Student Modeling within a Computer Tutor for Mathematics: Using Bayesian Networks and Tabling Methods

Yutao Wang
Worcester Polytechnic Institute

Student Modeling within a Computer Tutor for Mathematics: Using Bayesian Networks and Tabling Methods

by

Yutao Wang

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

August 2015

APPROVED:

Abstract

Intelligent tutoring systems rely on student modeling to understand student behavior. The result of student modeling can provide assessment for student knowledge, estimation of student's current affective states (ie boredom, confusion, concentration, frustration, etc), prediction of student performance, and suggestion of the next tutoring steps.

There are three focuses of this dissertation. The first focus is on better predicting student performance by adding more information, such as student identity and information about how many assistance students needed. The second focus is to analyze different performance and feature set for modeling student short-term knowledge and longer-term knowledge. The third focus is on improving the affect detectors by adding more features.

In this dissertation I make contributions to the field of data mining as well as educational research. I demonstrate novel Bayesian networks for student modeling, and also compared them with each other. This work contributes to educational research by broadening the task of analyzing student knowledge to student knowledge retention, which is a much more important and interesting question for researchers to look at. Additionally, I showed a set of new useful features as well as how to effectively use these features in real models. For instance, in Chapter 5, I showed that the feature of the number of different days a students has worked on a skill is a more predictive feature for knowledge retention. These features themselves are not a contribution to data mining so much as they are to education research more broadly, which can used by other educational researchers or tutoring systems.

TABLE OF CONTENTS

Chapter 1: Introduction

In this dissertation, several analysis and models that I have tried to improve student modeling are described. The main question is how various factors influence student performance. In Intelligent Tutoring Systems, the most common information that is gathered from students is student performances, student identity, skill identity. In my work, I used various modeling techniques to analyze the influence of different aspects of this information. Bayesian Networks, as a major method, is optimal in capture the temporal nature when modeling changing student knowledge. Other methods I used include regression models, which are good at integrating different factors to make predictions; tabling models, which are great in terms of time efficiency. The tutoring system we gathering our data is the ASSISTments platform, in which middle schools students practicing math problems in an environment that multiple hints and attempts might be allowed.

This dissertation is organized into eight chapters. Chapter 2 and 3 shows different attempts in improving student models by utilizing new factors: student identity and the assistance information. The assessments of models were done by compare the predicting accuracy of student performances. Most of the models made the assumption that there is an unobservable variable (latent) that affects performance: student knowledge, and that the accuracy of predicting student performance indicates the accuracy of estimate student knowledge. Chapter 4 broadening the task of estimate student current knowledge and look into the question of estimate student long term knowledge. Regression models were used to quickly grab the features that might be important. Chapter 5, is the experiments and analysis about affect detectors, focuses on improving current model to estimate another usually unobservable variable: student affective states.

Chapter 2: Individualization for Modeling Student Knowledge

2.1 The Student Skill model

One of the most popular methods for modeling students' knowledge is Corbett and Anderson's Bayesian Knowledge Tracing (KT) model (Corbett & Anderson, 1995). The original Knowledge Tracing model does not allow for individualization. Recently, Pardos and Heffernan (Pardos & Heffernan, Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing, 2010) showed that more information about students' prior knowledge can help build a better fitting model and provide a more accurate prediction of student data. Our goal was to further explore the individualization of student parameters in order to allow the Bayesian network to keep track of each of the four parameters per student: prior knowledge, guess, slip and learning. We proposed a new Bayesian network model called the Student Skill model (SS), and evaluated it in comparison with the traditional knowledge tracing model in both simulated and real world experiments. The new model predicts student responses better than the standard knowledge tracing model when the number of students and the number of skills are large.

This chapter has been published as a short paper at the following venue:

Wang, Y. & Heffernan, N. (2012). The Student Skill Model. In *Proceedings of the 11th International Conference on Intelligent Tutoring Systems*. Springer. pp 399-404. (Wang & Heffernan, 2012)

Introduction

One of the most popular methods for modeling students' knowledge is Corbett and Anderson's (Corbett & Anderson, 1995) Bayesian Knowledge Tracing model. The original Knowledge Tracing model does not allow for individualization. Several researchers have tried to show the power of individualization. Corbett and Andersen presented a method to individualize students' parameters with a two phase process and reported mixed results (Corbett & Bhatnagar, 1997). Recently, Pardos and Heffernan (Pardos & Heffernan, Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing, 2010) showed that by a single process Bayesian network model: the prior per student model, more information about students' prior knowledge can help better fit model and provide more accurate prediction of student data. The result is inspiring; however, the author only looked into the students' prior knowledge and didn't extend the individualization to the other aspects of student knowledge, such as guess rate or learning rate. Pardos and Heffernan (Pardos & Heffernan, 2011) also tried a method where they trained all four parameters per student in a pre-process, then took those values and put them into a per skill model to learn how the user parameters interacted with the skill. This method requires a two phase data process, which is complicated to use in real-world.

Our goal was to further explore the individualization of student parameters in order to allow the Bayesian network to keep track of all our parameters per student as well as skill specific parameters simultaneously. We proposed a new Bayesian network model called the Student Skill model (SS), and evaluated it in comparison to the traditional Knowledge Tracing model (KT) in both simulation and real data experiments. The new model predicts student responses better than standard knowledge tracing model when the number of students and the number of skills are large.

The Student Skill Model

The Knowledge Tracing model assumes that all students have the same probability of knowing a particular skill at their first opportunity, or guess/slip in one skill, or learning a particular skill even though students seem likely differ in these aspects. Our goal was to add individualization into the original Knowledge Tracing model.

The new model we proposed in this paper is called the Student Skill model. It can learn four student parameters and four skill parameters simultaneously in a single phase process. The model is shown in Fig. 2.1.1.

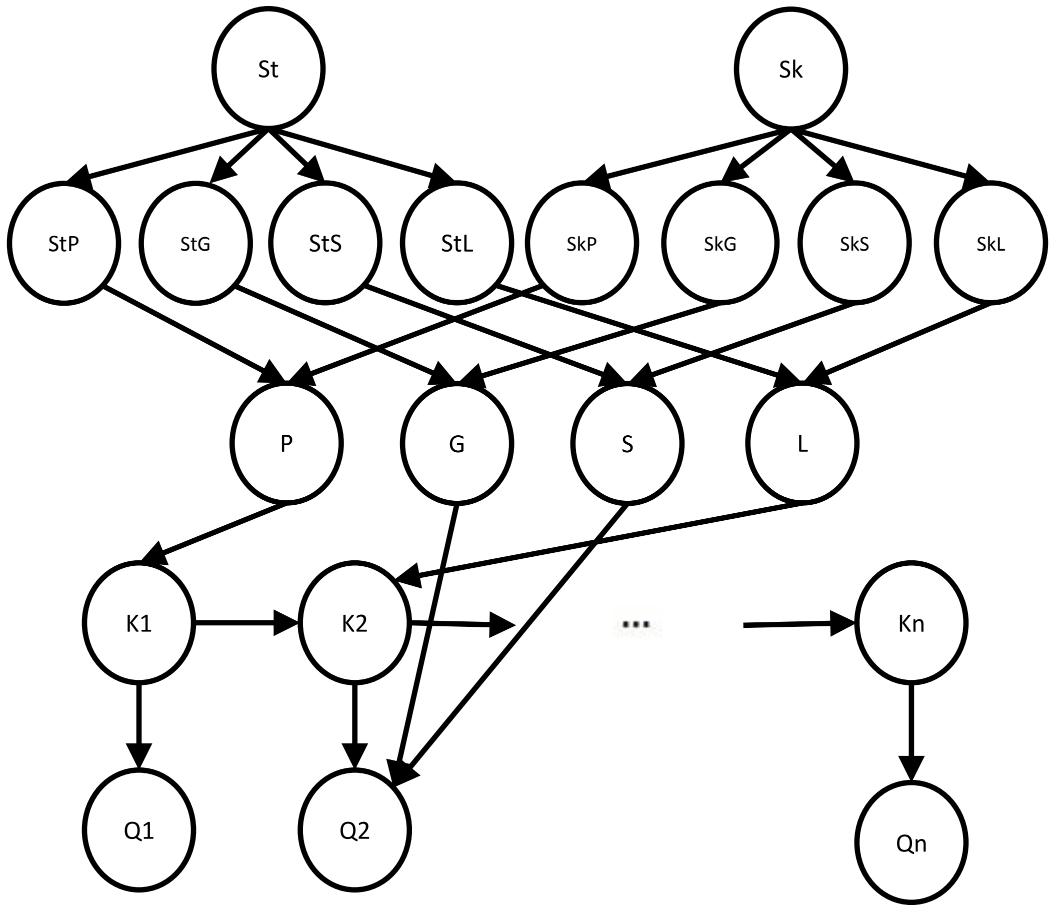


Fig. 2.1.1. The Student Skill model

The lowest two levels of this model are the same as the original Knowledge Tracing model (nodes K1~Kn and Q1~Qn in Fig. 2.1.1). The Student Skill model adds upper levels to represent the student and skill information and their interaction. We used two multinomial nodes to represent the identity of each student (node St in Fig.2.1.1) and each skill (node Sk in Fig. 2.1.1). Instead of pointing the student identity and the skill identity nodes directly to the knowledge nodes, which would result in a huge number of parameters, we added a level of nodes to represent the four student parameters (node StP, StG, StS and StL in Fig. 2.1.1) and the four skill parameters (node SkP, SkG, SkS and SkL in Fig. 2.1.1). Those parameter nodes are binary nodes that represent the high/low level of the corresponding parameters. For example, if the StP node is 1 for a student, then the student has high level of prior knowledge, and if the StP node is 0 for a student, means the student has low level of prior knowledge. The next level uses conditional probability tables to combines the influence of the student parameters and the skill parameters and generates the four standard Knowledge Tracing parameters (node P, G, S and L in Fig. 2.1.1) to be used in the lowest two levels.

The number of parameters in this model for n students and m skills can be computed as: $4\mathrm{n} + 4\mathrm{m} + 16$, while the number of parameters in the Knowledge Tracing model is: $4\mathrm{m}$. The cost of individualization is the additional $4\mathrm{n} + 16$ parameters.

Model Evaluation

The model is evaluated in both simulated and real data experiments. In our experiments, we used the Bayes Net Toolbox for Matlab developed by Murphy (Murphy, 2001) to implement the Bayesian network student models and the Expectation Maximization (EM) algorithm to fit the model parameters to the dataset. We choose initial parameters for each skill in Knowledge Tracing as follows: initial knowledge = 0.5, learning = 0.1, guess = 0.1, slip = 0.1.

Simulation Experiments

Methodology.

To evaluate the ability of the Student Skill model to function properly, in this experiment, we generated data from the Student Skill model and compared the prediction accuracy with the Knowledge Tracing model. The data records generated in the simulation represent student performances, with 1 representing correct and 0 representing incorrect. To simulate the random noise in the real data, we randomly flipped over 1% of the student performance data.

To split the training and testing data set, for each student, we randomly selected half of the skills data and put them into a training set. The remaining data went to the testing set. Both the Knowledge Tracing model and Student Skill model were trained and tested on the same dataset. A sequence of performances of given students and skills were predicted by both of these models.

Results.

Prediction accuracy is the selected metric for evaluating the results. In one simulation, the number of skills was set at 30 while the number of students was changed from 5 to 100 to observe the influence the number of student had on SS and KT respectively. Similarly, in another simulation, the number of students was set to be 30 while the number of skills was changed.

We observed that, in situations with a small number of students as well as those with a small number of skills, the Knowledge Tracing model outperformed the Student Skill model. However, when the number of students and the number of skills were increased, the performance of the Student Skill model improved and eventually exceeded the Knowledge Tracing model. The reason for this trend could be the fact that the Student Skill model contains more parameters than the Knowledge Tracing model, and with fewer data points, the model behaves less reliably.

We also compared the Student Skill model and the Knowledge Tracing model under different student parameter variance. The number of students and the number of skills were both set to 40, and the number of data points per student per skill was set to 10. The student variance was controlled by the real parameters used to generate simulated data. When the student variance was 0, all students shared the same parameters. We observed that the Student Skill model performs worse when there is no variance in student parameters. When the students are highly diverse, the Student Skill model outperformed the Knowledge Tracing model.

Real Data Experiments

One of the dangers of relying on simulation experiments is that the dataset may not reflect real-world conditions. Without evaluation using real data, the success of the new model during simulation could simply be caused by the data being generated from this model. To further evaluate the Student Skill model, we applied it to real datasets and again compared its performance with the Knowledge Tracing model.

Dataset.

The data used in the analysis presented here came from the ASSISTments platform, a freely available web-based tutoring system for 4th through 10th grade mathematics. We randomly pulled out the data of one hundred 12-14 year old 8th grade students and fifty skills from September 2010 to September 2011 school year. There are 53,450 total problem logs in the dataset.

Methodology.

The dataset was randomly split into four bins by student and skill in order to perform a four-fold cross-validation of the predictions and increase the reliability of the results. For each student, we made a list of the skills the student had seen and split that list randomly into four bins, placing all data for that student and that skill into the respective bin. There were four rounds of training and testing, during each round a different bin served as the test set, and the data from the remaining three bins served as the training set. Again, both the Knowledge Tracing model and the Student Skill model were trained and tested on the same dataset. A sequence of performances of the given students and skills were predicted by both of these models.

Results.

The accuracy of the prediction was evaluated in terms of the Root Mean Squared Error (RMSE). A lower value means higher accuracy. The cross-validation results are shown in Table 2.1.1.

Table 2.1.1. RMSE results of KT vs SS.

Fold ID	*SS*	*KT*	*P value*	*Student Level p value*
Fold1	0.4017	0.4055	0.0432	0.0404
Fold2	0.4194	0.4385	0.0459	0.0365
Fold3	0.4144	0.4348	0.0477	0.0451
Fold4	0.4441	0.4538	0.0420	0.0406
average	0.4199	0.4331	--	--

To test the reliability of the four folds experiment, we did a paired T test for each fold as well as the result of all the folds. The *p* value that compares the final RMSE of the SS model and the KT model of the four folds is 0.0439. The *p* value for each individual fold is shown in the fourth column. Our experiment shows that the difference between SS and KT is statistically significant, and the average RMSE shows that SS is more accurate than KT under our experimental conditions. We also did reliability analysis by computing RMSE for each student to account for the non-independence of actions within each student's dataset, and then compared each pair of models using a two tailed paired t-test. The Student Level *p* values are reported in the last column. All the results are statistically reliable.

Discussion and Future work

In this paper, we built a new Bayesian network model for modeling individual student parameters called the Student Skill model and compared it with the knowledge tracing model in both simulation and real data experiments.

In our experiments, we found that the Student Skill model is not always better than the Knowledge Tracing model. Under simulated conditions, we found that the new model is generally more accurate when the amount of students and skills are large. We are interested in other features that can indicate which model works better under what situations, in the hope that these two models can be combined in order to utilize both models' advantages.

Contribution

Several researchers have tried to show the power of individualization. Corbett and Anderson presented a method to individualize students' parameters with a two phase process: first run Knowledge Tracing on all the students and then run a separate regression to learn a set of slip, guess, learning and prior parameters per students. Pardos and Heffernan (Pardos & Heffernan, Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing, 2010) explored the individualized student prior, but did not learn all of the student parameters and skill parameters in one single model. We presented the SS model, which is elegant in accounting for individual differences (of learning rate, prior knowledge and guess and slip rates). Our simulation showed that we could reliably fit such a model. The simulation showed plausible results, such as that the SS model is better if more variation per student.

Our contribution is in presenting a model that allows us to use EM to learn parameters individualized to each student, while at the same time learn parameters for each skill. We presented simulation and real data experiments that showed this method can provide meaningful results. Knowledge Tracing is a special case of this model and can be derived by fixing the student parameters of the Student Skill model to the same values. In a practical sense, researchers need to figure out when the SS model can start to be used, as our simulation showed that SS is better than KT when 1) the number of skills a student has learned is high, and 2) the number of students is high.

2.2 Compare with Previous Method

One of the most popular methods for modeling students' knowledge is Corbett and Anderson's (Corbett & Anderson, 1995) Bayesian Knowledge Tracing (KT) model. The original Knowledge Tracing model does not allow for individualization. In this work, we focus on comparing two different individualized models: the Student Skill model and the two-phase model, to find out which is the best for formulating the individualization problem within a Bayesian networks framework.

This chapter has been published as a short paper at the following venue:

Wang, Y., & Heffernan, N. T. (2013) A Comparison of Two Different Method to Individualize Students and Skills. In *Proceedings of the 16th International Conference on Artificial Intelligence in Education*. pp. 836-840. (Wang & Heffernan, A Comparison of Two Different Method to Individualize Students and Skills, 2013)

Introduction

One of the most popular methods for modeling student knowledge is Corbett and Anderson's (Corbett & Anderson, 1995) Bayesian Knowledge Tracing model. The original Knowledge Tracing model does not allow for individualization. Recently, Pardos and Heffernan (Pardos & Heffernan, Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing, 2010) built a two phase individualization method where they trained four parameters per student at a pre-process, then took those values and put into a per skill model to learn how the user parameters interacted with the skill. This model is part of the final model that won the 2010 KDD Cup on educational data mining. The assumption this model made, which is we can learn student parameters first without any knowledge of skills seems unreasonable. Wang and Heffernan's work (Wang & Heffernan, The Student Skill Model, 2012) further explored the individualization of student parameters to allow the Bayesian network to keep track of four student parameters and four skill parameters simultaneously in one step in a model called the Student Skill model (SS), which seems more appealing to our desire for elegance. The goal of this paper is to answer two questions that this new individualization model raised. First, is this approach better than the two phase model that won the KDD Cup? And second, under what circumstances is it better?

Two Individualization Models

Fig.2.2.1. shows Pardos and Heffernan's two phased model. To train this model, the first step was to learn student parameters by using the Prior Per Student (Pardos & Heffernan, Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing, 2010) model by training on all skill data for an individual student one at a time. The second step was to include all of the student specific parameter information into a model, shown in Fig. 2.2.1 to learn skill related parameters.

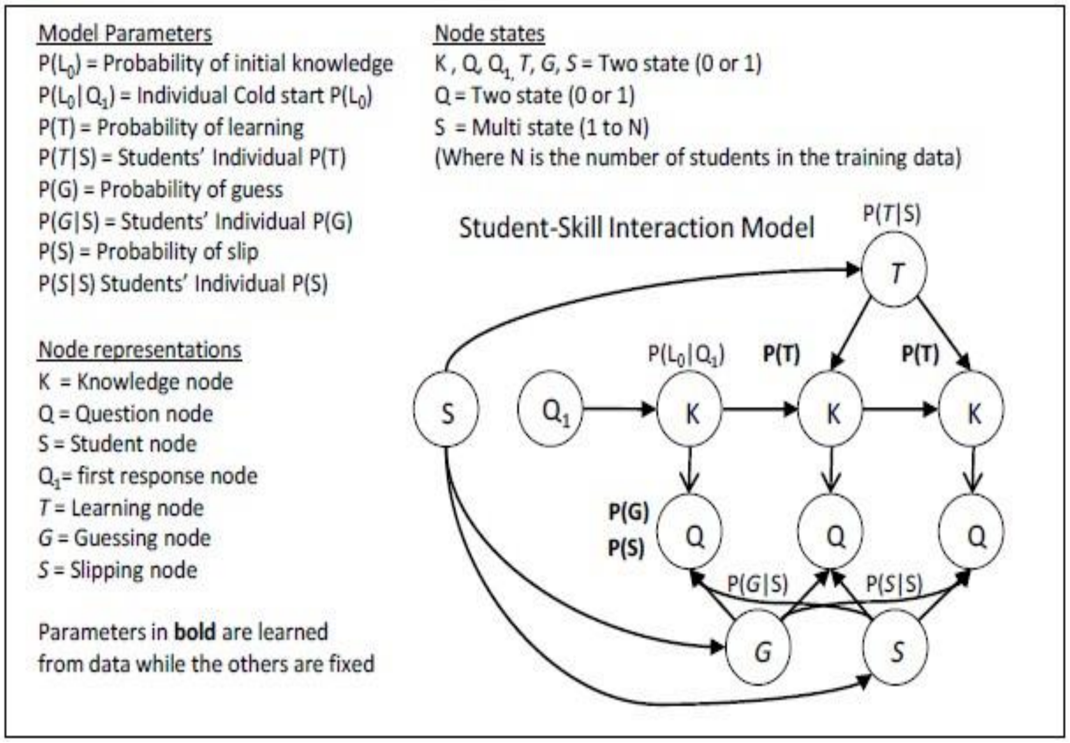

Model Parameters
P(L0) = Probability of initial knowledge
P(L0|Q1) = Individual Cold start P(L0)
P(T) = Probability of learning
P(T|S) = Students' Individual P(T)
P(G) = Probability of guess
P(G|S) = Students' Individual P(G)
P(S) = Probability of slip
P(S|S) Students' Individual P(S)
Node representations
K = Knowledge node
Q = Question node
S = Student node
Q1= first response node
T = Learning node
G = Guessing node
S = Slipping node
Parameters in bold are learned from data while the others are fixed
Node states
K, Q, Q1, T, G, S = Two state (0 or 1)
Q = Two state (0 or 1)
S = Multi state (1 to N)
(Where N is the number of students in the training data)
Student-Skill Interaction Model
P(T|S)
T
P(L0|Q1)
P(T)
P(T)
S
Q1
K
K
K
P(G)
P(S)
Q
Q
Q
P(G|S)
P(S|S)
G
S

Fig. 2.2.1. The Two Phase Model. Pardos &Heffernan (Pardos & Heffernan, 2011)

The second model that allows for individualization is called the Student Skill (SS) model (Wang & Heffernan, 2012). It can learn four student parameters and four skill parameters simultaneously in a single phase process. The model is shown in Fig. 2.2.2.

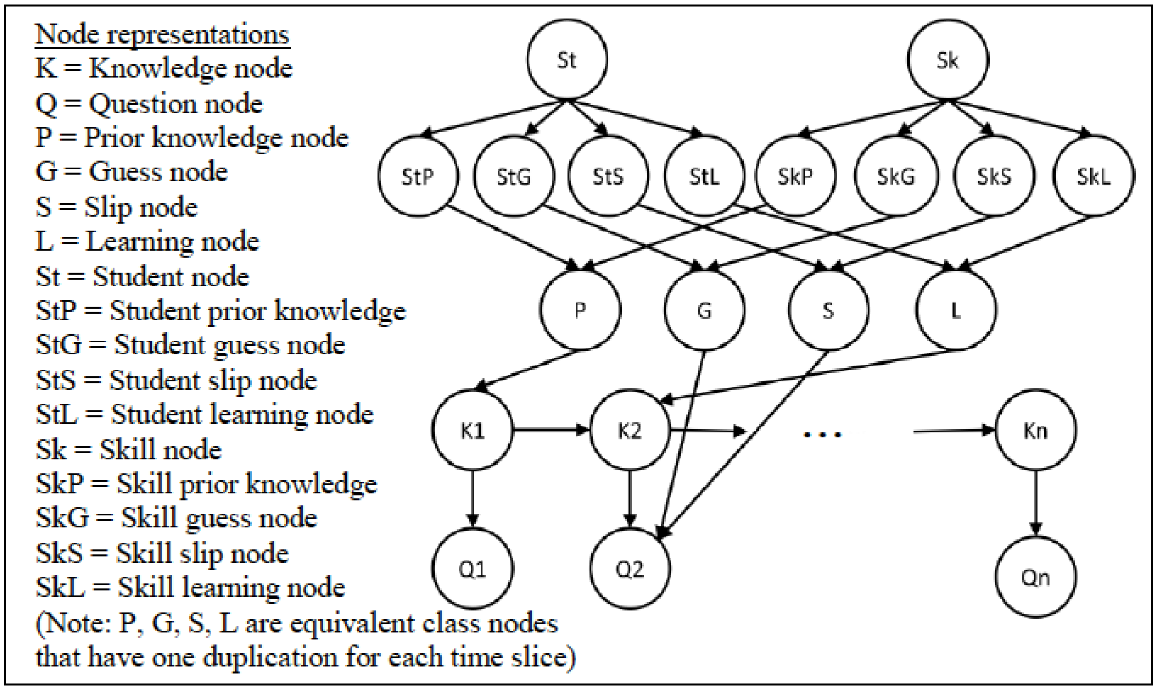


Fig. 2.2.2. The Student Skill model

Experiments

The two models were compared in both simulated and real data experiments. Only the real data result is reported here, simulation result is similar.

The data used in the analysis came from the ASSISTments platform, a freely available web-based tutoring system for 4th through 10th grade mathematics. We randomly pulled out data of one hundred 12-14 year old 8th grade students and fifty skills from the school year September 2010 to September 2011. There are in total 53,450 problem logs in the dataset. The dataset was randomly split into four bins in order to perform a four-fold cross-validation. For each student, we made a list of the skills the student had seen and split that list of skills randomly into four bins, placing all the data for that student for that skill into the respective bin. There were four rounds of training and testing where at each round a different bin served as the test set, and the data from the remaining three bins served as the training set. Both models were trained and tested on the same dataset.

The accuracy of the prediction was evaluated in terms of the Root Mean Squared Error (RMSE). Lower value means higher accuracy.

General Data Experiment

The purpose of the general data experiment was to determine which of the two individualization models works better in a real world Intelligent Tutoring System datasets. The cross-validation results are shown in Table 2.2.1.

