Workshop Approaching Twenty Years of Knowledge Tracing (BKT20y)

Knowledge Tracing is an extremely popular method for student modeling because of its capability to infer a student's dynamic knowledge state in real time as the student is observed solving a series of problems (Corbett & Anderson, 1995). After its introduction in 1995, many extensions to the original technique have been proposed to improve its predictive accuracy. Variants include: fitting model parameters to individuals rather than populations (e.g., Lee & Brunskill, 2012; Yudelson, Koediger, & Gordon, 2010), contextualizing model parameters based on past and current usage of an intelligent tutoring system (Baker, Corbett, & Aleven, 2008, Baker et al., 2010; GonzálezBrenes, 2014; Pardos et al., 2010) and on latent characteristics of students and problems (Khajah et al, 2014), clustering similar students and sharing parameters among them (Pardos et al, 2012), soft sharing of parameters via hierarchical Bayesian inference (Beck & Chang, 2007; Beck, 2007), and considering knowledge state as a continuous variable (SohlDickstein, 2013; Smith et al., 2004).

As we approach twenty years since the introduction of Knowledge Tracing, what lessons have we learned? This workshop's motivation is to open the floor for the discussion of the recent advances in Knowledge Tracing and student modeling in general, take stock of the promises and failures of current approaches, and work toward developing integrated approaches.

We gratefully acknowledge the following members of the workshop program committee:

Albert Corbett, Carnegie Mellon University
Neil Heffernan, Worcester Polytechic Institute
Zachary Pardos, University of California, Berkeley
Steve Ritter, Carnegie Learning, Inc.

The BKT20y workshop organizers
Michael Yudelson
José P. González-Brenes
Michael Mozer

Table of Contents BKT20y

Additionally, the workshop scheduling will include the full talk “EEG Helps Knowledge Tracing!” based on the following paper:

Xu, Y., K.-M. Chang, Y. Yuan, and J. Mostow. EEG Helps Knowledge Tracing! In Proceedings of the ITS2014 Workshop on Utilizing EEG Input in Intelligent Tutoring Systems. 2014: Honolulu, p. 43-48.

Choosing Sample Size for Knowledge Tracing Models *

Derrick Coetzee
University of California, Berkeley
dcoetzee@berkeley.edu

ABSTRACT

An important question in the practical application of Bayesian knowledge tracing models is determining how much data is needed to infer parameters accurately. If training data is inadequate, even a perfect inference algorithm will produce parameters with poor predictive power. In this work, we describe an empirical study using synthetic data that provides estimates of the accuracy of inferred parameters based on factors such as the number of students used to train the model, and the values of the underlying generating parameters. We find that the standard deviation of the error is roughly proportional to $1/\sqrt{n}$ where n is the sample size, and that model parameters near 0 and 1 are easier to learn accurately.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining

General Terms

Measurement,Theory.

Keywords

Educational data mining,knowledge tracing,sample size

1. INTRODUCTION

Simple Bayesian knowledge tracing models a student's observed responses to a sequence of items as a Markov process, with their knowledge state as a hidden underlying variable. If values are given for the four standard parameters, learning rate, prior, guess, and slip, the likelihood of a particular set of response sequences can be computed. Using standard search procedures like expectation maximization (EM), the parameter set giving the highest likelihood for a given set of sequences can be determined, provided that the procedure converges to the global maximum.

However, even if the procedure identifies the global maximum correctly and precisely, the resulting parameters may not reflect the actual parameters that generated the data; this is a *sampling error* effect. It's clearest with very small samples, such as samples of size 1, but exists with larger samples as well. Empirical studies with synthetic data generated from known parameters show that the inferred parameters for a given data set can differ substantially from the generating parameters, and this same issue would arise in real settings. An understanding of the magnitude of sampling error in a particular scenario can help to explain why the resulting model does or does not make effective predictions. Moreover, by providing a means to describe the distribution of possible generating parameter values, the uncertainty of calculations based on those parameters such as predictions can also be determined.

2. RELATED WORK

For simple problems, such as identifying the mean value of a parameter in a population, or the proportion of the population falling into a subgroup, there are simple and well-understood statistical approaches for determining sample size based on statistical power. Such analytic approaches are not immediately applicable to the problem of minimizing the HMM error function because of its complexity and high dimensionality.

Falakmasir et al [2] have noted that training time increases linearly with the size of the training set. Choosing an appropriate sample size for a certain desired level of accuracy can thus help to reduce training time, which is important both for research and in some real-time interactive tutor applications.

Nooraei et al [3] found that using only the 15 most recent data points from each student to train a knowledge tracing model yielded root mean-square error during prediction comparable to using the student's full history. For one data set, the most 5 recent items sufficed. Our study conversely does not vary the number of items per student, but instead varies the number of students and the four parameters generating the data. By allowing sample size to be reduced to meet a desired accuracy, our work offers an orthogonal method of further reducing training time.

De Sande [8] has suggested that as samples become larger, models with small parameter sets may no longer be rich enough to capture the sample's complexity. Thus our exclu-

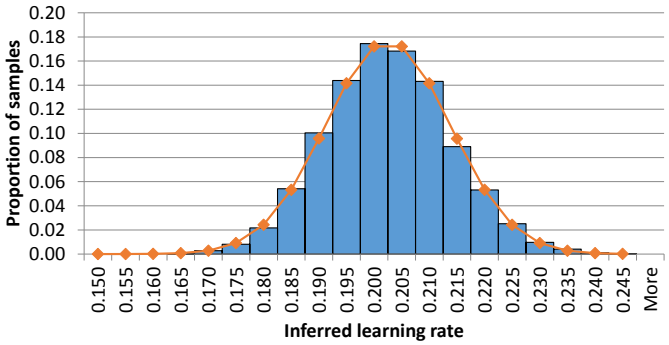


Figure 1: Given the fixed model learn=0.2, prior=0.4, guess=0.14, slip=0.05, we generated 10000 samples with 1000 students each, and for each, inferred all four parameters using EM. The distribution of the inferred learning rate parameter over the samples is above. The mean differs by 3×10^{-6} from the true generating parameter 0.2. The standard deviation is 0.01121, and the orange line shows the expected height of each bar if the proportions precisely followed a normal distribution. Scipy's normaltest [7] rejects that the distribution is perfectly normal ($p < 0.0002$), and a small amount of negative (left) skew is visible; the median is 0.00016 smaller than the mean. But the distribution is close enough to normal for our purposes.

sive reliance on a simple four-parameter BKT model even for very large samples is a limitation of our approach.

