Adapting Bayesian Knowledge Tracing to a Massive Open Online Course in edX

Zachary A. Pardos, Yoav Bergner, Daniel T. Seaton, David E. Pritchard
Massachusetts Institute of Technology
77 Massachusetts Ave.
Cambridge, MA 02139
{pardos, bergner, dseaton, dpritch}@mit.edu

ABSTRACT

Massive Open Online Courses (MOOCs) are an increasingly pervasive newcomer to the virtual landscape of higher-education, delivering a wide variety of topics in science, engineering, and the humanities. However, while technological innovation is enabling unprecedented open access to high quality educational material, these systems generally inherit similar homework, exams, and instructional resources to that of their classroom counterparts and currently lack an underlying model with which to talk about learning. In this paper we will show how existing learner modeling techniques based on Bayesian Knowledge Tracing can be adapted to the inaugural course, 6.002x: circuit design, on the edX MOOC platform. We identify three distinct challenges to modeling MOOC data and provide predictive evaluations of the respective modeling approach to each challenge. The challenges identified are; lack of an explicit knowledge component model, allowance for unpenalized multiple problem attempts, and multiple pathways through the system that allow for learning influences outside of the current assessment.

Keywords

Probabilistic Graphical Models, Bayesian Knowledge Tracing, MOOC, Resource model, edX

1. INTRODUCTION

Massive Open Online Courses (MOOCs) are a quickly emerging modality of learning in higher-education. They consist of various learning resources, often lecture videos, etexts, online office hours, assessments which include homework and exams, and have a specific time in which they begin and end, often corresponding closely to that of their residentially offered counter-parts. While the efficacy of MOOCs compared to their residential offerings is an open question; from the viewpoint of educational research, MOOCs provide several substantial advantages, most notably the detailed digital trail left by students in the form of log data and the size of the student cohorts, which are often several orders of magnitude larger than typical on-campus-only offerings.

Unlike Intelligent Tutoring Systems (ITS), MOOCs do not currently provide tutorial help on demand at the points of need; instead, the knowledge is self-sought and supplied by a redundancy of information across various types of resources resulting in a variety of student selected resources and pathways through the system. This rich data provided by MOOCs presents an opportunity to investigate the efficacy of student behavior under varying conditions; however, MOOCs currently lack a model of learning with which to instrument this exploration. In this paper we will show how existing learner modeling techniques based on Bayesian Knowledge Tracing can be adapted to the inaugural course, 6.002x: circuit design, on the edX MOOC platform. We identify three distinct challenges to modeling MOOC data in section 2, followed by a description of our evaluation methodologies in section 3, and finally results of the predictive evaluations of the respective modeling approach to each challenge in section 4.

1.1 Anatomy of the MOOC

The inaugural course on the edX platform, 6.002x (Spring 2012), was a 14 week-long online course featuring video lectures in weekly sequences interspersed with lecture problems, an online textbook, a discussion forum, and a course wiki. The web interface for the course is shown in Figure 1. While the sequence of videos and problems is suggested in the form of a timeline at the top of the interface, the student can take any path through the material they choose including skipping or revisiting content.

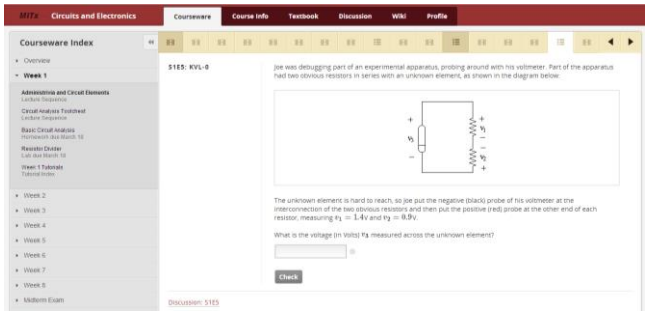

Figure 1. The interface for the 6.002x MOOC on edX. This screenshot shows a student answering a problem that is part of the Week 1 lecture sequence.

Student grades were based on 12 homework assignments and 12 virtual labs (weighted 15% for each category, with unlimited answer attempts allowed), a midterm and a final exam (30% and 40% respectively, with 3 attempts allowed). Although lecture problems did not count towards the grade, they were still marked correct and incorrect, with instant feedback as given on the homeworks. There were 289 scored elements (i.e. counting problem subparts) in 104 lecture sequence problems, 197 in 37 homework problems, 26 in 5 midterm problems and 47 in 10 final exam problems. The homework interface and scoring mechanism had some nuances that deserve elaboration.

Weekly homework assignments consisted of several problems which were all displayed on a single web page. A typical problem consisted of a figure plus several answer field "subparts" that prompted the user for input. Correctness feedback would be shown to the right of the answer fields in the form of a red "X" for incorrect (or blank answers) and a green checkmark for correct answers. This feedback was displayed after the student clicked the problem's "check" button, which simultaneously checked all answer fields within the problem. Students could answer the subparts in any order they chose however several problems' subparts required the incorporation of answers from a previous subpart. If a student answered all the subparts before their first

"check", the order in which she answered the subparts was not known, however many students elected to click the check button after each consecutive answer. Unlike most ITSs, homework was scored based on the last answer entered by the user instead of the first.

