Using Partial Credit and Response History to Model User Knowledge

Eric G. Van Inwegen Seth A. Adjei Yan Wang Neil T. Heffernan
100 Institute Rd
Worcester, MA, 01609-2280
+1-508-831-5569
{egvaninwegen, saadjei, ywang14, nth} @wpi.edu

ABSTRACT
User modelling algorithms such as Performance Factors Analysis and Knowledge Tracing seek to determine a student's knowledge state by analyzing (among other features) right and wrong answers. Anyone who has ever graded an assignment by hand knows that some answers are "more wrong" than others; i.e. they display less of an understanding of the skill(s) involved. This investigation seeks to understand the effects of progression through wrong answers to right answers in a way to determine how the "level" of wrongness affects future performance. The key findings are that A.) where in a series of opportunities a student reaches the goal impacts future performance, as does B.) the "level" of previous wrongness, even two questions before the current opportunity.

Right students are all alike;
every wrong student is wrong in his or her own way.
(with apologies to Ms. Karenina and Mr. Tolstoy)

1. INTRODUCTION
The use of algorithms to estimate student knowledge based on performance on intelligent tutoring systems (ITS) has been around for two decades. Two of the more well-known methods are knowledge tracing (KT) [6] and performance factors analysis (PFA) [11]. Both models use a student's right or wrong answers and develop a model to estimate the chance that a student has "learned" a particular skill. KT uses Bayes nets to determine four parameters per skill; PFA uses logistic regression to determine three parameters per skill. Although the order of correctness is incorporated into the models, both use only correctness as their input. Other pieces of information that may be collected by the ITS are neglected in these models.

ITS may collect any number of additional pieces of information about a student, their actions, their exact answers, etc. For example, Baker et. al. use over 20 features to make their predictions [2]. Some even make use of biometrics through additional sensors. (See Cavalio and D'Mello's review of several methods [3]. The goal of many of these algorithms is to try to make a computer tutor that is at least as responsive, observant, and effective as a human tutor would be. Incorporating more data about a student's affect can be seen as an attempt to give a computer access to the information that a human tutor would notice. However, the more detailed that a model becomes, the more computationally time-consuming it becomes. Also, as the number of inputs increases, fewer ITS's can make use of it (as a complex set of inputs may not be collected on all systems). One feature that might be incorporated into these algorithms is the use of the number of attempts and hints a student uses to answer a problem to classify more conditions than binary right and wrong and to look for the effect of how long it takes a student to achieve a particular classification.

Human teachers often employ the idea of partial credit, both as a motivational tool, and as a more accurate measure of knowledge (when compared to the binary correctness). Any teacher who has graded papers knows that some wrong answers (and workflow) demonstrate a nearly full understanding of a skill, while other wrong answers demonstrate a near-total lack of understanding. The idea of using dynamic testing (that is, a testing medium that gives hints to and tracks the number of attempts made by students) has been around since at least the 1980's. Bryant, Brown and Campione [5] compared traditional testing (binary correctness) to dynamic testing (tracking how many hints students needed to be successful). Others (e.g. Grigorenko and Sternberg) reviewed this kind of dynamic testing (among other methods) [8] and concluded that dynamic testing provides a more accurate measure [12]

Unfortunately, some ITS's can only determine the "worthiness" of a wrong answer if all wrong answers are somehow programmed in. Some ITS's do make use of pre-programmed wrong answers, but partial credit may or may not be given. Efforts before this one have been made to use partial credit to measure student knowledge [16]. E.g., in ASSISTments[1], wrong answers may be programmed to give a student a particular message, but A.) students are still marked completely wrong (and given no credit) and B.) all of these wrong answer messages must be programmed into the problems (which is incredibly time-intensive).

A more common method of assigning partial credit in ITS's is to give partial credit based on the number of attempts it takes a student to get the right answer [1] and/or the number of hints a student uses [9]. This is much faster to program, and does not require looking at all possible wrong answer to determine which ones show a limited understanding of the skill (as opposed to no understanding of the skill). The basic argument would be that a student who is "only slightly wrong" might figure out her mistake

[1] ASSISTments is an online learning system primarily for math, based out of Worcester Polytechnic Institute.

after only one wrong attempt, while a student who is "very wrong" might need several hints and several attempts before he can get the problem right. We are not analyzing specific wrong answers in this treatment; we using a student's partial credit history to modify the probability of that student getting the next question correct.

