Predictive Models of Student Learning

Zachary Alexander Pardos
Worcester Polytechnic Institute

Follow this and additional works at: https://digitalcommons.wpi.edu/etd-dissertations

Repository Citation

This dissertation is brought to you for free and open access by Digital WPI. It has been accepted for inclusion in Doctoral Dissertations (All Dissertations, All Years) by an authorized administrator of Digital WPI. For more information, please contact wpi-etd@wpi.edu.
Predictive Models of Student Learning

By

Zachary A. Pardos

A Dissertation

Submitted to the Faculty

Of the

WORCESTER POLYTECHNIC INSTITUTE

In Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

In

Computer Science

by

___________________________

April 2012

APPROVED:

_______________________________
Dr. Neil T. Heffernan
Advisor - WPI

_______________________________
Dr. Ryan S.J.D. Baker
Committee Member - WPI

_______________________________
Dr. Gabor Sarkozy
Committee Member - WPI

_______________________________
Dr. Kenneth Koedinger
External Committee Member - CMU
Abstract:

In this dissertation, several approaches I have taken to build upon the student learning model are described. There are two focuses of this dissertation. The first focus is on improving the accuracy with which future student knowledge and performance can be predicted by individualizing the model to each student. The second focus is to predict how different educational content and tutorial strategies will influence student learning. The two focuses are complimentary but are approached from slightly different directions. I have found that Bayesian Networks, based on belief propagation, are strong at achieving the goals of both focuses. In prediction, they excel at capturing the temporal nature of data produced where student knowledge is changing over time. This concept of state change over time is very difficult to capture with classical machine learning approaches. Interpretability is also hard to come by with classical machine learning approaches; however, it is one of the strengths of Bayesian models and aids in studying the direct influence of various factors on learning. The domain in which these models are being studied is the domain of computer tutoring systems, software which typically uses artificial intelligence to enhance computer based tutorial instruction. These systems are growing in relevance. At their best they have been shown to achieve the same educational gain as one on one human interaction. They have also received the attention of White House, which mentioned a tutor called ASSISTments in its National Educational Technology Plan. With the fast paced adoption of computer tutoring systems it is important to learn how to improve the educational effectiveness of these systems by making sense of the data that is being generated from them. The studies in this proposal use data from these educational systems which primarily teach topics of Geometry and Algebra but can be applied to any domain with clearly defined sub-skills and dichotomous student response data. One of the intended impacts of this work is for these knowledge modeling contributions to facilitate the move towards computer adaptive learning in much the same way that Item Response Theory models facilitated the move towards computer adaptive testing.
## Table of Contents

Chapter 1: Introduction ................................................................. 4-5

Chapter 2: Modeling Individualization in the Student Model ................. 5-15

Chapter 3: Evaluating the Identifiability of the Model Parameters ........... 15-23

Chapter 4: Fully Individualized Student Parameter Model and Random Forests 23-38

Chapter 5: Individualizing Parameters at the Content Level to Evaluate Individual Item Influences on Learning ................................................................. 38-45

Chapter 6: Individualizing Parameters at the Content Level to Evaluate Item Ordering Influences on Learning ................................................................. 45-57

Chapter 7: Using Content Individualized Parameters to Evaluate the Effectiveness of Different Types of Tutoring Interventions ......................... 57-70

Chapter 8: The Predictive Power of Adding Difficulty and Discrimination to the Item Individualization Model ................................................................. 71-80

Chapter 9: Summary of the Student Modeling and Tutoring Modeling Approaches. ................................................................. 80-89

Future Work ................................................................................. 89-91

Appendices. .................................................................................. 92-99

References ................................................................................. 99-103
Chapter 1: Introduction

In this dissertation, several approaches I have taken to model student learning using Bayesian Networks are described. There are two focuses of this dissertation. The first focus is on improving the accuracy with which future student performance can be predicted. The second focus is to predict how different educational content and tutorial strategies will influence learning. The two focuses are complimentary but are approached from slightly different directions. I have found that Bayesian Networks are strong at achieving the goals of both focuses. In prediction, they excel at capturing the temporal nature of data produced where student knowledge is changing over time. This concept of state change over time is very difficult to capture with classical machine learning approaches. Interpretability is also hard to come by with classical machine learning approaches; however, it is one of the strengths of causal models and aids in studying the direct influence of various factors on learning. The domain in which these models are being studied is the domain of computer tutoring systems, software which often uses artificial intelligence to enhance computer based tutorial instruction. These systems are growing in relevance. At their best they have been shown to achieve the same educational gain as one on one human tutoring (Koedinger et al., 1997). They have also received the attention of White House, which mentioned a tutoring platform named ASSISTments in its National Educational Technology Plan (Department of Education, 2010). With the fast paced adoption of tutoring systems it is important to learn how to improve the educational effectiveness of these systems by making sense of the data that is being generated from them. The studies in this proposal use data from these educational systems which primarily teach topics of Geometry and Algebra but can be applied to any domain with clearly defined sub-skills and dichotomous student response data.

This proposal is organized into nine chapters. Related work is referenced within each chapter and thus there is no separate chapter dedicated to related work. Increasing model prediction and assessment accuracy via individualized student parameters is one contribution of this work and is addressed in chapters 2-4 which describe general additions to the knowledge tracing model that produced favorable results. One such addition was the modeling of individual student attributes, such as individual prior knowledge and individual speed of learning. Introducing models which diagnose the effectiveness of the tutor and its content is the second contribution of this work. This area of study is described in chapters 5 through 9. The concepts of predictive ability and diagnostic tutor information are combined in the study described in chapter 7. Chapter 8 focuses on modeling item difficulty and discrimination within the Bayesian framework and its parallels to IRT. Algorithms and experiments are proposed to optimize assessment accuracy and student learning separately. Blending of assistance and assessment goals will be left as an open research question for the field. Chapter 9 serves as a summary of the student modeling approaches as well as the tutor modeling approaches and how both fit into the Bayesian framework designed around the Knowledge Tracing model.

There are a wide variety of models available for predicting student performance. The classical Item Response Theory (IRT) model has been in use for decades (Spada & McGaw, 1985). Derivatives of IRT such as the Linear Logistic Test Model (LLTM) (Scheiblechner, 1972) and modern successors in the intelligent tutoring system literature, such as Additive Factors Model (Cen, Koedinger, Junker, 2008), Conjunctive Factors Model (Cen, Koedinger, Junker, 2008) and Performance Factors Analysis (Pavlik, Cen, Koedinger, 2009), have explored modifications to the classical IRT model. However, none of these models track student knowledge or ability over time. Instead, these models capture a stationary ability parameter per student. This paradigm of treating student ability as a trait that is not changing is customary when evaluating students for testing purposes where assessment is paramount and minimal learning is assumed to be taking place during testing. This was the primary purpose of IRT as conceived by the psychometrics community. It continues to be how IRT is used as evident by its role as the
underlying model used for scoring in GRE (Graduate Record Examinations) testing, the standard test required for application to most graduate schools in the United States. The landscape of computer based tutoring systems poses different challenges from that of computer based testing systems and thus a different class of model is required. In Intelligent Tutoring Systems, Knowledge Tracing is the current state of the art in that class. The primary difference between testing and tutoring environments is the assumption of change in knowledge. Whereas with testing, knowledge can be modeled as a trait, in tutoring it is more appropriately modeled as a changing state. Bayesian Networks, based on belief propagation (Pearl, 2000), are particularly well suited for modeling and inferring changes in a latent state, such as knowledge, over time, given a set of evidence. In the case of tutoring systems, the evidence is customarily a student’s past history of correct and incorrect responses to problems. The Knowledge Tracing model, while being the de facto standard, was relatively immature compared to IRT style models. It lacked modeling of features important in IRT such as individual student ability traits and individual item traits, such as difficulty. This dissertation adds these important elements to the Knowledge Tracing model which both increases the model’s accuracy as well as functionality. The additions allow for characteristics of student learning to be assessed as well as learning effectiveness of tutor content. One of the intended impacts of this work is for these knowledge modeling advancements to accelerate the move to computer adaptive learning environments in much the same way that IRT facilitated the move to computer adaptive testing.

Chapter 2: Modeling Individualization in the Student Model

The field of intelligent tutoring systems has been using the well-known knowledge tracing model, popularized by Corbett and Anderson (1995), to track student knowledge for over a decade. Surprisingly, models currently in use do not allow for individual learning rates nor individualized estimates of student initial knowledge. Corbett and Anderson, in their original articles, were interested in trying to add individualization to their model which they accomplished but with mixed results. Since their original work, the field has not made significant progress towards individualization of knowledge tracing models in fitting data. In this work, we introduce an elegant way of formulating the individualization problem entirely within a Bayesian networks framework that fits individualized as well as skill specific parameters simultaneously, in a single step. With this new individualization technique we are able to show a reliable improvement in prediction of real world data by individualizing the initial knowledge parameter. We explore three difference strategies for setting the initial individualized knowledge parameters and report that the best strategy is one in which information from multiple skills is used to inform each student’s prior. Using this strategy we achieved lower prediction error in 33 of the 42 problem sets evaluated. The implication of this work is the ability to enhance existing intelligent tutoring systems to more accurately estimate when a student has reached mastery of a skill. Adaptation of instruction based on individualized knowledge and learning speed is discussed as well as open research questions facing those that wish to exploit student and skill information in their user models.

This chapter has been published at the following venue:
Introduction

Our initial goal was simple; to show that with more data about students’ prior knowledge, we should be able to achieve a better fitting model and more accurate prediction of student data. The problem to solve was that there existed no Bayesian network model to exploit per user prior knowledge information. Knowledge tracing (KT) is the predominant method used to model student knowledge and learning over time. This model, however, assumes that all students share the same initial prior knowledge and does not allow for per student prior information to be incorporated. The model we have engineered is a modification to knowledge tracing that increases its generality by allowing for multiple prior knowledge parameters to be specified and lets the Bayesian network determine which prior parameter value a student belongs to if that information is not known beforehand. The improvements we see in predicting real world data sets are palpable, with the new model predicting student responses better than standard knowledge tracing in 33 out of the 42 problem sets with the use of information from other skills to inform a prior per student that applied to all problem sets. Equally encouraging was that the individualized model predicted better than knowledge tracing in 30 out of 42 problem sets without the use of any external data. Correlation between actual and predicted responses also improved significantly with the individualized model.

Inception of knowledge tracing

Knowledge tracing has become the dominant method of modeling student knowledge. It is a variation on a model of learning first introduced by Atkinson & Paulson (1972). Knowledge tracing assumes that each skill has 4 parameters; two knowledge parameters and two performance parameters. The two knowledge parameters are: initial (or prior) knowledge and learn rate. The initial knowledge parameter is the probability that a particular skill was known by the student before interacting with the tutor. The learn rate is the probability that a student will transition between the unlearned and the learned state after each learning opportunity (or question). The two performance parameters are: guess rate and slip rate. The guess rate is the probability that a student will answer correctly even if she does not know the skill associated with the question. The slip rate is the probability that a student will answer incorrectly even if she knows the required skill. Corbett and Anderson (1995) introduced this method to the intelligent tutoring field. It is currently employed by the cognitive tutor, used by hundreds of thousands of students, and many other intelligent tutoring systems to predict performance and determine when a student has mastered a particular skill.

It might strike the uninitiated as a surprise that the dominant method of modeling student knowledge in intelligent tutoring systems, knowledge tracing, does not allow for students to have different learn rates even though it seems likely that students differ in this regard. Similarly, knowledge tracing assumes that all students have the same probability of knowing a particular skill at their first opportunity.

In this chapter we hope to reinvigorate the field to further explore and adopt models that explicitly represent the assumption that students differ in their individual initial knowledge, learning rate and possibly their propensity to guess or slip.

Previous approaches to predicting student data using knowledge tracing

Corbett and Anderson were interested in implementing the learning rate and prior knowledge individualization that was originally described as part of Atkinson’s model of learning. They accomplished this but with limited success. They created a two step process for learning the parameters of their model where the four KT parameters were learned for each skill in the first
step and the individual weights were applied to those parameters for each student in the second step. The second step used a form of regression to fit student specific weights to the parameters of each skill. Various factors were also identified for influencing the individual priors and learn rates (Corbett & Bhatnagar, 1997). The results of Corbett & Anderson’s work showed that while the individualized model’s predictions correlated better with the actual test results than the non-individualized model, their individualized model did not show an improvement in the overall accuracy of the predictions.

More recent work by Baker, Corbett & Aleven (2008) has found utility in the contextualization of the guess and slip parameters using a multi-staged machine-learning processes that also uses regression to fine tune parameter values. Baker’s work has shown an improvement in the internal fit of their model versus other knowledge tracing approaches when correlating inferred knowledge at a learning opportunity with the actual student response at that opportunity but has yet to validate the model with an external validity test.

One of the knowledge tracing approaches compared to the contextual guess and slip method was the Dirichlet approach introduced by Beck & Chang (2007). The goal of this method was not individualization or contextualization but rather to learn plausible knowledge tracing model parameters by biasing the values of the initial knowledge parameter. The investigators of this work engaged in predicting student data from a reading tutor but found only a 1% increase in performance over standard knowledge tracing (0.006 on the AUC scale). This improvement was achieved by setting model parameters manually based on the authors understanding of the domain and not by learning the parameters from data.

The ASSISTments System

Our dataset consisted of student responses from The ASSISTments System, a web based math tutoring system for 7th-12th grade students that provides preparation for the state standardized test by using released math problems from previous tests as questions on the system. Tutorial help is given if a student answers the question wrong or asks for help. The tutorial help assists the student learn the required knowledge by breaking the problem into sub questions called scaffolding or giving the student hints on how to solve the question.

THE MODEL

Our model uses Bayesian networks to learn the parameters of the model and predict performance. Reye (2004) showed that the formulas used by Corbett and Anderson in their knowledge tracing work could be derived from a Hidden Markov Model or Dynamic Bayesian Network (DBN). Corbett and colleagues later released a toolkit (Chang et al., 2006) using non-individualized Bayesian knowledge tracing to allow researchers to fit their own data and student models with DBNs.

The Prior Per Student model vs. standard Knowledge Tracing

The model we present in this chapter focuses only on individualizing the prior knowledge parameter. We call it the Prior Per Student (PPS) model. The difference between PPS and Knowledge Tracing (KT) is the ability to represent a different prior knowledge parameter for each student. Knowledge Tracing is a special case of this prior per student model and can be derived by fixing all the priors of the PPS model to the same values or by specifying that there is only one shared student ID. This equivalence was confirmed empirically.
The two model designs are shown in Figure 1. Initial knowledge and prior knowledge are synonymous. The individualization of the prior is achieved by adding a student node. The student node can take on values that range from one to the number of students being considered. The conditional probability table of the initial knowledge node is therefore conditioned upon the student node value. The student node itself also has a conditional probability table associated with it which determines the probability that a student will be of a particular ID. The parameters for this node are fixed to be 1/N where N is the number of students. The parameter values set for this node are not relevant since the student node is an observed node that corresponds to the student ID and need never be inferred.

This model can be easily changed to individualize learning rates instead of prior knowledge by connecting the student node to the subsequent knowledge nodes thus training an individualized P(T) conditioned upon student as shown in Figure 2.
Parameter Learning and Inference

There are two distinct steps in knowledge tracing models. The first step is learning the parameters of the model from all student data. The second step is tracing an individual student’s knowledge given their respective data. All knowledge tracing models allow for initial knowledge to be inferred per student in the second step. The original KT work by Corbett & Anderson that individualized parameters added an additional step in between 1 and 2 to fit individual weights to the general parameters learned in step one. The PPS model allows for the individualized parameters to be learned along with the non-individualized parameters of the model in a single step. Assuming there is variance worth modeling in the individualization parameter, we believe that a single step procedure allows for more accurate parameters to be learned since a global best fit to the data can now be searched for instead of a best fit of the individual parameters after the skill specific parameters are already learned.

In our model each student has a student ID represented in the student node. This number is presented during step one to associate a student with his or her prior parameter. In step two, the individual student knowledge tracing, this number is again presented along with the student’s respective data in order to again associate that student with the individualized parameters learned for that student in the first step.

EXTERNAL VALIDITY: STUDENT PERFORMANCE PREDICTION

In order to test the real world utility of the prior per student model, we used the last question of each of our problem sets as the test question. For each problem set we trained two separate models: the prior per student model and the standard knowledge tracing model. Both models then made predictions of each student’s last question responses which could then be compared to the students’ actual responses.

Dataset description

Our dataset consisted of student responses to problem sets that satisfied the following constraints:

- Items in the problem set must have been given in a random order
- A student must have answered all items in the problem set in one day
- The problem set must have data from at least 100 students
- There are at least four items in the problem set of the exact same skill
- Data is from Fall of 2008 to Spring of 2010
Forty-two problem sets matched these constraints. Only the items within the problem set with the exact same skill tagging were used. 70% of the items in the 42 problem sets were multiple choice, 30% were fill in the blank (numeric). The size of our resulting problem sets ranged from 4 items to 13. There were 4,354 unique students in total with each problem set having an average of 312 students ($\sigma = 201$) and each student completing an average of three problem sets ($\sigma = 3.1$).

### Table 1. Sample of the data from a five item problem set

<table>
<thead>
<tr>
<th>Student ID</th>
<th>1st response</th>
<th>2nd response</th>
<th>3rd response</th>
<th>4th response</th>
<th>5th response</th>
</tr>
</thead>
<tbody>
<tr>
<td>750</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>751</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>752</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

In Table 1, each response represents either a correct or incorrect answer to the original question of the item. Scaffold responses are ignored in our analysis and requests for help are marked as incorrect responses by the system.

### Prediction procedure

Each problem set was evaluated individually by first constructing the appropriate sized Bayesian network for that problem set. In the case of the individualized model, the size of the constructed student node corresponded to the number of students with data for that problem set. All the data for that problem set, except for responses to the last question, was organized into an array to be used to train the parameters of the network using the Expectation Maximization (EM) algorithm. The initial values for the learn rate, guess and slip parameters were set to different values between 0.05 and 0.90 chosen at random. After EM had learned parameters for the network, student performance was predicted. The prediction was done one student at a time by entering, as evidence to the network, the responses of the particular student except for the response to the last question. A static unrolled dynamic Bayesian network was used. This enabled individual inferences of knowledge and performance to be made about the student at each question including the last question. The probability of the student answering the last question correctly was computed and saved to later be compared to the actual response.

### Approaches to setting the individualized initial knowledge values

In the prediction procedure, due to the number of parameters in the model, care had to be given to how the individualized priors would be set before the parameters of the network were learned with EM. There were two decisions we focused on: a) what initial values should the individualized priors be set to and b) whether or not those values should be fixed or adjustable during the EM parameter learning process. Since it was impossible to know the ground truth prior knowledge for each student for each problem set, we generated three heuristic strategies for setting these values, each of which will be evaluated in the results section.

### Setting initial individualized knowledge to random values

One strategy was to treat the individualized priors exactly like the learn, guess and slip parameters by setting them to random values to then be adjusted by EM during the parameter learning process. This strategy effectively learns a prior per student per skill. This is perhaps the most naïve strategy that assumes there is no means of estimating a prior from other sources of information and no better heuristic for setting prior values. To further clarify, if there are 600 students there will be 600 random values between 0 and 1 set for for each skill. EM will then
have 600 parameters to learn in addition to the learn, guess and slip parameters of each skill. For the non-individualized model, the singular prior was set to a random value and was allowed to be adjusted by EM.

**Setting initial individualized knowledge based on 1st response heuristic**

This strategy was based on the idea that a student’s prior is largely a reflection of their performance on the first question with guess and slip probabilities taken into account. If a student answered the first question correctly, their prior was set to one minus an ad-hoc guess value. If they answered the first question incorrectly, their prior was set to an ad-hoc slip value. Ad-hoc guess and slip values are used because ground truth guess and slip values cannot be known and because these values must be used before parameters are learned. The accuracy of these values could largely impact the effectiveness of this strategy. An ad-hoc guess value of 0.15 and slip value of 0.10 were used for this heuristic. Note that these guess and slip values are not learned by EM and are separate from the performance parameters. The non-individualized prior was set to the mean of the first responses and was allowed to be adjusted while the individualized priors were fixed. This strategy will be referred to as the “cold start heuristic” due to its bootstrapping approach.

**Setting initial individualized knowledge based on global percent correct**

This last strategy was based on the assumption that there is a correlation between student performance on one problem set to the next, or from one skill to the next. This is also the closest strategy to a model that assumes there is a single prior per student that is the same across all skills. For each student, a percent correct was computed, averaged over each problem set they completed. This was calculated using data from all of the problem sets they completed except the problem set being predicted. If a student had only completed the problem set being predicted then her prior was set to the average of the other student priors. The single KT prior was also set to the average of the individualized priors for this strategy. The individualized priors were fixed while the non-individualized prior was adjustable.

**Performance prediction results**

The prediction performance of the models was calculated in terms of mean absolute error (MAE). The mean absolute error for a problem set was calculated by taking the mean of the absolute difference between the predicted probability of correct on the last question and the actual response for each student. This was calculated for each model’s prediction of correct on the last question. The model with the lowest mean absolute error for a problem set was deemed to be the more accurate predictor of that problem set. Correlation was also calculated between actual and predicted responses.

<table>
<thead>
<tr>
<th>P(L_0) Strategy</th>
<th>Most accurate predictor (of 42)</th>
<th>Avg. Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PPS</td>
<td>KT</td>
</tr>
<tr>
<td>Percent correct heuristic</td>
<td>33</td>
<td>8</td>
</tr>
<tr>
<td>Cold start heuristic</td>
<td>30</td>
<td>12</td>
</tr>
<tr>
<td>Random parameter values</td>
<td>26</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 2 shows the number of problem sets that PPS predicted more accurately than KT and vice versa in terms of MAE for each prior strategy. This metric was used instead of average MAE to
avoid taking an average of averages. With the percent correct heuristic, the PPS model was able to better predict student data in 33 of the 42 problem sets. The binomial with \( p = 0.50 \) tells us that the probability of 33 success or more in 42 trials is \(< 0.05 \) (cutoff is 27 to achieve statistical significance), indicating a result that was not the product of random chance. In one problem set the MAE of PPS and KT were equal resulting in a total other than 42 (33 + 8 = 41). The cold start heuristic, which used the 1st response from the problem set and two ad-hoc parameter values, also performed well; better predicting 30 of the 42 problem sets which was also statistically significantly reliable. We recalculated MAE for PPS and KT for the percent correct heuristic this time taking the mean absolute difference between the rounded probability of correct on the last question and actual response for each student. The result was that PPS predicted better than KT in 28 out of the 42 problem sets and tied KT in MAE in 10 of the problem sets leaving KT with 4 problem sets predicted more accurately than PPS with the recalculated MAE. This demonstrates a meaningful difference between PPS and KT in predicting actual student responses. The correlation between the predicted probability of last response and actual last response using the percent correct strategy was also evaluated for each problem set. The PPS model had a higher correlation coefficient than the KT model in 32 out of 39 problem sets. A correlation coefficient was not able to be calculated for the KT model in three of the problem sets due to a lack of variation in prediction across students. This occurred in one problem set for the PPS model. The average correlation coefficient across all problem sets was 0.1933 for KT and 0.3515 for PPS using the percent correct heuristic. The MAE and correlation of the random parameter strategy using PPS was better than KT. This was surprising since the PPS random parameter strategy represents a prior per student per skill which could be considered an over parameterization of the model. This is evidence to us that the PPS model may outperform KT in prediction under a wide variety of conditions.

**Response sequence analysis of results**

We wanted to further inspect our models to see under what circumstances they correctly and incorrectly predicted the data. To do this we looked at response sequences and counted how many times their prediction of the last question was right or wrong (rounding predicted probability of correct). For example: student response sequence [0 1 1 1] means that the student answered incorrectly on the first question but then answered correctly on the following three. The PPS (using percent correct heuristic) and KT models were given the first three responses in addition to the parameters of the model to predict the fourth. If PPS predicted 0.68 and KT predicted 0.72 probability of correct for the last question, they would both be counted as predicting that instance correctly. We conducted this analysis on the 11 problem sets of length four. There were 4,448 total student response sequence instances among the 11 problem sets. Tables 3 and 4 show the top sequences in terms of number of instances where both models predicted the last question correctly (Table 3) and incorrectly (Table 4). Tables 5-6 show the top instances of sequences where one model predicted the last question correctly but the other did not.

<table>
<thead>
<tr>
<th># of Instances</th>
<th>Response sequence</th>
<th># of Instances</th>
<th>Response sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1167</td>
<td>1 1 1 1</td>
<td>251</td>
<td>1 1 1 0</td>
</tr>
<tr>
<td>340</td>
<td>0 1 1 1</td>
<td>154</td>
<td>0 1 1 0</td>
</tr>
<tr>
<td>253</td>
<td>1 0 1 1</td>
<td>135</td>
<td>1 1 0 0</td>
</tr>
<tr>
<td>252</td>
<td>1 1 0 1</td>
<td>106</td>
<td>1 0 1 0</td>
</tr>
</tbody>
</table>
Table 5. Predicted correctly by PPS only

<table>
<thead>
<tr>
<th># of Instances</th>
<th>Response sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>175</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>84</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>72</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>61</td>
<td>1 0 0 0</td>
</tr>
</tbody>
</table>

Table 6. Predicted correctly by KT only

<table>
<thead>
<tr>
<th># of Instances</th>
<th>Response sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>75</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>54</td>
<td>1 0 0 1</td>
</tr>
<tr>
<td>51</td>
<td>0 0 1 1</td>
</tr>
<tr>
<td>47</td>
<td>0 1 0 1</td>
</tr>
</tbody>
</table>

Table 3 shows the sequences most frequently predicted correctly by both models. These happen to also be among the top 5 occurring sequences overall. The top occurring sequence [1 1 1 1] accounts for more than 1/3 of the instances. Table 4 shows that the sequence where students answer all questions correctly except the last question is most often predicted incorrectly by both models. Table 5 shows that PPS is able to predict the sequence where no problems are answered correctly. In no instances does KT predict sequences [0 1 1 0] or [1 1 1 0] correctly. This sequence analysis may not generalize to other datasets but it provides a means to identify areas the model can improve in and where it is most strong. Figure 3 shows a graphical representation of the distribution of sequences predicted by KT and PPS versus the actual distribution of sequences. This distribution combines the predicted sequences from all 11 of the four item problem sets. The response sequences are sorted by frequency of actual response sequences from left to right in descending order.

![Response sequences for four question problem sets](image)

**Fig. 3.** Actual and predicted sequence distributions of PPS (percent correct heuristic) and KT

The average residual of PPS is smaller than KT but as the chart shows, it is not by much. This suggests that while PPS has been shown to provide reliably better predictions, the increase in performance prediction accuracy may not be substantial.

**CONTRIBUTION**

In this work we have shown how any Bayesian knowledge tracing model can easily be extended to support individualization of any or all of the four KT parameters using the simple technique of
creating a student node and connecting it to the parameter node or nodes to be individualized. The model we have presented allows for individualized and skill specific parameters of the model to be learned simultaneously in a single step thus enabling global best fit parameters to potentially be learned, a potential that is prohibitive with multi step parameter learning methods such as ones proposed by Corbett et al. (1995) and Baker et al. (2008).

We have also shown the utility of using this technique to individualize the prior parameter by demonstrating reliable improvement over standard knowledge tracing in predicting real world student responses. The superior performance of the model that uses PPS based on the student’s percent correct across all skills makes a significant scientific suggestion that it may be more important to model a single prior per student across skills rather than a single prior per skill across students, as is the norm.

