Responding to Moments of Learning

by

Adam B. Goldstein

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

May 2011

APPROVED:

Professor Neil Heffernan, Major Co-Advisor

Professor Ryan Baker, Major Co-Advisor

Professor Sonia Chernova, Thesis Reader

Professor Craig Wills, Head of Department

Abstract

In the field of Artificial Intelligence in Education, many contributions have been made toward estimating student proficiency in Intelligent Tutoring Systems (cf. Corbett & Anderson, 1995). Although the community is increasingly capable of estimating *how much* a student knows, this does not shed much light on *when* the knowledge was acquired. In recent research (Baker, Goldstein, & Heffernan, 2010), we created a model that attempts to answer that exact question. We call the model P(***J***), for the probability that a student **j**ust learned from the last problem they answered. We demonstrated an analysis of changes in P(***J***) that we call "spikiness", defined as the maximum value of P(***J***) for a student/knowledge component (KC) pair, divided by the average value of P(***J***) for that same student/KC pair. Spikiness is directly correlated with final student knowledge, meaning that spikes can be an early predictor of success. It has been shown that both over-practice and under-practice can be detrimental to student learning, so using this model can potentially help bias tutors toward ideal practice schedules.

After demonstrating the validity of the P(***J***) model in both CMU's Cognitive Tutor and WPI's ASSISTments Tutoring System, we conducted a pilot study to test the utility of our model. The experiment included a balanced pre/post-test and three conditions for proficiency assessment tested across 6 knowledge components. In the first condition, students are considered to have mastered a KC after correctly answering 3 questions in a row. The second condition uses Bayesian Knowledge Tracing and accepts a student as proficient once they earn a current knowledge probability (L_n) of 0.95 or higher. Finally, we test P(***J***), which accepts mastery if a student's P(***J***) value spikes from one problem and the next first response is correct. In this work, we will discuss the details of deriving P(***J***), our experiment and its results, as well as potential ways this model could be utilized to improve the effectiveness of cognitive mastery learning.

Acknowledgements

This research was supported via NSF grant award number DGE0742503, by the Pittsburgh Science of Learning Center, NSF award number SBE-0836012, a NSF CAREER award to Professor Neil Heffernan, and by U.S. Department of Education program grant #R305A070440. For the full list of over a dozen funders, please see http://www.webcitation.org/5xp605MwY.

I would like to thank Sujith Gowda for his help with statistical analyses, Matthew Dailey for all of his assistance with data retrieval, as well as Vincent Aleven and Zachary Pardos for helpful comments and suggestions. My gratitude to Michele Fulk, Kim Thienpont, and Courtney Mulcahy for their patience and graciousness helping me conduct my study.

A special thank you to both Neil Heffernan, my co-advisor, and Cristina Heffernan, my fellowship coordinator, for including me in their incredible project. Thanks to the entire ASSISTments team for all of the support and fun in the past two years and love to my family for their encouragement and understanding. Above all, thank you to Ryan Baker, my other co-advisor, without whom I don't think I would have ever finished this work. His constant assistance and guidance taught me a lot about science, a bit about life, and everything I know about being a researcher.

Contents

List of Figures

List of Tables

1 Introduction

In recent years, educational data mining and knowledge engineering methods have led to increasingly precise models of students' knowledge as they use intelligent tutoring systems (ITS). Student modeling has a rich history within the fields of ITS and artificial intelligence in education more broadly (cf. Goldstein, 1979; Burton, 1982; Sleeman, 1982). Educational data mining/machine learning methods began to play a role in student modeling fairly early, including work to automate the process of discovering models of students' procedures (e.g. Sleeman, Langley, & Mitchell, 1982; Langley & Ohlsson, 1984) and work to understand the roots of student errors (e.g. VanLehn, 1990). By the mid-1990s, model-fitting procedures based on student performance data had become a standard part of the student models used in intelligent tutoring systems (cf. Corbett & Anderson, 1995; Martin & VanLehn, 1995; Shute, 1995). These models were reasonably accurate at inferring the probability that a student knew a specific skill or concept (constructs recently referred to as knowledge components, abbreviated KC – cf. Koedinger & Corbett, 2006) and whether a student possessed or lacked specific incorrect "bug rules." Student knowledge was inferred by these models using the student's pattern of correct responses and non-correct responses (e.g. errors and hint requests) up until the current time, typically through a procedure where an estimate of the student's knowledge is updated after each response.

In recent years, researchers have attempted to extend student knowledge modeling to predict student knowledge more precisely based on information beyond just correctness. For instance, Beck et al. (2008) developed a model that assessed the probability of learning at a given moment differently if a student requested help than if they made an error. While this approach provided useful information about help's utility, the resultant model did not have significantly improved predictive power. Johns and Woolf (2006) studied the possibility that knowledge modeling could be made more accurate by modeling gaming the system at the same time, leading to slight improvements in cross-validated prediction of future performance. Baker, Corbett, & Aleven (2008) extended Bayesian Knowledge Tracing with contextualized estimation of the probability that the student guessed or slipped, leading to better prediction of future correctness. More recent work has suggested that the exact framework from Baker et al.'s research in 2008 leads to poorer prediction of post-test scores, but that information on contextual slip can be used in other fashions to predict post-test scores more precisely than existing methods (Baker et al., 2010). Pardos and Heffernan (2010) extended Bayesian Knowledge Tracing with improved estimation of student knowledge priors based on initial performance, showing statistically significantly better prediction of within-tutor performance. Other knowledge tracing frameworks have attempted to model performance on problems or problem steps that involve multiple skills at the same time (cf. Pavlik & Anderson, 2009; Pardos, Beck, Ruiz, & Heffernan, 2008), and have focused on predicting a student's speed of response in addition to just correctness (cf. Pavlik & Anderson, 2008).

Creating more precise models of student learning has several benefits. First of all, to the extent that student practice is assigned based on knowledge assessments (cf. Corbett & Anderson, 1995), more precise knowledge models will result in better tailoring of practice to individual student needs (cf. Cen, Koedinger, & Junker, 2007). Second, models of student knowledge have become an essential component in the development of models of student behavior within intelligent tutoring systems. Knowledge models have been employed as key components of models of many constructs, including models of appropriate help usage (Aleven, McLaren, Roll, & Koedinger, 2006), gaming the system (Baker, Corbett, Roll, & Koedinger, 2008; Walonoski & Heffernan, 2006), and off-task behavior (Baker, 2007; Cetintas, Si, Xin, Hord, & Zhang, 2009). More precise knowledge models can form a more reliable component in these analyses, and as such increase the fidelity of models of behavior.

However, while these recent extensions to models of student knowledge have created the potential for more precise assessment of student knowledge at a specific time, these models do not tell us *when* the knowledge was acquired. In this paper, we will introduce a model that can infer the probability that a student learned a knowledge component (KC) at a specific step during the problem-solving process. Note that this probability is ***not*** the same thing as P(***T***) in standard Bayesian Knowledge Tracing (a full explanation will be given later in this paper). Creating a model that can infer this probability will create the potential for new types of analyses of student learning, as well as making existing types of analyses easier to conduct. For example, this type of approach may allow us to study the differences between gradual learning, such as the strengthening of a memory association, governed by power/exponential improvements in accuracy and performance (Newell & Rosenbloom, 1981; Heathcote, Brown, & Mewhort, 2000) and "eureka" moments within learning, where a skill or concept is suddenly understood (cf. Lindstrom & Gulz, 2008). Both types of learning are known to occur (Anderson & Lebiere, 2006), but the conditions leading to sudden "eureka" moments in learning – for example, the moment of insight in an insight problem (Duncker, 1945; Metcalfe & Wiebe, 1987) – are still incompletely known (Bowden et al., 2005). It has been argued that the traditional paradigm for studying insight, focused on laboratory experiments using highly difficult problems thought to require a single insight for success, is insufficient to fully understand insight (Bowden et al., 2005). This has led to finer-grained research on insight using EEG and FMRIs (Bowden et al., 2005; Kounios et al., 2008). With the extremely large data sets now available for intelligent tutors (Koedinger et al., 2010), and a metric that can assess whether learning is steady or sudden, it may be possible to expand insight research further, to learn about the conditions that are associated with "eureka" moments during in-vivo learning of complex academic skills and concepts over long periods of time.

In addition, studying the relationship between behavior and immediate learning will be facilitated by having a concrete numerical measure of immediate learning. Prior methods for studying these relationships have required either looking only at the single next performance opportunity (cf. Cocea, Hershkovitz, & Baker, 2009), a fairly coarse learning measure, or have required interpreting the difference between model parameters in Bayesian Knowledge Tracing (cf. Beck et al., 2008), a non-trivial statistical task. For the same reasons, studying which items are most effective (and in which order they are most effective) (cf. Beck & Mostow, 2008; Pardos & Heffernan, 2009; Pardos, Dailey, & Heffernan, 2010) will be facilitated with the addition of a concrete numerical measure of immediate learning. Creating models of a student's learning, moment-by-moment, may even enable distinctions between behaviors associated with immediate learning and behaviors associated with learning later on, and enable identification of the antecedents of later learning. For example, perhaps some types of help lead to immediate better learning but others aid by preparing the student for future learning (cf. Bransford & Schwartz, 1999) so that differences in performance can only be seen after additional practice has occurred.

In the following sections, we will present an approach for labeling data in terms of student immediate learning, a machine-learned model of student immediate learning (and indicators of goodness of fit), and two examples of the type of “discovery with models” analysis that this type of model enables. In that analysis, we will investigate whether learning is differentially “spiky” between different KCs, with learning occurring abruptly for some KCs and more gradually for other KCs, as well as how different “spike patterns” can potentially correlate to external learning metrics. It is worth noting that “spikiness” is defined as the maximum value of P(***J***) for a student/knowledge component (KC) pair, divided by the average value of P(***J***) for that same student/KC pair. This will be revisited in depth later in the paper. We will also discuss attempts to improve our model and explore the model’s potential utility with a pilot study comparing it to other models of cognitive mastery learning.

2 Data

Before discussing the procedure used to model learning moment-by-moment, we will discuss the data used for its derivation. We worked with additional data sets in the analyses covered in sections 5 and 6, but they will be described later. For the derivation of our model, data was obtained from two Intelligent Tutoring Systems: The Middle School Cognitive Tutor (Koedinger, 2002) from Carnegie Mellon University and the ASSISTment Tutoring System (Razzaq et al., 2005) from the Worcester Polytechnic Institute (We refer to the Middle School Tutor by its original name, since this was the version studied in this paper; An updated version of this tutor is now distributed commercially by Carnegie Learning, Inc. as Bridge to Algebra).

Within each environment, each student works independently, completing Mathematics problems online. Both systems teach and then assess students' proficiency with a variety of knowledge components. The Cognitive Tutor's content is motivated by state-mandated Mathematics curricular standards within the United States, organized into lessons by curricular themes. The ASSISTment system provides a centralized certification system where content is vetted by domain experts at WPI and promoted to teachers within the system. The inspiration for the themes of those problem sets began out of questions found in a state Mathematics exam, the Massachusetts Comprehensive Assessment System (Razzaq et al., 2005). Today, ASSISTments focuses primarily on allowing teachers to selectively assign daily content to build knowledge component proficiency.

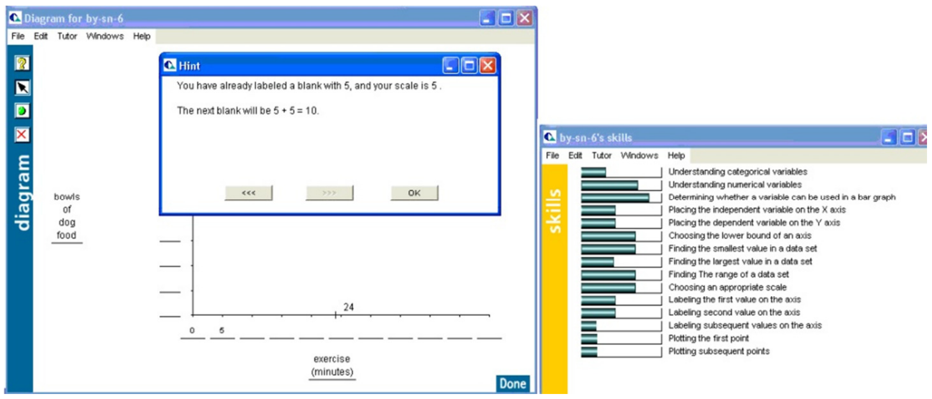
To encourage learning, both tutors have multiple means of supporting learners encountering difficulties with specific knowledge components, such as choosing the X axis variable for a graph, or computing the volume of a cube. Both environments include buggy messages, which are tailored feedback for when a common misconception can be detected within student behavior. Each system supports multi-level on-demand hints to students. One difference is that problem-solving steps are always reified within the Middle School Cognitive Tutor, whereas these steps are only reified with ASSISTments if the student makes an error or requests help. In the following sections, we will discuss these systems in greater detail, along with the population that used each of these systems within the data sets obtained for the research in this paper.

2.1 The Cognitive Tutor

Cognitive Tutors (Koedinger & Corbett, 2006) break down each mathematics problem into the steps used to solve it in order to reify student thinking and provide feedback on each step of the problem-solving process. Within Cognitive Tutors, each mathematics problem typically is composed of a set of problem steps, and each problem step is typically mapped to a single knowledge component. Cognitive Tutors assess student knowledge using Bayesian Knowledge Tracing (Corbett & Anderson, 1995), which uses the student's performance history within the system to calculate a running estimate of the probability that the student is proficient at particular knowledge components. Given these probabilities, the system implements cognitive mastery learning (Corbett, 2001), which is a tutor style that gives students practice on a knowledge component until the student has demonstrated mastery of that skill or concept. Cognitive Tutors give tailored feedback when a student's answer is indicative of a known misconception, and offer multi-level on-demand hints to students, which start at a conceptual level and become increasingly specific until the student is given the answer (as in Figure 2.1).