3. METHODOLOGY

In our experiments we relied on a simple standard Bayesian knowledge tracing model with four parameters: learning rate, prior, guess and slip. There is only one value for each parameter, and no specialization by student or problem. Each synthetic student responded to five items; we do not vary this parameter in this study, since Nooraei et al [3] report that increasing this parameter has diminishing returns, but future work may investigate it.

We generate separate datasets for each of our experiments. In each case, we enumerate a sequence of models (each specified by values for learn, prior, guess, slip, sample size), and for each of those models, we generate a large number of random samples consistent with that model. For example, for a particular model, we may generate 1000 samples each containing 1000 students.

We then run EM on each sample to find the parameter set giving the maximum likelihood value. All parameters are permitted to vary during the search. EM is run starting at the generating parameters and run until fully converged (within 10^{-12} or until 100 iterations are complete). Starting at the generating parameters is not feasible in a realistic setting, but here it allows EM to run quickly and consistently reach the global minimum. As shown in Figure 1, the parameter values inferred from these samples approximate a normal distribution with a mean equal to the generating parameter.

Finally, we take all samples generated from a single model and, for each parameter, record the mean and standard deviation of the inferred values for that parameter. We chose the number of samples generated for each model large enough so that these statistics remain stable under repeated runs. Mean values for each parameter were consistently near the generating parameter, typically within at most 0.1 standard deviations. Standard deviation provides an estimate of variation in the inferred parameter values, and is plotted. Different models yield different standard deviation values.

Because of the very large number of large samples involved in this approach, we use the *fastHMM* C++ BKT library designed by Pardos and Johnson [5] to quickly generate datasets and perform EM, invoked from a Matlab script.

3.1 Varying one parameter

In our first experiment, we start with typical, plausible values for all four parameters: learn=0.2, prior=0.4, guess=0.14, slip=0.05. These values are consistent with prior work that found large guess and slip values (> 0.5) to be implausible in most scenarios [6], and in our 5-problem scenario, the chance of learning the material by the end is about 67%, which is reasonable.

Then, for each of the four parameters, we hold the other parameters at their single plausible value, and vary the remaining parameter from 0 to 1 in steps of 0.01. This results in 404 total parameter sets.

For each parameter set, we generate 1000 random samples of 1000 students each. In this experiment, the number of students is fixed at 1000, which is large enough to consistently produce a standard deviation not exceeding 0.03 — this avoids the boundary effects near 0 and 1 that would occur for very small samples.

In this experiment, we focus on the variance of our estimates of the parameter that is being varied, and don't consider variance of the other (fixed) parameters.

3.2 Interactions between parameters

In this experiment, similiar to the first, we hold three parameters fixed (learn=0.2, prior=0.4, guess=0.14), and vary slip between 0 and 1 in steps of 0.01. This gives 101 parameter sets. For each, we generate 1000 random samples of 1000 students each. However, in this experiment we examine variance of our estimates of all four parameters, rather than just the one being varied (slip). This experiment helps to demonstrate to what extent varying one parameter can affect the difficulty of accurately inferring other parameters.

3.3 Varying sample size

In our third experiment, we fix the value of all four parameters, but vary the sample size in powers of two from 2 to 2097152. For sample sizes below 10000, we generate 1000 samples of that size, while for those above we generate 100 samples. The parameter values are heuristically chosen based on the prior experiments above to generate large error values (but not necessarily the worst possible error). We examine how variation of our estimates of all four parameters varies with sample size, and identify any trends.

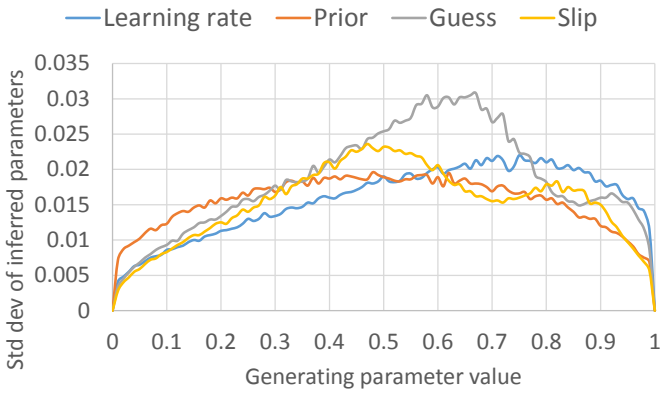


Figure 2: Variation of inferred parameters, based on underlying generating parameter. For each curve, all parameters other than one being examined are fixed at plausible values. Values near 0 and 1 are the easiest to infer accurately, and each parameter exhibits a unique pattern.

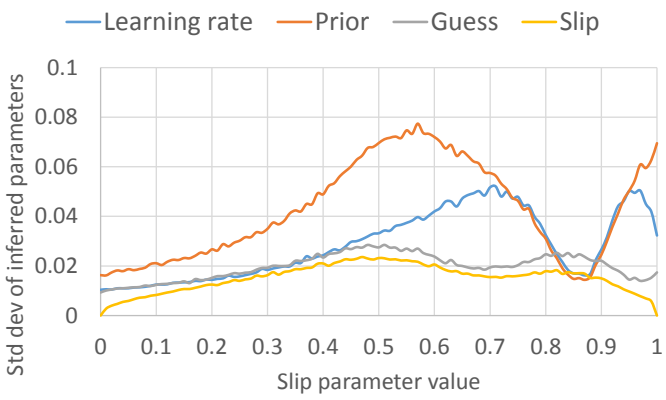


Figure 3: As the slip parameter is varied and the other parameters are held fixed (learn=0.2, prior=0.4, guess=0.14), the error in our inference of all other parameters varies in a strong and complex fashion, indicating interactions in the inference of different parameters.

3.4 Interaction between sample size and parameters

In our final experiment, we vary both the learning rate (from 0 to 1 in steps of 0.01) and the sample size (between the values 1000, 10000, 100000) at the same time. This enables us to examine whether there is any interaction between parameters and sample size. For 1000 and 10000 students we use 1000 samples, while for 100000 students we use 100 samples, to reduce runtime.

