Limits to Accuracy: How Well Can We Do at Student Modeling?

Joseph E. Beck
Computer Science Department
Worcester Polytechnic Institute
josephbeck@wpi.edu

Xiaolu Xiong
Computer Science Department
Worcester Polytechnic Institute
xxiong@wpi.edu

ABSTRACT

There has been a large body of work in the field of EDM involving predicting whether the student's next attempt will be correct. Many promising ideas have resulted in negligible gains in accuracy, with differences in the thousandths place on RMSE or R^2. This paper explores how well we can expect student modeling approaches to perform at this task. We attempt to place an upper limit on model accuracy by performing a series of cheating experiments. We investigate how well a student model can perform that has: perfect information about a student's incoming knowledge, the ability to detect the exact moment when a student learns a skill (binary knowledge), and the ability to precisely estimate a student's level of knowledge (continuous knowledge). We find that binary knowledge model has an AUC of 0.804 on our sample data, relative to a baseline PFA model with a 0.745. If we weaken our cheating model slightly, such that it no longer knows student incoming knowledge but simply assumes students are incorrect on their first attempt, AUC drops to 0.747. Consequently, we argue that many student modeling techniques are relatively close to ceiling performance, and there are probably not large gains in accuracy to be had. In addition, knowledge tracing and performance factors analysis, two popular techniques, correlate with each other at 0.96 indicating few differences between them. We conclude by arguing that there are more useful student modeling tasks such as detecting robust learning or wheel-spinning, and estimating parameters such as optimal spacing that are deserving of attention.

Keywords

Cheating experiments, student modeling, limits to accuracy, knowledge tracing, performance factors analysis

1. INTRODUCTION

The field of educational data mining has seen many papers published on the topic of student modeling, frequently predicting next item correctness (e.g. [1-6]). Next item correctness refers to the student modeling task where the student's past performance on this skill is known, and the goal is to predict whether the student will respond correctly or incorrectly to the current item. This task was the topic of the KDD Cup in 2010. It is typically assumed that data from other students are also available to aid in fitting modeling parameters. This research area certainly appeared to be ripe grounds for rapid improvement, with reported R^2 values for Performance Factors Analysis (PFA; [7]) and Bayesian knowledge tracing [8] of 0.07 and 0.17, respectively [9]. PFA and Bayesian knowledge tracing were two better known, baseline techniques, and their apparent poor performance left tremendous room for improvement by developing more refined modeling techniques.

Researchers tried a variety of approaches to improve accuracy. One natural idea was to consider awarding students partial credit for their attempts. Many researchers use a simple, binary scoring metric of full points for a student who responds correctly on the first attempt with no hints, and zero points for a student who makes any mistakes or requests any hints. Thus, there is no distinction between a student who makes a mistake and corrects himself 3 seconds later, and a student who asks the system to tell him the answer and types it in — both are simply marked as "incorrect." Work on partial credit decreased the amount of credit awarded in proportion to the number of hints requested [3]. By accounting for student partial credit, it improved model accuracy from an R^2 of 0.1903 to 0.1922[1].

Another potential weakness in student models is that the domain models are developed by human experts, who are often guided by intuition. Perhaps an approach that uses data to automatically refine student models will result in a better fit to the data? Across eight datasets where model accuracy was available for the original and the data-generated models, the model fit (un-weighted average, computed by the authors) improved slightly from 0.4143 to 0.4020. However, perhaps the primary outcome of the work was better estimates of the rates at which students learn skills, which is certainly a useful artifact.

Some approaches were possibly larger successes. One underlying assumption is that there is one set of model parameters. For example, all students have the same initial knowledge of a particular skill; all students learn the skill at the same rate, etc. Relaxing that assumption and modeling students as two separate distributions improved R^2 from 0.162 to 0.205, and AUC from 0.74 to 0.77[10]. However, to the authors' knowledge, no one has tried to replicate this work on another dataset, so the results should be treated with skepticism.

Many techniques assume that all students have the same initial knowledge of a particular skill. Such an assumption is clearly incorrect, as student knowledge typically varies considerably. So why not incorporate such flexibility into our models? Some interesting work on extending knowledge tracing allowed student initial knowledge to vary based on initial performance [11]. The main finding was that model fit was notably improved, from an R^2 of 0.0374 to 0.1236. However, on replication, this approach of customizing initial student knowledge was found to perform worse than the baseline knowledge tracing technique with an R^2 of 0.089 vs. 0.1257[2][12]. This later study was also interesting in that it tested different techniques for estimating model parameters,

[1] Note that RMSE, R^2 and AUC values are not comparable across studies due to differing datasets.

[2] The R^2 statistics for both studies were computed by the authors of this paper for consistency.

of the three attempts studied, model fit varied from an R^2 of 0.1203 to 0.1257. There had been prior work experimenting with different methods of parameter estimation with conflicting results about which approach worked better [9, 13]. We find ourselves agreeing with the authors of [12] that "It is not yet clear what features of a specific data set (and the tutor it comes from) are associated with better or worse performance for specific types of student models." By creating a machine-learned ensemble of student models and features, they managed to improve A' from 0.705 to 0.769 [12]. This is a definitely large improvement in model accuracy, but raises questions of interpretability, which we will discuss later in the paper.