Table 2.2.1. RMSE of SS vs 2-phase

Fold ID	*SS*	*2-phase*
Fold 1	0.447	0.452
Fold 2	0.438	0.451
Fold 3	0.422	0.420
Fold 4	0.445	0.446

The average RMSE of the Student Skill model is 0.438, which is better than the Two Phase model 0.442. However, paired t-test result has $p > 0.05$, which indicates that the result is not statistically reliable.

Filtered Data Experiment

The assumption we tried to verify in this experiment is that, in the first phase of the two phase model, when the model tries to determine which are the student parameters without knowing the skill information, the students that have done only easy skills will be more likely to get "better" parameters (better here means indicating he/she is a good student) than the students who have done only hard skills, and this inaccuracy in estimating student parameters would affect the Two Phase model's results, and causes a difference in model performance compared to the Student Skill model.

We filtered our dataset according to our assumption through the following steps and then compared the two models again on this filtered dataset.

a) Group skills to hard/medium/easy using percent correctness, in order to ensure that skills are very different, we threw out the medium group and kept only the hard and easy group skills;

b) Find student group A that contains students who have done both hard and easy skills;

c) Find student group B who have done only hard skills;

d) Find student group C who have done only easy skills;

e) Randomly select equal numbers of students from all three groups and use the data logs that are from only the hard and easy skills to build the dataset.

The cross-validation results are shown in Table 2.2.2.

Table 2.2.2. RMSE result of SS vs 2-phase in Filtered Real Data Experiment

Fold ID	*SS*	*2-phase*
Fold 1	0.423	0.428
Fold 2	0.425	0.423
Fold 3	0.419	0.427
Fold 4	0.423	0.423

The average RMSE of the Student Skill model is 0.423, which is better than the Two Phase model 0.426. The paired t-test on the prediction residual of all of the data points has $p < 0.05$, which indicates that the two models do perform differently when we filtered the data.

Conclusion

In this paper, we were able to show that the two different individualized Knowledge Tracing models perform similarly in general, yet different under certain circumstances.

2.3 The Most Important Parameter to Individualize

The intelligent tutoring system field is concerned with ways of personalizing to the student. The current best work coming out of personalization is the Student Skill model. They compared their model to the state-of-the-art Knowledge Tracing model, and achieved a reliable improvement over standard Knowledge Tracing. One limitation of their work is that they only investigated one particular way of personalizing, which individualizes all four standard Knowledge Tracing parameters: Prior, Learn, Guess and Slip simultaneously. But are all four of these parameters equally good at predicting student learning? It may be better if we just use some of the parameters to personalize the model. More generally, we wanted to address the research question: What are the most important features to personalize? In this work, we first attacked several issues with the Student Skill model and made reliable improvements. On top of that, we systematically explored all possible ways of incorporating student features into the model. We found that prior and slip are the two most important features, and the best model is the one with all four parameters individualized. Additionally, the one parameter that can be dropped without any hurt to performance is guess.

This chapter is the longer version of a poster that has been published at the following venue:

Gu, J., Wang, Y. & Heffernan, N. T. (2014). Personalizing Knowledge Tracing: Should We Individualize Slip, Guess, Prior or Learn rate? In *Proceedings of the 12th International Conference on Intelligent Tutoring Systems*. pp 647-648. (Gu, Wang, & Heffernan, 2014)

Introduction

The traditional way of modeling student knowledge is Corbett and Anderson's Knowledge Tracing (KT) model (Corbett & Anderson, 1995). Pardos and Heffernan (Pardos & Heffernan, Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing, 2010) proposed the Prior Per Student model, which learns a parameter that represents student prior knowledge for each student. By allowing a single parameter per student in this way, they allowed their model to be "individualized" on student prior knowledge. They reported that this model was better than KT in both predictive accuracy and convergence properties. That work led to Wang and Heffernan's idea that if we individualize all four of the standard KT parameters (Prior, Learn, Guess and Slip), we might get an even better performance. They introduced the Student Skill (SS) model (Wang & Heffernan, 2012) and showed that it was better than both KT and Pardos and Heffernan's model. The largest limitation of their work is they only investigated one particular way of personalizing, which personalized all the four student features simultaneously, and their model ignored the constraints of data sparsity. It's always going to be the case that if you have lots of data per student, the benefit of individualization will be good. But that leaves open the question: if you have a small amount of data per student, what should you attempt to individualize first? If you have even more data, what should you individualize secondly? And more generally, is individualization always going to be good, or is it the case that for best performance, it's better to individualize just a subset of the four parameters? So our research questions are RQ1: What is the one single parameter that is best to individualize upon? RQ2: Which parameter can be dropped without hurt to the performance? RQ3: What the best model if we individualize two of the parameters? Finally, we want to answer the critical research question: what are the most important features to personalize? Furthermore, if not all the parameters are essential, by reducing the number of parameters in the model, the complexity of the model fitting procedure can also be dramatically reduced. The practice of this work is that, as MOOCs and other large providers of educational data are making predictions upon log data, we suggest that they personalize their system based upon our work.

Model

The Knowledge Tracing model is one of the most popular ways of modeling student knowledge, which is based on two knowledge parameters: learning rate and prior knowledge, and two performance parameters: guess rate and slip rate. The Student Skill model adds student individualization into the original Knowledge Tracing model. In Wang & Heffernan (Wang & Heffernan, 2012), they said, “Knowledge Tracing is a special case of this model and can be derived by fixing the student parameters of the Student Skill model to the same values.” While this is roughly true, but it does not give the exact procedure and there are extra parameters learned that have no equivalent in the traditional way of doing Knowledge Tracing. In detail, it used four interaction nodes to reduce the number of parameters in the Conditional Probability Table (CPT). As a result, the CPTs of the knowledge nodes were set to fixed functions of the learning node and the knowledge node at the previous time step. Although it significantly reduces the number of parameters, the performance could also be compromised.

To overcome the shortcomings of the original SS model, we extended the number of the interaction node and the number of each student and skill feature node to the number of time slices. The structure of the improved SS model is shown in Fig. 2.3.1. The parameters of knowledge nodes and performance nodes become fixed functions of their parent interaction nodes. For example, the CPT of the learning node is set to [0, 1], where the probability of learning is directly derived from the interaction learning node. The CPTs of the four interaction nodes, which can have one or two parent nodes, are the only CPTs that change for different structures we explored. If the interaction node has only one parent node, its CPT is set to [0, 1], and it just passes the same parameters of the student/skill features. If the interaction node has two parent nodes, its CPT is set to [0, 0.5, 0.5, 1], where the second and third parameters are learned to differentiate the weights for student and skill features.

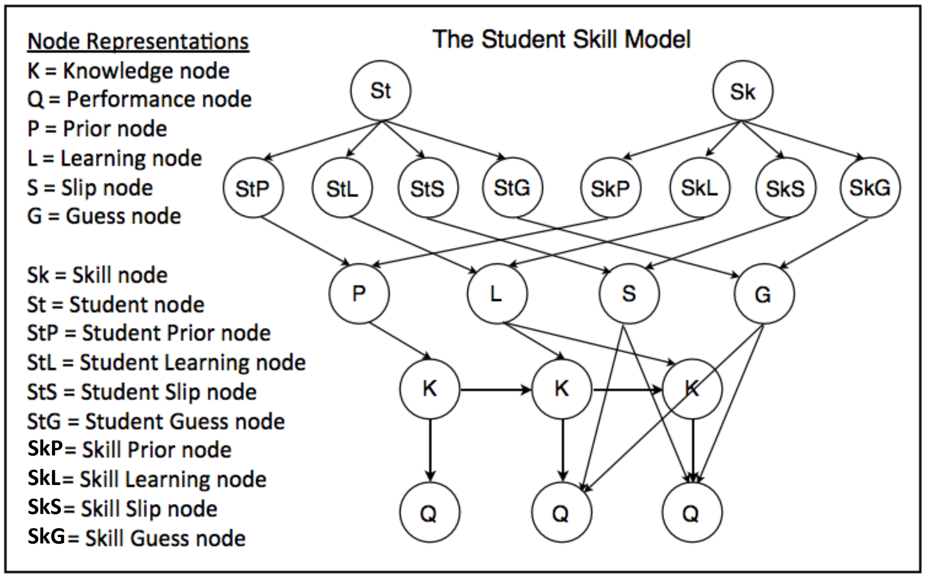


Fig.2.3.1. Student Skill model.

Methodology

The goal of our experiment is to search the best structures of the SS model. In the SS model structure, both the student and the skill node have four feature nodes (prior, learning, guess and slip). The interaction nodes combine the feature nodes for student and skill. If we want to exclude one feature from the student or the skill node, we can simply remove the link from the feature node to its corresponding interaction node in the Bayesian Network. Thus, by employing different combinations of links from the feature nodes to the interaction nodes, the models we want to examine can be constructed. Our first hypothesis is that, as the traditional way of constructing the Knowledge Tracing model is to fit each KT model for each skill, all the four features for skill should always be included. Thus, we tried all of the 16 possible ways of selecting the student features with all four skill features incorporated. The model with all four features from student is the original SS model, and the one with no features from student is the counterpart of the KT model. Furthermore, we also suspected that for each parameter we might want to choose between skill and student rather than take both. Therefore, we also tried another 16 ways of selecting either one of the features from student or skill. In total, we tried 32 different ways of incorporating features from student and skill and validated the 8 best structures using a larger dataset.

Dataset

The dataset we considered come from the 2009-2010 school year of ASSISTments, which is a free online platform developed at Worcester Polytechnic Institute. In the system, a student attempts a number of problems while working through an assignment. We select those student-skill sequences with less than or equal to 10 attempted opportunities. To explore all the structures, we selected a dataset (D1) with 81 distinct students and 78 distinct skills with 113,672 data points. To further validate the result, we selected another larger dataset (D2) with 1775 distinct students, 123 distinct skills and 695,732 data points. For all the experiments, we used Expectation Maximization (EM) as the model fitting procedure. To insure the reliability of the results, a five-fold cross validation was performed for each experiment. Considering that a missing student or skill in either the training data or the test data would compromise the performance, we had to make sure both datasets contain proportionate number of students and skills. The approach we used is: for each skill, we assign different students into 5 folds.

The measures of performance we used are Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Area Under the Curve (AUC) and R Squared (R2). Larger AUC, larger R2, smaller RMSE, and smaller MAE indicate better performance. After doing cross validations, we also did a t-test for each of the measures between every two of the models.

Results

Performance of the improved SS model

To validate that the SS model has a reliably better performance, we compared the predictive performance of the improved SS model to those of the original SS model, the Skill model (The SS model with no student parameter individualized, which is just another way of doing KT), and the original KT model. Table 2.3.1 shows the results in terms of Root Mean Square Error (RMSE), Mean Absolute Error (MAE) and Area Under the Curve (AUC). We observed that the improved SS model has remarkably better performance in RMSE and AUC compare to the KT model, which confirmed the fact that the SS model is significantly better than the original KT model. Surprisingly, although the Skill model is just the counterpart of KT, it also showed reliably better performance than KT. The Student Skill model structure in some way might be better than the standard way of doing KT. Additionally, the improved SS model also showed better performance than the original one, which indicates that our way of individualizing is indeed beneficial.

Table 2.3.1. Performance of KT model

	RMSE	*MAE*	*AUC*
KT	0.452	0.325	0.707
Improved SS	0.426	0.358	0.736
Skill	0.427	0.365	0.737
Original SS	0.434	0.370	0.715

RQ1: What is the one single parameter that is best to individualize upon?

First, we listed all the models with only one feature individualized at the student level. Table 2.3.2 shows the predictive performance of the models. The models are showed in ascending order by their RMSE values. According to the results, if we only want to individualize one feature, we probably should individualize prior, as it has the best RMSE, MAE, AUC and R2 values among all the four models. Besides prior, slip is also an important feature, whose corresponding model only has slightly worse AUC and R2 values compare to the model with prior. According to this result alone, we think prior and slip are the two most important features for personalizing.

Table 2.3.2. Performance of models with only one parameter individualized (P: Prior; L: Learn; G: Guess; S: Slip; St: the parameter was individualized at the student level)

RMSE	*MAE*	*AUC*	*R2*	*P*	*L*	*G*	*S*
0.433	0.370	0.692	0.102	St			
0.433	0.370	0.694	0.100				St
0.435	0.370	0.685	0.092		St		
0.435	0.372	0.685	0.092			St	

RQ2: Which parameters can be dropped without hurt to the performance?

We then tested all four models with only one feature missing from student. The model with the best performance is the one without the guess parameter individualized, which indicates that per student guess information is the least valuable. On the other hand, the model without prior and the model without slip are the two worst models in this case, which again indicates that slip and prior are the two most important features. This time, slip is slightly more important than prior as the model without slip is worse in terms of MAE, AUC and R2.

Table 2.3.3. Performance of models with three parameters individualized (P: Prior; L: Learn; G: Guess; S: Slip; St: the parameter was individualized at the student level)

RMSE	*MAE*	*AUC*	*R2*	*P*	*L*	*G*	*S*
0.431	0.368	0.699	0.107	St	St		St
0.432	0.370	0.696	0.103	St		St	St
0.432	0.370	0.695	0.103		St	St	St
0.432	0.369	0.692	0.102	St	St	St	

RQ3: What the best model if we individualize two of the parameters?

As shown in Table 2.3.4, if we only want to individualize two parameters, prior and slip would be the choice. Again, this is consistent with the previous two results, that prior and slip are the two most important features. Additionally, we noticed that the best models always had slip feature individualized, which seemingly indicates that slip is even more important than prior.

Table 2.3.4. Performance of models with two parameters individualized (P: Prior; L: Learn; G: Guess; S: Slip; St: the parameter was individualized at the student level)

RMSE	*MAE*	*AUC*	*R2*	*P*	*L*	*G*	*S*
0.4312	0.3681	0.6987	0.1069	St			St
0.4323	0.3698	0.6955	0.1023			St	St
0.4325	0.3695	0.6946	0.1014		St		St
0.4326	0.3699	0.6930	0.1010	St		St	
0.4329	0.3692	0.6921	0.0998	St	St		
0.4343	0.3709	0.6863	0.0942		St	St	

Validation of the top models

The differences between the best models and the SS model with all four parameters individualized in previous experiments are not statistically reliable. We selected the best models, and validated their performance using a much larger dataset containing 1755 students. We want to check if there's enough data for each student, whether or not the results still hold. As we can observe from Table 2.3.5, the fully individualized SS model is ranked first in the table, indicating that if we have enough data for each student, in order to gain a better performance, we should always individualize all four parameters for student.

Table 2.3.5. Performance of the best models (P: Prior; L: Learn; G: Guess; S: Slip; St: the parameter was individualized at the student level)

Model Index	*RMSE*	*MAE*	*AUC*	*R^2*	*P*	*L*	*G*	*S*
1	0.426	0.358	0.736	0.166	St	St	St	St
2	0.426	0.358	0.736	0.165	St	St		St
3	0.426	0.360	0.735	0.162	St		St	St
4	0.427	0.360	0.734	0.160	St			St
5	0.430	0.366	0.727	0.150		St	St	St
6	0.434	0.370	0.715	0.133				

Furthermore, we did a t-test between every pair of the top models, and reported the ones that were reliably different (p value < 0.05) in Table 2.3.6, where the model indexes are the same as in Table 2.3.5. As we can see, model 1 (the fully connected SS model) and model 2 (the one only without student guess parameter), model 1 and model 3 (the one only without student learn parameter) were not reliably different in some of the metrics, which indicates that guess and learn are the two parameters not important to personalize. This, again, implies that slip and prior are more important. Overall, it is hard to distinguish if one model is better than its adjacent models in the table. However, the difference is palpable if we compare the models on the top to the models on the bottom.

Table 2.3.6. t-tests among the best models (R: reliably different in RMSE; A: reliably different in AUC; M: reliably different in MAE; r: reliably different in R2 All: reliably different in all four measures)

Model Index	**2**	**4**	**5**	**3**	**1**
6	All	All	All	All	All
2		R M r	All	R M r	A r
4			All	R M r	All
5				All	All
3					R M r

From the results, we can see if we only individualize one feature for student, prior or slip would be the most important feature to personalize. Comparably, slip is even more important than prior, because almost all the models with slip individualized at both skill and student level has significantly better RMSE, MAE and AUC values than those of other models. Also, the feature slip can capture the ceiling effect of student knowledge, so it is possible that different students have different upper bounds of knowledge. Furthermore, of all these models evaluated, the best model was not reliably different with the one with prior, learn and slip individualized at the student level. The result indicates that students' guessing abilities do not differ a lot.

Models with complimentary features for student and skill

Finally, we also tested the performances of the models containing features individualized at either the student or the skill level. The total number of models is 16 as both student and skill have four features. According to our results, the model with only individualized slip for student is marginally superior to the baseline model, although not reliably different, which again indicates that student slip is indeed a very important feature to personalize. As expected, the baseline model ranked second among all the models and is significantly better in R2 value than the one that individualized all four features at student level.

Contributions

In this paper, we investigated the research question: which features of student are most important to individualize in a Bayesian Knowledge Tracing framework. The paper makes several contributions to large-scale student modeling.

Many researchers have shown the effect of personalization, but no one in the ITS field has looked at what parameters are most important to individualize. For example, the Student Skill model has shown significantly better performance than the standard KT model, but was only personalized in one particular way: personalized all four student features simultaneously. We extended the work by exploring more structures of the model and searched for the best way of personalization. The results show that if we only individualize one feature for student, the most valuable feature would be slip or prior. It is reasonable that students' prior knowledge differ greatly. Since slip represents the probability of a wrong answer given the student knows the skill, teachers or tutoring systems may need to pay attention to the students with large slip rates to check if they lose interest after mastering a skill or if they are still confused with some aspect in the skill while already mastered the major part of it, and take different actions accordingly.

Secondly, the single best model is the one with all four parameters personalized for student, but is not reliably different than the one without student guess, which also means guess is dispensable if we don't want to individualize all four parameters because of lack of information of students.

In addition, by attacking several problems with the original Student Skill model, we also made reliable improvement to the model. And interestingly, we found that the SS model without any individualization at the student level, which should be just KT, is actually reliably better than the normal KT model. This indicates that our way of modeling student learning is successful.

Future Work

Our finding that prior and slip are so important is a novel contribution. But we did not answer the question, why is this so? What is it about prior and slip that gives this extra boost in precision? This raises new question about what might be better ways to individualize.

The largest limitation of this work is that it has only been evaluated on one large dataset from ASSISTments. It is possible that the results may differ when using data from other tutoring systems. Furthermore, the fitting procedure of the SS model takes a long time and cannot be used in real time. Considering that the reliably better performance of the SS model is still worthwhile, the tradeoff between number of parameters in individualization and the time cost of the model fitting procedure is worth further exploration. Finally, we are also considering individualize the parameters at school, class and teacher level instead of just student, because we suppose that some features like the prior knowledge may differ more between different classes and schools.

In our scheme of individualization, it is an all or nothing approach: either each student is individualized on a parameter or not. But this of course is unwise, and if you had an algorithm that worked in real time, you would want to start out by using skill level parameters only; slowly over time as you accumulate more data on students' prior and slip rates, you would want to start to use individualized parameters. Fundamentally, the goal of individualization in real time is the Holy Grail of individualization, and our field needs a lot of work to be able to come up with an algorithm that will work in real time to better model student knowledge.

2.4 Class vs. Student in the Student Skill model

For decades, intelligent tutoring systems researchers have been developing various methods of student modeling. Most of the models, including two of the most popular approaches: Knowledge Tracing model and Performance Factor Analysis, all have similar assumption: the information needed to model the student is the student's performance. However, there are other sources of information that are not utilized, such as the performance on other students in same class. This chapter extends the Student-Skill extension of Knowledge Tracing, to take into account the class information, and learns four parameters: prior knowledge, learn, guess and slip for each class of students enrolled in the system. The paper then compares the accuracy using the four parameters for each class versus the four parameters for each student to find out which parameter set works better in predicting student performance. The result shows that modeling at coarser grain sizes can actually result in higher predictive accuracy, and data about classmates' performance is results in a higher predictive accuracy on unseen test data.

This chapter has been published at the following venue:

Wang, Y., Beck, J.E. (2013) Class vs. Student in a Bayesian Network Student Model. In *Proceedings of the 16th International Conference on Artificial Intelligence in Education*. pp. 151-160. (Wang & Beck, 2013)

Introduction

Student modeling is crucial for Intelligent Tutoring Systems (ITS) to improve and to provide better tutoring for students. For decades, researchers in ITS have been developing various methods of modeling students. Two of the most popular approaches are Bayesian Knowledge Tracing (KT) (Corbett & Anderson, 1995), which uses a dynamic Bayesian Network to model student learning, and Performance Factor Analysis (PFA) (Pavlik, Cen, & Koedinger, 2009), which uses a logistic regression to predict student performance. Both techniques have a similar underlying assumption that two things are needed to model the student: one component concerns the domain, such as skill information in KT and PFA models, or item information in the PFA model; the other component is the student's problem solving performance on the skill.

However, there are other sources of knowledge that are not utilized, such as the performance of other students in the same class. Instead, only *this* student's previous performances are taken into account. Imagine there is a class of 20 students, 19 of whom get the first item on a skill wrong, and you want to predict the performance of the 20th student's first item on the skill. Intuitively, predicting that this student would also respond incorrectly seems like a safe bet. However, current student models such as KT and PFA will not be affected by those 19 incorrect responses, as they were all made by other students. What would the effect on predictive accuracy be if which class a student is currently in was factored into student models? Our intuition is that class perhaps contains important information such as the student's prior knowledge about a skill. Since all students in a class share a common teacher, curriculum, and assigned homework problems, we should expect similarities in performance. Our goal is to capitalize on this dependency to improve student modeling.

In fact, the US Institute for Educational Sciences requires grant proposals' power analyses to discount the sample size if there are multiple students in the same classroom, due to their lack of independence from each other (most statistical tests require each sample to be independent). Given that we know this dependence effect exists statistically, why not make use of it? In this paper, we are focusing on utilizing the class information to improve student modeling and trying to determine under which circumstances, using other students' information could be more beneficial than using current student's individual information.

Approach

This section briefly introduces the Student Skill model and the modification of it in order to allow class level individualization. The modified model also allows us to run experiments on various combinations of student and class information to determine whether or not the class information is better than the student information for each parameter.

Model

Knowledge Tracing is one of the most popular methods for modeling student knowledge. The original Knowledge Tracing model do not allow for individualization, and assumes that all students have the same probability of knowing a particular skill at their first opportunity, or slipping (making a careless mistake) on a skill, or learning a particular skill. This assumption is almost certainly invalid, as students are likely to differ in these aspects. Several researchers have tried to show the power of individualization (Wang & Heffernan, 2012) (Pardos & Heffernan, Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing, 2010). The model we use in this work is built upon one of the individualization model called the Student Skill model (Wang & Heffernan, 2012). The idea of the Student Skill model is that rather than estimating a learning rate for each skill, instead view learning rate as being a function of the skill and of this individual learner. Perhaps some skills are learned more quickly or slowly than others, and perhaps some students learn more quickly or slowly than others. By combining both effects, it is possible to more accurately model the student.

The Student Skill model structure is shown in Fig. 2.4.1. The goal of the Student Skill model is to add individualization into the original Knowledge Tracing model. It can learn four student parameters and four skill parameters simultaneously. The lowest two levels of this model are the same as the original Knowledge Tracing model (nodes K1…Kn and Q1…Qn in Fig. 2.4.1). The Student Skill model adds upper levels to represent the student and skill information and their interaction. Two multinomial nodes are used to represent the identity of each student (node St in

Fig. 2.4.1) and each skill (node Sk in Fig. 2.4.1). Instead of pointing the student identity and the skill identity nodes directly to the knowledge nodes, which will result in an exponentially increasing number of parameters, we instead added a level of nodes to represent the four student parameters (node StP, StG, StS and StL in Fig. 2.4.1) and the four skill parameters (node SkP, SkG, SkS and SkL in Fig. 2.4.1). Those parameter nodes are binary nodes which represents the high/low level of the corresponding parameters. For example, if the StP node is 1 for a student, means the student has high level of prior knowledge, and if the StP node is 0 for a student, means the student has low level of prior knowledge. Then the next level combines the influence of the student parameters and the skill parameters and generated four standard Knowledge Tracing parameters (node P, G, S and L in Fig. 2.4.1) to be used in the lowest two levels. In this way, we generate a knowledge tracing model that is custom-fit to each learner and for each skill.