The analyses presented in this paper to derive our model are conducted on data from 232 students' use of a Cognitive Tutor curriculum for middle school mathematics (Koedinger, 2002), during the 2002-2003 school year. All of the students were enrolled in mathematics classes in one middle school in the Pittsburgh suburbs that used Cognitive Tutors two days a week as part of its regular mathematics curriculum, year round. None of the classes were composed predominantly of gifted or special needs students. The students were in the 6th, 7th, and 8th grades (approximately 12-14 years old), but all used the same curriculum (it was an advanced curriculum for 6th graders, and a remedial curriculum for 8th graders).

Each of these students worked through a subset of 35 different lessons within the Cognitive Tutor curriculum, covering a diverse selection of material from the middle school mathematics curriculum. Middle school mathematics, in the United States, generally consists of a diverse collection of topics, and these students' work was representative of that diversity, including lessons on combinatorics, decimals, diagrams, 3D geometry, fraction division, function generation and solving, graph interpretation, probability, and proportional reasoning. These students made 581,785 transactions (either entering an answer or requesting a hint) on 171,987 problem steps covering 253 knowledge components.



You have already labeled a blank with 5, and your scale is 5 .

The next blank will be 5 + 5 = 10.

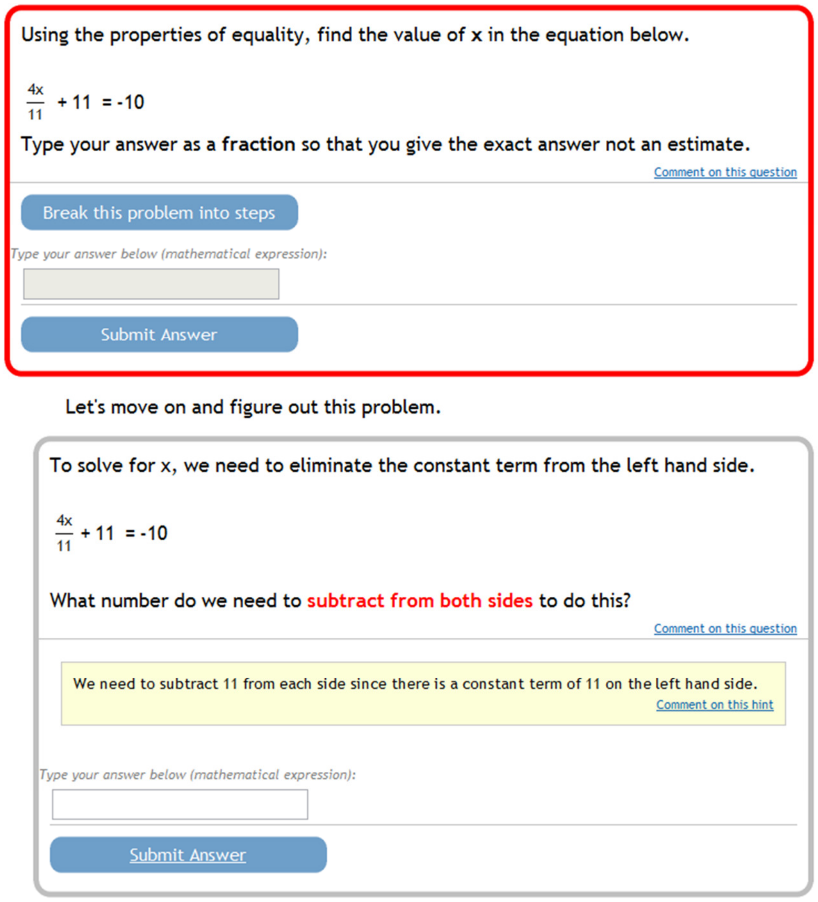
- Understanding categorical variables
- Understanding numerical variables
- Determining whether a variable can be used in a bar graph
- Placing the independent variable on the X axis
- Placing the dependent variable on the Y axis
- Choosing the lower bound of an axis
- Finding the smallest value in a data set
- Finding the largest value in a data set
- Finding The range of a data set
- Choosing an appropriate scale
- Labeling the first value on the axis
- Labeling second value on the axis
- Labeling subsequent values on the axis
- Plotting the first point
- Plotting subsequent points

Fig 2.1. A student reads a bottom-out hint within the Middle School Cognitive Tutor.

290,698 additional transactions were not included in either these totals or in our analyses, because they were not labelled with KCs, information needed to apply Bayesian Knowledge Tracing.

2.2 ASSISTments

The other ITS studied in this paper is the ASSISTment tutoring system (Razzaq et al., 2005). ASSISTment is used to assess student proficiency, for homework assignments (Mendicino, Razzaq, & Heffernan, 2009), and for preparation for standardized exams (Koedinger, McLaughlin, & Heffernan, in press). Within this paper, we analyze data drawn from students using the Mastery Learning feature of ASSISTments, where a student repeatedly receives problems focusing on a specific knowledge component until the student demonstrates mastery. Within ASSISTments' mastery learning, proficiency is assessed in a very different manner than in Cognitive Tutors. ASSISTments allows teachers to set a threshold for the number of problems a student must correctly answer in a row in order to be considered proficient at that knowledge component. That threshold is termed the Mastery Limit. Though this approach is very likely to assess mastery less accurately than Bayesian Knowledge Tracing, it is preferred by some teachers as being easier to understand and control. Within this particular study,


Using the properties of equality, find the value of x in the equation below.

$\frac{4x}{11} + 11 = -10$

Type your answer as a **fraction** so that you give the exact answer not an estimate.

Comment on this question
Break this problem into steps
Type your answer below (mathematical expression):
Submit Answer

Let's move on and figure out this problem.

To solve for x, we need to eliminate the constant term from the left hand side.

$\frac{4x}{11} + 11 = -10$

What number do we need to subtract from both sides to do this?

We need to subtract 11 from each side since there is a constant term of 11 on the left hand side.

Figure 2.2. A student reading a hint for the first level of scaffolding in the ASSISTment tutoring system.

problem sets had a Mastery Limit of either 3 or 5. In order to prevent exhaustion and wasted time, students were allowed to attempt no more than 10 problems pertaining to a specific knowledge component each day. When that number was exceeded, the student was locked out of the problem set until the next day.

As with Cognitive Tutors, ASSISTments provides feedback on incorrect answers and multi-level hints that terminate with bottom-out hints. The ASSISTments system also offers scaffolding, which breaks down the current problem into simpler steps, in order to reify the thinking needed to solve the problem. The intention is to make visible the specific part of the student's thought process that is incorrect. Each step of the scaffolding is a problem unto itself, capable of containing multi-level hints, and each requiring a new answer. The last step of scaffolding always requires the student to again attempt the original question. Within ASSISTments for mathematics, each mathematics problem typically maps to a single knowledge component, with a scaffold step in some cases (but not at all) involving a different knowledge component.

The analyses presented in this paper to derive our model are conducted on data from 4187 students' use of mathematics ASSISTments between December 2008 and March 2010. This sample was primarily composed of students in Massachusetts, but included substantial proportions of students from other parts of the USA, including Virginia and South Carolina. The sample was primarily composed of high school students, although some middle school students were also included. These students completed Mastery Learning problem sets involving 53 knowledge components, across a range of areas of mathematics, including algebra, probability, number sense with fractions and decimals, geometry, and graph interpretation. The patterns of usage varied considerably between the many schools involved in this data set. These students made 413,428 transactions (either entering an answer or requesting a hint) on 179,144 problems.

3 Detecting Learning, Moment-By-Moment

Within this section, we present a model that predicts the probability that a student has learned a specific knowledge component at a specific problem step. We refer to this probability as P(***J***), short for "Just Learned". This model is developed using a procedure structurally similar to Baker, Corbett, & Aleven's (2008) contextualization of the guess and slip parameters of Bayesian Knowledge Tracing, using a two-step process. Considerably greater detail on this procedure will be given in the following sections, but we give a brief summary here.

First, training labels of the probability that a student learned a KC at a specific problem step are generated. The development of these labels is based on the overall idea that learning is indicated when a student does not know a skill at one point and then starts performing correctly afterwards. These training labels are generated using a combination of predictions of current student knowledge from standard Bayesian Knowledge Tracing and data on future correctness, integrated using Bayes' Theorem. This process generates training labels of the probability that a student learned a KC at a specific problem step. In essence, we use evidence from both the past and future to assess the probability that learning occurred at a specific time.

Using these labels, a model is trained. This model uses a broad feature set, but includes absolutely no data from the future. The result is a model that can be used either at run-time or retrospectively, to assess the probability that a KC is learned at each practice opportunity. We present results for this process on two distinct data sets, from a Cognitive Tutor and Math ASSISTments.

3.1 Labeling Process

The first step of our process is to label each problem step ***N*** in the data set (i.e. the ***N***th opportunity for the given student to use the given KC) with the probability that the student learned the KC at that time, to serve as inputs for machine learning. Determining exactly when a student is thinking about a specific problem step is a non-trivial task, as students often continue thinking about a step even after entering a correct answer (e.g. Shih, Koedinger, & Scheines, 2008). Our specific working definition of "learning at step ***N***" is learning the KC between the instant after the student enters their first answer for step ***N***, and the instant that the student enters their first answer for step ***N+1***. In doing so, we likely include some amount of time when the student is thinking about the next step and omit some amount of time when the student is thinking about the current step (specifically, time before the first answer on the current step). It is impossible under current methods to avoid some bias; we choose to bias in this direction because we believe that learning processes such as self-explanation are more likely to occur after an answer (whether correct or incorrect) or help request, than before the student answers a step for the first time (at which point the student usually does not know for certain if their answer and process is correct).

We label step ***N*** using information about the probability the student knew the KC before answering on step ***N*** (from Bayesian Knowledge Tracing) and information on performance on the two following steps (***N+1***, ***N+2***). Using data from future actions gives information about the true probability that the student learned the KC during the actions at step ***N***. For instance, if the student probably did not know the KC at step ***N*** (according to Bayesian Knowledge Tracing), but the first attempts at steps ***N+1*** and ***N+2*** are correct, it is relatively likely that the student learned the KC at step ***N***. Correspondingly, if the first attempts to answer steps ***N+1*** and ***N+2*** are incorrect, it is relatively unlikely that the student learned the KC at step ***N.***

We assess the probability that the student learned the KC at step ***N,*** given information about the actions at steps ***N+1*** and ***N+2*** (which we term $\boldsymbol{A_{+1+2}}$), as:

$$\mathrm{P}(\boldsymbol{J}) = \mathrm{P}(\sim\boldsymbol{L_n} \wedge \boldsymbol{T} \mid \boldsymbol{A_{+1+2}})$$

Note that this probability is assessed as P($\sim\boldsymbol{L_n} \wedge \boldsymbol{T}$), the probability that the student did not know the KC and learned it, rather than P(***T***). Within Bayesian Knowledge Tracing, the semantic meaning of P(***T***) is actually P($\boldsymbol{T} \mid \sim\boldsymbol{L_n}$): P(***T***) is the probability that the KC will be learned, if it has not yet been learned. P(***T***)'s semantics, while highly relevant for some research questions (cf. Beck et al., 2008; Koedinger, 2002), are not an indicator of the probability that a KC was learned at a specific moment. This is because the probability that a student learned a KC at a specific step can be no higher than the probability that they do not currently know it. P(***T***), however, can have any value between 0 and 1 at any time (though recall that P(***T***)'s value is constant for each skill, both within our P(***J***) model and within classic Bayesian Knowledge Tracing). For low values of P($\boldsymbol{L_n}$), P(***T***) will approximate the probability that the student just learned the KC. However, for high values of P($\boldsymbol{L_n}$), P(***T***) can take on extremely high values even though the probability that the KC was learned at that moment is very low.