A fair question is why improvements in student model accuracy have been so limited? In general, improvements in model accuracy have been minimal, particularly given the relatively low baseline performances. Improving a model with an R^2 of 0.9 is challenging, but improving one that starts at an R^2 of 0.17 should be simpler. The ideas listed above were sensible, but improvements have generally been modest, and often do not replicate across data sets. The results generate two questions:

1. What is it about this prediction problem that makes it difficult?
2. Is there perhaps a much lower upper limit on model accuracy than might otherwise be suspected?

The motivation for this paper was to explore potential reasons behind the inability to create highly accurate models.

2. CHEATING EXPERIMENTS WITH THEORETIC MODELS

Our first investigation into the plausible performance ceiling of student modeling is done using *cheating experiments*. The idea of a cheating experiment is to test a methodology, simulating some non-existent technology as part of it as a means of discovering how well a technique would perform if certain limitations are removed. The key element of a cheating experiment is relaxing certain limitations in scientific knowledge or methodology, but not to create an artifact that is too powerful. For example, for our task of predicting student correct next response, one cheating experiment would be an algorithm that simply peeks into the future, and predicts whatever the student will do. While this approach would certainly be very powerful, it does not give us much guidance about limiting factors on performance as the only conclusion one could draw would be "a student modeling technique that could see the future with perfect accuracy would do a very good job." Therefore, we focus on more limited, but still currently infeasible, extensions of a student model's capabilities.

This paper investigates three aspects of student modeling. First, we explore how a student model would do with a perfect detector of learning [14]. Second, we investigate how important understanding student incoming knowledge is. Third, we examine how a continuous estimate of knowledge would perform.

Thus, the goal of our analysis is to estimate how well we could perform at student modeling if we had a perfect model of several aspects of learner cognition. However, we first give our baseline assumptions, then describe our data, and finally provide baseline model performance when trained on those data.

2.1 UNDERLYING ASSUMPTIONS

We were interested in understanding what factors limit our ability to model the student. One input, typically implicit in student modeling research, is the domain model the maps items to skills (sometimes called a "Q-matrix"). This model enables us to map student performance on an item to a particular skill in the domain. If this aspect of the system is poorly done, model accuracy can suffer. The authors are unaware of any large gains in accuracy by refining a "reasonably constructed" transfer model; we add the restriction as we are certain that refining a randomly generated transfer model would improve accuracy. Recent work [6] found a slight improvement in accuracy from refining models, but the effect was not large. Therefore, we do not consider improvements in the transfer model within this paper.

Furthermore, we assume that we do not know the underlying model generating students' responses. While it is certainly possible to make such an assumption and to use, for example, knowledge tracing, to generate student responses, such an approach assumes far too much, and is of questionable applicability to real-world tutoring scenarios. Therefore, rather than generate hypothetical student responses and estimate our ability to recover our initial models, we simply use the student performance data as provided and compute our predictive accuracy.

2.2 DATA DESCRIPTION AND BASELINE MODEL

For our analyses, we use two datasets. The first is from the ASSISTments (www.assistments.org) web-based tutor. These data are from 343 eighth-grade students (approximately 13 years old) in four classes in urban school districts in the Northeastern United States. There were 86,528 first attempts at responding to a mathematics problem, and students were correct 64.5% of the time. The domain is represented as 104 mathematics skills. Our second dataset is the 2010 KDD Cup dataset, from the Cognitive Algebra Tutor. We used one of the training datasets, and filtered out rows with missing values resulting in 607,026 rows of data with students correct 75.5% of the time. These data are from 574 students working on 158 skills in mathematics. Although both systems involve math skills, they are actually rather different from each other. ASSISTments serves primarily as computer-assisted practice for students' nightly homework and review lessons, while the Cognitive Tutor is part of an integrated curriculum and has more support for learners during the problem-solving process.

For our baseline approach, we have selected the Performance Factors Analysis (PFA; [7]) model. A PFA model takes the form of a logistic regression model, where the independents are the number of correct and incorrect responses, and the difficulty of the item the student is attempting. As PFA estimates the impact on performance by weighting types of learning opportunities differently (correct vs. incorrect responses), it can be seen as a variant of learning decomposition [15]. For our data, we have found that PFA typically does a better job at predicting [9] the data than Bayesian knowledge tracing [8]. Thus it is more appropriate as a baseline metric. For both our baseline approach and our cheating models, we represent data separately for each student on each skill. Thus, when we discuss successive student attempts, we mean successive attempts on the same skill, and ignore intervening problems on other skills.