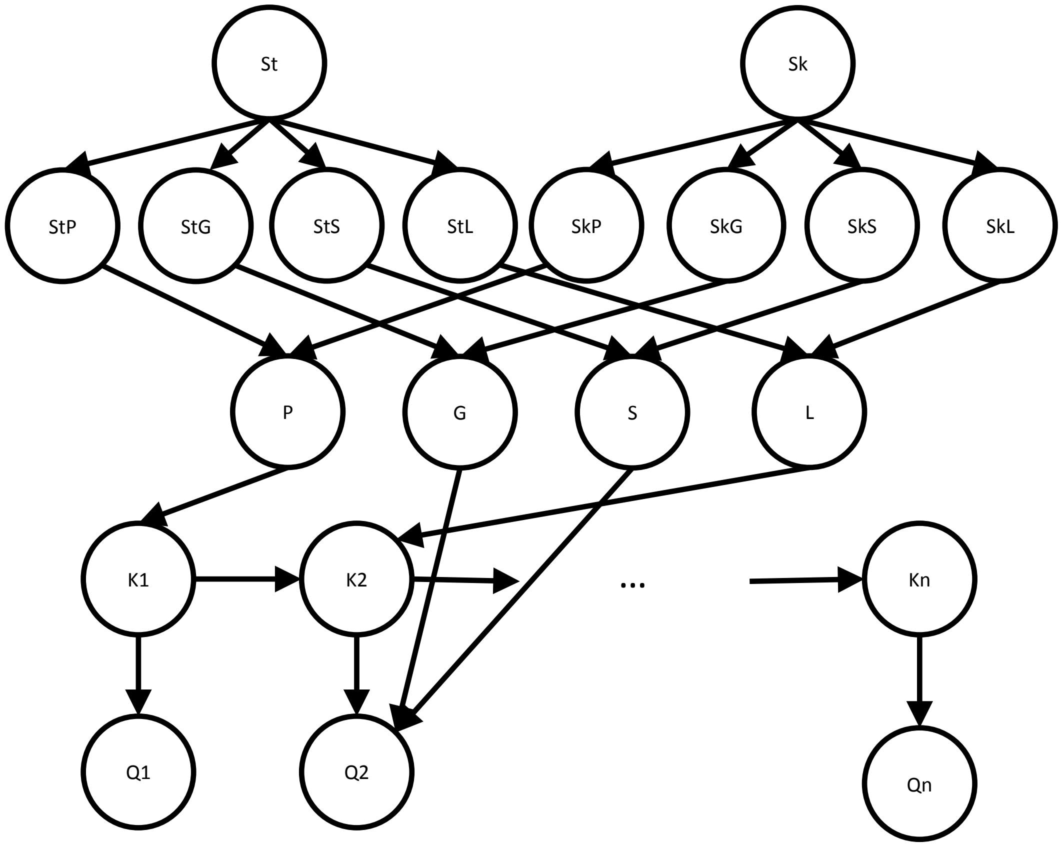


Fig. 2.4.1. The Student Skill model

One drawback of the Student Skill model is that it requires a large number of parameters. In addition to estimating four parameters per skill, it must also estimate four parameters per student. Given that many datasets have considerably more users than skills, this inflation in the number of parameters is a large concern. Therefore, we considered methods for reducing the number of parameters in our model, to enable them to better generalize to unseen data. One approach is, rather than modeling the students as individuals, to instead model which mathematics class the student is enrolled in. Students within the same class have the same teacher, textbook, homework, and may even be grouped by ability in the subject. Given that, in our datasets, there are typically about 24 students per class, modeling class-level effects has 24 times as much data to estimate parameters. In addition, if we only model class parameters, we only have to estimate 1 set of parameters for each *class* of students, rather than 1 set for each individual students. Thus, the use of class information can be seen as a coarser grain-size individualization compared to the Student Skill model. We demonstrate the Class Skill model in figure 2.4.2, and the nodes are identified as follows:

— St: A multinomial node represents each student’s identity, observable.

— Sk: A multinomial node represents each skill’s identity, observable.

— StP: Student Prior Knowledge, binary node, latent.

— StG: Student Guess rate, binary node, latent.

— StS: Student Slip rate, binary node, latent.

— StL: Student Learning rate, binary node, latent.

— SkP: Skill Prior Knowledge, binary node, latent.

— SkG: Skill Guess rate, binary node, latent.

— SkS: Skill Slip rate, binary node, latent.

— SkL: Skill Learning rate, binary node, latent.

— P: Prior Knowledge of a particular student and a particular skill, binary node, latent.

— G: Guess rate of a particular student and a particular skill, binary node, latent.

– S: Slip rate of a particular student and a particular skill, binary node, latent.

– L: Learning of a particular student and a particular skill, binary node, latent.

– K1~Kn: Knowledge, binary node, latent.

– Q1~Qn: Question performance, binary node, latent.

The Student Skill model can easily be changed to consider the class information rather than the student information by replacing the St node to be a class node (Cl), and the parameters StP, StG, StS and StL will be turned into class prior (ClP), class guess (ClG), class slip (ClS) and class learning rate (ClL).

Instead of simply using class information to replace the student information, which is still considering only one resource of information, this paper combines these two models together to explore whether knowing which class a student is in is a better predictor than knowing which student, for each parameter in the model. For example, perhaps slip rate is best modeled at the individual student level, while learning rate is best estimated at the class level? Therefore, we have run experiments with different ways of combine the two resources of information trying to determine which parameter is best modeled using which source of information.

As shown in Fig. 2.4.2, the model is almost the same as the Student Skill model in Fig. 2.4.1. The only difference is the addition of the class (Cl) node, which is a multinomial node, represents which class a student is in. Nodes StP, StG, StS, StL turns into StP/ClP, StG/ClG, StS/ClS, StL/ClL, which means the nodes can either be a student level parameter or a class level parameter. The dash line between node Cl and node StP/ClP is a potential relationship in the model, as well as the dash line between node St and node Stp/ClP. If we choose one of these two dash lines, the other one will be ignored as if it does not exist. For example, if we choose to use class information for prior knowledge, the dash line between St and node StP/ClP is ignored, and the node StP/ClP only contains the class prior (ClP). The same assumption is hold for all the other dash lines and parameters of class and student: StS/ClS, StG/ClG, StL/ClL.

Based on this model, by choosing different dash lines, we can test the best combination of class and student parameters and find the variability.

In our experiment, we used the Bayes Net Toolbox for Matlab developed by Murphy (Murphy, 2001) to implement the Bayesian network student models and the Expectation Maximization (EM) algorithm to fit the model parameters to the dataset. The EM algorithm finds a set of parameters that maximize the likelihood of the data by iteratively running an expectation step to calculate expected likelihood given student performance data and a maximization step to compute the parameters that maximize that expected likelihood.

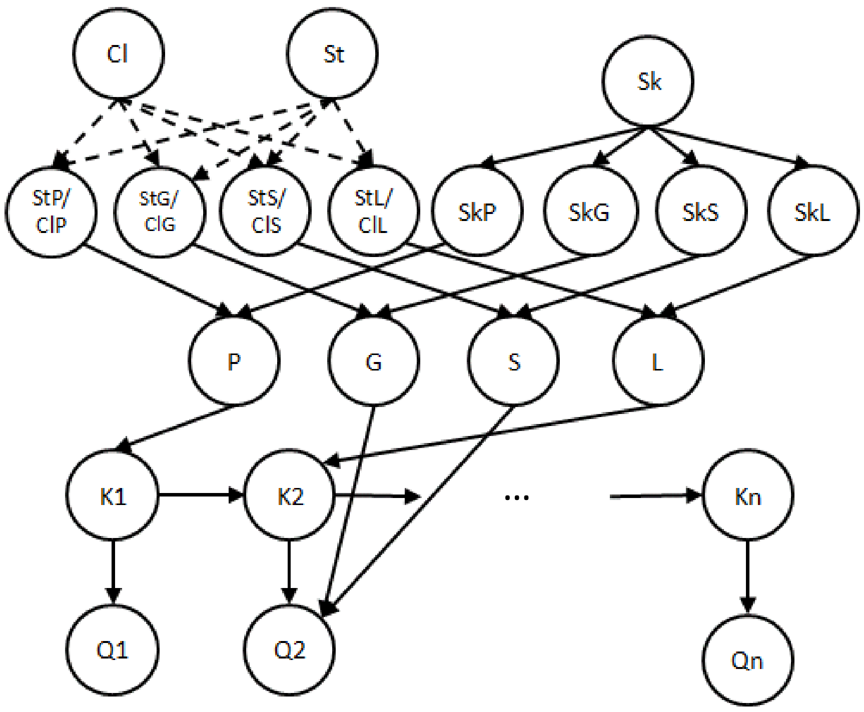


Fig. 2.4.2. Combination of Class Skill model and Student Skill model

Data and Model-fitting

The data used in the analysis presented here came from the ASSISTments platform (www.assistments.org), a freely available web-based tutoring system for 4th through 10th grade mathematics. The performance of a question is marked as wrong if the first response is incorrect, or if the student asks for help.

We randomly sampled data of one hundred 12-14 year old 8th grade students from 4 classes and fifty skills from the school year September 2010 to September 2011. There are in total 53,450 problem solved in the dataset.

To make sure there were sufficient data in the training set to estimate parameters for students and skills, we divide the dataset into a training set and a test set using the following strategy: for each student, for every skill that she was practicing we flipped a coin and assigned this student-skill pair into either the training set or into the testing set. This process enables us to have a broad coverage of students and skills in the training set, to enable generalization to the testing set. However, we do not have data for the same student-skill pair in both the training and in the testing data. In this way, we maintain a relatively independent test set, but still enable our approach to see enough types of data to estimate all of the required parameters.

In the experiment, we estimate each knowledge tracing parameter using data about the skill, and either data about this student's or the student's classmates' performance on this skill. Thus, for each parameter we tried two ways of estimating its value. We examined each combination of settings for all four knowledge tracing parameters (P, G, S, L) To simplify the problem, we group the performance parameters, guess and slip, together. This leaves us in total 2^3= 8 different combinations in parameters. The models and experimental results are shown in the next section.

Results

The accuracy of the predictions was evaluated in terms of the Root Mean Squared Error (RMSE), with lower values meaning higher accuracy. We compared different models to analyze the best individualization level for prior Knowledge (K0), learning rate (L) and Guess and Slip (G/S) respectively. That is, for each of the parameters (K0, L, G/S), we choose Class level individualization or Student level individualization, there are in total 8 possible combinations. The different combination models and their RMSE results on the test set are shown in Table 2.4.1.

The first column shows which parameter is chosen for the prior knowledge, the second column shows which parameter is chosen for the learning rate, the third column shows which parameter is chosen for the performance parameters (guess and slip), the fourth column shows the RMSE result of each model on the test dataset. We order the rows in this table based on the RMSE on the test set, with the top rows representing higher accuracy on the test set.

Table 2.4.1. RMSE result on test and training data

K0	*L*	*G/S*	*RMSE*
Class	Student	Class	0.413
Class	Class	Class	0.415
Class	Student	Student	0.417
Class	Class	Student	0.419
Student	Student	Student	0.421
Student	Student	Class	0.423
Student	Class	Class	0.424
Student	Class	Student	0.425

For comparison, the standard Knowledge Tracing model produces an RMSE of 0.428 on the test data, which is less accurate than all of the models we experimented with in Table 2.4.1. Therefore, it appears that both of the class level and the student level individualization can help improve Knowledge Tracing’s predictive accuracy.

A second point of comparison is our baseline Student Skill model, represented in the 5th row in this table (underlined), which represents estimating all of the parameters using information about each student. Thus, each student has a customized estimate of prior knowledge (K0), learning (L), and guess (G) and slip (S), as they are derived from the student node. In this case, model in Fig.

2.4.2 degenerates to be the same as the Student Skill model in Fig. 2.4.1. The fact that this model is only at the middle of the table shows that, it is not as strong as other methods of estimating parameters.

In other words, sometimes it is better to use the class information rather than using individual student information. This result could occur if students within a class do not vary very much on a particular parameters. In that case, it would be better to estimate that parameter for the entire class to take advantage of the larger quantity of data. For example, the fact that the 4th row, which has prior and learning comes from class information, and guess and slip comes from the student information results in lower RMSE value on the test data than the 5th row, indicates that the prior knowledge and learning rate may be better estimated through the class information rather than estimated from completely individualization of student. Back to the example at the beginning of this paper, this means that for prior knowledge, and guess and slip rate, knowing the information of all of the other students in the class may be slightly more beneficial than only knowing the information of the current student. If all of the other students in the class do not know a skill initially, it is more likely the current student do not know the skill either, no matter how good the student is on other skills.

Among all of these models, the best mode (the first row in the table) is the one with prior knowledge (K0) and performance parameters (guess and slip) derived from the class information, and the learning rate (L) is derived from individual student information. The result seems plausible because all students in a class normally get the same instruction, thus might have similar prior knowledge (K0) about a particular skill, and some students learn faster than others, thus the learning rate (L) would be beneficial from individual student information. To be clear, we are not asserting that all students have the same prior knowledge, as some students will not complete homework or might not pay attention in class. However, within a class, prior knowledge varies less than the other parameters, and, at least in this instance, the potential benefit of customizing K0 to each student is not worth the additional parameters.

Besides finding the best combination of grain-sizes for estimating various parameters, there are also some interesting general trends visible in Table 2.4.1. The most interesting one is that prior knowledge (K0) is always better modeled at the class level: the top 4 rows are all with class information used to estimate the K0 parameter. This result confirms our intuition that all students in a class tend to have similar prior knowledge, which could be caused by the fact that they are going through similar instructions, or the fact that similar students are tend to be assigned to the same classroom.

The trend in learning rate (L) is the opposite as the trend for prior knowledge. Since the bottom two rows both have class information as the resource for learning rate, student information seems to be a more powerful resource. Therefore, within a class, students' ability to learn mathematics appears to vary more than their prior knowledge. However, these differences appear to be rather small: comparing the first and second lines results in a difference in RMSE of 0.002; similarly, comparing the third and fourth lines also results in a difference in RMSE of 0.002. This difference is rather small, so estimating learning rate at the class level or at the student level works approximately equally well.

As for the performance parameters (guess and slip), there seems to be a general advantage to modeling these effects at the class level, but the trend is not completely clear. We expected guess and slip behaviors to vary considerably within a class, and to be better modeled at the student level. Therefore, we found this result somewhat surprising.

Contributions, future work, and conclusions

This paper makes three main contributions. Philosophically, it considers the learner's classmates as a viable source of information for predicting the learner's behavior. This source of information seems to have been overlooked by the ITS community.

The second contribution this paper makes computational, as it extends the Bayesian knowledge tracing framework to take into account the class information. Our model structure enables us to model parameters at the class- or student-level, and to mix and match grain sizes within an experiment. In a similar effort, a PFA-like model was modified to account for class-level information (Xiong, Beck, & Li, 2013).

The third contribution this paper makes is empirical. Our results suggest that initial knowledge of a skill is probably best modeled at the class level. Prior work either assumed the initial knowledge is determined either by the skill itself or a combination of the student and skill. This paper's experimental results suggest that student modelers should consider additional sources of power for understanding learners.

Currently, the way we utilize the student and class information is to consider using either class parameters or student parameters. That is, each of the models we compared considered using one source of power for each of the parameters, but not both. It is possible that we can look at both sources information simultaneously and even take into account the fact that a student is a member of a class, to build a hierarchically structured model that blends the two sources of information together. In this model, class could be the parent node of different students. The model is easy for people to understand and interpret, yet we are not sure if a complex Bayesian Network representation of this model can be properly built and learn back the expected parameters. Both experiments with real and simulated data will be helpful for evaluating such approaches. It is also unclear if the model will be practical given the large number of parameters required.

One issue that we have not yet addressed is whether the performance parameters (guess and slip) should be grouped together. In this paper, we group the performance parameters together to simplify the experiments based on the assumption that these two parameters are both related to performance and should have similar properties with respect to the best grain size for modeling. Yet, it is likely that guess and slip behaves very differently at the class level compared to the student level. For example, some type of instruction may cause all students in the class very likely to guess the correct answer for some skills, even though the students do not fully understand the skill. We suspect that slip is best modeled at the individual level. The mixed result in the performance parameters could perhaps become clearer if we run more experiments with separate guess and slip parameters.

Another question that we are interested in exploring is whether the results about class-level parameters transfer across years? Currently, our evaluation looks at only one year's data and generates the test and training set from that year. This approach has the normal cold start problem, that if it is the start of a new school year and we have no information about the class yet, what would be a reasonable information to use to build the student model? One possible solution that we are interested in is to use the class information of previous school years. If we can find a class that we have data from previous years that is similar to a current class, we might be able to use the information from that class to start building the model for the current class. How to define similarity of different classes, however, is a challenging question. We could look at the teacher or use the very first performance of each student in the class as an estimate of prior knowledge. We could also choose a set of similar previous classes and use the average of their parameters instead of choose only one from all. Or, we could use whichever prior class has the highest predictive accuracy for this student, as in (Gong, Beck, & and Ruiz, 2012).

Finally, from a broader perspective, class can be seen as a group of students, thus is a natural way of clustering students. There are literatures that focus on clustering in student modeling such as

(Trivedi, Pardos, & Heffernan, 2011) (Song, Sarkozy, Trivedi, Wang, & Heffernan, 2013). What are the differences and connections between using class and using other clustering methods? Class could be an effect of the teacher or ability grouping; in this case, using clustering algorithms on features such as teacher and student ability could result in similar clusters as classes. There are also other levels of abstraction and natural clustering, such as which grade or school a student is in, exploring models that utilizing these new sources of information is also new and interesting.

In summary, this paper introduces a framework for using a dynamic Bayesian network to model parameters as a combination of student-skill effects, or class-skill effects. We have found that using either source of knowledge is more accurate than a standard knowledge tracing model. By selectively estimating some parameters at a coarser grain size, we are able to improve accuracy a bit over the class-skill model.

Chapter 3: The Predictive Power of the Assistance Information

3.1 Partial credit

Both Knowledge Tracing and Performance Factors Analysis, are examples of student modeling frameworks commonly used in AIED systems (i.e., Intelligent Tutoring Systems). Both of them use student correctness as a binary input, but student performance on a question might better be represented with a continuous value representing a type of partial credit. Intuitively, a student who has to make more attempts, or has to ask for more hints, deserves a score closer to zero, while students who asks for no hints and just needs to make a second attempt on a question should get a score close to one. In this work, we present a simple change to the Knowledge Tracing model and a simple (non-optimized) method for assigning partial credit. We report our real data experiment result in which we compared the original Knowledge Tracing (OKT) model with this new Knowledge Tracing model that uses partial credit as input (KTPC). The new model outperforms the traditional model reliably. The practical implication of this work is that this new technique can be widely used easily, as it is a small change from the traditional way of fitting KT models.

This chapter has been published at the following venue:

Wang, Y. & Heffernan, N. (2013). Extending Knowledge Tracing to allow Partial Credit: Using Continuous versus Binary Nodes. In *Proceedings of the 16th International Conference on Artificial Intelligence in Education*. pp. 181-188. (Wang & Heffernan, Extending Knowledge Tracing to allow Partial Credit: Using Continuous versus Binary Nodes, 2013)

Introduction

In many important student models, such as the Knowledge Tracing model and the Performance Factor Analysis (Pavlik, Cen, & Koedinger, 2009), student performance is presented as a binary value of correct or incorrect. The amount of assistance a student needed to eventually get a problem correct is ignored in these models. Feng and Heffernan (Feng & Heffernan, 2010) showed that we can predict student performance better by accounting for amount of assistance they received, but they did not provide the field with a model that could be used in “run time” to predict individual responses. Arroyo, et al. (Arroyo, Cooper, Burleson, & Woolf, 2010) showed how to use this information to predict learning gains. Their work suggests that using hints and attempts to model student behavior online could be effective.

There is good work in the psychometric literature on using partial credit, which goes back 30 years. Psychometricians have shown that different multiple choice answers might worth different credits (Masters, 1982) (Tang, 1996). For instance, choice A might be totally wrong but choice B is close, choice C is the correct answer.

More recently, a new type of partial credit is coming online. For instance, Attila and Powers (Attali & Powers, 2010) at the Educational Testing Service showed they could better predict student GRE scores if they let students make multiple attempts. Their score on a question would go down by a third for each attempt (students could only make three attempts). Our work generalizes their work in two ways. First, we show how to incorporate the partial credit score into a model with learning (i.e., Knowledge Tracing) as their model did not model learning. Second, we show how to incorporate penalties for each hint student request.

In our previous work (Wang, Heffernan, & Beck, 2010), we presented a naïve algorithm to assign partial credit, and showed it accounts for some variance in student knowledge. But in that work, we did not present a model that could do this task. In this paper we want to see if we can improve one of the dominant methods of student modeling (i.e., the Knowledge Tracing model) by relaxing the assumption of binary correctness: replacing the discrete performance node with continuous partial credit node.

Approach

Knowledge Tracing with Continuous Performance Node

The Knowledge Tracing model shown in Fig. 3.1.1 has been widely used in ITS to model student knowledge and learning over time. It has become the dominant method of student modeling and many variants have been developed to improve its performance (Baker, et al., 2010) (Pardos & Heffernan, Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing, 2010). Knowledge Tracing uses one latent and one observable dynamic Bayesian network to model student learning. As shown in Fig. 3.1.1, four parameters are used for each skill, with two for student knowledge (initial knowledge and probability of learning the skill) and the other two for student performance (the probability of guessing correctly when the student doesn't know the skill and the probability of slipping when the student does know the skill).

The structure of the Knowledge Tracing model with a continuous performance node is the same as the original Knowledge Tracing model. The only difference is how we set up the "Student Performance" node. The idea is straight forward, yet there has never been positive result reported in this field. Some other Intelligent Tutoring System groups, such as LISTEN (http://www.cs.cmu.edu/~listen/) tried this approach before but failed for unknown reasons.

In this model, instead of assign the "*Guess*" and "*Slip*" parameters in a CPT table as the original Knowledge Tracing model, we assigned two Gaussian distributions for "*Guess*" and "*Slip*" with given standard deviations. Four parameters: *guess_mu*, *guess_sigma*, *slip_mu*, *slip_sigma*, are used to describe the two Gaussian distributions.

Similarly, when we predict student performance, we also get a Gaussian distribution with a mean value and a standard deviation value, in which the mean value will be the prediction and the standard deviation contains the information of how good the prediction is. In this work, we are not using the standard deviation of the prediction, but it has potential to be useful in the future to determine how confident we are in our prediction.

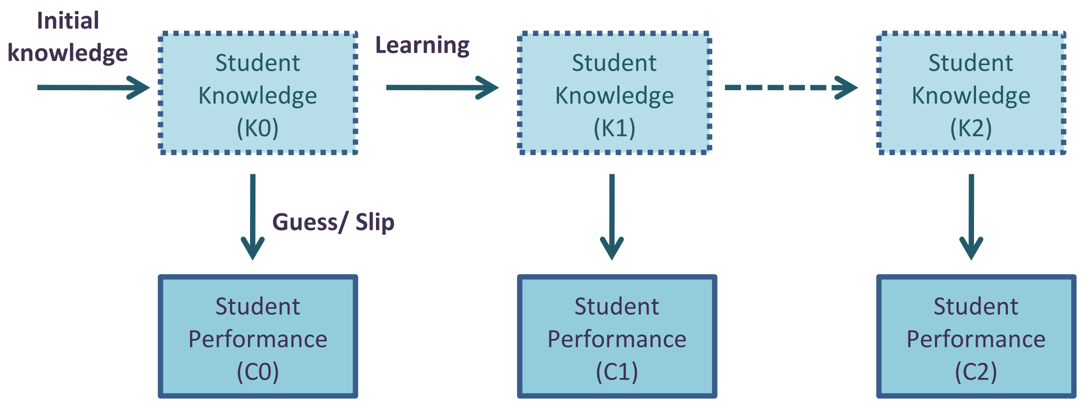


Fig. 3.1.1. Knowledge Tracing model

In our experiment, we used the Bayes Net Toolbox for Matlab developed by Murphy (2001) to implement Knowledge Tracing and the Expectation Maximization (EM) algorithm. The EM algorithm finds a set of parameters that maximizes the likelihood of the data. Since EM can be sensitive to initial conditions (Pardos & Heffernan, Navigating the parameter space of Bayesian Knowledge Tracing models: Visualization of the convergence of the Expectation Maximization algorithm, 2010), we report the initial settings. We used *initial knowledge* = 0.5, *learning* = 0.1, *guess_mu* = 0.1, *guess_sigma* = 0.02, *slip_mu* = 0.1, *slip_sigma* = 0.02 for the KTPC model, and *knowledge* = 0.5, *learning* = 0.1, *guess* = 0.1, *slip* = 0.1 for the OKT model.

Make the Correctness Continuous: Partial Credit

Partial credit can be assigned in different ways. In our experiment, we are using the algorithm that was mentioned in our previous poster (Wang, Heffernan, & Beck, 2010) to make the correctness to be continuous. Since we never introduce the algorithm completely, it is described in this section in detail.

In a model with binary performance, a student would get a ‘1’ if he/she answered the problem correctly on the attempt without asking for a hint and ‘0’ otherwise. For the purpose of this paper we created a scoring method that would give students a score between ‘0’ and ‘1’ according to how many attempts and how many hints they required to answer a question correctly based on intuition. We are well aware that this method could be optimized in lots of ways, for example, should each hint cost the same, or should the first hint cost less? As shown in our result, this simple method is effective and we leave to others different ways to optimize it.

Intuitively, the more hints that are asked for, the less likely it is that the student understands the skill, so we penalize a student for each hint asked for by what we call the hint penalty, which is 1 divided by the total number of hints available. For example, if there are 4 hints possible and a student asks for three of them and then gets the problem correct he/she would get a .25 score. In a similar manner, more attempts indicate a lower possibility of understanding the required skill, and we penalize each attempt. The size of the penalty depends upon whether the question type is “multiple choice” or “Fill in the Blank”. In our data set, we have about 80% questions that are “Fill in the Blank” questions, for which we picked a penalty 0.1 for each wrong attempt. For multiple choice questions with *x* choices, the penalty was computed by one over the number of remaining multiple choice options minus one. So a true false question will have a penalty of one if a student guessed wrong. If there were 4 choices, a student’s first wrong attempt would get a penalty of 1/3, a second wrong attempt would get a penalty of ½, and a third wrong attempt would get a penalty of 1.

After computing hint penalty (*phint*) for each hint and attempt penalty (*pattempt*) for each attempt, we add them together to compute the total hint penalty (*total_phint*) and the total attempt penalty (*total_pattempt*) for this problem. If the number is less than zero we make it zero. The last column of Table 3.1.2 shows two examples of formula doing this calculation.

Table 3.1.1 shows the details of computing partial credit for scaffolding questions.