4. RESULTS

4.1 Varying one parameter

As described in section 3.1, in this experiment we vary each parameter between 0 and 1 while holding the other parameters fixed, and examined how the variation in our inference of that parameter changed with its value. As shown in Figure 2, parameters with values near 0 or 1 are easier to accurately estimate, while those with values in the 0.4 to 0.8 range are more difficult to infer. Each parameter exhibits a unique pattern, with prior behaving worst for small values, guess behaving worst for values in the middle, and learning rate performing worst for the largest values. Slip is unique in having two peaks in its curve near 0.5 and 0.8.

4.2 Interactions between parameters

As described in section 3.2, in this experiment we vary slip between 0 and 1 while keeping the other parameters fixed, and examine how the variation of all four inferred parameters varies, as shown in Figure 3. All variance values exhibit a strong, complex dependence on the slip parameter—in particular there is a dramatic and unexpected drop from large variance to small variance around slip=0.85. We conclude that the variance of an inferred parameter depends not only on the value of that parameter, but also the values of other parameters.

4.3 Varying sample size

We fix the parameters at the values empirically determined in section 4.1 to give maximum variance (roughly based on the maximums of the curves, with prior and guess at 0.5, and learning rate and slip at 0.67). Because section 4.2 suggests that there are interactions between parameters, this may not give the worst-case variance possible of all combinations, but it is a reasonable starting point for realistic values.

As described in section 3.3, sample size is varied in powers of two from 2 to 2097152. Figure 4 shows the result, suggesting that (except for very small samples) the standard deviation of the error is roughly proportional to $n^{-0.5}$, or $1/\sqrt{n}$, where n is the sample size. For these particular parameter values, slip is consistently inferred most accurately, learning rate is inferred least accurately, and guess and prior are between the two and are similar.

4.4 Interaction between sample size and parameters

In our final experiment, as described in section 3.4, we vary both the learning rate and the sample size at the same time. The standard deviation curves for the three sample sizes are then plotted on the same plot, each divided by the $1/\sqrt{n}$ factor, where n is the sample size, as shown in Figure 5. The curves are nearly identical, and we find no evidence of interaction between parameters and sample size, but we can't rule out interaction for other combinations of parameter values. This also offers additional evidence for the $1/\sqrt{n}$ trend from the previous section.

5. DISCUSSION

Because accuracy is good for parameter values near 0 and 1, this implies that for large enough samples, boundary effects (in which the distribution of error is skewed because values outside of the 0-1 range are not permitted) are not a serious concern.

Interactions between parameters are complex, suggesting that attempting to characterize error in each parameter independently is unlikely to yield good predictions of error. Moreover, attempts to model these interactions analytically

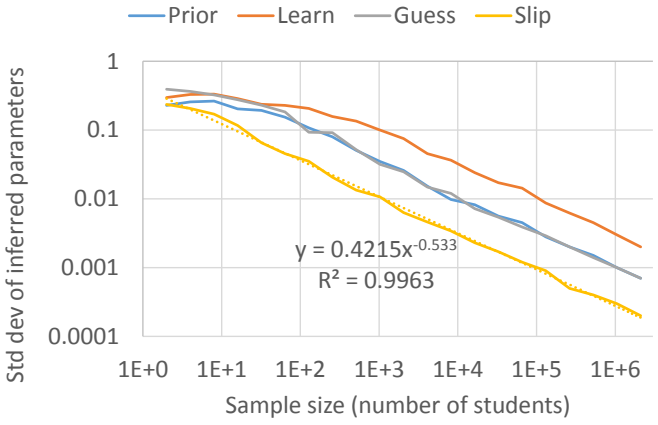


Figure 4: Accuracy of inferred parameters, based on sample size (training set size), with fixed parameters (prior=guess=0.5, learning=slip=0.67). This is a log-log plot, and (once the $y = 0.1$ level is reached) the lines each remain straight and have slope of roughly -0.5. This suggests that the standard deviation of the error is roughly proportional to $1/\sqrt{n}$, where n is the sample size.

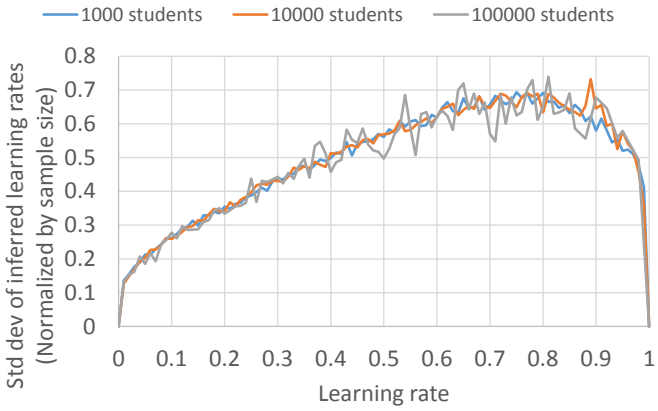


Figure 5: Here we vary learning rate from 0 to 1, and also vary sample size between the values 1000, 10000, and 100000. The resulting standard deviations are divided by $1/\sqrt{n}$ to normalize for improvement in error due to increased sample size. The resulting curves are nearly identical; the curve for 100000 students appears noisier only because of a lower number of samples (100 instead of 1000). We find no evidence of interaction between sample size and the learning rate.

may be challenging because they cannot be fit well by low-degree polynomials. A more viable strategy is to form a conservative estimate of error by conducting a grid search of parameter sets that are plausible in a given scenario. On the other hand, once the range of variances at a particular (sufficiently large) sample size is characterized, Figure 4 and Figure 5 show that altering the sample size has a uniform and predictable effect on the error.

The main result that standard deviation is proportional to $1/\sqrt{n}$ suggests that, in order to decrease the margin of error in the estimate of a parameter by a factor of 2, an increase in sample size by a factor of 4 is required. Additionally, Figure 4 shows that achieving even a single valid significant digit in the learning rate requires sample sizes of 1000 students or more. This suggests that studies using BKT with less than 1000 students should be considered carefully for sampling error.