DISCUSSION AND FUTURE WORK

We hope this chapter is the beginning of a resurgence in attempting to better individualize and thereby personalize students’ learning experiences in intelligent tutoring systems.

We would like to know when using a prior per student is not beneficial. Certainly if in reality all students had the same prior per skill then there would be no utility in modeling an individualized prior. On the other hand, if student priors for a skill are highly varied, which appears to be the case, then individualized priors will lead to a better fitting model by allowing the variation in that parameter to be captured.

Is an individual parameter per student necessary or can the same or better performance be achieved by grouping individual parameters into clusters? The relatively high performance of our cold start heuristic model suggests that much can be gained by grouping students into one of two priors based on their first response to a given skill. While this heuristic worked, we suspect there are superior representations and ones that allow for the value of the cluster prior to be learned rather than set ad-hoc as we did. Ritter et al. (2009) recently showed that clustering of similar skills can drastically reduce the number of parameters that need to be learned when fitting hundreds of skills while still maintaining a high degree of fit to the data. Perhaps a similar approach can be employed to find clusters of students and learning their parameters instead of learning individualized parameters for every student.

Our work here has focused on just one of the four parameters in knowledge tracing. We are particularly excited to see if by explicitly modeling the fact that students have different rates of learning we can achieve higher levels of prediction accuracy. The questions and tutorial feedback a student receives could be adapted to his or learning rate. Student learning rates could also be reported to teachers allowing them to more precisely or more quickly understand their classes of students. Guess and slip individualization is also possible and a direct comparison to Baker’s contextual guess and slip method would be an informative piece of future work.

We have shown that choosing a prior per student representation over the prior per skill representation of knowledge tracing is beneficial in fitting our dataset; however, a superior model is likely one that combines the attributes of the student with the attributes of a skill. How to design this model that properly treats the interaction of these two pieces of information is an open research question for the field. We believe that in order to extend the benefit of individualization to new users of a system, multiple problem sets must be linked in a single Bayesian network that uses evidence from the multiple problem sets to help trace individual student knowledge and more fully reap the benefits suggested by the percent correct heuristic.

This work has concentrated on knowledge tracing, however, we recognize there are alternatives. Draney, Wilson and Pirolli (1995) have introduced a model they argue is more parsimonious than knowledge tracing due to having fewer parameters. Additionally, Pavlik, Cen & Koedinger (2009) have reported using different algorithms, as well as brute force, for fitting.
the parameters of their models. We also point out that more standard models that do not track knowledge such as item response theory that have had large uses in and outside of the ITS field for estimating individual student and question parameters. We know there is value in these other approaches and strive as a field to learn how best to exploit information about students, questions and skills towards the goal of a truly effective, adaptive and intelligent tutoring system.

Chapter 3: Evaluating the Identifiability of the Model Parameters

Bayesian Knowledge Tracing (KT) models are employed by the cognitive tutors in order to determine student knowledge based on four parameters: learn rate, prior, guess and slip. A commonly used algorithm for learning these parameter values from data is the Expectation Maximization (EM) algorithm. Past work, however, has suggested that with four free parameters the standard KT model is prone to converging to erroneous degenerate states depending on the initial values of these four parameters. In this work we simulate data from a model with known parameter values and then run a grid search over the parameter initialization space of KT to map out which initial values lead to erroneous learned parameters. Through analysis of convergence and error surface visualizations we found that the initial parameter values leading to a degenerate state are not scattered randomly throughout the parameter space but instead exist on a surface with predictable boundaries. A recently introduced extension to KT that individualizes the prior parameter is also explored and compared to standard KT with regard to parameter convergence. We found that the individualization model has unique properties which allow it to avoid the local maxima problem.

This chapter was published at the following venue:

INTRODUCTION
Knowledge Tracing (KT) models (Corbett & Anderson, 1995) are employed by the cognitive tutors (Koedinger et al., 1997), used by over 500,000 students, in order to determine when a student has acquired the knowledge being taught. The KT model is based on two knowledge parameters: learn rate and prior and two performance parameters: guess and slip. A commonly used algorithm for learning these parameter values from data is the Expectation Maximization (EM) algorithm. Past work by Beck & Chang (2007), however, has suggested that with four free parameters the standard KT model is prone to converging to erroneous degenerate states depending on the initialized values of these four parameters. In this work we simulate data from a model with known parameter values and then brute force the parameter initialization space of KT to map out which initial values lead to erroneous learned parameters. Through analysis convergence and error surface visualizations we found that the initial parameter values leading to a degenerate state are not scattered randomly throughout the parameter space but instead exist on a surface within predictable boundaries. A recently introduced extension to KT that individualizes the prior parameter is also explored and compared to standard KT with regard to parameter convergence. We found that the individualization model has unique properties which allow for a greater number of initial states to converge to the true parameter values.
The Expectation Maximization (EM) algorithm is a commonly used algorithm used for learning the parameters of a model from data. EM can learn parameters from incomplete data as well as from a model with unobserved nodes such as the KT model. In the cognitive tutors, EM is used to learn the KT prior, learn rate, guess and slip parameters for each skill, or production rule. One requirement of the EM parameter learning procedure is that initial values for the parameters be specified. With each iteration the EM algorithm will try to find parameters that improve fit to the data by maximizing the log likelihood function, a measure of model fit. There are two conditions that determine when EM stops its search and returns learned parameter results: 1) if the specified maximum number of iterations is exceeded or 2) if the difference in log likelihood between iterations is less than a specified threshold. Meeting condition 2, given a low enough threshold, is indicative of algorithm parameter convergence, however, given a low enough threshold, EM will continue to try to maximize log likelihood, learning the parameters to greater precision. In our work we use a threshold value of 1e-4, which is the default for the software package used, and a maximum iteration count of 15. The max iteration value used is lower than typical, however, we found that in the average case our EM runs did not exceed more than 7 iterations before reaching the convergence threshold. The value of 15 was chosen to limit the maximum computation time since our methodology requires that EM be run thousands of times in order to achieve our goal.

Past work in the area of KT parameter learning

Beck & Chang (2007) explained that multiple sets of KT parameters could lead to identical predictions of student performance. One set of parameters was described as the plausible set, or the set that was in line with the authors’ knowledge of the domain. The other set was described as the degenerate set, or the set with implausible values such as values that specify that a student is more likely to get an item wrong if they know the skill. The author’s proposed solution was to use a Dirichlet distribution to constrain the values of the parameters based on knowledge of the domain.

Corbett & Anderson’s (1995) approach to the problem of implausible learned parameters was to impose a maximum value that the learned parameters could reach, such as a maximum guess limit of 0.30 which was used in Corbett & Anderson’s original parameter fitting code. This method of constraining parameters is still being employed by researchers such as Baker, Corbett & Aleven (2008) in their more recent models.

Alternatives to EM for fitting parameters were explored by Pavlik, Cen & Koedinger (2009), such as using unpublished code by Baker to brute force parameters that minimize an error function. Pavlik et al. (2009) also introduced an alternative to KT, named PFA and reported an increase in performance compared to the KT results. Gong, Beck and Heffernan (2010) however are in the process of challenging PFA by using KT with EM which they report provides improved prediction performance over PFA with their dataset.

While past works have made strides in learning plausible parameters they lack the benefit of knowing the true model parameters of their data. Because of this, none of past work has been able to report the accuracy of their learned parameters. One of the contributions of our work is to provide a closer look at the behavior and accuracy of EM in fitting KT models by using synthesized data that comes from a known set of parameter values. This enables us to analyze the learned parameters in terms of exact error instead of just plausibility. To our knowledge this is something that has not been previously attempted.
Methodology

Our methodology involves first synthesizing response data from a model with a known set of parameter values. After creating the synthesized dataset we can then train a KT model with EM using different initial parameter values and then measure how far from the true values the learned values are. This section describes the details of this procedure.

Synthesized dataset procedure

To synthesize a dataset with known parameter values we run a simulation to generate student responses based on those known ground truth parameter values. These values will later be compared to the values that EM learns from the synthesized data. To generate the synthetic student data we defined a KT model using functions from MATLAB’s Bayes Net Toolbox (BNT) created by Kevin Murphy (2001). We set the known parameters of the KT model based on the mean values learned across skills in a web based math tutor called ASSISTment (Pardos et al., 2008). These values which represent the ground truth parameters are shown in Table 7.

Table 1. Ground truth parameters used for student simulation

<table>
<thead>
<tr>
<th>Prior</th>
<th>Learn rate</th>
<th>Guess</th>
<th>Slip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform random dist.</td>
<td>0.09</td>
<td>0.14</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Since knowledge is modeled dichotomously, as either learned or unlearned, the prior represents the Bayesian network’s confidence that a student is in the learned state. The simulation procedure makes the assumption that confidence of prior knowledge is evenly distributed. 100 users and four question opportunities are simulated, representing a problem set of length four. Each doubling of the number of users also doubles the EM computation time. We found that 100 users was sufficient to achieve parameter convergence with the simulated data. Figure 1 shows pseudo code of the simulation.

```plaintext
KTmodel.lrate = 0.09
KTmodel.guess = 0.14
KTmodel.slip = 0.09
KTmodel.num_questions = 4
For user 1 to 100
    prior(user) = rand()
    KTmodel.prior = prior(user)
    sim_responses(user) = sample.KTmodel
End For
```

Figure 4. Pseudo code for generating synthetic student data from known KT parameter values

Student responses are generated probabilistically based on the parameter values. For instance, the Bayesian network will roll a die to determine if a student is in the learned state based on the student’s prior and the learn rate. The network will then again role a die based on guess and slip and learned state to determine if the student answers a question correct or incorrect at that opportunity. After the simulation procedure is finished, the end result is a datafile consisting of
100 rows, one for each user, and five columns; user id followed by the four incorrect/correct responses for each user.

**Analysis procedure**

With the dataset now generated, the next step was to start EM at different initial parameter values and observe how far the learned values are from the true values. A feature of BNT is the ability to specify which parameters are fixed and which EM should try to learn. In order to gain some intuition on the behavior of EM we decided to start simple by fixing the prior and learn rate parameters to their true values and focusing on learning the guess and slip parameters only. An example of one EM run and calculation of error is shown in the table below.

**Table 3. Example run of EM learning Guess and Slip of KT model**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>True value</th>
<th>EM initial value</th>
<th>EM learned value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guess</td>
<td>0.14</td>
<td>0.36</td>
<td>0.23</td>
</tr>
<tr>
<td>Slip</td>
<td>0.09</td>
<td>0.40</td>
<td>0.11</td>
</tr>
</tbody>
</table>

\[
\text{Error} = \frac{\text{abs(Guess}_{\text{true}} - \text{Guess}_{\text{learned}}) + \text{abs(Slip}_{\text{true}} - \text{Slip}_{\text{learned}})}{2}
\]

= 0.11

The true prior parameter value was set to the mean of the simulated priors (In our simulated dataset of 100 the mean prior was 0.49). Having only two free parameters allows us to represent the parameter space in a two dimensional graph with guess representing the X axis value and slip representing the Y axis value. After this exploration of the 2D guess/slip space we will touch on to the more complex three and four free parameter space.

**Grid search mapping of the EM initial parameter convergence space**

One of the research questions we wanted to answer was if the initial EM values leading to a degenerate state are scattered randomly throughout the parameter space or if they exist within a defined surface or boundary. If the degenerate initial values are scattered randomly through the space then EM may not be a reliable method for fitting KT models. If the degenerate states are confined to a predictable boundary then true parameter convergence can be achieved by restricting initial parameter values to within a certain boundary. In order to map out the convergence of each initial parameter we iterated over the entire initial guess/slip parameter space with a 0.02 interval. Figure 5 shows how this grid search exploration of the space was conducted.
We started with an initial guess and slip of 0 and ran EM to learn the guess and slip values of our synthesized dataset. When EM is finished, either because it reached the convergence threshold or the maximum iteration, it returns the learned guess and slip values as well as the log likelihood fit to the data of the initial parameters and the learned parameters (represented by LL\textsubscript{start} and LL\textsubscript{end} in the figure). We calculated the mean error between the learned and true values using the formula in Table 3. We then increased the initial slip value by 0.02 and ran EM again and repeated this procedure for every guess and slip value from 0 to 1 with an interval of 0.02.

**RESULTS**

The analysis procedure produced an error and log likelihood value for each guess/slip pair in the parameter space. This allowed for visualization of the parameter space using Guess\textsubscript{initial} as the X coordinate, Slip\textsubscript{initial} as the Y coordinate and either log likelihood or mean absolute error as the error function.

**Tracing EM iterations across the KT log likelihood space**

The calculation of error is made possible only by knowing the true parameters that generated the synthesized dataset. EM does not have access to these true parameters but instead must use log likelihood to guide its search. In order to view the model fit surface and how EM traverses across it from a variety of initial positions, we set the Z-coordinate (background color) to the LL\textsubscript{start} value and logged the parameter values learned at each iteration step of EM. We overlaid a plot of these EM iteration step points on the graph of model fit. This combined graph is shown below in figure 4 which depicts the nature of EM’s convergence with KT. For the EM iteration plot we tracked the convergence of EM starting positions in 0.10 intervals to reduce clutter instead of 0.02 intervals which were used to created the model fit plot. No EM runs reached their iteration max for this visualization. Starting values of 0 or 1 (on the borders of the graph) do not converge from the borders because of how BNT fixes parameters with 0 or 1 as their initial value.
Figure 7. Model fit and EM iteration convergence graph of Bayesian Knowledge Tracing. Small white dots represent parameter starting values. Green dots represent the parameter values at each EM iteration. The red dots represent the resulting learned parameter values and the large white dot is ground truth. The background color is the log likelihood (LL\textsubscript{start}) of the parameter space. Dark blue represent better fit.

This visualization depicts the multiple global maxima problem of Knowledge Tracing. There are two distinct regions of best fit (dark blue); one existing in the lower left quadrant which contains the true parameter values (indicated by the white “ground truth” dot), the other existing in the upper right quadrant representing the degenerate learned values. We can observe that all the green dots lie within one of the two global maxima regions, indicating that EM makes a jump to an area of good fit after the first iteration. The graph shows that there are two primary points that EM converges to with this dataset; one centered around guess/slip = 0.15/0.10, the other around 0.89/0.76. We can also observe that initial parameter values that satisfy the equation: guess + slip <= 1, such as guess/slip = 0.90/0.10 and 0.50/0.50, successfully converge to the true parameter area while initial values that satisfy: guess + slip > 1, converge to the degenerate point.

**KT convergence with all four parameters being learned**

For the full four parameter case we iterated through initial values of the prior, learn rate, guess and slip parameters from 0 to 1 with a 0.05 interval. This totaled 194,481 EM runs (21\textsuperscript{4}) to traverse the entire parameter space. For each set of initial positions we logged the converged
learned parameter values. In order to evaluate this data we looked at the distribution of converged values for each parameter across all EM runs.

Figure 5. Histograms showing the distribution of learned parameter values for each of the four Knowledge Tracing parameters. The first row shows the parameter distributions across all the EM runs. The second row shows the parameter distributions for the EM runs where initial guess and slip summed to less than 1.

The first row of histograms in Figure 5 shows the distribution of learned parameter values across all EM runs. Generally, we can observe that all parameters have multiple points of convergence; however, each histogram shows a clear single or bi-modal distribution. The prior and learn rate appear to be the parameters that are easiest to learn since the majority of EM runs lead to values near the true values. The guess and slip histograms exhibit more of the bi-modal behavior seen in the two parameter case, with points of convergence at opposite ends of the parameter space. In the two parameter case, initial guess and slip values that summed to less than one converged towards the ground truth coordinate. To see if this trend generalized with four free parameters we generated another set of histograms but only included EM runs where the initial guess and slip parameters summed to less than one. These histograms are shown in the second row.

Evaluating an extension to KT called the Prior Per Student model

We evaluated a recently introduced model (Pardos & Heffernan, 2010a) that allows for individualization of the prior parameter. By only modeling a single prior, Knowledge tracing makes the assumption that all students have the same level of knowledge of a particular skill before using the tutor. The Prior Per Student (PPS) model challenges that assumption by allowing each student to have a separate prior while keeping the learn, guess and slip as global parameters. The individualization is modeled completely within a Bayesian model and is accomplished with the addition of just a single node, representing student, and a single arc, connecting the student node to the first opportunity knowledge node. We evaluated this model using the two-parameter case, where guess and slip are learned and learn rate and prior are fixed to their true values.
Figure 6. EM convergence graphs of the Prior Per Student (PPS) model (left) and KT model (right). Results are shown with ground truth datasets with guess/slip of 0.30/0.30, 0.50/0.50 and 0.60/0.10.
Figure 6 shows that the KT models, in the right column, all have three separate points of convergence and only one of those points are near the ground truth coordinate (white dot). Unlike KT, the PPS models, in the left column, have a single point of convergence regardless of the starting position and that single point is near the ground truth values. The red lines in the second PPS model indicate that the maximum iteration count was reached. In the case of the PPS model there were as many prior parameters as there were students and these parameters were all set to the values that were generated for each simulated student as seen in the line “KTmodel.prior = prior(user)” in figure 1.

The PPS model was shown in Pardos & Heffernan (2010a) to provide improved prediction over standard knowledge tracing with real world datasets. The visualizations shown in figure 6 suggest that this improved prediction accuracy is likely due in part to the PPS model’s improved parameter learning accuracy from a wider variety of initial parameter locations.

However, we found that the model performed just as well, and in some cases better, when using that did not depend on knowing any ground truth prior values. This cold start heuristic essentially specifies two priors, either 0.05 or 0.95. A student associated with one of those two priors depending on their first question response; students who answered incorrectly on question 1 were given the 0.05 prior, students who answered correctly were give the 0.95 prior. This is very encouraging performance since it suggests that single point convergence to the true parameters is possible with the PPS model without the benefit of comprehensive individual student prior knowledge estimates.

**DISCUSSION AND FUTURE WORK**

An argument can be made that if two sets of parameters fit the data equally well then it makes no difference if the parameters used are the true parameters. This is true when prediction of responses is the only goal. However, when inferences about knowledge and learning are being made, parameter plausibility and accuracy is crucial. It is therefore important to understand how our student models and fitting procedures behave if we are to draw valid conclusions from them. In this work we have depicted how KT exhibits multi-modal convergence properties due to its multi-modal log likelihood parameter space. We demonstrated that with our simulated dataset, choosing initial guess and slip values that summed to less than one allowed for convergence towards the ground truth values in the two parameter case and in the four parameter case, applying this same rule resulted in a convergence distribution with a single mode close to the ground truth value.

This research also raises a number of questions such as how KT models behave with a different assumption about the distribution of prior knowledge. What is the effect of increased number of students or increased number of question responses per student on parameter learning accuracy? How does PPS converge with four parameters and what does the model fit parameter convergence space of real world datasets look like? These are questions that are still left to be explored by the EDM community.

**Chapter 4: Fully Individualized Student Parameter Model and Random Forests**

This work presents the modeling and machine learning techniques used to win 2nd student prize and 4th overall in the 2010 KDD Cup competition on Educational Data Mining. The KDD Cup gave 600 contestants 30 million data points from high school students' use of a computer tutoring system called the Cognitive Tutor. Contestants had to predict which problems students answered correctly or incorrectly. This competition
produced valuable scientific insight and advanced the state of the art in student modeling. This work, published in the journal of machine learning research, presents a new model that incorporates individual student traits, learned from the data, in order to make better predictions of their future performance. Random Forests, a machine learning algorithm based on decision trees, was also used to leverage rich feature sets engineered from the 30 million rows of student data. The various predictions made by the two methods were ensembled using ensemble-selection to provide the final prediction. The combination of these techniques represents a new level of accuracy in student modeling in this field. Following this competition, an educational data mining course was taught at WPI and several publications from first year graduate students in that course have stemmed from advancements and observations made during this competition.

This chapter is accepted for publication at the following venue:

INTRODUCTION
The datasets for the 2010 Knowledge Discover and Data Mining Cup came from Intelligent Tutoring Systems (ITS) used by thousands of students over the course of the 2008-2009 school year. This was the first time the Association for Computing Machinery (ACM) used an educational data set for the competition and also marked the largest dataset the competition has hosted thus far. There were 30 million training rows and 1.2 million test rows in total occupying over 9 gigabytes on disk. The competition consisted of two datasets from two different algebra tutors made by Carnegie Learning. One came from the Algebra Cognitive Tutor system; this dataset was simply called “Algebra”. The other came from the Bridge to Algebra Cognitive Tutor system whose dataset was aptly called “Bridge to Algebra”. The task was to predict if a student answered a given math step correctly or incorrectly given information about the step and the students past history of responses. Predictions between 0 and 1 were allowed and were scored based on root mean squared error (RMSE). In addition to the two challenge datasets, three datasets were released prior to the start of the official competition. Two datasets were from the two previous years of the Carnegie Learning Algebra tutor and one was from the previous year of the Bridge to Algebra tutor. These datasets were referred to as the development datasets. Full test labels were given for these datasets so that competitors could familiarize themselves with the data and test various prediction strategies before the official competition began. These datasets were also considerably smaller, roughly 1/5th the size of the competition datasets. A few anomalies in the 2007-2008 Algebra development dataset were announced early on; therefore that dataset was not analyzed for this article.

Summary of methods used in the final prediction model
The final prediction model was an ensemble of Bayesian Hidden Markov Models (HMMs) and Random Forests (bagged decision trees with feature and data re-sampling randomization). One of the HMMs used was a novel Bayesian model developed by the authors, built upon prior work (Pardos & Heffernan, 2010a) that predicts the probability of knowledge for each student at each opportunity as well as a prediction of probability of correctness on each step. The model learns individualized student specific parameters (learn rate, guess and slip) and then uses these parameters to train skill specific models. The resulting model that considers the composition of user and skill parameters outperformed models that only take into account parameters of the skill. The Bayesian model was used in a variant of ensemble selection (Caruana and Niculescu-Mizil,
and also to generate extra features for the decision tree classifier. The bagged decision tree classifier was the primary classifier used and was developed by Leo Breiman (Breiman, 2001).

The Anatomy of the Tutor
While the two datasets came from different tutors, the format of the datasets and underlying structure of the tutors was the same. A typical use of the system would be as follows; a student would start a math curriculum determined by her teacher. The student would be given multi step problems to solve often consisting of multiple different skills. The student could make multiple attempts at answering a question and would receive feedback on the correctness of her answer. The student could ask for hints to solve the step but would be marked as incorrect if a hint was requested. Once the student achieved “mastery” of a skill, according to the system, the student would no longer need to solve steps of that skill in their current curriculum, or unit.

The largest curriculum component in the tutor is a unit. Units contain sections and sections contain problems. Problems are the math questions that the student tries to answer which consist of multiple steps. Each row in the dataset represented a student’s answer to a single step in a problem. Determining whether or not a student answers a problem step correctly on the first attempt was the prediction task of the competition.

Students’ advancement through the tutor curriculum is based on their mastery of the skills involved in the pedagogical unit they are working on. If a student does not master all the skills in a unit, they cannot advance to the next lesson on their own; however, a teacher may intervene and skip them ahead.

Format of the datasets
The datasets all contained the same features and the same format. Each row in a dataset corresponded to one response from a student on a problem step. Each row had 18 features plus the target, which was “correct on first attempt”. Among the features were; unit, problem, step and skill. The skill column specified which math skill or skills were associated with the problem step that the student attempted. A skill was associated with a step by Cognitive tutor subject matter experts. In the development datasets there were around 400 skills and around 1,000 in the competition datasets. The Algebra competition set had two extra skill association features and the Bridge to Algebra set had one extra. These were alternative associations of skills to steps using a different bank of skill names (further details were not disclosed). The predictive power of these skill associations was an important component of our HMM approach.

![Figure 8](image.png)

**Figure 8.** The test set creation processes as illustrated by the organizers

The organizers created the competition training and test datasets by iterating through all the students in their master dataset and for each student and each unit the student completed,
selecting an arbitrary problem in that unit and placing into the test set all the student’s rows in that problem. All the student’s rows in that unit prior to the test set problem were placed in the training set. The rows following the selected problem were discarded. This process is illustrated in Figure 1 (compliments of the competition website).

**Missing data in the test sets**

Seven columns in the training sets were intentionally omitted from the test sets. These columns either involved time, such as timestamp and step duration or information about performance on the question, such as hints requested or number of incorrect attempts at answering the step. Competition organizers explained that these features were omitted from the test set because they made the prediction task too easy. In internal analysis we confirmed that step duration was very predictive of an incorrect or correct response and that the value of the hints and incorrects column completely determined the value of the target, “correct on first attempt”. This is because the tutor marks the student as answering incorrect on first attempt if they receive help on the question, denoted by a hint value of greater than 0. The incorrects value specified how many times the student answered the step incorrectly.

In the development datasets, valuable information about chronology of the steps in the test rows with respect to the training rows could be determined by the row ID column; however, in the challenge set the row ID of the test rows was reset to 1. The test row chronology was therefore inferred based on the unit in which the student answered problem steps in. A student’s rows for a given unit in the test set were assumed to come directly after their rows for that unit in the training set. While there may have been exceptions, this was a safe assumption to make given the organizers description of how the test rows were selected, as described in section 1.3.

**Data preparation**

The first step to being able to work with the dataset was to convert the categorical, alphanumeric fields of the columns into numeric values. This was done using perl to hash text values such as anonymized usernames and skill names into integer values. The timestamp field was converted to epoch and the problem hierarchy field was parsed into separate unit and section values. Rows were divided out into separate files based on skill and user for training with the Bayes Nets.

Special attention was given to the step duration column that describes how long the student spent answering the step. This column had a high percentage of null and zero values making it very noisy. For the rows in which the step duration value was null or zero, a replacement to the step duration value was calculated as the time elapsed between the current row’s timestamp and the next row’s timestamp for that same user. Outlier values for this recalculated step time were possible since the next row could be another day that the student used the system. It was also the case that row ID ordering did not strictly coincide with timestamp ordering so negative step duration values occurred periodically. Whenever a negative value or value greater than 1,000 seconds was encountered, the default step duration value of null or zero was kept. The step duration field was used for feature generation described in the Random Forests section.

**Creating an internal validation dataset**

An internal validation dataset was created in order to provide internal scoring of various prediction models. Besides using the scoring to test the accuracy of the Bayesian Networks and Random Forests methods it was also used to test various other approaches such as neural networks, linear regression and SVMs. A validation dataset was created for each of the competition datasets from the training datasets by taking all the rows in the last problem of each student’s units and placing them in the validation set and the remaining data into an internal training set. This process was meant to mirror the processes used by the organizers to create the official test set, described in section 1.3. The only difference was that the last problem in a unit
was selected instead of an arbitrary problem in a unit. The missing features from the official test sets were also removed from the created validation sets. By fashioning the validation sets after the official test set, a high correlation between validation and test set results should be achieved. A second validation set was also created so that ensemble methods could be tested internally. This set was created from the training rows that were not placed into the first validation set. The second validation set constituted rows from students’ second to last problem in each of their units.

**Knowledge Component columns in the dataset**
The Knowledge Component (KC) columns in the dataset described the skill or skills involved in the row’s problem step. Different KC columns used a different group of skills to describe a problem step. The KCs are used in Cognitive Tutors to track student learning over the course of the curriculum. KC skill associations that more accurately correlated with the student’s knowledge at that time will also more accurately predict future performance. Because of this it was important to explore which KC columns most accurately fit the data for each dataset.

**Rows of data where a KC column had no value**
There were a large percentage of rows (~20-25%) in both the training and test sets in which one or more KC columns had no value. That is, no skill was associated with the problem step. The Bayesian model needs skill associations to predict performance so this issue needed to be addressed. The solution was to treat null KC values as a separate skill with ID 1, called the NULL skill. A skill that appears in a separate unit is considered a separate skill so there were as many null ID skills as there were units. These null skill steps were predicted with relatively low error (RMSE ~0.20). In personal communication with Carnegie Learning staff after the competition, it was suggested that the majority of the null steps were most likely non math related steps such as clicking a button or other interface related interactions.