Table 3.1.1. The algorithm of computing partial credit.

```
function pc = partial_credit(problem){
if first attempt correct then
return pc = 1
else if problem has no scaffold then
pc = 1 - #hint * phint – total_pattempt
if pc<0 then return pc = 0
return pc
else
for each scaffold question i in the problem do
pc_scaffold(i)  =  partial_credit(scaffold(i))
end for
pc = 0.9 * average(pc_scaffold(i))
return pc
}
```

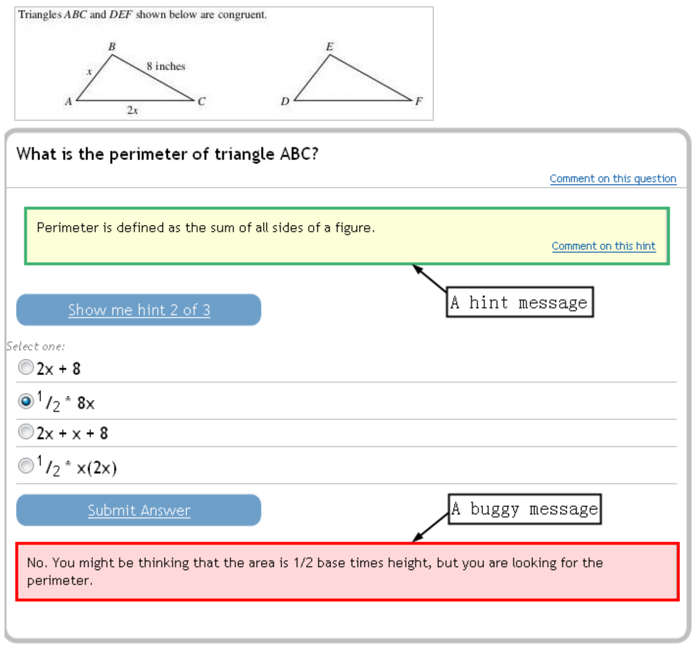
Our dataset has a special type of feedback called scaffolding. Since it’s only a small amount of our data this detail might not be that important. But for completeness, we wanted to describe this. For those problems with scaffolding questions, if a student gets the original question wrong, the system will give the student a series of questions we call “scaffolding” that walk the student through the steps. Each of these scaffolding questions has hints and so can be scored with this partial credit function just like normal questions. The only question left is how to score the “original question”. If a student gets a question wrong and is given three scaffolding questions, the total credit of the whole problem is computed by averaging the partial credit scores of the three scaffolding questions and penalized by 10% for answering the original question incorrectly. If a student got the original question wrong but then got all the scaffolding questions correct, he/she should get a score close to 1, which in our method would be 0.9. Again these parameters such as 0.9 are not optimized and could be learned from data in future work.

The algorithm is used only for testing the effect of relaxing the assumption of binary correctness in a Knowledge Tracing model.

Evaluation

The Tutoring System and Dataset

Our dataset consisted of student responses from ASSISTments, a web based tutoring system for 7th-12th grade students that provides preparation for the state standardized test by using released math items from previous years' tests as questions. The tutorial helps 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.


Triangles *ABC* and *DEF* shown below are congruent.

What is the perimeter of triangle ABC?

Perimeter is defined as the sum of all sides of a figure.

Select one:

- ○ $2x + 8$
- ● $^1/_2 * 8x$
- ○ $2x + x + 8$
- ○ $^1/_2 * x(2x)$

No. You might be thinking that the area is 1/2 base times height, but you are looking for the perimeter.

Fig. 3.1.2. Assistance in ASSISTment

Fig. 3.1.2. shows an example of a hint. A second type of assistance is presented if the student clicks on (or types in) an incorrect answer, at which point the student is given feedback that he/she answered incorrectly (sometimes, but by no means always, the student will get a context-sensitive message called "buggy message").

The data consisted of 52,529 log records during the period Jan 2009-Feb 2009 where each log record is similar to one row in Table 3.1.2, which shows the details of one problem done by one student. We use the same data format as the KDD Cup 2010: Educational Data Mining Challenge (https://pslcdatashop.web.cmu.edu/KDDCup/FAQ/#data-format). Table 3.1.2 shows an example of the type of data we used.

Table 3.1.2. An example of a few rows of data, showing how we calculate partial credit

1.Row	*2.Student*	*3.Problem*	*4.Step*	*5.Incorrects*	*6.Hints*	*7.Error Rate*
1	S01	WATERING_VEGGIES	(WATERED-AREA Q1)	0	0	0
2	S01	WATERING_VEGGIES	(TOTAL-GARDEN Q1)	2	1	1

8.Knowledge component	*9.Opportunity Count*	*10.Number of Choices (If Multiple Choice)*	*11.Total Hints Available*	*12.Partial Credit*
Circle-Area	1	4 Choice Multiple Choice	2	1
Rectangle-Area	1	Fill in the Blank	3	1-2*0.1-1*1/3=0.46

There are in total 12 columns, the first 9 columns in the table are straight from the KDD Cup data format (https://pslcdatashop.web.cmu.edu/KDDCup/rules_data_format.jsp), and we added three extra columns, which are used for partial credit. In particular, column 10 "Number of Choices (if Multiple Choice)" was added to describe if the problem is multiple choice problem or not, and how many choices there are. Total number of hints available for the problem is put in column 11, to help compute the partial penalty per hint. The last hint always gives away the answer, so if a student asked for all of the hints, their score should be zero. This column allows us to give a bigger penalty for hints if the number of total available hints is small. Column 12 is for showing how we compute the partial credit score, a continuous value between 0 and 1 that the student would get given the data log. Note that the original KT model will only use the 7th column, “Error Rate”, as model input; while the KT with partial credit model will only use the 12th column, “Partial Credit”. The 7th column is generated as 1 if the student answered the problem correctly, otherwise 0.

Results

To evaluate how well the new model fits the data, we used the Root Mean Squared Error (RMSE) to examine the predictive performance on an unseen test set. Lower values for RMSE indicate better model fitting. There were randomly 2,313 student data in the test set and 3,297 students in the training set.

Table 3.1.3 shows the result of the comparison of the two different models, the original Knowledge Tracing(OKT) model and the Knowledge Tracing with partial credit(KTPC) model.

Table 3.1.3. original KT (OKT) vs KT with partial credit (KTPC)

Model	***RMSE***	
	Partial	**Binary**
OKT	0.4128	0.4637
KTPC	0.2824	0.4572

We compared the RMSE in predicting the partial credit performance and in predicting the traditional binary performance respectively. The Knowledge Tracing with partial credit model has lower RMSE value in both situations. The lower left column shows that KTPC does a great job in predicting partial credit scores, which is expected. The top left cell shows that OKT can do some reasonable job of predicting partial credit scores. The more interesting result is the right column, which shows that OKT has higher RMSE than the KTPC in predicting binary performances.

We determined whether the difference between these two models is statistically reliable by computing the RMSE value for each student to account for the non-independence of student actions, and then compared these two models using a two tailed paired t-test.

The t-test *p* value of the RMSE between using the original Knowledge Tracing model and the Knowledge Tracing with partial credit model to predict the partial credit is 0. The *p* value between using the original Knowledge Tracing model and the Knowledge Tracing with partial credit model to predict the binary performance is $p < .001$. The degree of freedom of the t-test is 2,312 (since we are doing a student level t-test, the degree of freedom is the same as the number of students in the test set). Thus, the Knowledge Tracing with partial credit model is statistically reliably better at predicting student performance than the original Knowledge Tracing model.

Conclusions and Future Work

In this paper, we extended Bayesian Network student modeling to include continuous performance node. The effectiveness is demonstrated by incorporating a partial credit algorithm that assigns continuous performance given detailed student responses. Experiment results show that relaxing the assumption of binary correctness in student modeling can help improve predictions of student performance. This also proves that our intuition based heuristic for partial credit might be broadly applicable.

One topic we are interested in exploring is other partial credit schemes, for example, a method to refine the algorithm to generate partial credits that can better fit student data and more accurately infer student knowledge. Also, since we observed some abnormal parameters in the performance parameters (guess/slip), we are interested in finding out why the parameters are so different compare to normal Knowledge Tracing model.

Contributions

Moving from binary performance to continuous performance could make Intelligent Tutoring Systems more flexible. In this paper, on one hand, we extended the Knowledge Tracing framework to include a continuous performance node. This allows the Knowledge Tracing model to combine with all possible continuous performances such as essay score, speech recognition score. On the other hand, we presented an understandable and easy to refine algorithm to assign partial credit according to detailed student responses. This algorithm is one of many possible ways to convert student detailed responses into a continuous value.

The model presented in this paper enhanced student model accuracy by improving upon the classic Knowledge Tracing model. The result shows that the new model makes statistical reliably improvement in predicting both students' partial credit performances and binary performances. Also, freely available code is shared online, which could be useful for researchers that are trying to do the same task.

3.2 The Assistance Model

An important aspect of Intelligent Tutoring Systems is providing assistance to students as well as assessing them. The standard state-of-the-art algorithms (Knowledge Tracing and Performance Factor Analysis) for tracking student knowledge, however, only look at the correctness of student first response and ignore the amount of assistance students needed to eventually answer the question correctly. In this chapter, we propose the Assistance Model (AM) for predicting student performance using information about the number of hints and attempts a student needed to answer the previous question. We built ensemble models that combine the state-of-the-art algorithms and the Assistance Model together to see if the Assistance Model brings improvements. We used an ASSISTments dataset of 200 students answering a total of 4,142 questions generated from 207 question templates. Our results showed that the Assistance Model did in fact reliably increase predictive accuracy when combined with the state-of-the-art algorithms.

<u>*This chapter has been published at the following venue:*</u>

Wang, Y. & Heffernan, N. (2011). The "Assistance" Model: Leveraging How Many Hints and Attempts a Student Needs. In *Proceedings of the 24th International FLAIRS Conference*. (Wang & Heffernan, 2011)

Introduction

Understanding student behavior is crucial for Intelligent Tutoring Systems to improve and to provide better tutoring for students. For decades, researchers in ITS have been developing various methods of modeling student behavior using their performance as observations. One example is the Knowledge Tracing model (Corbett & Anderson, 1995), which uses a dynamic Bayesian network to model student learning. A second, called Performance Factor Analysis (Pavlik, Cen, & Koedinger, 2009), has recently been outperforming Knowledge Tracing (Gong, Beck, &

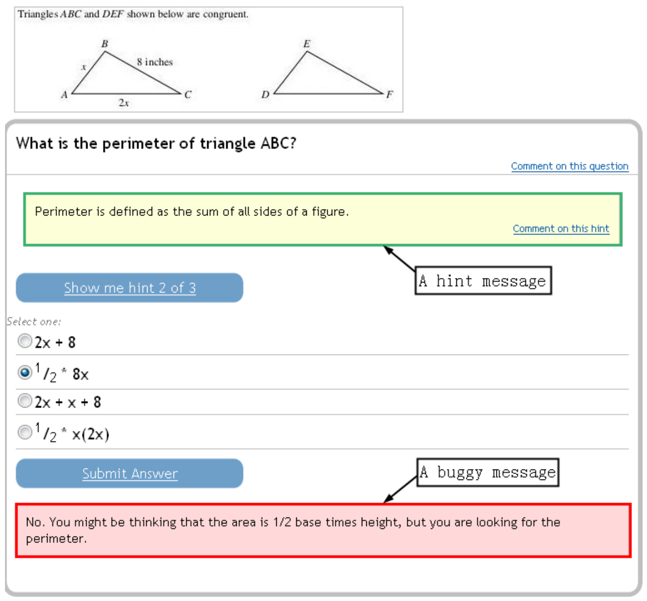
Heffernan, 2010) by using a logistic regression to predict student performance. In these models, however, the amount of assistance a student requires to answer a question correctly is not utilized. Only the student's first attempt is taken into account, and if a hint is requested, the question is marked wrong. But what would the effect on predictive accuracy be if the number of hints and attempts requested was factored into the model? Presumably, students that require more hints or attempts have lower knowledge (even though there are rare instances where this generalization does not hold; see (Shih, Koedinger, & Scheines, 2008). We use the term "assistance" to refer to the two quantities: the number of hints and the number of attempts required by a student to answer a question. For those not familiar with ITS, most will not let a student progress to the next question until they have answered the current question correctly; thus, all students eventually get each question right. Our intuition is that low knowledge students are perhaps more likely to require additional hints and attempts.

Feng and Heffernan (Feng & Heffernan, 2010) showed that they could use this sort of information to better predict a state test score (i.e. the Massachusetts Comprehensive Assessment Systems math test). However, they did not give a model that would function "online" as students are working; they limited themselves to only predicting state test scores. Arroyo, Cooper, etc. (Arroyo, Cooper, Burleson, & Woolf, 2010) showed how to use this information to predict learning gains. Their work suggests that using hints and attempts to model student behavior online could be effective. Furthermore, in our previous work (Wang, Heffernan, & Beck, 2010) we found even more evidence that the number of hints and attempts contain more predictive power than binary performance, and have the potential to enhance current student modeling techniques.

In this paper, we continue to explore the possibility of utilizing assistance information in tutoring systems to better model student behavior and better predict student performance.

The Tutoring System and Dataset

The data used in the analysis presented here came from the ASSISTments system, a freely available web-based tutoring system for 4th through 10th grade mathematics. The system gives tutorial assistance if a student makes a wrong attempt or asks for help. Fig. 3.2.1. shows an example of a hint, which is one type of assistance. A second type of assistance is presented if they click on (or type in) an incorrect answer, at which point the student is given feedback that they answered incorrectly (sometimes, but by no means always, students will get a context-sensitive message we call a “buggy message”).


Triangles ABC and DEF shown below are congruent.

What is the perimeter of triangle ABC?

Perimeter is defined as the sum of all sides of a figure.

Select one:

- $2x + 8$
- [✓ $^1/_2 * 8x$]
- $2x + x + 8$
- $^1/_2 * x(2x)$

No. You might be thinking that the area is 1/2 base times height, but you are looking for the perimeter.

Fig. 3.2.1. Assistance in ASSISTment

We used data from four Mastery Learning classes conducted in 2009. Mastery Learning is a strategy that requires students to continually work on a problem set until they have achieved a criterion (typically three consecutive correct answers). Questions in each problem set are generated randomly from several templates and there is no problem-selection algorithm used to choose the next question. We assume the difficulty of each question is dependent on its template (even though in theory, some randomly generated numbers might be easier than others). Two hundred 12-14 year old 8th grade students participated in these classes and generated 17,776 problem logs from 93 problem sets. Each problem set was generated from an average of 2.2 templates. The correctness of each answer was logged, as well as the number of hints required and the number of attempts made to answer each question. We only used data from a problem set for a given student if they had reached the mastery criterion. This data was collected in a suburban middle school in central Massachusetts. Students worked on these problems in a special "math lab" period, which was held in addition to their normal math class.

Specific Tested Models

We chose two popular yet very different models: KT and PFA for comparison when exploring the probability of adding assistance information into currently successful student models. We then developed an Assistance Model to infer the probability of a correct response to a given question based on previous assistance information.

KT

In our experiment, we used the Bayes Net Toolbox for Matlab developed by Murphy (Murphy, 2001) to implement Knowledge Tracing, and the Expectation Maximization (EM) algorithm to fit the model to the dataset. The EM algorithm finds a set of parameters that maximize the likelihood of the data by iteratively running an expectation step to calculate expected likelihood given student performance data and a maximization step to compute the parameters that maximize that expected likelihood. There have been reported issues of local maxima when using the EM algorithm. Pardos and Heffernan (Pardos & Heffernan, Navigating the parameter space of Bayesian Knowledge Tracing models: Visualization of the convergence of the Expectation Maximization algorithm, 2010) concluded, based on a simulation study, that with the initial parameters of this algorithm in a reasonable range (the sum of initial guess and slip value is smaller than 0.5), the algorithm will always converge to a point near the true parameter value. In our experiments, we choose initial parameters for each skill as follows: *initial knowledge* = 0.5, *learning* = 0.1, *guess* = 0.1, *slip* = 0.1.

PFA

Another model we used in our experiments is Performance Factor Analysis. PFA is a more recently proposed student modeling approach which has been shown to be superior to Knowledge Tracing in several papers (Pavlik, Cen, & Koedinger, 2009) (Gong, Beck, & Heffernan, 2010).

The model uses a logistic regression with student performance as the dependent variable, question identities as factors and the same skill practicing matrix used in Learning Factor Analysis (Cen, Koedinger, & Junker, 2006) as independent variables. The skill practicing matrix contains the number of prior successes and the number of prior failures for each skill at each point (question) in the problem set. After training, the model gains a parameter for each question representing its difficulty and two parameters for each skill representing the impact of previous successes and previous failures on this question.

Due to the fact that questions are randomly generated as a student progresses through a problem set, we used question template identity as factors rather than question identity in the logistic regression model. Moreover, we did not bound the PFA parameters to prevent negative learning rate.

Assistance Model

Our previous work on using assistance information (Wang, Heffernan, & Beck, 2010) was based on the intuition that the more assistance a student requires in answering a question, the lower the probability that student possesses the knowledge. In this study we develop a purely data driven model which makes no assumptions about how assistance information reflects student knowledge. The motivation behind building this model is to see directly from the data what the connection is between requiring assistance with a question and the probability of getting the next question right. To do so, we build a parameter table in which row indices represent the number of attempts a student required in the previous question and column indices represent the number of hints the student asked. Each cell contains the probability that the student will answer the current question correctly. For this value, we simply use the percentage of students who answered the current question correct when the previous question satisfies the row and column.

Statistically, using percent correct as a representation of the probability of correctness requires a large amount of data. In order to ensure each cell in the parameter table contains a sufficient number of data points while still preserving the granularity needed for distinguishing different assistance requirements, we separated the possible number of attempts into 3 interpretable bins:

• One attempt: the student only tried once to get the correct answer.

• Small amount of attempts: the student tried a reasonably small number (set to 2~5 in our experiments) of times.

• Large amount of attempts: the student tried 6 or more times to get the correct answer. It is likely that the student had difficulties in solving the problem, or was gaming the system.

Most responses that fall into the third bin come from fill in questions due to the fact that there are only a small amount of choices in multi-choice questions.

We also separated the possible number of hints into 4 different bins. To normalize the difference in the number of hints contained in each question, we used the percentage of total hints as a measurement rather than the raw number of hints.

• No hint: the student didn't ask for any hint;

• Small amount of hints: the hint percentage the student requested is in the range of (0, 50%];

• Large amount of hints: the hint percentage the student requested is in the range of [50%~100%);

• To the bottom hint: the student asked for all of the hints.

Table 3.2.1 shows an example of parameter table we computed from the training data. We can observe that in general, the more attempts or hints a student requires, the lower the probability that the student can answer the next question correctly. This confirms our intuition about assistance information: students requiring more assistance to solve a problem probably have less corresponding knowledge. Another interesting observation that can be made is that the lowest probability in the table occurs when students required all of the hints and then attempted only once to get the previous question right. This indicates that an alternating behavior of asking for a hint and then attempting to answer could be a more effective pattern of learning.

Table 3.2.1. Parameter table in AM

	attempt=1	***1<attempt<6***	***attempt>=6***
hint_percent=0	0.7376	0.7169	0.6328
0<hint_percent<=.5	0.6454	0.6926	0.6528
.5<hint_percent<1	0.6269	0.6058	0.5409
hint_percent=1	0.3929	0.4835	0.4382

The Assistance Model has only 12 parameters that inform the relationship between different assistance patterns and the probability of a correct response to the next question. It does not take into account any skill, student or question information and does not model student learning since it only looks into one previous question to make a prediction. The model utilizes assistance information exclusively. The computational cost of the model is extremely low, which makes it a good complement to other models that do not take into account assistance information.

Model Combination

In explaining student behavior and predicting student performance, Knowledge Tracing, Performance Factor Analysis and the Assistance Model will give very different results due to the different pieces of information they use and the different assumptions they make to build the models. By combining the Assistance Model with Knowledge Tracing and Performance Factor Analysis models, we add assistance information into these algorithms, which we believe will give us better prediction results than these models alone. In this section, we explored two different methods of combining models.

Averaging

Pardos & Heffernan (Pardos & Heffernan, 2011) demonstrated that a simple averaging technique can lead to higher prediction accuracy than either of the two methods that they were comparing by themselves. Similarly, we decided to average the Assistance Model and Knowledge Tracing/Performance Factor Analysis models together. Presumably, if a group of models have high accuracies and uncorrelated errors, we can get lower error by averaging them.

Regression

Using averaging to combine the predictions of different models makes the assumption that the different models' predictions should have the same weight, which may not necessarily be the case. To address this problem in our experiments, we also constructed a linear regression model with student performance as the dependent variable and prediction results from the Assistance Model and other models as independent variables, in order to find the best weights for the models we intend to combine. If one of the models is more useful than the other, this regression will allow us to learn which model should be weighted more heavily in making a prediction.

Results

In order to evaluate the Assistance Model and test the model combination methods, we ran experiments on the dataset with random 80% of students' data in each skill used as training data and random 20 % unseen students' data as test data.

Specific tested model results

To evaluate how well each of the specific tested models fit the data, we used three metrics to examine the predictive performance on the unseen test set: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and AUC (Area Under ROC Curve). Lower values for MAE and RMSE indicate better model fit while higher values for AUC reflect a better fit. The number of parameters in each model is also reported for comparison purposes.

Table 3.2.2. Accuracy results of three specific tested models.

	MAE	***RMSE***	***AUC***	***# of***
Baseline	0.4231	0.4600	0.5000	1
AM	0.3894	0.4449	0.6116	12
PFA	0.3797	0.4435	0.7004	393
KT	0.3535	0.4254	0.7329	372

Table 3.2.2 shows the results of the comparison for the three metrics. We used a method that always predicts the mean value as the baseline. In all of the three evaluation metrics, the performance of KT is better than the performance of PFA, which is better than AM. The fact that AM has the lowest predicting accuracy is reasonable considering the small number of total parameters in the model.

In the Mastery Learning dataset, each problem set contains questions of a single skill, and no multi-skill questions are considered; thus, the number of parameters in PFA = 2*# of problem sets + # of question templates = 2*93+207, while the number of parameters in KT = 4*# of problem sets = 4*93.

We determine whether the difference between two models is statistically significant by computing each evaluation metric's value for each student to account for the non-independence of their actions, then comparing each pair of specific tested models using a two tailed paired t-test. The results are shown in Table 3.2.3. We compute the *p* value of all three metrics between pairs such as (AM, PFA) to see if the models are reliably different from each other. The bold values in Table 3.2.3 show the statistically significant differences between corresponding pairs and metrics. As shown in the table, the differences in RMSE and AUC between AM and PFA, and the difference in AUC between KT and PFA are not significant. Other than those, most of the metric values of the different models in Table 3.2.2 are significantly different from each other.

Table 3.2.3. Reliability of difference between two specific tested models.

	MAE	***RMSE***	***AUC***
(AM, Baseline)	**0.0000**	**0.0000**	**0.0209**
(KT, AM)	**0.0000**	**0.0000**	0.4903
(PFA, AM)	**0.0365**	0.5717	0.2049
(KT, PFA)	**0.0000**	**0.0000**	0.3618

Our result showing PFA having a lower accuracy than KT is inconsistent with some other works comparing PFA with KT. The difference may have been caused by a certain property of our experimental dataset. Because PFA uses 1 parameter per question and two parameters per skill while KT uses 4 parameters per skill, it is possible that KT works better in a dataset where skill differences are greater than question variety. Mastery Learning data contains data from different skills, and questions of a particular skill are of similar difficulty levels in order to serve the purpose of random selection.

The accuracy of AM is lower than both that of PFA and KT, yet significantly higher than the baseline. This indicates that although applying AM alone may lead to worse prediction due to a lack of complexity in the model, it still contains valuable information that has the potential to help improve other models' results.

Combined model results

In order to answer the question of whether or not adding the assistance information into an existing model could lead to a more accurate student performance prediction, we still use MAE, RMSE and AUC as evaluation metrics and continue to report the number of parameters in the final model as measurement of model complexity.

The result is shown in Table 3.2.4. For comparison, we also computed the accuracy of combining PFA and KT.

Table 3.2.4. Accuracy results of four combined models.

	MAE	*RMSE*	*AUC*	*# of params*
AVG(AM,PFA)	0.3845	0.4304	0.7191	405
LR(AM,PFA)	0.3732	0.4303	0.718	407
AVG(AM,KT)	0.3714	0.4261	0.7358	384
LR(AM,KT)	0.3515	0.4216	0.7369	386
AVG(PFA,KT)	0.3666	0.4246	0.7381	765
LR(PFA,KT)	0.3532	0.4236	0.7396	767

In Table 3.2.4, AVG represents the averaging combining method and LR represents the linear regression combining method. The number of parameters used in averaging combining method is equal to the sum of the parameter numbers of each individual method, while the linear regression combining method requires two additional parameters to indicate the impact of each model with respect to the final prediction.

In the linear regression combining method, the regression model is trained on a training dataset. The resulting formula for combining PFA and AM on our dataset is:

-0.3052+0.8241*AM_prediction+0.6003*PFA_predicion;

The formula for combing KT and AM is:

-0.1026+0.2078*AM_prediction+0.9373*KT_prediction;

The formula for combing KT and PFA is:

-0.0600+0.1689*PFA_prediction+0.9145*KT_prediction.

The parameters indicate that in the LR(AM, PFA) model, AM is the main influencer of the final prediction, while in the LR(AM, KT) and LR(PFA, KT) model, KT is the main influencer.