For performance metrics, we use the Area Under the Curve (AUC) and R^2. AUC is an approximation of A'; it is a commonly used metric when comparing student modeling techniques. The upper limit on AUC is 1.0, and the practical lower limit is 0.5[3]. AUC evaluates techniques based on how well they *order* their predictions. For four problems, if a model predicts that a student has a 95%, 90%, 87%, and 86% chance of responding correctly and the student gets the first two items correct and the next two items incorrect, the AUC will be a perfect 1.0 (assuming a threshold of 50% is used, which would be a poor choice in this scenario). Even though the model predicts the student is likely to get the last two items correct, since those items are *relatively less* likely to be correct than the first two, AUC gives a perfect score. R^2 is based on the squared error between the predicted and actual value, but is normalized relative to the variance in the dataset. A perfect R^2 value is 1.0, while 0 is a lower bound for (non-pseudo) R^2. R^2 is similar to Root Mean Squared Error (RMSE), but is more interpretable due to the normalization step. For example, it is unclear whether an RMSE of 0.3 is good or bad, perhaps a better error could be obtained simply by predicting the mean value? However, an R^2 of 0.8 indicates the model is account for most of the variability in the data. For computational simplicity, we do not use the pseudo- R^2 method such as Nagelkerke in this paper. Neither AUC nor R2 is a perfect evaluation metric, but, combined; they account for different aspects of model performance (relative ordering, and absolute accuracy, respectively) and provide us a basis for evaluating our models.

Table 1 shows performance of the baseline PFA model on both the ASSISTments and KDD Cup data. We can see that the model does not fit the KDD Cup data set as well as the ASSISTments data. Also, the AUC scores are reasonably high, indicating PFA is able to order its predictions relatively well. However, the lower R^2 values indicate the magnitude of the errors is still substantial.

Table 1. Performance of baseline PFA model

Data source	AUC	R^2
ASSISTments	0.745	0.170
KDD Cup	0.713	0.100

2.3 CHEATING MODEL 1: A PERFECT DETECTOR OF LEARNING AND INITIAL KNOWLEDGE

Our first cheating experiment involves what would be a useful piece of technology: a *perfect* detector of the moment of student learning [14]. When a student is practicing a skill, there is hopefully an "aha!" moment where the student has a large jump in understanding the skill. This cheating model simulates the ability to detect such learning. In addition, the model is also aware of whether the students begin using the tutor with knowledge of the skill. Even though it has no data about the student on this skill, it can apply its learning detector on the first attempt when the student sits down. This cheating model behaves by examining the student's next response, and if the student will respond correctly, this model will mark the student as just learned the skill, and predict a correct response.

However, this model is not permitted to cheat for *unlearning* or forgetting. For example, if the model believes the student knows the skill, it will predict a correct response. If the next response is incorrect, it must first make an error on that response by predicting correct, and then may reevaluate whether the student really knows the skill. To make this decision, the cheating model is permitted to peek ahead at the future student responses and decide whether it wants to change its mind about the student's knowledge and mark him as not knowing the skill.

Although this procedure may sound baroque, there are two reasons for it. First, a cheating model that perfectly detected learning as well as forgetting would never make a mistake, and not produce a useful result. Second, most student modeling approaches focus on learning, and ignore the impact of forgetting [8] (with a few exceptions, such as [2]). Since forgetting is often caused by interference [16], and it is difficult to know all the relevant stimuli to which the student is exposed, this aspect would be difficult to model. To be clear, although Cheating Model 1 (CM1) does not have a perfect detector of forgetting, it is not monotonic in its predictions. That is, when presented with evidence the student does not know the skill (i.e., an incorrect response), it is permitted to backtrack in its estimate of student knowledge.

For an example of how CM1 performs, see Table 2. For the first item, CM1 is permitted to know whether the student knows the skill, and so predicts an incorrect student response. For the second item, CM1's perfect detector of learning enables it to realize the student has learned the skill (by peeking ahead at the next response), and so it predicts a correct student response. Two items later, the student makes a mistake, and as CM1 believes the student knows the skill, the model makes an incorrect prediction (bold, underlined entry). The model then peeks ahead to determine whether it is better to ignore this transient slip, or whether it should change its mind about whether the student knows the skill. Since there is a second incorrect response, CM1 can improve its accuracy by believing the student does not know the skill. Thus, CM1 adjusts its predictions to be in best accordance with future student data, but is not permitted to predict the forgetting before it receives direct evidence.

Table 2. Predictions for CM1 and CM2 on sample student performance data. Bold underlined entries indicate incorrect predictions

Correct?	CM1	CM2i	CM2c	CM2m
0	0	0	**<u>1</u>**	**<u>1</u>**
1	1	1	1	1
1	1	1	1	1
0	**<u>1</u>**	**<u>1</u>**	**<u>1</u>**	**<u>1</u>**
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1

For a more formal definition, Figure 1 provides the pseudocode for the Cheating Model #1 (CM1).

On the ASSISTments data, this initial cheating model has an AUC of 0.804 and an R^2 of 0.50. On the KDD Cup dataset, it has an AUC of 0.762 and R^2 of 0.453. Our cheating model clearly outperforms PFA (as it should) on both datasets; AUC is increased by 0.5 or 0.6, and R^2 by 0.35.