**Handling of KC values with multiple skills**
There can be one or more skills associated with a step for any of the KC columns. Modeling multiple skills with Knowledge Tracing is significantly more complex and is not a standard practice in student modeling. To avoid having to model multiple skills per step, the KC values with multiple skills were collapsed into one skill. Two strategies for collapsing the values were tried for each KC column. The first was to keep only the most difficult skill. This approach is based on the hypothesis that skills compose conjunctively in an ITS. Difficulty was calculated based on percent correct of all rows in the training set containing that skill. KC models applying this strategy will be labeled with “-hard” throughout the text. The second way of collapsing multiple skill values was to treat a unique set of skills as a completely separate skill. Therefore, a step associated with “Subtraction” and “Addition” skills would be merged into the skill of “Subtraction-Addition”. KC models applying this strategy will be labeled with “-uniq” throughout the text. The result of this processing was the generation of two additional skill models for each KC column for each challenge set. All of the development dataset analysis in this chapter uses only the unique strategy, for brevity.

**BAYESIAN NETWORKS APPROACH**
Bayesian Networks were used to model student knowledge over time. A simple HMM with one hidden node and one observed node has been the standard for tracking student knowledge in ITS and was introduced to the domain by Corbett and Anderson (Corbett & Anderson, 1995). In this model, known as Knowledge Tracing, a student’s incorrect and correct responses to questions of a particular skill are tracked. Based on the parameters of the HMM for that skill and the student’s past responses, a probability of knowledge is inferred. In the Cognitive Tutor, students who know a skill with 95% probability, according to the HMM, are considered to have mastered that skill.
There are four parameters of the HMM and they can be fit to the data using Expectation Maximization (EM) or a grid search of the parameter space. We used EM with a max iteration of 100. EM will also stop if the log likelihood fit to the data increases by less than 1e-5 between iterations. While this simple HMM was the basis of our Bayesian Networks approach, additional models which utilized the parameters learned by the simpler models were utilized for prediction.

The Prior Per Student Model (Simple Model)
Standard knowledge tracing has four parameters. A separate set of parameters are fit for each skill based on students’ sequences of responses to steps of that skill. The intuition is that students will learn a skill over time. The latent represents knowledge of that skill and the two transition probabilities for the latent are prior knowledge and learning rate. Prior knowledge is the probability that students knew the skill prior to working on the tutor. Learning rate is the probability that students will transition from the unlearned to the learned state between opportunities to answer steps of that skill. The probability of transitioning from learned to unlearned (forgetting) is fixed at zero since the time between responses is typically less than 24 hours. Forgetting is customarily not modeled in Knowledge Tracing; however, it certainly could be occurring given a long enough passage of time between opportunities. The two emission probabilities are the guess and slip rate. Guess is the probability of observing a correct response when the student is in the unlearned state. Slip is the probability of observing an incorrect response when the student is in the learned state. Prior work by the authors has shown that modeling a separate prior per student in the training and prediction steps can increase the accuracy of the learned parameters (Pardos & Heffernan, 2010b) as well as prediction accuracy (Pardos & Heffernan, 2010a). In parameter analysis work, simulated datasets created from a known distribution were analyzed by the standard knowledge tracing model and by one that allowed for a prior per student based on the student’s first response. The prior per student model resulted in more accurate convergence to the ground truth parameter values regardless of initial parameter values for EM parameter learning. The standard Knowledge Tracing model, however, was very sensitive to initial parameter values in converging to the ground truth parameters.

Figure 9. Prior Per Student (PPS) model parameters and topology

Figure 2 shows the Prior Per Student (PPS) model topology. In this model the student node acts as a unique student identifier with values that range from 1 to N where N is the number of students in the dataset; however, we have found that modeling only two distinct priors and assigning a student to one of those priors based on their first response is an effective heuristic. We refer to this as the cold start heuristic. If a student answers the first observed step incorrectly, they are assigned a prior of 0.10, if they answer the step correctly; they are assigned a prior of 0.85.
These values were chosen *ad-hoc* based on experimentation with this and other datasets. One alternative to the *ad-hoc* setting is to let the two prior seeding values be adjusted and learned from data. These values may be capturing guess and slip probabilities so another alternative is to have the prior seeding values be the same as the guess and slip values. We tested these three strategies with the two development datasets and found the following results, shown in Table 1.

<table>
<thead>
<tr>
<th>Algebra (development)</th>
<th>Bridge to Algebra (development)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Strategy</strong></td>
<td><strong>RMSE</strong></td>
</tr>
<tr>
<td>1 adjustable</td>
<td>0.3659</td>
</tr>
<tr>
<td>2 guess/slip</td>
<td>0.3660</td>
</tr>
<tr>
<td>3 Ad-hoc</td>
<td>0.3662</td>
</tr>
</tbody>
</table>

Table 8. Results of prior seeding strategies on the two development datasets

Table 1 shows that for the algebra (development) datasets, the difference between the *ad-hoc* and adjustable strategy was 0.0003. This appeared to be a small benefit at the time and the extra free parameters of the adjustable strategy added to the compute time of the EM runs. While the guess/slip strategy added less compute time than the adjustable strategy, the *ad-hoc* value strategy was chosen to be used going forward with all models used for the competition datasets because of the small difference in RMSE and because this strategy had already been more carefully studied in past work (Pardos & Heffernan, 2010b). Another reason *Ad-hoc* was chosen is because it appeared to be the best strategy in the bridge to algebra dataset when initially calculated. Upon closer inspection for this article, the *Ad-hoc* prediction was missing around 250 rows compared to the other strategy predictions. After correcting this, the guess/slip strategy appears favorable.

**Limiting the number of student responses used**

The EM training for skills with high amounts of student responses would occupy over 8GB of virtual memory on the compute machines. This was too much as the machines used to run these models had only 8GB and reaching into swap memory caused the job to take considerably longer to finish. The skills with high amounts of data often had over 400 responses by one student. To alleviate the memory strain, limits were placed on the number of most recent responses that would be used in training and prediction. The limits tested were 5, 10, 25, 150 and none.

<table>
<thead>
<tr>
<th>Algebra (development)</th>
<th>Bridge to Algebra (development)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Limit</strong></td>
<td><strong>RMSE</strong></td>
</tr>
<tr>
<td>1 25</td>
<td>0.3673</td>
</tr>
<tr>
<td>2 150</td>
<td>0.3675</td>
</tr>
<tr>
<td>3 none</td>
<td>0.3678</td>
</tr>
<tr>
<td>4 10</td>
<td>0.3687</td>
</tr>
<tr>
<td>5 5</td>
<td>0.3730</td>
</tr>
</tbody>
</table>

Table 9. Results of limiting the number of most recent student responses used for EM training

Table 2 shows the prediction RMSE on the development sets when limiting the number of most recent student responses used for training and prediction. A surprising result was that very few responses were needed to achieve the same or better results as using all data. In the algebra (development) set, 25 was the best limit of the limits tried and was the second best limit in the bridge to algebra (development) set. This prediction improvement was a welcomed bonus in addition to eliminating the memory issue which would have been compounded when working with the much larger competition sets. A limit of 25 would be used for all subsequent models.
Distribution of skill parameters

Using the PPS model: learn, guess and slip rates were learned from the data for all 387 skills in the algebra (development) set and 442 skills in the bridge to algebra (development) set. The distribution of the values of those parameters is shown with histograms in Figure 3.

![Histograms showing the distribution of learning, guess, and slip rates for Algebra and Bridge to Algebra development sets.](image)

**Figure 3.** Distribution of skill parameters in the algebra and bridge to algebra development sets

The X axis of the histograms in Figure 3 is the value of the parameter and the Y axis is the occurrence of that parameter value among the skills in the dataset. These parameters were learned from the data using EM with the prior per student model (cold start heuristic). Figure 3 shows that both datasets are populated with skills of various learning rates with a higher frequency of skills that are either very hard or very easy to learn. Both datasets have a high frequency of skills that are both hard to guess and hard to slip on. The Algebra (development) set appears to have slightly more skills with higher slip rates than bridge to algebra (development).

Prediction performance of the KC models in the challenge datasets

Unlike the development sets, the challenge datasets had multiple KC columns which gave different skill associations for each step. The bridge to algebra set had two KC columns while the algebra set had three. As described in section 2.2.2, two versions of each KC model were created; each using a different strategy for converting multi skill step representations to a single skill. The results in Table 3 describe the KC model and RMSE. KC model “2-hard”, for instance, refers to the 2nd KC model for that dataset with “use the hardest skill” applied for multiple skill steps while KC model “2-uniq” refers to the 2nd KC model using “treat a set of skills as a separate skill”.

<table>
<thead>
<tr>
<th>Algebra (challenge)</th>
<th>Bridge to Algebra (challenge)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KC model</strong></td>
<td><strong># Skills</strong></td>
</tr>
<tr>
<td>1</td>
<td>3-hard</td>
</tr>
<tr>
<td>2</td>
<td>3-uniq</td>
</tr>
<tr>
<td>3</td>
<td>1-hard</td>
</tr>
<tr>
<td>4</td>
<td>1-uniq</td>
</tr>
<tr>
<td>5</td>
<td>2-uniq</td>
</tr>
<tr>
<td>6</td>
<td>2-hard</td>
</tr>
</tbody>
</table>

*Table 3. Prediction accuracy of the KC models in both challenge datasets*
The most significant observation from Table 3 is the considerably better performance of the third KC model in the algebra set. The difference of 0.0185 between the algebra KC models 3-hard and 1-hard is greater than the RMSE difference between the first and tenth overall finisher in the competition. The differences between the multiple skill approaches were negligible. Table 3 also shows the number of skills in each competition datasets per KC model with the hard and unique multi-skill reduction strategy applied. The unique strategy always created more rules but the difference is most prominent for KC column 1. The table also shows how the various KC models differ in skill granularity; Algebra model 2-hard has only 606 skills used to associated with steps while Algebra model 3-hard used 2,359 skills to associate with those steps. Among the “hard” models, the more skills the KC model had, the better it performed.

It is important to note that the Bayesian models only made predictions when there existed previous responses by the student to the skill being predicted. If no prior skill data existed no prediction was made. No previous skill information for a student was available in a significant portion of the test data (~10%). Therefore, the RMSE scores shown in Table 3 represent the RMSE only for the predicted rows and not the entire test set. It was also the case that total number of predicted rows for each KC model differed by ~1,200, likely due to a Bayesian skill prediction job not finishing or other processing anomaly. While 1,200 rows only constitutes 0.2% of the total algebra test rows it was a significant enough difference to cause the algebra 3-uniq KC model to appear to have a lower RMSE than 3-hard and for the bridge to algebra KC model 1-uniq to appear to have a lower RMSE than 1-hard in our preliminary RMSE calculations. Because of this, all subsequent models run during the competition were created using 3-uniq and 1-uniq. The RMSE scores in Table 3 are the corrected calculations based only on the test rows that all the KC model predictions had in common which was 435,180/508,912 (86%) rows for algebra and 712,880/774,378 (92%) rows for bridge to algebra. The additional prediction rows were filled in by Random Forests for the final submission.

**The Student-Skill Interaction Model (Complex Model)**

The more complex model expanded on the simple model considerably. The idea was to learn student specific learn, guess and slip rates and then use that information in training the parameters of skill specific models. The hypothesis is that if a student has a general learning rate trait then it can be learned from the data and used to benefit inference of how quickly a student learns a particular skill and subsequently the probability they will answer a given step correctly. This model was created during the competition and has not been described previously in publication.

The first step in training this model was to learn student parameters one student at a time. Student specific parameters were learned by using the PPS model by training on all skill data of an individual student one at a time. The rows of the data were skills encountered by the student and the columns were responses to steps of those skills. All responses per skill started at column 1 in the constructed training set of responses. Some skills spanned more columns than others due to more responses on those skills. EM is able to work with this type of sparsity in the training matrix.

The second step was to embed all the student specific parameter information into the complex model, called the Student-Skill Interaction (SSI) model, shown in Figure 4. Parameters were then learned for the SSI model given the student specific parameter values. After the parameters were trained the model could be used to predict unseen data given past history of responses of a student on a skill. Depending on the learning rate of the skill and the learning rate of the user, the model would forecast the rate of acquiring knowledge and give predictions with increasing probability of correct on each subsequent predicted response for a student on steps of a particular skill.

The limitation of the model is that it requires that a plentiful amount of data exists for the student in order to train their individual parameters. The format of the competition’s data was ideal for this model since the students in the training set also appeared in the test set and because student data was available in the training set for a variety of skills.
There was an SSI model trained for each skill but each SSI model was fixed with the same student specific parameter data. For example, the list of student learning rates is placed into the conditional probability table of the $T$ node. There are six parameters that are learned in the SSI model. The effect of the student parameter nodes is to inform the network which students have high or low learn, guess or slip rates and allow the skill parameters to be learned conditioning upon this information. For example, two learning rates will be learned for each skill. One learning rate for if the student is a high learner (described in the $T$ node) and one learning rate for if the student is a low learner. The same is done for the skill’s guess and slip parameters. These values can be different for each skill but they are conditioned upon the same information about the students. While a student may have a high individual learn rate, the fast-student learn rate for a difficult skill like Pythagorean Theorem may be lower than the fast-student learn rate for subtraction. The model also allows for similar learn rates for both fast and slow student learners. Results of SSI vs. PPS are shown in Table 4. The improvement is modest but was greater than the difference between 1st and 3rd place overall in the competition. The difference between SSI and PPS squared errors were significant for both datasets at the $p << 0.01$ level using a paired t-test.

<table>
<thead>
<tr>
<th>Algebra (challenge)</th>
<th>Bayesian model</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SSI (KC 3-2)</td>
<td>0.2813</td>
</tr>
<tr>
<td>2</td>
<td>PPS (KC 3-2)</td>
<td>0.2835</td>
</tr>
<tr>
<td>Improvement: 0.0022</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bridge to Algebra (challenge)</th>
<th>Bayesian model</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SSI (KC 1-2)</td>
<td>0.2824</td>
</tr>
<tr>
<td>2</td>
<td>PPS (KC 1-2)</td>
<td>0.2856</td>
</tr>
<tr>
<td>Improvement: 0.0032</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 4. Results of the SSI model vs. the PPS model.

#### Distribution of student parameters

Individual student learn, guess and slip rates were learned from the data for all 575 student in the algebra (development) set and 1,146 student in the bridge to algebra (development) set. The distribution of the values of those parameters for each dataset is shown in Figure 5.
Figure 5 shows that users in both datasets have low learning rates but that a small portion of students posses learning rates in each range. Moderate guessing and low slipping existed among students in both datasets. The majority of the parameters learned fell within plausible ranges.

RANDOM FORESTS APPROACH
Leo Breiman’s Random Forests© (Breiman, 2001) were used to make predictions based on a rich set of features from the training and testing sets. Random Forests is a variant on bagged decision trees. Random Forests trains an ensemble of decision tree classifiers or regression trees and uses bagging to combine predictions. Each tree selects a portion of the features at random and a random resampling of the data to train on. This approach required feature engineering and feature extraction as opposed to the HMM approach which required student responses grouped by skill.

Parameters of the Random Forest algorithm
MATLAB’s TreeBagger implementation of bagged decision trees was used. Regression mode was used so that the end prediction would be a value between 0 and 1 representing the probability of the binary class. The number of features for each tree to sample was left at its default for regression mode; 1/3rd the number of features. The two parameters that were modified were MinLeaf and NumTrees. MinLeaf is the minimum number of observations needed per tree leaf. This is recommended to be set at 1 for classification and 5 for regression; however, the optimal values for this parameter were often between 15 and 65 based on testing with a small validation set. The NumTrees parameter is the number of random decision trees trained. The rule of thumb is to use a value of 50 or greater. Values between 50 and 800 were tried. For some of the feature sets a randomly chosen 50,000 rows were used for training and 50,000 for testing in order to do a parameter search of the optimal MinLeaf parameter. MinLeaf was searched from 1 to 100 in increments of 1 and NumTrees was set at 50 for this parameter search. NumTrees did not appear to affect the optimal MinLeaf value chosen; however, this was not tested thoroughly. It is possible that there is a different optimal MinLeaf value depending on NumTrees. Each tree trained by Random Forests resamples from the training data, with replacement. The size of the resampled training data can be set, however, this was left at its default value which was to create a resampled set the same size as the original training set.
Feature Extraction

Feature sets for random forest training and prediction were created. Some were created based on KCs while others were based on user and problem properties. The largest set contained 146 featured. The features created for training had to also be created for the official test sets, which contained missing features that were not missing in the official training sets. With this in mind, the strategy was to aggregate existing features from the training set into the test set. An example of this would be creating a feature called “average step duration of student on skill”. The feature of “step duration” exists in the official training set but not the test set. So, in order to add this feature to the test set, the average step duration of student 10 on skill B, for example, was calculated for the data in the training set and every row in the test set that contained student 10 and skill B was given this aggregated value for the “average step duration of student on skill” feature column. This process was used to create many of the features for the test set. The training dataset had to contain the same features as the test set in order for the test set features to be of use.

In order to accomplish this, the internal validation sets were utilized. Since the validation set was of the same format as the test set, the same feature creation procedure was run on the validation set using the remainder of the training set data not used in the validation set. Figure 6 depicts the portions of the dataset that were used for generating the feature rich datasets that the Random Forests ultimately trained on.

Figure 6 shows, for instance, how the non-validation training rows (nvtrain) in addition to validation set 2 (val2) were used to generate features for the feature rich validation set 2 (frval2). Only nvtrain was used to generate missing test set related features, such as “average step duration of student on skill”, for frval2; however, val2 could still be used to generate features that were available in the official test set, such as “number of questions answered by student in this problem”.

Random Forests was trained on frval1 to predict frval2 and trained on frval2 to predict frval1 as part of a 2-fold internal cross-validation. The datasets frval1 and frval2 were combined when training the Random Forests models to make predictions on the official test set. The cross-validated frval1 and frval2 predictions were combined as the validation set for Ensemble selection. The Bayesian network SSI model was also used to generate features as well as its predictions for the feature rich sets. To produce features for frval2, only nvtrain was used to train parameters for the model. To produce features for frval1, only data from nvtrain+val1+val2 were used and to produce features for the official test set, data from the entire training set were used (nvtrain+val1+val2).
Percent correct features
For each skill, the percent correct of steps associated with that skill was calculated for each section, problem and step the skill was associated with including the overall percent correct for steps of that skill. This was done for each of the skill models in each of the challenge datasets. Percent correct was also calculated for each student by unit, section, problem, step and overall percent correct. These features were joined into the test sets that will be used as training sets. The joining looks at the user, skill, unit, section, problem and step of the row in the test set and adds the appropriate ten percent correct features to it, five from user and five from skill.

Student progress features
These features were based upon previous performance of a student in the training set prior to answering the test row. Many of these features were adopted from work on gaming the system (Baker et al., 2008) which is a type of behavior a student can exhibit when he or she is no longer trying to learn or solve the problem but instead is clicking through help and hints in the problem. Features of student progress that were generated included the following:

- The number of data points: [today, on the first day of using the tutor, since starting the tutor, on the first day of starting the current unit]
- The number of correct answers among the last [3, 5, 10] responses
- The percent correct among the last [3, 5, 10] responses
- The number of steps out of the last 8 in which a hint was requested
- The mean number of hints requested in the last 10 steps
- The mean number of incorrect attempts in the last 10 steps
- The number of days since [starting the tutor, starting the unit]
- The sum of the last [3, 10] z-scores for [step duration, hints requested, incorrect attempts]

Z-scores were calculated by first calculating the mean and standard deviation of step duration, hints requested and incorrect attempts on a step for each skill. A z-score for step duration, for instance, was calculated by taking the step duration of a student on the last step and subtracting the mean step duration for that skill and then dividing that by the standard deviation step duration for that skill. The sum of the last three such z-scores constituted a feature. In addition to the features listed above, identical features were generated specific to the skill associated with the test row. For example, the feature “number of data points today” would become “number of data points of skill X today” where skill X is the skill associated with the test row that the feature value is being generated for. There was often not enough past data for a particular skill to calculate the feature. Because of this, the skill specific version of student progress feature set covered fewer rows than the non-skill specific version.

Bayesian HMM features
The SSI model, which was run for each skill in the KC model of a dataset, generated various outputs that were treated as features for the Random forests. The features generated included:

- The predicted probability of correct for the test row
- The inferred probability of knowing the skill
- The absolute value of the inferred probability of knowing the skill subtracted by the predicted probability of correct
- The number of students used in training the parameters
- The number of data points used in training the parameters
- The final EM log likelihood fit of the parameters divided by the number of data points
- The total number of steps in the predicted test problem
- The number of steps completed thus far in the predicted test problem
The number of steps completed divided by the total number of steps in the test problem

Similar to the skill specific student progress features, the Bayesian HMM features required that prior skill data for the student be available. If such data were not available, no features were created for that test row. Because of this the Bayesian feature set did not cover all the test rows.

Random forest prediction results
After generating features for the three datasets (two validation sets and the official test set) based on the top KC models, Random forests were trained on the two validations sets. The RMSE results for the validation sets are shown in Table 5 for the best performing Random Forest parameter combination for the full feature set. Coverage percentage is also included indicating what percentage of the total validation rows were predicted by the feature set. Prediction using only basic percent correct features to train on is also included as a baseline.

<table>
<thead>
<tr>
<th>Feature set</th>
<th>RMSE</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 All features</td>
<td>1.4942</td>
<td>87%</td>
</tr>
<tr>
<td>2 Percent correct+</td>
<td>0.2824</td>
<td>96%</td>
</tr>
<tr>
<td>3 All features (fill)</td>
<td>0.2847</td>
<td>97%</td>
</tr>
</tbody>
</table>

Table 5. Random forest prediction results

Table 5 shows that with all features, Random forests predict 92% of the bridge to algebra test set with an RMSE of 0.2712. This is outstanding prediction accuracy given that the winning RMSE for this dataset on the leaderboard was 0.2777. The problem was that the remaining 8% of the test rows represent students who do not have past data for the skill being predicted and this group was particularly difficult to predict. The “All features (fill)” was an attempt to fill in the missing values of “All features” by using the mean value of a column in place of its missing values and retraining on this “filled in” dataset. This approach provided some benefit over using just percent correct features to train the Random Forests with the bridge to algebra set but performed worse than the percent correct features in the algebra set. An improvement on this would have been to take the mean value for the column among the rows of the same skill. A step further still would have been to predict or impute the missing values using Random Forests. Time ran out in the competition so these last two steps became approaches for future investigation.

Feature set importance
The importance of the various feature sets was calculated by turning on the Random Forests out of bag permuted variable error calculation. This feature allowed the model to permute the value of each feature and then observe the change in mean standard error among tree predictions. A higher positive change in error indicates more importance.

<table>
<thead>
<tr>
<th>Feature set</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Student progress</td>
<td>1.4942</td>
</tr>
<tr>
<td>2 Percent correct (Skill)</td>
<td>1.3615</td>
</tr>
<tr>
<td>3 Student progress (Skill)</td>
<td>1.3094</td>
</tr>
<tr>
<td>4 Percent correct (User)</td>
<td>1.2732</td>
</tr>
<tr>
<td>5 SSI model features</td>
<td>0.9993</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature set</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Percent correct (User)</td>
<td>2.1831</td>
</tr>
<tr>
<td>2 Student progress</td>
<td>2.0989</td>
</tr>
<tr>
<td>3 Student progress (Skill)</td>
<td>1.8118</td>
</tr>
<tr>
<td>4 SSI model features</td>
<td>1.6436</td>
</tr>
<tr>
<td>5 Percent correct (Skill)</td>
<td>1.5950</td>
</tr>
</tbody>
</table>

Table 6. Random Forests average variable importance for each feature set
A Random Forests model was trained on frval2 with the all features dataset with 200 trees and min leaf of 15. The average importance of each variable within a feature set was calculated to produce the results in Table 6. The table shows that the student progress feature set was highly important in both datasets. The percent correct features of the user were most important in the bridge to algebra set; however, the percent correct features of skill were the least important. Inversely, in the algebra set, the percent correct features of skill were more important than percent correct feature of the user. The importance of user features on bridge to algebra is perhaps one reason why the user and skill oriented SSI model showed a greater improvement over the skill only PPS model on bridge to algebra. The SSI model features added value but made the least impact on error on average. This was the only feature set containing features and a prediction from another classifier. This characteristic could have made it difficult for the decision trees to find additional exploitable patterns in these features.

**Ensemble selection**

A variant of ensemble selection (Caruana & Niculescu-Mizil, 2004) was used to blend the collection of Random Forests and Bayesian networks generated predictions. Because of the varying number of test rows covered by the predictions of each model, a special ensemble initialization technique was created whereby the best model was chosen first based on lowest validation set RMSE and subsequent models were chosen based on the RMSE of the predicted rows excluding the rows already added to the initialized ensemble. This allowed for models to be used for the portions of the data in which they excelled. For instance, the rows of the test set containing skills sparsely seen by the user were best predicted by a model that was not a top predicting model overall.

After the initialization, all models were averaged with the current ensemble to determine which resulted in the best improvement to RMSE. The processes stopped when no averaging of models would improve RMSE with respect to the validation set. Only three models were chosen in the averaging stage for the bridge to algebra set and two for the algebra set. In this ensemble selection procedure, the validation set RMSE is minimized and the same actions are performed on the official test predictions as on the validation predictions. Since two validation sets had been made, we were able to confirm that this ensemble selection procedure decreased the RMSE on a hold out set and confirmed the benefit on the official test set through feedback from the leaderboard. Table 7 shows the models chosen during the initialization processes and what percent of the test rows were covered after adding the prediction’s rows to the ensemble. There were 76 models for ensemble selection of the algebra set and 81 for the bridge to algebra set. This included the Bayesian model predictions and Random forest predictions with various parameters.

<table>
<thead>
<tr>
<th>Algebra (challenge)</th>
<th>Bridge to Algebra (challenge)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prediction file</strong></td>
<td><strong>RMSE</strong></td>
</tr>
<tr>
<td>1刘  Rf600m35_allFeat</td>
<td>0.2762</td>
</tr>
<tr>
<td>2刘  SSI_KC_3-unq</td>
<td>0.2758</td>
</tr>
<tr>
<td>3刘  Rf100m15_hints</td>
<td>0.2839</td>
</tr>
<tr>
<td>4刘  Rf100m15_pctCor</td>
<td>0.2840</td>
</tr>
</tbody>
</table>

RMSE after blending (2 models): 0.2834  
RMSE after blending (3 models): 0.2780

**Table 7.** Ensemble selection procedure and RMSE improvement on the hill climbing set

Table 7 shows that the most accurate model chosen for both datasets was a Random Forests model. The second model chosen was the Bayesian SSI model illustrating that the Bayesian model captured variance not captured by the Random Forests models. This was likely due to the Bayesian model’s ability to competently model the temporal nature of the data.
CONCLUSION

Combining user features with skill features was very powerful in both Bayesian and Random Forests approaches. Prediction error was very low for rows that had sufficient data to compile a complete user and skill feature set however error was very high for rows where the user did not have sufficient skill data. In order to increase prediction accuracy for these rows, imputing missing features could be very beneficial. Handling these rows is a worthy area of future study since prediction error of these rows substantially increased overall RMSE. Feature selection would likely have also improved prediction and a closer study of individual features importance is an important open question for future work.