5.1 Confidence Intervals and Decreasing Training Time

As noted in Figure 1, provided that the sample size is large enough, the distribution of samples is approximated well by a normal distribution, and the standard deviations computed in synthetic simulations such as the preceding ones can be used to compute confidence intervals containing the true generating parameters (e.g. 95% of possible values are within two standard deviations). Parameters used in these simulations can be set either by using domain knowledge, and/or by conservatively selecting values that give poor accuracy.

To use our results to decrease training time for a large data set, one approach is to create many small samples (e.g. 100 of size 1000) by sampling uniformly randomly with replacement from the full data set. By training on these, we can estimate the variance of our estimates of each parameter at a sample size of 1000. Then, given a desired level of accuracy and a desired probability of achieving it, we can use $1/\sqrt{n}$ to estimate the best final sample size. If the estimated sample size exceeds the data size, this suggests that more data needs to be gathered.

6. IDENTIFIABILITY PROBLEM

Although we have in this work considered a particular generating parameter set to be the correct and desired parameters, BKT exhibits an Identifiability Problem [1] in which there are an infinite family of four-parameter solutions that make the same predictions. This creates the risk that a solution that appears to be far from the generating parameters is actually very close to an equivalent parameter set (or an equivalent solution is).

Van de Sande [9] more specifically characterized BKT (in its HMM form) as a three-parameter system in which two systems having the same slip, learning rate, and A value will yield the same predictions, where A is given by

$$A = (1 - \text{slip} - \text{guess})(1 - \text{prior}).$$

One way to address the issue is to perform both data gener-

ation and parameter search in this reduced three-parameter system; this would be similar to our current approach, but error in the A parameter is more difficult to interpret. Intuitively, we expect search in a lower-dimensional space to give better accuracy with the same amount of data. However, Van de Sande also notes that the algorithm form of BKT has no analytic solution, and so the degree to which BKT is underdetermined may depend on the specific application.

Beyond the underdetermined nature of BKT, there are also information-theoretic bounds that limit the accuracy of inferring parameters regardless of the system. In particular, given a collection of at least k different parameter sets, and student data that can only take on $< k$ values, there is no procedure that can reliably infer the generating parameters without error. As the size of the data continues to decrease, the minimum possible error increases. Although these bounds are general, they typically apply only to very small data sets.

7. CONCLUSIONS AND FUTURE WORK

We've only explored a small part of the space of input parameters that can affect inferred parameter accuracy; the possible interactions between parameters are complex and not fully understood. It would also be useful to examine different sizes of problem sets, scenarios where different students complete different numbers of problems, models where parameters such as learning rate and guess/slip are per problem, and models where priors are measured per student (as in Pardos and Heffernan [4]).

Although it seems intuitive that insufficient sample size can lead to poor parameter estimates with poor predictive power, this deserves verification: it's not clear which errors will damage prediction and which are benign. An empirical synthetic study that examines prediction accuracy could assess this cheaply. Going a step further, it would be useful to simulate an interactive tutoring system and assess a cost function that penalizes the system for both incorrect assessment of mastery, and for failing to assess mastery when it is reached. By applying weights to these error types, the simulation could represent the real-world cost of inaccurate parameters in such a system.

Another important direction is extending our results to real-world data. There are a few approaches. One is to use a very large real-world data set and use its inferred parameters as the ground-truth generating parameters, then examine smaller subsets to determine whether parameters are inferred less accurately. If the BKT model is appropriate, we expect to observe similar relationships between sample size and variance as with our synthetic data. This approach can be compared to one experiment of Ritter [6] (Figure 4), in which they took a large real data set and computed mean-squared error using the best-fit parameters on subsets with smaller number of students ranging from 5 to 500.

There are other approaches to real-world validity. One would be a survey of prior BKT applications, to identify whether there is a consistent relationship between sample size and reported prediction accuracy. A third approach would be a controlled experiment in which two groups of very different sizes each use an ITS, the BKT is trained on the resulting data, and then the groups continue to use the ITS and their learning performance is examined (note however that asymmetric group sizes limit statistical power).

Finally, an analytical model that can explain some of our empirical results—such as the skewed normal distribution of inferred parameter values, the improvements in parameter inference near 0 and 1 parameter values, or the $1/\sqrt{n}$ relationship between sample size and standard deviation—would be a valuable contribution.

8. ACKNOWLEDGMENTS

We thank Zachary A. Pardos for his *fastHMM* C++ BKT library [5], for providing helpful comments on this work, and for designing the assignment which inspired it.

9. REFERENCES

[1] J. E. Beck and K.-M. Chang. Identifiability: A fundamental problem of student modeling. In *Proceedings of the 11th International Conference on User Modeling*, UM '07, pages 137–146, Berlin, Heidelberg, 2007. Springer-Verlag.

[2] M. H. Falakmasir, Z. A. Pardos, G. J. Gordon, and P. Brusilovsky. A spectral learning approach to knowledge tracing. In *6th International Conference on Educational Data Mining (EDM 2013).*, pages 28–35. International Educational Data Mining Society, 2013.

[3] B. B. Nooraei, Z. A. Pardos, N. T. Heffernan, and R. S. J. de Baker. Less is more: Improving the speed and prediction power of knowledge tracing by using less data. In M. Pechenizkiy, T. Calders, C. Conati, S. Ventura, C. Romero, and J. C. Stamper, editors, *EDM*, pages 101–110. www.educationaldatamining.org, 2011.

[4] Z. A. Pardos and N. T. Heffernan. Modeling individualization in a bayesian networks implementation of knowledge tracing. In P. D. Bra, A. Kobsa, and D. N. Chin, editors, *UMAP*, volume 6075 of *Lecture Notes in Computer Science*, pages 255–266. Springer, 2010.

[5] Z. A. Pardos and M. J. Johnson. Scaling cognitive modeling to massive open environments (in preparation). 2015.

[6] S. Ritter, T. K. Harris, T. Nixon, D. Dickison, R. C. Murray, and B. Towle. Reducing the knowledge tracing space. In T. Barnes, M. C. Desmarais, C. Romero, and S. Ventura, editors, *EDM*, pages 151–160. www.educationaldatamining.org, 2009.

[7] SciPy v0.13.0 Reference Guide: scipy.stats.normaltest. `http://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.normaltest.html`, May 2013. [Online; accessed 24-April-2014].