In this paper, we are analyzing a dataset from ASSISTments from the years 2012-2013. (The dataset contains ~ 500K student-problem instances; the content is mainly middle-school mathematics.) We analyze the student entries for patterns of attempts, hints use, and a simplistic order of actions to determine "bins" of students. We are also able to analyze the data to seek patterns of moving through bins (that is, as a single student uses more or less assistance on subsequent problems), and when in a particular opportunity count a bin (or sequence of bins) is encountered. We build off of our earlier work presented at the Learning Analytics & Knowledge Conference, 2015.

1.1 Background

In our previous work [13], we built off of other works that looked at attempt use, hint use (Assistance Model – AM – [15]), and simple sequence of action (Sequence of Action model – SOA – [7 and 17]), and modified and combined these models to make our own. We looked at the combination of number of attempts used to get the right answer, hint use, whether the "bottom-out hint" (BOH) was used, and a simplistic order of actions. In our model, the values for each parameter were:

- attempt use: 1, 2, 3, 4, (5+)
- hint use: 0, 1, 2, (3+)
- first action: hint or attempt
- BOH: used or not used

This gave us 35 different combinations. By analyzing the similarities of actions and future performance - defined as the average next problem correctness (NPC) and found by using pivot tables on 80% of the dataset, the 35 bins were combined into only 16. This gave us the "Fine-Grain Action" model (FGA). Table 1 shows the bins and re-grouped bins, and the NPC values.

Table 1a: The Fine-Grain-Action model 1st action = attempt

<table>
<tr><th></th><th>1 att.</th><th>2 att.</th><th>3 att.</th><th>4 att.</th><th>5 + att.</th></tr>
<tr><td>0 hint</td><td>0.8156
Bin 1</td><td>0.7380
Bin 2</td><td>0.6771
Bin 3</td><td>0.6380
Bin 4</td><td>0.6211
Bin 5</td></tr>
<tr><td>1 hint</td><td>-----</td><td colspan="2">0.7012
Group A</td><td colspan="2">0.6321
Group C</td></tr>
<tr><td>2 hint</td><td>-----</td><td colspan="4" rowspan="2">0.5812
Group E</td></tr>
<tr><td>3+ hint</td><td>-----</td></tr>
<tr><td>BOH</td><td colspan="5">0.5099
Group G</td></tr>
</table>

Table 1b: The Fine-Grain-Action model 1st action = hint

<table>
<tr><th></th><th>1 att.</th><th>2 att.</th><th>3 att.</th><th>4 att.</th><th>5 + att.</th></tr>
<tr><td>0 hint</td><td>-----</td><td>-----</td><td>-----</td><td>-----</td><td>-----</td></tr>
<tr><td>1 hint</td><td>0.7083
Bin 6</td><td colspan="2">0.6192
Group B</td><td colspan="2">0.5702
Group D</td></tr>
<tr><td>2 hint</td><td>0.5250
Bin 11</td><td colspan="4" rowspan="2">0.4688
Group F</td></tr>
<tr><td>3+ hint</td><td>0.4118
Bin 16</td></tr>
<tr><td>BOH</td><td colspan="5">0.3396
Group H</td></tr>
</table>

1.2 Research Questions

Extending from our previous analysis, we have three questions we want to address here:

1.) What is the significance of the bins?
   a) What is the statistical significance of the different bins? E.g. are bins "x" and "y" (arbitrary names) reliably different?
   b) Can the bins be re-grouped into larger groups without loss of predictive power? (E.g. Why 16? Why not 35 or 3?)
2.) Can the sequence of students moving through "Super Bins" be used to make more accurate predictions? (E.g. Is there a difference in expected outcome when comparing a student who moves from Super Bin 3 to 1 vs. 5 to 1?)
3.) Should all wrong answers be treated equally? Can we use reasonably simple and replicable methods to identify what student actions demonstrate different levels of understanding of the material?
   a) Is there an impact of bin sequence and / or opportunity count on predicted outcome?

2. METHODS

2.1 Creating the "SuperBins" (Method 1)

A quick glance at the next problem correctness (NPC) values in Table 1 shows that some bins are very nearly equivalent. When displayed in the above format, local values vary enough to warrant the bins. However, when put in order by bin values (which are just the mean NPC for instances falling into that category), we can now run a simple t-test (two tailed) analysis to compare one bin to the one that comes immediately after. This gives us Table 2.

Table 2: The bins from the FGA reordered and showing the p-value that compares one bin to the one immediately below.