We can find P(***J***)'s value with a function using Bayes' Rule:

$$\mathrm{P}(\sim\boldsymbol{L_n} \wedge \boldsymbol{T} \mid \boldsymbol{A_{+1+2}}) = \frac{P(\boldsymbol{A_{+1+2}} \mid \sim\boldsymbol{L_n} \wedge \boldsymbol{T}) * P(\sim\boldsymbol{L_n} \wedge \boldsymbol{T})}{P(\boldsymbol{A_{+1+2}})}$$

The base probability P($\sim L_n$ ^ T) can be computed fairly simply, using the student's current value for P($\sim L_n$) from Bayesian Knowledge Tracing, and the Bayesian Knowledge Tracing model's value of P(T) for the current KC:

$$P(\sim \boldsymbol{L_n} \wedge \boldsymbol{T}) = P(\sim \boldsymbol{L_n})P(\boldsymbol{T})$$

The probability of the actions at time $N+1$ and N+2, P(A_{+1+2}), is computed as a function of the probability of the actions given each possible case (the KC was already known, P(L_n), the KC was unknown but was just learned, P($\sim L_n$ ^ T), or the KC was unknown and was not learned, P($\sim L_n$ ^ $\sim T$)), and the contingent probabilities of each of these cases.

$$P(\boldsymbol{A_{+1+2}}) = P(\boldsymbol{A_{+1+2}} \mid \boldsymbol{L_n})\, P(\boldsymbol{L_n}) + P(\boldsymbol{A_{+1+2}} \mid \sim \boldsymbol{L_n} \wedge \boldsymbol{T})\, P(\sim \boldsymbol{L_n} \wedge \boldsymbol{T}) + P(\boldsymbol{A_{+1+2}} \mid \sim \boldsymbol{L_n} \wedge \sim \boldsymbol{T})\, P(\sim \boldsymbol{L_n} \wedge \sim \boldsymbol{T})$$

The probability of the actions at time $N+1$ and N+2, in each of these three cases, is a function of the Bayesian Knowledge Tracing model's probabilities for guessing (G), slipping (S), and learning the KC (T). In order to calculate the probability of each possible case of estimated student knowledge, we must consider all four potential scenarios of performance at actions $N+1$ and N+*2*. In the formulas below, correct answers are written **C** and non-correct answers (e.g. errors or help requests) are written **~C**. The possible scenarios are: correct/correct (**C**, **C**); correct/wrong (**C**, **~C**); wrong/correct (**~C**, **C**); and wrong/wrong (**~C**, **~C**):

$$P(\boldsymbol{A_{+1+2}} = \boldsymbol{C}, \boldsymbol{C} \mid \boldsymbol{L_n}) = P(\sim \boldsymbol{S})^2 \qquad P(\boldsymbol{A_{+1+2}} = \boldsymbol{C}, \sim \boldsymbol{C} \mid \boldsymbol{L_n}) = P(\boldsymbol{S})P(\sim \boldsymbol{S})$$
$$P(\boldsymbol{A_{+1+2}} = \sim \boldsymbol{C}, \boldsymbol{C} \mid \boldsymbol{L_n}) = P(\boldsymbol{S})P(\sim \boldsymbol{S}) \qquad P(\boldsymbol{A_{+1+2}} = \sim \boldsymbol{C}, \sim \boldsymbol{C} \mid \boldsymbol{L_n}) = P(\boldsymbol{S})^2$$
$$P(\boldsymbol{A_{+1+2}} = \boldsymbol{C}, \boldsymbol{C} \mid \sim \boldsymbol{L_n} \wedge \boldsymbol{T}) = P(\sim \boldsymbol{S})^2 \qquad P(\boldsymbol{A_{+1+2}} = \boldsymbol{C}, \sim \boldsymbol{C} \mid \sim \boldsymbol{L_n} \wedge \boldsymbol{T}) = P(\boldsymbol{S})P(\sim \boldsymbol{S})$$
$$P(\boldsymbol{A_{+1+2}} = \sim \boldsymbol{C}, \boldsymbol{C} \mid \sim \boldsymbol{L_n} \wedge \boldsymbol{T}) = P(\boldsymbol{S})P(\sim \boldsymbol{S}) \qquad P(\boldsymbol{A_{+1+2}} = \sim \boldsymbol{C}, \sim \boldsymbol{C} \mid \sim \boldsymbol{L_n} \wedge \boldsymbol{T}) = P(\boldsymbol{S})^2$$

$$P(\boldsymbol{A_{+1+2}} = \boldsymbol{C}, \boldsymbol{C} \mid \sim \boldsymbol{L_n} \wedge \sim \boldsymbol{T}) = P(\boldsymbol{G})P(\sim \boldsymbol{T})P(\boldsymbol{G}) + P(\boldsymbol{G})P(\boldsymbol{T})P(\sim \boldsymbol{S})$$
$$P(\boldsymbol{A_{+1+2}} = \boldsymbol{C}, \sim \boldsymbol{C} \mid \sim \boldsymbol{L_n} \wedge \sim \boldsymbol{T}) = P(\boldsymbol{G})P(\sim \boldsymbol{T})P(\sim \boldsymbol{G}) + P(\boldsymbol{G})P(\boldsymbol{T})P(\boldsymbol{S})$$
$$P(\boldsymbol{A_{+1+2}} = \sim \boldsymbol{C}, \boldsymbol{C} \mid \sim \boldsymbol{L_n} \wedge \sim \boldsymbol{T}) = P(\sim \boldsymbol{G})P(\sim \boldsymbol{T})P(\boldsymbol{G}) + P(\sim \boldsymbol{G})P(\boldsymbol{T})P(\sim \boldsymbol{S})$$
$$P(\boldsymbol{A_{+1+2}} = \sim \boldsymbol{C}, \sim \boldsymbol{C} \mid \sim \boldsymbol{L_n} \wedge \sim \boldsymbol{T}) = P(\sim \boldsymbol{G})P(\sim \boldsymbol{T})P(\sim \boldsymbol{G}) + P(\sim \boldsymbol{G})P(\boldsymbol{T})P(\boldsymbol{S})$$

Once each action is labelled with estimates of the probability P(***J***) that the student learned the KC at that time, we use these labels to create machine-learned models that can accurately predict P(***J***) at run time. The original labels of P(***J***) were developed using future knowledge, but the machine-learned models predict P(***J***) using only data about the action itself (no future data).

3.2 Features

In order to predict the training labels of P(***J***) created in the previous step, we distil a set of features that can be used as predictors. These features are quantitative (or binary) descriptors of key aspects of each problem step that have a reasonable potential to be statistically associated with the construct of interest, whether learning occurred at a specific moment. These features are then used within machine learning (discussed in the next section).

For each problem step, we used a set of features describing the first action on problem step ***N***. In the case of the Cognitive Tutor, the list consisted of 23 features previously distilled to use in the development of contextual models of guessing and slipping (cf. Baker, Corbett, & Aleven, 2008). These features had in turn been used in prior work to develop automated detectors of off-task behavior (Baker, 2007) and gaming the system (Baker et al., 2008). In the case of ASSISTments, a similar but non-identical list of 22 features was distilled (differing primarily due to the different features and organization of problems in Math ASSISTments). The actual features selected for incorporation into the final models is given in Tables 3.1 and 3.2. The list of features inputted into the machine learning algorithm was:

- Assessments of correctness:
    - Percent of all past problems that were wrong on this KC.
    - Total number of past problems that were wrong on this KC.
    - Number of last 5 problems that were wrong.
    - Number of last 8 problems that were wrong.

- Measurements of time:
  - Time taken (SD faster/slower than average across all students).
  - Time taken in last 3 actions (SD off average).
  - Time taken in last 5 actions (SD off average).
  - Total time spent on this KC across all problems.
  - Time since the current KC was last seen.
- Data on hint usage:
  - First response is a help request.
  - Bottom-out hint is used.
  - Number of last 8 problems that used the bottom-out hint.
  - Second to last hint is used (ASSISTments only) – indicates a hint that gives considerable detail but is not quite bottom-out.
  - Number of last 5 problems that included a help request.
  - Number of last 8 problems that included a help request.
- Data on scaffolding usage (ASSISTments only):
  - Problem ends with scaffolding.
  - Problem ends with automatic scaffolding.
  - The problem is scaffolding of a prior problem.
  - Total scaffolding opportunities for this KC in the past.
- Other measurements:
  - Total problems attempted in the tutor so far.
  - Total practice opportunities on this KC so far.
  - Response is chosen from a list of answers (Multiple choice, etc).
  - Response is filled in (No list of answers available).
  - Working during school hours (between 7:00 am and 3:00 pm) (ASSISTments only).

It may seem counter-intuitive to use a feature set that was designed originally to capture off-task behavior and gaming for detecting learning moment-by-moment, but this approach has advantages; in particular, if it leads to a model with reasonable goodness of fit, then this suggests that developing a successful detector of learning moment-by-moment does not require an extensive new process of feature engineering. Feature engineering and extraction can often be one of the most time-consuming aspects of educational data mining and data mining in general (for instance, devising what the features should be, and selecting the cut-offs used in features such as ‘the number of the last 5 problems that included a help request’). Bypassing this time-consuming step increases the feasibility of creating models of this nature. Discussion of potential additional features, and further feature engineering we have conducted for this model, is described in detail in the discussion section. In addition to these features, we also included two additional features that were used in prior models of gaming the system and off-task behavior. These features are the probability that the student knew the KC before the first attempt on action $\boldsymbol{N}$, P($\boldsymbol{L_{n-1}}$), and the probability that the student knew the KC after the first attempt on action $\boldsymbol{N}$, P($\boldsymbol{L_n}$). There are some arguments against including these features, as P(~$\boldsymbol{L_n}$) is part of the construct being predicted, P(~$\boldsymbol{L_n}$ ^ $\boldsymbol{T}$). However, the goal of this model is to determine the probability of learning, moment-by-moment, and the students' current and previous knowledge levels, as assessed by Bayesian Knowledge Tracing, are useful information towards this goal. In addition, other parameters in the model will be more interpretable if these features are included. Without these terms, it would be difficult to determine if a parameter was predicting $\boldsymbol{T}$ or ~$\boldsymbol{L_n}$. With these terms, we can have greater confidence that parameters are predictive of learning (not just whether the KC was previously unknown) because $\boldsymbol{L_n}$ is already accounted for in the model. However, in accordance with potential validity concerns stemming from including P($\boldsymbol{L_{n-1}}$) and P($\boldsymbol{L_n}$) in the model, we will also present goodness-of-fit statistics from models not including these features.

3.3 Machine Learning

Given the labels and the model features for each student action within the tutor, we conducted linear regression within RapidMiner (Mierswa et al., 2006) to develop models that predict P($\boldsymbol{J}$). This resulted in a set of numerical predictions of P($\boldsymbol{J}$), one for each problem step that a student completed. In each case, M5' feature selection (Hall, 2000) was used to determine which features were incorporated into the models. Linear regression with M5' feature selection creates regression trees, a tree of linear regression models, and then conducts linear regression on the set of features used in the tree. Although this approach might seem somewhat non-straightforward, compared to simpler approaches such as stepwise regression, it has been shown to lead to better model performance than several other feature selection algorithms (Hall, 2000) and is now the default setting for linear regression in several data mining packages, including both RapidMiner (Mierswa et al., 2006) and Weka (Witten & Frank, 2005). The machine learned models generated for each system (including all features in the final models) are listed below in Table 3.1 and Table 3.2.

To validate the generalizability of our models, we checked our results with 6-fold cross-validation, at the student level (e.g. detectors are trained on five groups of students and tested on a sixth group of students). By cross-validating at this level, we increase confidence that detectors will be accurate for new groups of students.

The goodness of the models was validated using the Pearson correlation coefficient between the training labels of P(***J***) for each step, and the values predicted for P(***J***) for the same step by the machine-learned models. As both set of values are quantitative, and there is a one-to-one mapping between training labels and predicted values, linear correlation is a reasonable metric.

3.4 Results

Overall, the models produced through machine learning were successful at predicting P(***J***). The full model, trained on the full set of features, achieved good correlation between the training labels and model predictions, for each tutoring system. For the Cognitive Tutor data, the model achieved a correlation of 0.446 to the training labels previously generated for each problem step, within 6-fold student-level cross-validation. Similarly, the model for ASSISTments data achieved a correlation coefficient of 0.397 to the training labels previously generated for each problem step.

The two models are shown below in Tables 3.1 and 3.2. As with any multiple-parameter linear regression model (and most other model frameworks as well), interpretability of the meaning of any individual parameter is not entirely straightforward. This is because every parameter must be considered in the context of all of the other parameters – often a feature's sign can flip based on the other parameters in the model. Hence, significant caution should be taken before attempting to interpret specific parameters as-is. It is worth noting that approaches that attempt to isolate specific single features (cf. Beck et al., 2008) are significantly more interpretable than the internal aspects of a multiple parameter regression model such as this one. It is also worth remembering that these features apply to the first action of problem step ***N*** whereas the labels pertain to the student's learning between the first action of problem step ***N*** and the first action of problem step ***N+1***. Hence, the features of this model can be interpreted more as representing the immediate antecedents of the moment of learning than as representing the moment of learning itself – though they do accurately predict learning, moment-by-moment.

Although the degree of correlation was acceptable, one curious aspect of this model is that it tended to underestimate values of P(***J***), particularly those that were relatively high in the original labels (e.g. >0.02). The difference between the model values of P(***J***) and the original label is highly correlated to the original label, with a correlation of 0.95 in the Cognitive Tutor and 0.87 in ASSISTments. Hence, the predicted values of P(***J***) for training labels with high values remained higher than the predicted values of P(***J***) for training labels with lower values (hence the model's reasonable correlation to the labels). However, the predicted values of P(***J***) for training labels with high values were lower, in absolute terms, than the original training labels for those data points. This problem could be addressed by weighting the (rarer) high values more heavily during model-fitting, although this approach would likely reduce overall correlation. Another possible solution would be to fit the data using a logarithmic (or other) function that scales upwards more effectively than a linear function; as will be seen later, the differences between maximum and minimum spikiness are large enough that non-linear regression may be more appropriate than our current approach. Nevertheless, within the current model it is likely to be more straightforward to interpret differences in P($\boldsymbol{J}$) than absolute values.

As discussed earlier, one potential concern with these models is that they incorporate $\boldsymbol{L_{n-1}}$ and $\boldsymbol{L_n}$ while ~$\boldsymbol{L_n}$ is used in the training labels. As discussed above, we do not consider this a primary concern, as our main goal is to fit the learning part of the equation (rather than the "already-learned" part); but to validate that our models are not simply predicting ~$\boldsymbol{L_n}$, we re-fit the models without $\boldsymbol{L_{n-1}}$ and $\boldsymbol{L_n}$. When models were fit for the Cognitive Tutor and ASSISTments that excluded $\boldsymbol{L_n}$ and $\boldsymbol{L_{n-1}}$, these models achieved lower cross-validated correlations than the full models. For Cognitive Tutors, a correlation of 0.438 was achieved, as compared to the correlation of 0.446 obtained for the full model. For ASSISTments, a correlation of 0.301 was achieved, as compared to the correlation of 0.397 obtained for the full model. We can compute the statistical significance of the difference in correlation (between the full and restricted models) in a way that accounts for the non-independence between students, by computing a test of the significance of the difference between two correlation coefficients for correlated samples (cf. Ferguson, 1971) for each student, and then aggregating across students using Stouffer's Z (Rosenthal & Rosnow, 1991). According to this test, the difference between the two models is highly statistically significant, both for the Cognitive Tutor data, $Z=116.51$, $p<0.0001$, and for the ASSISTments data, $Z = 66.34$, $p<0.001$.