[8] B. Van de Sande. Applying three models of learning to individual student log data. In *6th International Conference on Educational Data Mining (EDM 2013).*, pages 193–199. International Educational Data Mining Society, 2013.

[9] B. Van de Sande. Properties of the bayesian knowledge tracing model. *Journal of Educational Data Mining*, 5(2):1–10, 2013.

A Unified 5-Dimensional Framework for Student Models

Yanbo Xu and Jack Mostow
Carnegie Mellon University Project LISTEN
RI-NSH 4103
5000 Forbes Ave, Pittsburgh, PA 15213
{yanbox, mostow}@cs.cmu.edu

ABSTRACT
This paper defines 5 key dimensions of student models: whether and how they model time, skill, noise, latent traits, and multiple influences on student performance. We use this framework to characterize and compare previous student models, analyze their relative accuracy, and propose novel models suggested by gaps in the multi-dimensional space. To illustrate the generative power of this framework, we derive one such model, called HOT-DINA (Higher Order Temporal, Deterministic Input, Noisy-And) and evaluate it on synthetic and real data. We show it predicts student performance better than previous methods, when, and why.

Keywords
Knowledge tracing, Item Response Theory, temporal models, higher order latent trait models, multiple subskills, DINA.

1. Introduction
Morphological analysis [1] is a general method for exploring a space of possible designs by identifying key attributes, specifying possible values for each attribute, and considering different combinations of choices for the attributes. Structuring the space in this manner compares different designs in terms of which attribute values they share, and which ones differ. Characterizing the space of existing designs in terms of these attributes exposes gaps in the space, suggesting novel combinations to explore.

Some prior work on student modeling has used this approach to characterize spaces of possible knowledge tracing models. Knowledge tracing (KT) [2] generally has 4 or 5 parameters: the probability *slip* of failing on a known skill; the probability *guess* of succeeding on an unknown skill; the probability *knew* of knowing a skill before practicing it; the transition probability *learn* from not knowing the skill to knowing it; and sometimes the transition probability *forget* from knowing the skill to not knowing it, usually assumed to be zero.

Mostow et al. [3] defined a space of alternative parameterizations of a given KT model, based on whether they assigned each knowledge tracing parameter a single overall value, a distinct value for each individual student and/or skill, or different values for different categories of students and/or skills. Thus the number of values to fit is 4 if using a single global value for each parameter, but with separate probabilities for each <student, skill> pair, the number of values to fit is 4 × # students × # skills. This work ordered the space of possible parameterizations of a single model by the number of values to fit.

Xu and Mostow [4] factored the space of different knowledge tracing models in terms of three attributes: how to ***fit*** their parameters, how to ***predict*** students' performance from their estimated knowledge, and how to ***update*** those estimates based on observed performance. We will use this factoring in Section 3.2.

Section 2 introduces the proposed framework. Section 0 describes HOT-DINA, a novel knowledge tracing method that the framework inspired. Sections 4 and 5 evaluate HOT-DINA on synthetic and real data, respectively. Section 6 concludes.

2. A Unified 5-Dimensional Framework
We characterize student models in terms of these five dimensions:

Temporal effect: skills time-invariant vs. time-varying.
- Static, e.g. IRT [5] and PFA [6]
- 2 or more fixed time points, e.g. at pre- and post-test
- Dynamic, e.g. KT [2]

Skill dimensionality: single skill vs. multiple skills at a step.

Credit assignment: how credit (or blame) is allocated among influences on the observed success (or failure) of a step. Mostow et al. [3] define a space of KT parameterizations. Corbett and Andersen [2] originally fit KT per skill. Pardos and Heffernan [7] individualized KT and fit parameters per student. Wang and Heffernan [8] simultaneously fit KT per student and per skill. In contrast, multiple-skills models require combination functions to assign credit or blame among the skills. Product KT [9] assigns full responsibility to each skill and multiplies the estimates. Conjunctive KT [10] assigns fair credit or blame to skills and multiplies the estimates. Weakest KT [11] credits or blames the weakest skill and takes the minimum of the estimates. LR-DBN [12] apportions credit or blame and performs logistic regression over the estimates. We summarize credit assignment methods as:
- Contingency table
  - Per student
  - Per skill
  - Per <student, skill>
  - Per student + per skill
- Binary or probabilistic
  - Conjunctive (min)
  - Independent (product)
  - Disjunctive (max)
- Other
  - Compensatory (+)
  - Mixture (weighted average)
  - Logistic regression (sigmoid)

Higher order: treat static student properties as latent traits or not. We say IRT [5] models "higher order" effects because it estimates static student proficiencies independent of skill properties such as skill difficulty in 1PL (1 Parameter Logistic), skill discrimination in 2PL, and skill guess rate in 3PL. De la Torre [13] first combined IRT with static Cognitive Diagnosis Models such as

Published in CEUR-WS:
BKT20y workshop (Yudelson, González-Brenes and Mozer)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

NIDA (Noisy Inputs, Deterministic And Gate) [14-16] and DINA (Deterministic Inputs, Noisy And Gate), and proposed higher order latent trait models (HO-NIDA and HO-DINA). Xu and Mostow [17] used IRT to estimate the probability of knowing a skill initially in a higher order knowledge tracing model (HO-KT).

Noise: how to represent errors in model, or discrepancies between what a student knows versus does. KT assumes students may guess a step correctly even though they don't know its underlying skill(s), or slip at a step even though they know its skill(s). Such "noise" is also characterized in other models, including single-skill KT variants such as PPS (Prior Per Student) [7] and SSM (Student Skill Model) [8], and IRT models such as 3PL. NIDO and DINO respectively add noise either before or after combining estimates of multiple skills. We refer to these noise modeling methods as:

- None
- Slip/Guess
- NIDO (noisy input, deterministic output)
- DINO (deterministic input, noisy output)

Table 1 summarizes student models in the proposed unified 5-dimensional framework. Note that we only discuss known cognitive models (e.g. Q-matrix) in this paper, so we omit methods that discover unknown cognitive models [18, 19].