Comparing Table 3.2.4 and Table 3.2.3, we find that LR(AM, PFA) is better than PFA and LR(AM, KT) and LR(PFA, KT) are better than KT in all metrics. The averaging combining method, on the other hand, does not demonstrate such clear trends of improvement.

We also did reliability analysis by computing metric values for each student to account for the non-independence of actions within each student's dataset, and then compared each pair of models using a two tailed paired t-test. The *p* values are reported in Table 3.2.5, in which bold values indicate the differences are statistically significant.

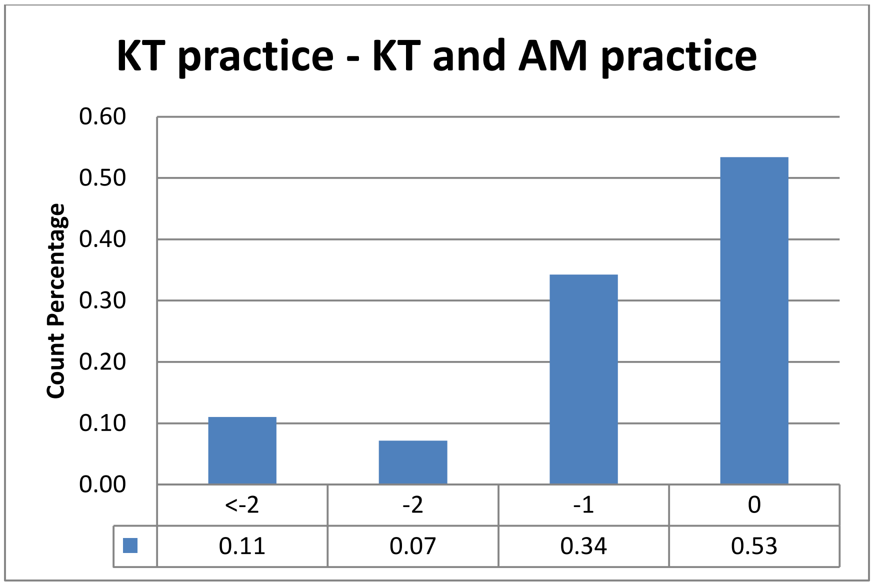
From Table 3.2.5 we can see that using the linear regression method to combine the AM model with PFA or KT will reliably increase the accuracy of PFA or KT respectively in all metrics. The fact that the accuracy of using linear regression to combine PFA and KT is not reliably different with KT alone could be caused by the low coefficient PFA has in the regression formula.

Table 3.2.5. Reliability of difference between combined models and specific tested models.

	MAE	***RMSE***	***AUC***
AVG(AM, PFA), PFA	**0.0365**	**0.0000**	0.8788
LR(AM, PFA), PFA	**0.0406**	**0.0000**	**0.0145**
AVG(AM, KT), KT	**0.0000**	**0.0420**	0.8398
LR(AM, KT), KT	**0.0009**	**0.0013**	**0.0095**
AVG(PFA, KT), KT	**0.0000**	**0.0042**	**0.0251**
LR(PFA, KT), KT	0.5536	0.4456	0.7725

We also want to know when combine KT and AM, whether or not we could save a student's practice time by detecting he/she master a skill sooner. Unfortunately the answer is no. We looked at when combine KT and AM, the difference in the number of questions students need to do in order to master a skill compare to the original KT model. We used the probability of answering a question correct > 0.9 as mastery threshold and computed the number of questions students need to do for AM and KT combination and KT model alone respectively for the test dataset. The real world mastery threshold in our dataset is three correct answers in a row. In this analysis, we discarded all the data points when students reached real world mastery before KT and AM combination or KT model consider them mastery. For the remaining 839 data points, the result is shown in Fig. 3.2.2. x-axis represents how many more questions KT model would require students do than KT and AM combined model, i.e. a "-1" shows KT model would require 1 less question than KT and AM combined model. y-axis shows the percentage of the count of such instances.

As we can see from Fig. 3.2.2., in 53% of the time, students need the same number of questions to be considered mastery, and in 47% of the time, when combine with AM model, more practice is needed for students to be considered mastery.



	<-2	-2	-1	0
■	0.11	0.07	0.34	0.53

Fig. 3.2.2. Difference of the number of questions students need to do

Discussion and Future Work

The model we proposed in this paper is a simple and fast method of utilizing assistance information. Experiments show this model alone doesn't provide better performance prediction than other more complicated models; however, combining this model with other models will reliably improve the predictive accuracy of that model.

This work is the beginning of an attempting to utilize assistance information in intelligent tutoring systems in order to better predict student performance. There are several questions that we are interested in exploring.

One question is how to improve the Assistance Model by adding more parameters. Currently we use only 12 parameters for all of the data, which assumes that all of the skills and all of the students share the same parameters given a certain assistance pattern. Skill identity parameters and student individualizing parameters can be easily added into the Assistance Model by computing parameter tables for each skill or each student separately. The resulting model will contain much more information than the Assistance Model currently has, thus theoretically resulting in better predictive accuracy. Although this modification will significantly increase the number of parameters in the model, the cost will remain low due to the fact that the Assistance Model is a table querying method, which requires no complex computation. The Assistance Model could also be extended to take into account the learning curve by building a parameter table using assistance information from a sequence of problems rather than only the previous problem. All of these modifications will face the problem that the data points for computing each table's parameters may become sparse and may not contain enough information. Thus analysis of parameter table reliability is required.

Another question worth exploring is the development of a more interpretable combining method of different models. We would like to know if there are rules which can guide us to choose one model over another given different circumstances.

Contribution

For many years, most assessment work inside Intelligent Tutoring Systems has looked only at student first response and ignored the amount of assistance a student needed to eventually get a problem correct. While we could have lived to have figure out an elegant way to invariable this information into a Bayesian network, in this work, we took a much simpler approach, and simply predicted student correctness on questions by "tabulating" up the number of times students got the next question correct, broken out by the number of hints and attempts the student had to make. This method ignores everything the student did other than the assistance they get on the previous question. Therefore, it makes senses that when coupled (i.e, ensemble) with Knowledge Tracing or Performance Factor Analysis, it does better than either model alone. We encourage the field to look at other ways of modeling the multiple different sources of information. There clearly is a lot of information in the number of hints and attempts.

The method can be easily applied to any current student models. All they have to do is tabulate the percentage change a student get a problem correct, broken out by the number of hints and attempts used on the previous problem.

3.3 Assistance Model Extension: The Sequence of Action Model

Intelligent Tutoring Systems (ITS) have been proven to be efficient in providing student assistance and assessing their performance when they do their homework. Many research projects have been done to analyze how students' knowledge grows and to predict their performance from within intelligent tutoring system. Most of them focus on using correctness of the previous question or the number of hints and attempts students need to predict their future performance, but ignore the sequence of hints and attempts. In this research work, we build a Sequence of Actions (SOA) model taking advantage of the sequence of hints and attempts a student needed for the previous question to predict students' performance. A two-step modeling methodology is put forward in the work, which is a combination of Tabling method and the Logistic Regression. We compared SOA with Knowledge Tracing (KT) and Assistance Model (AM) and combinations of SOA/AM and KT. The experimental results showed that the Sequence of Action model has reliably better predictive accuracy than KT and AM and its performance of prediction is improved after combining with KT.

<u>*This chapter has been published at the following venue:*</u>

Zhu, L., Wang, Y., & Heffernan, N. T. (2014). The Sequence of Action Model: Leveraging the Sequence of Attempts and Hints. *BKT20y Workshop of Educational Data Mining*. (Zhu, Wang, & Heffernan, 2014)

Introduction

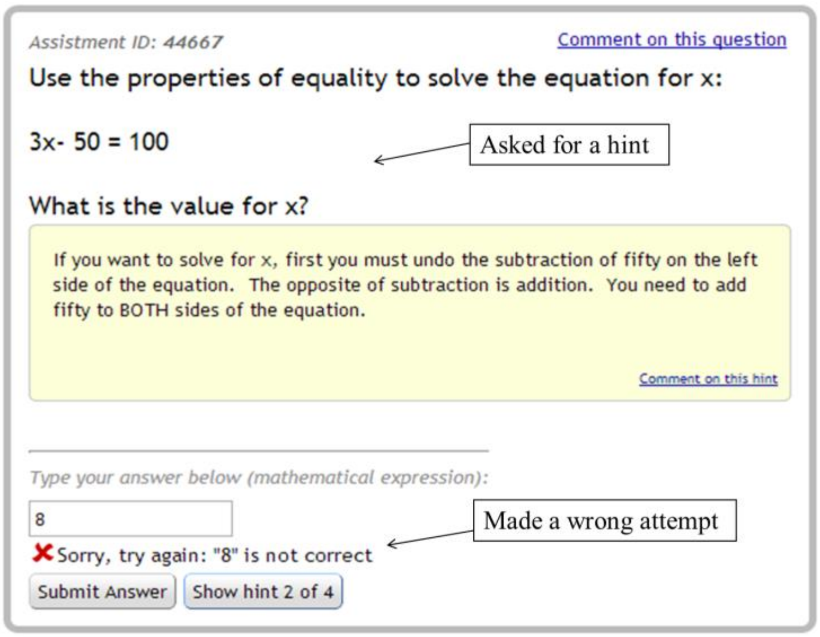
Not too much attention is paid to the interaction data generated when students interacts with computer tutors. Shih et al (Shih, Koedinger, & Scheines, 2010) utilized Hidden Markov Model clustering to discover different strategies students used while working on an ITS and predicted learning outcomes based on these strategies. Their work was based on data set consists of a series of transactions and each transaction is a <Student, Step, Action, Duration> tuple. This model takes into account both students' action, attempt or help request, and action duration. The experimental result of their Stepwise-HMM-Cluster model shows that persistent attempts lead to better performance than hint-scaffolding strategy. Some papers have shown the value of using the raw number of attempts and hints. The National Educational Technology Plan cited Feng, Heffernan and Koedinger's work (Feng, Heffernan, & Koedinger, 2006) and the User Modeling community gave it an award for best paper for showing that the raw number of hints and attempts is informative in predicting state test scores. Wang and Heffernan (Wang & Heffernan, 2011) built an Assistance Model (AM) and generated a performance table based on students' behavior of doing the previous question. Hawkins et. al. (Hawkins W. , Heffernan, Wang, & Baker, 2013) extended AM by looking at students' behavior for the two previous questions.

These educational data mining models that utilize the number of assistance students request and the number of attempts they make to predict students' performance have ignored the sequencing of students' interaction with ITS. Consider a thought experiment. Suppose you know that Bob Smith asked for one of the three hints and makes one wrong answer before eventually getting the question correct. What if someone told you that Bob first made an attempt then had to ask for a hint compared to he first requested a hint and then made a wrong attempt. Would this information (whether he started with an attempt or a hint) add value to your ability to predict whether Bob will get the next question correct? We suspected that a student who first makes an attempt tends to learn by himself and has higher probability to master the knowledge and answer the next same question correct.

In this paper, we present the SOA model and compare it to the KT model and the Assistance model, as well as the combined models to see if knowing sequence of action information could improve upon a standard Knowledge Tracing model, or even upon knowing number of hints and number of attempts.

The Tutoring System and Dataset

The data we were using comes from the ASSISTments platform, which is an online tutoring system for K12 students that gives immediate feedback to teachers, students and parents. The ASSISTments gives tutorial assistance if a student makes a wrong attempt or asks for help. Fig. 3.3.1 shows an example of a hint, which is one type of assistance. Other types of assistance include scaffolding questions and context-sensitive feedback messages, known as "buggy messages".


Assistment ID: 44667

Comment on this question

Use the properties of equality to solve the equation for x:

3x- 50 = 100

What is the value for x?

If you want to solve for x, first you must undo the subtraction of fifty on the left side of the equation. The opposite of subtraction is addition. You need to add fifty to BOTH sides of the equation.

Comment on this hint

Type your answer below (mathematical expression):

8

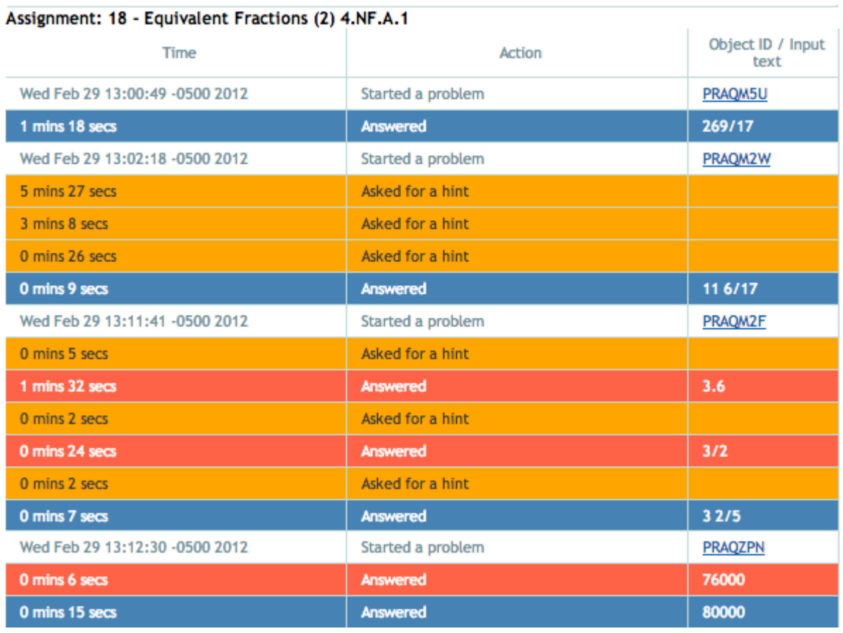
Sorry, try again: "8" is not correct

Fig. 3.3.1. Assistance in ASSISTments. Which is first: asking for a hint or make an attempt?

Fig. 3.3.1 shows a student who asked for a hint (shown in yellow and also indicated by the button says "Show hint 2 of 4"), but it also shows that the student typed in eight and got feedback that that was wrong. Though Fig. 3.3.1 shows the number of hints and attempts, but interestingly, you cannot tell whether the student asked a hint first or made an attempt first. This papers argument is that this information is very important.

ASSISTments records all the details about how a student does his/her homework and tests, from which scientists can get valuable material to investigate students' behavior and their learning process. These records include the start time and end time of a student does a problem, the time interval between a student makes an attempt and he/she asks for a hint, the number of attempts a student makes and the number of hints a student asks, as well as the answer and result for each attempt a student makes.

Fig. 3.3.2 shows an example of a detailed sequence of action recorded by the system. A row in blue means that the answer is correct; a row in red means that the answer is wrong; and a row in orange means that the student asks for a hint. We can see that this student answered the first question PRAQM5U correctly on his first attempt. The sequence of action is 'a' ('a' represents an attempt). For the second problem PRAQM2W, he/she asked 3 hints before making the correct answer. The sequence of action is 'hhha' ('h' represents a hint). For the third question PRAQM2F, he/she alternatively asked for hints and made attempts. The sequence of action is 'hahaha'. For the last question PRAQZPN, he/she made 1 wrong attempt before making the correct answer. The action sequencing is 'aa'.


Assignment: 18 - Equivalent Fractions (2) 4.NF.A.1

Time	Action	Object ID / Input text
Wed Feb 29 13:00:49 -0500 2012	Started a problem	PRAQM5U
1 mins 18 secs	Answered	269/17
Wed Feb 29 13:02:18 -0500 2012	Started a problem	PRAQM2W
5 mins 27 secs	Asked for a hint	
3 mins 8 secs	Asked for a hint	
0 mins 26 secs	Asked for a hint	
0 mins 9 secs	Answered	11 6/17
Wed Feb 29 13:11:41 -0500 2012	Started a problem	PRAQM2F
0 mins 5 secs	Asked for a hint	
1 mins 32 secs	Answered	3.6
0 mins 2 secs	Asked for a hint	
0 mins 24 secs	Answered	3/2
0 mins 2 secs	Asked for a hint	
0 mins 7 secs	Answered	3 2/5
Wed Feb 29 13:12:30 -0500 2012	Started a problem	PRAQZPN
0 mins 6 secs	Answered	76000
0 mins 15 secs	Answered	80000

Figure 2. Students' action records in ASSISTments

We used data from one Mastery Learning classes. Mastery Learning is a strategy that requires students to continually work on a problem set until they have achieved a preset criterion (typically three consecutive correct answers). Questions in each problem set are generated randomly from several templates and there is no problem-selection algorithm used to choose the next question.

Sixty-six 12-14 year-old, 8th grade students participated in these classes and generated 34,973 problem logs. We only used data from a problem set for a given student if they had reached the mastery criterion. This data was collected in a suburban middle school in central Massachusetts. Students worked on these problems in a special “math lab” period, which was held in addition to their normal math class.

If a problem only has one hint, the hint is the answer of the problem and is called the bottom hint. After a student asks for a bottom hint, any other attempt is meaningless because he or she already knows the answer. In the experiment, we only consider the problem logs that have at least two hints. And the answer will be marked as incorrect if students ask for a hint or the first attempt is incorrect. Moreover, we excluded these problem logs: 1) students quit the system immediately after they saw the question so that the action logs were blank or 2) after students requested hints, no attempt was made.

We equally split 66 students into six groups, 11 students in each, to run 6-fold cross validation. We trained the SOA model and the KT model on the data from five of the groups and then computed the prediction accuracy on the sixth group. We did this for all six groups.

Specific Tested Models

KT

Knowledge Tracing (KT) (Corbett & Anderson, 1995) is one of the most common methods that are used to model the process of student's knowledge gaining and to predict students' performance. KT is a Hidden Markov Model (HMM) with a hidden node (student knowledge node) and an observed node (student performance node). It assumes that a skill has 4 parameters; two knowledge parameters: prior and learn, and two performance parameters: guess and slip. The goal of KT is to estimate the student knowledge from his/her observed actions.

Assistance Model

Motivated by the intuition that students who need more assistance have lower probability possessing the knowledge, Wang and Heffernan (Wang & Heffernan, 2011) built a pure data driven "Assistance" model to disclose the relationship between assistance information and students' knowledge.

As described in section 3.2, the Assistance Model builds a parameter table, in which row indices represent the number of attempts a student required, column indices represent the number of hints the student asked, and each cell contains the probability that the student will answer the next question correctly. Table 3.3.1 is the parameter table gained from our dataset. Similar with Wang and Heffernan's experimental results, this parameter table confirms that students requiring more assistance to solve a problem probably have less corresponding knowledge.

Table 3.3.1. Assistance Model parameter table, average across six folds

	attempt= 1	***1<attempt<6***	***attempt>=6***
hint_percent = 0	0.8410	0.7963	0.7808
0<hint_percent<=.5	0.6286	0.6933	0.6741
.5<hint_percent<1	0.4494	0.6290	0.6522
hint_percent = 1	0.4293	0.6147	0.6218

The Sequence of Action Model

The Sequence of Action (SOA) model we present takes advantage of the order information about how students make attempts and ask for hints. Different students have different sequence of actions. Some students answered correctly only after one attempt and some students kept trying many times. Some students asked for hints and made attempts alternatively, which could indicate that they were learning by themselves. In this data, there are 217 different sequences of actions. Intuitively, students' actions reflect their study attitude, which determines their performance. Based on the assumption that students who make more attempts are tend to master knowledge better than students who ask for more hints, we divided them into five categories or bins: (1) One Attempt: the student correctly answered the question after one attempt; (2) All Attempts: the student made many attempts before finally get the question correct; (3) All Hints: the student only asked for hints without any attempts at all; (4) Alternative, Attempt First: the students asked for hints and made attempts alternatively and made an attempt at first; (5) Alternative, Hint First: the students asked for hint and made attempts alternatively and asked for a hint first. Table 3.3.2 shows some examples of the action sequences in each category.

Table 3.3.2. Sequence of Action Category and Examples

Sequence of Action Category/ Bin Name	***Examples***
One Attempt/Bin 'a'	a
All Attempts/Bin 'a+'	aa, aaa,..., aaaaaaaaaaaa
All Hints/Bin 'h+'	ha, hha,..., hhhhhhha
Alternative, Attempt First/Bin 'a-mix'	aha, aahaaha,..., aahhhaaa
Alternative, Hint First/Bin 'h-mix'	haa, haha,..., hhhhaha

All sequences end with an attempt, because in ASSISTments a student cannot continue to next question unless he/she fills in the right answer of the current problem. In Table 3.3.2, 'a' stands for answer and ‘h’ stands for hint. An action sequence “ahha” means that a student makes an attempt and then asks for two hints before he/she types the correct answer and move on to the next question.

Sequence of Action Tabling

After divide all of sequence of actions into five categories, we use a tabling method, which computes the correctness percentage of the next question from the training data. For each fold, a separate table is generated by counting the total number and the correct number of the next question of each bin. After counting, a next correct percent is calculated by dividing *Next Correct Count* by *Total Count*.

Table 3.3.3. Next correct percent table of training group of fold 1

Bin	***Total Count***	***Next Correct Count***	***Next Correct Percent***
‘a’	22964	19157	0.834
‘a+’	3538	2690	0.760
‘h+’	335	172	0.513
‘a-mix’	2030	1318	0.649
‘h-mix’	72	37	0.513

Table 3.3.3 shows the table computed for fold 1. Tables for other folds are similar. From Table 3.3.3, we can see that the percent of next-question-correct is highest among students only using one attempt since they master the skill the best. They can correctly answer the next question with the same skill. For students in ‘a+’ bin, they are more self-learning oriented, they try to learn the skill by making attempts over and over again. So they get the second highest next-question-correct percent. But for students in the ‘h+’ category, they do the homework only relying on the hints. It is reasonable that they don’t master the skill well or they don’t even want to learn, so their next-question-correct percent is very low.

The alternative sequence of action reflects students' learning process. Intuitively, these students have positive attitude for study. They want to get some information from the hint, based on which they try to solve the problem. But the results for the two alternative categories are very interesting. Though students in these two categories alternatively ask for hints and make attempts, the first action can somewhat decide their learning altitude and final results. For students who make an attempt first, if they get the question wrong, they try to learn it by asking for hints. But for students who ask for a hint first, they seem to have less confidence in their knowledge. Although they also make some attempts, from the statistics of action sequence, they tend to ask for more hints than making attempts. The shortage of knowledge or the negative study attitude makes their performance as bad as the students asking exclusively for hints first.

Logistic Regression

The second part of the SOA model makes use of a logistic regression model and information we get from the first part of SOA, i.e. tabling method.

Although the next correct percentage we get from the tabling method can reflect the trend of next correct percentage, the table is very rough. So we use it as a feature in our logistic regression prediction model.

The dependent variable ***Next Correct*** of the logistic regression model has two states: correct and incorrect. The independent variables are ***Skill_ID*** and ***Credit*** (the next correct percentage generated by the tabling method). ***Skill_ID*** was treated as a categorical factor, while ***Credit*** was treated as a continuous factor. There are in total 51 skills. As mentioned before, there are six folds and each fold has its own next correct percentage table.

We used Binary Logistic Regression in SPSS to train the model. Logistic coefficients are fitted through Expectation Maximization of at most 20 steps. Some of the coefficients of the first fold are shown in Table 3.3.4.

Table 3.3.4. Coefficients of logistic regression model of fold 1

Parameters	*Value*
β_0(Intercept)	-1.679
$\beta_{1.0}$(skill_id 16)	0.322
$\beta_{1.1}$(skill_id 17)	-0.007
$\beta_{1.2}$(skill_id 24)	20.168
...	...
$\beta_{1.50}$(skill_id 371)	0.470
β_2(Credit)	3.286

Model Combination

Since the SOA model uses completely different information from KT, we expected a potential improvement from combing SOA results with the predictions from KT. We combined models using two different methods.

The first method was simply average the SOA and KT predictions. Presumably, if a group of models have high accuracies and uncorrelated errors, we can get lower error by averaging them. To compare with the combination of AM model and KT model, we also computed the average of these two models.

The second method was a linear regression model with student performance as the dependent variable. This method takes into account the fact that different models' predictions may have different weight in the final prediction. If one of the models is more useful than the other, this method will allow us to learn which model should be weighted more heavily. SPSS was used to train linear regression models. The function for KT and AM is:

-0.322+0.639*AM_prediction+0.769*KT_prediction;

The function for KT and SOA is:

-0. 004+0. 687*SOA_prediction+0. 321*KT_prediction;

We did not combine AM and SOA, because both of them use information about hints and attempts. From the functions, we can tell that SOA weights heavier than KT, which indicates that SOA is more useful than KT in making a prediction.

Experimental Results

Compare AM, SOA and KT

To evaluate how well each of the specific tested models (SOA, AM, KT) and the combined models fit the data, we used three metrics on unseen test fold: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Area Under the Curve (AUC). Lower values for MAE and RMSE and higher values for AUC indicate better model fit.

Table 3.3.5 shows values of the three metrics of the six-fold across validation, which are calculated by averaging corresponding numbers obtained from each validation.

Table 3.3.5. Prediction accuracy of KT, SOA, AM and Ensemble

	MAE	***RMSE***	***AUC***
AM	0.3007	0.3844	0.5795
SOA	0.2871	0.3767	0.6786
KT	0.2939	0.3790	0.6735
LR(AM, KT)	**0.2874**	0.3759	0.6824
LR(SOA, KT)	0.2878	0.3762	0.6813
AVG(SOA, KT)	0.2876	**0.3757**	**0.6836**

Similar with Wang and Heffernan’s results (Wang & Heffernan, 2011), the performance of linear regression combination of AM and KT, named as LR(AM, KT) is better than KT itself, which indicates information about the number of hints and attempts improves the prediction of KT model. Overall, the combinations of any two models have higher prediction accuracy, especially the average ensemble of SOA and KT, named as AVG(SOA, KT), which has better accuracy than other combinations. Also, the linear regression of AM and KT has better prediction accuracy than linear regression of SOA and KT. However, from the 2- tailed paired t-test results in Table 3.3.6, the difference between any two model combinations are not significant.