The strong performance of the Knowledge Tracing based PPS and SSI models demonstrated the power of the HMM assumption of learning in educational datasets. Only using the students’ past sequence of correct and incorrect responses by skill, the HMM model’s predictions rivaled that of the Random Forests approach which required substantial feature engineering. The Random Forests predictions, however, were able to increase the level of prediction accuracy by leveraging features not included in the HMMs. Random forests has not been previously used in the ITS community, however, given its standalone performance and performance in concert with HMMs, they would be a valuable option to consider for future research in student performance prediction.

Notes on other machine learning techniques attempted: Neural networks with 1-3 hidden layers were tried with layer node sizes iterated between 2 and 100. The predictive performance of the NNs was far below that of bagged decision trees. SVMs were also tried with both linear and non-linear kernels. The linear kernel SVM parameters were explored using a coarse grid search and then a higher resolution search around the areas of low RMSE found in the first search. This approach resulted in prediction accuracies comparable to the neural network predictions.

Notes on hardware and software used: A 30 node rocks cluster with 4 CPUs per node and a 6 node rocks cluster with 8 CPUs per node were used to train the ~10,00 Bayesian skill models for the competition and to generate the feature sets. All skills for a KC model could be run in 2 days using the SSI model and 12 hours using the PPS model. Kevin Murphy’s Bayes Net Toolbox for MATLAB was used to construct and train the Bayesian Networks models. One 16 core and one 8 core machine with 32gigs of RAM each were used to run the Random Forests classification using MATLAB’s TreeBagger function. The Parallel Computing Toolbox was used to parallelize the training of the Random forests decision tree classifiers over 8 processor cores. Random forests prediction took 2 to 14 hours depending on the number of trees specified (50-800).

Chapter 5: Individualizing Parameters at the Content Level to Evaluate Individual Item Influences on Learning

Researchers that make tutoring systems would like to know which pieces of educational content are most effective at promoting learning among their students. Randomized controlled experiments are often used to determine which content produces more learning in an ITS. While these experiments are powerful they are often very costly to setup and run. The majority of data collected in many ITS systems consist of answers to a finite set of questions of a given skill often presented in a random sequence. We propose a Bayesian method to detect which questions produce the most learning in this random sequence of data. We confine our analysis to random sequences with four questions. A student simulation study was run to investigate the validity of the method and boundaries on what learning probability differences could be reliably detected with various numbers of users. Finally, real tutor data from random sequence problem sets was analyzed. Results of the simulation data analysis showed that the method reported high reliability
in its choice of the best learning question in 89 of the 160 simulation experiments with seven experiments where an incorrect conclusion was reported as reliable (p < 0.05). In the analysis of real student data, the method returned statistically reliable choices of best question in three out of seven problem sets.

This chapter was published at the following venue:

INTRODUCTION

Researchers that make tutoring systems would like to know which bits of educational content are most effective at promoting learning by students, however a standard method of figuring that out does not exist in ITS, other than by running costly randomized controlled experiments. We present a method that can determine which bits of content are most effective. We believe this method could help other researchers with a variety of different datasets particularly systems that present items in a randomized order. Cognitive Tutor (Koedinger et al., 1997), ANDES (Gертнер & VanLehn, 2000), IMMEX (Stevens, 2006), Mastering Physics and SQL-Tutor (Mitrovic, 2003) are examples of systems that sometime give students a sequence of items in a randomized order and also have vast amounts of data.

In addition to systems typically presented to the AIED audience, traditional Computer Aided Instruction (CAI) systems often have this property of sometimes giving students items of a given skill in a randomized order. For instance, a modern web-based CAI system called studyIsland.com has data of this type from over 1,000 participating schools. The research questions is, can we come up with a method that would allow us to analyze these existing datasets to realize which questions, plus tutorial help in some cases, are most effective at promoting learning.

The intuition for the method exhibited in this chapter is based on the idea that if you consistently see correct answers come after a certain question more than other questions, you may be observing a high learning gain question. While questions of the same skill may differ slightly in difficulty, questions with high difficulty deviation from the mean are likely tapping a different, harder skill as shown in learning factors analysis (Cen, Koedinger & Junker, 2006). We propose to use static Bayesian networks and Expectation Maximization to learn which items cause the most learning. Guess and slip rates will account for question difficulty variation. We will accommodate for all permutations of orderings of the items by building networks for each ordering but will allow the conditional probability tables of each question to be shared across the networks.

SIMULATION

In order to determine the validity of this method we chose to run a simulation study exploring the boundaries of the method’s accuracy and reliability. The goal of the simulation was to generate student responses under various conditions that may be seen in the real world but with the benefit of knowing the underlying best learning question.
Model design

The model used to generate student responses is an eight node static Bayesian network depicted in Figure 1. The top four nodes represent a single skill and the value of the node represents the probability the student knows the skill at each opportunity. The bottom four nodes represent the four questions in the simulation. Student performance on a question is a function of their skill value and the guess/slip of the question. Guess is the probability of answering correctly if the skill is not known. Slip is the probability of answering incorrectly if the skill is known. Learning rates are the probability that a skill will go from “not known” to “known” after encountering the question. The probability of the skill going from “known” to “not known” (forgetting) is fixed at zero. The design of this model is similar to a dynamic Bayesian network or Hidden Markov Model with the important distinction that the probability of learning is able to differ between opportunities. This ability allows us to model different learning rates per question and is key to both the generation of student data in the simulation and analysis using the purposed method.

![Diagram of the simulation network model](image)

**Figure 1.** Simulation network model for a given student with a prior of 0.27 and question sequence 2 4 3 1

While the probability of knowing the skill will monotonically increase after each opportunity, the generated responses will not necessarily do the same since those values are generated probabilistically based on skill knowledge and guess and slip.

Student parameters

Only two parameters were used to define a simulated student; a prior and question sequence. The prior represents the probability the student knew the skill relating to the questions before encountering the questions. The prior for a given student was randomly generated from a beta distribution that was fit to list of skill priors from a previous analysis of real tutor data (Pardos et al., 2008). The mean prior for that year across all skills was 0.31 and the standard deviation was 0.20. The beta distribution fit an α of 1.05 and β of 2.43. The question sequence for a given student was generated from a uniform distribution of sequence permutations.

Tutor Parameters

The 12 parameters of the tutor simulation network consist of four learning rate parameters, four guess parameters and four slip parameters. The number of users simulated was: 100, 200, 500, 1000, 2000, 4000, 10000, and 20000. The simulation was run 20 times for each of the 8 simulated user sizes totaling 160 generated data sets, referred to later as experiments. In order to faithfully simulate the conditions of a real tutor, values for the 12 parameters were randomly generated using the means and standard deviations across 106 skills from a previous analysis of real tutor data (Pardos et al., 2008). In order to produce probabilistic parameter values that fit within 0 and 1, equivalent beta distributions were used. Table 1 shows the distributions that the parameter values were randomly drawn from at the start of each run.
Table 1. The distributions used to generate parameter values in the simulation

<table>
<thead>
<tr>
<th>Parameter type</th>
<th>Mean</th>
<th>Std</th>
<th>Beta dist $\alpha$</th>
<th>Beta dist $\beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
<td>0.086</td>
<td>0.063</td>
<td>0.0652</td>
<td>0.6738</td>
</tr>
<tr>
<td>Guess</td>
<td>0.144</td>
<td>0.383</td>
<td>0.0170</td>
<td>0.5909</td>
</tr>
<tr>
<td>Slip</td>
<td>0.090</td>
<td>0.031</td>
<td>0.0170</td>
<td>0.6499</td>
</tr>
</tbody>
</table>

Running the simulation and generating new parameter values 20 times gives us a good sampling of the underlying distribution for each of the 8 user sizes. This method of generating parameters will end up accounting for more variance than the real world since guess and slip have a correlation in the real world but will be allowed to independently vary in the simulation which means sometimes getting a high slip and high guess, which is rarely observed in actual tutor data.

METHODOLOGY

The simulation consisted of three steps: instantiation of the Bayesian network, setting CPTs to values of the simulation parameters and student parameters and finally sampling of the Bayesian network to generate the students’ responses.

To generate student responses the 8 node network was first instantiated in MATLAB using routines from the Bays Net Toolbox\(^1\) package. Student priors and question sequences were randomly generated for each simulation run and the 12 parameters described in section 1.3 were assigned to the four questions. The placement of the question CPTs were placed with regard to the student’s particular question sequence. The Bayesian network was then sampled a single time to generate the student’s responses to each of the four questions; a zero indicating an incorrect answer and a one indicating a correct answer. These four responses in addition to the student’s question sequence were written to a file. A total of 160 data files were created at the conclusion of the simulation program. Each of these data files were then analyzed by the learning detection method. The analysis method’s accuracy and reliability results for the experiments are summarized in section 3.

Analysis

The purpose of the learning detection method is to calculate the learning rates of questions which are presented in a random sequence and determine which question has the highest learning rate and with what reliability. The simulation study gives us the benefit of knowing what the ground truth highest learning rate question is so we may test the validity of the method’s results.

Model design

The analysis model was based on the same structure as the simulation model, however, the eight node simulation model only needed to represent a single question sequence at a time. The challenge of the analysis model was to accommodate all question sequences in order to learn the parameters of the model over all of the students’ data. In order to accomplish this, 24 eight node networks were created representing all permutations of four question sequences. While the 24 networks were not connected in the Bayesian network’s directed acyclic graph, they are still a part of one big Bayesian network whose parameters are tied together with equivalence classes, discussed in the next sub section.

\(^1\) Kevin Murphy’s Bayes Net Toolbox is available at: http://bnt.sourceforge.net/
Equivalence classes

Equivalence classes allow the 120 CPTs of the 24 networks to be reduced to eight shared CPTs and a single prior. Even though there are 96 (24*4) question nodes in the full network, they still only represent 4 unique questions and therefore there are still only four learning rates to be determined. Equivalence classes tie all of the learning rate CPTs for a given question into a single CPT. They also tie the 96 question guess and slip CPTs in to four CPTs, one per question. In the Bayesian network, the learning rate CPTs for a question is represented in the CPT of the skill node following question. Therefore the learning rate equivalence class for question 2, for instance, is always set in the CPT of the skill node that comes after the skill node for question 2. Question 2’s learning rate equivalence class would appear in 18 of the 24 networks since in 6 of those networks question 2 is the last question in the sequence. The first skill node in a sequence always represents the prior.

Methodology

The analysis method consisted of three steps: splitting the data file into 20 equal parts, loading the data in to the appropriate evidence array location based on sequence ID and then running Expectation Maximization to fit the parameters of the network for each of the 20 parts individually.

The motivation behind splitting the data was to test the reliability of the results among independent groups of student. By counting the number of times the most frequent high learning rate question appears we can compare that to the null hypothesis that each of the four questions is equally likely to have the highest learning rate. This was calculated with a two-tailed binomial probability for hypothesis testing. The binomial is of the “k out of N” type where k is the number of times the most frequent high learning rate question occurred (the mode) and N is the number of samples (20). P is the probability that the outcome could occur by chance. Since the outcome is a selection of one out of four questions, the P value here is 0.25. This binomial p value calculation tells us the probability that the outcome came from the null hypothesis that all questions have an equal chance of being chosen as best. A count of 10 or more would result in a p of < 0.05.

Since the 192 (24*8) node analysis network represented every permutation of question sequences, care had to be taken in presenting the student response evidence to the network. We used the sequence ID from each line of the data file to place the four responses of each student in the appropriate position of the evidence array. Expectation Maximization was then run on the evidence array in order to learn the equivalence class CPTs of the network. Starting points for the EM parameter estimation were set to mean values from previous research (learning rates: 0.08, guess: 0.14, slip: 0.06) with the exception of the prior which was initialized at 0.50.

One of the limitations of our method is that it does not scale gracefully; the number of network nodes that need to be constructed is exponential in the number of items. This is one reason why we did not consider problem sets greater than four. We encourage researchers to investigate ways of scaling this method to large problem sets.

RESULTS

The purpose of the simulation was to provide a means for verifying the validity of the Bayesian learning detection method. While real data was the ultimate goal, the simulation study was necessary to seed ground truth in question learning rates and verify that the method could detect the correct highest learning rate question and that the p value was a good indicator of the believability of the result.
We found that the method reported a reliable (p < 0.05) highest learning rate question in 89 out of the 160 experiments and in 82 of those 89 the reported highest learning rate question was the correct one as set by the simulation (7.8% error). In order to analyze what size learning rate differences the method could detect, the learning rate difference of the simulation’s set highest and second highest learning rates was calculated for each experiment. The minimum learning difference was 0.001 and the max was 0.234. This list of differences was then discretized into four bins corresponding to a learning difference range. The learning ranges were set to achieve equal frequency such that each bin contained 40 experiment results. Bins corresponded to the following learning difference ranges: (0.001-0.0165], (0.0165-0.038], (0.038-0.0715] and (0.0715-0.234). For each range, the percentage of results, with p < 0.05 and a correct question choice, was calculated for each number of simulated users and plotted. The results are exhibited in the plot shown in Figure 2.

![Figure 2. Plot of the frequency of detecting reliable learning rate differences](image)

The plot shows a general increase in the likelihood of a reliable result as the number of users increase. The plot also shows that it was harder to detect smaller learning rate differences than large learning rate differences.

While 20,000 users were required to have a greater than 50% chance of reliably detecting the smallest difference of 0.0010-0.0165, only 500 were needed to detect any of the larger and more common differences with the same chance of a reliable result.

To test how well the method could identify no difference in learning we ran 14 experiments where the learning rates of all questions were set to zero and 14 experiments where the learning rates of all questions were set to 0.08. In these cases where the learning rates were all set the same, the method correctly concluded that there was no reliable best question in 26 of the 28 experiments (7% error).

**ANALYSIS OF REAL TUTOR DATA**

We applied this technique on real student data from our math tutoring system called ASSISTment. High school students ages 16-17 answered problem sets of four math questions at their school’s computer lab two to three times per month. Each problem set was completed in a single day and the sequence of the problems were randomized for each student. Each problem contained hints and scaffolds that students would encounter if they answered the problem incorrectly. The method does not distinguish between the learning value of the scaffold content and the learning value of working through the main problem itself. Only responses to the main problem were considered and answers were only marked as correct if given on the first attempt.
Dataset

Student responses from seven problem sets of four questions each were analyzed. While there are problem sets of different sizes on the system, four is the average size of these problem sets. The problems in a given problem set were chosen by a subject matter expert to correspond to a similar skill. The data was collected during the 2006-2007 school year and the number of users per problem set ranged from 160 to 800. This data from the tutor log file was organized in the same format as the simulation study data files. A sequence ID was also given to each student’s response data indicating what order they saw the questions in.

Results

The analysis calculated a separate learning rate and guess and slip parameter for each of the four questions in the seven problem sets. The mean of the learning rates was 0.081 (similar to the mean used in the simulation) with a standard deviation of 0.035. The mean guess value was 0.18 which was within 1 std of the simulation guess mean, however the mean slip value was unusually high at 0.40. The average number of EM iterations was 95 with many of the runs stopping at the pre-set 100 iteration max.

Table 1. Learning rate results from analysis of student response from problem sets in the ASSISTment tutor

<table>
<thead>
<tr>
<th>Problem set</th>
<th>Users</th>
<th>Best question</th>
<th>p value</th>
<th>prior</th>
<th>q1 rate</th>
<th>q2 rate</th>
<th>q3 rate</th>
<th>q4 rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>800</td>
<td>2</td>
<td>0.0652</td>
<td>0.6738</td>
<td>0.1100</td>
<td>0.1115</td>
<td>0.1017</td>
<td>0.1011</td>
</tr>
<tr>
<td>11</td>
<td>560</td>
<td>4</td>
<td>0.0170</td>
<td>0.5909</td>
<td>0.0958</td>
<td>0.0916</td>
<td>0.0930</td>
<td>0.1039</td>
</tr>
<tr>
<td>14</td>
<td>480</td>
<td>3</td>
<td>0.0170</td>
<td>0.6499</td>
<td>0.1365</td>
<td>0.0977</td>
<td>0.1169</td>
<td>0.1063</td>
</tr>
<tr>
<td>25</td>
<td>440</td>
<td>1</td>
<td>0.0652</td>
<td>0.7821</td>
<td>0.1392</td>
<td>0.0848</td>
<td>0.1157</td>
<td>0.1242</td>
</tr>
<tr>
<td>282</td>
<td>220</td>
<td>1</td>
<td>0.0039</td>
<td>0.7365</td>
<td>0.1574</td>
<td>0.0999</td>
<td>0.0991</td>
<td>0.1004</td>
</tr>
<tr>
<td>33</td>
<td>200</td>
<td>4</td>
<td>0.4394</td>
<td>0.7205</td>
<td>0.1124</td>
<td>0.1028</td>
<td>0.1237</td>
<td>0.1225</td>
</tr>
<tr>
<td>39</td>
<td>160</td>
<td>3</td>
<td>0.0652</td>
<td>0.6180</td>
<td>0.0853</td>
<td>0.1192</td>
<td>0.1015</td>
<td>0.0819</td>
</tr>
</tbody>
</table>

Statistically reliable results were reported in three of the seven problem sets as shown in Table 1. The numbers in the best question column and question learn rate column headers correspond to the IDs that were arbitrarily assigned to the questions.

CONTRIBUTION

We have presented a method that has been validated with a simulation study and shown to provide believable conclusions. While the power of the method could be improved with a different significance test procedure, the algorithm in its current form reports false conclusions less than 8% of the time, roughly in line with a 0.05 p value threshold. This method has broad applicability and can be used by many scientists who have collected responses in a randomized order. We believe researchers could easily adapt this method to identify poor learning content as well as identifying the learning of items that give no tutoring or feedback.

We know of no prior work that has shown how to learn about the effectiveness of a question, other than the typical method of conducting costly randomized controlled experiments. In some aspects, this method seems similar to treating a randomized sequence of items as a set of randomized controlled experiments and could possibly be modified as an approach to a more general problem.

We claim this method could be important, for if we can learn what content is effective at promoting learning, we are one step closer to the elusive dream of building self-improving intelligent tutoring systems that can figure out the most effective material to present to students.
Future Work
A comparison between this Bayesian method of question analysis and an application of learning decomposition (Beck & Mostow, 2008) should be made. Our colleague (Feng & Heffernan, 2009) is pursuing the same research questions as we are, using the learning decomposition method and the same dataset. There is evidence to suggest that a Bayesian method may be the most powerful (Beck et al., 2008) however we would like to confirm this by applying both methods to the same simulated datasets. Our analysis of the effect of item order on learning is in submission (Pardos & Heffernan, 2009a) and makes use of similar modeling techniques that were introduced here.

Chapter 6: Individualizing Parameters at the Content Level to Evaluate Item Order Influences on Learning

The well established, gold standard approach to finding out what works in education research is to run a randomized controlled trial (RCT) using a standard pre-test and post-test design. RCTs have been used in the intelligent tutoring community for decades to determine which questions and tutorial feedback work best. Practically speaking, however, ITS creators need to make decisions on what content to deploy without the luxury of running an RCT. Additionally, most log data produced by an ITS is not in a form that can be evaluated for learning effectiveness with traditional methods. As a result, there is much data produced by tutoring systems that we as education researchers would like to be learning from but are not. In prior work we introduced one approach to this problem: a Bayesian knowledge tracing derived method that could analyze the log data of a tutoring system to determine which items were most effective for learning among a set of items of the same skill. The method was validated by way of simulations. In the current work we further evaluate this method and introduce a second, learning gain, analysis method for comparison. These methods were applied to 11 experiment datasets that investigated the effectiveness of various forms of tutorial help in a web-based math tutoring system. We found that the tutorial help chosen by the Bayesian method as having the highest rate of learning agreed with the learning gain analysis in 10 out of 11 of the experiments. An additional simulation study is presented comparing the statistical power of each method given different sample sizes. The practical impact of this work is an abundance of knowledge about what works that can now be learned from the thousands of experimental designs intrinsic in datasets of tutoring systems that assign items or feedback conditions in an individually-randomized order.

This chapter was published at the following venue:

Additional portions of this chapter are under review at the following venue:

Introduction
Corbett and Anderson style knowledge tracing (Corbett & Anderson, 1995) has been successfully used in many tutoring systems to predict the probability of a student knowing a knowledge component after seeing a set of questions relating to that knowledge component. We present a
method that allows us to detect if the learning value of an item might be dependent on the 
particular context the question appears in. We will model learning rate of an item based on which 
item comes immediately after it. This will allow us to identify rules such as: item A should come 
before B, if such a rule exists. Question A could also be an un-acknowledged prerequisite for 
answering question B. After finding such relationships between questions, a reduced set of 
sequences can be recommended. The reliability of our results is tested with a simulation study in 
which simulated student responses are generated and the method is tasked with learning the 
underlying parameters of the simulation.

We presented a method (Pardos & Heffernan, 2009a) that used similar analysis techniques to 
this one, where an item effect model was used to determine which items produced the most 
learning. That method had the benefit of being able to inform Intelligent Tutoring System (ITS) 
researchers of which questions, and their associated tutoring, are or are not producing learning. 
While we think that method has much to offer, it raised the question of whether the learning value 
of an item might be dependent on the particular context it appears in. The method in this chapter 
is focused on learning based on item sequence.

The Tutoring System

Our datasets consisted of student responses from ASSISTments, a web based math tutoring 
system for 7th-12th grade students that provides preparation for the state standardized test by using 
released math items from previous state tests as questions on the system. Figure 1 shows an example of 
a math item on the system and tutorial help that is given if the student answers the question wrong or asks for 
help. The tutorial help assists the student in learning the required knowledge by breaking each problem into sub 
questions called scaffolding or giving the student hints on how to solve the question. A question is only marked 
as correct if the student answers it correctly on the first attempt without requesting help.

Item templates in ASSISTments

Our mastery learning data consists of responses to multiple questions generated from an item template. A 
template is a skeleton of a problem created by a content developer in our web based builder application. For 
example, the template would specify a Pythagorean Theorem problem, but without the numbers for the 
problem filled in. In this example the problem template could be: “What is the hypotenuse of a right triangle 
with sides of length X and Y?” where X and Y are variables that will be filled in with values when 
questions are created from the template. The solution is also dynamically determined from a solution template 
specified by the content developer. In this example the solution template would be, “Solution = 
sqrt(X^2+Y^2)”. Ranges of values for the variables can be specified and more advance template 
features are available to the developer such as dynamic graphs, tables and even randomly selected 
cover stories for word problems. Templates are also used to construct the tutorial help of the

Figure 1. An example of an 
ASSISTment item where the 
student answers incorrectly and is 
given tutorial help.

sqrt(X^2+Y^2)". Ranges of values for the variables can be specified and more advance template 
features are available to the developer such as dynamic graphs, tables and even randomly selected 
cover stories for word problems. Templates are also used to construct the tutorial help of the
template items. Items created from these templates are used extensively in the mastery learning problem sets as a pragmatic way to provide a high volume of items for students to practice a particular skill on.

Knowledge Tracing

The Corbett and Anderson method of “knowledge tracing” (Corbett & Anderson, 1995) has been useful to many intelligent tutoring systems. In knowledge tracing there is a set of questions that are assumed to be answerable by the application of a particular knowledge component which could be a skill, fact, procedure or concept. Knowledge tracing attempts to infer the probability that a student knows a knowledge component based on a series of answers. Presumably, if a student had a response sequence of 0,0,1,0,0,1,1,1,1,1,1 where 0 is an incorrect first response to a question and 1 is a correct response, it is likely she guessed the third question but then learned the knowledge to get the last 6 questions correct. The Expectation Maximization algorithm is used in our research to learn the parameters of our model from data, the probability of learning from a particular item order being one such parameter.

![Figure 2. Knowledge Tracing model for question sequence (2 1 3)](image)

Figure 2 depicts a typical knowledge tracing three question static Bayesian network. The top three nodes represent a single skill and the inferred value of the node represents the probability the student knows the skill at each opportunity. The bottom three nodes represent three questions on the tutor. The four parameters of a standard knowledge tracing model are prior, learn, guess and slip. Student performance on a question is a function of their skill knowledge and the guess and slip of the question. Guess is the probability of answering correctly if the skill is not known. Slip is the probability of answering incorrectly if the skill is known. Learning rate is the probability that a skill will go from “not known” to “known” between each opportunity. The probability of the skill going from “known” to “not known” (forgetting) is fixed at zero. Knowledge tracing assumes that there is the same probability of learning at each opportunity regardless of the particular question being solved or help being given. This is shown in Figure 2’s learn rates of 0.08, that remain the same. The basic design of our model is similar to a dynamic Bayesian network or Hidden Markov Model used in knowledge tracing but with the distinction that the probability of learning is able to differ between opportunities based on item order. For example, in Figure 2, the first learn rate of 0.08 would be associated with item order [2 1], and the second learn rate of 0.08 could be different and would be associated with item order [1 3]. This modification allows us to model different learning rates per question or per question order.

THE ITEM ORDER EFFECT MODEL

In the model we call the item order effect model we look at what effect item order has on learning. We set a learning rate for each pair of items and then test if one pair is reliably better for learning than another. For instance, should question A come before question B or vice versa? With our three item problem sets there will be six ordered pairs which are (3,2) (2,3) (3,1) (1,3)
(2,1) and (1,2). This model allows us to train the learning rates of all six ordered pairs simultaneously along with guess and slip for the questions by using shared parameters to link all occurrences of pairs to the same conditional probability table. For example, the ordered pair (3,2) appears in two sequence permutations; sequence (3,2,1) and sequence (1,3,2) as shown in Figure 3. Student response from both sequences would be used to fit the learn rate for (3,2).

The model in Figure 3 demonstrates how the CPT sharing works and how the Item Effect Model from (Pavlik, Presson, Koedinger, 2007) was constructed. That model required every sequence permutation of items to be represented. For a three item problem set, six sequence permutations (N * N – N) are needed. Six is computationally tractable, however the required sequences grows exponentially. For a seven item problem set, the number of sequence permutations jumps to 5,040. This made the model not scalable to larger problem sets, such as mastery learning problem sets or longer problem sets in other tutors such as the cognitive tutor. In the next section we introduce an improved model that scales gracefully.

**Improvements to the Item Order Effect Model**

The basic requirements of the item order effect model is that each ordered item pair have a single learn rate and each item have a single guess and slip. Modeling each permutation and sharing the CPTs accomplished this but was not scalable or efficient. Another implementation was architected that meets the requirements and does not have a memory requirement that grows exponentially with the length of the problem set. This new approach was accomplished by using conditional nodes; multi value nodes that identify the question or item sequence being observed. This is similar to the technique used to individualize parameters (Pardos & Heffernan, 2010a). All possible two item orders for a problem set are enumerated and given an order ID. This order ID is stored in a lookup table as shown in Table 1.

**Table 1. Item order ID lookup table**

<table>
<thead>
<tr>
<th>Item order</th>
<th>(1, 1)</th>
<th>(1, 2)</th>
<th>(1, 3)</th>
<th>(2, 1)</th>
<th>(2, 2)</th>
<th>(2, 3)</th>
<th>(3, 1)</th>
<th>(3, 2)</th>
<th>(3, 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order ID</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

Item pair (3,2)'s learning rate is the probability that if the skill was not known at question three it will be known at question two. This is the probability of learning the skill. The five other item pairs have their own CPTs in the full network.
In addition to a response sequence, each student has an item sequence and an order sequence that is constructed from the item sequence and order lookup table. An example of student sequence data is shown in Table 2 and is represented in the new model in Figure 4. There are five CPTs in total in this model and the same number of free parameters as the old model. There is one CPT for the question sequence nodes, one for the order sequence nodes, one for the skill nodes (except the first), one for the first skill node (prior) and one for the question nodes. The guess and slip rates of all the questions are stored in a single CPT since the question node is conditioned on the question sequence node. The CPT for the question node would look like Table 3.

