In comparing the performance between algorithms, we will consider the percent improvement (PI) of each over the Naive benchmark, which has an accuracy of roughly 0.66 and an RMSE and AUC of 0.5 in each case. To obtain PI, we measure percent increase for accuracy and AUC (higher is better), and percent decrease for RMSE (lower is better). We repeated the CV procedure 5 times for each algorithm in each dataset (*i.e.,* 25 runs each) and averaged the results.

Active users over the full course. We first evaluate the algorithms on $\Omega^{20,92}$, to only consider the active users who have taken at least 20 questions. Fig. 7(a) tabulates the results for each of the algorithms and metrics, and Fig. 8(a) shows the PI in each case. Notice that VID-N and VID-A both outperform the three standard algorithms at least slightly for each of the metrics. For accuracy and AUC, the improvement differential is marginal, with an increase of 0.27% (in both cases) from Biases to VID-A and VID-N, respectively, while for RMSE, it is more pronounced, with an increase of 0.89% from Biases to VID-A. Among the standards, the KNN algorithm performs the worst in all cases, which is consistent with results in other work (*e.g.,* [7]), but one surprising point is that MF has slightly lower performance than Biases even though it adds factor dimensions. Among the clickstream algorithms, we see that VID-A performs better than VID-N on the accuracy metric but that on RMSE and AUC they are roughly equivalent.

All users over the first two weeks. Next, we evaluate the algorithms on $\Omega^{0,20}$, to consider all users in the first two course weeks, *i.e.,* the beginning of the course. Fig. 7(b) tabulates the absolute metrics, and Fig. 8(b) shows the percent improvements. Compared with the previous case, each algorithm has lower performance, which is expected since we have less information to learn from for each user and quiz. Additionally, we see that MF now slightly outperforms Biases, and that VID-A has substantially higher performance than VID-N overall; this second point highlights the utility of including both η and σ in the video-watching scheme. Finally, we see that both of the video-watching algorithms again outperform the standard algorithms for each metric. But the remarkable result here is the incremental improvement,

[9] 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 $\{20, ..., 40\}$.

Algorithm	KNN	Biases	MF	VID-N	VID-A
Accuracy	0.7062	0.7234	0.7231	0.7238	**0.7252**
RMSE	0.4371	0.4321	0.4327	**0.4277**	**0.4276**
AUC	0.7359	0.7581	0.7580	**0.7594**	**0.7593**

(a) $\Omega^{20,92}$: Active users over full course.

Algorithm	KNN	Biases	MF	VID-N	VID-A
Accuracy	0.6963	0.6977	0.6977	0.7078	**0.7117**
RMSE	0.4622	0.4473	0.4464	0.4382	**0.4364**
AUC	0.6664	0.6864	0.6876	0.7066	**0.7115**

(b) $\Omega^{0,20}$: All users over first two weeks.

Algorithm	KNN	Biases	MF	VID-N	VID-A
Accuracy	0.6864	0.6968	0.6961	0.7023	**0.7070**
RMSE	0.4619	0.4479	0.4475	0.4400	**0.4380**
AUC	0.6796	0.7039	0.7046	0.7221	**0.7276**

(c) Ω: All users over full course.

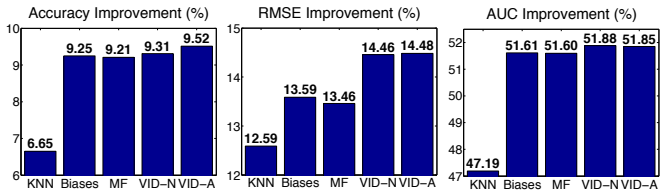
Fig. 7: Absolute performance metrics obtained from evaluating the algorithms over different subsets of the course data. Bold denotes the best achieved in each case.

which shows an increase of 2.06%, 2.02%, and 4.78% for accuracy, RMSE, and AUC from MF to VID-A. Hence, it is reasonable to conclude that video-watching data is particularly useful for performance prediction early in a course when it is not yet clear which users are active.

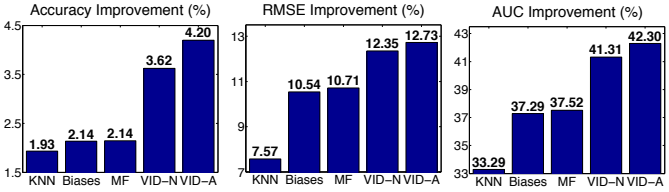
All users over the full course. Finally, we perform an evaluation over Ω to consider learning on the full dataset. Fig. 7(c) tabulates the absolute metrics, and Fig. 8(c) shows the percent improvements. Compared with the previous two cases, for both accuracy and AUC the algorithms show lower performance than $\Omega^{20,92}$ but higher than $\Omega^{0,20}$; on the other hand, the RMSE improvements are roughly consistent with those from $\Omega^{0,20}$. The improvement of VID-A compared with the standards is 1.55%, 1.91%, and 4.60% for each metric, which is substantial but not as high as for $\Omega^{0,20}$.

C. Discussion, Intuition, and Extensions

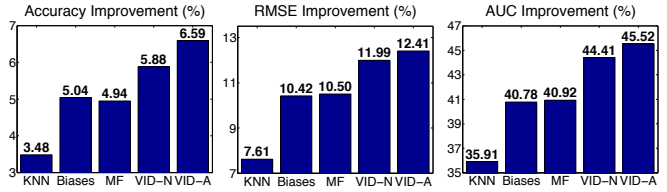
Benefit of behavioral data to early prediction. The evaluations consider one type of student behavior – video-watching data – to see how it can improve performance prediction. We see that incorporating it can achieve gains over standard algorithms, but that the highest benefit comes from applying it 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 $\mathbf{u}_i$). 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



(a) $\Omega^{20,92}$: Active users over full course.