1.2 Dataset

The course drew 154,000 registrants, however; only 108,000 entered the course with around 10,000 completing the course through the final. Among those, 7,158 received a certificate for having earned at least a 60% weighted average. Our dataset consisted of 2,000 randomly chosen students from the certificate earners. A further reduction of the dataset was made by randomly selecting ten problems (and their subparts) from each of the three types of assessments; homework, lecture sequence, and exam problems.

The data for this course originated from JSON log files produced on the Amazon EC2 cloud, where the edX platform is hosted. The original log files were separated out into individual user files and the JSON records were parsed into a human readable time series description of user interaction with components of the MOOC. The final data preparation step compiled an event log by problem, consisting of one line per student event relevant to that problem. This included time spent on the event, correctness of each subpart, when the student entered or changed an answer, the attempt count of that answer, and resources accessed by the student before and between responses. An example of this data format is shown in Table 1.

Table 1. Example of the event log format of our distilled dataset

User	Res	Time	Resp1	Resp2	Count1	Count2
9	video	2m 30s	- -	- -	- -	- -
9	answer	10m 5s	correct	correct	1	1
10	book	4m 41s	- -	- -	- -	- -
10	book	40s	- -	- -	- -	- -
10	answer	20s	incorr.	- -	1	- -
10	answer	15s	incorr.	- -	2	- -
10	answer	1m 8s	incorr.	incorr.	3	1
10	answer	28s	- -	correct	- -	2
10	video	2m 10s	- -	- -	- -	- -
10	answer	6s	correct	- -	4	- -

1.3 Bayesian Knowledge Tracing

Knowledge Tracing (KT) [1] comes from the motivation to implement mastery learning [19], where every student is allowed to learn skills at his or her own pace and does not continue on to more complex material until mastery of pre-requisites has been achieved. It is based on a simplification of the ACT-R theory of skill acquisition [2] and is tasked with making this inference of mastery in the Cognitive Tutors, among other ITS. To achieve this end, simpler mastery criterion exist such as N-correct in a row to master, which is used by the ASSISTments Platform in their skill builder problem sets [3] and in the Khan Academy tutor where the term proficiency is used instead of mastery [4]. In a Cognitive Tutor, acquirable knowledge, whether declarative or procedural, is defined by fine-grained atomic pieces called Knowledge Components (KCs), typically defined by a subject matter expert. Answer steps in the tutor are tagged with these KCs and a student's past history of responses indicates his or her level of mastery of the KC. In this context, mastery is inferred to have occurred when there is a high probability (usually >= 0.95) that the KC is known by the student.

The initial KT model was not introduced as a Bayesian model; however, its formulas were found [6] to be perfectly represented by a Dynamic Bayesian Network [20], which has become the standard representation referred to as Bayesian Knowledge Tracing (BKT). The standard BKT model is defined by four parameters; prior knowledge $p(L_o)$[1], probability of learning $p(T)$, probability of guessing $p(G)$, and probability of slipping $p(S)$. Based on these parameters, inference is made about the student's probability of knowledge at time opportunity n, $p(L_n)$. The parameters and inferred probability of knowledge can also be used to predict the correctness of a student response with:

$$p(Correct_n) = p(L_n) \cdot p(\neg S) \; + \; p(\neg L_n) \cdot p(G)$$

KCs vary in difficulty and amount of practice needed to master on average, so values for these parameters are KC dependent and can be fit to training data such as log data from a previous cohort of students. Parameter fitting is often accomplished using Expectation Maximization (EM) or a grid-search of the parameters that maximizes a loss function such as sum of squared residuals of the predicted probability of a correct answer and the observed correctness. Neither fitting procedure has proved consistently superior to the other [5, 21], however; grid-search, while faster at fitting the basic BKT model, grows exponentially with the number of parameters which is a concern for extensions to BKT with higher parameterization. With both methods of parameter fitting, the objective is to define parameters that result in a projection of performance that best matches the observed data, which is the students' temporal sequence of correct and incorrect responses to questions of a particular KC.

The use of Knowledge Tracing has two stages, the stage in which the four parameters are learned, and the stage where an individual student's knowledge is being inferred from their responses. During the inference stage, the probability of knowledge at time n, given an observation, is calculated from a student's response with the following when a correct response is observed:

$$p(L_n|Correct_n) = \frac{p(L_n) \cdot p(\neg S)}{p(L_n) \cdot p(\neg S) + \; p(\neg L_n) \cdot p(G)}$$

And with the following when an incorrect response is observed:

$$p(L_n|Incorrect_n) = \frac{p(L_n) \cdot p(S)}{p(L_n) \cdot p(S) + \; p(\neg L_n) \cdot p(\neg G)}$$

The $p(L_n)$ on the right side of the formula is the prior probability of knowledge at that time, while $p(L_n|Evidence_n)$ is the posterior probability of knowledge calculated after taking an observation at that time into account. Both formulas are applications of Bayes Theorem and calculate the likelihood that the explanation for the observed response is that the student knows the KC. Since the student will be presented with feedback, there is a chance to learn. The probability the student will learn the KC from the opportunity is captured by this formula which calculates the new prior after adding in the probability of learning:

$$p(L_n) = p(L_{n-1}|Evidence_{n-1}) + p(\neg L_{n-1}|Evidence_{n-1}) \cdot p(T)$$

These formulas are used in the task of determining mastery, however; this model of knowledge has been extended to serve as a

[1] The name "$P(L_o)$" was used to denote the prior parameter in [1]. In a BKT model, this is symbolically equivalent to $p(L_1)$.

platform to study learning phenomenon [7, 8, 9]. It is this capacity for discovery that we aim to enable in MOOCs by adapting BKT approaches.