Table 3.3.6. Reliability when compare KT, SOA, AM, and Ensemble

	MAE	*RMSE*	*AUC*
AM vs SOA	**0.000**	**0.000**	**0.000**
AM vs KT	**0.000**	**0.000**	**0.000**
AM vs LG(AM, KT)	**0.000**	**0.000**	**0.000**
AM vs LR(SOA, KT)	**0.000**	**0.000**	**0.000**
AM vs AVG(SOA, KT)	**0.000**	**0.000**	**0.000**
SOA vs KT	**0.000**	**0.000**	**0.037**
SOA vs LG(AM, KT)	0.298	**0.030**	0.083
SOA vs LR(SOA, KT)	**0.000**	**0.001**	**0.006**
SOA vs AVG(SOA, KT)	**0.020**	**0.000**	**0.003**
KT vs LR(AM, KT)	**0.000**	**0.000**	**0.000**
KT vs LR(SOA, KT)	**0.000**	**0.000**	**0.000**
KT vs AVG(SOA, KT)	**0.000**	**0.000**	**0.000**
LR(AM, KT) vs LR(SOA, KT)	0.265	0.296	0.469
LR(AM, KT) vs AVG(SOA, KT)	0.271	0.138	0.079
LR(SOA, KT)vs AVG(SOA, KT)	0.258	**0.001**	**0.010**

To examine whether there is significant difference between these models, we did a 2-tailed paired t-test. The *p* values are shown in table 3.3.6. We observe that most of the differences between two models are reliable, except for the difference between AM and KT combination and SOA and KT combination. This could be caused by that both SOA and AM use the information about students' actions of hints and attempts.

Further Analysis for SOA and KT

From previous results, we observed that the best model is AVG(SOA,KT). In order to better investigate this combination, we ran student level and skill level analysis.

Table 3.3.7 and 3.3.8 are the student level results across 66 students to account for the non-independence of their actions. Take MAE as an example, for each student, a MAE is calculated based on all data available for that student. Then an average value for MAE is computed based on MAE of all students. Table 3.3.8 shows the t-test *p* value for each pair of these three models, where the remaining degrees of freedom on all of the tests is 65.

Table 3.3.7. Student Level accuracy of KT, SOA and Ensemble

	MAE	***RMSE***	***AUC***
KT	0.2939	0.3790	0.6738
SOA	0.2871	0.3767	0.6786
AVG(KT, SOA)	0.2905	0.3765	0.6811

Table 3.3.8. Student level reliability of difference of KT, SOA and Ensemble

	MAE	***RMSE***	***AUC***
KT vs SOA	**0.0000**	**0.0000**	0.0551
KT vs AVG	**0.0000**	**0.0000**	**0.0000**
SOA vs AVG	**0.0000**	0.0698	0.0698

Table 3.3.9 and 3.3.10 are the skill level results across all 51 skills. From table 3.3.9, we observe that the AUC value for all of the models are very low, which indicates these models do not make a good classification at the skill level. The t-test *p* value with remaining degrees of freedom 50 is shown in table 3.3.10.

Table 3.3.9. Skill level accuracy of KT, SOA and Ensemble

	MAE	*RMSE*	*AUC*
KT	0.3064	0.3762	0.4675
SOA	0.2942	0.3713	0.4769
AVG(KT, SOA)	0.3003	0.3710	0.492

Table 3.3.10. Skill Level reliability of difference of KT, SOA and Ensemble

	MAE	*RMSE*	*AUC*
KT vs SOA	**0.0000**	**0.0136**	0.3492
KT vs AVG	**0.0000**	**0.0002**	**0.0003**
SOA vs AVG	**0.0000**	0.3982	**0.0059**

The student and skill level analysis generate similar conclusions, that SOA and KT ensemble outperform KT in all of the three metrics. When compare the ensemble model with SOA alone, the result is not so clear.

Discussion and Future Work

In this paper, we put forward a Sequence of Action (SOA) model which makes use of the clicking sequence of students making attempts and asking for hints. We conducted six-fold cross validation experiments. The experimental result shows that SOA has reliably higher prediction accuracy than Knowledge Tracing and Assistance Model. The average combination of the SOA and KT has the highest prediction accuracy. In sum, the sequence of students' action provides important information in predicting students' performance.

This work is the beginning of utilizing the sequence of asking for hints and making attempts recorded by intelligent tutoring systems to better predict student performance. There are many open spaces for us to explore. For example, the experiment data we used comes from ASSISTments, does SOA model still make a big difference if use data from other intelligent tutor systems? How much can the performance of SOA model be improved after combined with other prediction models such as PFA (Pavlik, Cen, & Koedinger, 2009)? What is the SOA model's performance if we use a student action sequence of several previous question when train the model? How does SOA perform after individualization?

Contribution

Predicting student performance is an important part of the student modeling task in Intelligent Tutoring Systems. A large portion of papers at EDM have focused on this. Many models and techniques have been used to model and investigate students' performance. However, little attention has been paid to the temporally sequential actions of students when interacting with the tutoring systems. To our knowledge we are the first to use the temporal sequencing of hints and attempts. It turns out that by paying attention to this we can better predict student performance. In this paper, we introduce the Sequence of Action model which makes use of the click-stream data of the sequence of making attempts and asking for hints when students do their homework using an Intelligent Tutoring System. Students' actions can be very different from each other, but we found there are some useful patterns.

According to our six-fold cross validation experiments and paired two tailed t-test, both on student level and skill level, Sequence of Action model has reliably higher prediction accuracy than KT and AM, the later uses the number of hints students ask and the number of attempts students make. Furthermore, we combine SOA and KT using average and linear regression methods, and the ensemble model's performance is better than SOA, KT and the ensemble model of AM and KT. This indicates that the sequential information of student action does contain more information about students' learning than the count information of student action.

Chapter 4: Analysis about Knowledge Retention

4.1 Incorporating Factors Influencing Knowledge Retention

The goal of predicting student behavior on the immediate next action has been investigated by researchers for many years. However, a fair question is whether this research question is worth all of the attention it has received. This chapter investigates predicting student performance after a delay of 5 to 10 days, to determine whether, and when, the student will *retain* the material seen. Although this change in focus sounds minor, two aspects make it interesting. First, the factors influencing retention are different than those influencing short-term performance. Specifically, we found that the number of student correct and incorrect responses were not reliable predictors of long-term performance. This result is in contrast to most student-modeling efforts on predicting performance on the next response. Second, we argue that answering the question of whether a student will retain a skill is more useful for guiding decision making of intelligent tutoring systems (ITS) than predicting correctness of next response. We introduce an architecture that identifies two research topics that are meaningful for ITS decision making. Our experiments found one feature in particular that was relevant for student retention: the number of distinct days in which a student practiced a skill. This result provides additional evidence for the spaced practice effect, and suggests our models need to be aware of features known to impact retention.

This chapter has been published as a short paper at the following venue:

Wang, Y., Beck, J.E. (2012). Incorporating Factors Influencing Knowledge Retention into a Student Model. In *Proceedings of the 5th International Conference on Educational Data Mining*. pp. 201-203. (Wang & Beck, 2012)

Introduction

The field of the educational data mining (EDM) has been focusing on predict correctness of the next student response for many years, e.g. (Beck, 2003). Very little work has been done with respect to longer-term prediction. Two common approaches for student modeling are knowledge tracing (Corbett & Anderson, 1995) and performance factors analysis (Pavlik, Cen, & Koedinger, 2009). Both of these approaches focus on examining past student performances, and predicting whether the student's next response will be correct or incorrect. The source of power for both of these techniques is the student's pattern of correct and incorrect responses. In fact, that input is the only piece of information knowledge tracing (KT) uses (beyond which skill the problem is associated with). KT observes whether the student responds correctly or not, and uses its performance parameters, guess and slip, to update its estimate of the student's knowledge. KT takes the form of a dynamic Bayesian network, where each time slice represents an item the student is working on.

Performance factors analysis (PFA) works similarly, and keeps track of the number of correct and incorrect responses the student has made of in this skill. In addition, some versions of PFA also make use of an item difficulty parameter to account for item complexity. PFA takes the form of a logistic regression model, predicts whether the student will respond to an item correctly, and estimates coefficients for the number of correct and incorrect responses that maximize predictive accuracy.

A connection with student modeling is mastery learning. In a mastery learning framework, a student continues to practice a skill until it is "mastered." The exact definition of mastery varies, but typically involves recent student performance. For example, the KT framework suggests that the probability a student knows a skill exceeds 0.95, then the student has mastered the skill. The

ASSISTments project (www.assistments.org) uses a simpler heuristic of three consecutive correct responses to indicate mastery.

However, there is evidence that strictly local measures of student correctness are not sufficient. Specifically, students do not always retain what they have learned. Aside from the psychology literature, e.g. (Anderson, 1993) (Ebbinghaus, 1885) (George & John, 1994), there has been work within student modeling that demonstrated students were likely to forget some material after a delay. Qiu et al. (Qiu, Qi, Lu, Pardos, & Heffernan, 2011) extended the Knowledge Tracing model, to take into account that students exhibit forgetting when a day elapses between problems in the tutor.

Researchers in the ITS field are currently using short-term retention as an indicator for mastery learning. However, for a cumulative subject like mathematics, we are more concerned with the ability of the students to remember the knowledge they learned for a long period of time. Pavlik and Anderson (Pavlik & Anderson, 2005) studied alternative models of practice and forgetting, and confirmed the standard spacing effect in various conditions and showed that wide spacing of practice provides increasing benefit as practice accumulates, and less forgetting afterwards as well, which is consistent with classic cognitive science results (Cain & Willey, 1939).

Problem and Approach

Although the fields of student modeling and EDM have focused on short-term student performance, there is nothing inherent in student modeling or in EDM that requires such a focus. Conceptually, it is possible to construct models that predict other constructs of interest, such as whether the student will remember a skill after a period of time. Why would we want to construct such a model? We argue that whether a student will not only respond correctly on an item right away, the mastery approach used by KT, but whether the student will remember enough to respond correctly, after taking a break from working with the tutor, is a better definition of mastery. At best, it is unclear how to apply a short-term model such as KT or PFA for such a decision-making task. However, if we could build such a detector, we could deploy it in a computer tutor and use it to decide when to stop presenting items to students. Perhaps a student who gets several items correct in a row, and masters the skill by traditional definition, will be predicted to not retain the skill and should receive additional practice.

The approach we use is straightforward: rather than attempting to predict every next student performance, instead we focus on student performances that occur after a long delay. In this way, even though we are not explicitly modeling the forgetting process, our student modeling approach captures aspects of performance that relate to student long-term retention of the material. It is reasonable that the field of student modeling did not start with long-term retention, as only a small minority of student practice opportunities takes place after a long delay. Therefore, such restrictions would result in a too-small data set to train the student model parameters. However, with the advent of large educational sets, such restrictions become less relevant.

The data used in this analysis came from the ASSISTments system, a freely available web-based tutoring system for 4th through 10th grade mathematics (approximately 9 through 16 years of age). The system is mainly used in urban school districts of the Northeast United States. Students use it in lab classes that they attend periodically, or for doing homework at night.

We collected data from school year September 2010 to September 2011, which consisted of 15,931 students who solved at least 20 problems within ASSISTments. We filtered out skills that have fewer than 50 students. As a result, we have 2,363,982 data records. Each data record is recorded right after a student answered a problem, and logged relevant information including the identity of the student, the problem identity and skills required to solve it, the correctness of the student’s first response to this problem, the duration the student spent on this problem, and the timestamp when the student start and finish solving this problem.

For this task, we defined a student as retaining a skill if he was able to respond correctly after approximately a week. We instantiated a week as any duration between 5 and 10 days, and choose the time interval of 5-10 days as our objects to analyze. We randomly selected one fourth of the students as training data, which result in 27,468 final data records. Note that less than 5% of the data are relevant for training a model of student retention. Thus, this problem requires large data sets.

Student Retention Analysis

RQ1: Is student retention predictable?

To answer this question, we built a logistic regression model, using the 27,468 data points with delayed practice opportunities described previously. The dependent variable is whether the student responded correctly on this delayed outcome. We used user identity (user_id) and skill identity (skill_id) as factors (fixed effets) in this model. We used the following features as covariates, treating incorrect responses as a 0 and correct responses as a 1:

- *n_correct*: the number of prior student correct responses on this skill; This feature along with *n_incorrect*, the number of prior incorrect responses on this skill are both used in PFA models;

- *n_day_seen*: the number of distinct days on which students practiced this skill. This feature distinguishes the students who practiced more days with fewer opportunities each day from those who practiced fewer days but more intensely, and allow us to evaluate the difference between these two situations. This feature was designed to capture certain spaced practice effect in students data;

- *e_mean_performance*: the exponential moving mean of students' previous performances, using a decay of 0.7. For a given student and a given skill, use *opp* to represent the opportunity count the student has on this skill, we compute the exponential moving mean of students' previous performance using formula: *e_mean_performance(opp) = e_mean_performance(opp-1)*0.7 + correctness(opp)*0.3*. The exponential moving mean method allows us to examine current status with a decaying memory of history data. The number 0.7 is selected based on experimenting with different values.

- *e_mean_time*: the exponential moving mean of students' previous response time, using a decay of 0.7. Similar with *e_mean_performance*, for a given student and a given skill, the formula of the exponential moving mean of students' previous response time is: *e_mean_time(opp) = e_mean_time(opp-1)*0.7 + response_time(opp)*0.3*;

- *slope_3*: the slope of students' most recent three performances. The slope information helps capture the influence of recent trends of student performance;

- *delay_since_last*: the number of days since the student last saw the skill. This feature was designed to account for a gradual forgetting of information by the student;

- *problem_difficulty*: the difficulty of the problem. The problem_difficulty term is actually the problem easiness in our model, since it is represented using the percent correct for this problem across all students. The higher this value is, the more likely the problem can be answered correctly.

It is important to note that the features were computed across all of the data, not just the items on which the student had not practiced the skill for 5 to 10 days. For example, the n_correct feature is computed across *all* of the student practices on the skill, not just those practices with a 5 to 10 day delay interval. However, we only create a row in our data set for such delayed retention items (thus there are 27,468 rows). After training the model on the ASSISTments data, we got a R^2 of 0.25. Since this model fit represents training-data fit, it is optimistic. But the model fit is at least strong enough to conclude that student retention appears to be predictable.

The Beta coefficient values and p-values for each covariate are shown in Table 4.1.1.

In this table, the positive B values mean the larger the covariate is, the more likely the student respond to this problem correctly. To our surprise, the influence of the *n_correct* and the *n_incorrect* features are not reliably different than 0. The features *n_day_seen* and *e_mean_performance*, on the contrary, are reliable predictors of student retention. In other words, for predicting long-term retention, the number of days on which the student practiced the skill is important, as is his recent performance. This result is consistent with cognitive "spaced practice effect" result (Perruchet, 1989). The raw number of correct and incorrect responses is not a meaningful predictor. We expected that response time would be relevant to retention, due to its connection to automaticity and mastery (Anderson, 1993).

Table 4.1.1. Parameter table of covariates in Model1.

Covariate	***B***	**p-value**
n_correct	-0.003	.330
n_incorrect	-0.005	.245
n_day_seen	0.055	.000
e_mean_performance	0.813	.000
e_mean_time	0.073	.043
slope_3	-0.033	.444
delay_since_last	-0.015	.182
problem_difficulty	5.926	.000

From the likelihood ratio tests of the training set, we found that the *skill_id* and *user_id* are also both important features in this model. This indicates that student performance on retention items varies by skill and by student. It is tempting to claim that retention varies by student, but this claim is premature as the user_id factor models student performance on retention items. However, such performance is composed of how well the student learned the material as well as how much of that knowledge was retained. A student could have a strong memory, but if he never learned the material his user_id factor would be low. Therefore, user_id does not solely represent retention.

To strengthen the results, we built test set to validate the model. Since the users of testing set are different from those of the training set, we cannot look up user parameters directly for users in the testing set. Instead, we use the mean value of user parameters of the model as an approximation of the user parameter in the testing set. We also did the same thing for the skills that only appear in the testing set.

The R2 of this model on the testing set is 0.17, indicating a reasonable model fit in-line with other attempts at using PFA

RQ2: Does forgetting vary by student?

We would like to separate the impact of the user_id feature into student knowledge and student retention. To accomplish this task, first, we started from the logistic regression model that we used in section 0, removed the factor *user_id* and substituted a covariate *non_1st_pcorrect*. The feature *non_1st_pcorrect* is the percent of a student's non-first attempts of the day that are correct. The intuition is that a student's first attempt on a skill each day is the one that is most affected by retention. By considering the student's overall performance, but excluding these items, we are estimating the student's proficiency in the domain in a way that is less contaminated with forgetting, and is thus a purer estimate of the student's knowledge. We trained this model on the same data as the previous model. The feature *non_1st_pcorrect* has an estimated Beta coefficient of 3.878, with a *p*-value 0.000. We got an R^2 of 0.210 on the data, which is a reasonable model fit. The difference in model fit is caused by the substituting the percent correct, on non-first encounters, for user_id.

We were curious as to the cause of this difference in model fits, and investigated the residuals from our model. The question is whether the residual was systematic, and could be predicted by user_id. We fit a general linear model with *user_id* as a random effect, and the residual as the dependent variable. The R^2 of this model is 0.235. Thus, the residual in our model, after accounting for student overall percent correct in contexts where forgetting was minimal, does vary systematically by user_id. Thus it appears that there is some construct beyond performance, such as forgetting, that varies by student.

Although it is tempting to claim this term represents student forgetting, it is necessary to validate the construct (Crocker & Algina, 1986) we have modeled. To test whether we have modeled retention, we first extracted the student random effects from our GLM. We then computed the correlation between that term, and each student's difference in performance between the first and second question on a skill that occurs each day. Our belief is that this difference in performance is related to student forgetting, since a large increase in performance from the first to the second item suggests the student is remembering old information. Unfortunately, the correlation between these terms was negligible, so we are still searching for what our per-student effect is actually modeling.

CONTRIBUTIONS

This paper makes three main contributions. First, the mastery learning notion is expanded to take into account the long-term effect of learning. In comparison to the traditional view that Corbett and Anderson brought up in their seminal work (Corbett & Anderson, 1995), which looks at only the immediate knowledge, this paper looks at broader notion of knowing a skill.

The second contribution this paper makes is extending the PFA model (Pavlik, Cen, & Koedinger, 2009) with features that are likely to be relevant for retention. Most prior work has focused on concepts such as item difficulty or amount of assistance required to solve a problem. However, those features focus on student performance and properties of items, not on broad characterizations of performance. Our study confirmed that the long-term knowledge appears to vary by skill, and possibly by student. In addition, the number of days on which a student practiced a skill is relevant, and could be an important feature in directing ITS decision making to enhance retention. This result confirms the spaced practice effects in a larger scope; also we found that the number of correct responses seem to be not so important in predicting knowledge retention.

The third contribution this paper makes is on discovering a new problem that is actionable by ITS. Previous student models focus on estimating student current knowledge, which is powerful for EDM, and an efficient use of data for testing a model, but provides limited guidance for tutorial decision making. This paper proposed a diagram of ITS action cycle that can be used to discover new problems in the EDM field that can lead to higher mastery learning rate in ITS systems.

One goal of EDM is to address questions that are relevant for tutorial decision making of ITS. Currently, many ITS simply present a sequence of problems and evaluate student performance right after the student finished these problems to see if the student mastered the given skill. This process does not have the mechanism for the system to review students' knowledge after a time period, nor know about students' long term performance. It is dangerous for ITS to promote a student on the basis of short term performance. We propose the follows diagram shown in Fig. 4.1.1, which allows ITS to aim for students long-term mastery learning.

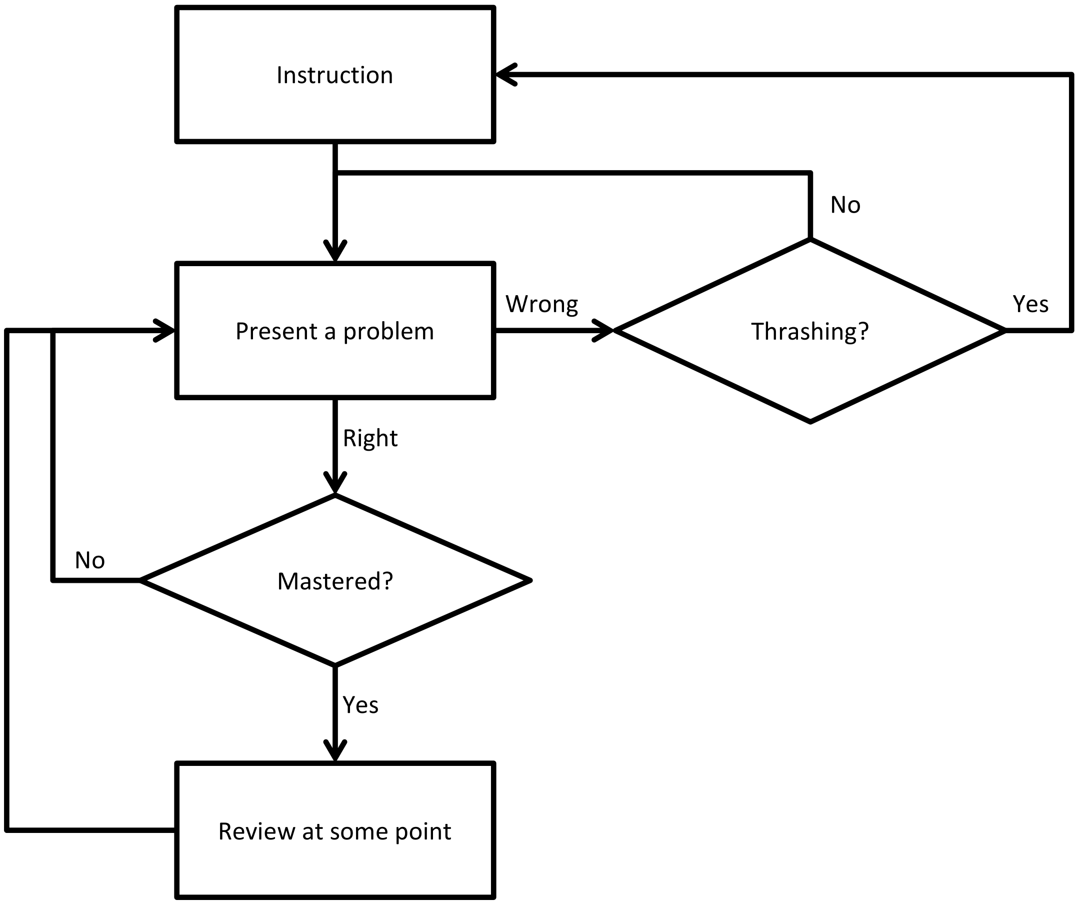


Fig. 4.1.1. Enhanced ITS mastery learning cycle

This paper focuses on the diamond on the left side, whether the student has mastered a skill. Rather than using local criteria to decide whether mastery has occurred, we trained a model to decide mastery based on predicted performance. Beyond this EDM work, some review mechanism for ITS seems warranted, as for cumulative domains, such as mathematics, ensuring that students have retained their knowledge is critical. Correspondingly, we have added a review mechanism to ASSISTments.

Another interesting EDM problem is the diamond on the right: when is a student likely to fail to master a skill in a timely manner and (statistically) exhibit negative behaviors such as gaming? We have made progress on this problem as well, and dubbed the phenomenon "thrashing." If a student is unlikely to master a skill via problem solving, it is essential to do something else, such as peer tutoring, having the teacher intervene, or providing instruction within the ITS.

What we like about both of these problems is that they are rich challenge problems for EDM, and provide actionable information for ITS to make use of in their decision making. If a student appears likely to retain a skill, it is probably not necessary to keep presenting items. If a student is likely to not master a skill, it is probably not productive to keep presenting problems.

Future Work and Conclusions

There are three questions that we are interested in exploring. First, do students vary in how quickly they forget? Our first attempt at teasing apart the user_id factor gave inconclusive results, but this area is important enough to warrant further study. Another issue that we are interested in addressing is what are additional features that relate to forgetting? The field of psychology is rich in ideas, but there has been little existing work in student modeling.

Finally, we would like to deploy this model to a working ITS in the field. On one hand, this can help verify the model; on the other hand, this could be used to improve the ITS systems to help student achieving long-term mastery learning.

This paper present an ITS mastery learning diagram, which brings up useful problems in EDM that needs more work. In this paper we concentrate on estimating student knowledge retention and discovered some useful features for this task. Also, we were able to conclude student long-term performance is predictable, even when a student's ability to remember a skill comes in to play.

4.2 The Effect of Automatic Reassessment and Relearning

Intelligent Tutoring Systems (ITS) give assessments to estimate a student's current knowledge. A great deal of work in the past years, (e.g. KDD Cup 2010) has focused on predict students immediate next performance, while what is important is will the student retain that knowledge for later use. Some previous studies have started to investigate this question by trying to predict student retention after a time interval of several days. We created a novel system that would automatically reassess and allow students to relearn the material to enhance a student's long-term knowledge. It is showed before that this intervention raised student learning, and now we are wondering if it also makes assessment of student long-term knowledge better (i.e, more predictive power). The result shows that the reassessment and relearning information is very useful in assessing student long-term knowledge.