Table 1. A unified 5-dimensional framework for student models

Student models	Temporal effect	Skill dimensionality	Credit assignment	Higher order effect	Noise model
IRT 1PL (Rasch model) [5] IRT 2PL (2 Parameter Logistic) [5]	Static	Single skill	Per student + per skill	Latent trait	None
IRT 3PL (3 Parameter Logistic) [5]					Slip/Guess
LLM (Linear Logistic Model) [16]		Multiple skills	Sigmoid	No latent trait	None
LFA (Learning Factor Analysis) [20] PFA (Performance Factor Analysis) [6]					
NIDA [14-16]			Product		NIDO
DINA [14-16]					DINA
LLTM (Linear Logistic Test Model) [21]			Sigmoid	Latent trait	None
HO-NIDA [13]			Product		NIDO
HO-DINA [13]					DINO
KT [2]	Dynamic	Single skill	Per skill	No latent trait	Slip/Guess
PPS (Prior Per Student) [7]			Per student		
SSM (Student Skill Model) [8]			Per student + Per skill		
HO-KT [17]				Latent trait	
DIR (Dynamic IRT 1PL) [22]					None
KT+NIDA [23]		Multiple skills	Product	No latent trait	NIDO
Product KT [9]					
CKT [10]					
Weakest KT [11]			Minimum		
KT+DINA [23]			Product		DINO
LR-DBN [12]			Sigmoid		
HOT-NIDA [Section 0]			Product	Latent trait	NIDO
HOT-DINA [Section 0]					DINO

Table 2. Comparative framework to train, predict and update multiple-skills models

Student models	Train	Predict	Update
CKT	Train skills separately. Assign each skill full responsibility.	Multiply skill estimates.	Update skills together. Bayes' equations assign responsibility.
Product KT			Update skills separately, each with full responsibility.
Weakest KT (Blame weakest, credit rest)		Minimum of skill estimates.	
Weakest KT (Update weakest skill)			Update only the weakest skill.
HOT-NIDA HOT-DINA [Section 3.2]	Train skills together. Assign each skill full responsibility.	Multiply skill estimates.	
KT+NIDA/DINA			Update skills together, each with full responsibility.
LR-DBN	Train skills together. Logistic regression assigns responsibility.	Logistic regression on skill estimates.	Update skills together. Logistic regression assigns responsibility.

Table 2 (adapted from [4]) expands **Credit assignment** in terms of how to **train**, **predict** and **update** skills, e.g. to assign full responsibility to every skill, blame the weakest skill and credit the rest, update only the weakest skill, or use logistic function.

The tables suggest transformations of models along the dimensions in the framework. For example, Dynamic IRT [22] varies student proficiency by time, transforming static IRT to dynamic. KT+NIDA/DINA [23] varies skill estimates by time, transforming static NIDA/DINA to dynamic. HO-NIDA/DINA/KT adds latent traits, transforming NIDA/DINA/KT to higher order. LLM [16] and LLTM [21] change the combination function, transforming conjunctive models to logistic models. In Section 0 we generate a novel student model by transforming HO-KT to a multi-skill model.

3. A Higher-Order Temporal Student Model to Trace Multiple Skills: HOT-DINA

Xu and Mostow [17] extended the static IRT model into HO-KT (Higher Order Knowledge Tracing), which accounts for skill-specific learning by using the static IRT model to estimate the probability Pr(*knew*) of knowing a skill before practicing it. By generalizing to steps that require conjunctions of multiple skills, we arrive at a combined model we call HOT-DINA (Higher Order Temporal, Deterministic Input, Noisy-And). Note we can transform it into HOT-NIDA simply by changing its noise type.

3.1 HOT-DINA = IRT + KT + DINA

Let $\{\mathbf{Y}^{(0)}, \mathbf{Y}^{(1)}, \ldots, \mathbf{Y}^{(t)}, \ldots\}$ denote a sequential dataset recorded by an intelligent tutor system, where $Y_{nj}^{(t)} = 1$ iff student n correctly performs a step that requires skill j at time t. KT is a Hidden Markov Model (HMM) that models a binary hidden state $\mathbf{K}^{(t)}$ indicating if the student knows the skill at time t. The probability of knowing the skill is *knew* at time $t = 0$, and then changes based on the student's observed performance on the skill, according to the standard KT parameters *slip*, *guess*, *learn*, and *forget* (usually set to zero).

KT can fit these four parameters (taking *forget* = 0) for each <student, skill> pair, but the resulting large number of values to fit is likely to cause over-fitting. Thus, Corbett and Andersen [2] originally proposed to estimate *knew* per student, and *learn, guess* and *slip* per skill. IRT assumes a latent trait that represents a student's underlying proficiency in all the skills. For example, the Two Parameters Logistic (2PL) IRT model assumes that the probability of a student's correct response is a logistic function of a unidimensional student proficiency θ with two skill-specific parameters: discriminability a and difficulty b (see Equation 1).

$$P(Y = 1) = \frac{1}{1 + \exp(-1.7a(\theta - b))}$$

Equation 1. The logistic function of 2PL model

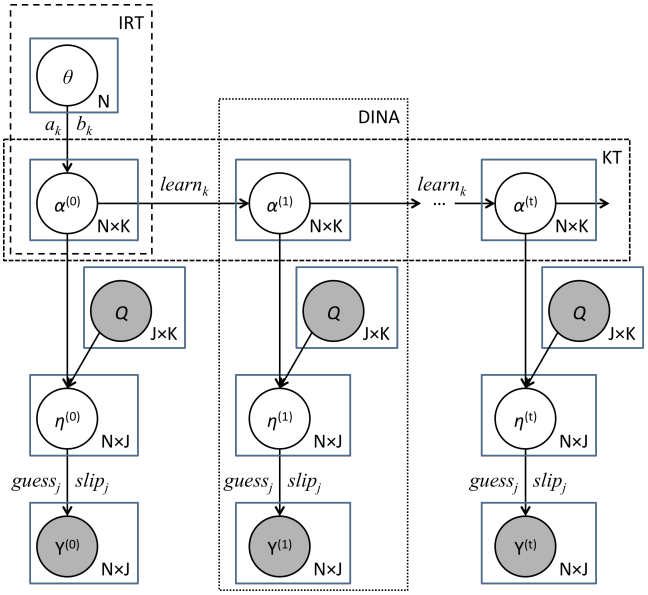
The two skill parameters determine the shape of the IRT curve. As a student's proficiency increases beyond the skill difficulty, the student's chance of performing correctly surpasses 50%. The skill discriminability reflects how fast the logit (log odds) increase or decrease when the proficiency changes. Thus IRT fits parameters individually on each dimension, without losing the information from the other. HO-KT uses 2PL to estimate *knew* in KT, by fitting student specific proficiency θ_n, skill discriminability a_j and skill difficulty b_j. It then uses KT to trace each skill, by fitting skill-specific $learn_j$, $guess_j$ and $slip_j$. Thus, HO-KT models students' initial overall knowledge before they practice any skills; then it updates its estimates of students' knowledge of each individual skill by observing additional practice on the skill. It also models two attributes of the skills, difficulty and discriminability, which are assumed to be constants that do not change over time.