2. Model Adaptation Challenges

In order to build a foundation for measuring learning phenomena in the MOOC, several differences between MOOCs and Intelligent Tutoring Systems need to be addressed. The first is the lack of a subject matter expert mapping of the KCs associated with questions in the system. The second challenge is the attempt-until-correct scoring of the homework and lecture sequence problems. Lastly, we will address the open interface of the virtual learning environment which allows for users to take different pathways through the course which influences learning rates within a KC differently depending on path.

2.1 Lack of a KC model

The term "learning" can have broad meanings, however; in mastery contexts it is referred to with respect to a particular skill, or knowledge component being acquired. The mapping of these skills to questions, commonly referred to as a Q-matrix [10], as well as the enumeration of the skills, often comes from a subject matter expert. These skills have been referred to as cognitive operations in the psychometrics literature [11] and the processes of identification of skills is commonly referred to as cognitive task analysis in the context of ITS [12] and expert systems. Learning curves analysis [13], a KC mapping evaluation technique, asserts that evidence of a good skill mapping is a monotonically decreasing error rate across opportunities to answer questions within a skill. Similarly, fluency is expected to increase (decreasing time to solve) across correct answers to a particular skill. A unidimensional view of questions within a MOOC or a subject such as Geometry, for instance, would result in a noisy performance and fluency plot since error rates and response times would jump as soon as new topic material was introduced in the curriculum.

While subject matter expert defined knowledge components or learning objectives are planned for select future MOOC offerings, they are not common and do not exist in the 6.002x course data used in this paper. Therefore, our goal was to utilize elements of the course structure to inform a mapping of KCs to questions. We chose to leverage the problem and subpart structure of assignments, where the problem itself would serve as the KC and its subparts would be the questions belonging to the KC. The rationale for this choice was that the professor of the course often has a particular concept in mind that they wish to tap with each problem. Performance on the subparts is evidence of the student grasping this concept. The benefit to this type of mapping is that it is domain agnostic and can be used as a baseline KC model for any MOOC. The drawback is that it does not allow for longitudinal assessment of learning over more than one week since answers to a given KC will only occur within a problem in a particular week's assignment. Reduced model fit is another drawback as Corbett & Conrad [14] evaluated a similar superficial mapping of questions to course problem structure and found that this indeed sacrificed achieving more systematic, smother learning curves. Nevertheless, we believe this mapping is a reasonable start which allows for phenomenon to be studied within a problem (which we coin "problem analytics") and the methods and models described here can be applied with a different KC model swapped in, derived by a subject matter expert, inferred from the data, or a hybridization of the two [15].

2.1.1 Basic model definition

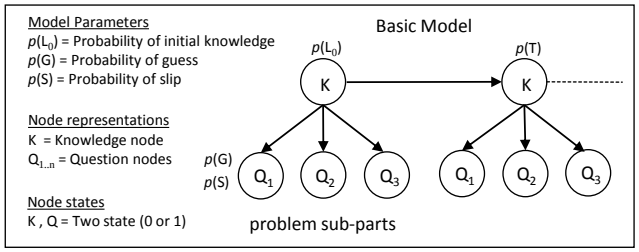

Figure 2. The basic model – a retrofit BKT model to capture answers to multiple questions in a single time slice and using homework problem as the KC. The number of parameters in this model is: **4**

Our most basic retrofitting of the BKT model to the MOOC is shown in Figure 2. In this model, which we will refer to as the "basic" model, the homework problem is the latent knowledge, K, and the observed questions are the subparts of the homework problem. When student knowledge is in the learned state this means the student has the knowledge required to answer all of the subparts. Whereas traditional application of BKT has only a single observed random variable causally linked to from the latent variable, in this model we had to accommodate for observation of multiple subpart observations at once. For example, these are the calculation steps for inferring the probability of knowledge at the second time slice when a student answers subparts one and two incorrectly on the first click of the problem check button and the third subpart correctly on the second click of the problem check button (leaving parts one and two unchanged).

First, the posterior is calculated given an incorrect answer to the first subpart: $p(L_1|Q_1 = Incorrect) = \frac{p(L_0)\cdot p(S)}{p(L_0)\cdot p(S) + p(\neg L_0)\cdot p(\neg G)}$

Next, the posterior is updated again given an incorrect answer to the second subpart:

$$p(L_1 \mid Q_1 = Incorrect \;\&\; Q_2 = Incorrect) = \frac{p(L_1 \mid Q_1 = Incorrect)\cdot p(S)}{p(L_1 \mid Q_1 = Incorrect)\cdot p(S) + p(\neg L_1 \mid Q_1 = Incorrect)\cdot p(\neg G)}$$

Steps one and two are interchangeable, including when correct and incorrect responses are observed.