All the learn rates of the various item order pairs are also stored in the same CPT. In this model every student has her own question, order and response sequence that is presented as evidence. Evidence for all students is used by EM to fit the guesses, slips, prior and item order learning rates. Compared to the original model, this model is substantially faster and significantly more accurate at recovering the true parameters as detailed in the simulation section.

**Reliability Estimates Using the Binomial Test**

In order to derive the reliability of the learning rates fit from data we employed a binomial test by randomly splitting the response data into 10 by student. We fit the model parameters using data from each of the 10 bins separately and counted the number of bins in which the learning

---

2 The binomial test was run with the MATLAB command: binopdf(successes, trials, 1/outcomes)
rate of one item pair was greater than its reverse, (3,2) > (2,3) for instance. We call a comparison of learning rates such as (3,2) > (2,3) a rule. The null hypothesis is that each rule is equally likely to occur. A rule is considered statistically reliable if the probability that the result came from the null hypothesis is <= 0.05. For example, if we are testing if ordered pair (3,2) has a higher learning rate than (2,3) then there are two possible outcomes and the null hypothesis is that each outcome has a 50% chance of occurring. Thus, the binomial test will tell us that if the rule holds true eight or more times out of ten then it is <= 0.05 probable that the result came from the null hypothesis. This is the same idea as flipping a coin 10 times to determine the probability it is fair. The less likely the null hypothesis, the more confidence we can have that the coin is not fair, or in our case, that the item order is reliably better at producing learning. If the learning rate of (3,2) is greater than (2,3) with p <= 0.05 then we can say it is statistically reliable that question three and its tutoring followed by question two better helps students learn the skill than question two and its tutoring followed by question three. Based on this conclusion it would be recommended to give sequences where question three comes before two.

An Item order can also be tested for having a reliably higher learning rate than the other item orders by calculating 1-binocdf(N-1, T, 1/O) where N is the number of bins that the order learning rate was higher than all the others, T is the total number of bins and O is the total number of item orders. If 1-binocdf(N-1, T, 1/O) is <= 0.05 then the item order has a learn rate that is reliably high, compared to all the others in the problem set. Multiple reliably high orders are possible in a single problem set.

DATASETS

The fixed length problem set dataset we analyzed was from the 2006-2007 school year. Subject matter experts made problem sets called GLOPS (groups of learning opportunities). The idea behind the GLOPS was to make a problem set where the items in the problem set related to each other. They were not necessary strictly related to each other through a formal skill tagging convention but were selected based on their similarity of concept according to the expert. We chose the five three item GLOPS that existed in the system each with between 295 and 674 students who had completed the problem set. Items do not overlap across GLOP problem sets. The items in the five problem sets were presented to students in a randomized order. Randomization was not done for the sake of this research in particular but rather because the assumption of the subject matter expert was that these items did not have an obvious progression requiring that only a particular sequence of the items be presented to students. In other words, context sensitivity was not assumed. We only analyzed responses to the original questions which meant that a distinction was not made between the learning occurring due to answering the original question and learning occurring due to the help content. The learning from answering the original question and scaffolding will be conflated as a single value for the item.

Our variable length dataset consists of student responses to ten mastery learning problem sets in ASSISTments from the 2009-2010 school year. These ten mastery learning problem sets were the problem sets with the most data in the system that contained two or more templates. The mastery sets are also referred to as “Skill Building” sets in the tutor and are designed to give students practice on a particular skill. Each problem set has a different skill topic and the problem set is filled with problems from two or more templates that relate to the skill being taught. Students must get three questions correct in a row (without asking for a hint) in order to complete the problem set. A random template is chosen for the student’s next problem. This could result in the same template being presented to the student consecutively which is why an item order such as (3,3) is tracked. In this analysis, the templates will be treated like items and the learning rate of pairs of template orders will be learned. Students can answer a maximum of ten questions per day in any one mastery problem set. We decided to only consider a student’s first day of responses to a problem set in this analysis.
**ITEM ORDER EFFECT RESULTS**

We ran the analysis method on our problem sets and found reliable rules in two out of the five fixed length problem sets. In the mastery learning problem sets we found reliable ordering effects in four out of the ten.

**Order effect results in fixed length problem sets**

The results below show the item pair learning rate parameters for the two problem sets in which reliable rules were found. The 10 bin split was used to evaluate the reliability of the rules while all student data for the respective problem sets were used to train the parameters shown below.

<table>
<thead>
<tr>
<th>Problem Set</th>
<th>Users</th>
<th>Learning probabilities of Item Pairs</th>
<th>Reliable Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(3,2)</td>
<td>(2,1)</td>
</tr>
<tr>
<td>24</td>
<td>403</td>
<td>0.1620</td>
<td>0.0948</td>
</tr>
<tr>
<td>36</td>
<td>419</td>
<td>0.1507</td>
<td>0.1679</td>
</tr>
</tbody>
</table>

As shown in Table 4, there was one reliable rule found in each of the problem sets. In problem set 24 we found that item pair (3,2) showed a higher learning rate than (2,3) in eight out of the 10 splits giving a binomial p of 0.0439. Item pair (1,3) showed a higher learning rate than (3,1) also in eight out of the 10 splits in problem set 36. Other statistically reliable relationships can be tested on the results of the method. For instance, in problem set 36 we found that (2,1) > (3,1) in 10 out of the 10 bins. This could mean that sequence (3,1,2) should not be given to students because question three comes before question one and question two does not. Removing sequence (3,1,2) is also supported by rule (1,3) > (3,1). In addition to the learning rate parameters, the model simultaneously trains a guess and slip value for each question. Those values are shown below in Table 5.

<table>
<thead>
<tr>
<th>Question #</th>
<th>Problem Set 24</th>
<th></th>
<th>Problem Set 36</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Guess</td>
<td>Slip</td>
<td>Guess</td>
<td>Slip</td>
</tr>
<tr>
<td>1</td>
<td>0.17</td>
<td>0.18</td>
<td>0.33</td>
<td>0.13</td>
</tr>
<tr>
<td>2</td>
<td>0.31</td>
<td>0.08</td>
<td>0.31</td>
<td>0.10</td>
</tr>
<tr>
<td>3</td>
<td>0.23</td>
<td>0.17</td>
<td>0.20</td>
<td>0.08</td>
</tr>
</tbody>
</table>

**Order effect results in mastery learning problem sets**

The results in Table 6 show the four mastery problem sets where reliable rules were detected. Each problem template is briefly described and the reported rules represent reliable orderings for those templates. For example, a rule of (2,1) means that a student who receives template 2 followed by template 1 has a reliably greater chance of learning than a student who receives the templates in the reverse order. Screen caps of questions from each of the four “Integer subtraction” templates can be found in the Appendix A. Rules (2,1) and (3,1) from “Integer subtraction” were almost significant in another problem set “Integer addition”, suggesting that these orderings may generalize. For the fractions problem set, the best order was found to be template 4 followed by template 4. This says that answering the percent to fraction template twice is significantly more beneficial than any other two template orders in acquiring the knowledge that leads to positive performance on the templates in that problem set.
### Table 6. Mastery learning order effect results

<table>
<thead>
<tr>
<th>Problem set skill name</th>
<th>Number of students</th>
<th>Template # and description</th>
<th>Rules (template #, template #) Significant best order</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fraction/pct./dec. conv.</strong></td>
<td>1019</td>
<td>1 fraction → percent</td>
<td>Rules: (1,2) (1,5) (4,3) (6,4) (6,5)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 decimal → percent</td>
<td>Sig. best order: (4,4) lrate=0.513</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 percent → decimal</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 percent → fraction</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 percent → fraction [slightly longer hints]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 percent → decimal [percent has decimal, ex. 70.34%]</td>
<td></td>
</tr>
<tr>
<td><strong>Absolute value</strong></td>
<td>877</td>
<td>1</td>
<td>(+ or -) x</td>
</tr>
<tr>
<td><strong>Dec. to fraction conv.</strong></td>
<td>920</td>
<td>2</td>
<td>x-y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td><strong>Integer subtraction</strong></td>
<td>1024</td>
<td>1 positive - negative</td>
<td>Rules: (2,1) (3,1) (3,4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 negative - positive</td>
<td>Sig. best order: (2,1) lrate=0.142</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 negative - negative</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 positive - positive</td>
<td></td>
</tr>
</tbody>
</table>

### SIMULATION

In order to determine the validity of the item order effect method we chose to run a simulation study exploring the boundaries of the method’s accuracy and reliability. The goal of the simulation was to generate student responses under various conditions that may be seen in the real world and test if the method would accurately infer the underlying parameter values from the simulated student data. This simulation model assumes that learning rates have distinct values and that item order effects of some magnitude always exist and should be detectable given enough data.

#### Model design

The model used to generate student responses is a six node static Bayesian network similar to the one depicted in Figure 2 from section 1.2. While the probability of knowing the skill will monotonically increase after each opportunity, the generated responses (0s and 1s) will not necessarily do the same since those values are generated probabilistically based on skill knowledge and guess and slip. Simulated student responses were generated one student at a time by sampling from the six node network.

#### Student parameters

Only two parameters were used to define a simulated student, a prior and question sequence. The prior represents the probability the student knew the skill relating to the questions before encountering the questions. The prior for a given student was randomly generated from a distribution that was fit to a previous year’s ASSISTment data (Pardos & Heffernan, 2010a). The mean prior for that year across all skills was 0.31 and the standard deviation was 0.20. In order to draw probabilistic parameter values that fit within 0 and 1, an equivalent beta distribution was used. The beta distribution fit an α of 1.05 and β of 2.43. The question sequence for a given student was generated from a uniform distribution of sequence permutations.
Tutor Parameters

The 12 parameters of the tutor simulation network consist of six learning rate parameters, three guess parameters and three slip parameters. The number of users simulated was: 200, 500, 1000, 2000, 4000, 10000, and 20000. The simulation was run 20 times for each of the seven simulated user sizes totaling 140 generated data sets, referred to later as experiments. In order to faithfully simulate the conditions of a real tutor, values for the 12 parameters were randomly generated using the means and standard deviations across 106 skills from a previous analysis (Pardos et al., 2008) of ASSISTments data. Table 7 shows the distributions that the parameter values were randomly drawn from and then assigned to questions and learning rates at the start of each run.

Table 7. The distributions used to generate parameter values in the simulation

<table>
<thead>
<tr>
<th>Parameter type</th>
<th>Mean</th>
<th>Std</th>
<th>Beta dist α</th>
<th>Beta dist β</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
<td>0.086</td>
<td>0.063</td>
<td>8.1492</td>
<td>82.7107</td>
</tr>
<tr>
<td>Guess</td>
<td>0.144</td>
<td>0.383</td>
<td>1.0511</td>
<td>2.4256</td>
</tr>
<tr>
<td>Slip</td>
<td>0.090</td>
<td>0.031</td>
<td>12.5325</td>
<td>74.6449</td>
</tr>
</tbody>
</table>

Running the simulation and generating new parameter values 20 times gives us a good sampling of the underlying distribution for each of the seven user sizes. This method of generating parameters will end up accounting for more variance than the real world since standard deviations were calculated for values across problem sets as opposed to within. Also, guess and slip have a correlation in the real world but will be allowed to independently vary in the simulation which means sometimes getting a high slip but low guess, which is rarely observed in actual ASSISTments data. It also means the potential for generating very improbable combinations of item pair learning rates.

Simulation Procedure

The simulation consisted of three steps: instantiation of the Bayesian network, setting CPTs to values of the simulation parameters and student parameters and finally sampling the Bayesian network to generate the students’ responses.

To generate student responses the six node network was first instantiated in MATLAB using routines from the Bayes Net Toolbox package. Student priors and question sequences were randomly generated for each simulation run and the 12 parameters described in section 3.3 were assigned to the three questions and item pair learning rates. The question CPTs and learning rates were positioned with regard to the student’s particular question sequence. The Bayesian network was then sampled a single time to generate the student’s responses to each of the three questions; a zero indicating an incorrect answer and a one indicating a correct answer. These three responses in addition to the student’s question sequence were written to a file. A total of 140 data files were created at the conclusion of the simulation runs, all of which were to be analyzed by the item order effect detection method. The seeded simulation parameters were stored in a log file for each experiment to later be checked against the method's findings. An example of an experiment’s output file for 500 users is shown in Table 8 below.

Table 8. Example output from data file with N=500

<table>
<thead>
<tr>
<th>Simulated User</th>
<th>Sequence identifier</th>
<th>1st Q</th>
<th>2nd Q</th>
<th>3rd Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>500</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Each data file from the simulation was split into 10 equal parts and each run separately through the analysis method just as was done in analysis of real tutor data. This analysis step would give a result such as the example in Table 9 below.

<table>
<thead>
<tr>
<th></th>
<th>(3,2)</th>
<th>(2,1)</th>
<th>(3,1)</th>
<th>(1,2)</th>
<th>(2,3)</th>
<th>(1,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Split 1</td>
<td>0.0732</td>
<td>0.0267</td>
<td>0.0837</td>
<td>0.0701</td>
<td>0.0379</td>
<td>0.642</td>
</tr>
<tr>
<td>Split 10</td>
<td>0.0849</td>
<td>0.0512</td>
<td>0.0550</td>
<td>0.0710</td>
<td>0.0768</td>
<td>0.0824</td>
</tr>
</tbody>
</table>

In order to produce a p value and determine statistical reliability to the p < 0.05 level the binomial test is used. The method counts how many times (3,2) was greater than (2,3) for instance. If the count is greater than eight then the method considers this an identified rule. Even though there are six item pairs there is a maximum of three rules since if (3,2) > (2,3) is a reliable rule then (3,2) < (2,3) is not. In some cases finding two rules is enough to identify a single sequence as being best. Three rules always guarantee the identification of a single sequence. The method logs the number of rules found and how many users (total) were involved in the experiment. The method now looks "under the hood" at the parameters set by the simulation for the item pair learning rates and determines how many of the found rules were false. For instance, if the underlying simulated learning rate for (3,2) was 0.08 and the simulated learning rate for (2,3) was 0.15 then the rule (3,2) > (2,3) would be a false positive (0.08 < 0.15). This is done for all 140 data files. The total number of rules is three per experiment thus there are 420 rules to be found in the 140 data files.

**Simulation Results**

The average percent of found rules per simulated user size is plotted in Figure 5 below. The percentage of false positives is also plotted in the same figure and represents the error.

![Figure 5. Results of simulation study](image-url)
Figure 5 shows that more users allows for more rules about item order to be detected. It also shows that the false positive rate remains fairly constant, averaging around the 6% mark. From 200 users to 1,000 users the average percentage rules found was around 30% which would correspond to about 1 rule per problem set (0.30 * 3). This percentage rises steadily in a linear fashion from 500 users up to the max number of users tested of 20,000 where it achieves a 69% discovery rate which corresponds to about two rules per problem set on average. The error starts at 13% with 200 users and then remains below 10% for the rest of the user sizes. The overall average percent of rules found across users sizes is 43.3%. The overall average false positive rate is 6.3% which is in line with the binomial p value threshold of 0.05 that was used and validates the accuracy of the method's results and dependability of the reported binomial p value.

**New model vs. original model performance**

Improvements to the Item Order Effect Model have been introduced in this chapter, allowing it to scale to larger problem sets. The behavior of this new model was tested against the original model by running the new model on the same simulation data and comparing its performance against the original model which was used to produce the result in the previous sub section. True positives and false positives of each model are plotted in Figure 6.

![Figure 6. New model vs. original model true positive and false positive rate by number of users](image)

The result showed that the conclusions drawn by the new model are very similar to that of the original model. This is a positive sign since it indicates that the item length limitation has been addressed without loss of statistical power. The similarity of the two method’s learn rate estimates were compared using a paired t-test. The simulation consisted of 140 experiments with 6 item order learning rates each totaling 840 learn rate estimates. The learn rates of each method were compared and the paired t-test concluded that the two are reliably different from one another. The mean of the 840 estimated learn rates for the original model was 0.7920, while the mean of the new model’s learn rates was 0.0848. The linear correlation coefficient of the two models estimates was 0.9840, indicating that the learn rate estimates of the two models are highly correlated, despite having statistically reliably different means. The new model made similar conclusions to the original model, as show in Figure 6, but with a slightly increased mean distribution. In order to determine which model’s learn rate estimates were more accurate, we compared to the ground truth learn rates of the simulation. The mean of the true learn rates was
0.0863. A paired t-test between the true estimates and the original model gave a p = 0.0017, suggesting that the original model estimates’ mean is reliably different than the true mean. A paired t-test between the true estimates and the new model gave a p = 0.5650, suggesting that the new model estimates’ mean is not reliably different than the true mean. Because of these results we believe we can claim that the new model is more accurate than the original model.

We also found that an additional benefit of the new model is substantially increased efficiency in terms of compute time. The times to run each of the 140 experiment datasets was recorded for each of the models. The time for each model increased linearly with the number of simulated students. The compute time was divided by the number of simulated students for each model and experiment. The result was a calculated average of 0.7038 seconds per student for the original model and 0.0212 seconds per student for the new model. This amounts to a dramatic speedup of 33x over the original model. With this efficiency increase, calculation of effects for large problem sets with thousands of users is very tractable.

**Future Work**

The split 10 procedure has the effect of decreasing the amount of data the method has to operate on for each run. A more efficient sampling method may be beneficial, however, our trials using resampling with replacement for the simulation instead of splitting resulted in a high average false positive rate (>15%). A more sensitive test that takes into account the size of the difference between learned parameter values would improve reliability estimates. The binomial accuracy may also be improved by using a Bonferroni correction as suggested by a reviewer. This correction is used when multiple hypotheses are tested on a set of data (i.e. the reliability of item ordering rules). The correction suggests using a lower p value cut-off.

There is a good deal of work in the area of trying to build better models of what students are learning. One approach (Barnes, 2005) uses a matrix of skill to item mappings which can be optimized (Cen, Koedinger, Junker, 2006) for best fit and used to learn optimal practice schedules (Pavlik, Presson, Koedinger, 2007) while another approach attempts to find item to item knowledge relationships (Desmarais, Meshkinfam, Gagon, 2006) such as prerequisite item structures using item tree analysis. We think that the item order effect method introduced here and its accompanying chapter (Pardos & Heffernan, 2009a) have parallels with these works and could be used as a part of a general procedure to try to learn better fitting models.

**CONTRIBUTION**

We have made improvements to the model that address the scalability issue of the original. In addition to being 33x more computationally efficient, the new model was also shown to have significantly more accurate learning rate estimates than the original model. This method has been shown by simulation study to provide reliable results suggesting item orderings that are most advantageous to learning. Many educational technology companies (i.e. Carnegie Learning Inc. or ETS) have hundreds of questions that are tagged with knowledge components. We think that this method, and ones built off of it, will facilitate better tutoring systems. In (Pardos & Heffernan, 2009a) we used a variant of this method to figuring out what items are causing the most learning. In this chapter, we presented a method that allows scientists to see if the items in a randomly ordered problem set produce the same learning regardless of context or if there is an implicit ordering of questions that is best for learning. Those best orderings might have a variety of reasons for existing. Applying this method to investigate those reasons could inform content authors and scientists on best practices in much the same way as randomized controlled experiments do but by using the far more economical means of data mining.
ACKNOWLEDGEMENTS

We would like to thank the Worcester Public Schools and the people associated with creating ASSISTments listed at www.ASSISTments.org including investigators Kenneth Koedinger and Brian Junker at Carnegie Mellon and also Dave Brown and Carolina Ruiz at Worcester Polytechnic Institute for their suggestions. We would also like to acknowledge funding from the U.S. Department of Education’s GAANN and IES grants, the Office of Naval Research, the Spencer Foundation and the National Science Foundation.

Chapter 7: Using Item Individualized Parameters to Evaluate the Effectiveness of Different Types of Tutoring Interventions

The well established, gold standard approach to finding out what works in education research is to run a randomized controlled trial (RCT) using a standard pre-test and post-test design. RCTs have been used in the intelligent tutoring community for decades to determine which questions and tutorial feedback work best. Practically speaking, however, ITS creators need to make decisions on what content to deploy without the luxury of running an RCT. Additionally, most log data produced by an ITS is not in a form that can be evaluated for learning effectiveness with traditional methods. As a result, there is much data produced by tutoring systems that we as education researchers would like to be learning from but are not. In prior work we introduced one approach to this problem: a Bayesian knowledge tracing derived method that could analyze the log data of a tutoring system to determine which items were most effective for learning among a set of items of the same skill. The method was validated by way of simulations. In the current work we further evaluate this method and introduce a second, learning gain, analysis method for comparison. These methods were applied to 11 experiment datasets that investigated the effectiveness of various forms of tutorial help in a web-based math tutoring system. We found that the tutorial help chosen by the Bayesian method as having the highest rate of learning agreed with the learning gain analysis in 10 out of 11 of the experiments. An additional simulation study is presented comparing the statistical power of each method given different sample sizes. The practical impact of this work is an abundance of knowledge about what works that can now be learned from the thousands of experimental designs intrinsic in datasets of tutoring systems that assign items or feedback conditions in an individually-randomized order.

This chapter was published at the following venue:

INTRODUCTION

The well-established, gold standard approach to finding out what works in an intelligent tutoring system is to run a randomized controlled trial (RCT) using a standard pre-test and post-test design. RCTs have been used in the intelligent tutoring systems (ITS) community for decades to determine best practices for a particular context. Practically speaking, however, RCTs are often only run once, without replication, and the conditions are often only applied to one context or skill. Lack of replication raises questions about the significance of the findings (Ioannidis, 2005) and limiting the scope of the experiment to only a single context or skill, as is often the case, results in uncertainty about the generality of the intervention. More data is necessary to address these concerns. Gathering data via RCTs is costly, both in terms of financing and class time. In a
typical RCT design, students are not receiving feedback on pre and post-tests which often makes the experiments less desirable to teachers. While data collected from RCTs are sparse, log data collected from Intelligent Tutoring Systems are abundant and the rate of collection is increasing daily. This treasure trove of data is the next frontier in researching pedagogical effectiveness. With these ITS providing a steady stream of data, what is needed to learn from the data is 1) randomization of item order or feedback in the system where possible and 2) reliable tools to analyze the data. Thusly, we propose that the approach to solving the experimentation issues is a data mining approach. In prior work (Pardos & Heffernan, 2009a) we introduced a model to address this problem: a Bayesian networks method that analyzes the log data of a tutoring system to determine which items are most effective for learning among a set of items of the same skill. In this work we apply this method to a variety of different forms of tutor log data and introduce an alternative, learning gains based method for comparison.

ASSISTments – a web-based tutoring system and research platform

Our datasets consisted of student responses from ASSISTments (Razzaq et al, 2005), a web based math tutoring platform that is best known for its 4th-12th grade math content. Figure 1 shows an example of a math item on the system and tutorial help that is given if the student answers the question wrong or asks for help. The tutorial help assists the student in learning the required knowledge by breaking each problem into sub questions called scaffolding or giving the student hints on how to solve the question. A question is only marked as correct if the student answers it correctly on the first attempt without requesting help.

**Item templates in ASSISTments**

Our skill building data consists of responses to multiple questions generated from an item template. A template is a skeleton of a problem created by a content developer in our web based builder application. For example, the template would specify a Pythagorean Theorem problem, but without the numbers for the problem filled in. In this example the problem template could be: “What is the hypotenuse of a right triangle with sides of length X and Y?” where X and Y are variables that will be filled in with values when questions are created from the template. The solution is also dynamically determined from a solution template specified by the content developer. In this example the solution template would be, “Solution = sqrt(X^2+Y^2)”. Ranges of values for the variables can be specified and more
Advance template features are available to the developer such as dynamic graphs, tables and even randomly selected cover stories for word problems. Templates are also used to construct the tutorial help of the template items. Items created from these templates are used extensively in the skill building problem sets as a pragmatic way to provide a high volume of items for students to practice particular skills on. While these templates could be tested for their educational effectiveness, this chapter focuses on evaluating the different feedback of each template.

**ANALYSIS MODEL**

**Knowledge Tracing**

The Bayesian model we used in our analysis is based on Knowledge tracing (Corbett & Anderson, 1995; Rye, 2004) which has been the predominant modeling technique used in tracking student knowledge in intelligent tutoring systems for over a decade. Knowledge tracing is based on a Hidden Markov Model representation of knowledge with the observed nodes representing correct or incorrect answers to questions of a particular skill and the latent representing whether or not the student is in the learned or unlearned state. The four parameters of knowledge tracing are: prior, guess, slip and learn rate. The prior is the probability that students already knew the skill before answering questions on the tutor. The guess is the probability of a student answering correctly even if he is in the unlearned state. Slip is the probability of answering incorrectly even if he is in the learned state. The learn rate is the probability of transitioning from unlearned to learned after each item. With these four parameters, Knowledge tracing can be used to predict the probability that a student knows the skill at hand given his or her past sequence of incorrect or correct responses to questions of the same skill.

The procedure for fitting these four parameters varies. Exhaustive search and Expectation Maximization (EM) are the two most common ways to fit the knowledge tracing parameters to data. Recent work comparing the predictive accuracy of the two fitting methods (Gong, Beck & Heffernan, In Press) showed mixed results with EM leading to more accurate prediction of student performance but exhaustive search leading to higher correlation of the prior knowledge parameter to student pre-test scores. While exhaustive search, at some precision level, can be tractable for the four parameter case, it becomes less practical as the number of parameters in a model increases. In our current work we use the EM algorithm due to the increased number of parameters in our model.

**The Item Effect Model**

The basic Knowledge tracing model assumes that the probability of learning on each opportunity, or piece of learning content, is the same. That is to say that the model makes a simplifying assumption that a student has the same probability of learning from every piece of tutoring, regardless of quality. Given that RCTs have shown that different interventions can cause significant differences in learning, we have reason to challenge this assumption. The Item Effect model breaks from this assumption and allows the learning rate between each opportunity to differ based on the item that the student just attempted to solve or tutoring that the student encountered. This is accomplished by associating each item or feedback condition with its own learning rate parameter. The Item Effect model also allows for each item to have its own guess and slip rate as opposed to Knowledge tracing, which has one guess and slip parameter for each skill. This, in essence, models the differing difficulty of items and takes that in to account when fitting the learn rates. The fact that items are related to the same skill is captured by their connection in the same Bayesian network. Knowledge tracing is a special case of the Item Effect
model, in which each question can cause different amounts of learning. We stick with the assumption used in Knowledge tracing that assumes the probability of forgetting to be zero but fundamentally neither KT nor Item Effect are limited to this assumption.

The Item Effect Model, depicted at the bottom left of Figure 2, allows for a learning rate as well as a guess and slip rate per item, to be learned. The guess rate is the probability that a student will answer an item correctly even if they do not know the skill involved. The slip rate is the probability that a student will answer an item incorrectly even if they know the skill involved. Items of the same skill are connected via the latent variable of knowledge. The probability of forgetting is set to zero so the probability of knowledge can only stay the same or increase with each opportunity the student has to answer an item of a particular skill.

Figure 2. An example of the difference between Knowledge tracing and the Item Effect Model including the models’ knowledge and performance parameters.

The Item Effect Model looks for when a student is believed to have transitioned from the unlearned state to the learned state, which is generally indicated by incorrect answers followed by correct answers to a series of items. Intuitively, the model credits the last item answered incorrectly as most probably causing this learning. If the model observes a pattern of learning that frequently occurs after a particular item, that item is attributed with a higher learn rate than the other items in the problem set being considered. The probabilities of learning associated with the items are relative to the other items in the problem set and are indicative of a particular item’s ability to cause positive performance on the other items as determined by the parameter fitting procedure.