Table 3.1. The machine learned model of the probability of learning at a specific moment for the Cognitive Tutor. In the unusual case where output values fall outside the range {0,1}, they are bounded to 0 or 1. The model is expressed as a regression equation, with each feature's non-unitized parameter coefficient (weight) given. Computing the value of the equation gives the predicted value of P(***J***).

Feature	**P(*J*) =**
Answer is correct	- 0.0023
Answer is incorrect	+ 0.0023
Action is a help request	- 0.00391
Response is a string	+ 0.01213
Response is a number	+ 0.01139
Time taken (SD faster (-) / slower (+) than avg. across all students)	+ 0.00018
Time taken in last 3 actions (SD off avg. across all students)	+ 0.000077
Total number of times student has gotten this KC wrong total	- 0.000073
Number of times student requested help on this KC, divided by number of problems	- 0.00711
Number of times student made errors on this KC, divided by number of problems	+ 0.0013
Total time taken on this KC so far (across all problems), in seconds	+ 0.0000047
Number of last 5 actions which involved same interface element	- 0.00081
Number of last 8 actions that involved a help request	+ 0.00137
Number of last 5 actions that were wrong	+ 0.00080
At least 3 of last 5 actions involved same interface element & were wrong	- 0.037
Number of opportunities student has already had to use current KC	- 0.0000075
The probability the student knew the KC, after the current action (L_n)	- 0.053
The probability the student knew the KC, before the current action (L_{n-1})	+ 0.00424
Constant Term	+ 0.039

Table 3.2. The machine learned model of the probability of learning at a specific moment for the ASSISTments system. In the unusual case where output values fall outside the range {0,1}, they are bounded to 0 or 1. The model is expressed as a regression equation, with each feature's non-unitized parameter coefficient (weight) given. Computing the value of the equation gives the predicted value of P(***J***).

Feature	**P(*J*) =**
Answer is correct	- 0.0429
Action is a hint request	- 0.0216
Current problem is original (Not scaffolding of another)	+ 0.0078
Response is input by the user (Not just selected from a list of multiple choice)	+ 0.0058
Time taken to complete the current problem	+ 0.0215
Time taken in last 3 actions (SD off avg. across all students)	+ 0.1866
Total number of times student has gotten this KC wrong total	- 0.0798
Total time taken on this KC so far (across all problems), in seconds	- 0.0346
Number of last 5 actions that involved a help request	- 0.0953
Number of last 8 actions that were wrong	- 0.0401
Percentage of past problems that the student has gotten wrong	+ 0.0184
Amount of time that has passed since this KC was last seen	- 0.0399
Whether or not the problem was completed during school hours (M-F 8:00-3:00)	- 0.0038
Total number of problems the student has attempted altogether in the system	+ 0.0078
The probability the student knew the KC, before the current action (L_{n-1})	- 0.0605
Constant term	+ 0.0957

One interesting aspect of this model (and the original labels) is that the overall chance of learning a KC on any single step is relatively low within the two tutors. However, there are specific circumstances where learning is higher. Within both systems, many of these circumstances correlate to time spent, and the student's degree of persistence in attempting to respond. In addition, in both systems, there is a positive correlation associated with an incorrect answer, potentially suggesting that students learn by making errors and then considering why the answer was incorrect. In particular, within the Cognitive Tutor, larger numbers of past errors appear to predict more current learning than larger numbers of past help requests. This result appears at a surface level to be in contrast to the findings from (Beck, Chang, Mostow, & Corbett, 2008), but is potentially explained by the difference between learning from requesting help once – the grain-size studied in (Beck, Chang, Mostow, & Corbett, 2008) – and learning from requesting the same help sequence many times across problems. It may be that learning from errors (cf. VanLehn, Siler, & Murray, et al., 2003) is facilitated by making more errors, but that learning from help does not benefit from reading the same help multiple times. Another possible explanation for this difference is that the help studied in (Beck et al., 2008) was much briefer than the multi-step problem-solving hints used in the Cognitive Tutor and ASSISTments studied here.

4 Improvements to the Model

Though the model proposed in this paper has been successful at predicting its training labels, and in turn at producing a distilled measure that can predict students' final knowledge, there are several ways that this model can be improved and refined.

First, it may be possible to improve the quality of the model's training labels. The approach proposed in this paper is only one way to infer learning, moment-by-moment. To give a very simple example, data from only two future actions was utilized in generating the training labels. It is possible that using data from a greater number of future actions may result in more accurate labels; correspondingly, it is possible that the probability of guess or slip for many problems may be sufficiently low that only one future action is needed, and incorporating a second future action will increase the noise.

Additionally, the equations used in this paper are currently based off an unmodified form of Bayesian Knowledge Tracing – however, recent work in our group has shown benefits from using contextual models of guess and slip (e.g. Baker, Corbett, & Aleven, 2008), including improved prediction of post-test scores (Baker, Corbett, et al., 2010). It is possible that using contextual estimations of guess and slip when generating training labels may lead to higher precision – although there is correspondingly some risk that models of P(***J***) generated in this fashion may end up over-fitting to the errors in the contextual guess and slip models.

Other approaches to generating training labels are also possible. For instance, it may be reasonable to compute the derivative of each student's learning curve. This would be impractical to do for correctness (which is binary, resulting in a curve that is not smooth), but could very well be feasible for a more quantitative measure of student performance, such as time, or assistance score (cf. Feng, Heffernan, & Koedinger, 2006).

Even when using the exact same formal approach to generating training labels, it may also be possible to improve P(***J***) accuracy by investigating other methods for calculating values of P($\boldsymbol{L_n}$) and P($\boldsymbol{L_{n-1}}$) as components for P(***J***) models. Currently these features are generated based on models created via brute force/grid search. However, other researchers have found evidence suggesting benefits for Expectation Maximization (cf. Gong, Beck, & Heffernan, 2010) and for contextualized estimation of student initial knowledge and learning rates (cf. Pardos & Heffernan, 2010). Rather than replacing the estimates of P($\boldsymbol{L_n}$) and P($\boldsymbol{L_{n-1}}$) in the existing model with estimates generated in another fashion, it is possible to use all of these estimates in a combined model, and determine if the different estimates of learning so far have unique and separate variance for predicting learning at each moment.

One issue to consider in attempting to improve the training labels used in this model is how to validate that one set of training labels is better than another. One method may be to compare the models obtained from different training labels. For example, the approach used here led to an accurate prediction of a widely-used measure of final knowledge in the tutor, final P($\boldsymbol{L_n}$) from Bayesian Knowledge Tracing; therefore, two sets of training labels could be compared in terms of how well the resultant models predict this measure. There are other potential comparisons of the resultant models that can be conducted as well – for instance, measures of learning on a post-test would provide an assessment of whether the spikes of learning assessed by this method are associated with the transfer of knowledge out of the tutoring environment, and could be used to assess how well different training labels capture this measure of the generalizability of student learning.

Regardless of what the training labels are, it may also be possible to improve a model of P(***J***) by broadening and improving the feature set. Within this paper, we used a set of features that have been used for several previous problems, and were successful at fitting P(***J***). It seems likely that a set of features expressly engineered for this problem would perform even more successfully. Similarly, it may be possible to use estimates of other constructs, such as contextual guessing, slipping (Baker, Corbett, & Aleven, 2008), and gaming the system (Baker, Corbett, Roll, & Koedinger, 2008), within models of learning moment-by-moment.

4.1 After the First Response

After our original model, one approach that we did implement in an attempt to capture additional variance with the P(***J***) model was incorporating additional features that covered subsequent actions, i.e. tracking student behaviour in the tutor following a first response that is wrong or is a help request. We hypothesized that while many current models in the field primarily consider first responses, it is possible that student learning occurs in between those main responses where students have an opportunity to assess their second guess or continue to seek additional help. As noted earlier, our model seems to indicate that it is following an incorrect response that a student is more likely to acquire knowledge, and so it logically follows that data about what occurs after that wrong answer but before the next problem is where the learning actually happens. With that in mind, we devised an expanded feature set for P(***J***) that included 51 features, specifically focusing on actions after the first response. They are listed here:

- Data on Hint Usage:
  - Action is a help request and a first response.

    - Problem ended with a hint immediately followed by a correct response.
    - Bottom out hint was used.
    - Second to last hint was used. Doesn't count if there were <= 2 hints total.
    - Number of hints taken after the first response.
    - Percent of non-first response actions from the past where the student has requested help on this KC.
    - Number of last 5 first responses that involved help request.
    - Number of last 8 first responses involved help request.
    - Number of last 8 problems that ended in using a bottom out hint.
- Assessments of Correctness:
    - Percent of past opportunities where student has made errors on this KC.
    - Percent of past first response opportunities where student has made errors on this KC.
    - Total number of times student has gotten this KC wrong on the first attempt.
    - Number wrong actions so far on the same problem. Help actions don't count, these must be pure answer attempts that are incorrect.
    - Number of last 5 first responses which were wrong.
    - Number of last 8 first responses which were wrong.
- Measurements of Time (Pruned more than 300 seconds taken):
    - Time taken on first response (SD faster (-) or slower (+) than average across all students).
    - Time taken on last 3 first responses (calculated in SD off average across students).
    - Time taken on last 5 first responses (calculated in SD off average across students).
    - Total time taken on this KC so far on all actions.
    - Total time taken on first responses for this skill so far (across all problems).
    - Time taken so far on this problem (adding up all the actions so far).
    - Time divided by number of hints in this problem.
    - Time divided by number of incorrect responses on this problem.
    - Time since the KC was last seen in a first response.
    - Shortest amount of time taken before a hint in this problem.
    - Shortest amount of time taken after a hint in this problem.
    - Shortest amount of time taken before a wrong response in this problem.
    - Shortest amount of time taken after a wrong response in this problem.
    - Longest amount of time taken before a hint in this problem.
    - Longest amount of time taken after a hint in this problem.
    - Longest amount of time taken before a wrong response in this problem.
    - Longest amount of time taken after a wrong response in this problem.
- Data on scaffolding usage (ASSISTments only):
    - Problem ends by user choosing to scaffold.
    - Problem ended by automatically invoked scaffolding by the system.
    - Current problem is scaffolding of a prior problem.
    - Time taken on a first response that is scaffolding.
    - Time taken total on a problem that is scaffolding.
    - Total time on scaffolding ever for this skill.
    - Total number of scaffolding problems on this KC so far.
    - Number of problems on this KC that were scaffolding/problems.

    - Time taken on first response that is scaffolding (SD faster (-) or slower (+) than average across all students).
    - This is a first response, scaffolding, and a help request.
    - The total hint requests made after first responses on scaffolding problems.
- Other measurements:
    - Number of questions attempted ever in the system.
    - Number of first opportunities student has already had to use current KC.
    - Number of non-first actions student has spent on current KC.
    - Total number of incorrect non-first response actions on a KC.
    - Response is filled-in (Not selected from a list).
    - Response is multiple choice or chosen from some sort of list.
    - Action is during school hours (Considered between 8:00am – 3:00pm).
    - Total number of non-first attempt actions made on this problem.

Based on this feature set, we re-ran the same Linear Regression in Rapid Miner with 6 fold student-level cross validation as we did with the original models for ASSISTments and the Cognitive Tutor. This resulted in the model shown below in table 4.1.

Table 4.1. The machine learned model of the probability of learning at a specific moment for the ASSISTments system, including subsequent actions. In the unusual case where output values fall outside the range {0,1}, they are bounded to 0 or 1. The model is expressed as a regression equation, with each feature's non-unitized parameter coefficient (weight) given. Computing the value of the equation gives the predicted value of P(***J***).

Feature	**P(*J*) =**
Problem is not a scaffold of another problem	+ 0.0006526
Problem ends by using the bottom out hint	- 0.0030143
Total hints used in subsequent actions	- 0.0223965
Correct subsequent action made after a non-bottom out hint taken	+ 0.0022031
Percent of subsequent actions made so far that were wrong	- 0.006608
Second to last hint was used	- 0.002062
Number of past 8 problems that ended by using the bottom out hint	+ 0.0079025
Percent of subsequent actions made so far that were wrong	- 0.0018803
Number of wrong subsequent actions made on the current problem	- 0.01174506
Sum time taken on all subsequent actions for this problem	+ 0.0442851
Sum time taken for subsequent actions in last 3 problems (SD faster /slower than avg.)	+ 0.0071275
Sum time taken for subsequent actions in last 5 problems (SD faster/slower than avg.)	+ 0.0016349
Time taken on subsequent actions divided by incorrect actions	+ 0.0070249
Total amount of time spent on subsequent actions for this KC	+ 0.00035206
Shortest amount of time taken before a hint in this problem's subsequent actions	- 0.0279119
Longest amount of time taken before a hint in this problem's subsequent actions	+ 0.0034354
Longest amount of time taken before an action following a hint in this problem	+ 0.0071563
Longest amount of time taken before making an incorrect action in this problem	+ 0.00646102
Total subsequent actions ever taken for this KC	+ 0.001885819
Problem ends with system forcing student into scaffolding	- 0.00276552
Time taken to make a first response on a scaffolding problem	- 0.00719845
Time taken on subsequent actions for this problem, which is scaffolding	- 0.01870045
Sum time taken for subsequent actions in last 3 problems that were scaffolding	+ 0.1166828
Total time spent on subsequent actions on this KC in scaffolding	+ 0.00049546
Number of hints taken in subsequent actions on this scaffolding problem	- 0.04347385

In contrast to our expectations, we did not find that subsequent action features captured a significant amount of additional variance. Although this new 25 feature model did achieve a higher correlation coefficient of 0.429, the overhead involved in distilling these new features is intensive for implementation in a real world ITS. This would seem to indicate that a model to predict P(***J***) can be achieved without the effort of distilling subsequent action features, which is the approach that we took in the pilot study described in section 6. However, this result is not necessarily conclusive on the subject, and it may be possible that there are other highly relevant features that we did not test, or that in a different ITS with a different sample group a significant improvement can be seen. As described earlier, there are several ways to improve the accuracy of a P(***J***) model, and an expanded feature set it only one that we have explored so far.