The prior for knowledge at the second time slice is then calculated by applying the probability of learning to the posterior:

$$p(L_2) = p(L_1 \mid Q_1 = Incorrect \;\&\; Q_2 = Incorrect) + p(\neg L_1 \mid Q_1 = Incorrect \;\&\; Q_2 = Incorrect)\cdot p(T)$$

Finally, the posterior probability of knowledge at the second time slice is calculated given the observation of a correct answer on the third question: $p(L_2|Q_3 = Correct) = \frac{p(L_2)\cdot p(\neg S)}{p(L_n)\cdot p(\neg S) + p(\neg L_2)\cdot p(G)}$

2.2 Multiple unpenalized answer attempts

The Cognitive Tutors allow for multiple answer attempts, as does the ASSISTments Platform, however; the scoring policy for those systems is to score only the first response to each question and students are aware of this policy. The assumption is therefore that the most informative response is the first response and in a standard application of BKT, only the first responses to questions are used to train and update the model. In the MOOC, three responses are allowed on the exam problems and unlimited responses on the homework and lecture sequence problems. The scoring policy for all problems is to score the last response. Since

students are aware of this policy, it cannot be assumed that the most informative response is the first. For example, some students may decide to employ a quick heuristic on their first attempt instead of thinking through the problem as was observed among male users in an intro physics course [16]. Using only the last response is also problematic as these responses tend to have a very high percent correct, at least in homework and lecture problems, and a large amount of information would be lost in trying to model learning with only these responses. It is therefore an open and empirical research question as to where the most information exists in student answer attempts and so we define a model that allows the data to give us the answer. Past approaches have used regression to set BKT guess and slip parameters based on a host of contextual features [22], however these models, by admission, have not considered multiple attempts within a question.

Studies on test data where students are allowed multiple unpenalized attempts suggest that more information is contained in later responses (higher IRT discrimination) [17]. In addition to evaluating if a BKT model with attempt count information outperforms the basic BKT model in predictive accuracy, we also inspect the parameters of the model for each attempt count to observe if the trends seen in past studies reemerge in our data.

2.2.1 Count model definition

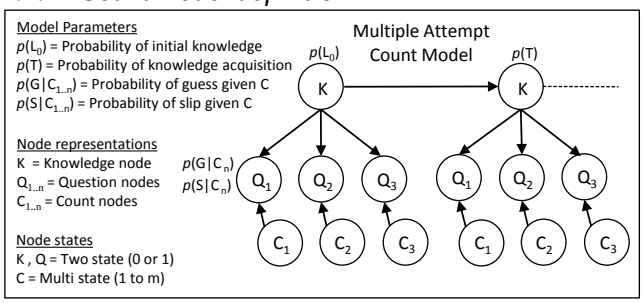

Figure 3 The count mode – conditioning question guess and slip on answer attempt count to allow information gained from responses to vary. The number of parameters in this model is: $\mathbf{2 + 2 \cdot (\#\ of\ attempt\ counts\ modeled, m)}$

The guess and slip parameters of the model dictate the amount of information gained about the latent variable from a correct or incorrect response; a guess and slip of zero in the Bayesian update calculation would mean that the value of the responses was 100% reflective of the binary state of the latent variable, while a guess and slip of 0.50 represents the maximum uncertainty regarding a response. Allowing for a different guess and slip parameter depending on attempt count therefore allows the model to capture a differing amount of information gained at each attempt. This is our modeling approach to multiple unpenalized attempts which we will refer to as the "count" model.

In the model, shown in Figure 3, count nodes, which are observable random variables, are added for every subpart since users can be on different attempt counts for different subparts. The size of the count nodes correspond to the number of attempt counts chosen to model. Inspection of the dataset showed that only ~4% of attempts were 5th attempts, therefore the size of the attempt count node was set to 6 which was also the count used for any attempt count over 6. This setting was fairly ad-hoc and could be improved upon by setting based on empirical evaluation. While the attempt count node contains a prior parameter, this was not counted as a free parameter but was instead fixed to the observed distribution of count attempts in the training data.

2.2.2 Allowing for difficulty/information gain to differ among subparts

Recent work has extended BKT to allow for different guess and slip parameters to be modeled per item in a model coined KT-IDEM (Item difficulty effect model) [3]. In ASSISTments, each problem template within a skill builder problem set was allowed to fit different guess and slip parameters, and in the Cognitive Tutor this was done at the level of the problem, where all steps of a given KC shared a guess/slip with one another within a problem but steps of the same KC that appeared in a different problem could fit different guess and slip parameter values. In both systems, prediction accuracy was improved by ~15% when there was ample data to fit each set of parameter (6 or more data points per parameter). This can be seen as allowing for variation in question difficulty among questions in a KC, or in the case of the Cognitive Tutor, allowing for variation in KC performance depending on problem context. It can also be interpreted as modulating the information gained about the latent variable depending on the question in much the same way as the count nodes in the count model modulate the information gained about the latent variable from responses depending on attempt count.

This item parameterization technique was applied to our MOOC data in the basic and count model, creating two additional models referred to as the "idem" and "idem count" model. In the idem model, the number of parameters increases to: $2 + 2 \cdot (\#\ of\ subparts)$ and in the idem count model, increases to: $2 + 2 \cdot (\#\ of\ subparts) \cdot (\#\ of\ attempt\ counts\ modeled)$.