Due to the fashion in which the Item Effect Model looks for patterns of learning, it requires that either the feedback conditions or the items in a problem set be assigned to students in an individually-randomized order. The Item Effect Model in fact models every permutation of a set of items. This means that when analyzing a 3 item problem set, all 6 permutations of sequences are modeled. Randomization is required much in the same way that an RCT requires randomization in the assignment of conditions. Neither hypothesis testing approach can identify
learning effects from a single linear sequence without randomization at some level. Item learning rates from a linear sequence would likely be the result of an ordering effect. Ordering effects can also be detected by the Item Effect Model by associating learning rates to item pairs (Pardos & Heffernan, 2009b).

In order to determine the statistical reliability of differences in learning rates, the data is randomly split into 10 bins of equal size. The method is run on each of the 10 bins and the probability of one item out of M items having a higher learning rate in N or more bins is determined by the binomial function: 1 - binocdf(N-1,10,1/M). In this chapter an N ≥ 8 (p=0.0547) will be considered statistically significant. The binomial value is very similar to a chi-squared test. The p value result from a chi-squared test of two conditions with an outcome of 8:2 where an equal distribution is expected is 0.0578. Given a large enough sample size, the number of bins can be increased to give greater p value precision. A rule of thumb is to have at least 30 data points in each bin.

**ANALYZING EXPERIMENTS IN SKILL BUILDING CONTENT**

Skill building is a type of problem set in ASSISTments that consists of items, often from a number of different templates, all pertaining to the same skill. Students are marked as having completed the problem set when they answer three items correctly in a row without asking for help. In these problem sets items are selected in a random order and students spend a lot of time in this type of instruction making the data produced from these problem sets especially lucrative to mine. The skill building problem sets are similar in nature to mastery learning (Corbett, 2001) in the Cognitive Tutors (Koedinger, Anderson, Hadley & Mark, 1997). However, in the Cognitive Tutors mastery is achieved when a knowledge-tracing model believes that the student knows the skill with 0.95 or better probability. Much like the other problem sets in ASSISTments, skill builder problem sets are assigned by the teacher at his or her discretion and the problem sets they assign often conform to the particular math curriculum their district is following.

This type of content is a currently untapped place in the tutor for investigators to be testing different types of tutorial feedback. By making multiple types of tutorial feedback that can be chosen at random by the system when a student asks for help or gets an item wrong, investigators can test hypotheses by embedding experiments within mastery problem sets. A similar approach to embedding experiments through the random selection of types of feedback has been explored (Aist & Mostow, 2000) using Project Listen, a reading tutor. After data is gathered, the student response sequences can be analyzed and the learning rates of each strategy calculated using the Item Effect Model. A statistical significance test is employed with the method to tell investigators the probability that their result occurred by chance. We ran 5 such tutorial feedback experiments embedded in mastery learning content. The following sections will show how this data were analyzed with the model and the conclusions that can be drawn from this analysis.

**The tutorial feedback experiments**

We planned out five experiments to investigate the effectiveness of various types of tutorial feedback shown in Table 1. The choices of feedback types were selected based on past studies of effective tutor feedback and interventions (Kim, Weitz, Heffernan, & Krach, 2009; Razzaq & Heffernan, 2006; Razzaq & Heffernan, 2009) that have been run on ASSISTments. To create the experiments we took existing mastery learning problem sets from various math subjects and created two types of feedback conditions for each item in the problem set. The two types of feedback corresponded to the conditions we had planned for that experiment. This authoring process was made less tedious by utilizing the template feature described in the introduction to create the two types of tutorial help templates for each item template in the problem sets.
Table 10. The five planned mastery tutoring experiments and a description of their subject matter and the two types of tutor feedback being tested.

<table>
<thead>
<tr>
<th>Experiment #</th>
<th>Condition A</th>
<th>Condition B</th>
<th>Subject Matter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Solution (Steps)</td>
<td>TPS</td>
<td>Ordering fractions and decimals</td>
</tr>
<tr>
<td>2</td>
<td>Solutions</td>
<td>Worked Example</td>
<td>Finding percents</td>
</tr>
<tr>
<td>3</td>
<td>Hints</td>
<td>TPS</td>
<td>Equation solving (Easy)</td>
</tr>
<tr>
<td>4</td>
<td>Solution (Steps)</td>
<td>Solution</td>
<td>Equation solving (Medium)</td>
</tr>
<tr>
<td>5</td>
<td>Solution (Steps)</td>
<td>TPS</td>
<td>Equation solving (Hard)</td>
</tr>
</tbody>
</table>

The five types of tutorial feedback tested:

- **TPS**: Tutored Problem Solving (Kim, 2009; Razzaq & Heffernan, 2006; Razzaq & Heffernan, 2009). This is the default scaffolding feedback in ASSISTments. Students are asked to solve a number of short problems that break the original problem up into steps.
- **Worked Example**: In this condition (Kim, 2009) students are shown a complete solution to a problem similar to the one they were originally asked to solve.
- **Solution**: In this condition (Razzaq & Heffernan, 2009) students are shown a complete solution to the exact question they were originally asked to solve.
- **Solution (Steps)**: In this condition (Razzaq & Heffernan, 2009) students were shown a complete solution to the problem they were originally asked to solve but broken up in to steps. The student needed to click a check box confirming he or she had read the current solution step to move on.
- **Hints**: In this condition (Razzaq & Heffernan, 2006) students were given text-based hints on how to solve the problem. Students had the opportunity to attempt to answer the original question again at any time. If the student asked for additional hints, the hints would start informing the student exactly how to solve the problem. The last hint would tell them the correct answer to the original problem.

An example of the two feedback conditions from experiment #1, Solution (steps) vs. TPS, can be seen in Appendix B.

**Modeling skill building content containing multiple types of tutoring**

We adapted the Item Effect Model to suit our needs by making small changes to the assumption of what an item represents. In the standard Item Effect Model, an item directly represents the question and tutorial feedback associated with that item. Since we were concerned with the effectiveness of multiple types of tutorial feedback for the same items, we evaluated the learning rates of the feedback instead of the item.

For example, suppose we had a skill building problem set that used two templates, 1 and 2, and we also have two types of tutor feedback, A and B, that were created for both templates. We might observe student responses like the ones in Table 2.
Table 11. Example of skill building data from two students.

<table>
<thead>
<tr>
<th>Student 1 response sequence</th>
<th>Question 1</th>
<th>Question 2</th>
<th>Question 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Incorrect</td>
<td>Incorrect</td>
<td>Correct</td>
</tr>
<tr>
<td>Student 1 item sequence</td>
<td>Template 1, Feedback A</td>
<td>Template 2, Feedback B</td>
<td>Template 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Student 2 response sequence</th>
<th>Question 1</th>
<th>Question 2</th>
<th>Question 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Incorrect</td>
<td>Correct</td>
<td>Correct</td>
</tr>
<tr>
<td>Student 2 item sequence</td>
<td>Template 1, Feedback B</td>
<td>Template 1</td>
<td>Template 1</td>
</tr>
</tbody>
</table>

Table 2 shows two students’ response patterns and the corresponding templates (1 and 2) and tutorial feedback assigned (A and B). Student 1 answers the first two items incorrectly but the last one correctly. Student 2 answers the first item incorrectly but the remaining two correctly. If we assume both students learned when they started to answer items correctly then we can look at which tutorial strategy directly preceded the correct answers and credit that tutorial strategy with the learning. In the example data above, tutorial feedback B precedes both students’ correct responses. In essence, this is the information the Item Effect method uses to determine the learning rates of types of tutorial feedback, albeit in a more probabilistic fashion than the observational method just described. For analyzing the learning effect of different types of tutor feedback we assume all items in a problem set to have the same guess and slip rate since the guess and slip of an item is independent of its tutorial feedback. Some students may encounter both feedback types in the problem set, creating a within subject type of design; however, since the feedback is chosen at random, this occurrence would be by chance. Due to the increase in complexity of the model with longer sequences of items, only students’ first three responses in the problem sets were analyzed. A solution to this limitation of the model has been demonstrated in recent item difficulty modeling work (Pardos & Heffernan, 2011) but not yet applied to the Item Effect model. Besides computational complexity, there are other reasons to only consider the first few responses. One reason is selection bias that exists in considering longer sequences. That is, lower aptitude students who take more opportunities to master a skill also generate longer response sequences. Considering the complete sequence of these students would bias the parameter learning processes since they have produced more data points.

Learning gain analysis

To evaluate the Bayesian analysis method, we wanted to compare it to a more traditional learning gain approach. We defined the traditional learning gain approach as follows. The idea was to treat an item as a pre-test, assign the student to a condition based upon the feedback they received on that item, and view the next item as a post-test. First we selected only those students who answered their first question incorrectly because these are the only students that would have seen the tutoring on that question. These students were then split into two groups determined by the type of feedback (condition A or B) they received on that question. We let the second question pose as a post-test and took the difference between their first response and their second response to be their learning gain. Since only incorrect responses to the first question were selected, the gain will either be 0 or 1. We then did the same procedure for all students who answered the second question incorrectly, with their posttest consisting of their performance on the third question. This approach allows for a student to be assigned to multiple conditions, and even to the same condition twice. In this case, each learning gain is viewed independently of the other. To determine if the difference in learning gains was significant we ran a t-test on the student learning
gains of each condition. We do not claim this to be the most powerful statistical analysis that can be achieved but we do believe that it is sound, in that a claim of statistical significance using this method can be believed.

**Results**

The Item Effect Model fit learning rates for each condition per experiment based on the data. The condition (A or B) with the higher learning rate for each experiment was chosen as Best. For the learning gain analysis the condition with the higher average gain score was chosen as Best. Table 3 shows the results of the analysis including the number of students in each experiment and the respective gain and learning rate results for the respective methods. In each of the experiments, the condition with the highest gain score was also the condition with the highest Bayesian learning rate. The learning rate and guess/slip values are based on fitting the parameters to the entire dataset, while the p-value is calculated from the binomial 10 bin split, described in the Item Effect method section.

**Table 12.** Analysis of the six tutoring experiments by the learning gain analysis method and the Item Effect Model. The methods agreed on the best condition in 6 of the 6 experiments.

<table>
<thead>
<tr>
<th>Exp.</th>
<th>Users</th>
<th>gain A</th>
<th>gain B</th>
<th>p-value</th>
<th>Guess</th>
<th>Slip</th>
<th>Irate A</th>
<th>Irate B</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>155</td>
<td>0.6875</td>
<td>0.7059</td>
<td>0.8734</td>
<td>0.26</td>
<td>0.08</td>
<td>0.2469</td>
<td>0.3188</td>
<td>0.17</td>
</tr>
<tr>
<td>2</td>
<td>559</td>
<td>0.2611</td>
<td>0.2246</td>
<td>0.3038</td>
<td>0.06</td>
<td>0.11</td>
<td>0.1586</td>
<td>0.1259</td>
<td>0.38</td>
</tr>
<tr>
<td>3</td>
<td>774</td>
<td>0.3544</td>
<td>0.3992</td>
<td>0.2735</td>
<td>0.15</td>
<td>0.12</td>
<td>0.1090</td>
<td>0.1229</td>
<td>0.38</td>
</tr>
<tr>
<td>4</td>
<td>505</td>
<td>0.4024</td>
<td>0.3827</td>
<td>0.6725</td>
<td>0.17</td>
<td>0.18</td>
<td>0.1399</td>
<td>0.0996</td>
<td>0.38</td>
</tr>
<tr>
<td>5</td>
<td>189</td>
<td>0.2142</td>
<td>0.3418</td>
<td>0.0696</td>
<td>0.12</td>
<td>0.13</td>
<td>0.2143</td>
<td>0.3418</td>
<td>0.38</td>
</tr>
<tr>
<td>5*</td>
<td>138</td>
<td>0.2833</td>
<td>0.4225</td>
<td>0.0994</td>
<td>0.14</td>
<td>0.17</td>
<td>0.0834</td>
<td>0.1480</td>
<td>0.05</td>
</tr>
</tbody>
</table>

The traditional analysis did not find a significant difference in the conditions of any of the experiments while the Item Effect Model found a significant difference in one experiment, 5* - Equation solving (hard) - (Solution steps vs. TPS). The reliably better condition for this experiment was B, the tutored problem solving condition. This was also the experiment where students who saw condition A were given a bad solution due to a typo. These results show that the Item Effect Model successfully detected this typo effect and found it to be statistically significantly reliable. The learning gain analysis agreed that condition B was superior but not at the p<0.05 level. Experiment 5* represents the data from students before the typo was fixed, data collected after the fix is represented in the normal experiment 5 where condition B was still best but no longer significantly so.

Also included in the table are the guess and slip rates learned by the Item Effect Model for each experiment. It is noteworthy that the highest guess rate learned of 0.26 was for the only experiment (#1) whose items were multiple-choice (four answer choices). It is also noteworthy to observe that the slip value for the easy equation solving experiment (#3) has the lowest probability of slip of the three equations solving experiments (3-5). These observations are evidence of a model that learns highly interpretable parameter values; an elusive but essential trait when the interpretation of parameters is informing pedagogical insights.

**Aptitude-treatment Interactions**

In order to evaluate if there is an aptitude-treatment interaction effect, for each experiment we split the students into two equal-sized groups based on their system wide performance excluding
their responses to these experimental problem sets. These two groups were re-run through the Item Effect Model and their learning gains recalculated. Several studies have found an interaction effect among high and low proficiency students when comparing tutored problem solving (TPS) to worked out solution tutorial feedback (Razzaq & Heffernan, 2009; VanLehn et al, 2005). The studies have shown through a repeated measure traditional RCT experiment that low knowledge students benefitted most from TPS, and high knowledge students showed greater benefit from being shown the problem’s solution, instead. Two of our experiments, #1 and #5, compare these two strategies. Experiment #1 was the only experiment with a significant interaction result, which confirmed the benefit of TPS over Solutions for low proficiency students found in prior work. Interaction results are shown in Table 4.

**Table 4.** Mean learning gains and learning rates (lrate) by experiment interaction group. The second condition for all experiments in the table was tutored problem solving (TPS). The first condition is Solution (steps) except for experiment 3 where the first condition is Hints.

<table>
<thead>
<tr>
<th>Exp.</th>
<th>Users</th>
<th>Group</th>
<th><strong>Learning gain analysis</strong></th>
<th><strong>Item Effect Model analysis</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Other Gain</td>
<td>TPS Gain</td>
</tr>
<tr>
<td>1</td>
<td>78</td>
<td>High</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>1</td>
<td>78</td>
<td>Low</td>
<td>0.5833</td>
<td>0.6428</td>
</tr>
<tr>
<td>5</td>
<td>80</td>
<td>High</td>
<td>0.3030</td>
<td>0.5312</td>
</tr>
<tr>
<td>5</td>
<td>80</td>
<td>Low</td>
<td>0.1568</td>
<td>0.2127</td>
</tr>
<tr>
<td>3</td>
<td>316</td>
<td>Boy</td>
<td>0.2932</td>
<td>0.3773</td>
</tr>
<tr>
<td>3</td>
<td>390</td>
<td>Girl</td>
<td>0.4475</td>
<td>0.4179</td>
</tr>
</tbody>
</table>

Experiment 1 was the ordering fractions topic problem set and experiment 5 was the high difficulty equation solving problem set. Prior work (Razzaq & Heffernan, 2009) showing interaction effects were for the skills of Slope-and-Intercept and Symbolization. The statistical significance of the learning gains is not reported for the high group in experiment 1 as these students never answered two consecutive questions incorrectly and hence always learned. We see that TPS caused the most learning in both high and low knowledge students, with the Item Effect model detecting a reliable difference for low knowledge students on ordering fractions. It is worth remarking that when analyzing all five experiments for proficiency interaction, the guess rate for the high knowledge group is greater than the low knowledge and conversely the slip rate is lower, the interpretation being that high knowledge students are better at guessing and make fewer mistakes.

A gender interaction was found in previous studies of fifth grade students for hints vs. highly interactive hints (Arroyo, Beck, Woolf, Beal, & Schultz, 2000). In those studies girls tended to learn more than boys. Also among the girls, highly interactive hints caused more learning than low interactive hints. Our TPS condition is similar to these highly interactive hints which require input from the user and our hints are similar to their low interactive hints. Experiment 3, equation solving (easy), tested these two tutorial feedback strategies. We split the 774 students in experiment 3 into two groups, boys (316) and girls (390). Since gender information is not kept in the system, the student’s gender was inferred from their first and middle names, with the remaining 38 students being excluded as their gender could not be determined. Learning gains and learning rates were then calculated for each of these groups. Our experiment 3 interaction results agree with Arroyo (2000) in that girls are learning more on average, although we did not replicate the benefit, among girls, of highly interactive hints (TPS) over Hints. We find slight evidence for the opposite interaction but are far from reliable differences.
For these aptitude treatment interactions, the Item Effect Model was able to detect a significant effect and one which replicated a previous study conducted by means of a proper RCT. However, this was the only significant result returned suggesting that more data is necessary and that the interaction effects may not be very large.

ANALYZING RCT DESIGNS WITH FEEDBACK ON ALL ITEMS

We wanted to investigate the performance of our model on data that took the form of a more traditional RCT. Instead of running new randomized controlled trials, we searched our log data for regular problem sets that were not intended to be RCTs but that satisfied a pre-test/condition/post-test experimental design. For this analysis, the focus was not on which particular tutor feedback was better but rather which items, and their tutorial help, were significantly better.

**Looking for RCT designs in the log data of randomized problem sets**

The data used in this analysis also came from ASSISTments. We identified four item problem sets in which the items were given in a random order and where each of the four items tested the same skill. Unlike the items in the skill building datasets, these items had only one type of tutoring for each item. Once we identified such problem sets we selected pairs of sequences within each problem set where the first and third items presented to the students were the same. For example, for items A,B,C,D we looked at the specific sequence pairs of orderings CADB and CBDA where item C would serve as the pre-test and item D as the post-test and items A and B would serve as the two conditions. We required that students had completed the four items in a single day and that there were at least 50 students of data for each of the sequence pairs. Randomization of item sequences in these older problem sets was not always uniform due an implementation detail of the tutor. Because of this, only two sequences which had nearly equal distribution were used for this analysis. This problem was not present for the skill building datasets.

**Face validity**

We wanted to investigate the face validity of the methods’ choice of best condition. To do this, the three authors of this work and a third subject matter expert, who all have served as K-12 educators in mathematics, served as judges. They were told which two items were used as the pre and post-test and which were used as the conditions. They were also able to inspect the tutoring of the items and then judge which of the two items that served as the conditions were more likely to show learning gains. Their answers were recorded and later compared to the condition chosen by the two data mining based hypothesis testing methods. The judges made their predictions before the data analysis was conducted.

**Modeling learning in problem sets with RCT data sequences**

One difference between the modeling of these datasets and the mastery learning datasets is the reduction in sequences and the decision to let each item have its own guess and slip value. Observing only two sequence permutations is not the ideal circumstance for the Item Effect Model but represents a very common design structure of experiment data that will serve as a relevant benchmark.

**Learning gain analysis**
Since this particular data more closely resembled an RCT, we were able to use a more familiar learning gain analysis as testing method of comparison. Learning gain was calculated by taking the post-test minus the pre-test for each student in their respective condition. To calculate if the learning gains of the two conditions were statistically significantly different, a t-test was used.

Results

Table 6 shows the results of the two analysis methods as well as the best condition picks of the four raters in the subject matter expert survey. For each experiment the condition groups were found to be balanced at pre-test. There was one experiment, #4, in which both methods agreed on the best condition and reported a statistically significant difference between conditions. There was a disagreement on experiment #3; however, the Bayes method did not claim a statistically significant different between condition for this experiment.

Table 6. Analysis of the five RCT style experiments by the traditional hypothesis testing method and the Item Effect Model. The methods agreed on the best condition in 4 of the 5 experiments and agreed on statistical significance in one experiment.

<table>
<thead>
<tr>
<th># Users</th>
<th>Learning gain analysis</th>
<th>Item Effect Model</th>
<th>Judges’ Picks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#</td>
<td>Users</td>
<td>Best</td>
</tr>
<tr>
<td>1</td>
<td>149</td>
<td>A</td>
<td>0.74</td>
</tr>
<tr>
<td>2</td>
<td>197</td>
<td>A</td>
<td>0.36</td>
</tr>
<tr>
<td>3</td>
<td>312</td>
<td>A</td>
<td>0.04</td>
</tr>
<tr>
<td>4</td>
<td>208</td>
<td>A</td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>247</td>
<td>B</td>
<td>0.44</td>
</tr>
</tbody>
</table>

The subject matter experts all agreed on four of the five experiments. On the experiment where one judge disagreed with the others, #3, the majority of judges selected the condition that the learning gain analysis chose as best. This experiment was also one in which the two methods disagreed on the best condition but only the traditional method showed statistical significance. On the experiment in which both methods of analysis showed a statistical difference, #4, there was total agreement among our subject matter experts and both of the methods. On average the learning gain analysis agreed more with the judges’ choices and also found significance in two of the experiments whereas the Item Effect Method only found significance in one. However, a correlation coefficient of 0.935 was calculated between the two methods’ significance values indicating that The Item Effect method’s significance is highly correlated with that of learning gain analysis for these RCT style datasets. The similarity in the results of the familiar learning gain analysis to the Item Effect Model gives credence to the technique of evaluating effects based on inspecting the parameters of a Bayesian model fit with EM from data.

ANALYZING SYNTHESIZED DATA

The analysis of real world data in this work demonstrates the similarity in conclusions of both datamining-oriented hypothesis-testing methods but could not evaluate the accuracy of either method’s conclusions due to a lack of ground truth to the experiments. In order to gain some insight into the power and accuracy of each method, a simulation study was conducted. The study simulates students answering a set of four items. The ability of each method to reliably detect which item had the highest learn rate is the focus of the study. The simulation data used in this
study is the same simulation dataset described in the original Item Effect paper (Pardos & Heffernan, 2009a). That study showed how increasing amounts of data are necessary to detect smaller differences in learning rates. In order to compare hypothesis-testing methods, we also apply the learning gain analysis to the synthesized data to see the rate of true and false positives over varying numbers of simulated students.

**Simulation procedure**

The simulation procedure involved running 20 experiments for each user size. The number of users simulated was: 100, 200, 500, 1000, 2000, 4000, 10000 and 20000. For each experiment, four learning rates and four guess and slip values were randomly generated for each of the four items. Each simulated user was assigned a random ordering of the four items. Next, a synthesized data set was generated for each simulated user. Specifically, a response sequence of 0s and 1s was generated for each simulated user by randomly sampling from that user’s network. The Bayesian and learning gain analysis methods then evaluated this synthesized dataset in the same way real world datasets were evaluated. Since the ground truth parameters of the simulation were known, we can observe when either method is making the correct claim about which item is best. If a method selects an item as best (highest learning gain or learning rate) and does so with a reported $p \leq 0.05$, this is considered a true positive. If the method selects the wrong item as best with a $p \leq 0.05$, this is considered a false positive. The percentage of the 20 experiments that are true or false is the true and false positive rate. In the original Item Effect paper, the number of bins created for the binomial test was 20 due to the larger synthesized data sizes and the N chosen for significance was 10 ($p = 0.0139$), we retain the same significance procedure for this analysis. For more details on the simulation methodology please see Pardos and Heffernan (2009a).

**Simulation results**

The results were that the Item Effect model showed higher statistical power with a true positive rate (TP) of 50% compared to the Learning gain analysis, which had a true positive rate of 33%. These rates are averaged rates across the user sizes. The learning gain analysis, however, had a lower false positive (FP) rate of 2% compared to the Item Effect’s 4%. Both false positives rates were reasonable considering 5% is the expected error rate when claiming significance at the $p < 0.05$ level.
The plot in Figure 3 shows how the TP and FP rates change with number of simulated students (amount of data). Both methods show a low TP rate with 100 and 200 users that more than triples with 500 users. From 500 to 2,000 users the Item Effect method continues to increase its TP rate while the Learning gain analysis apparently benefits very little from the extra data. From 4000 users and upwards, both methods increase in TP with the Item Effect model taking a small dip in TP at 4,000. The Item Effect model has a higher TP rate at every user size but the gap is smaller at 500 and below (0.06 average difference) compared to 1,000 users and above where the average Item Effect advantage in TP rate is 0.23 on average. This analysis suggests that when the number of users in a study is large (on the order of 1,000s) the Item Effect model provides a substantially better chance of identifying effects. From 1,000 users on, the Item Effect model captures no less than 37% more true positives (at users = 4,000) and at most, 100% more true positives (at users = 2,000) than the learning gain analysis. While this is not a common cohort size for RCTs it is increasingly common place in ITS. The conclusion, given the assumptions of this simulation study, is that the Item Effect Model is better than the learning gain analysis for analyzing the learning effects of interventions in large scale tutor data. This simulation study also reinforces the notion that reliable, interpretable and objectively accurate parameters can be fit with EM and that accuracy is converging to 100% with increased amounts of data.

**CONTRIBUTIONS**

In this work we presented two data analysis approaches to hypothesis testing with empirical validation of some of the capabilities and benefits of the Item Effect Model introduced by Pardos and Heffernan (2009a). We conducted five original experiments comparing established forms of tutorial feedback with which to compare a learning gains analysis approach to the Item Effect Model. In conducting the experiments by inserting multiple tutorial feedback conditions into items in skill building problem sets, we were able to demonstrate an example of running an
experiment without interrupting the learner’s curriculum and without giving lengthy pre and post-tests that consume valuable instructional time.

A second contribution of this work was to highlight how random orderings of questions could be analyzed as if they were RCTs. This is a powerful idea; that simply randomizing the order of items creates implicit experimental designs that allow for discovery of valuable pedagogical insights about what causes learning. This work brings the field closer to the goal of automating the analysis of embedded experiments to continuously improve tutor effectiveness, as expressed by Aist and Mostow (2000). We have expanded on this idea by evaluating not only the effectiveness of feedback but also the effectiveness of items given individually-randomized orderings. We have also run our methods against synthesized datasets in order to give an expectation of the statistical power and accuracy of each method given varying amounts of data. The simulation suggested that with lower data amounts ($\leq 500$) a learning gain analysis is very comparable to the Item Effect model, however, with greater amounts of data the Item Effect model demonstrates substantially greater power in detecting effects.

We believe that the methodology we used to calculate significance in the Item Effect Model could be made more powerful but decided to err on the side of caution by using a method that we were confident was not going to lead us to draw spurious conclusions. It was encouraging to see how the Item Effect Model performed when compared to a more traditional hypothesis testing method of analysis. The model agreed with the traditional tests in 10 out of the 11 experiments and detected an effect that the traditional method could not; a typo effect in a condition of one of our mastery learning problem set experiments. We believe this work builds confidence in the practice of interpreting Bayesian model parameters to evaluate intervention effectiveness. The implication of these results is that ITS researchers can explore their datasets using either data mining based hypothesis testing method without concern that they will draw spurious conclusions and that the Item Effect Model will likely be able to detect significance where the learning gains analysis cannot when the scale of students is in the thousands. Randomization of tutor decisions and use of data mining based effect evaluation methods such as the Item Effect Model can be value tools to improve ITS.

**FUTURE WORK**

One of the practical application goals for developing this method is to build this analysis into the infrastructure of the tutor in order to be able to identify content that reliably produces and does not produce learning and make sure that content is brought to the attention of teachers and tutor curriculum creators. Half of the new content in ASSISTments is being generated by teachers and external web content is also being integrated as an option for tutorial help. With the influx of this type of content, it will become even more important to have an automated system that can quickly find what is working for students and what is not.