5 The Spikiness of Student Learning

A key way that the model presented here can be scientifically useful is through its predictions, as components in other analyses. Machine-learned models of gaming the system, off-task behavior, and contextual slip have proven useful as components in many other analyses (cf. Baker, 2007; Cocea et al., 2009; Walonoski & Heffernan, 2006). Models of the moment of student learning may turn out to be equally useful.

One research area that models of the moment of student learning may shed light on is the differences between gradual learning (such as strengthening of a memory association) and learning given to "eureka" moments, where a KC is understood suddenly (cf. Lindstrom, 2008). Predictions of momentary learning for a specific student and KC can be plotted, and graphs which are "spiky" (e.g. which have sudden peaks of learning) can be distinguished from flatter

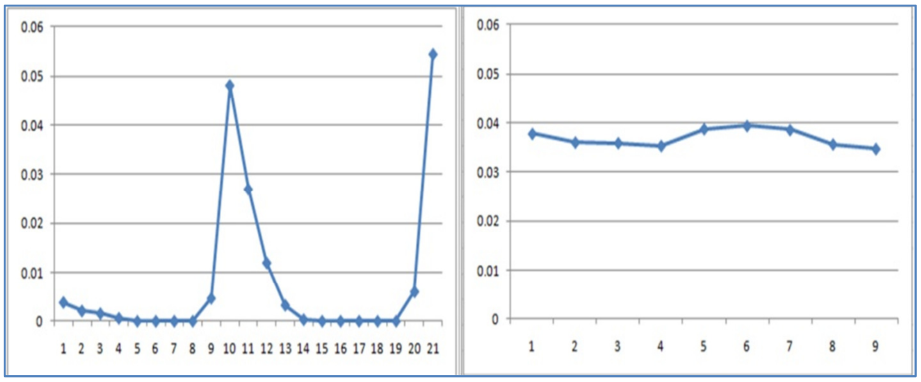


Fig. 5.1. An example of a single student's performance on a specific KC. "Entering a common multiple" (left) results in a "spiky" graph, indicating eureka learning. "Identifying the converted value in the problem statement of a scaling problem" (right) results in a relatively smooth graph, indicating more gradual learning. The X axis shows how many problem steps have involved the current KC, and the Y axis shows values of P(***J***).

graphs, which indicate more gradual learning. Examples of students' experiencing gradual learning and eureka learning are shown in Figure 5.1. The degree to which learning involves a eureka moment can be quantified through a measure of "spikiness", defined as the maximum value of P(***J***) for a student/KC pair, divided by the average value of P(***J***) for that same student/KC pair. This measure of spikiness is bounded between 1 (minimum spikiness) and positive infinity (maximum spikiness). Below we will detail the analysis of spikiness in the data from both the Cognitive Tutor and ASSISTments, first at the KC level and then at the student level.

As a quick note, it is worth mentioning that the graph on the left in Figure 5.1 shows two spikes, rather than just one spike. This pattern was fairly common in both data sets. Investigating this phenomenon is out of the scope of the current paper, but understanding why some spiky graphs have two spikes, and others have just one, will be an interesting and potentially fruitful area for future investigation.

5.1 Spikiness by Knowledge Component

Spikiness may be influenced by the number of opportunities to practice a KC, as more opportunities may (by random variation) increase the potential maximum value of P(***J***). Therefore, to compare spikiness between KCs, we only consider KCs practiced at least 6 times, and only consider the first 20 opportunities to use that KC.

Within our data from the Cognitive Tutor, spikiness values range for KCs between {1.12, 113.52}, M=8.55, SD=14.62. For ASSISTments, we found a range of {1.62, 12.45}, M=3.65, SD=1.79. As can be seen in figure 5.2, the most frequently occurring spikiness values in the Cognitive Tutor range from 1.25-1.75, somewhat lower than the modal range in ASSISTments, 2.75-4.75. It appears that the Cognitive Tutor spikiness values have a longer tail than the ASSISTments data, perhaps even with a second mode. This may imply that there are two groups of knowledge components in the Cognitive Tutor, a low-spikiness group and a high-spikiness group. There is some evidence for this possibility. The Cognitive Tutor tracks a larger number of knowledge components than ASSISTments, potentially including trivially easy tasks. The knowledge component with maximum spikiness of 113.52 is "ENTER-CORRECT-ANSWER-INTO-BOX". However, this KC's high maximum spikiness appears to be due to rare outlier performance. This KC was attempted 1488 times, with 1485 occurrences being correct responses (99.997%); essentially perfect performance for almost every student. Hence learning was almost zero for this skill in the tutor, making it possible for rare blips – specifically, initial slips followed by perfect performance – to lead to artifactually high estimates of P(***J***) in rare cases. Metrics such as the maximum are vulnerable to rare blips of this nature. By contrast, the KC with maximum spikiness in ASSISTments was "ordering integers", a genuine cognitive KC, which was attempted 2648 times with 2403 occurrences being correct (90.74%). Hence, it appears that some of the high spikiness seen in Cognitive Tutors comes from very easy KCs.

It will be a valuable area of future work to analyze the factors leading some knowledge components to have high spikiness, while others have low spikiness. We leave a complete analysis of the factors leading to differences in spikiness for future work, as this is likely to be a substantial question for future inquiry. However, as a preliminary statement, it appears that compound KCs, KCs that may actually be composed of multiple KCs, are less prone to spikes. For instance, the spikiest KC in ASSISTments is the straightforward skill of ordering integers, whereas the least spiky skill is algebraic solving. Algebraic solving may involve operations with both integers and real numbers or various algebraic properties. The second spikiest KC in ASSISTments is computing the perimeter of a polygon. Polygons come in many shapes and sizes, measured in different ways. The gradual learning that a non-spiky graph represents is potentially showing continual gains in cognitive understanding of the many sub-KCs that the current problem includes. On the other hand, ordering integers is a much finer-grained KC, and we can frequently see "eureka" style learning for students. This pattern is seen in both of the two tutors. Thus, we hypothesize that P(***J***) can be especially useful for finer-grain KC models. Another factor worth noting is that algebraic solving, as well as perimeter of a polygon, have the shared characteristic of having many prerequisite KCs. If prerequisites are clear, then tutors could potentially be biased by using P(***J***) to induce spikes for each sub-KC to eventually obtain mastery of the primary KC. As noted, it will be a valuable area of future study to see whether these two factors are generally predictive of differences in KC spikiness and what other factors predict spikiness.

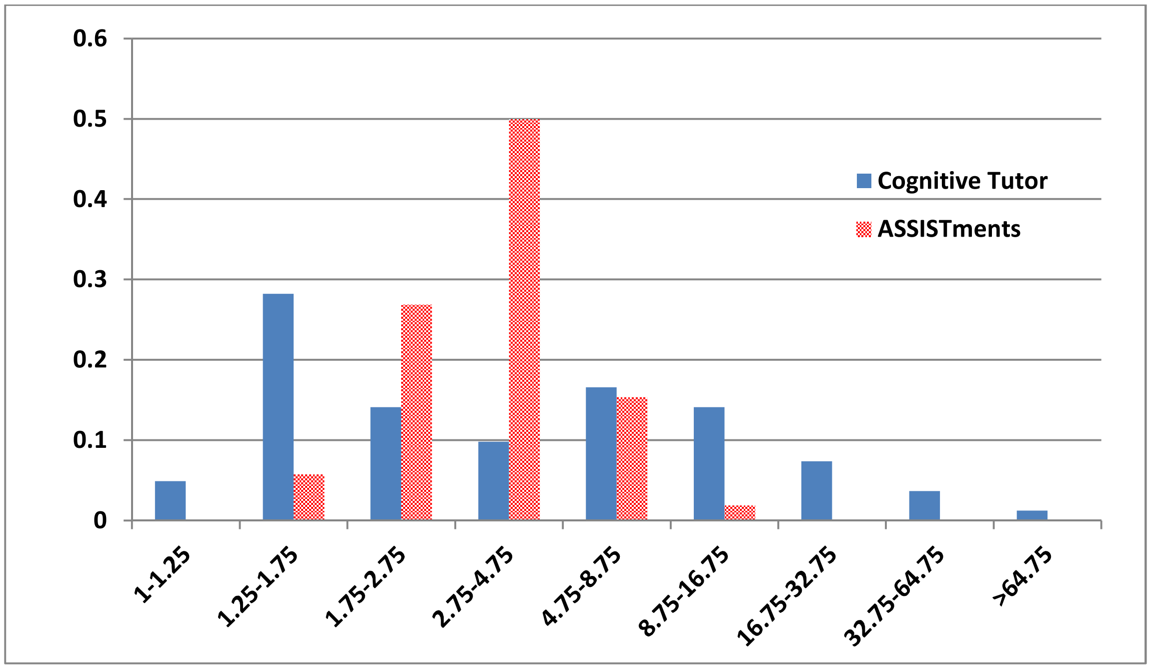


Fig. 5.2. Frequency of KC spikiness levels in the two tutors (Cognitive Tutor on the left, ASSISTments on the right). The x-axis is the range of spikiness (displayed with logarithmic scale) and the y-axis is the percent frequency of each bin.

5.2 Spikiness by Student

Within our data from the cognitive tutor, spikiness values range for students between {2.22, 21.81}, M=6.81, SD=3.09, considerably less spikiness (on the whole) than the differences in spikiness seen between KCs. For ASSISTments, we found that spikiness values range between {1.12, 15.423}, M=3.09, SD=1.49, which is a slightly larger range than was seen for skills in ASSISTments. Interestingly, the student spikiness ranges are much more similar between the Cognitive Tutor data and the ASSISTments data than the KC spikiness ranges are, suggesting

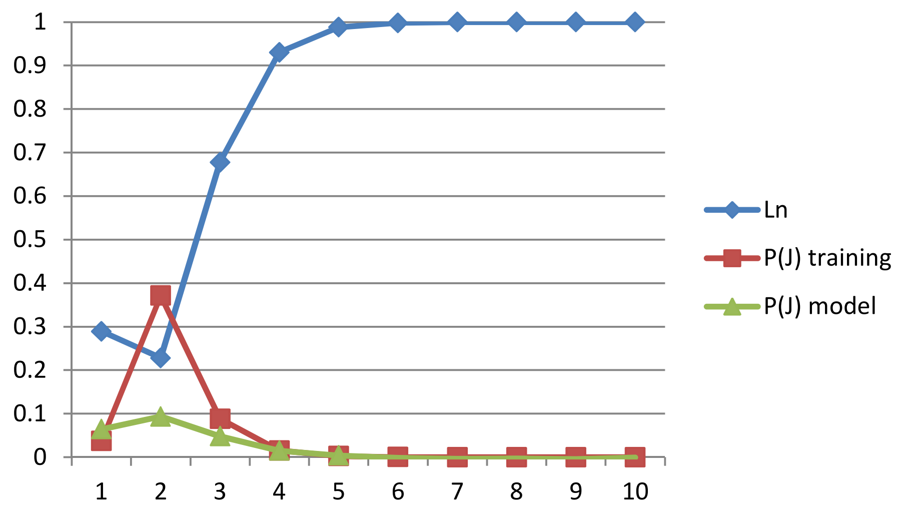


Fig 5.3. One student's performance on a single KC in the ASSISTment System. The x-axis denotes the number of opportunities to practice the KC.

that the tutors may have been more different from each other than the students who used them.