[3] Although technically AUC can be below 0.5, in that case the model's accuracy would be improved, and the resulting AUC would be above 0.5, simply by inverting all of its predictions.

```
Cheating Model 1

Input: Student performance data d

for student in d do
    for skill in d[student] do
        previous_prediction = none;
        for problem in d[student][skill] do:
            if (previous_prediction is none) then
                prediction = response;
            else if (previous_prediction == 0) then
                prediction = response;
            else if (previous_prediction == 1) then
                if (previous_prediction was right) then
                    prediction = 1;
                else
                    prediction = response;
                end if
            end if
            previous_prediction = prediction;
        end for
    end for
end for
```

Figure 1. Pseudocode for Cheating Model 1 (CM1)

2.4 CHEATING MODEL 2: WHAT IS THE IMPACT OF KNOWING STUDENT INCOMING KNOWLEDGE?

The first cheating experiment provided substantial gains in model performance. However, these gains are a result of two pieces of non-existent technology: a perfect detector of learning, and a perfect detector of incoming knowledge. What if we ablate the model slightly, and remove its ability to know student incoming knowledge? Thus, our model will retain its ability to detect "aha!" moments by the learner, but cannot necessarily correctly predict the student's first attempt.

To handle imperfect first prediction, we consider three baseline models:

1. A model that assumes the student knows all skills so will respond Correctly (CM2c) on first attempts.
2. A model that assumes the student knows nothing, so will respond Incorrectly (CM2i) on first attempts.
3. A model that assumes the student answers correctly on easy items, instantiated as those items that are answered correctly a Majority (CM2m) of the time across all students.

Table 2 provides an example of how CM2c, CM2i, and CM2m perform relative to CM1. As can be seen, the only difference is on how each of these models predicts the first element. Since CM2c predicts the first response will be correct, it makes an error in prediction. For this example, we have arbitrarily assigned the first item a difficulty of 0.4, so since the majority of students get this item correct, CM2m predicts the current student will. For a more formal definition, the pseudocode for CM2 can be seen in Figure 2

The impact of prior knowledge for the ASSISTments data may be seen in Table 4; assuming the student will respond incorrectly on the first attempt is the best (simple) approach for prediction. For the ASSISTments dataset, the drop-off in accuracy is noticeable: AUC drops from 0.804 to 0.747, just slightly better than the baseline PFA model's 0.745. Similarly R^2 drops from 0.5 to 0.239—a very substantial drop, and moderately better than PFA's 0.17. For ASSISTments, understanding student initial knowledge is important.

In the KDD Cup dataset, as can be seen in Table 5, the results were broadly similar to those for ASSISTments, with one major points of divergence: the impact of understanding student first problem performance is much less dramatic. For the KDD Cup data, the AUC only drops from 0.762 to 0.754. One explanation is in this dataset there are relatively more problems solved per student per skill. Within ASSISTments, on average each student solves 4.7 problems on each skill he works on. In the Cognitive Algebra Tutor, students practice 19 problems per skill. Thus, since more problems are solved by each student in each skill in the Cognitive Algebra Tutor, initial knowledge estimation has a smaller impact on accuracy. ASSISTments is probably on the lower end of amount of practice per skills; while the Cognitive Tutors, due to the integrated curriculum, is probably on the higher end. Here we see another example of the point made in [12] that there is often inconsistency in approaches across datasets. A hypothesis consistent with our data is that a good model of incoming student knowledge is more useful in scenarios when there are fewer data per skill. We suspect modeling prior knowledge is also more effective when students are more heterogeneous; if all students (don't) know a skill, there is little point in modeling their incoming knowledge separately.

```
Cheating Model 2:

Input: Student performance data d

for student in d do
    for skill in d[student] do
        previous_prediction = none
        for problem in d[student][skill] do:
            if (previous_prediction is none) then
                if CM2c:
                    prediction = 1;
                else if CM2i
                    prediction = 0;
                else if CM2m
                    if (problem_difficulty >= 0.5) then
                        prediction = 0;
                    else
                        prediction = 1;
                    end if
                end if
            else if (previous_prediction == 0) then
                prediction = response;
            else if (previous_prediction == 1) then
                if (previous_prediction was right) then
                    prediction = 1;
                else
                    prediction = response;
                end if
            end if
            previous_prediction = prediction;
        end for
    end for
end for
```

Figure 2. Pseudocode for Cheating Model 2 (CM2)

2.5 CHEATING MODEL 3: ESTIMATING CONTINUOUS KNOWLEDGE AND PERFORMANCE

The final cheating model takes a clue from CM2m, which bases its predictions on item difficulty. Rather than simply assuming that a learner knows a skill or does not, CM3 maintains a *degree of knowledge* for each learner. Our semantics are that knowledge and difficulty are both in the range [0,1]. Larger values represent higher degrees of learner knowledge and more difficult items. If item difficulty is less than or equal to knowledge, this model maintains the learner will respond correctly. Otherwise the learner will respond incorrectly to the item. In this manner, it can represent a student who can respond correctly to some items within a skill, but get other items wrong. The intuition is that the model raises knowledge just high enough to account for student correct responses. On observing an incorrect response, it has the option to decrease student knowledge. The reasoning is similar to that for CM1: a model that could increase and decrease knowledge estimates at will (before seeing the student's response) would achieve perfect accuracy. When a student answers an incorrect response, it can decrease its knowledge estimate arbitrarily low, and will lower it enough to account for later incorrect responses.