A second practical future application of the model is to use it to improve the tutor skill model or question skill tagging, also called the Q-matrix. An item that has a very low learn rate but a high guess may contain only one of a number of skills it is tagged with and which are represented in the other items in the problem set. It could also contain a pre-requisite skill and not the same skill as represented in the other items. An item with a very low learn rate but a high slip could indicate an item that has more skills involved than it is tagged with or could indicate that it is tagged with the wrong skill.

Lastly, the Item Effect model estimates reliability by measuring the reliability of EM fit parameters on different samples of the data. This method leads to sparse samples when the dataset size is small and also requires smaller samples in order to achieve finer grained reliability calculations. A reliability estimation that calculates values per student (such as inferred knowledge gain) instead of per data sample may lead to more statistical power in determining item and feedback effectiveness.
Chapter 8: The Predictive Power of Adding Difficulty and Discrimination to the Item Individualization Model

Many models in computer education and assessment take into account difficulty. However, despite the positive results of models that take difficulty into account, knowledge tracing is still used in its basic form due to its skill level diagnostic abilities that are very useful to teachers. This leads to the research question we address in this work: Can KT be effectively extended to capture item difficulty and improve prediction accuracy? There have been a variety of extensions to KT in recent years. One such extension was Baker's contextual guess and slip model. While this model has shown positive gains over KT in internal validation testing, it has not performed well relative to KT on unseen in-tutor data or post-test data; however, it has proven a valuable model to use alongside other models. The contextual guess and slip model increases the complexity of KT by adding regression steps and feature generation. The added complexity of feature generation across datasets may have hindered the performance of this model. Therefore, one of the aims of our work here is to make the most minimal of modifications to the KT model in order to add item difficulty and keep the modification limited to changing the topology of the model. We analyze datasets from two intelligent tutoring systems with KT and a model we have called KT-IDEM (Item Difficulty Effect Model) and show that substantial performance gains can be achieved with this minor modification that incorporates item difficulty.

This chapter was published at the following venue:

INTRODUCTION

Many models in computer education and assessment take into account difficulty. Item Response Theory (IRT) is one such popular model. IRT is used in Computer Adaptive Testing (CAT) and learns a difficulty parameter per item (John, Mahadeyan & Woolf, 2006). This makes IRT models very powerful for predicting student performance; however the model learning processes is expensive and is not a practical way of determining when a student has learned a particular skill because it does not model learning. Despite the predictive power of IRT, the Cognitive Tutors (Koedinger & Corbett, 2006) employ standard Knowledge Tracing (KT) to model students’ knowledge and determine when a skill has been learned (Corbett & Anderson, 1995). Knowledge Tracing is used because it is a cognitively diagnostic form of assessment which is beneficial to both student and teacher. The parameters for a KT model need only be learned once, typically at the beginning of the school year (based on the past year’s data) and the inference of individual student’ knowledge of a skill can be executed with very little computation. Models like IRT that take into account item difficulty are strong at prediction, and model such as KT that infer skills are useful for their cognitively diagnostic results. This leads us to our research question: Can KT be effectively extended to capture item difficulty and improve predictive?

There have been a variety of extensions to KT in recent years. One such extension was Baker, Corbett & Aleven’s (2008) contextual guess and slip model. While this model has shown positive gains over KT in internal validation testing, it has not performed well relative to KT on unseen in-tutor data or post-test data; however, it has proven a valuable model to use alongside other models. Likewise, the contextual slip model (Baker et al., 2010) also suffered the same
inadequacies on in-tutor data prediction. The contextual guess and slip model increased the complexity of KT by adding regression steps and feature generation. The added complexity of feature generation across datasets may have hindered the performance of this model. Therefore, one of the aims of our work in this chapter was to make the most minimal of modifications to the KT model in order to add item difficulty and keep the modification limited to slight changes to the topology of the model.

Knowledge Tracing

The standard Bayesian Knowledge Tracing (KT) model has a set of four parameters which are typically learned from data for each skill in the tutor. These parameters dictate the model's inferred probability that a student knows a skill given that student's chronological sequence of incorrect and correct responses to questions of that skill thus far. The two parameters that determine a student's performance on a question given their current inferred knowledge are the guess and slip parameters and these parameters are where we will explore adding question level difficulty. The guess parameter is the probability that a student will answer correctly even if she does not know the skill while the slip parameter is the probability that the student will answer incorrectly when she knows the skill. Skills that have a high guess rate can be thought of, intuitively, as easy (a multiple choice question for example). Likewise, skills that have a low guess and/or a higher rate of mistakes (high slip) can be thought of as hard. Based on this intuition we believe a questions' difficulty can be captured by the guess and slip parameter. Therefore, we aim to give each question its own guess and slip thereby modeling a difficulty per item.

Figure 1 depicts the standard KT model. The three latent nodes representing knowledge are above the three observable nodes representing questions in the tutor. The depiction is showing an unrolled dynamic Bayesian topology for modeling a sequence of three questions but this chain can continue for an arbitrary number of questions a student answers. The guess and slip parameters are represented by $P(G)$ and $P(S)$ respectively. The two knowledge parameters, which dictate the state of the knowledge node, are the probability of learning, $P(T)$, and probability of initial knowledge, $P(L_0)$, also referred to as prior probability of knowledge or just $P(L_0)$ is the probability that a student knows the skill before answering the first question and $P(T)$ is the probability that a student will transition from not knowing the skill to knowing it.

![Figure 1](image.png)

**Figure 1.** The standard Knowledge Tracing model

While knowledge is modeled as a binary variable (a student is either in the learned or unlearned state), the inferred probability of knowledge is a continuous value. Once that probability reaches
0.95, the student can be assumed to have learned the skill. The Cognitive Tutors use this threshold to determine when a student should no longer be asked to answer questions of a particular skill.

**KNOWLEDGE TRACING: ITEM DIFFICULTY EFFECT MODEL (KT-IDEM)**

One of our stated goals was to add difficulty to the classical KT model without going outside of the Bayesian topology. To do this we used a similar topology design to that which was demonstrated in Pardos & Heffernan's (2010a) student individualization paper. In that work a multinomial node was added to the Bayesian model that represented the student. The node(s) containing the parameters which the authors wished to individualize were then conditioned based on the student node, thus creating a parameter per student. For example, if one wished to individualize the prior parameter, the student node would be connected to the first knowledge node since this is where the prior parameter's CPT is held. A separate prior could then be set and learned for each student. Practically, without the aid of a pre-test, learning a prior for every student is a very difficult fitting problem, however, simplifying the model to represent only two priors and assigning students to one of those priors based on their first response has proven an effective heuristic for improving prediction by individualizing the prior.

In a similar way that Pardos & Heffernan showed how parameters could be individualized by student, we individualized the guess and slip parameter by item. This involved creating a multinomial item node, instead of a student node, that represents all the items of the particular skill being fit. This means that if there were 10 distinct items in the skill data, the item node would have values ranging from 1 to 10. These values are simply identifiers for the items which can arbitrarily be assigned. The item node is then connected to the question node (Fig 2) in the topology, thus conditioning the question's guess/slip upon the value of the item node. In the example of the 10 item dataset, the model would have 10 guess parameters, 10 slip parameters, a learn rate and a prior, totaling 22 parameters versus standard KT's 4 parameters. It is possible that this model will be over parameterized if a sufficient amount of data points per item is not met; however, there has been a trend of evidence that suggests models that have equal or even more parameters than data points can still be effective such as was shown in the Netflix challenge (Bell & Koren, 2007) and 2010 KDD Cup on Educational Data Mining (Yu et al., 2010).

![Knowledge Tracing – Item Difficulty Effect Model](image)

**Figure 2.** The KT-IDEM topology depicting how the question node (and thus the guess/slip) is conditioned on the item node to add item difficulty to the KT model.

Figure 2 illustrates how the KT model has been altered to introduce item difficulty by adding an extra node and an arc for each question. While the standard KT model has a single $P(G)$ and $P(S)$,
KT-ITEM has a $P(G)$ and $P(S)$ for each item, for example $P(G|I=1)$, $P(G|I=2)$... $P(G|I=10)$, stating that there is a different guess parameter value given the value of the item node. In the example in Figure 2, the student sees items with IDs 3, 1, 5 and then 2. This information is fully observable and is used in model training, to fit appropriate parameters to the item $P(G|I)$ and $P(S|I)$, and in model tracing (prediction), to inform which items a particular student has encountered and make the appropriate inference of knowledge based on the answer to the item. By setting a student’s item sequence to all 1s during training and tracing, the KT-IDEM model represents the standard KT model, therefore the KT-IDEM model, which we have introduce in this chapter, can be thought of as a more general KT model. This model can also be derived by modifying models created by the authors for detecting the learning value of individual items (Pardos, Dailey & Heffernan, In Press).

DATASETS

We evaluate the KT and KT-IDEM models with two datasets from two separate real world tutors. The datasets will show how the models perform across a diverse set of different tutoring scenarios. The key factor of KT-IDEM is modeling a separate guess and slip parameter for every item in the problem set. In these two datasets, the representation of an item differs. In the ASSISTments dataset, a problem template is treated as an item. In the Cognitive Tutor dataset, a problem, which is a collection of steps, is treated as an item. The sections bellow provide further descriptions of these systems and the data that were used.

The ASSISTments Platform

Our first dataset consisted of student responses from ASSISTments (Razzaq et al., 2005), a web based math tutoring platform which is best known for its 4th-12th grade math content. Figure 3 shows an example of a math item on the system and tutorial help that is given if the student answers the question wrong or asks for help. The tutorial help assists the student in learning the required knowledge by breaking each problem into sub questions called scaffolding or giving the student hints on how to solve the question. A question is only marked as correct if the student answers it correctly on the first attempt without requesting help.

Item templates in ASSISTments

Our skill building dataset consists of responses to multiple questions generated from an item template. A template is a skeleton of a problem created by a content developer in the web based builder.
application. For example, a template could specify a Pythagorean Theorem problem, but without the numbers for the problem filled in. In this example the problem template could be: “What is the hypotenuse of a right triangle with sides of length X and Y?” where X and Y are variables that will be filled in with values when questions are generated from the template. The solution is also dynamically determined from a solution template specified by the content developer. In this example the solution template would be, “Solution = sqrt(X^2+Y^2)”. Ranges of values for the variables can be specified and more advance template features are available to the developer such as dynamic graphs, tables and even randomly selected cover stories for word problems. Templates are also used to construct the tutorial help of the template items. Items generated from these templates are used extensively in the skill building problem sets as a pragmatic way to provide a high volume of items for students to practice particular skills on.

**Skill building datasets**

Skill building is a type of problem set in ASSISTments that consists of hundreds of items generated from a number of different templates, all pertaining to the same skill or skill grouping. Students are marked as having completed the problem set when they answer three items correctly in a row without asking for help. In these problem sets items are selected in a random order. When a student has answered 10 items in a skill building problem set without getting three correct in a row, the system forces the student to wait until the next calendar day to continue with the problem set. The skill building problem sets are similar in nature to mastery learning (Corbett, 2001) in the Cognitive Tutors; however, in the Cognitive Tutors, mastery is achieved when a knowledge-tracing model believes that the student knows the skill with 0.95 or better probability. Much like the other problem sets in ASSISTments, skill builder problem sets are assigned by the teacher at his or her discretion and the problem sets they assign often conform to the particular math curriculum their district is following.

We selected the 10 skill builder datasets with the most data from school year 2009-2010, for this chapter. The number of students for each problem set ranged from 637 to 1285. The number of templates ranged from 2-6. This meant that there would be at max 6 distinct sets of guess/slips associated with items in a problem set. Because of the 10 item/day question limit, we only considered a student’s first 10 responses per problem set and discarded the remaining responses. Only responses to original questions were considered. No scaffold responses were used.

**The Cognitive Tutor: Mastery Learning datasets**

Our Cognitive Tutor dataset comes from the 2006-2007 “Bridge to Algebra” system. This data was provided as a development dataset in the 2010 KDD Cup competition (Pardos & Heffernan, In Press). The Cognitive Tutor is designed differently than ASSISTments. One very relevant difference to this work is that the Cognitive Tutor presents a problem to a student (Fig 4) that can consist of questions (also called steps) of many skills. Students may enter their answers to the various questions pertaining to the

**Figure 4.** A Geometry problem within the Cognitive Tutor
A problem in an answer grid (Fig 5). The Cognitive Tutor uses Knowledge Tracing to determine when a student has mastered a skill. A problem in the tutor can also consist of questions of differing skills. However, once a student has mastered a skill, as determined by KT, the student no longer needs to answer questions of that skill within a problem but must answer the other questions which are associated with the unmastered skill(s).

The number of skills in this dataset was substantially larger than the ASSISTments dataset. Instead of processing all skills, a random sample of 12 skills were selected. Some questions consisted of multiple skills. Instead of separating out each skill, a set of skills associated with a question was treated as a separate skill. The Cognitive Tutor separates lessons into pieces called Units. A skill name that appears in one Unit was treated as a separate skill when appearing in a different Unit. Some skills in the Cognitive Tutor consist of trivial tasks such as “close-window” or “press-enter”. These types of non-math related skill were ignored. To maintain consistency with the per student data amount used in the ASSISTments dataset, the max number of responses per student per skill was also limited to the first 10.

**METHODOLOGY**

A five-fold cross-validation was used to make predictions on the datasets. This involved randomly splitting each dataset into five bins at the student level. There were five rounds of training and testing where at each round a different bin served as the test set, and the data from the remaining four bins served as the training set. The cross-validation approach has more reliable statistical properties than simply separating the data in to a single training and testing set and should provide added confidence in the results since it is unlikely that the findings are a result of a “lucky” testing and training split.

**Training the models**

Both KT and KT-IDEM were trained and tested on the same cross-validation data. The training phase involved learning the parameters of each model from the training set data. The parameter learning was accomplished using the Expectation Maximization (EM) algorithm. EM attempts to find the maximum log likelihood fit to the data and stops its search when either the max number of iterations specified has been reached or the log likelihood improvement is smaller than a specified threshold. The max iteration count was set to 200 and threshold was set to the BNT default of 0.001. Initial values for the parameters of the model were set to the following, for both models: \( P(G) \) of 0.14, \( P(S) \) of 0.09, \( P(L_0) \) of 0.50, and \( P(T) \) of 0.14. This set of values were found to be the average parameter values across skills in a previous analysis of ASSISTments data using students from

**Performing predictions**

Each run of the cross-validation provided a separate test set. This test set consisted of students that were not in the training set. Each response of each student was predicted one at a time by

![Figure 5. Answer entry box for the Geometry problem in Fig 4.](image)
both models. Knowledge tracing makes predictions of performance based on the parameters of the model and the response sequence of a given student. When making a prediction on a student’s first response, no evidence was presented to the network except for the item identifier associated with the question. Since no individual student response evidence is presented on the first response, predictions of the first response are based on the models’ prior and guess/slip parameters alone. This meant that, within a fold, KT will make the same prediction for all students’ first response. KT-IDEM’s first response may differ since not all students’ first question is the same and the guess/slip differs based on the question. When predicting the student’s second response, the student’s first response was presented as evidence to the network, and so on, for all of the student’s responses 1 to N.

RESULTS

Predictions made by each model were tabulated and the accuracy was evaluated in terms of Area Under the Curve (AUC). AUC provides a robust metric for evaluating predictions where the value being predicted is either a 0 or a 1 (incorrect or correct), as is the case in our datasets. An AUC of 0.50 always represents the score achievable by random chance. A higher AUC score represents higher accuracy.

ASSISTments Platform

The cross-validated model prediction results for ASSISTments are shown in Table 1. The number of students as well as the number of unique templates in each dataset is included in addition to the AUC score for each model. A Delta column is also included which shows the KT-IDEM AUC subtracted by the KT AUC score. A positive Delta indicates that there was an improvement in accuracy by using KT-IDEM instead of standard KT. A negative indicates that accuracy declined when compared to KT.

Table 1. AUC results of KT vs. KT-IDEM on the ASSISTments datasets. The Delta column reports the increase (+) or decrease (−) in accuracy by using KT-IDEM.

<table>
<thead>
<tr>
<th>Skill</th>
<th>#students</th>
<th>#templates</th>
<th>KT</th>
<th>KT-IDEM</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>756</td>
<td>3</td>
<td>0.616</td>
<td>0.619</td>
<td>+0.003</td>
</tr>
<tr>
<td>2</td>
<td>879</td>
<td>2</td>
<td>0.652</td>
<td>0.671</td>
<td>+0.019</td>
</tr>
<tr>
<td>3</td>
<td>1019</td>
<td>6</td>
<td>0.652</td>
<td>0.743</td>
<td>+0.091</td>
</tr>
<tr>
<td>4</td>
<td>877</td>
<td>4</td>
<td>0.616</td>
<td>0.719</td>
<td>+0.103</td>
</tr>
<tr>
<td>5</td>
<td>920</td>
<td>2</td>
<td>0.696</td>
<td>0.697</td>
<td>+0.001</td>
</tr>
<tr>
<td>6</td>
<td>826</td>
<td>2</td>
<td>0.750</td>
<td>0.750</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>637</td>
<td>2</td>
<td>0.683</td>
<td>0.689</td>
<td>+0.006</td>
</tr>
<tr>
<td>8</td>
<td>1285</td>
<td>3</td>
<td>0.718</td>
<td>0.721</td>
<td>+0.003</td>
</tr>
<tr>
<td>9</td>
<td>1024</td>
<td>4</td>
<td>0.679</td>
<td>0.701</td>
<td>+0.022</td>
</tr>
<tr>
<td>10</td>
<td>724</td>
<td>4</td>
<td>0.628</td>
<td>0.684</td>
<td>+0.056</td>
</tr>
</tbody>
</table>

The results from evaluating the models with the ASSISTments datasets are strongly in favor of KT-IDEM (Table 1) with KT-IDEM beating KT in AUC in 9 of the 10 datasets and tying KT on the remaining dataset. The average AUC for KT was 0.669 while the average AUC for KT-IDEM was 0.69. This difference was statistically significantly reliable (p = 0.035) using a two tailed paired t-test.
Cognitive Tutor

The cross-validated model prediction results for the Cognitive Tutor are shown in Table 2. The number of students, unique problems and data points in each skill dataset are included in addition to the AUC score for each model. The ratio of data points per problem (the number of data points divided by the number of unique problems) is also provided to show the average amount of data there was per problem.

Table 2. AUC results of KT vs KT-IDEM on the Cognitive Tutor datasets. The AUC of the winning model is marked in bold

<table>
<thead>
<tr>
<th>Skill</th>
<th>#students</th>
<th>#prob</th>
<th>#data</th>
<th>#data/#prob</th>
<th>KT</th>
<th>KT-IDEM</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>133</td>
<td>320</td>
<td>1274</td>
<td>3.98</td>
<td>0.722</td>
<td>0.687</td>
<td>-0.035</td>
</tr>
<tr>
<td>2</td>
<td>149</td>
<td>102</td>
<td>1307</td>
<td>12.81</td>
<td>0.688</td>
<td>0.803</td>
<td>+0.115</td>
</tr>
<tr>
<td>3</td>
<td>116</td>
<td>345</td>
<td>1090</td>
<td>3.16</td>
<td>0.612</td>
<td>0.605</td>
<td>-0.007</td>
</tr>
<tr>
<td>4</td>
<td>116</td>
<td>684</td>
<td>1062</td>
<td>1.55</td>
<td>0.694</td>
<td>0.653</td>
<td>-0.041</td>
</tr>
<tr>
<td>5</td>
<td>159</td>
<td>177</td>
<td>1475</td>
<td>8.33</td>
<td>0.677</td>
<td>0.718</td>
<td>+0.041</td>
</tr>
<tr>
<td>6</td>
<td>116</td>
<td>396</td>
<td>1160</td>
<td>2.93</td>
<td>0.794</td>
<td>0.497</td>
<td>-0.297</td>
</tr>
<tr>
<td>7</td>
<td>133</td>
<td>320</td>
<td>1267</td>
<td>3.96</td>
<td>0.612</td>
<td>0.574</td>
<td>-0.038</td>
</tr>
<tr>
<td>8</td>
<td>116</td>
<td>743</td>
<td>968</td>
<td>1.30</td>
<td>0.679</td>
<td>0.597</td>
<td>-0.082</td>
</tr>
<tr>
<td>9</td>
<td>149</td>
<td>172</td>
<td>1431</td>
<td>8.32</td>
<td>0.585</td>
<td>0.720</td>
<td>+0.135</td>
</tr>
<tr>
<td>10</td>
<td>148</td>
<td>177</td>
<td>1476</td>
<td>8.34</td>
<td>0.593</td>
<td>0.626</td>
<td>+0.033</td>
</tr>
<tr>
<td>11</td>
<td>149</td>
<td>172</td>
<td>1431</td>
<td>8.32</td>
<td>0.519</td>
<td>0.687</td>
<td>+0.168</td>
</tr>
<tr>
<td>12</td>
<td>123</td>
<td>128</td>
<td>708</td>
<td>5.53</td>
<td>0.574</td>
<td>0.562</td>
<td>-0.012</td>
</tr>
</tbody>
</table>

The overall performance of KT vs. KT-IDEM is mixed in this Cognitive Tutor dataset. The average AUC of KT was 0.6457 while the average AUC for KT-IDEM was 0.6441; however, this difference is not statistically reliably difference (p = 0.96). As alluded to earlier in the chapter, over parameterization is a potential issue when creating a guess/slip per item. In this dataset this issue becomes apparent due to the considerably high number of problems (avg. 311) compared to the number of templates in ASSISTments (avg. 3). Because of the high number of problems, and thus high number of parameters, the data points per problem ratio (dpr) becomes highly important. The five of the twelve datasets with a dpr > 6 were all predicted more accurately by KT-IDEM, with most showing a substantially higher accuracy over KT (+ 0.10 avg. AUC improvement). Among these five datasets, the average AUC of KT was 0.6124 and the average AUC of KT-IDEM was 0.7108. This difference was statistically reliably (p = 0.02). For the skill datasets with dpr < 6, the loss in accuracy was relatively low (~0.04) with the exception of skill 6 that produced a KT-IDEM AUC of 0.497 a score which was 2 standard deviations lower than the mean KT-IDEM score for Cognitive Tutor. This skill dataset had 396 problems with the most frequent problem accounting for 25% of the data points and the 2nd most frequent problem accounting for only 0.3%. This was exceptionally unbalanced relative to the other skill sets and served as an example of the type of dataset that the KT-IDEM model does not perform well on.

DISCUSSION AND FUTURE WORK

The training of the models in this chapter was accomplished by splitting up a cohort of students into a test and training set through cross-validation. If a previous year’s cohort of students were used instead, this may increase the number of training samples due to not requiring a portion of the data to be held out. This will also raise the issue of which guess and slip values to use for an item that has been added after the previous year’s data was collected and thus was not in the
training set. One approach is to use the average of all the learned guess/slip values or use the standard KT model guess/slip values for that question.

The results for the Cognitive Tutor showed that the average number of data points per problem largely determined if the accuracy of KT-IDEM would be greater than KT. It could be that some problems within a skill dataset have high amounts of data while some problems have low amounts. To improve the accuracy of KT-IDEM, the guess/slip values for the low data problems in the model could be replaced with KT’s guess/slip values. This would ensure that when predicting performance on high data items, KT-IDEM parameters would be used and KT parameters would be used on low data items. The model parameter fitting could potentially be improved by using information such as average percent correct and number of hints requested to set the initial guess/slip values for each item instead of using default guess/slip values.

An open area for future work would be to improve assessment speed by choosing items based on their guess/slip values learned with KT-IDEM. The standard computer adaptive testing paradigm is focused on assessment, not learning. To accomplish quick assessment, these tests select the questions that give the optimal amount of information about a student’s ability based on their response. In an IRT model, this criterion is called item discrimination. A response to an item with high discrimination results in a larger change in the student’s assessed ability level than a response to a lower discrimination item. Likewise, in KT-IDEM, guess and slip can also capture discrimination. When an item has a zero guess and zero slip, the student’s response is completely representative of their knowledge; however, when the guess and slip are closer to 0.50, the response has less of an impact on the updated probability of knowledge. In order to optimize the selection of questions for assessment, questions can be selected that maximize the change in probability of knowledge given an incorrect response and the change in probability of knowledge given a correct response to the selected question. Questions eligible for selection should have had sufficient data used to train their guess/slip values, otherwise erroneously high or low guess/slip values are likely to be learned and would not represent the true discrimination of the item. While this method could minimize the number of questions needed to assess a student, the questions which lead to the most learning do not necessarily correspond to the questions which are best for assessment. The Item Effect Model (Pardos et al., In Press) has been used to determine item learning value with a Knowledge Tracing approach and could complement KT-IDEM for choosing the appropriate questions which blend assistance and assessment.

**CONTRIBUTION**

With the ASSISTments Platform dataset, KT-IDEM was more accurate than KT in 9 out of the 10 datasets. KT scored an AUC of 0.669 on average while KT-IDEM scored an AUC of 0.699 on average. This difference was statistically significant at the p < 0.05 level. With the Cognitive Tutor dataset, overall, KT-IDEM is not statistically reliably different from KT in performance prediction. When dpr is taken into account, KT-IDEM is substantially more accurate (0.10 average gain in AUC over KT). This improvement when taking into account dpr is also statistically reliable at the p < 0.05 level.

We have introduced a novel model for introducing item difficulty to the Knowledge Tracing model that makes very minimal changes to the native topology of the original mode. This new model, called the KT Item Difficult Effect Model (IDEM) provided reliably better in-tutor performance prediction on the ASSISTments Skill Builder dataset. While overall, the new model was not significantly different from KT in the Cognitive Tutor, it was significantly better than KT on datasets that provided enough data points per problem.

We believe these results demonstrate the importance of modeling item difficulty in Knowledge Tracing when sufficient data is available to train the model. The real world implication of improved accuracy in assessment is less student time spent over practicing and
improved accuracy of skill reports given to teachers. Accurate guess and slip parameters per item with KT-IDEM also opens up the capability for a tutoring system to select questions with low guess and slip and thus optimizing the number of questions needed for assessment while remaining inside the model tracing paradigm.

Chapter 9: Summary of the Student Modeling and Tutoring Modeling Approaches

The current paradigm in student modeling, Knowledge Tracing, has continued to show the power of its simplifying assumption of knowledge as a binary and monotonically increasing construct, the value of which directly causes the outcome of student answers to questions. Recent efforts have focused on optimizing the prediction accuracy of responses to questions using student models. Incorporating individual student parameter interactions has been an interpretable and principled approach which has improved the performance of this task, as demonstrated by its application in the 2010 KDD Cup challenge on Educational Data. Performance prediction, however, can have limited practical utility. The greatest utility of such student models can be their ability to model the tutor and the attributes of the tutor which are causing learning. Harnessing the same simplifying assumption of learning used in student modeling, we can turn this model on its head to effectively tease out the tutor attributes causing learning and begin to optimize the tutor model to benefit the student model.

This chapter was published at the following venue:

INTRODUCTION

The beginning of the current paradigm in student modeling, known as Knowledge Tracing (Corbett & Anderson 1995) started with Atkinson’s approach to modeling instruction (Atkinson & Paulson 1972). An adaptation of the Bayesian computations from Atkinson and a simplification of the more complex ACT-R cognitive architecture (Anderson 1993), Knowledge Tracing has firm roots in learning theory. However, it is its use in practice that has drawn the majority of attention to the model. The Cognitive Tutors™, used by over 500,000 students, annually, employ Knowledge Tracing to determine when a student has learned a particular skill and when to subsequently end practice of that skill. The real world adoption of the model has made it a popular yard stick for gauging the relative performance of new models, of which there have been many (Desmarais & Baker 2011).