2.3 Multiple pathways through the system

In many virtual learning environments, particularly in K-12, students complete one set of problems at a time and their path through the system is either fixed or the interface inhibits switching between problem sets. In the 6.002x MOOC, multiple problems are displayed on a page and students are frequently observed returning to a problem after answering another [18]. Besides learning from other problems, the redundancy of information found among the book pages, videos, wiki, and discussion board also allow the student to self-select his or her own path to acquiring the knowledge needed to complete the assignments. This means that influences on learning can come from a variety of sources in the learning environment, unlike most ITS where the learning can be assumed to come from feedback and tutorial help provided within the problem at hand. While this poses a challenge, in terms of capturing the variance in student learning, it also provides a rich trail of information and variety of pathways through the system that can be data mined and modeled.

Table 1 in section 1.2 illustrates how students can weave in and out of resources while answering assignment questions. It shows how one student answered a question correct after viewing a video and another student answered the same question incorrect until encountering a video, after which the hypothetical student answered the question correct. The consideration of resource influence on learning can be posed as a credit/blame inference problem, where, depending on problem answers and resource access in the aggregate, resources can be credited with learning or blamed for being ineffective. This model is at the very early stages of research and considers only resource type, which can be one of the following seven types: book, video, wiki, discussion, tutorial, answer to another problem, and answer to the problem at

hand. Up to the last five resource access events before a response are used, earlier events are discarded.

2.3.1 Resource model definition

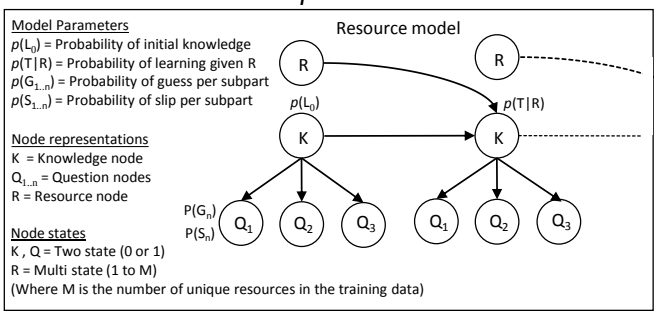

Figure 4. The resource model – based off of the idem model with resource access information added and hypothesized to influence learning. The number of parameters in this model is: $\mathbf{1 + 2 \cdot (\#\ of\ subparts) + (\#\ of\ resource\ types\ defined)}$

The resource model was built from the idem model without attempt count taken into consideration. The model is the same except for the addition of the observable resource node, R, which conditions the learning parameter, $p(T|R)$. At each time slice, the observable, R, is given the value corresponding to the current resource type being accessed. This model generalizes the idem model and can be made mathematically identical by removing all resources types except for "answer to the problem at hand", which represents the standard learning parameter capturing the benefit of feedback. When a non-problem resource is accessed, the R node gets the value of that resource type and a time slice with no question answer input is used.

3. Training and Evaluation Methodology

All five models (basic, count, idem, idem count, and resource model) were evaluated with a 5-fold student level cross-validation where the 2,000 students and their respective data were randomly assigned to one of five bins. Models were trained on the data in four bins and predictions were made on the data of the students in the fifth bin. This training/testing procedure was repeated five times, such that each bin was used once as the testing set. This evaluation procedure was run for all models on the 10 lecture, homework, and exam problems sampled in the dataset with the exception of the resource model which was only run on the homework problems. The premise of a cross-validation is to investigate if the variance captured by the models generalizes to held out data. If it does, indicated by improved predictive performance over a simpler model, the assumption is that the variance captured by the more complex model is real and reproducible. Ideally, training can be done on a previous course cohort and tested on the data of a cohort from a subsequent offering of the course. In the absence of this kind of training/test data, student level cross-validation serves as a strong substitute.

3.1 Model training details

The models used Expectation Maximization (EM) to fit parameters to the training sets with the same set of ad-hoc initial parameter values used for all models: $p(L_o) = 0.20$, $p(T) = 0.10$, $p(G) = 0.10$, $p(S) = 0.15$. Due to the data being restricted to a limited number of computing resources at the time of evaluation, a low maximum EM iteration count of 5 was set to make the cross-fold evaluations tractable in the time period allotted. Each cross-validation fold for homework took on average 12.8 hours of compute time per model running on an Intel i5 2.6Ghz machine. The lowest compute time model was the basic model with 10.7 hours per fold and the highest was the resource model with 15.1 hours per fold. Lecture and exam problems took $1/10^{th}$ the time to evaluate suggesting that more answer events occurred in the homework. For future runs, more tractable compute times could be sought with a more aggressive filtering of homework students with excessively long attempt counts or by cutting off response sequence at a particular count.

3.2 Model prediction detail

After the parameters of the model are trained, each student answer in the test set is predicted one student at a time and one time slice at a time for that student. This prediction procedure is identical to previous literature evaluating KT with the difference of accommodating for multiple responses per time slice. Walking through the prediction procedure; response data for the first student in the test set is loaded. On the first time slice, observable evidence, other than the response, is entered such as attempt counts and resource type being accessed. If there is an answer recorded for one or multiple subparts in the first time slice, the model is told which subpart or subparts were answered and makes a prediction of the student's response(s) based on the parameters learned from the training set. There will always be at least one response in each time slice except for in the case of the resource model where a time slice can represent a resource access. This prediction is logged along with the actual response. After prediction, the model is told what the student's real responses were and the model applies the Bayesian update formula to calculate a posterior and then applies the learning transition formula to calculate the new prior for the next time slice. This processes is repeated until the end of the student's response sequence and the next student is evaluated. Past answers of a student in the test set are used to predict their responses in the next time slice, however; student responses in the test set are not used to aid in prediction of other students in the test set. This form of testing, where data is utilized temporally within an instance, is not typical among classifier evaluations, however it is a principled way of evaluating student models since a real-world implementation of the model would have the benefit of a student's past responses in order to predict future performance.