Bin	NPC	stdev	n	p-value	Ordinal
1	0.8156	0.3878	215,870	< 0.0001	1st
2	0.7380	0.4397	22,229	0.0055	2nd
6	0.7083	0.4545	1,958	0.5827	3rd
A	0.7012	0.4577	3,414	0.0162	4th
3	0.6771	0.4676	5,616	0.0009	5th
4	0.6380	0.4806	2,326	0.7168	6th
C	0.6321	0.4822	1,408	0.4941	7th
5	0.6211	0.4851	2,518	0.9416	8th
B	0.6192	0.4856	407	0.1339	9th
E	0.5812	0.4934	4,011	0.8154	10th
D	0.5702	0.4950	114	0.3782	11th
11	0.5250	0.4994	541	0.4851	12th
G	0.5099	0.4999	40,652	0.0781	13th
F	0.4688	0.4990	465	0.1252	14th
16	0.4118	0.4922	289	0.0141	15th
H	0.3396	0.4736	13,989	-----	16th

In Table 2, the p-value analysis comparing the bin of that line to the one below it allows us to identify natural break points and groups. Bins are regrouped according to these break points. That

is, bins are grouped together as long as two bins fail to be statistically different. This gives us five "SuperBins" (Table 3).

It may seem somewhat arbitrary to keep bins 16 and H separate (with a p-value of 0.0141), while grouping A and 3 together (with a p-value of 0.0162). We could argue that we used a deciding value of 0.015, but that would be an arbitrary value. The real reason for keeping 16 and H separate is that the action of using the bottom out hint (and using a hint as the first action) seems to be different than any other combination of actions and should be kept separate. Throughout the rest of this analysis, we will see that the results of keeping this bin separate as its own SuperBin gives us more predictive ability.

This gives us a useful and relevant way to regroup bins that are not reliably different. One can easily make the argument against the 16 bins in FGA that, if two bins are not statistically different, why have them? By combining statistically similar bins, there is more meaning (in prediction) to assigning a particular value for the next problem correctness, even if the recombination "smooths over" the different ways that a student could arrive at a particular prediction.

Table 3: The five "SuperBins" with their predictive values, and relevant statistics. The colors are used consistently throughout the paper for clarity sake.

SuperBin	NPC	stdev	n	p-value
1	0.8156	0.3878	215,870	<< 0.0001
2	0.7380	0.4398	22,229	<< 0.0001
3	0.6902	0.4624	11,015	<< 0.0001
4	0.5297	0.4991	52,731	<< 0.0001
5	0.3396	0.4736	13,989	----

If we use the colors to remake a condensed Table 1, we can see that the SuperBins are locally consistent within the FGA. This is significant in that it suggests that, although many of the 16 bins from FGA may be statistically similar, these similarities (and differences) occur logically throughout the chart. (See Table 4.)

Table 4: FGA color coded according to SuperBins.

Hints	1 att.	2 att.	3 att.	4 att.	5+ att.
0	Bin 1, 0.816	Bin 2, 0.738	Bin 3, 0.677	Bin 4, 0.638	Bin 5, 0.621
1	Bin 6, 0.708	Grp A 0.701		Grp C 0.632	
		Grp B 0.619		Grp D 0.570	
2	Bin 11 0.525	Grp E 0.581			
		Grp F 0.469			
3+	Bin 16 0.412	Grp E			
		Grp F			
BOH	**attempt 1st**	**Grp G 0.510**			
	hint 1st	**Grp H, 0.340**			

It is also worth noting that, although the bin numbers that went into the SuperBins may seem random, there is a pattern. SuperBin 1 consists of students who get a problem right. SB2 is populated by only students who made only one wrong attempt (and used no hints) before getting the answer right on their own. SB3 comes from three bins that represent only a small number of attempts / hint use. SB4, which incorporates the bulk of the FGA bins, is anything left, except for using the bottom-out hint, with the first action being hint use. We can now use these SuperBins as the identifier of "wrongness".

Table 5: Meaning (in terms of attempt and hint use) and interpretation of "wrongness" of the five SuperBins

SuperBin	Meaning	"Wrongness"
1	Student got it right	Right
2	Student made one wrong attempt, and then got it right.	Barely wrong
3	Student used a few attempts, and 0 or 1 hint.	Partially wrong
4	Student used many attempts and/or hints.	Significantly wrong
5	Student could not start without a hint, and needed the answer.	Completely wrong

In ASSISTments, a *must* get the right answer before moving onto the next question, no matter how many attempts they make or hints they use. Clearly, a student who makes one wrong attempt and then gets the answer right with no hints demonstrates that their thinking was "less wrong" than a student who makes a series of attempts and uses many hints before getting to the correct answer. SuperBins give us a working definition of "wrongness".