To incorporate DINA into HO-KT, we still model a hidden binary state in each step to indicate whether a student knows the overall skill used in the step, denoted as $\eta_{nj}^{(t)}$ for student n with skill j at time t. However, we also model a hidden binary state $\alpha_{nk}^{(t)}$ to indicate whether student n knows skill k at time t. Given a matrix $\mathbf{Q} = \{Q_{jk}\}$, indicating whether the overall skill j requires skill k, we conjoin the skills as follows:

$$\eta_{nj}^{(t)} = \prod_{k=1}^{K} (\alpha_{nk}^{(t)})^{q_{jk}}$$

Equation 2. Conjunction of skills in HOT-DINA

This formula gives us the DINA (Deterministic Input, Noisy-And gate) structure [15], with the conjunction as the "and" gate and *guess* and *slip* as the noise. Thus by combining HO-KT with DINA, we obtain the HOT-DINA higher order temporal model to trace multiple skills. Figure 1 shows how the plate diagram for HOT-DINA integrates IRT, KT, and DINA.



θ_n: Proficiency of student n.

a_k: Discrimination of subskill k.

b_k: Difficulty of subskill k.

Q_{jk}: 1 if step j requires subskill k;
0 otherwise.

$\alpha_{nk}^{(t)}$: 1 if student n knows subskill k at time t;
0 otherwise.

$\eta_{nj}^{(t)}$: 1 if student n knows the skill of step j at time t;
0 otherwise.

Figure 1. Graphical representation of Higher-Order Temporal DINA (HOT-DINA) to trace multiple skills

Equation 3 shows the formula for using 2PL to estimate the probability *knew* of a student knowing a skill at time *t = 0*:

$$P(knew_{nk}) = P\left(\alpha_{nk}^{(0)} = 1\right) = \frac{1}{1 + \exp\left(-1.7\, a_k(\theta_n - b_k)\right)}$$

Equation 3. 2PL to estimate *knew* in HOT-DINA

Equation 4 shows the formula for tracing the skills with skill-specific *learn* and zero *forget*:

$$P\left({\alpha_{nk}}^{(t)} = 1 \middle| {\alpha_{nk}}^{(t-1)} = 0\right) = learn_k$$

$$P\left({\alpha_{nk}}^{(t)} = 0 \middle| {\alpha_{nk}}^{(t-1)} = 1\right) = forget_k = 0$$

Equation 4. Knowledge tracing of skills in HOT-DINA

Equation 5 shows the likelihood of a student's performance given the hidden state $\eta^{(t)}$ and the skill-specific *guess* and *slip*:

$$L\left(Y_{nj}^{(t)} = 1 \middle| \eta_{nj}^{(t)}\right) = {guess_j}^{\left(1-\eta_{nj}^{(t)}\right)} \times (1 - slip_j)^{\eta_{nj}^{(t)}}$$

$$L\left(Y_{nj}^{(t)} = 0 \middle| \eta_{nj}^{(t)}\right) = (1 - guess_j)^{\left(1-\eta_{nj}^{(t)}\right)} \times {slip_j}^{\eta_{nj}^{(t)}}$$

Equation 5. Likelihood in HOT-DINA

3.2 How to Train, Predict, and Update

Following the organization of Table 2, Section 3.2.1 details how HOT-DINA trains the skills together and assigns each skill full responsibility; Section 3.2.2 specifies how HOT-DINA predicts student performance by using a product of skill estimates; and Section 3.2.3 shows how HOT-DINA updates the weakest skill.

3.2.1 Training the model with MCMC

We estimate the parameters of HOT-DINA using Markov Chain Monte Carlo (MCMC) methods, which require that we specify the prior distributions and constraints for every parameter. We assume that student general proficiency θ_n is normally distributed with mean 0 and standard deviation 1. The skill discrimination a_n is positive and uniformly distributed between 0 and 2.5, while the skill difficulty b_n is also normally distributed with mean 0 and standard deviation 1. *Learn* has prior Beta (1,1), whereas *guess* and *slip* have uniform prior from 0 to 0.4.

Thus, the priors on each parameter are:

$\theta_n \sim Normal(0,1)$

$b_k \sim Normal(0,1)$

$a_k \sim Uniform(0,2.5)$

$learn_k \sim Beta(1,1)$

$guess_j \sim Uniform(0,0.4)$

$slip_j \sim Uniform(0,0.4)$

We use the following conditional distributions for each node:

$\alpha_{nk}^{(0)} | \theta_n \sim Bernoulli(\{1 + \exp(-1.7\, a_k(\theta_n - b_k))\}^{-1})$

${\alpha_{nk}}^{(t)} |\, {\alpha_{nk}}^{(t-1)} = 0 \sim Bernoulli(learn_k)$

${\alpha_{nk}}^{(t)} |\, {\alpha_{nk}}^{(t-1)} = 1 \sim Bernoulli(1)$

${Y_{nj}}^{(t)} | {\eta_{nj}}^{(t)} = 0 \sim Bernoulli(guess_j)$

${Y_{nj}}^{(t)} | {\eta_{nj}}^{(t)} = 1 \sim Bernoulli(1 - slip_j)$

Given $\boldsymbol{\eta}$ as a conjunction of $\boldsymbol{\alpha}$, the likelihood of **Y** given $\boldsymbol{\eta}$, the conditional independence of $\boldsymbol{\alpha}^{(0)}$ given $\boldsymbol{\theta}$, and of $\boldsymbol{\alpha}^{(t)}$ given $\boldsymbol{\alpha}^{(t-1)}$, the posterior distribution of ***θ, a, b, a, η, learn (l), guess (g)*** and ***slip(s)*** given **Y** is