3.3 Accuracy metric used

The metric chosen to evaluate the goodness of model prediction performance is Area Under the Curve (AUC) also known as Area under the Receiver Operator Curve (ROC). The metric is also equivalent to A' (A-prime). It measures a classifier's ability to discriminate between binary classes, in our case - between incorrect and correct responses. It is an accuracy metric which ranges from 0 to 1, where 1 represents perfect discrimination between responses, 0 represents perfectly inaccurate discrimination (always the opposite of the real value), and 0.50 represents classifier performance that is no better than chance. Approximations are often used to calculate AUC, such as approximate integration under the true positive vs. false positive plot of classifier performance, however the exact calculation provides a much improved intuition for the metric. To calculate AUC exactly: enumerate every possible pairing of positive and negative examples (#positive examples * #negative examples). For each pair, check if the classifier's prediction of the positive example is higher or equal to the negative example. The AUC is the percentage of the pairings in which this is true. The AUC metric is therefore a type of ranking metric. So long as all positive class predictions are higher than negative class prediction, a perfect AUC score will be achieved. Deviations from this perfect ranking result in lower AUC. This evaluation makes the AUC

metric favorable for detecting differences between two models' ability to discriminate between a correct and incorrect answer.

We calculated AUC for each problem by comparing all predicted responses to the actual responses to subparts of the problem. Since the models in our studies are primarily being used to study learning and performance phenomenon in the aggregate, we used this evaluation instead of the equally employed evaluation of averaging over student AUCs per problem [5]. The per student evaluation is more appropriate when a model's performance at the individual student mastery prediction task is of primary concern.

4. Results

Summarized cross-validated prediction results of the four models; basic, count, idem, and idem count, are presented in this section for the three problem types; homework, lecture, and exam. In addition to predicting our 2,000 sampled students, the models are also evaluated on a smaller 200 student sample to test the reliability of the results with less data. These results are summarized in the next subsection. An analysis of the count model parameters is presented in section 4.2 followed by a deeper analysis of the IDEM model in section 4.3. A two-tailed paired t-test over problems was used to test if the difference in AUC scores between models was statistically reliably different.

4.1 All results and training using less data

A review of the models, their salient features, and number of parameters is shown in Table 2. Results of predicting the 3 problem types with the four models with sample of 200 and 2,000 students are shown in Figure 4. We will first discuss the results of evaluating the 2,000 student sample.

Table 2. Model name, parameters, and description addressing how the challenges described in section 2 were addressed.

Model	Description
basic	Lack of KC model addressed by defining problems as KCs and their subparts as questions of the KC. Retrofit BKT model to allow responses to multiple questions in a single time slice. *parameters = 4*
count	Basic model extended to capture possible variation in information gained from responses due to unpenalized multiple answer attempts. *parameters = 2+2·(#counts)*
idem	Basic model extended to account for variation between questions within a KC (subparts within a problem). *parameters = 2+2·(#subparts)*
idem count	IDEM model with multiple attempt extension. *parameters = 2+2·(#subparts) ·(#counts)*

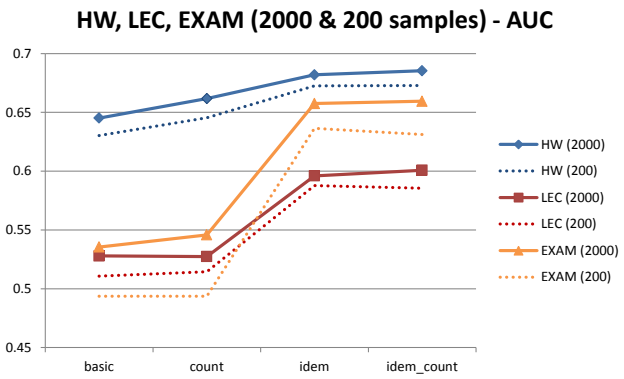

Figure 5. Cross-validated AUC results of the four models by problem type and amount of student data used.

The basic model scored an AUC of 0.6451 on homework, 0.5279 on lecture problems, and 0.5355 on exams. The homework score rivals scores achieved applying BKT to Cognitive Tutor (0.6457) and ASSISTments (0.6690) data [3]; however, the lecture and exam scores are not far above the performance of random chance. A potential upper bound benchmark for the model results presented is 0.7693, achieved in ASSISTments using a blended combination of classifiers [5].

Accounting for different information gain depending on attempt count (count model) resulted in a small but statistically significant gain in the homework (+0.0167 AUC, p = 0.008) but no statistically significant change in lecture or exam prediction performance.

Allowing for individual item guess and slip parameters (idem model) to account for differences among questions in our problem-subpart KC model resulted in the largest and most significant improvement among models. Performance improved in homework (+0.0368 AUC, p = 0.002), lecture problems (+0.0681 AUC, p = 0.013), and most prominently in exams (+0.1220 AUC, $p < 0.001$).

Adding the count extension on top of the idem model did not result in any statistically significant performance improvement.