2.2 Impact of previous bin; 2 SuperBin (2SB) combinations (Method 2)

Looking at the sequence of students "moving" through SuperBins can help us to better understand how a student's knowledge on a skill is changing. As we look at a student's performance on one skill, progression through SuperBins would indicate that the student's knowledge is improving; most humans would call this "learning." Likewise, a student who gets an answer right, and then regresses could have "slipped" (to use KT terminology loosely).

The first (and simplest) method to look at the impact of previous SuperBins on future success is to look at two-bin combinations. That is, after the first problem, we will look at not just the SuperBin a student falls into on opportunity n, but also the SuperBin they were in on opportunity (n-1). This gives 25 different combinations. Our naming convention is (current).(previous). Thus, 2.1 is a student who is in SuperBin 2 (used one wrong attempt before getting a problem right on the second try) and was in SuperBin 1 (got the problem right on the first attempt). To use knowledge tracing language, 2.1 could represent a "slip". Two-SuperBin code 1.2 is a student who was in SuperBin 2 and has improved to SuperBin 1. Two-SuperBin codes run from [(1.1-1.5) - (5.1-5.5)].

Table 6 (next page) illustrates the impact of the previous question's "wrongness" on the outcome after the current question. For instance, if we compare the values of the 1.x family, we should not be surprised that the 1.1 (two correct in a row) has the highest probability of success on the next problem. However, the four other two-bin combinations (1.2-1.5) all have (statistically significantly) different predictions for the next problem. That is, how wrong a student was on the previous question can be an indicator for how likely they are to get a question right, even after they have gotten one right.

Perhaps the best demonstration of the importance of using a partial credit metric (of some sort) is to compare the predicted outcomes for 2.2 and 5.5. In both cases, the students would be marked wrong on two consecutive problems. However, a student who manages to make a mistake and then correct themselves with no

aid (twice) is (un-surprisingly) much more likely to get the next problem correct than one who needs the answer given to them (and won't even start without a hint). A student in 2.2 has a nearly 70% chance of success on the next problem, while a student in 5.5 has a mere 16.7% chance! Without looking at partial credit, they would be marked equally wrong.

Table 6: Two SuperBin Combinations. Code 1.x refers to students who are currently in SuperBin 1 and who were in SuperBin x on the last problem. "Families" (1.x, 2.x, etc.) are color coded according to current SuperBin. Codes without decimal (bolded) are values from Table 3. The p-values compare a 2SB to the one below it.

2SB	NPC	n	p-value
1	**0.816**	**215,870**	
1.1	0.840	121,317	< 0.0001
1.2	0.806	15,440	< 0.0001
1.3	0.775	7,085	< 0.0001
1.4	0.703	26,109	< 0.0001
1.5	0.655	4,317	-----
2	**0.738**	**22,229**	
2.1	0.783	11,421	< 0.0001
2.2	0.699	2,137	0.5322
2.3	0.688	1,016	< 0.0001
2.4	0.608	2,850	0.4391
2.5	0.587	373	-----
3	**0.690**	**11,015**	
3.1	0.733	4,799	< 0.0001
3.2	0.637	796	0.3058
3.3	0.611	674	0.3582
3.4	0.590	1,433	0.5120
3.5	0.567	233	------
4	**0.530**	**52,731**	
4.1	0.617	19,044	< 0.0001
4.2	0.551	2,321	0.5579
4.3	0.561	1,336	< 0.0001
4.4	0.434	15,452	< 0.0001
4.5	0.380	3,263	------
5	**0.340**	**13,989**	
5.1	0.540	2,155	0.8180
5.2	0.548	228	0.2658
5.3	0.491	165	< 0.0001
5.4	0.332	3,165	< 0.0001
5.5	0.167	4,429	-----

2.3 Impact of opportunity count on 2SB combination predictions (Method 3)

The data set we are analyzing has been limited to only up to opportunity counts of 20. (This was done to speed the analyses.) Even with 25 two-SuperBin combinations, there was enough information in the data set to run a linear regression on the effect of when a two-SuperBin combination was reached. E.g. there is a difference between students who reach 1.1 (two right in a row) on opportunity 2 versus opportunity 20.

To create this model, pivot tables in excel were used to find the average next problem correctness (NPC) on two-SuperBin combinations that fall on particular opportunities. Although not all two-SuperBin combinations were achieved on all opportunities, there was enough information to run a linear regression. This, of course, gives an intercept and slope. The model was applied using the regression, not by using the actual calculated values.