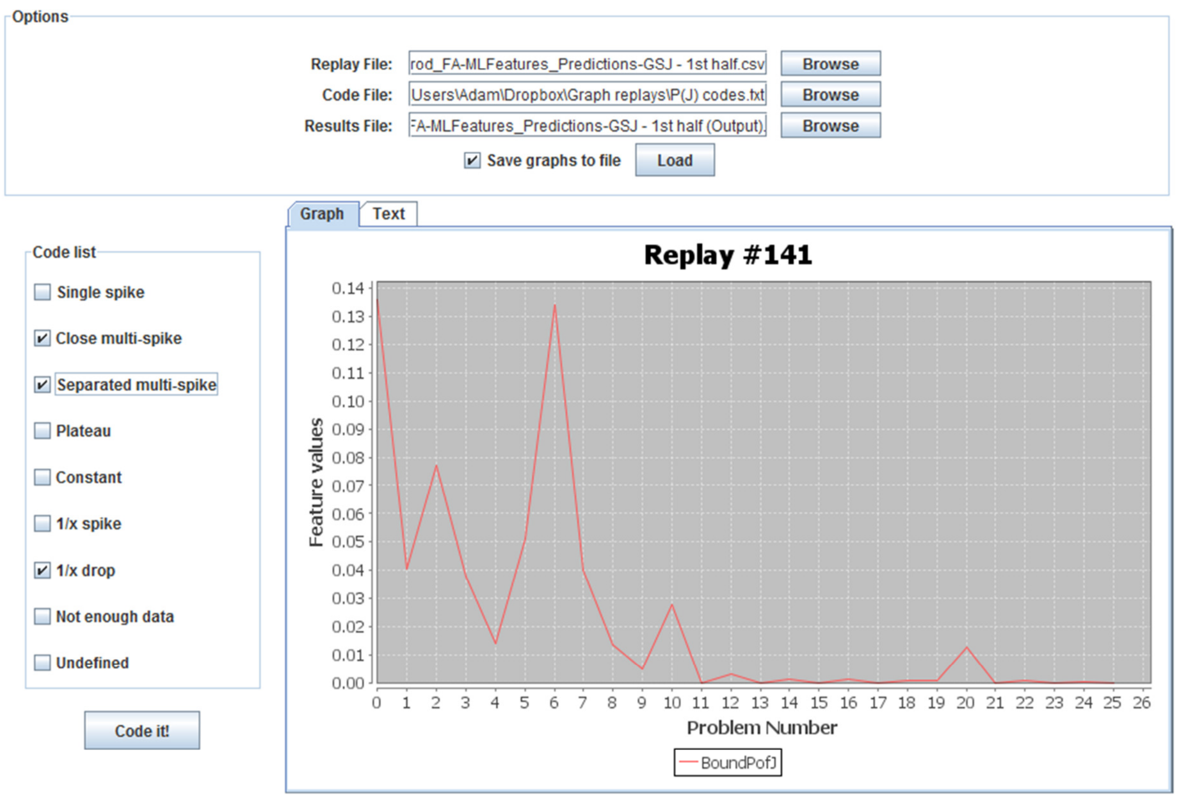
Interestingly, however, a student's spikiness is a good predictor of their final knowledge; the correlation between a student's average final P(L_n) and their average spikiness is a very high 0.71 in the Cognitive Tutor data, which is statistically significantly different than chance, F(1,228)=230.19, $p<0.0001$. In ASSISTments, the correlation is 0.50, which is also statistically significantly different than chance, F(1,4187)=1397.71, $p<0.0001$. These results suggest that learning spikes may be an early predictor of whether a student is going to achieve successful learning of the material in a tutor, as can be seen in figure 5.3. As can be seen, there is a spike for both the P($\boldsymbol{J}$) training labels and P($\boldsymbol{J}$) model at the second action, before the sequence of correct actions (3rd action, 4th action, 5th action) that lead to the higher P($\boldsymbol{L_n}$) values seen afterwards. It is worth noting that this spike is seen in both the P($\boldsymbol{J}$) training labels and P($\boldsymbol{J}$) model, although the model has lower peak values than the training labels (as discussed earlier) – point 2 for the model is still approximately double in magnitude as point 1 and point 3.

Besides potentially being an early predictor of eventually learning material, spikiness analysis can help to reveal possible ways an ITS can improve on its methodology of assessing mastery. From figure 5.3, we can see that the student continues to receive practice even after reaching mastery, according to Bayesian Knowledge Tracing. This limitation can be addressed by adding Bayesian Knowledge-Tracing into ASSISTments. However, while Bayesian Knowledge Tracing with well-fit parameter values can reduce over-practice (Cen, Koedinger, & Junker, 2007), current implementations of Bayesian Knowledge Tracing still often need a significant amount of data to conclude that a student has achieved mastery. For instance, note that the spike in P($\boldsymbol{J}$) in figure 5.3 occurred several actions before $\boldsymbol{L_n}$ reached a mastery level (and is seen in both the training set and the model, though in a more visually obvious fashion in the training set). Our hypothesis is that P($\boldsymbol{J}$) can be used to refine Bayesian Knowledge Tracing, so that the estimate of mastery goes up more, when spikes occur, than simply what would be indicated by the model's skill-level estimate of P($\boldsymbol{T}$). In doing so, it will be important to avoid using P($\boldsymbol{J}$) overly aggressively, creating a risk of under-practice. For one thing, though spikes predict eventual learning, this does not imply for certain that the learning is complete at the time of the spike (for instance, the student may have learned the knowledge component but need more practice in order to maintain long-term memory). In addition, even immediately after a spike, P($\boldsymbol{J}$) does not drop down to 0 (as can be seen in figure 5.3). Therefore, more work will be needed to figure out how much additional practice is needed, after a spike, for student learning to be maintained.

5.3 Graph Replays

While studying P(J) spikiness, we discovered several repeated graph shapes that intuitively seemed to model various kinds of learning, e.g. acquiring deep vs shallow knowledge. We also received feedback from other groups hypothesizing about the nature of what can be termed the "double spike", as seen in figure 5.1. One hypothesis was that the troughs between spikes representing time when a student is exhibiting off-task behavior; after achieving mastery, it is possible that the student still has not proven proficiency to Bayesian Knowledge Tracing and thus gets bored. Another possibility is that multiple spikes are seen for multi-faceted knowledge components (e.g. Pythagorean Theorem) and each spike represents an opportunity to learn these sub-KCs. Although at least one spike was shown to correlate to final knowledge, we did not have a method for analysing the difference between single and double spikes.

Past work has shown that text replays, a method for generating labels (Baker, Corbett, & Wagner, 2006) that can be used to train classifiers of student behaviour (Baker & Carvalho, 2008; Sao Pedro et. al., 2010), are a convenient method for rapidly labeling logged data. Since the double-spikiness is perhaps easier to visualize graphically, we invented a similar labeling method called graph replays that charts students' P(***J***) values for each skill. After marking each graph with defined features (e.g. single spike, double spike, wave) we can attempt to find correlations between shape and various forms of learning. In order to expedite the labeling process, we wrote a program called the GraphReplayer. The GraphReplayer is written in Java and accepts several input files that define labels that will be used as well as entire data sets marked with identifiers to notify the program what delimits between replays. The program presents the user with the target value, e.g. P(***J***), graphed over some interval, e.g. problems for a given assignment or knowledge component. The user selects from the label list the option that matches the graph that they see, and the GraphReplayer will store the result as well as a copy of the graph.


Options

Replay File: rod_FA-MLFeatures_Predictions-GSJ - 1st half.csv

Code File: Users\Adam\Dropbox\Graph replays\P(J) codes.txt

Results File: FA-MLFeatures_Predictions-GSJ - 1st half (Output).

[✓] Save graphs to file

Code list

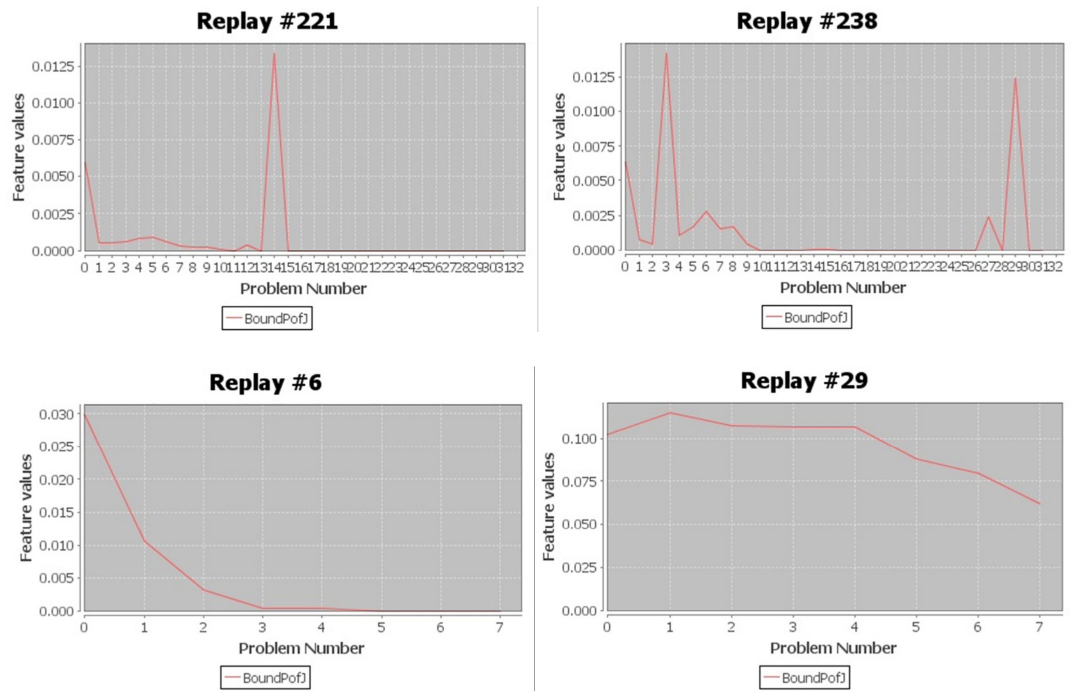
[ ] Single spike
[✓] Close multi-spike
[✓] Separated multi-spike
[ ] Plateau
[ ] Constant
[ ] 1/x spike
[✓] 1/x drop
[ ] Not enough data
[ ] Undefined

Fig. 5.4. Image of the GraphReplayer displaying a P(***J***) replay in tag mode.

For our purposes, we used a 2009 Genetics data set from Carnegie Learning that included 72 students working on 9 knowledge components (cf. Baker, Corbett, et. al., 2010). Associated with this data were various learning metrics, including both conceptual and problem solving pre-test/post-test scores, a transfer test, preparation for future learning test, and a problem solving retention test. The hypothesis behind this work was that we could first correlate between graph shape and learning metrics and also that this data set would allow us to create detectors of the important P(***J***) shapes.

5.3.1 Replays with labels

Our first attempt with graph replays was essentially identical to the text replay process; there are given labels and all graphs must fit one or another (or be marked as undefined or "not enough data"). The labels that we chose were decided on by our team simply by shared encounters with certain graph shapes during analyses from research described earlier in this paper. They are listed below with example graphs from the replayer in figure 5.5 and are mapped to label names given in table 5.1 below.



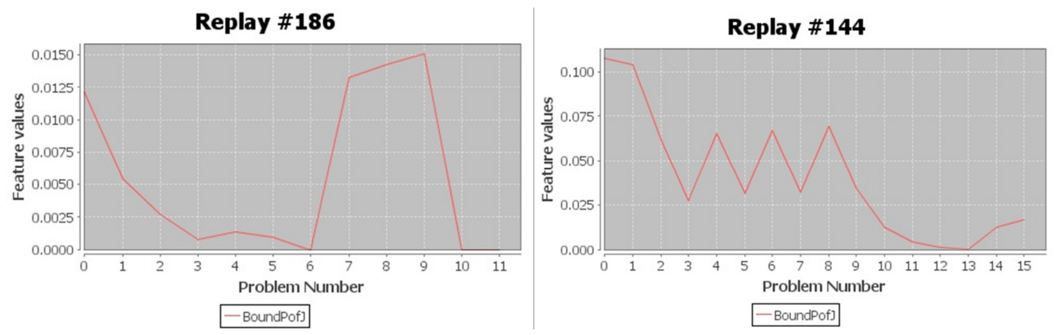


Fig. 5.5. Examples of P(***J***) graph replays matched to label names in table 5.1.

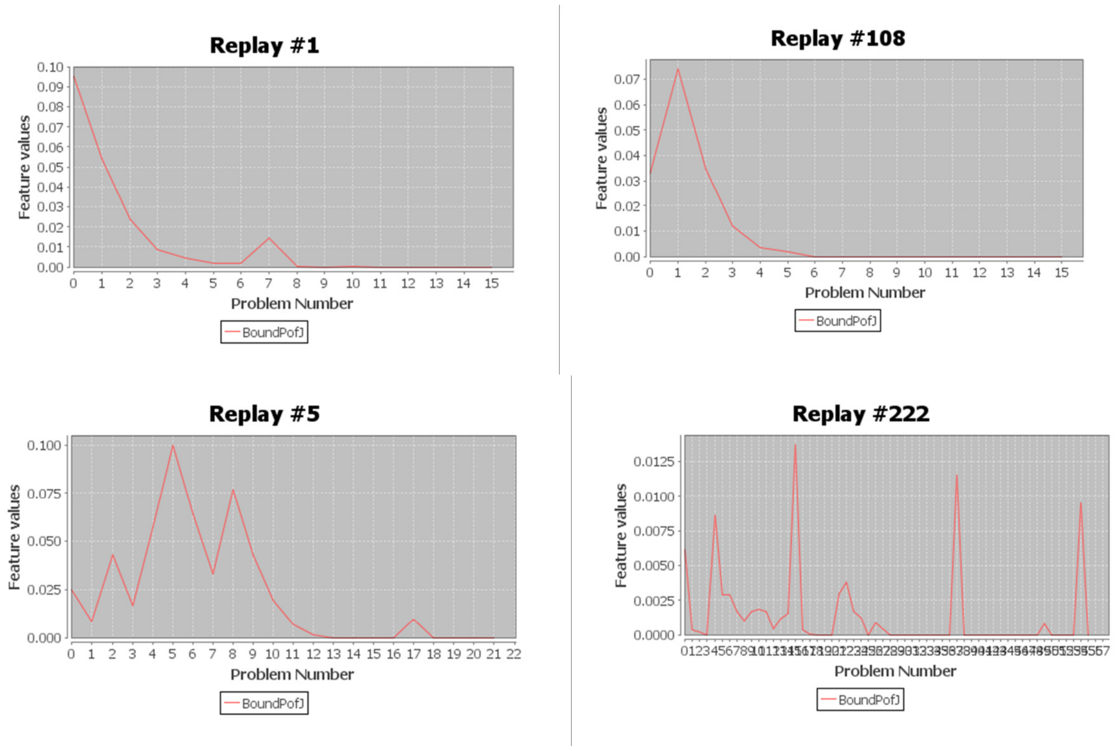
Table 5.2. Names of P(***J***) graph replays matched to example numbers in figure 5.6.