(b) $\Omega^{0,20}$: All users over first two weeks.



(c) Ω: All users over full course.

Fig. 8: Percent increases in prediction performance relative to the benchmark for each scenario. VID-N and VID-A are seen to outperform the standard algorithms in each case for each metric; however, the incremental gain is the highest when considering VID-A in $\Omega^{0,20}$.

CFA prediction in these settings as well.

Clickstream algorithms. VID-A was seen to outperform VID-N substantially in the second two dataset partitions considered. As discussed in Sec. IV-C, the difference between them is that the smaller η for VID-A causes the KDE for clickstream quantities 1-3,5 to generate more, smaller-width intervals (*e.g.,* for $\Omega^{20,92}$ the average number of intervals across these quantities was 13 for VID-A and only 2.4 for VID-N), since σ compensates for overfitting on small sample sizes. Note also that η in VID-A is close to those used in Fig. 3, except that for the preliminary analysis we focused only on the 2-3 intervals we identified visually.

In terms of these algorithms, we make three points here for future investigation: (1) we believe the performance would have been higher by including separate η_f and σ_f parameters for each Quantity f (or groups of f), (2) an alternative to Bayesian adjustment could have been a constraint which requires (in the continuous case) the difference in density area $|\int_{v_1}^{v_2}(p_f^1(v) - p_f^0(v))dv| \geq a$ in order for $[v_1, v_2]$ to be an interval, where a is another parameter to tune, and (3) due to the discriminative nature of the clickstream quantities, a decision tree-type algorithm [13] that uses these quantities for branching may enhance prediction quality further.

SLN graph applications. Robust CFA prediction can be useful to a MOOC course staff in numerous ways. Many of these come in the form of SLN graph structures that could give keen insight into student learning. We give a few examples:

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,[10] 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.

[10] `http://www.3nightsdone.org`

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

[1] A. Brown, "Moocs make their move," *The Bent*, vol. 104, no. 2, pp. 13–17, 2013.

[2] C. G. Brinton, M. Chiang, S. Jain, H. Lam, Z. Liu, and F. M. F. Wong, "Learning about social learning in moocs: From statistical analysis to generative model," *To appear, IEEE Trans. Learning Technol.*, 2014.

[3] K. Stephens-Martinez, M. A. Hearst, and A. Fox, "Monitoring moocs: which information sources do instructors value?" in *Proceedings of the first ACM conference on Learning@ scale*. ACM, 2014, pp. 79–88.

[4] C. G. Brinton and M. Chiang, "Social learning networks: A brief survey," in *Information Sciences and Systems (CISS), 2014 48th Annual Conference on*. IEEE, 2014, pp. 1–6.

[5] M. Chiang. (2012, sept) Networks: Friends, money, and bytes. [Online]. Available: https://www.coursera.org/course/friendsmoneybytes

[6] M. Reckase, *Multidimensional item response theory*. Springer, 2009.

[7] A. Toscher and M. Jahrer, "Collaborative filtering applied to educational data mining," *KDD Cup*, 2010.

[8] B. Beheshti, M. C. Desmarais, and R. Naceur, "Methods to find the number of latent skills." in *Proceedings of the 5th International Conference on Educational Data Mining*. ERIC, 2012, pp. 81–86.

[9] Y. Bergner, S. Droschler, G. Kortemeyer, S. Rayyan, D. Seaton, and D. E. Pritchard, "Model-based collaborative filtering analysis of student response data: Machine-learning item response theory." in *Proceedings of the 5th International Conference on Educational Data Mining*. ERIC, 2012, pp. 95–102.

[10] A. S. Lan, A. E. Waters, C. Studer, and R. G. Baraniuk, "Sparse factor analysis for learning and content analytics," *Journal of Machine Learning Research*, vol. 15, pp. 1959–2008, 2014.

[11] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference*. ACM, 2008, pp. 426–434.

[12] R. Bekele and W. Menzel, "A bayesian approach to predict performance of a student (bapps): A case with ethiopian students," *algorithms*, vol. 22, no. 23, p. 24, 2005.

[13] Z. Pardos and N. Heffernan, "Using hmms and bagged decision trees to leverage rich features of user and skill from an intelligent tutoring system dataset," *Journal of Machine Learning Research*, 2011.

[14] N. T. Nghe, P. Janecek, and P. Haddawy, "A comparative analysis of techniques for predicting academic performance," in *FIE'07. 37th Annual*. IEEE, 2007, pp. T2G–7.

[15] N. Thai-Nghe, L. Drumond, T. Horváth, and L. Schmidt-Thieme, "Using factorization machines for student modeling." in *UMAP Workshops*, 2012.

[16] J. Kim, P. J. Guo, D. T. Seaton, P. Mitros, K. Z. Gajos, and R. C. Miller, "Understanding in-video dropouts and interaction peaks inonline lecture videos," in *Proceedings of the first ACM conference on Learning@ scale conference*. ACM, 2014, pp. 31–40.

[17] D. J. Sheskin, *Handbook of parametric and nonparametric statistical procedures*. crc Press, 2003.

[18] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[19] M. Chiang, *Networked Life: 20 Questions and Answers*. Cambridge University Press, 2012.

[20] C.-W. Hsu, C.-C. Chang, C.-J. Lin *et al.*, "A practical guide to support vector classification," 2003.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion *et al.*, "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[22] S. Rendle, "Factorization machines with libFM," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 3, pp. 57:1–57:22, May 2012.

[23] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 1–27, 2011.