2.4 Impact of 3 SuperBin (3-SB) combinations (Method 4)

Just as the state of the previous SuperBin could have an effect on future performance, it is conceivable that the SuperBin two opportunities back could have an effect. Consider the following two hypothetical students and their first three SuperBins:

Table 7: Two hypothetical students and their SuperBin values on three questions.

Student	Q1	Q2	Q3	Q4
Alice	SB 2	SB 1	SB 1	?
Barney	SB 5	SB 1	SB 1	?

Intuitively, we would expect Alice to have a higher probability of success on question 4 than Barney. Alice almost got the first question right, while Barney needed to use the bottom out hint (and used a hint as his first action). Although they both got questions 2 and 3 correct, their performances on question 1 are drastically different. To user models such as KT and PFA, however, they were both equally "wrong" on question 1.

To identify a 3-SuperBin combination, we will use the two-SuperBin code and add a decimal, we would have (current).(previous)(n-2) or [1.11-5.55]; this gives 125 three-SuperBin combinations. In the example above, after question 3 (and as the model predicts their correctness on question 4), Alice would be in 1.12, while Barney is in 1.15.

We are now looking at 125 combinations; some of these combinations have too few instances to have a prediction value that is reliable. 47 out of 125 3-SB combinations have fewer than 100 instances; eight combinations have 10 or fewer instances. Instead of using 125 different values (many of which would unreliable), we will use a linear regression to approximate values for the impact of the (n-2) SuperBin. However, it is a slightly complex process.

In order to have "smooth" regressions, some assumptions are made:

1.) The effects can be modelled linearly. E.g., for the regression to the (n-1) SuperBin prediction = intercept + slope*SuperBin (n-1).
2.) The effect of the (n-2) SuperBin value will be similar in pattern to the effect of SuperBin (n-1), but reduced in effect. (In other words, we would expect that 1.1x to follow the basic pattern of 1.x, but with a smaller change in values)
3.) Even though many of the three-SuperBin combinations are unreliable due to small numbers of instances, the average slope of a "family" could be used to deduce the effect size that is applied to the pattern found in assumption 2.

To create the model, five regression lines (one each for 1.x, 2.x, 3.x, 4.x, and 5.x) were created by simply using the average next problem correctness as the y-values and the decimal (previous SuperBin) as the x-values.

Next, twenty-five regressions were run for 1.1x - 5.5x. Although many of the three-SuperBin combinations were too small to be reliable, we used the average slope from a "family" (e.g. 1.3x) to adjust the effect from the two-SuperBin combination regressions. E.g., the regression lines for 1.1x - 1.5x were found and averaged.

To approximate the slopes of 1.1x-1.5x, the slopes of 1.x - 5.x were used, but multiplied by the ratio of the average (1.1x-1.5x) to the average (1.x - 5.x). Since the intercepts from (1.x-5.x) might not have the same meaning when compared to (1.1x - 1.5x), the intercepts from the three-bin regressions were left as is. Table 8 (below) shows the 2-SuperBin regressions (found using the values in Table 6), followed by the actual regression values for one of the 3-SuperBin families, and the idealized slopes.

2.5 First Possible Opportunity Count

Lastly, when fitting our methods (many of which would have to be some combination of the above four versions), we decided to separate SuperBins and combinations by the first available opportunity count, and all others. In our numbering scheme, we used a "dummy code" of 09 to designate that we are looking at the average of NPC for only the first available opportunity count. See next section for examples which may help.

2.6 Method Examples

We now arrive at the methods by which our model is applied. To see the differences between the methods, it may be useful to look at the same hypothetical sequence of SuperBins for two imaginary students and compare the different methods. (See Table 9, next page.) In all methods below, we compare "Chuck" and "Denise" and the parameters that would be used to predict their success. It's important to note that method 1 identifies the SuperBin into which each student is placed on questions 1-4; this does not change throughout the methods.

The simplest method uses only the average NPC for all SuperBins, and pays no attention to opportunity count or SuperBin combinations. This is Method 1. This can be thought of as a simplified FGA.

The prediction for (e.g.) question 5 is based solely on the SuperBin value for question 4. SuperBin values are modified by a multinomial logistic regression based on skill. This gives a total number of parameters as 5 + 1/skill.