There has been a focus in the literature on within-tutor predictive performance as the primary benchmark of comparison between models (Pardos et al. 2012). This was also the benchmark used to rank solutions to the recent Knowledge Discovery and Data Mining (KDD) Cup on Educational Data, a high profile annual data mining competition organized by the Association for Computing Machinery (ACM). An extension to Knowledge Tracing which individualized model parameters per student was part of a solution that placed 4th in the competition (Pardos & Heffernan, in press). While the primary application of Knowledge Tracing has been to infer student knowledge, the model can be extended to make inferences about the effect of various components of the tutor on learning.

In this chapter we overview the techniques in which Knowledge Tracing’s Bayesian framework has been extended to incorporate attributes of the student to improve prediction. We
also look at how model extensions have expanded to various attributes of the tutor and allowed for the learning effect of those attributes to be observed.

**The Bayesian Knowledge Tracing Model**

An average student can be modeled as a statistical processes with probability $P(L_0)$ of knowing the skill being practiced before instruction beings. If the student begins with not knowing the skill then she will likely answer the first problem incorrectly but can guess the correct answer with probability $P(G)$. If the student begins with knowing the skill then she will likely answer the first problem correctly but can make a mistake, or slip, with probably $P(S)$. A student who begins with not knowing the skill will learn the skill with probability $P(T)$ between the first and second opportunities and between all subsequent opportunities until the skill is learned. These probabilities; $P(L_0)$, $P(G)$, $P(S)$ and $P(T)$ comprise the set of parameters of Knowledge Tracing with which student knowledge and performance is modeled. This process is equivalent to that of a Hidden Markov Model (HMM). In an HMM, $P(G)$ and $P(S)$ are referred to as the emission parameters, while $P(T)$ is the transition parameter. In the context of Intelligent Tutoring Systems, $P(G)$ and $P(S)$ are referred to as the performance parameters, with $P(L_0)$ and $P(T)$ being the knowledge parameters. In Knowledge Tracing, the probability of forgetting is fixed at zero. The parameters $P(L_0)$ and $P(T)$ affect the projected probability of knowledge over time in a similar fashion to learning curve analysis (Martin et al. 2005). Note that the projected probability of knowledge at the next opportunity to answer a question of the same skill, $P(L_{n+1})$, does not involve the performance parameters and is calculated with the following formula:

$$P(L_n|\text{Response}_n) + (1 - P(L_n|\text{Response}_n))P(T)$$

If no response at opportunity $n$ exists then the prior probability of $L_n$ is used.

Reasoning about the value of the latent given observations of correct or incorrect responses is a separate task involving the guess and slip parameters. The closer to zero the guess and slip parameters, the less uncertainty exists about the latent of knowledge, given an observation. Given a high guess value, a longer sequence of correct responses would be necessary to have 0.95 or greater certainty in the skill being known (the threshold at which the Cognitive Tutors reach the conclusion of mastery). The posterior probability of knowledge, which is the updated probability of knowledge after observing some evidence, $P(L_n|\text{Response}_n)$, is calculated by the following formula, given an observation of a correct answer to a question:

$$\frac{P(L_n)(1 - P(S))}{P(L_n)(1 - P(S)) + (1 - P(L_n))P(G)}$$

Given an observation of an incorrect answer to a question, the following formula is used:

$$\frac{P(L_n)P(S)}{P(L_n)P(S) + (1 - P(L_n))(1 - P(G))}$$

The initial introduction of Knowledge Tracing by Corbett & Anderson used Bayesian update rules to calculate the inference of knowledge, however; it wasn’t until 2004 that Reye demonstrate that these update rules could be completely modeled within the framework of a Dynamic Bayesian Network (Reye 2004). The work referred to in this chapter uses static, unrolled Dynamic Bayesian Networks, which are the equivalent of a DBN for a fixed number of time steps.
**Parameter fitting**

Either grid-search or Expectation Maximization (EM) can be used to fit the parameters of the model to the data. Details of both methods and their predictive performance have been an active topic of discussion in the student modeling literature (Pardos et al. 2012). With the standard knowledge tracing parameters, grid-search runs faster but its runtime increases exponentially with the addition of parameters to the model. The runtime of EM, however, follows a power function with increasing numbers of parameters and is a widely used algorithm for fitting parameters of HMMs, making it a preferred choice when fitting the more complex, individualized models which will be presented in later sections.

**Identifiability**

The standard objective in training parameters of a model is to achieve goodness of fit to the data. The objective in training parameters for a model being used for cognitively diagnostic purposes is two-fold. With such a model, parameter plausibility is also an objective. With four parameters it is possible that the same goodness of fit to the data can be achieved with two entirely different sets of parameter solutions (Beck & Chang 2007). While this is not an issue for data prediction, it is problematic for meaningful inference of the latent of knowledge, which is the primary use of Knowledge Tracing in the Cognitive Tutors. Various mends to the problem have been employed such as bounding parameter values when using grid-search, setting the initial parameter position to plausible values instead of random values when using EM, and individualizing the prior parameter to achieve an improved baseline of traction for plausible parameter convergence (Pardos et al. 2012).

**Modeling Student Individualization**

Standard Knowledge Tracing makes the simplifying assumption that all students learn a skill at the same rate and begin practicing a skill with the same prior knowledge. Individualization of these parameters can break this simplifying assumption and has shown improvement over standard Knowledge Tracing in performance prediction in the Cognitive Tutor for Algebra (Pardos & Heffernan, in press) and for Genetics as well as the ASSISTments tutor’s non-skill building problem sets (Pardos & Heffernan 2010), although; using prior knowledge individualization did not improve prediction in the ASSISTments skill-building problem sets (Pardos et al. 2012).

Corbett & Anderson took a regression approach to individualization that trained the general set of four parameters learned per skill and then used a regression to add in a student weight for each parameter that spanned skills. While incorporation of individual weights resulted in higher correlation of predictions to a post-test, the weights did not improve the accuracy of the predictions of within-tutor student responses. We will discuss an individualization approach proposed by Pardos & Heffernan (2010) that takes a similar angle to Corbett & Anderson but adheres to a strictly Bayesian formulation. New criticism of the model will also be presented as well as novel suggestions for improvement.

**Student Individualization (multistep)**

The individualization model used in the KDD Cup competition used a multistep training process of individualizing the student parameters whereby a separate model was first trained for each student and then combined with a model trained for each skill (Pardos & Heffernan, in press). This resulted in $U + S$ models being trained where $U$ was the number of students and $S$ was the number of skills.

The first step was to learn parameters for each student. In standard Knowledge Tracing, skill parameters are learned by training from a dataset where the rows are different students who have provided responses to the skill and the columns are the students’ answers to the skill at different opportunities. To train student parameters, the dataset was transformed to have the rows be
different skills a particular student has provided responses to and the columns be the student’s responses to those skills at different opportunities. Figure 1 shows the difference between a dataset organized for skill parameter training vs. one organized for student parameter training.

<table>
<thead>
<tr>
<th>Skill Dataset (Pythagorean Theorem)</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
</tr>
<tr>
<td>Christopher</td>
</tr>
<tr>
<td>Sarah</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Student Dataset (Christopher)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Responses</td>
</tr>
<tr>
<td>Addition</td>
</tr>
<tr>
<td>Pythagorean</td>
</tr>
<tr>
<td>Subtraction</td>
</tr>
</tbody>
</table>

Figure 1. Example datasets prepared for training skill parameters (above) and student parameters (below)

The result of the first step was a $P(L0)$, $P(G)$, $P(S)$ and $P(T)$ parameter fit for each student. The next step was to train per skill models that incorporated all of the student parameters. For simplicity of presentation here we will demonstrate incorporating only the individual student learning rate, $P(T)$, although the technique generalizes to the other parameters as well.

Figure 2 shows a Bayesian network approach to incorporating the individual student learn rates, represented in the $H$ node, into the skill model. In this step, $P(L0)$, $P(G)$, $P(S)$ and $P(T|H)$ parameters are learned per skill. The student parameters, $P(H|Student)$, are fixed to the values learned in step 1 and are constant for each skill model. They are stored in a Conditional Probability Table (CPT) belonging to the $H$ node, which is a binary node that stands for High-Learner. A student ID is included in each row of the skill response dataset in order to reference the appropriate individual student learn rate associated with the evidence. The individual learn parameters dictate the probability that the $H$ node is true or not. Since the learning rate per skill is conditioned on the value of the binary $H$ node, two learning rates per skill are trained; one for high-learners, $H$, and one for non-high-learners, $H$. The formula for calculating the probability of knowledge at the next opportunity, $P(L_{n+1})$, in this model is:

$$P(L_n|Response_n) + (1 - P(L_n|Response_n))P(T|H)P(H|Student)$$

The formulas for calculating the posterior probabilities and probabilities of correct answers do not differ from standard Knowledge Tracing.
The strength of this model is that it incorporates individual student learn rates into the model in a way that is massively parallelizable at each step. The student parameter models can be learned completely independently of one another, as can the skill models, after the student models have completed. This is of significant benefit to computation time if cluster resources are available and a large dataset is being processed, such as the 21010 KDD Cup datasets, one of which had 6,000 users and 900 skills.

There are several weaknesses to this parallelizable two-step approach, however. One is that the students must have answered a similar distribution of skills (by difficulty) in order for the individual student learning rates to be comparable to one another. For example, if an average learning rate student answers only skills which are easy to learn, she will likely receive a high individual learn rate. However, if a high learning rate student answers only skills which are difficult, she will have a learning rate lower than the other student but only because the two students completed skills of disparate difficulty. The second weakness is lack of normalization of the individual parameters when incorporated in the skill model. The effect of this is that the difference in a skill’s high-learner learning rate and not-high-learner learning rate can only be as large as the difference between the smallest and the largest individual student learning rate. The individual parameters must be normalized to allow for greater play in the skill learn rates. Normalizing probabilities is a concern, however, in the case where the trained model is applied to a new student with an individual learning rate that is higher or lower than the minimum or maximum pre-normalized student learning rate.

![Bayesian network of the multistep model which incorporates the individualized student learning rates](image)

**Figure 2.** Bayesian network of the multistep model which incorporates the individualized student learning rates.
**Student Individualization (single step)**

The two issues of 1) an equal skill distribution requirement and 2) lack of normalization in the *high-learner* node, which exist in the multistep model, can be addressed with a single step individualized model. This model trains skill and student parameters simultaneously. This allows for individual student parameters to be fit in the context of all skill models, thus no longer requiring equal skill distribution among students. It also allows for the individual student parameters, such as the learn rates in the *high-learner* node, to be of any magnitude between 0 and 1 that best fit the global model, instead of being limited to the minimum and maximum student $P(T)$ values. This serves to no longer confine the disparity between *high-learner* and *non-high-learner* conditioned skill learn rates.

![Bayesian Network of the Single Step Model](image)

**Figure 3.** Bayesian network of the single step model which simultaneously fits skill and student parameters

This single step model, shown in Figure 3, trains skill and student parameters simultaneously by adding a *Skill* node to the model, which is a multinomial node with values ranging from 1 to $M$ where $M$ is the number of skills in the training data. The skill parameters are made conditionally dependent on the Skill node, allowing for $P(G)$, $P(S)$, $P(T|H)$ and $P(L_0)$ parameters to be trained per skill, for all skills at once. A student ID as well as a Skill ID is included in the rows for the skill dataset to properly associate the evidence with both skill and student. The individualized student learn parameters in the *high-learner* node must be initialized to some values before training. This might appear to be an initialization and convergence problem for large numbers of students but this is no more a problem than was present in the multistep method. In both methods, the initial values of the student parameters can be set to the same value or initialized randomly within some plausible bound. The additional data present in this single step model should help constrain the parameter values and result in better overall model performance compared to the multistep method.

The drawback to this approach is that the model is fit not just in a single step but in a single training of EM. This means high single threaded compute time for EM convergence as well as high memory load, since the entire dataset is being fit to at once instead of a single user’s data or a single skill’s data at once as was the maximum load seen in the multistep method. One way in
which to reduce the data size while still fitting parameters for all students and skills is to cluster students and or skills at some K and only include the response sequences, or a sampling of response sequences, representative of the clusters during training. At K equal to M or N, the result would be equivalent to using all data. As K decreased, so should the model fit but a happy medium value of K should exist such that the data size is tractable and performance is still above that of the multistep model.

**Modeling the Effects of the Tutor**

Individualization at the student level tells us something interesting about the student; how fast they learn, how much they have retained from past instruction, but learning something about the tutor and how it affects learning can be more actionable as it sheds light on ways in which to improve instruction to better assist and assess the student.

**Individualization of Educational Content in the Tutor**

Before the effects of the tutor on learning can be measured, the difficulty of individual questions, or piece of educational content in the tutor, must be controlled for. In order to accomplish this, a separate guess and slip parameter can be fit for each question in a skill or problem set. Fitting separate guess and slip parameters per question modulates the difficulty and also the information gain among the questions. As described in the introduction section, guess and slip values closer to zero allow for lower uncertainty in the inference of the latent of knowledge. Different guess and slip values for each question allows for the appropriate amount of information, about whether or not a correct answer should translate to knowledge of the skill, to be gained from a response. A correct response and inference of knowledge should, by virtue of the HMM design, transfer to the next opportunity to answer the next question of the same skill. Therefore, the amount of information gain for each question, set through the guess and slip parameters, expresses the relative relation between performance and knowledge among the questions. The utility of individualization of question guess and slip is maximized when the order in which questions are presented to students is randomized for each student.

![Figure 4. Pythagorean theorem questions (A), (B) and (C)](image)

Consider the three Pythagorean theorem questions (A,B,C) in Figure 4. All three questions ask the student to find the hypotenuse length; (A) does so with a lake cover story, (B) uses a house cover story and (C) uses no cover story at all. They all have a button below the picture that provides the student with assistance if pressed. The first two questions provide help in the form of hints while the third question provides help in the form of step by step tutored problem solving, otherwise known as scaffolding. A dataset representing student answers to these questions might look like the following, in Figure 5, where the identifying letter IDs of the questions serve as the attribute values.
### Skill Dataset (Pythagorean Theorem)

<table>
<thead>
<tr>
<th></th>
<th>Responses</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>0 1 1</td>
<td>C A B</td>
</tr>
<tr>
<td>Chris</td>
<td>0 1 0</td>
<td>B A C</td>
</tr>
<tr>
<td>Sarah</td>
<td>1 1 1</td>
<td>A B C</td>
</tr>
</tbody>
</table>

**Figure 5.** Example dataset of student responses and question IDs serving as the attribute at each opportunity.

It could be imagined, given more data, that these questions vary in difficulty among one another, with question C being answered correctly 33% of the time, B being answered correctly 66% of the time, and question A being answered correctly 100% of the time. The model in Figure 6 shows how question level individualization of difficulty, via the guess and slip parameters, has been accomplished in a Bayesian network (Pardos & Heffernan 2011b).

**Figure 6.** Bayesian network of the Knowledge Tracing Item Difficulty Effect Model (KT-IDEM), showing the conditional dependence of $P(G)$ and $P(S)$ on Attribute.

In this model, the question node is conditionally dependent on the attribute value which changes at each opportunity and is representing the different Pythagorean theorem questions from our dataset example. Applying this model has shown to significantly benefit skill-builder problem sets (randomized) in the ASSISTments Platform as well as linear sequence Cognitive Tutor for Algebra except for skills in which very small amounts of data per problem exist to train the individual guess and slip parameters (Pardos & Heffernan 2011b). When greater than 6 data points existed per problem on average, the KT-IDEM model outperformed regular KT.

While this example describes individualizing question guess and slip based on question ID, any other attribute, such as answer field type (multiple-choice or fill in the blank, for example), could take its place as an attribute.
Now that the difficulty (or information gain) of each question is controlled for, the endeavor of measuring the learning effect of each question can be taken on. The \( P(T) \) parameter in Knowledge Tracing is the probability of learning between each opportunity. Imagine if instead of a constant \( P(T) \) at every opportunity, the probability of learning between opportunities was dependent upon which Pythagorean theorem question was just viewed. Since the questions also provide different tutoring, a difference in learning could be expected between them. The application of this intuition is shown in the model in Figure 7 (Pardos & Heffernan 2011).

Figure 7. Bayesian network of the Item Effect Model showing the condition dependence of \( P(T) \) on ‘A’ at \( n-1 \).

Relative question learning rate information can bring content with low learning value to the attention of content creators to either revise or replace. It also allows researchers to evaluate what aspects of the tutor are promoting student learning so that these aspects, such as effective pedagogy and content ordering, can be replicated.

Like the KT-IDEM model, this model is not limited to using question ID as the attribute values. In the question example, the tutorial help types of scaffold and hint could be the attribute values as was done in Pardos, Dailey & Heffernan (2012) where this model was used to evaluate the effectiveness of different tutorial strategies across different skill-builder problem sets. A learning gain analysis was also run on the data and the Bayesian model’s tutorial strategy learning rates correlated with the learning gains in 10 of the 11 problem-sets. Further research using in vivo experiment data to validate against is ongoing.

CONCLUSION

In this chapter we have overviewed techniques for individual student and tutor parameter incorporation into the Bayesian Knowledge Tracing Model and summarized work of ours that has demonstrated some of the potential in this approach. The Bayesian formulation of student and tutor modeling appears to be an elegant one for representing different hypothesis of how learning may or may not be taking place in the tutor.
ACKNOWLEDGEMENTS

This research was supported by the National Science foundation via grant “Graduates in K-12 Education” (GK-12) Fellowship, award number DGE0742503, and Neil Heffernan’s NSF CAREER grant, award number REC0448319. We also acknowledge the many additional funders of the ASSISTments Platform found here: http://www.webcitation.org/5ym157Yfr.

Special thanks to my Ph.D. committee and other faculty for their support and advice; David Brown, Carolina Ruiz, Joseph Beck, Bill Far, Michael Gennert and Brigham Anderson. Also thanks to WPI technical staff Mike Voorhis, Jesse Banning and Sia Najafi for accommodating the long and laborious compute jobs and administrative staff Christine Caron, Refie Cane and Tricia Desmarais for their moral support. Finally, thanks, of course, to my great friends, family and colleagues in the ASSISTments and Science ASSISTments labs including graduated Ph.Ds Leena and Mingyu and soon to be graduating students; Mike, Dovan, Yutao, Yue and Jozsef.

Future Work

We have shown how Bayesian causal models can be built to predict student future performance as well as assess qualities of items in the tutor. An area of future study is how the information obtained from these models could be used to optimize the selection of items within the tutor for some criteria. This task of item selection is considered the outer-loop of the tutor by Aleven et al. (2009) whereas the task of choosing the appropriate interventions and types of feedback within a problem is considered the inner-loop. Valuable criteria in education to optimize would be assistance and assessment. Optimizing for assistance would be selecting items that are most likely to result in the highest probability of learning the skill being tutored. Item and Item Order models could be used for making this selection. Optimizing for assessment would be selecting the items that are most likely to result in the most confident inference of knowledge about the student. KT-IDEM which models guess and slip per item could be utilized for this selection. Koedinger & Aleven (2007) cover the topic of when to assist and when not to assist in a tutor from the cognitive science perspective. This work of item selection and using Bayes models of items could be used to continue that work.

Optimizing Item Selection for Maximum Assessment

As chapter 8 described, computer adaptive testing (CAT) has covered this task of item selection to optimize assessment but it does so with the assumption that knowledge, the construct being measured, is not changing over time. A tutoring system is explicitly designed to increase knowledge over time and thus computer tutor data breaks this essential assumption required for CAT. In order to optimize assessment, students could simply be given items that do not provide feedback or other tutorial information; however, this is wasting students’ time in a tutor that could otherwise be spent learning. Furthermore, prediction of future external assessment tests has been shown to be equally as accurate when trained on strictly testing data as when trained on data where the student is learning throughout (Feng, Heffernan & Koedinger, 2009), thus it is important to devise a method of item selection optimization for assessment that works in an environment where the student’s knowledge is changing.

Knowledge Tracing is already a model that is intended for assessing students’ change in knowledge over time and repeated evaluation of its formidable prediction accuracy validates it as a good fitting model. In knowledge tracing; however, items of the same skill are not differentiated in any way since they all share the same guess, slip and learn parameters. In chapter 8, the KT-IDEM model was introduced which adds per item parameters of guess and slip. The guess and
slip parameters could be looked at, intuitively, as either representing the difficulty or information gain (discrimination) of an item. For the purposes of item selection, the representation of discrimination is most relevant since that is the quality which represents information gain, a concept synonymous with assessment.

Assessment in knowledge tracing happens through inference of the latent variable, knowledge. Since this is a stochastic model, the most information of a variable is known when it is furthest from 0.50 probability, or in other words, when it has the minimal entropy. Therefore, the algorithm to optimize assessment would be one that minimizes entropy on the latent knowledge node. The measure of entropy commonly used is Shannon’s Entropy, represented by H(X) named after Ludwig Boltzmann’s H-theory (entropy of an ideal gas) and defined as:

\[ H(X) = -p(X) \log_2 p(X) \]

Step 1: Calculate the reduction in entropy given a correct and incorrect response to every possible next item

\[ \Delta H_{i,r} = H(P(L_n)) - H(P(L_{n+1} | i, r)) \]

Where the reduction in entropy \( \Delta H \) is calculated for each item \( i \) in the set of possible next items and for each possible response \( r \in \{0,1\} \) to those items. The value of \( P(L_{n+1} | i, r) \) is calculated with:

\[
P(L_{n+1} | i, r) = \begin{cases} 
\frac{P(L_n) \cdot (1 - P(S_i))}{P(L_n) \cdot (1 - P(S_i)) + (1 - P(L_n)) \cdot P(G_i)} & \text{for } r = 1 \\
\frac{P(L_n) \cdot P(S_i)}{P(L_n) \cdot P(S_i) + (1 - P(L_n)) \cdot (1 - P(G_i))} & \text{for } r = 0
\end{cases}
\]

Values of entropy reduction have now been calculated but simply taking the \( i \) and \( r \) that maximize \( \Delta H \) might result in a response being selected that is not probable. For instance, the student may not know the skill yet, but a correct response to some next item would have minimized the entropy. Therefore, the uncertainty in prediction response should be considered.

Step 2: Calculate the probability of correctly answering each possible next item

\[ C_{i,1} = \frac{P(L_n) \cdot (1 - P(S_i)) + (1 - P(L_n)) \cdot P(G_i)}{1 - C_{i,0}} \]

Step 3: Calculate the item, \( i \), that maximizes the entropy reduction while taking in to account the posterior probability of the response.

\[ \arg\max f(i) = \sum_{r=1}^{1} \Delta H_{i,r} C_{i,r} \]

**Optimizing Item Selection for Maximum Learning**

While assessment is important in a tutoring system, particularly in determining when to stop tutoring a student, student learning is the primary focus. Using the Item Effect Model described in chapter 5, a probability of learning, \( P(T) \), can be fit for every item. An item selection algorithm could be defined based on maximizing knowledge, \( P(L_{n+1}) \). The algorithm could look something like the following:
In this formula, $P(T_i)$ represents the probability that a student will learn the skill from doing item $i$. With this selection optimization the same item would always be chosen unless there were a rule in the tutor preventing items from being seen twice. If this were the case, the algorithm would start with the item with highest probability of learning and select the next best at each step. As chapter 6 described, the $i$ could also represent a learning probability of a particular tutoring strategy that is associated with multiple items. In this case, the algorithm would select the same tutoring strategy every time (the one with the most learning) or go down the list of strategies one at a time. It is likely the algorithm would run out of strategies to choose from if only allowed to select a particular strategy once, so it likely that the tutor does not have a rule against a strategy being seen more than once. In either case, the sequence of strategies selected would likely not capture more optimal sequence for learning such as were reported by Rau, Aleven & Rummel (2010) in their in-vivo experiments. The experiments showed that it was better to give students a block of items of strategy A followed by a block of B than it was to interleave A and B; however, after experience with both A and B, students benefited from interleaving. This sort of pattern would not be found by a selection algorithm that does not take into account item order effects.

An item order effect model (Pardos & Heffernan 2009b) was developed and found ordering effects of pairs of items such that a probability $P(T_{ij})$ was fit to express that there was some probability of learning if a student answered item $i$ and then item $j$ next. A formula to maximize learning based on item order might look something like this:

$$\text{argmax } f(i) = P(L_n) + (1 - P(L_n)) \cdot P(T_i)$$

In this formula $i$ is the item that the student has just completed and $j$ is the next item being considered. It is possible that the maximum $P(T_{ij})$ is when $i$ and $j$ are the same item or strategy. In this case, the algorithm would keep selecting the same item or strategy. Furthermore, this algorithm would not be able to find interesting patterns such as the blocking then interleaved pattern. A research question worth investigating is if more complex patterns of practice leading to higher learning gains can be detected using the Bayesian causal models. The problem is a difficult once since there is not nearly enough data to capture all the possible sequences of strategies in most scenarios. For instance, given three possible strategies and a sequence length of eight, the total number of sequence is $3^8$ or 6,561 sequences. Multiple instances of a sequence would have to be seen to get a stable estimate of its learning gain, so that number would far higher to be useful. There is, however, an abundance of occurrences of subsequences in the data. It is worth investigating if advantageous longer sequences can be inferred by piecing together learning information about smaller subsequences.
APPENDICIES

APPENDIX A

Integer subtraction templates

Template 1: positive – negative
What is \((-12) - 21\)?

Here you have to subtract a positive integer from a negative one. You can add the absolute values of the numbers and keep them negative.

So for \((-12) - 21\), perform the following addition:
\[12 + 21\]

\[12 + 21 = 33\]

Thus,
\[(-12) - 21 = -33\]

Type in \(-33\)

Type your answer below (mathematical expression):

Submit Answer

Template 2: negative – positive
What is \((-12) - (-17)\)?

Here you have to subtract a negative integer from a negative one.
Whenever you have a negative integer minus a negative integer, you can change the minus negative sign to an addition sign.

\[\begin{align*}
\text{Comment on this hint}
\end{align*}\]

In the problem you have,
\[-12 - (-17) = -12 + 17\]

Now you have a positive integer added to a negative integer.
We can start at the positive value 17 and count back the negative value -12.

This is the same as doing 17 - 12.

So,
\[-12 - (-17)\]
\[= -12 + 17\]
\[= 17 - 12\]

(If the number being subtracted is larger then the answer will be negative!)

\[\begin{align*}
\text{Comment on this hint}
\end{align*}\]

\[17 - 12 = 5\]

Therefore,
\[-12 - (-17) = 5\]

Type in 5.

\[\begin{align*}
\text{Comment on this hint}
\end{align*}\]

Type your answer below (mathematical expression):

Submit Answer

Template 3: negative – negative
What is 10 - 12?

Here you have a positive integer subtracted from a positive integer.

You can start at the positive value 10 and count back the negative value 12.

You can count back by ones. If possible you can first count back by tens and then count back by ones.

(Here the number being subtracted is larger thus the answer will be negative!)

10 - 12 = -2

Type in: -2

Type your answer below (mathematical expression):

Submit Answer

Template 4: positive – positive
APPENDIX B

Experiment #1 (Ordering fractions and decimals):

Original question

One way to order the numbers from least to greatest is to convert them all to decimal.

Let’s start by converting 1/4 to a decimal. We divide 1 by 4 to get

1 ÷ 4 = 0.25

Condition A: Step by step Solutions feedback
APPENDIX C
List of research areas and work I published with colleagues in those areas during this Doctorate.

Clustering

Knowledge Representation

Student modeling individualization and prediction

Tutor evaluation (modeling content for its assessment and learning value)


REFERENCES