Evaluation of all models using 1/10th the number of users resulted in the same relative model performance trends as with the complete sample. This gives us confidence in the reproducibility of the results. Overall predictive performance with the smaller sample decreased most in exam prediction, followed by lecture and homework.

The early stage resource model failed to show gains. In fact, while better than the basic model, it was statistically significantly lower performing than the idem model it was extended from (-0.009 AUC, p = 0.008).

4.2 Count model

There was no effect in modeling attempt count in exams, perhaps because exam attempts were limited to three. There was also no effect on lecture sequence problems, possible because lecture problems are ungraded and students make fewer attempts. However, a small improvement was found on homework, where the most multiple attempt behavior is observed.

We look at the guess and slip parameters of the count model for each answer attempt count to observe at which attempt the most information is being gained. We averaged the learned guess and slip values over the 5 training folds and found that slip stayed almost stationary (+/- 0.02) around its initial parameter value

while the guess value precipitously declined with attempt count. The average guess values for each homework problem are plotted in Figure 6.

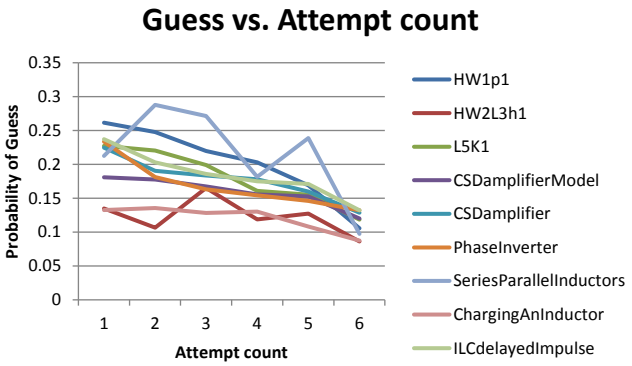

Figure 6. Learned guess parameters at each attempt count for problems in the homework.

A lower guess value means that the model can gain more confidence that the student knows the KC after observing a correct answer. Higher average guess values on the first attempt than the 6th attempt could suggest that students who struggle for longer before answering correctly are more likely to know the KC. Alternatively, lower guess values with attempt counts could simply be returning the student to the same probability of knowledge as when he began the sequence (which had been lowered due to consecutive incorrect answers).

4.3 Item difficulty model (IDEM)

The IDEM model accounted for a large amount of additional variance on top of the basic model, particularly on exam questions and least so on homework. Assuming that the 15 exam problems (midterm + final) were drafted to cover the same space of material as the 37 homework problems, it is possible that there was a higher within-problem variance among exam problem than homework problems, explaining the more dramatic improvement in modeling individual questions in exams over homework. Individual exam, lecture, and homework problem performance is shown in Figures 7, 8, and 9 respectively.

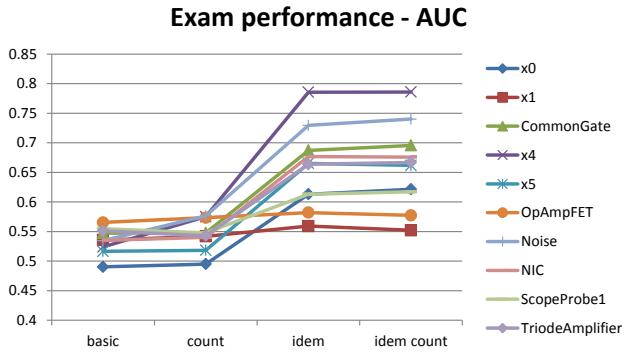

Figure 7. Individual model performance on each exam problem.

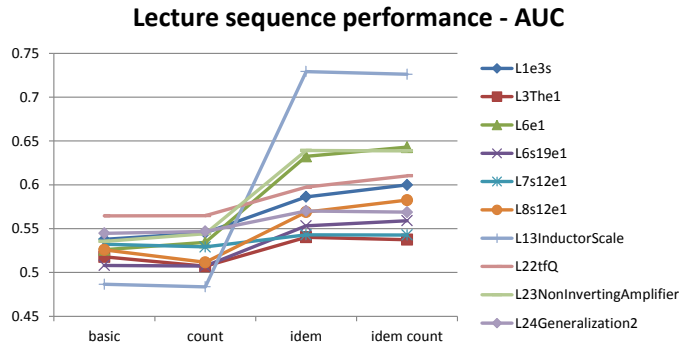

Figure 8. Individual model performance on each lecture problem

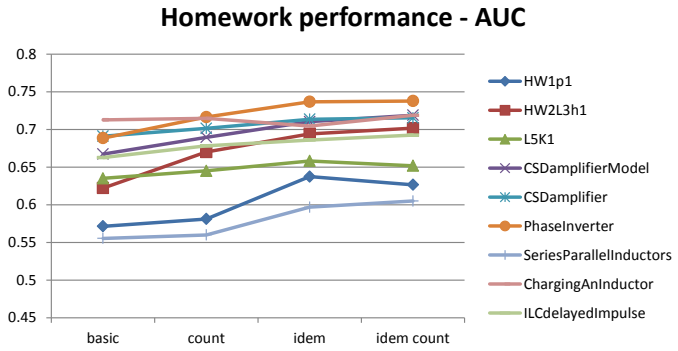

Figure 9. Individual model performance on each homework.