In method 2, the prediction of NPC for question 1 is based on the average value for the SuperBin, but only including values from the first opportunities. (The "dummy code" of 09 is used to indicate first opportunity only.) All questions from then on use the value for the two-bin combinations. This gives a total number of parameters of 30 + 1/skill. (Five for SBx.09, and 25 for 1.1-5.5, plus the regression to skill)

In method 3, the prediction of NPC from question 1 is based on SuperBin at first opportunity, while all others are based on the regression to opportunity count values. This gives a total number of parameters of 55 + 1/skill. (Five for SB x.09, and 50 for the intercept and slope of the 25 different two-SuperBin combinations, plus the regression to skill)

In method 4, the prediction of NPC for question 1 and 2 are based on SuperBin x.09 and 2-SuperBin combinations x.y09. For question 3 and on, the prediction is based on the linear regression to the SuperBin of (n-2). This gives a total number of parameters of 65 + 1/skill. (Five for SB x.09, 25 for 2SB combo x.y09, 25 for the intercepts, five for the slope of 1.x-5.x, and five for the slope modification parameter, plus the regression to skill). The slope and intercept in the regressions in method 4 are not the same as those in method 3.

A demonstration of the application of all four methods can be found in Table 9 below. Method 1, being simply the single SuperBin prediction identifies a "score" or "condition" for the hypothetical students. The other methods start with this information.

Table 8: demonstration of idealization of regression to third bin using second bin regression values.

2 SB "family"	m	b
1.x	-0.047	0.898
2.x	-0.048	0.818
3.x	-0.038	0.741
4.x	-0.059	0.686
5.x	-0.096	0.704
3 SB "family" actual	m actual	b actual
1.1x	-0.032	0.913
1.2x	-0.031	0.845
1.3x	-0.025	0.089
1.4x	-0.034	0.778
1.5x	-0.018	0.695
3 SB "family" idealized	m idealized	b actual
1.1x	-0.023	0.913
1.2x	-0.023	0.845
1.3x	-0.018	0.089
1.4x	-0.029	0.778
1.5x	-0.047	0.695

Table 9: Hypothetical application of four different methods; it is important to note that the methods are different, but the results of "Chuck" and "Denise" are not. "X.09" (or "X.Y09") is a code meaning prediction values are derived from the first available bin only. E.g. "1.09" uses only the scores from SuperBin 1 and the first opportunity.

Method 1: SuperBin Only ("SB_1")					
Student	Q1	Q2	Q3	Q4	Q5
Chuck	SB1	SB2	SB1	SB1	...
Denise	SB5	SB3	SB1	SB2	...
Method 2: Two-SuperBin combinations ("SB_2")					
Student	Q1	Q2	Q3	Q4	Q5
Chuck	SB 1.09	2SB (2.1)	2SB (1.2)	2SB (1.1)	...
Denise	SB 5.09	2SB (3.5)	2SB (1.3)	2SB (2.1)	...
Method 3: Two-SuperBin combinations, with opportunity regression ("SB_3")					
Student	Q1	Q2	Q3	Q4	Q5
Chuck	SB 1.09	b(2.1) +m(2.1)*2	b(1.2) + m(1.2)*3	b(1.1) + m(1.1)*4	...
Denise	SB 5.09	b(3.5) +m(3.5)*2	b(1.3) + m(1.3)*3	b(2.1) + m(2.1)*4	...
Method 4: Two-SuperBin combinations, with third bin regression ("SB_4")					
Student	Q1	Q2	Q3	Q4	Q5
Chuck	SB 1.09	2SB (2.109)	b'(1.2) + m'(1.2)*1	b'(1.1) + m'(1.1)*2	...
Denise	SB 5.09	2SB (3.509)	b'(1.3) + m'(1.3)*5	b'(2.1) + m'(2.1)*3	...

3. RESULTS

In order to better show methods, many of the tables that would be considered "results" are found throughout the paper. We hope this does not inconvenience the reader too much at this time.

Tables 1 through 5 show that a statistical analysis of student actions (based on next problem correctness) can simplify a complex table, while still retaining meaningful groupings of student actions. In our last paper, we argued that not only should hint use and attempt count be used in the model, but a simple action-order analysis should be included. We can point out that the regrouping process does not contradict this conclusion. Had group A not been split from group B, the model might not have fared so well.

Table 6 demonstrates that there is an effect of the previous SuperBin that will modify the prediction of the current SuperBin. For example, we can see that students in SuperBin 1 who were just in SuperBin 5 have almost a 20% (absolute) less chance of success on the next problem when compared to a student who was in SuperBin 1 twice running. This may not be too surprising, as SuperBin 1 represents getting the answer right. However, there is still a roughly 15% (absolute) difference in expected outcomes between 2SB 1.2 and 1.5. Both of these represent a student who got a problem wrong, and then got the next right. Algorithms such as KT or PFA would treat these conditions as identical.