This chapter has been published as a short paper at the following venue:

Wang, Y. & Heffernan, N. (2014). The Effect of Automatic Reassessment and Relearning on Assessing Student Long-term Knowledge in Mathematics. In *Proceedings of the 12th International Conference on Intelligent Tutoring Systems*. pp 490-495. (Wang & Heffernan, 2014)

Introduction

The ASSISTments project is premised on the notion our schools are asked to do too much testing. Every minute testing is a minute stolen from instruction. The solution is to use data from students learning for assessment purposes. Intelligent Tutoring Systems (ITS) give assessments to estimate student current knowledge and predicts student performance on the immediate next action has been investigated by many researchers. But what if our goal is not to ask "do they know this right now?" but "will they retain this knowledge later?". This is a more important question because the purpose of education is to teach students so that they can retain it rather than immediately understand it but quickly forget. Some previous studies (Wang & Beck, 2012) (Xiong, Li, &

Beck, 2013) have investigated this question by trying to predict student performance after a several days interval. In this paper, instead, we are trying to predict student performance after a much longer – six months interval.

Compared to traditional assessment, the dynamic assessment (Sternburg & Grigorenko, 2002) that we are conducted in this study utilizes the amount of assistance that students require to judge the depth of student knowledge. We would not be the first to show that letting students learn could help assessing. Different researchers showed that by offering increasingly more explicit prewritten hints in response to incorrect responses, better assessment can be achieved (Brown, Bryant, & Campione, 1983) (Campione & Brown, 1985) (Attali, 2011). ASSISTments itself has been used in the past along similar lines (Feng, Beck, Heffernan, & Koedinger, 2008) and has been shown that we can better predict students state test scores if we use the number of hints, their responses and other student data.

We created a novel system that would automatically reassess and allow students to relearn the material to enhance students' long-term knowledge. We call it the Automatic Reassessment and Relearning System (ARRS). Details on how ARRS works can be found here (ARRS study). The ARRS system gives us an opportunity to investigate two interesting questions. First, do the models for assessing student knowledge retention several days later perform differently from those for assessing student knowledge retention after a longer time interval (six months)? Second, can we do better in assessing student knowledge retention after six months by utilizing the extra information gathered from the ARRS system? The main difference between this study and previous ones is that not only features of student learning behavior, but also features of student relearning behavior were investigated.

Different logistic regression models were built and analyzed to address these two questions. The result showed that given the same feature set, higher accuracy can be achieved in assessing shorter interval knowledge retention than the longer interval retention, which indicates that assessing longer interval retention could be a harder task. With the extra information of student reassessment and relearning, however, we were able to assess student longer interval retention even better than the shorter interval retention. This result suggests that reassessment and relearning information is very useful in assessing student long-term knowledge.

Methods

The Tutoring System and Dataset

The data used here came from two ARRS experiment classes in the ASSISTments platform in school year 2010-2011. ARRS is a sub-system build in the ASSISTment platform, which automatically reassess student a week later, a month later, and then finally two months after a student originally masters a skill (master here means achieve a preset level -- typically three consecutive correct answers). If students fail the reassessment, they will be given an opportunity to relearn the topic until master it again.

There were 128 students, 33 skills and 53,449 data instances in this experiment. Students were separated into groups 1 and 2, and skills were separated into groups A and B. At the beginning of the experiment, all students completed a first assignment of each skill. Then group 1 students did group A skills assignments in the ARRS while group 2 students did group B skills assignments in the ARRS. After six months, all students were given a one item per skill posttest.

To simplify the analysis, in this study we focused on the first reassessment and relearning phase, that means only data from the first assignment (*first phase*) and the one week later reassessment and relearning assignment (*second phase*) were included in this study. We also excluded 29 students since they missed either the first assignment or the posttest for some skills. We excluded student skill pairs in the ARRS condition where seven days later reassessment or relearning was not finished. These data pre-processing result in 1,538 student skill pairs for the control condition and 1,587 student skill pairs for the ARRS condition.

Models and Analysis

Logistic regression models were built to assessing student long term knowledge. Features includes the prior knowledge *firstp_pretest*, information of student's original learning process: *firstp_avg_correct, firstp_avg_phint, firstp_avg_attempt, firstp_nquestions*, the prior knowledge at seven days later *secondp_pretest*, and information of student re-learning process: *secondp_avg_correct, secondp_avg_phint, secondp_avg_attempt, secondp_nquestions*. Forward input stepwise procedure was conducted to eliminate useless features.

We used The Root Mean Squared Error (RMSE) of predicting a posttest score as a measure of assessing accuracy. *secondp_pretest* was the target for assessing shorter term retention, and *posttest* was the target when assessing longer term retention.

A 5- fold cross validation was done for all of the models. That is, we randomly separated all student skill pairs into five folds, and ran all the models five times. Each time the models were trained on four folds and tested on the remaining one fold.

RQ1: Do the models for assessing student knowledge retention several days later perform differently from those for assessing student knowledge retention after a longer time interval?

To answer this question, we built two comparable models as shown in Table 4.2.1. The Shorter-term_Phase1_Model used features from the first assignment to predict student knowledge retention one week later, while the Longer-term_Phase1_Model used features from the first assignment to predict student knowledge retention six months later. To avoid the influence of the relearning in predicting the longer term knowledge retention, we used control group data to evaluate the Longer-term_Phase1_Model. And we used ARRS group data to evaluate the Shorter-term_Phase1_Model because there is no data on control group's shorter term knowledge retention.

Table 4.2.1. Short-term_Phase1_Model (SP1) vs. Long-term_Phase1_Model (LP1)

Model	***Dependent***	***Data***	***Feature Selected***	***RMSE***
SP1	*phase2_pretest*	ARRS	*firstp_avg_correct* *firstp_avg_attempt*	0.4049
LP1	*posttest*	Control	*firstp_avg_correct* *firstp_avg_phint* *firstp_avg_attempt*	0.4296

Since both conditions had the same group of students, we were able to compute a student level paired t-test to determine whether the RMSE difference between these two models was statistically reliable. The result is statistically reliable, t(98) = 2.58, p = 0.01. The result suggests that the six months knowledge retention is harder to assess than the seven days knowledge retention is not surprising, and some may say it's trivial. However, the short term model helped us in setting up a baseline for the models of assessing longer term retention to compare with.

RQ2: Can we do better in assessing student knowledge retention six months later by utilizing the extra information gathered from the ARRS system?

Similar to RQ1, we built several logistic regression models as shown in Table 4.2.2. All these models predicted the posttest score using the ARRS group data.

Phase1and2_Model used all the features from both the first assignment, and the ARRS assignment seven days later in a single stepwise logistic regression model.

To improve upon the Phase1and2_Model, we considered the fact that some of the ARRS student skill pairs do not have relearning features because they answered their reassessment question correctly. This caused large amount of missing data when we use a single model to describe all the ARRS data. We then built a model called Phase1and2_Combined_Model, which was the combination of two sub-models: Phase1and2_norelearning_Model, and Phase1and2_relearning_Model. The Phase1and2_norelearning_Model ran on the student skill pairs in which the students did not need to relearn the material for the skill, while the Phase1and2_relearning_Model ran on the student skill pairs in which the students needed and finished the relearning assignment.

Table 4.2.2. Phase1and2_Model vs. Phase1and2_Combined_Model

Model	***Data***	***Feature Selected***	***RMSE***
Phase1and2_Model	ARRS	*firstp_avg_correct* *firstp_avg_phint* *secondp_avg_correct* *secondp_nquestions*	0.3886
Phase1and2_Combined_Model	ARRS	--	0.3861
Phase1and2_norelearning_Model	ARRS no relearn	*firstp_avg_correct* *firstp_avg_phint*	--
Phase1and2_relearning_Model	ARRS relearning	*firstp_avg_correct* *secondp_avg_correct* *secondp_nquestions*	--

In Table 4.2.2, the feature column of Phase1and2_Combined_Model is empty, because it is the combination of two different models (Phase1and2_norelearning_Model and Phase1and2_relearning_Model). Also, the RMSE column for Phase1and2_norelearning_Model and Phase1and2_relearning_Model is empty, because the dataset of these two models are different with other models in the table, thus the RMSEs of these two models are not comparable with other models'.

Until now, we could draw two conclusions. First, by comparing Longer-term_Phase1_Model in Table 4.2.1 and Phase1and2_Model in Table 4.2.2, we observed the extra features gathered from ARRS did improve the accuracy in assessing student long term knowledge. A student level paired t-test suggested this improvement was statistically reliable, t(98) = 4.61, p < .001. Compared to the short term model Shorter-term_phase1_Model, however, this new long term model has a better, but not reliably better, RMSE, t(98) = 1.82, p = 0.07. Second, by separating models according to whether or not a student needed relearning for one skill, we were able to further improve the model for assessing long term knowledge. Although the improvement between the Phase1and2_Model and the Phase1and2_Combined_Model was not reliable, t(98) = 1.05, p = 0.30, amazingly, the Phase1and2_Combined_Model was able to reliably improve upon the short term model Shorter-term_Phase1_Model in Table 1, t(98) = 2.02, p < 0.05. This proved again the importance of the relearning features, especially the average correctness in the relearning phase and the number of questions students need to relearn a material.

Discussion and Future Work

In this paper, we compared model performance between assessing student shorter interval knowledge retention and longer interval knowledge retention. Results suggested that longer interval knowledge retention is harder to assess. We then investigated the effect of the extra features gathered from ARRS and concluded that relearning features are useful in assessing long-term knowledge retention.

One limitation of this work is the amount of data. The experiment was a pioneer study of ARRS, and has only several thousands of data instances. Verifying the result of this study in a larger dataset or a different tutoring system could be helpful.

Another limitation is that we only used the information from the one week later reassessment and relearning phase. Future study could further investigate the predicting power of data from the later phases of two weeks, one month, and two months later.

Contributions

This paper analyzed data gathered from a novel system, which automatically reassesses student knowledge and allows them to relearn the material, to evaluate its power in assessing student long-term knowledge and makes several contributions.

First, assessing student current knowledge has been investigated by researchers in ITS for many years. Recently some researchers have discovered the difference between assessing current knowledge and knowledge retention several days later. We further explored this topic, and compared the model performance between assessing student shorter interval retention (seven days later) and longer interval retention (six months later). Results showed that the latter is a harder problem to address. We then concentrated on improving the assessing accuracy of the longer interval retention.

Second, for the task of predict student knowledge six months later, compared to other dynamic testing methods, we not only looked at the assessment power of the features in student learning process, but also the assessment power of these features in student relearning process. To do so, instead of using data from a single session, we also used data from a second session at one week later. We built and analyzed different stepwise logistic regression models to see if number of problems (learning speed) and other features (hints and attempt) of the second session help the prediction. Result shows that having data from both the learning session and the relearning session lead to better prediction. More interestingly, it shows that tracking how much relearning (measured by the number of problems student need to finish before re-mastery a skill) students need was a useful predict. This indicates that student relearning time is a useful indicator of the depth of student knowledge.

Furthermore, we found that making separate models according to weather or not a student needs relearning for a skill gives better prediction, and surprisingly, even better than predicting students' shorter interval retention.

Chapter 5: Other Work Related to Student Modeling

5.1 Attempts to Improve Affect Detectors

The Baker et al affect detectors on boredom, frustration, confusion and engagement concentration with ASSIStments dataset have been greatly studied and were used to predict state tests scores, college enrollment, and even whether a student majors in a STEM field. In this section, we presented three attempts to improve upon current affect detectors. The first attempt analyzed the effect of missing skill tags in the dataset to the accuracy of the affect detectors. The result shows a small improvement on all of the four detectors after correctly tagged the missing skill values. The second attempt added four features that were related to student classes for feature selection. The third attempt added two features that described information about student common wrong answers for feature selection. Result showed that two out of the four detectors were improved by adding the new features.

This chapter has been published as a short paper at the following venue:

Wang, Y., Heffernan, N. & Heffernan, C. (2015). Towards Better Affect Detectors: Effect of Missing Skills, Class Features and Common Wrong Answers. In *Proceedings of the 5th International Learning Analytics & Knowledge Conference*. pp 31-35. (Wang, Heffernan, & Heffernan, 2015)

Introduction

Affect detection in educational systems is important in understanding different student affect and their impacts. Correctly detect student affect could also potentially help guide interventions to improve student engagement, reduce student confusion, frustration or boredom. In recent years, sensor free affect detection (D'Mello, Craig, Witherspoon, McDaniel, & Graesser, 2008) (Baker, et al., 2012) (Sabourin, Mott, & Lester, 2011) has gained more and more attention. This approach can be easily applied to various real-world educational systems for students' affect detection without requirement of sensor systems which could be expensive and less robust to classroom conditions.

Currently, the best sensor free affect detectors were built by Baker et al (Baker, et al., 2012) on the cognitive tutor dataset, which can be used detect student engaged concentration (Craig, Graesser, Sullins, & Gholson, 2004), confusion, frustration and boredom solely from students' log data that already exists inside current educational systems, including students' interactions within the interface. The detectors were then rebuilt using the ASSISTments platform dataset, and helped various of researches, including San Pedro et al's work on how affect is shaped by knowledge (San Pedro, Baker, Gowda, & Heffernan, 2013). Hawkins et al investigated how interface design influences affect (Hawkins, Heffernan, & Baker, 2013). Pardos et al (Pardos, Baker, San Pedro, Gowda, & Gowda, 2013) investigated how affect influences learning, and used affect states to predict state tests scores. San Pedro et al (San Pedro, Baker, Bowers, & Heffernan, 2013) even used the affect detector to analyze how affect influences the eventual decision to attend college, including college enrollments and whether a student majors in a STEM field.

The original sensor free affect detection method has produced detectors that are better than chance, but not substantially better. However, after three years since built, there is not much reporting about improvement upon the original sensor free affect detectors. In general, a feature based learning analytics model could be improved in three different ways: 1) generating more accurate features; 2) further features engineering; 3) search for alternate methods for aggregating data. In this paper, we tried three attempts in the first two ways to improve the detectors.

The first attempt was correcting the missing skill tags in the ASSISTments dataset. We found that the model was based upon the ASSISTments data that included almost 1/4 of questions that were not tagged with any skill. We decided to run experiments to see if by tagging these questions with correct skills the detectors could perform dramatically better. The result could tell us how sensitive these detectors are to missing skill values, and could potentially provide us with more accurate detectors. We refit all of the four affect detectors and reported our findings. Somewhat surprisingly, getting these questions retagged hardly increased our ability to predict affect state. This result suggests that it should be safe to use the affect detectors with certain amount of skill tags missing.

The second attempt we tried was adding four features that describe information about student classes into the feature pool of the affect detectors for feature selection. Class is one of the most common objects that are studied in the educational field. However, when building student models such as models for predicting student performance or estimating student affective states, class level features are rarely considered. Wang et al (Wang & Beck, 2013) showed in a student model that class level parameters could be useful. In our experiments, result showed that class features also helped improve two out of the four affect detectors.

Current Massive Open Online Courses (MOOCs) often collect features that the Learning Analytics community is already taking advantage of, such as the correctness of student performance; the total time students take in answering problems, the number of hints they ask for, or number of attempts they make. In this paper, however, we took a look at a novel feature: whether the student has made a common wrong answer. Previous research in our lab showed that the particular common wrong answers are predictive of “next problem correctness”. Also that the group of data logs in which all the answers are not common (namely uncommon wrong answer group) were more likely to be followed by a wrong attempt on the next problem for this particular student and skill. We could like to see in this study, whether or not common wrong answers could also help improve affect detectors.

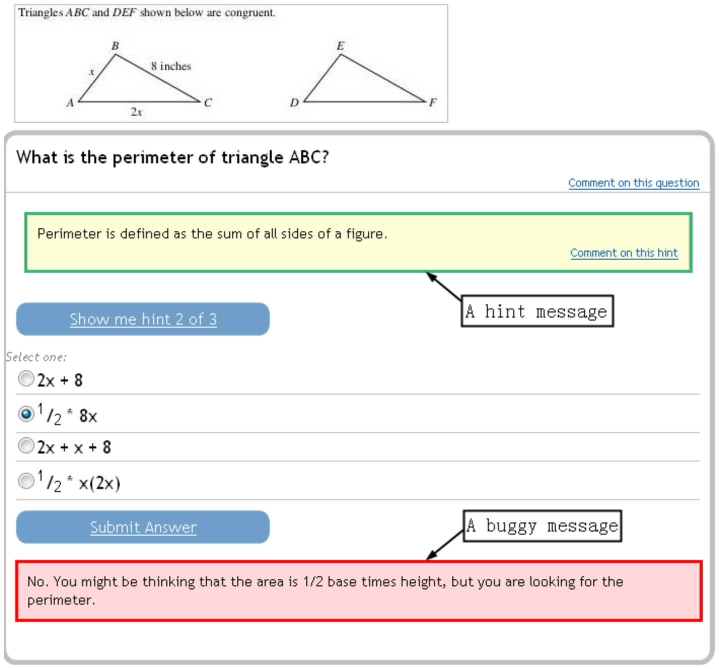
We are not the first to think that common wrong answer is useful. Many researchers at ETS and others used this idea. But their work is only for multiple choice questions. In most of the MOOCs and tutoring platform such as ASSISTments, more generalized common wrong answers could be easily studied with the historical answering text data that could be relied upon. We operationalized a common wrong answer as one that at least 10% of those students that got the question wrong gave that answer. All answers that were not common were named "uncommon wrong answers". The third attempt we tried to improve sensor free affect detectors was adding two features that describe information about how common the student answer was into the feature pool of the affect detectors for feature selection. Result suggested that common wrong answer was useful information in estimating student affects.

Methodology

Dataset and features

The data used in the analysis presented here came from the ASSISTments system, a freely available web-based tutoring system for 4th through 10th grade mathematics. The system gives tutorial assistance if a student makes a wrong attempt or asks for help. Fig. 5.1.1. shows an example of a hint, which is one type of assistance. A second type of assistance is presented if they click on (or type in) an incorrect answer, at which point the student is given feedback that they answered incorrectly (sometimes, but by no means always, students will get a context-sensitive message we call a "buggy message").

Our dataset also provides a special type of assistance called scaffolding as in Fig. 5.1.2. For those problems with scaffolding questions, if a student gets the original question wrong, the system will give the student a series of questions we call "scaffolding" that walk the student through the steps of solving the original question.


Triangles *ABC* and *DEF* shown below are congruent.

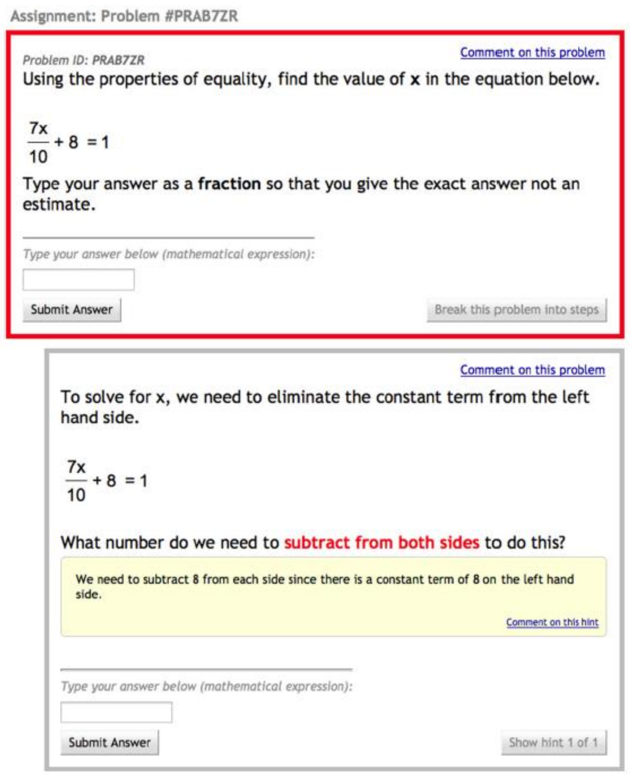
What is the perimeter of triangle ABC?

Perimeter is defined as the sum of all sides of a figure.

- 2x + 8
- [✓ $^1/_2$ * 8x]
- 2x + x + 8
- $^1/_2$ * x(2x)

No. You might be thinking that the area is 1/2 base times height, but you are looking for the perimeter.

Fig. 5.1.1. Hint and buggy message in ASSISTments


Assignment: Problem #PRAB7ZR

Problem ID: PRAB7ZR

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

$$\frac{7x}{10} + 8 = 1$$

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

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

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

$$\frac{7x}{10} + 8 = 1$$

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

We need to subtract 8 from each side since there is a constant term of 8 on the left hand side.

Fig. 5.1.2. Scaffolding in ASSISTments

The students and features in this study was same as the previous studies of sensor free affect detectors of ASSISTments data (Ocumpaugh, Baker, Gowda, Heffernan, & Heffernan, 2014). Students in this study are drawn from middle schools in the Northeastern United States that represent three different populations: rural, suburban and urban students. The ground truth labels of student affect are obtained using quantitative field observations (QFOs) of educationally relevant affect categories. The QFOs were obtained using the Baker-Rodrigo Observation Method Protocol (Ocumpaugh, Baker, & Rodrigo, 2012). BROMP coders record the affective state of each student individually, in a predetermined order that is enforced by the Human Affect Recording Tool (HART) application (Baker, et al., 2012). The BROMP coding scheme and HART recording tool allowed us to accurately match each field observation window to the 20-second clip of that student's interactions that are recorded in the software's log file.

Same as in the original affect detection paper, student actions within the educational system were synchronized to the field observations according to the same internet-time server to distill features for affect detection. 58 features, including temporal features, skill-based features, features based on the number of errors, the number of correct answers and the number of hints requested, were developed using the action data during and prior to the twenty seconds prior to data entry by the observer. These 58 features were then aggregated using mean, min, max and sum aggregator across the action to generate a total of 232 features that were used in the development of the affect detectors.

Examples of features can be seen in the result section.

Classification methods and evaluation measures

In order to classify all four affect states: boredom, engaging concentration, frustration and confusion, each affective state was predicted separately by applying standard data mining classification algorithms (including LinearRegression, decision trees, step regression, Naïve

Bayes, JRip, J48, REPTree, BayesianLogisticRegression, and K*) within RapidMiner 5 (Mierswa, Wurst, Klinkenberg, Scholz, & Euler, 2006), a software system that facilitates data mining analysis. This resulted in four detectors, one for boredom, confusion, engaged concentration, and frustration respectively.

Each detector was evaluated using five-fold student-level cross validation. In this process, students are split randomly into five groups. Then, for each possible combination, a detector is developed using data from four groups of students before being tested on the fifth "held out" group of students. This method of cross validation insured that the detectors will be accurate for new students.

For model selection and evaluation purpose, goodness metrics were used. Since no single metric fully captures every aspect of a model, two goodness metrics were used to determine which detectors were most effective: Cohen's Kappa (Cohen, 1960) and A' (Hanley & McNeil, 1982). Cohen's Kappa assesses the degree to which the detector is better than chance at identifying which clips involve a specific affective state. A Kappa of 0 indicates that the detector performs at chance, and a Kappa of 1 indicates that the detector performs perfectly. A Kappa of 0.2 would indicate that the detector is 20% better than chance. A' is the probability that the algorithm will correctly identify whether a specific affective state is present or absent in a specific clip. A model with an A' of 0.5 performs at chance, and a model with an A' of 1.0 performs perfectly. In these analyses, both of the goodness metrics was applied at the level of the clip (the 20-second interval of the log file, which is the basic unit for our analysis).

A forward selection feature selection process was conducted for each of the machine learning algorithms using cross-validated kappa on the original (e.g. non-re-sampled) data set as the goodness metric. In this process, the single feature that most improves model goodness is added into the final model repeatedly until adding additional features no longer improves model goodness. Prior to feature selection, all features with cross-validated kappa equal to or below zero in a single-feature model were omitted from consideration.

Missing skill problem

One of the biggest problems in "big-data" analysis is the missing data problem. In our dataset, we noticed that around 24% of the data has missing skill tags. All the logs that missed skill tags were treated as a single skill: "no-skill". Since skill is one of the most important features in the educational dataset, we are intrigued to see how much improvement could be achieved if all the skills were tagged properly.

To do so, we exported all 388 problems that had no skill tags in our dataset and manually tagged them with correct skills. We than regenerated all of the 232 features with the new dataset with more accurate skill tags and rebuilt all of the four affect detectors. The goal is to find out how much improvement, if any, can be achieved by generating more accurate skill related features and how sensitive the sensor free affect detectors were to missing skill tags.

Class features

Currently, all of the features that were used in the sensor free affect detectors were generated for each student independently. However, student affect could be influenced by the behavior of the class that they belonged to. Intuitively, a student could feel less frustrated when he/she was better than most of other students in his/her class. By the same token, a student was more likely to feel bored if we observed that most of the other students were bored. In order to capture some of the information related to student class, we developed four new features:

- pCorrectClass: the percentage of correctness of all previous questions answered in the class;

- pCorrectStudentPercentileRank: the student percentile rank of average performance in the class so far for this student;
- nClassData: number of previous data points for this class;
- nClassStudent: number of students has been seen in this class so far;

These four features can be seperated into two groups: pCorrectClass was designed to describe the average performance of the class that the student belonged to. This could potentially be useful for normalizing the effect of student performance on student affect. nClassData was designed to evaluate the robustness of the feature pCorrectClass. When the number of data points for this class was too small, we probably should trust less about the pCorrectClass feature.

The pCorrectStudentPercentileRank feature was designed to describe how good the student did in compare to his/her peer classmates. nClassStudent and nClassData could be used together to imply how much we should trust the feature pCorrectStudentPercentileRank.