Table 3 shows how CM3 performs given the same student performance data as before, but also incorporate item difficulty information. CM3, like CM1, is permitted to peek ahead on the first student performance. Since the student responds incorrectly, the student's knowledge is set to be just under what is required. Since the student responds correctly to the next item, the knowledge is increased to 0.7 to be just sufficient. The student responds incorrectly to the next item, but its difficulty is higher than the student's knowledge, so CM3 predicts the student will get the item wrong and no update to student knowledge is required. Two items later, the student responds incorrectly to an item of difficulty 0.65. This response causes the model to make a mistake as this item is lower than the student's knowledge of 0.7. CM3 responds by decreasing the knowledge, not to .649, but to 0.599. The reason is that CM3 looks ahead, and determines what level of knowledge will best predict the current streak of incorrect responses, and sets knowledge to the maximal level. Since the student gets the next item, with a difficulty of 0.6, incorrect, knowledge is set just below that point. A formal definition of CM3 is provided in Figure 3.

Table 3. Example of predictions and updating knowledge estimates for CM3. Bold underlined entries indicate incorrect predictions

Correct?	Item difficulty	Prediction	Knowledge estimate
0	0.4	0	0.399
1	0.7	1	0.7
0	0.8	0	0.7
1	0.6	1	0.7
0	0.65	**<u>1</u>**	0.599
0	0.6	0	0.599
1	0.3	1	0.599

The performance of CM3 on the ASSISTments dataset is seen in the first row of Table 4. CM3, due to its ability to incorporate continuous levels of knowledge, is the strongest performer on the ASSISTments dataset by a large margin. Apparently representing knowledge as a binary value, even with a model with a perfect detector of learning, results in a considerable weakness. Representing gradations of student knowledge appears to be much more effective.

The performance of CM3 on the KDD Cup data is seen in the first row of Table 5. Again, continuous knowledge resulted in strong performance. For the KDD Cup data, we were a bit stymied as to the meaning of "item difficulty". For these results, we used a concatenation of problem name and step name. However, many such pairs were only attempted by 1 student, leading to considerable over-fitting. Using just the problem name suffers from the problem of underspecificity, and gives an AUC and R^2 of 0.798 and 0.442, respectively.

Table 4. Full performance results on ASSISTments data

	Initial knowledge	Continuous knowledge	AUC	R^2
CM3	Known	Yes	0.884	0.634
CM1	Known	No	0.804	0.5
CM2i	Assume incorrect	No	0.747	0.239
PFA			0.745	0.17
CM2m	Based on difficulty	No (except first item)	0.724	0.273
CM2c	Assumed correct	No	0.678	0.266

Cheating Model 3:

Input: Student performance data d

```
for student in d do
    for skill in d[student] do
        knowledge = first_response == 1? first_response.difficulty : first_response.difficulty -.001;
        for problem in d[student][skill] do:
            if (knowledge > difficulty) then
                prediction = 1;
            else
                prediction = 0;
            end if
            if (response == response.previous and response.previous is not none) then
                continue;
            else if (response == 1)
                next_response = response.next;
                while (next_response == 1)
                    knowledge = max (knowledge, next_response.difficulty);
                     next_response = next_response.next;
                end while
            else if (response == 0)
                next_response = response.next;
                if (next_response == 0)
                    while (response == 0)
                        knowledge = min (knowledge, next_response.difficulty - .001)
                        next_response = next_response.next;
                    end while
                else
                    while (next_response == 1)
                        knowledge = max (knowledge, next_response.difficulty)
                        next_response = next_response.next;
                    end while
                end if
            end if
        end for
    end for
end for
```

Figure 3. Pseudocode for Cheating Model 3 (CM3)

3. EMPIRICAL CHEATING EXPERIMENTS

In addition to the theoretic cheating experiments, we also examine data from recent work [12] on ensembling multiple techniques together. This dataset[4] is of interest as it provides the predictions of multiple student modeling techniques as a means of estimating

[4] Kindly provided by the paper authors.

an upper bound on performance. For this work we focus on four approaches to Bayesian knowledge tracing (BKT), and a PFA model. The first two BKT variants use different search techniques to estimate the model parameters; the third approach restricts the training data to find more relevant cases; the fourth variant extends the BKT model by allowing each student to have a different level of initial knowledge:

1. Brute force (BF): estimates model parameters by exhaustively searching the set of initial knowledge, learning, guess, and slip parameters.
2. Expectation maximization (EM): finds the model parameters that maximize the data likelihood.
3. Less data (LD): a variant of knowledge tracing that uses fewer training data.
4. Prior per student (PPS): rather than assuming all students have the same prior knowledge for a skill, it makes initial knowledge conditional on student performance.