5. Contribution

We have presented a first foray into applying a model of learning to a MOOC. We identified three challenges to model adaptation and found that modeling variation in question difficulty resulted in the largest performance gain given our definition of KC. While our KC definition as problem with subparts as members is not ideal for measuring learning throughout the course, it nevertheless resulted in AUC performance accuracy rivaling that of prediction within systems with subject matter expert defined KC models. While we elucidated the potential for knowledge discovery given the unique variation in resource access in MOOC data, much work is left to demonstrate that this information can be seized on to produce more accurate results. This raises the question of how the efficacy of resources generalizes and the contexts and background information that needs to be considered to identify what works and for whom. Our solutions to the first two challenges, of lack of a KC model and multiple unpenalized attempt counts, will serve as an initial foundation for an efficacy assessment framework for MOOCs.

REFERENCES

[1] Corbett, A. T., & Anderson, J. R. (1995), Knowledge tracing: modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction 4(4), 253-278.

[2] Anderson, J. (1993). *Rules of the mind*. Hillsdale, NJ: Lawrence Erlbaum Associates

[3] Pardos, Z. & Heffernan, N. (2011) KT-IDEM: Introducing Item Difficulty to the Knowledge Tracing Model. In Konstant et al (Eds.) Proceedings of the 20th International Conference on User Modeling, Adaptation and Personalization (UMAP 2011). pp. 243-254.

[4] Gonul, F., & Solano, R. (2012). Innovative Teaching: An Empirical Study of Computer Aided Instruction in Quantitative Business Courses. Available at SSRN 2057992.

[5] Pardos, Z.A., Gowda, S. M., Baker, R. S.J.D., Heffernan, N. T. (2012) The Sum is Greater than the Parts: Ensembling Models of Student Knowledge in Educational Software. ACM SIGKDD Explorations, 13(2)

[6] Reye, J. (2004). Student modelling based on belief networks. International Journal of Artificial Intelligence in Education: Vol. 14, 63-96.

[7] Beck, J., Chang, K. M., Mostow, J., & Corbett, A. (2008). Does help help? Introducing the Bayesian Evaluation and Assessment methodology. In Intelligent Tutoring Systems (pp. 383-394). Springer Berlin/Heidelberg.

[8] Pardos, Z.A., Dailey, M. & Heffernan, N. (2011) Learning what works in ITS from non-traditional randomized controlled trial data. The International Journal of Artificial Intelligence in Education, 21(1-2):45-63.

[9] Rau, M., Pardos, Z.A. (2012) Interleaved Practice with Multiple Representations: Analyses with Knowledge Tracing Based Techniques. In Proceedings of the 5th annual International Conference on Educational Data Mining. Crete, Greece. Pages 168-171

[10] Tatsuoka, K.K. (1983) Rule space: An approach for dealing with misconceptions based on item response theory. Journal of Educational Measurement, 20, 345-354.

[11] Spada, H., & McGaw, B. (1985). The assessment of learning effects with linear logistic test models. Test design: New directions in psychology and psychometrics, 169-193.

[12] Lovett, M. C. (1998). Cognitive task analysis in service of intelligent tutoring system design: A case study in statistics. In Proceedings of the Fourth Conference on Intelligent Tutoring Systems. pp. 234-243. Springer-Verlag.

[13] Martin, B., Mitrovic, T., Mathan, S., & Koedinger, K.R. (2011). Evaluating and improving adaptive educational systems with learning curves. User Model User-Adap Inter (2011) 21:249–283.

[14] Anderson, J. R., F. G. Conrad, and A. T. Corbett (1989) Skill acquisition and the LISP Tutor. Cognitive Science, 13, 467-505.

[15] Koedinger, K.R., McLaughlin, E.A., & Stamper, J.C. (2012). Automated student model improvement. In Proceedings of the Fifth International Conference on Educational Data Mining. pp. 17-24.

[16] Kortemeyer, G. (2009). Gender differences in the use of an online homework system in an introductory physics course. Physical Review Special Topics-Physics Education Research, 5(1), 010107.

[17] Attali, Yigal. "Immediate Feedback and Opportunity to Revise Answers Application of a Graded Response IRT Model." Applied Psychological Measurement 35.6 (2011): 472-479.

[18] Bergner, Y., Pardos, Z.A., Seaton, D., Pritchard, D.E. (under review) Two steps forward one step back: analysis of out of sequence problem checking behavior in the edX system. Submitted to the 16th International Conference on Artificial Intelligence in Education.

[19] Bloom, B. S. (1968) Learning for mastery. In Evaluation Comment, 1. Los Angeles: UCLA Center for the Study of Evaluation of Instructional Programs.

[20] Murphy, Kevin Patrick. (200) Dynamic bayesian networks: representation, inference and learning. Diss. University of California.

[21] Gong, Y., Beck, J.E., Heffernan, N.T., 2010. Comparing Knowledge Tracing and Performance Factor Analysis by Using Multiple Model Fitting Procedures. In Proceedings of the 10th International Conference on Intelligent Tutoring Systems, 35-44.

[22] Baker, R.S.J.d., Corbett, A.T., Aleven, V. (2008) Improving Contextual Models of Guessing and Slipping with a Truncated Training Set.Proceedings of the 1st International Conference on Educational Data Mining, 67-76.