Replay Example Number	**Replay Label**
221	Single spike
238	Double spike
6	1/x
29	Constant
186	Plateau
144	Wave

The labeling process was completed by two members of our team. In order to verify that we were in sync, we took a sample file of the first 1000 rows from our data set, which was approximately 100 replays, each coded the sample, and calculated Cohen's kappa (1960). Our inter-rater agreement was 0.87, which is considered acceptable, so we split the entire file into two halves and labeled the rest of the replays. We checked to see if there was a correlation between the kinds of graphs a student produced from their work and the various metrics of learning described above, but unfortunately there were no statistically significant results. The highest value was that the "wave" label was inversely correlated (-0.327) with problem solving retention test scores. It is possible to theorize about the possible correlations we saw, but since they were so weak, it is not valid to do so.

5.3.2 Replays with tags

After finding no statistically significant correlations from the labeling method, we reconsidered our approach. Although we recorded a high inter-rater reliability, we did notice that some graphs shared attributes between several of the established labels that we had discussed. For instance, some graphs may have three or more separated spikes, which is somewhat of an ambiguous case between double spike and wave, and the situation only gets more convoluted the longer the replay. With that in mind, we adapted our method so that the GraphReplayer could generate a "tagging" mode where rather than users assigning labels to graphs, they would mark all relevant characteristics. Some labels translated directly into tags, such as the single spike, but other values were removed and new ones were added. The adapted list for this attempt is shown below in figure 5.6, mapped to the information given in table 5.2.



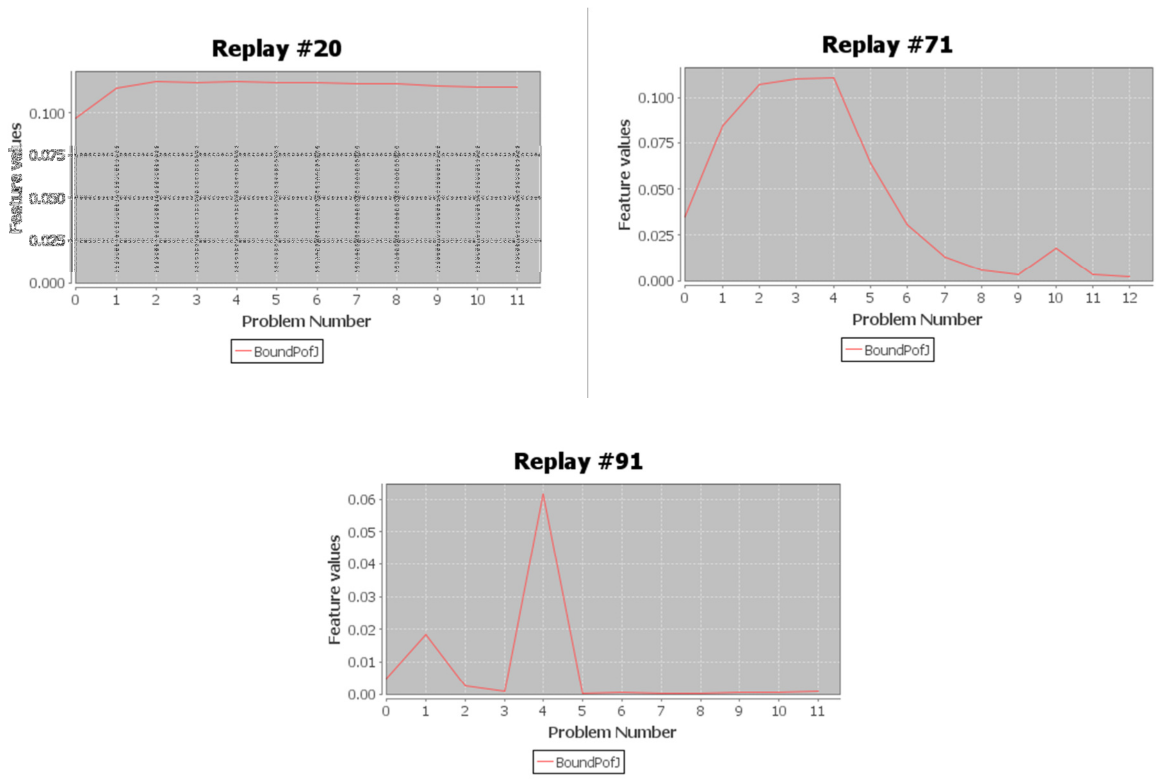


Fig. 5.6. Examples of P(***J***) graph replays matched to tagging names in table 5.2.

Table 5.2. Names of P(***J***) graph replays matched to example numbers in figure 5.6.

Replay Example Number	**Replay Tag**
1	1/x drop
108	1/x spike
5	Close multi-spike
222	Separated multi-spike
20	Constant
71	Plateau
91	Single spike

In this iteration, we achieved a kappa value of 0.71 averaged across tags, but again did not find particularly strong correlations between any tag and any metric of learning. For the process of checking correlations, we tried both concatenating all tags to graphs and counting all tags as individuals, i.e. any graph that involved tag t is multi-counted for all relevant tags. The former approach yielded 29 unique tagging groups, with the most significant value being that "plateau" is inversely correlated (-0.409) with preparation for future learning. The latter showed that "1/x spike" is correlated (0.338) to problem solving retention test scores. It is possible to speculate that a plateau represents repeated missed learning opportunities and that a 1/x spike is indicative of a student who enters the problem set already proficient, but because these correlations were again not statistically significant, we will not hypothesize further about these results. It is possible these results were due to a small sample size, so working with a larger data set is planned for future work.

Although we ultimately did not find statistically significant correlations between graph shape and learning metrics for the data set that we chose, there is a substantial amount of future work to be done. As discussed earlier, only some knowledge components seem to systematically produce spiky behaviour, whereas others tend to have gradual learning rates; our Genetics data set only involved 9 knowledge components and it is conceivable that this sample is simply not conducive to spikiness replays. We found that the "1/x drop" tag and "1/x" label accounted for more than 50% of all graphs, which suggests that both our tags and labels are not at fine enough of a grain, or we have some sort of biased data set. The 1/x shape could represent several phenomena, possibly indicating that students are either immediately mastering the given knowledge component or entering the problem set already proficient. Applying the graph replay process to additional data sets, as well as attempting different labeling and tagging patterns, is a plan for future research.

6 Piloting P(J)

After demonstrating the validity of our model across multiple Intelligent Tutoring Systems, our next goal is to explore the utility of P(***J***). If P(***J***) can be used as an early detector of eventual success or conceptual understanding, then it is possible that tutors can provide students with a better optimized amount of time and problems on any given knowledge component. The goal is to present enough practice that students learn to the best of their ability, but no more than is necessary so that they maximize the use of their time. Although we do not expect to see P(***J***) result in higher learning gains as compared to any other method of cognitive mastery learning, we are curious to see if the moment-to-moment detection of learning can result in more efficient use of student time.

As described in section 2, both ASSISTments and the Cognitive Tutor employ different methods of determining proficiency for a given student on a given knowledge component. While the Cognitive Tutor awards the label of mastery for achieving an $\boldsymbol{L_n}$ of 0.95 or greater, ASSISTments asks the student to get 3 questions in a row correct without an incorrect response or help request in between. There has not yet been a study conducted to compare 3-in-a-row to Bayesian Knowledge Tracing (BKT) and so it is our intention to analyze all three methods together as we explore how P(***J***) can be utilized.

In order to involve P(***J***) in a study that actually affects students at runtime rather than doing a post-hoc analysis on data, we needed to add the ability to an ITS to use the P(***J***) value to intervene and recognize students as having mastered a given KC. We programmed new problem set types into ASSISTments so that we can present the same kind of content to the same audience across our three styles of mastery learning. The 3-in-a-row style behaves as usual, and Bayesian Knowledge Tracing functions identically to the Cognitive Tutor. For the new P(***J***) condition, mastery is awarded if the system sees the student spike from one question and then answer the next problem correct on the first attempt. Due to the nature of P(***J***), it is possible that a spike occurs from an incorrect response or help request. This allows for a "wrong-correct" sequence to result in mastery, which is rare in Bayesian Knowledge Tracing and impossible with 3-in-a-row. For each knowledge component in the ASSISTments system, we calculated the average of maximum P(***J***) values seen for students in our prior post-hoc analyses. Whenever a student's P(***J***) value hits that average at runtime, they are considered to have spiked and that student will potentially trigger the test out condition. Because this setup is prone to only intervening on the upper half of spiky students, we took the distribution of maximum P(***J***) values and altered the threshold for spikes to allow for roughly an additional 25% more students to be considered spiking. This is not a proven method for how our model can be used in real time, but our first step here was to run a small pilot to assess the tractability of P(***J***) mastery, so we will detail that experiment and its preliminary results in the following subsections.

6.1 Experiment Design

Our pilot study was conducted with 54 students from 3 middle schools close to the Worcester Polytechnic Institute. All participants were administered a balanced pre-test and post-test, where half were given Assessment 1 (See Appendix A1) as a pre-test and Assessment 2 (See Appendix A2) as a post-test, while the other half were presented with the opposite setup. The contents covered six different knowledge components: Order of Operations, Division of Positive Decimals, Square Roots, Percent Of, Proportions, and Probability Compound, which map to M.8.8-applying-properties-of-operators, N/A, N.2.8-using-common-irrational-numbers, N.10.8-estimating-and-computing-various-numbers, N.3.8-ratios-and-proportions, and D.4.8-understanding-concept-of-probabilities from MCAS strands, respectively. The pre-tests and post-tests were all 18 questions long with 3 questions for each KC.

Students were then assigned six problem sets, one per KC from the pre-test and post-test. The condition problem sets were based off of variabilized templates, which is a common practice in ASSISTments for mastery learning. Essentially, only a handful of types of problems are written and then the system generates questions that are structurally identical but use different numbers. An example of questions given to students derived from templates is available in Appendix A3.

All students were exposed to all interventions, so two of the six KCs were 3-in-a-row, two were Bayesian Knowledge Tracing, and two were P(J). Students were organized into columns based on their overall performance in ASSISTments from approximately a year's worth of usage as part of their regular curriculum. The condition assignments were evenly given in a Latin Squared format, as shown in table 6.1.

Table 6.1. An example of pre-test, condition, and post-test distribution from the pilot study.

Student ID	**Pre-test**	**Skill 1-2**	**Skill 3-4**	**Skill 5-6**	**Post-test**
92704	Assessment 1	3-in-a-row	BKT	P(***J***)	Assessment 2
86456	Assessment 2	P(***J***)	3-in-a-row	BKT	Assessment 1
83099	Assessment 1	BKT	P(***J***)	3-in-a-row	Assessment 2
92788	Assessment 2	3-in-a-row	BKT	P(***J***)	Assessment 1
83104	Assessment 1	P(***J***)	3-in-a-row	BKT	Assessment 2
82555	Assessment 2	BKT	P(***J***)	3-in-a-row	Assessment 1

Students were given their work as homework assignments and were thus allowed to work at their own pace. ASSISTments for homework is a common part of the routine for all participants involved, so this was not unusual.

6.2 Results

Given that the pilot study was conducted on such a small sample size, we did not expect to have sufficient statistical power. There appeared to be a very slight gain from total pre-test score to total post-test score, M=0.047, SD=0.171. On a condition by KC basis, there seemed to be an equally slight gain, with M=0.00058, SD=0.323 for 3-in-a-row, M=0.046, SD=0.37 for BKT, and M=0.061, SD=0.36 for P(***J***). There appeared to be a slight difference between conditions for the number of problems seen per KC, with M=4.16, SD=2.23 for 3-in-a-row, M=5.17, SD=3.91 for BKT, and M=3.79, SD=3.31 for P(***J***). Similarly for the amount of time spent for KCs between conditions there seemed to be a slight difference, with M=154.34 seconds, SD=192.29 seconds for 3-in-a-row, M=307.63 seconds, SD=562.48 seconds for BKT, and M=195.76 seconds, SD=250.75 seconds for P(***J***).

Although we cannot yet make confident assertions about the utility of P(***J***) or the way it compares to 3-in-a-row or Bayesian Knowledge Tracing, the pilot study provided an important opportunity to learn lessons in preparation for a full study, which will be conducted in the near future. For that experiment, we will properly run statistical analyses so that we can confidently observe and analyze differences that occur between conditions, but that was not the goal of this target study. This first step did help us with some simple but important tasks such as finding bugs in the new implementations within ASSISTments and better understanding the kind of instructions that will ensure students properly complete all experiment work in the proper order. We know now that our method of detecting P(***J***) spikes is tractable. Additionally, and perhaps most importantly, we've learned that our future experiment should contain knowledge components that are less well known to the students. We chose KCs that we found to be spiky from prior research, but even spiky skills do not have much room for knowledge gains if students are already at a mastery level for the KCs. The average pre-test score was a 72% and the average post-test score was a 77%, so it would be beneficial to the study to use knowledge components that are less familiar to our sample group.

7 Discussion and Conclusions

In this paper, we have presented models of P(***J***), the probability that a student learned a specific KC on a specific opportunity to practice and learn that KC. This paper does so in the context of two intelligent tutoring systems, the Cognitive Tutor and ASSISTments. Though this model builds off of past attempts to contextualize student modeling (e.g. Baker, Corbett, & Aleven, 2008) and to study the impact of different events on learning (e.g. Beck et al., 2008; Pardos & Heffernan, 2009), this model is distinct from prior models of student learning, focusing on assessing the likelihood of learning on individual problem steps. We show that the model achieves correlation in the range of 0.4-0.45 to the labels of this construct. In addition, we also find that the model's assessments of P(***J***) can be used to distill a secondary measure, the "spikiness" of learning, defined as the maximum momentary learning, divided by the average momentary learning. We find that a student's spikiness is a good predictor of their final knowledge in both Cognitive Tutors and ASSISTments. This finding suggests that P(***J***) represents a genuine measure of learning which can be studied further to shed light on the process of learning within intelligent tutoring systems.