Table 5. Full performance results on KDD Cup data

	Initial knowledge	**Continuous knowledge**	**AUC**	**R^2**
CM3	Known	Yes	0.887	0.673
CM1	Known	No	0.762	0.453
CM2i	Assume incorrect	No	0.754	0.353
CM2m	Based on difficulty	No (except first item)	0.713	0.357
PFA			0.713	0.1
CM2c	Assumed correct	No	0.711	0.356

In the original work on ensembling from which these data derive, the authors used machine learning approaches to find the best way of combining the models' predictions to create a more accurate model. Instead of that approach, we will consider how a model that managed to always select the best base model would do. Specifically, we instantiate our model as follows:

```
If student response is correct
  then prediction = max(BF, EM, LD, PPS, PFA)
  else prediction = min(BF, EM, LD, PPS, PFA)
```

Table 6 provides some sample predictions for the five algorithms and shows how our cheating experiment behaves. For each student response, the model selects whichever prediction is closest. Unlike the earlier cheating models, this one does not have explicit assumptions about learning or initial knowledge, but simply picks whichever prediction is closest. Thus, for the third student response, a (perhaps) unexpected incorrect response, this approach simply selects the lowest value. In other words, our empirical cheating experiment postulates the existence of a perfect ensembling approach that always selects the best of its options.

We computed our model's predictions across all 178,000 rows in the provided dataset. The performance of our model and baseline techniques PFA and KT-LD (best performing of the KT techniques [12]) is shown in Table 7. The empirical cheating technique strongly outperformed the baseline techniques, and appears to be somewhat better than the best ensembling technique found, which had an A' (approximately equivalent to AUC) of 0.769. Therefore, there may be room to develop more refined ensembling techniques, and discover additional features in order to improve predictive accuracy. We discuss the utility of this line of research in the Future Work section.

Table 6. Empirical cheating experiment

Correct?	BF	EM	LD	PPS	PFA	prediction
0	0.31	0.31	0.31	0.25	0.34	0.25
1	0.60	0.60	0.59	0.60	0.60	0.60
0	0.35	0.37	0.37	0.29	0.38	0.29
1	0.46	0.47	0.47	0.42	0.47	0.47
1	0.37	0.37	0.37	0.36	0.39	0.39

Table 7. Performance in empirical cheating experiments

	AUC	R^2
Empirical cheating	0.831	0.324
PFA	0.706	0.130
KT-LD	0.701	0.126

One suggestive item in Table 6 is that it appears that each of the student modeling techniques is making fairly similar predictions to its competitors. This phenomenon was also noted when comparing techniques on another data set [17]. To test this idea, we looked across all 178,000 data points, and found that the five student modeling techniques (BF, EM, LS, PPS, and PFA) intercorrelated with each other at 0.92 on average. The prior per student (PPS) model was the most idiosyncratic (and the worst performing), with an average correlation of 0.85 with the other models. If PPS is removed, the remaining four techniques intercorrelated at 0.96, an astonishingly high value. This high number is not an artifact of comparing variants of knowledge tracing with each other: PFA's predictions correlates with KT-LD's at 0.95. It should be noted that each of these techniques has typically been the subject of multiple papers investigating its strengths and weaknesses, and exploring different variations (e.g. item- vs. skill-based PFA [5]). However, it appears the major story is that all of the techniques are in large-scale agreement with each other.

4. MAIN RESULTS AND LIMITATIONS

This paper has estimated likely upper bounds on student modeling performance. Our approach was to consider the basic cognitive factors influencing student performance, and then construct a cheating experiment that perfectly models those factors. On ASSISTments data, we found that the ability to perfectly model student learning, but imperfect information about prior knowledge, led to a model that performed only slightly better than a baseline PFA model. This result is rather surprising.

The other striking result is that, in spite of being an active research area in the EDM, AIED, and ITS communities, competing student modeling approaches make remarkably similar predictions. Given the relative closeness of empirical results to our cheating models, and the high intercorrelations, a plausible conclusion is that the majority of the work in the field of predicting next item correctness has been done, and there are not large gains in performance remaining to be found.

The cheating models described in this paper are extremely powerful, and examine the basic cognitive inputs to student performance. We are unlikely to get perfect detectors of learning any time in the near future. For student modeling approaches that rely on determining when a student has learned, trying to infer incoming knowledge, and account for item difficulty, these cheating models provide a reasonable upper limit on accuracy. However, what of approaches those are not based on cognitive principles? For example, student mistakes could be due to lack of knowledge, or could be due to a careless error. Such careless errors appear to be non-random, as such mistakes have been found to be associated [18] with gaming the system [19], and there is work on *contextual* detectors of slip and guess [13]. The potential improvement from such work is not accounted for by the analyses presented in this paper.

In addition, approaches such as collaborative filtering [20] provide an avenue for non-cognitive approaches to improving student modeling. With collaborative filtering approaches, rather than modeling student knowledge explicitly, instead the goal is to find similar past students and use their performance to make a prediction for the current student (e.g., [21]).