When analyzing the 2SB combinations, the pattern is amazingly clear: the impact of wrongness does not disappear after one question, and the different levels have different (and predictable) impacts. Being in SuperBin 5 on the previous problem gives a student a worse outcome than 4; 4 is worse than 3, etc. There are only a few deviations from this pattern throughout Table 6. The p-value analysis indicates that the differences are reliable most of the time; that is, the patterns appear to be reliable, although a larger dataset is needed to state that definitively across all patterns.

One interpretation of the pattern of effect from the previous SuperBin would be that students in SuperBin 5 have more to learn than those in SuperBin 4, and that even getting the next question right is not a clear sign of having learned the knowledge component. The summary table (Table 5) gives another interpretation on this: the students in SuperBin 5 needed a hint before they even got started, and then needed the answer to finish. Clearly, these students are nowhere in the same state of learning as a student who makes one mistake and fixes their answer on their own (SB2).

This differentiation of "wrongness" demonstrates the power of looking at non-binary correctness. Perhaps the most dramatic observation is that a student who is wrong twice, but corrects themselves each time (2SB combination 2.2) is very different from a student who cannot start without a hint and cannot get to the correct answer on their own (2SB combination 5.5). To treat these two states as the same (wrong twice running) is to give up on information that can help differentiate a student who is nearly 70% likely to be correct on the next problem, verses one who as a paltry 16.7% chance (2.2 vs 5.5).

With these new predictions, we can compare predictions to other models. In Table 10, we compare the scores from RMSE, AUC, and R-squared. This shows that not only is the "SuperBin" method as valid as the FGA model (tying in two out of three metrics), taking opportunity regression (method 3) and 3-SB regression both improve on the basic SuperBins idea (method 1).

One table that a reader might be missing is one detailing the relation of 2SB to opportunity. Rather than add an eleventh table, we will summarize as: the R^2-values for the regressions ranged from 0.832 to 0.001; some are clearly not reliable. However, given the results in Table 10, we think that accounting for opportunity count by linear regression to the 2SB combinations is a worthwhile first approximation.

Table 10: Analysis of various knowledge models. Baseline predicts the average value of the training set. For AUC, 1.0 is ideal; 0.5 is no better than random. RMSE: 0.00 is ideal; 0.5 is no better than random. R^2: 1.00 is ideal, 0.0 is no better than random.

Method	AUC	RMSE	Rsqr
Baseline (predict mean)	0.500	0.446	0.000
PFA [11]	0.653	0.426	0.058
KT [6, 4, 10]	0.710	0.413	0.115
SOA [7, 17]	0.708	0.426	0.087
AM [15]	0.714	0.422	0.103
FGA [13]	0.715	0.400	0.128
SB method 1	0.715	0.411	0.128
SB method 3	0.726	0.407	0.142
SB method 4	0.727	0.406	0.145
Avg (methods 3 & 4)	0.728	0.406	0.145

4. CONCLUSIONS

The regrouping of 16 bins of the FGA into 5 "Super Bins" does not adversely affect the predictive power of the model (in two out of three metrics). In fact, by having fewer bins, we are able to look at history in a way we would not have, had we kept the 16 bins of the Fine-Grain Action model. This gives us a chance to improve on the FGA.

We can conclude that not all wrong answers[2] are equal, and that there is value to be gleaned from analyzing different wrong answers. The impact of "how wrong" an answer is has an effect even up to two answers later. That is, your "wrongness" two questions back can be used to make a better prediction for your next problem. (It is possible that wrongness further back could be used, but it would require a dataset that is larger by orders of magnitude.)

Not only is the combination of "wrongness" useful in making predictions, so too is the opportunity on which a student achieves a combination. That is, a student who gets the first two questions right is (usually) more likely to get the third right than a student who gets the 11^{th} and 12^{th} questions right is to get the 13^{th} correct.

It is perhaps not too surprising that this method is able to outperform established models such as PFA and KT. (And we will freely admit that the previous statement is limited only to this one dataset; more research is needed to definitively make this statement.) PFA and KT use only the information in binary correctness. A new model that outperforms existing models by using additional information does not negate the previous models; it merely shows that this information is worth incorporating into models of user knowledge.

[2] Or, more precisely, combinations of student actions that are treated as wrong answers; actual analysis of wrong answers is left to another paper.

4.1 Answers to the Research Questions

1.) The bins from FGA were useful, but needed to be regrouped. Regrouping by next problem correctness (and t-test analysis) kept local and logical groupings that yield meaningful descriptions of wrongness.

2.) The level of wrongness that a student demonstrates has an effect on more than just the current question. This effect is clear and reliable on the next problem and may impact the following.