We evaluated the effect of these four class features by adding them into the original 232 feature sets. The same forward inputting features selection, multiple classification algorithms and evaluation measures were used to build and evaluate the new affect detectors with class features.

To make sure the new detectors' performance can be directly compared with the original detectors. We used unique-id to ensure the data re-sampling and cross-validation folders were exactly the same between the new and old affect detectors.

Common Wrong Answers

As discussed in the introduction section, common wrong answer is a novel feature that has great potential in improving educational models. Intuitively, students that answered a common wrong answer could indicate certain misunderstanding and/or understanding of the problem. Previous work in our lab discovered a huge difference in students' next performance between a common wrong answer and a non-common wrong answer. But the effect of common wrong answer on student affects has never been studied.

We developed two different features that were related to the common wrong answers for the detectors to select from:

- answerPercentage: the percentage of this particular answer among all logs that answered this problem;
- commonWrongAnswer: a binary feature describes whether or not this answer is a common wrong answer; A common wrong answer was defined by answers that were given by at least 10% of the students that got the question wrong;

We evaluated the effect of common wrong answer features using the same method as the class features. First, the two features that were related to common wrong answers were added into the original 232 features. Then unique-id was used to generate exactly the same re-sampling and cross-validation dataset. Finally the same classification and evaluation measures were used to build and evaluate the new affect detectors with common wrong answer features.

Experimental Results

Effect of missing skills

After correcting missing skills, the detector performance was improved in all of the four detectors, but only for a small amount. This indicated that the sensor free affect detectors could be safely used on dataset with certain amount of missing skill tags, with only a small sacrifice of performance (less than 3% of average Kappa). This was good news for educational systems in which missing skill tags were inevitable (e.g. systems allowing teachers create their own problems without skill tags).

Effect of class features

After adding four class features, detector performance was improved in two out of all four detectors. Full results are shown in Table 5.1.1. For engaged concentration, the best algorithm with class features was the same as the original detectors: J-Rip. The engaged concentration detector achieved an A' of 0.743 and a Kappa of 0.423. For boredom, the best algorithm with class features changed from J48 as in the original detectors to K*. The boredom detector achieved an A' of 0.671 and a Kappa of 0.260.

The bold values in Table 5.1.1 showed the improved model results. For confusion and frustration detectors, the class features were not selected into the final models. Thus the algorithm and features for the confusion and frustration detectors were kept the same as the original detectors.

Table 5.1.1. Effect of class features

Detector	***Original***		With Class Features	
	A′	**Kappa**	**A′**	**Kappa**
Engaged Concentration	0.731	0.417	**0.743**	**0.423**
Confusion	0.625	0.146	0.625	0.146
Frustration	0.597	0.151	0.597	0.151
Boredom	0.662	0.243	**0.671**	**0.260**
Average	0.654	0.239	**0.659**	**0.245**

For the two improved detectors: engaged concentration and boredom, features automatically selected for each of the detectors during machine learning are listed in table 4.

The features for original engaged concentration detector involve actions where the student was more likely to have a history of more first responses but fewer errors and help requests on the skills in the clip. For the engaged concentration detector with class features, the percentage of correctness of the class and the number of data points in the class were selected in addition to the number of main problems done. This could indicate that in modeling concentration, class performance is more important than student individual performance.

The features for boredom detector indicated that bored students were more likely to work slowly but correctly. For the boredom detector with class features, the student percentile rank of average performance in the class and number of students were selected. These features replaced the performance feature that described how many incorrect answers the student answered before, while bring in new important features describing how many help students can get from the system, including whether or not the question is multiple choice question, and how many hints were asked. The result suggests that for boredom detector, student performance can be more effectively represented by students' correctness percentage rank.

Table 5.1.2. The features in the final detectors with class features

Engaged Concentration	
Original	**With Class Features**
Total first responses attempted in the tutor so far.	**The percentage of correctness of all the questions answered in the class so far**
The number of main problems seen in this 20 seconds	**Number of datapoints for this class so far**
The minimal number of first responses during school hours (between 7:00 am and 3:00 pm)	The number of main problems seen in this 20 seconds
The average time spent on first responses in answering scaffolding problems	The average of the number of first responses during school hours (between 7:00 am and 3:00 pm)
The average correctness in this 20 seconds	
The maximum number of previous incorrect actions and help requests for any skill in the clip	

Boredom	
Original	**With Class Features**
The sum of the number of first responses during school hours (between 7:00 am and 3:00 pm)	The sum of the number of first responses during school hours (between 7:00 am and 3:00 pm)
Sum of wrong answers in the past 8 problems	**The number of students has been seen in this class so far**
The average of response times for any skill in the clip	**The student percentile rank of average performance in the class so far for this student**
The average of the number of first responses during school hours (between 7:00 am and 3:00 pm)	The minimal number of multiple choice questions in this 20 seconds
Sum of wrong answers in the past 5 problems	The minimal number of hints in this 20 seconds
	The minimal number of questions that has a help request as the first response

Effect of common wrong answers

After adding two common wrong answer features, detector performance was again improved in two out of the four detectors. This improvement was the largest in our three attempts of improving affect detectors. Full results are shown in Table 5.1.3. The bold values showed the improved model results. For confusion and frustration, the common wrong answer features were not selected into the final models. Thus the algorithm, features and result for the confusion and frustration detectors were kept the same as the original detectors.

The binary version of common wrong answer feature was selected into the engaged concentration detector, students that give common wrong answers are more likely to be effectively working. The more detailed version of common wrong answer features was selected into the boredom detector. Certain, but not all, common wrong answers are related to guessing, which is a common behavior of bored students.

Table 5.1.3. Effect of common wrong answers

Detector	***Original***		With Common Wrong Answers	
	A′	**Kappa**	**A′**	**Kappa**
Engaged Concentration	0.731	0.417	**0.753**	**0.436**
Confusion	0.625	0.146	0.625	0.146
Frustration	0.597	0.151	0.597	0.151
Boredom	0.662	0.243	**0.675**	**0.263**
Average	0.654	0.239	**0.663**	**0.249**

Effect of all three attempts

We also tried all three attempts at the same time, and the results are shown in Table 5.1.4. The bold values showed the improved model results.

Table 5.1.4. Effect of all three attempts

Detector	***Original***		**With corrected skills**	
	A′	**Kappa**	**A′**	**Kappa**
Engaged Concentration	0.731	0.417	**0.754**	**0.436**
Confusion	0.625	0.146	**0.627**	**0.148**
Frustration	0.597	0.151	**0.602**	**0.157**
Boredom	0.662	0.243	**0.676**	**0.264**
Average	0.654	0.239	**0.665**	**0.251**

Discussion and conclusions

In this paper, we presented three attempts to improve current existing sensor free affect detectors with the ASSISTments dataset.

The first attempt analyzed the effect of missing skill tags in the dataset to the accuracy of the affect detectors. Not many researchers pay attention to the performance of models in dataset with missing values in the learning analytic field. The result shows only a small improvement in the detectors after correctly tagged the missing skill values. This suggests that it should be safe to use the affect detectors with certain amount of missing skills.

The second attempt added four features that describe information about student classes into the feature pool for feature selection. Class is one of the common objects that are studied In learning analytics analyses. Result showed that class features helped improve the concentration and the boredom detectors by 3.5% on average Kappa.

The third attempt added two features that describe information about how common the student answer was into the feature pool for feature selection. This approach achieved the best improvement.

This work is still at the early stage. We see it as one of the incremental steps to build a useful tool for understanding and automatically adapting to differences in learner affect. There is still substantial room for improvement in compare with expert coders' Kappa values (around 0.6 or 0.7). More features and different methods could be used to further improve the detectors. In the long-term, we could incorporate these detectors into the ASSISTments platform to help teachers to understand students' affective states or provide interventions aim for better learning outcomes.

Chapter 6: Discussion and Future Work

In this dissertation, we applied various technologies to model different aspects in student learning and behavior and evaluated the models. Contributions include new advanced student models with novel features, and also some scientific conclusions in learning science.

Currently, the models have only been evaluated on the ASSISTments dataset. Although most of the features in this dissertation could be easily generated by common intelligent tutoring systems, it is possible that the results may differ when using data from other tutoring systems. Verifying or comparing the results in different tutoring systems could be helpful.

One issue that we have not yet addressed is how to effectively apply these models in real world. Some of the models we built are easy to compute, and can be used in real time tutoring systems with little effort. But some of them, such as the Student Skill model, take a long time to train. To use these models directly in tutoring systems, offline training needs to be scheduled. To make sure the parameters generated are both efficient and accurate, the amount of data used in offline training needs to be decided carefully for each of the models with experiments. After the models are used in real time tutoring systems, randomized control trial experiments would be useful in further evaluating the models.

Another question that we are interested in exploring is the development of a combining method of different models. Each model has its own advantages and disadvantages. We would like to know if there are rules which can guide us to choose one model over another given certain circumstances.

REFERENCES

Anderson, J. (1993). *Rules of the Mind.* Lawrence Erlbaum.

Arroyo, I., Cooper, D., Burleson, W., & Woolf, B. (2010). Bayesian Networks and Linear Regression Models of Students' Goals, Moods, and Emotions. In *Handbook of Educational Data Mining* (pp. 323-338).

ARRS study. (n.d.). Retrieved from https://sites.google.com/site/assistmentsdata/arrs

Attali, Y. (2011). Immediate Feedback and Opportunity to Revise Answers : Application of a Graded Response IRT Model. *Applied Psychological Measurement*.

Attali, Y., & Powers, D. (2010). Immediate feedback and opportunity to revise answers to open-end questions. *Educational and Psychological Measures*, (pp. 22-35).

Baker, R. S., Corbett, A., & Aleven, V. (2008). More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. *Intelligent Tutoring Systems.*

Baker, R. S., Gowda, S., Wixon, M., Kalka, J., Wagner, A. Z., Salvi, A., et al. (2012). Towards Sensor-free Affect Detection in Cognitive Tutor Algebra. *The 5th International Conference on Educational Data Mining*, (pp. 126-133).

Baker, R., Corbett, A., Gowda, S., Wagner, A., MacLaren, B., Kauffman, L., et al. (2010). Contextual Slip and Prediction of Student Performance After Use of an Intelligent Tutor. *The 18th Annual Conference on User Modeling, Adaptation, and Personalization*, (pp. 52-63).

Baker, R., Corbett, A., Gowda, S., Wagner, A., MacLaren, B., Kauffman, L., et al. (2010). Contextual Slip and Prediction of Student Performance After Use of an Intelligent Tutor. *The 18th Annual Conference on User Modeling, Adaptation, and Personalization*, (pp. 52-63).

Baker, R., Corbett, A., Gowda, S., Wagner, A., MacLaren, B., Kauffman, L., et al. (2010). Contextual Slip and Prediction of Student Performance After Use of an Intelligent Tutor. *the 18th Annual Conference on User Modeling, Adaptation, and Personalization*, (pp. 52-63).

Beck, J. a. (2003). Predicting student help-request behavior in an intelligent tutor for reading. *Ninth International Conference on User Modeling.*

Beck, J. E. (2003). Predicting student help-request behavior in an intelligent tutor for reading. *Ninth International Conference on User Modeling.*

Brown, A. L., Bryant, N., & Campione, J. C. (1983). Preschool children's learning and transfer of matrices problems: Potential for improvement. *the Society for Research in Child Development meetings.* Detroit.

Cain, L., & Willey, R. (1939). The effect of spaced learning on the curve of retention. *Journal of Experimental Psychology*, 209-214.

Campione, J., & Brown, A. (1985). Dynamic assessment: One approach and some initial data. *Technical Report No. 361. Cambridge, MA: Illinois University, Urbana. Center for the Study of Reading*, ED269735.

Cen, H. K., & K., J. B. (n.d.). Learning Factors Analysis – A General Method for Cognitive Model Evaluation and Improvement. *ITS 2006*, (pp. 164-175).

Cen, H., Koedinger, K., & Junker, B. (2006). Learning Factors Analysis – A General Method for Cognitive Model Evaluation and Improvement. *ITS 2006*, (pp. 164-175).

Cohen, J. A. (1960). Coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20 (1), 37-46.

Corbett, A., & Anderson, J. (1995). Knowledge Tracing: Modeling the Acquisition of Proce-dural Knowledge. *User Modeling and User-Adapted Interaction*, (pp. 253-278).

Corbett, A., & Bhatnagar, A. (1997). Student Modeling in the ACT Programming Tutor: Adjusting a Procedural Learning Model with Declarative Knowledge. *User Modeling: Proceedings of the 6th International Conference*, (pp. 243-254).

Craig, S. D., Graesser, A. C., Sullins, J., & Gholson, B. (2004). Affect and learning: an exploratory look into the role of affect in learning with AutoTutor. *Journal of Educational Media*, 29, 3, 241–250.

Crocker, L., & Algina, J. (1986). Introduction to Classical and Modern Test Theory. *Fort Worth: Harcourt Brace Jovanovich College Publishers*, 482.

D'Mello, S. K., Craig, S., Witherspoon, A. W., McDaniel, B. T., & Graesser, A. C. (2008). Automatic Detection of Learner's Affect from Conversational Cues. *User Modeling and User-Adapted Interaction*, 18 (1-2), 45-80.

Ebbinghaus, H. (1885). *Memory: A Contribution to Experimental Psychology.* New York: Teachers College, Columbia.

Feng, M., & Heffernan, N. T. (2010). Can We Get Better Assessment From A Tutoring System Compared to Traditional Paper Testing? Can We Have Our Cake (Better Assessment) And Eat It too (Student Learning During The Test)? *The 10th International Conference on Intelligent Tutoring Systems.*

Feng, M., Beck, J., Heffernan, N., & Koedinger, K. (2008). Can an Intelligent Tutoring Sys-tem Predict Math Proficiency as Well as a Standardized Test? *the 1st International Conference on Education Data Mining*, (pp. 107-116).

Feng, M., Beck, J., Heffernan, N., & Koedinger, K. (2008). Can an Intelligent Tutoring Sys-tem Predict Math Proficiency as Well as a Standardized Test? *the 1st International Conference on Education Data Mining*, (pp. 107-116).

Feng, M., Heffernan, N. T., & Koedinger, K. R. (2006). Predicting state test scores better with intelligent tutoring systems: developing metrics to measure assistance required. *the Eighth International Conference on Intelligent Tutoring Systems*, (pp. 31-40).

Feng, M., Heffernan, N., & Koedinger, K. (2006). Predicting state test scores better with intelligent tutoring systems: developing metrics to measure assistance required. *the Eighth International Conference on Intelligent Tutoring Systems*, (pp. 31-40).

George, B. S., & John, A. E. (1994). Knowledge Taught in School: What Is Remembered? *Review of Educational Research Summer*, vol. 64 no. 2 253-286.

Gong, Y., Beck, J. E., & and Ruiz, C. (2012). Modeling Multiple Distributions of Student Performances to Improve Predictive Accuracy. *the 20th Conference on User Modeling, Adaptation, and Personalization.*

Gong, Y., Beck, J. E., & and Ruiz, C. (2012). Modeling Multiple Distributions of Student Perfor-mances to Improve Predictive Accuracy. *the 20th Conference on User Modeling, Adaptation, and Personalization.*

Gong, Y., Beck, J., & Heffernan, N. (2010). Comparing Knowledge Tracing and Performance Factor Analysis by Using Multiple Model Fitting Procedures. *the 10th International Conference on Intelligent Tutoring Systems*, (pp. 35-44).

Gong, Y., Beck, J., & N.T., H. (2010). Comparing Knowledge Tracing and Performance Factor Analysis by Using Multiple Model Fitting Procedures. *the 10th International Conference on Intelligent Tutoring Systems*, (pp. 35-44).

Gu, J. W. (2014). Personalizing Knowledge Tracing: Should We Individualize Slip, Guess, Prior or Learn rate? *the 12th International Conference on Intelligent Tutoring Systems*, (pp. 647-648).

Gu, J., Wang, Y., & Heffernan, N. T. (2014). Personalizing Knowledge Tracing: Should We Individualize Slip, Guess, Prior or Learn rate? *the 12th International Conference on Intelligent Tutoring Systems*, (pp. 647-648).

Hanley, J., & McNeil, B. (1982). The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology*, 143, 29-36.

Hawkins, W., Heffernan, N. T., & Baker, R. S. (2013). Which is more responsible for boredom in intelligent tutoring systems: students (trait) or problems (state)? *The 5th biannual Conference on Affective Computing and Intelligent Interaction*, (pp. 618-623).

Hawkins, W., Heffernan, N., Wang, Y., & Baker, S. (2013). Extending the Assistance Model: Analyzing the Use of Assistance over Time. *the 6th International Conference on Educational Data Mining*, (pp. 59-66).

Masters, G. N. (1982). A rasch model for partial credit scoring. *Psychometrica*, 149-174.

Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., & Euler, T. (2006). YALE: rapid prototyping for complex data mining tasks. *The 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp. 935-940).

Murphy, K. (2001). The Bayes Net Toolbox for Matlab. *Computing Science and Statistics: Proceedings of Interface*, (p. 33).

Ocumpaugh, J., Baker, R. S., & Rodrigo, M. M. (2012). *Baker-Rodrigo Observation Method Protocol (BROMP) 1.0. Training Manual version 1.0.* New York, NY: EdLab. Manila, Philippines: Ateneo Laboratory for the Learning Sciences.

Ocumpaugh, J., Baker, R., Gowda, S., Heffernan, N., & Heffernan, C. (2014). Population validity for educational data Mining models: A case study in affect detection. *British Journal of Educational Technology*, 45(3), 487–501.

Pardos, Z. A., Baker, R. S., San Pedro, M. O., Gowda, S. M., & Gowda, S. M. (2013). Affective states and state tests: investigating how affect throughout the school year predicts end of year learning outcomes. *The Third International Conference on Learning Analytics and Knowledge*, (pp. 117-124).

Pardos, Z., & Heffernan, N. (2010). Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing. *the 18th International Conference on User Modeling, Adaptation and Personalization*, (pp. 225-266).

Pardos, Z., & Heffernan, N. (2010). Navigating the parameter space of Bayesian Knowledge Tracing models: Visualization of the convergence of the Expectation Maximization algorithm. *the 3rd International Conference on EDM.*

Pardos, Z., & Heffernan, N. (2011). Using HMMs and bagged decision trees to leverage rich features of user and skill from an intelligent tutoring system dataset. *the Journal of Machine Learning Research W & C*.

Pavlik, P., & Anderson, J. (2005). Practice and Forgetting Effects on Vocabulary Memory: An Activation-Based Model of the Spacing Effect. *Cognitive Science*, 559-586.

Pavlik, P., Cen, H., & Koedinger, K. (2009). Performance Factors Analysis – A New Alterna-tive to Knowledge. *the 14th International Conference on Artificial Intel-ligence in Education*, (pp. 531-538).

Perruchet, P. (1989). The effect of spaced practice on explicit and implicit memory. *British Journal of Psychology*, 80: 113–130.

Qiu, Y., Qi, Y., Lu, H., Pardos, Z., & Heffernan, N. (2011). Does Time Matter? Modeling the Effect of Time with Bayesian Knowledge Tracing. *the 4th International Conference on Educational Data Mining*, (pp. 139-148).

Qiu, Y., Qi, Y., Lu, H., Pardos, Z., & Heffernan, N. T. (2011). Does Time Matter? Modeling the Effect of Time with Bayesian Knowledge Tracing. *the 4th International Conference on Educational Data Mining*, (pp. 139-148).

Sabourin, J., Mott, B., & Lester, J. (2011). Modeling Learner Affect with Theoretically Grounded Dynamic Bayesian Networks. *The 4th International Conference on Affective Computing and Intelligent Interaction*, (pp. 286-295).

San Pedro, M. O., Baker, R. S., Bowers, A. J., & Heffernan, N. T. (2013). Predicting college enrollment from student interaction with an Intelligent Tutoring System in middle school. *The 6th International Conference on Educational Data Mining*, (pp. 177-184).

San Pedro, M. O., Baker, R. S., Gowda, S. M., & Heffernan, N. T. (2013). Towards an understanding of affect and knowledge from student interaction with an Intelligent Tutoring System. *The 16th International Conference on Artificial Intelligence and Education*, (pp. 41-50).

Shih, B., K., K., & S., R. (2010). Unsupervised Discovery of Student Strategies. *the 3rd International Conference on Educational Data Mining*, (pp. 201-210).

Shih, B., Kenneth, K., & Richard, S. (2010). Unsupervised Discovery of Student Strategies. *the 3rd International Conference on Educational Data Mining*, (pp. 201-210).

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

Shih, B., Koedinger, K. R., & Scheines, R. (2010). Discovery of Student Strategies using Hidden Markov Model Clustering. *The 3rd International Conference on Educational Data Mining.*

Song, F., Sarkozy, G. N., Trivedi, S., Wang, Y., & Heffernan, N. T. (2013). Applying Clustering to the Problem of Predicting Retention within an ITS: Comparing Regularity Clustering with Traditional Methods. *the TwentySixth International Florida Artificial Intelligence Research Society Conference*, (pp. 527-532).

Sternburg, R., & Grigorenko, E. (2002). Dynamic testing: The nature and measurement of learning potential. *Cambridge University Press*.

Tang, K. L. (1996). Polytomous item response theory (IRT) models and their applications in large-scale testing problems: Review of the literature. *Educational Testing Service Technical Report*.

Trivedi, S., Pardos, Z. A., & Heffernan, N. T. (2011). The Utility of Clustering in Prediction Tasks. *the 17th Conference on Knowledge Discovery and Data Mining.*

Wang, Y. B. (151-160). Class vs. Student in a Bayesian Network Student Model. *the 16th International Conference on Artificial Intelligence in Education*, (p. 2013).

Wang, Y., & Beck, J. (2012). Incorporating Factors Influencing Knowledge Retention into a Student Model. *the Educational Data Mining Conference*, (pp. 201-203).

Wang, Y., & Beck, J. E. (2012). Incorporating Factors Influencing Knowledge Retention into a Student Model. *the Educational Data Mining Conference*, (pp. 201-203).

Wang, Y., & Beck, J. E. (2013). Class vs. Student in a Bayesian Network Student Model. *the 16th International Conference on Artificial Intelligence in Education*, (pp. 151-160).

Wang, Y., & Heffernan, N. (2011). The "Assistance" Model: Leveraging How Many Hints and Attempts a Student Needs. *The 24th International FLAIRS Conference.*

Wang, Y., & Heffernan, N. (2012). The Student Skill Model. *11th International Conference on Intelligent Tutoring Systems*, (pp. 399-404).

Wang, Y., & Heffernan, N. (2013). A Comparison of Two Different Method to Individualize Students and Skills. *the 16th International Conference on Artificial Intelligence in Education*, (pp. 836-840).

Wang, Y., & Heffernan, N. (2013). Extending Knowledge Tracing to allow Partial Credit: Using Continuous versus Binary Nodes. *the 16th International Conference on Artificial Intelligence in Education*, (pp. 181-188).

Wang, Y., & Heffernan, N. (2014). The Effect of Automatic Reassessment and Relearning on Assessing Student Long-term Knowledge in Mathematics. *the 12th International Conference on Intelligent Tutoring Systems*, (pp. 490-495).

Wang, Y., & Heffernan, N. T. (2012). The Student Skill Model. *The 11th International Conference on Intelligent Tutoring Systems*, (pp. 399-404).

Wang, Y., & Heffernan, N. T. (2013). A Comparison of Two Different Method to Individualize Students and Skills. *the 16th International Conference on Artificial Intelligence in Education*, (pp. 836-840).

Wang, Y., & Heffernan, N. T. (2013). Extending Knowledge Tracing to allow Partial Credit: Using Continuous versus Binary Nodes. *the 16th International Conference on Artificial Intelligence in Education*, (pp. 181-188).

Wang, Y., & Heffernan, N. T. (2014). The Effect of Automatic Reassessment and Relearning on Assessing Student Long-term Knowledge in Mathematics. *the 12th International Conference on Intelligent Tutoring Systems*, (pp. 490-495).

Wang, Y., Heffernan, N. T., & Beck, J. E. (2010). Representing Student Performance with Partial Credit. *the 3rd International Conference on Educational Data Mining.*

Wang, Y., Heffernan, N. T., & Heffernan, C. (2015). Towards Better Affect Detectors: Effect of Missing Skills, Class Features and Common Wrong Answers. *the 5th International Learning Analytics & Knowledge Conference*, (pp. 31-35).

Wang, Y., Heffernan, N., & Beck, J. (2010). Representing Student Performance with Partial Credit. *the 3rd International Conference on Educational Data Mining.*

Wang, Y., Heffernan, N., & Heffernan, C. (2015). Towards Better Affect Detectors: Effect of Missing Skills, Class Features and Common Wrong Answers. *the 5th International Learning Analytics & Knowledge Conference*, (pp. 31-35).

Xiong, X., Beck, J. E., & Li, S. (2013). Class distinctions: Leveraging class-level features to predict student retention performance. *the Proceedings of Artificial Intel-ligence in Education.*

Xiong, X., Li, S., & Beck, J. (2013). Will You Get It Right Next Week: Predict Delayed Performance in Enhanced ITS Mastery Cycle. *FLAIRS Conference.*

Xiong, X., Li, S., & Beck, J. E. (2013). Will You Get It Right Next Week: Predict Delayed Performance in Enhanced ITS Mastery Cycle. *FLAIRS Conference.*

Zhu, L., Wang, Y., & Heffernan, N. T. (2014). The Sequence of Action Model: Leveraging the Sequence of Attempts and Hints. *BKT20y Workshop of Educational Data Mining.*