5. FUTURE DIRECTIONS

It is unclear how much additional gain there is from refining student models to achieve ever higher predictive accuracy. Many promising approaches have resulted in little real-world improvement in accuracy. One drawback is the seductive combination of statistical hypothesis testing with increasingly-large datasets. It is possible to find statistically reliable results corresponding to very small effects. Even with a relatively small dataset of 48,000 item responses, a result with a p-value of 0.002 resulted in an improvement of less than 0.001 in R^2 [22]. While larger datasets enable us to estimate such miniscule quantities quite precisely (thus, the low p-value), it raises the question of whether this result *useful* in any way?

We should reflect on why so much effort is being devoted to the problem of predicting student next response. Two candidate answers are that's where the data are, and this task was the goal of the 2010 KDD Cup. Certainly, correctness performance on each item for each student is a very vast source of data. Ten years ago that argument would have been a strong rationale, but now there are large quantities of educational data of all sorts. As a thought experiment, imagine a research result were published in EDM 2014 with a new student modeling approach that achieved an A' of 0.9 (comparable to an AUC of 0.9, but A' has simpler semantics). Effectively that would mean that given a correct and an incorrect student response, this student model could determine which was which 90% of the time. Such an accomplishment would be a major step forward in our capabilities. But, what would we actually *do* with the model? This question is non-rhetorical, as the authors do not have a good answer. To be clear, there are plenty of useful problems our student models could address, such as the probability of a student receiving an "A" in the course, or whether he is ready to move and learn subsequent material.

Ironically, as a field we have settled on a common test problem that has little impact on tutorial decision making or on informing the science of learning. We got to this point for good reasons. Student modeling in ITS is primarily about the estimation of student knowledge. In addition to plentiful data at the item response level, one natural method of validating [23] an instrument is to compute its predictive validity. That is, how well does the measure correlate with things the construct should correlate with. If our model of student knowledge is a good one, it should have a high correlation with student performance on items. Thus, from an instrumentation standpoint our scientific approach is reasonable.

However, while showing that a measure has a high correlation is a necessary condition in validating a measure, it is never a sufficient condition [23]. In other words, constructing a student model with a higher predictive accuracy is not sufficient to create a better estimate of the student's knowledge. As a concrete example, consider ensembling methods, which consider the outputs of different student modeling approaches, and finds a means to combine their predictions with additional features to better predict student performance. Such approaches are in fact successful at noticeably raising the bar (e.g. [12]). However, is there any interpretable component relating to student knowledge? Can we use this model to predict whether an intervention will lead to more learning? If not, then what do we do with the model?

To be clear, this paper does not assert that the field of student modeling is completed. Rather, it makes a more modest claim: the research thread of predicting next item correctness is approaching limits to accuracy, and has probably progressed beyond a useful point. However, there remain several interesting, known problems in student modeling that can inform us about student learning, and have a clear correspondence to improving tutorial decision making.

First, consider the robust learning framework of the Pittsburgh Science of Learning Center. The components of robust learning are preparation for future learning (of related skills), transfer to novel contexts, and retention. Constructing a detector of the first two components of robust learning (e.g., [24]) is a worthwhile modeling goal. Other work has focused on predicting the third component, retention (e.g., [25]).

As a second example, work on the optimal interval to wait before presenting an item on the same skill would be useful. Items presented to close together temporally waste time on repetitive practice; too far apart risks having the student forget and having to relearn [26]. However, such intervals vary by student and skill. This problem can be seen as the complement of retention: how long can we wait before risking the student will forget the item?

As a third example, detectors of student behaviors that are out of bounds of our simplified model of the learner are a useful avenue to explore. Our model is that students are attempting to solve problems, and as a result are learning a little bit each time. But what if the student is bored [27] or frustrated and discouraged [28]? A recent example of such a detector is wheel-spinning [29], named after how a student spins his wheels and goes through the motions of learning, but learning repeatedly does not occur. Detecting and suggesting remediation for, such problems is an interesting third avenue to explore.

ACKNOWLEDGMENTS

We want to acknowledge the funding on NSF grant DRL-1109483 as well as funding of ASSISTments. See here (http://www.webcitation.org/67MTL3EIs) for the funding sources for ASSISTments