7.1 Potential Futures Uses of the Model

The P(***J***) model can be used in at least two ways going forward. First, it can be used as an analytical tool and as part of other models, much as models predicting how much a student has learned have been used (e.g. Aleven et al., 2006; Baker, 2007; Baker, Corbett, Roll, & Koedinger, 2008; Muldner et al., 2010). Within this paper, we present an analysis using P(***J***) to infer that KCs have greater variance in spikiness than students. Studying which aspects of KCs predicts spikiness may be a valuable tool for further research into what types of KCs are learned gradually or through "eureka" experiences. In addition, given the correlation between spikiness and final knowledge, models of P(***J***) are likely to prove useful for student knowledge modeling, as contextual guess and slip have been (e.g. Baker, Corbett, & Aleven, 2008), and in the long term may lead to more effective adaptation by Intelligent Tutoring Systems. We have frequently observed multiple spikes within one student's performance, which are currently an unexplained phenomenon and a topic of future research. It is possible that a single spike is a learning moment, and following questions are potentially over-practice, which has been shown to lead to gaming the system (Baker et al., 2008). If that is the case, the valley following the first spike could be explained by gaming, and the second (or third, fourth, etc.) spike is an indication that the student has become engaged again. In general, exploring the relationship between the dynamics of learning over time, as expressed by P(***J***) models, and other constructs in learning and engagement, has the potential to be a fruitful area of future work.

Given that we can potentially identify moments in which students learn, it may be possible to improve the adaptivity – and in particular the speed of adaptivity – of intelligent tutoring systems using these models. In particular P(***J***) models may be able to support work to use reinforcement learning to induce tutorial strategies (cf. Chi, VanLehn, & Litman, 2010). Work along these lines will be supported still further, if it becomes possible to not just identify learning spikes, but to identify their antecedents. It also may be possible to improve adaptivity just through the step of improving Bayesian Knowledge Tracing models themselves, using P(***J***). Recent work has suggested that contextualization of model parameters such as guess, slip, and initial knowledge can improve prediction within a tutoring system (Baker, Corbett, & Aleven, 2008; Pardos & Heffernan, 2010), although there is also evidence that contextual guess and slip models may over-fit to within-tutor performance (Baker et al., 2010). Studying ways to integrate multiple forms of contextualization, including P(***J***), to improve knowledge prediction both within the tutor and on later tests of knowledge may therefore be an important area of future work.

In conclusion, this paper has introduced models of the learning that occurs moment-by-moment, and one way to utilize these models to study student learning. There appears to be potential for improving these models' precision and using them in a variety of ways to study learning and improve adaptation by learning software. We have shown that the P(***J***) model is implementable for use at runtime in an ITS and that a study to compare that utilization of P(***J***) as an intervention to other methods of cognitive mastery learning will make for a tractable experiment. Exploring the amount of success that method may have, as well as studying exactly which potential uses of P(***J***) are the most productive, are topics for research in the very near future.

Appendix A

Assessment Tests

A. 1 Assessment Test 1

A. 2 Assessment Test 2

A. 3 Example Questions

Bibliography

Aleven, V., McLaren, B., Roll, I., Koedinger, K. (2006). Toward meta-cognitive tutoring: A model of help seeking with a Cognitive Tutor. *International Journal of Artificial Intelligence and Education*, 16, 101-128.

Anderson, J.R., Lebiere, C. (2006) *The Atomic Components of Thought.* Mahwah, NJ: Lawrence Erlbaum Associates.

Baker, R.S.J.d. (2007). Modeling and Understanding Students' Off-Task Behavior in Intelligent Tutoring Systems. *In: Proceedings of ACM CHI: Computer-Human Interaction*, 1059-1068.

Baker, R.S.J.d., Corbett, A.T., Aleven, V. (2008). More Accurate Student Modeling Through Contextual Estimation of Slip and Guess Probabilities in Bayesian Knowledge Tracing. *In: Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, 406-415.

Baker, R.S.J.d., Corbett, A.T., Roll, I., Koedinger, K.R. (2008). Developing a Generalizable Detector of When Students Game the System. *User Modeling and User-Adapted Interaction*, 18 (3), 287-314.

Baker, R.S.J.d., Corbett, A.T., Gowda, S.M., Wagner, A.Z., MacLaren, B.M., Kauffman, L.R., Mitchell, A.P., Giguere, S. (2010) Contextual Slip and Prediction of Student Performance After Use of an Intelligent Tutor. *Proceedings of the 18th Annual Conference on User Modeling, Adaptation, and Personalization*, 52-63.

Beck, J.E., Chang, K-m., Mostow, J., Corbett, A. (2008). Does Help Help? Introducing the Bayesian Evaluation and Assessment Methodology. *In: Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, 383-394.

Beck, J.E., Mostow, J. (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 the 9th International Conference on Intelligent Tutoring Systems*, 353-362.

Bowden, E.M., Jung-Beeman, M., Fleck, J., Kounios, J. (2005) New approaches to demystifying insight. *TRENDS in Cognitive Science, 9* (7), 322-328.

Bransford, J. D. & Schwartz, D. L. (1999). Rethinking transfer: A simple proposal with multiple implications. *In A. Iran-Nejad and P. D. Pearson (Eds.), Review of Research in Education*, 24, 61-100.

Burton, R.R. (1982) Diagnosing bugs in a simple procedural skill. In Sleeman, D.H. and Brown, J.S. (Eds.) *Intelligent Tutoring Systems.* London, UK: Academic Press.

Cen, H., Koedinger, K.R., Junker, B. (2007). Is Over Practice Necessary? Improving Learning Efficiency with the Cognitive Tutor. *In: Proceedings of the 13th International Conference on Artificial Intelligence and Education.*

Cetintas, S., Si, L., Xin, Y.P., Hord, C., Zhang, D. (2009). Learning to Identify Students' Off-Task Behavior in Intelligent Tutoring Systems. *In: Proceedings of the 14th International Conference on Artificial Intelligence in Education*, 701-703.

Chi, M., VanLehn, K., Litman, D. (2010). Do Micro-Level Tutorial Decisions Matter: Applying Reinforcement Learning to Induce Pedagogical Tutorial Tactics. *The 10th International Conference on Intelligent Tutoring Systems*, 224-234.

Cocea, M., Hershkovitz, A., Baker, R.S.J.d. (2009). The Impact of Off-task and Gaming Behaviors on Learning: Immediate or Aggregate? *In: Proceedings of the 14th International Conference on Artificial Intelligence in Education*, 507-514.

Corbett, A.T., Anderson, J.R.: Knowledge Tracing (1995). Modeling the Acquisition of Procedural Knowledge. *User Modeling and User-Adapted Interaction*, 4, 253-278.Duncker, K. (1945) On Problem Solving. *Psychological Monographs, 58,* 270.

Feng, M., Heffernan, N.T., Koedinger, K.R. (2006) Predicting State Test Scores Better with Intelligent Tutoring Systems: Developing Metrics to Measure Assistance Required. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems,* 31-40.

Ferguson, G.A. (1971). *Statistical Analysis in Psychology and Education*. New York: McGraw-Hill.

Goldstein, I.J. (1979) The genetic graph: a representation for the evolution of procedural knowledge. *International Journal of Man-Machine Studies, 11* (1), 51-77.

Gong, Y, Beck, J. E., Heffernan, N. T. (2010). Comparing Knowledge Tracing and Performance Factor Analysis by Using Multiple Model Fitting. *The 10th International Conference on Intelligent Tutoring Systems*.

Hall, M.A. (2000) Correlation-Based Feature Selection for Discrete and Numeric Class Machine Learning. *Proceedings of the 17th International Conference on Machine Learning,* 359-366.

Heathcote, A., Brown, S., Mewhort, D,J.K. (2000) The Power Law Repealed: The Case for an Exponential Law of Practice. *Psychonomic Bulletin and Review, 7,* 185-207.

Johns, J., Woolf, N. (2006) A dynamic mixture model to detect student motivation and proficiency. Paper presented at the *21st National Conference on Artificial Intelligence (AAAI-2006).* Boston, MA, USA.

Langley, P., Ohlsson, S. (1984) Automated Cognitive Modeling. *Proceedings of the National Conference on Artificial Intelligence,* 193-197.

Koedinger, K.R. (2002). Toward evidence for instructional principles: Examples from Cognitive Tutor Math 6. *In: Proceedings of PME-NA XXXIII (the North American Chapter of the International Group for the Psychology of Mathematics Education).*

Koedinger, K.R., Baker, R.S.J.d., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J. (2010) A Data Repository for the EDM commuity: The PSLC DataShop. In Romero, C., Ventura, S., Pechenizkiy, M., Baker, R.S.J.d. (Eds.) *Handbook of Educational Data Mining*. Boca Raton, FL: CRC Press, pp. 43-56.

Koedinger, K. R., & Corbett, A. T. (2006). Cognitive tutors: Technology bringing learning science to the classroom. *In K. Sawyer (Ed.), The Cambridge Handbook of the Learning Sciences.* Cambridge, MA: Cambridge University Press.

Koedinger, K. R., McLaughlin, E. A., & Heffernan, N. T. (in press). A Quasi-Experimental Evaluation of an On-line Formative Assessment and Tutoring System. To appear in *Journal of Educational Computing Research.*

Kounios, J., Fleck, J.I., Green, D.L., Payne, L., Stevenson, J.L., Bowden, E.M., Jung-Beeman, M. (2008) The Origins of Insight in Resting-State Brain Activity. *Neuropsychologica, 46* (1), 281-291.

Lindstrom, P., Gulz, A. (2008). Catching Eureka on the Fly. *In: Proceedings of the AAAI 2008 Spring Symposium.*

Martin, J., VanLehn, K. (1995). Student Assessment Using Bayesian Nets. *International Journal of Human-Computer Studies*, 42, 575-591.

Mendicino, M., Razzaq, L. & Heffernan, N. T. (2009). *Comparison of Traditional Homework with Computer Supported Homework: Improving Learning from Homework Using Intelligent Tutoring Systems*. Journal of Research on Technology in Education (JRTE), 41(3), 331-358.

Metcalfe, J., Wiebe, D. (1987) Intuition in Insight and Noninsight Problem Solving, *Memory and Cognition, 15* (3), 238-246.

Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T. (2006). YALE: Rapid Prototyping for Complex Data Mining Tasks. *In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD 2006), 935-940.

Newell, A., Rosenbloom, P.S. (1981) Mechanisms of Skill Acquisition and the Law of Practice. In J.R. Anderson (Ed.) *Cognitive Skills and their Acquisition,* 1-55. Hillsdale, NJ: Lawrence Erlbaum Associates.

Pavlik, P.I., Anderson, J.R. (2008). Using a Model to Compute the Optimal Schedule of Practice. *Journal of Experimental Psychology: Applied*, 14 (2), 101-117.

Pardos, Z., Beck, J.E., Ruiz, C., Heffernan, N.T. (2008). The Composition Effect: Conjunctive or Compensatory? An Analysis of Multi-Skill Math Questions in ITS. *In: Proceedings of the 1st International Conference on Educational Data Mining*, 147-156.

Pardos, Z. A., Dailey, M., Heffernan, N.T. (2010). Learning What Works in ITS from Non-traditional Randomized Controlled Trial Data. *The 10th International Conference on Intelligent Tutoring Systems*, 41-50.

Pardos, Z., Heffernan, N. (2009). Determining the Significance of Item Order in Randomized Problem Sets. *In: Proceedings of the 1st International Conference on Educational Data Mining*, 111-120.

Pardos, Z. A., Heffernan, N. T. (2010). Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing. *In Proceedings of the 18th International Conference on User Modeling, Adaptation and Personalization*. 255-266.

Razzaq, L., Feng, M., Nuzzo-Jones, G., Heffernan, N.T., Koedinger, K. R., Junker, B., Ritter, S., Knight, A., Aniszczyk, C., Choksey, S., Livak, T., Mercado, E., Turner, T.E., Upalekar. R, Walonoski, J.A., Macasek. M.A. & Rasmussen, K.P. (2005). The Assistment project: Blending assessment and assisting. In C.K. Looi, G. McCalla, B. Bredeweg, & J. Breuker (Eds.) *Proceedings of the 12th Artificial Intelligence in Education*, Amsterdam: ISO Press. pp. 555-562.

Rosenthal, R., Rosnow, R.L. (1991). *Essentials of Behavioral Research: Methods and Data Analysis.* Boston, MA: McGraw-Hill.

Shih, B., Koedinger, K., Scheines, R. (2008). *A Response Time Model for Bottom-Out Hints as Worked Examples*. In: Proceedings of the 1st International Conference on Educational Data Mining, 117-126.

Shute, V.J. (1995). SMART: Student modeling approach for responsive tutoring. *User Modeling and User-Adapted Interaction*, 5 (1), 1-44.

Sleeman, D.H. (1982) Assessing aspects of competence in basic algebra. In Sleeman, D.H., Brown, J.S. (Eds.) *Intelligent Tutoring Systems.* London, UK: Academic Press.

Sleeman, D.H., Langley, P., Mitchell, T.M. (1982) Learning from solution paths: an approach to the credit assignment problem. *AI Magazine, 3* (1), 48-52.

VanLehn, K. (1990) *Mind Bugs: The Origins of Procedural Misconceptions*. Cambridge, MA: MIT Press.

VanLehn, K., Siler, S., Murray, C., et. al. (2003). Why Do Only Some Events Cause Learning During Human Tutoring. *Cognition and Instruction*, 21 (3), 209-249.

Walonoski, J.A., Heffernan, N.T. (2006). Detection and analysis of off-task gaming behavior in intelligent tutoring systems. *In: Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, 382-391.

Witten, I.H., Frank, E. (2005) *Data Mining: Practical Machine Learning Tools and Techniques.* San Francisco, CA: Morgan Kaufmann.