3.) Not all wrong answers are identical. Knowledge estimation models such as KT and PFA leave out "levels" of wrongness that can be used to make a more accurate prediction of student success.

The paraphrased Anna Karenina quote at the start of the paper summarizes both our hypothesis and our findings: A careful analysis of wrong answers will help improve knowledge estimation models.

4.2 Novel Contributions

This paper seeks to show that there is information to be gained by treating different kinds of wrong answers as different. Presented herein is a statistical method of differentiating student actions into groups of actions that represent meaningful differences in performance. Use of these groups in a knowledge modelling algorithm can improve the results of the predictions, without needing continuous values (as in [14]).

4.3 Future Work

Although all of the linear regressions can be considered first-order approximations, the idealization of the third bins may be perhaps only a zeroth-order. As more data becomes available, we may be able to bypass the idealization and simply use 125 different parameters that are statistically reliable. Beyond improving the results of this model, the incorporation of other models that seek to use information from incorrect answers should bolster the performance of the model(s).

5. ACKNOWLEDGEMENTS

Special thanks also go out to the ASSISTments team for all the work they do in harvesting the data from the system. We also acknowledge and thank funding for ASSISTments from the NSF (1316736, 1252297, 1109483, 1031398, 0742503, and 1440753), the U.S. Dept. of Ed. GAANN (P200A120238), ONR's "STEM Grand Challenges," and IES (R305A120125, R305C100024).

6. REFERENCES

[1] Attali, Y., & Powers, D. (2010). Immediate feedback and opportunity to revise answers to open-ended questions. *Educational and Psychological Measurement*, *70*(1), 22-35.

[2] Baker, R. S., Goldstein, A. B., & Heffernan, N. T. (2010). Detecting the moment of learning. *Intelligent Tutoring Systems*. Springer Berlin Heidelberg.

[3] Calvo, R. A., & D'Mello, S. (2010). Affect detection: An interdisciplinary review of models, methods, and their applications. *Affective Computing, IEEE Transactions on*, *1*(1), 18-37.

[4] Chang, K., Beck, J., Mostow, J., & Corbett, A. (2006). A bayes net toolkit for student modeling in intelligent tutoring systems. Intelligent Tutoring Systems. Springer Berlin Heidelberg.

[5] Campione, J. C., Brown, A. L., & Bryant, N. R. (1985). Individual differences in learning and memory. In R. J. Sternberg (Ed.). *Human abilities: An information-processing approach*, New York: W. H. Freeman. pp. 103-126.

[6] Corbett, A. T., & Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, *4*(4), 253-278.

[7] Duong, H. D., Zhu, L., Wang, Y., & Heffernan, N. T. (2013). A Prediction Model Uses the Sequence of Attempts and Hints to Better Predict Knowledge: Better to Attempt the Problem First, Rather Than Ask for a Hint. *Proceedings of the 6th International Conference on Educational Data Mining.* 2013.

[8] Grigorenko, E. L., & Sternberg, R. J. (1998). Dynamic Testing. Psychological Bulletin, 124, 75-111.

[9] Hawkins, W., Heffernan, N., Wang, Y., & Baker, R. S. Extending the Assistance Model: Analyzing the Use of Assistance over Time.

[10] Murphy, K.: Bayes Net Toolbox for Matlab. < https://code.google.com/p/bnt/ > Accessed 4 September, 2014

[11] Pavlik Jr, P. I., Cen, H., & Koedinger, K. R. (2009). Performance Factors Analysis--A New Alternative to Knowledge Tracing. *Online Submission*.

[12] Sternberg, R.J., & Grigorenko, E.L. (2002). Dynamic testing: The nature and measurement of learning potential. Cambridge, England: Cambridge University Press.

[13] Van Inwegen, E. G., Adjei, S., Wang, Y., & Heffernan, N. T. An Analysis of the Impact of Action Order on Future Performance: the Fine-Grain Action Model, LAK2015, in publication

[14] Wang, Y., & Heffernan, N. (2013). Extending knowledge tracing to allow partial credit: using continuous versus binary nodes. *Artificial Intelligence in Education*. Springer Berlin Heidelberg.

[15] Wang, Y., & Heffernan, N. T. (2011). The" Assistance" Model: Leveraging How Many Hints and Attempts a Student Needs. *FLAIRS Conference*.

[16] Wang, Y., Heffernan, N. T., & Beck, J. E. (2010). Representing Student Performance with Partial Credit. *EDM*.

[17] Zhu, L., Wang, Y., & Heffernan, N. T. The Sequence of Action Model: Leveraging the Sequence of Attempts and Hints.