REFERENCES

1. Thai-Nghe, N., T. Horváth, and L. Schmidt-Thieme. 2011. Factorization models for forecasting student performance. In proceedings of Educational Data Mining. p. 6-8
2. Qiu, Y., Y. Qi, H. Lu, Z.A. Pardos, and N.T. Heffernan. 2011. Does Time Matter? Modeling the Effect of Time with Bayesian Knowledge Tracing. In proceedings of Fourth International Conference on Educational Data Mining. p.
3. Wang, Y., N. Heffernan, and J.E. Beck. 2010. Representing student performance with partial credit. In proceedings of Educational Data Mining. p. 335-336
4. Wang, Y. and N.T. Heffernan. 2011. Towards Modeling Forgetting and Relearning in ITS: Preliminary Analysis of ARRS Data. In proceedings of Fourth International Conference on Educational Data Mining. p.
5. Gong, Y. and J.E. Beck. 2011. Items, Skills, and Transfer Models: Which really matters for student modeling? In proceedings of Educational Data Mining. p. 81-90
6. Koedinger, K.R., E.A. McLaughlin, and J.C. Stamper. 2012. Automated Student Model Improvement. In proceedings of Educational Data Mining. p. 17-24
7. Pavlik, P.I., H. Cen, and K.R. Koedinger. 2009. Performance factors analysis—A new alternative to knowledge tracing. In proceedings of Proceedings of the 14th International Conference on Artificial Intelligence in Education. p.
8. Corbett, A.T. and J.R. Anderson, 1995. Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction. **4**: p. 253-278.
9. Gong, Y., J.E. Beck, and N.T. Heffernan, 2011. How to Construct More Accurate Student Models: Comparing and Optimizing Knowledge Tracing and Performance Factor Analysis International Journal of Artificial Intelligence and Education.
10. Gong, Y., J.E. Beck, and C. Ruiz. 2012. Modeling Multiple Distributions of Student Performances to Improve Predictive Accuracy. In proceedings of User Modeling and Adaptive Personalization. p. 102-113
11. Pardos, Z. and N. Heffernan. 2010. Modeling individualization in a Bayesian networks implementation of knowledge tracing. In proceedings of User Modeling, Adaptation, and Personalization. p. 255-266
12. Pardos, Z., R.S.J.D. Baker, S.M. Gowda, and N.T. Heffernan, 2012. The Sum is Greater than the Parts: Ensembling Models of Student Knowledge in Educational Software. ACM SIGKDD Explorations. **13**(2).
13. Baker, R.S.J.d., A.T. Corbet, and V. Aleven. 2008. More Accurate Student Modeling Through Contextual Estimation of Slip and Guess Probabilities in Bayesian Knowledge Tracing. In proceedings of Intelligent Tutoring Systems. p. 406-415
14. Baker, R.S.J.d., A.B. Goldstein, and N.T. Heffernan, 2011. Detecting Learning Moment-by-Moment. Journal of Artificial Intelligence in Education. **21**(1-2): p. 5-25.
15. Beck, J.E. and J. Mostow. 2008. How who should practice: Using learning decomposition to evaluate the efficacy of different types of practice for different types of students. In proceedings of Ninth International Conference on Intelligent Tutoring Systems. p. 355-362
16. Anderson, J.R., 1993. Rules of the Mind. Hillsdale, NJ: Lawrence Erlbaum Associates.
17. Pardos, Z., S.M. Gowda, R.S.J.d. Baker, and N.T. Heffernan. 2011. Ensembling Predictions of Student Post-Test Scores for an Intelligent Tutoring System. In proceedings of Educational Data Mining. p. 189-198
18. Gong, Y., J.E. Beck, N.T. Heffernan, and E. Forbes-Summers. 2010. The Impact of Gaming (?) on Learning. In proceedings of International Conference on Intelligent Tutoring Systems. p.
19. Baker, R.S.J.d., et al. 2006. Adapting to When Students Game an Intelligent Tutoring System. In proceedings of Proceedings of the 8th International Conference on Intelligent Tutoring Systems. p. 392-401
20. Breese, J., D. Heckerman, and C. Kadie. 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In proceedings of Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence. p.
21. Toscher, A. and M. Jahrer, 2010. Collaborative filtering applied to educational data mining. Journal of Machine Learning Research.
22. Li, S., X. Xiong, and J.E. Beck. 2013 (under review). Modeling student retention in an environment with delayed testing. In proceedings of Educational Data Mining. p.
23. Crocker, L. and J. Algina, 1986. Introduction to Classical and Modern Test Theory. Fort Worth: Harcourt Brace Jovanovich College Publishers. 482.
24. Baker, R.S.J.d., S.M. Gowda, A.T. Corbett, and J. Ocumpaugh. 2012. Towards Automatically Detecting Whether Student Learning is Shallow. In proceedings of Intelligent Tutoring Systems. p. 444-453
25. Wang, Y. and J.E. Beck. 2012. Using Student Modeling to Estimate Student Knowledge Retention. In proceedings of Educational Data Mining. p. 201-203
26. Pavlik, P.I. and J.R. Anderson, 2005. Practice and Forgetting Effects on Vocabulary Memory: An Activation-Based Model of the Spacing Effect. Cognitive Science. **29**(4): p. 559-586.
27. Baker, R.S.J.d., S.K. D'Mello, M.M.T. Rodrigo, and A.C. Graesser, 2010. Better to be frustrated than bored: The incidence, persistence, and impact of learners' cognitive-affective states during interactions with three different computer-based learning environments. International Journal of Human-Computer Studies. **68**(4): p. 223-241.
28. Arroyo, I., B.P. Woolf, D. Cooper, W. Burleson, and K. Muldner. 2011. The Impact of Animated Pedagogical Agents on Girls' and Boys' Emotions, Attitudes, Behaviors and Learning. In proceedings of Advanced Learning Technologies. p. 506-510
29. Beck, J.E. and Y. Gong. 2013. Wheel-spinning: student who fail to master a skill. In proceedings of Artificial Intelligence in Education. p. (in press)