$$P(\boldsymbol{\theta}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\alpha}, \boldsymbol{\eta}, \boldsymbol{l}, \boldsymbol{g}, \boldsymbol{s} | \boldsymbol{Y}) \propto L(\boldsymbol{Y} | \boldsymbol{g}, \boldsymbol{s}, \boldsymbol{\eta}, \boldsymbol{\alpha}) P\left(\boldsymbol{\alpha}^{(0)} \middle| \boldsymbol{\theta}, \boldsymbol{a}, \boldsymbol{b}\right)$$

$$\left(\prod_{t=1}^{T} P\left(\boldsymbol{\alpha}^{(t)} \middle| \boldsymbol{\alpha}^{(t-1)}, \boldsymbol{l}\right)\right) P(\boldsymbol{\theta}) P(\boldsymbol{a}) P(\boldsymbol{b}) P(\boldsymbol{l}) P(\boldsymbol{g}) P(\boldsymbol{s})$$

3.2.2 Predicting student performance

For inference, we introduce uncertainty to η_{nj}, and rewrite the Equation 2 as follows:

$$P\left(\eta_{nj}^{(0)} = 1\right) = \prod_{k=1}^{K} \left(\frac{1}{\exp\left(-1.7 a_k(\theta_n - b_k)\right)}\right)^{q_{jk}}$$

$P\left(\eta_{nj}^{(t)} = 1\right) = \prod_{k=1}^{K} (P(\alpha_{nk}^{(t)} = 1))^{q_{jk}})$ for t = 1,2,3…

Equation 6. Conjunction of skills in HOT-DINA inference

Then we predict student performance by using Equation 7:

$$P\left(Y_{nj}^{(t)} = 1\right) = (1 - slip_j) P\left(\eta_{nj}^{(t)} = 1\right) + guess_j (1 - P\left(\eta_{nj}^{(t)} = 1\right))$$

Equation 7. Prediction in HOT-DINA

3.2.3 Updating estimated skills

We update the estimates of latent states $\boldsymbol{\eta}$ and $\boldsymbol{\alpha}$ after observing actual student performance. The estimate of knowing a skill or a subskill should increase if the student performed correctly at the step. It is easy to update a skill by using Bayes' rule, as shown in Equation 8. The posterior $P(\eta_{nj}^{(t)} = 1 | Y_{nj}^{(t)} = 1)$ should be higher than $P(\eta_{nj}^{(t)} = 1)$ if and only if $(1\text{-}slip_j) > guess_j$.

$$P\left(\eta_{nj}^{(t)} = 1 \middle| Y_{nj}^{(t)} = 1\right) = \frac{P\left(Y_{nj}^{(t)} = 1 \middle| \eta_{nj}^{(t)} = 1\right) P\left(\eta_{nj}^{(t)} = 1\right)}{P\left(Y_{nj}^{(t)} = 1\right)}$$

$$= \frac{(1 - slip_j)\, P\left(\eta_{nj}^{(t)} = 1\right)}{(1 - slip_j)\, P\left(\eta_{nj}^{(t)} = 1\right) + guess_j \left(1 - P\left(\eta_{nj}^{(t)} = 1\right)\right)}$$

Equation 8. Bayes' rule to update η in HOT-DINA

Although we could update HOT-DINA by assigning full responsibility to each skill, it would be interesting to update the weakest (or say hardest) skill since HOT-DINA fits the parameter 'difficulty' for each skill. Thus, we update the skill that is the hardest among all the required skills in a step:

$$P\left(\eta_{nj}^{(t)} = 1 \middle| Y_{nj}^{(t)} = 1\right) = P\left(\alpha_{nk'}^{(t)} = 1 | Y_{nj}^{(t)} = 1\right) \prod_{k \neq k'} P(\alpha_{nk}^{(t)} = 1)$$

for $k = \arg\max_{k:\, Q_{jk} = 1} b_k$.

Equation 9. Update the hardest skill in HOT-DINA

In short, we extend HO-KT to the HOT-DINA higher order temporal model, which traces multiple skills. We use the MCMC algorithm to estimate the parameters, and update the estimates of a student knowing a skill given observed student performance. How well does the HOT-DINA model work? To

evaluate it, we performed a simulation study. Section 4 now describes the study and reports its results.

4. Simulation Study

To study the behavior of HOT-DINA, we generated synthetic training data for it according to the priors and conditional distributions defined in Section 3.2.1. Section 4.1 describes the synthetic data. One purpose of this experiment was to test how accurately MCMC can recover the parameters of HOT-DINA, as Section 4.2 reports. It is important not only to test how well a method works, but to analyze when and why. Thus another purpose was to determine how many students and observations are needed to estimate the difficulty and discriminability of a given number of skills, as Section 4.3 explains.

4.1 Synthetic Data

We use the following procedure to generate the synthetic data, with all the variables as defined in Section 3.2:

1. We chose K = 4 and J = 14, which results in a 14 × 4 **Q** matrix. The **Q** matrix, as shown below, indicates that we generate the skills by combining all the possible skills.

$$\mathbf{Q}^{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

2. We randomly generated θ_n from *Normal (0,1)* for *n = 1,..,N.*

3. We chose ***a***, ***b*** and ***l*** as shown in Table 3.

Table 3. True value of skill-specific discrimination, difficulty and learning rate in synthetic data simulation

k	1	2	3	4
a	1.5	1.2	1.9	1.0
b	-0.95	1.42	-0.66	0.50
learn	0.8	0.6	0.5	0.3

4. We randomly generated ***g*** and ***1-s*** from *Unif(0,0.4)* and *Unif (0.6,1)* respectively, as shown in Table 4.

Table 4. True value of skill-specific guess and not slip parameters in synthetic data simulation

j	1	2	3	4	5	6	7
guess	0.35	0.40	0.13	0.15	0.29	0.39	0.10
1-slip	0.67	0.66	0.67	0.90	0.65	0.60	0.61
j	8	9	10	11	12	13	14
guess	0.40	0.15	0.16	0.38	0.11	0.26	0.35
1-slip	0.81	0.74	0.76	0.73	0.83	0.89	0.85

5. We chose N = 100, T = 100, randomly picked one skill at each step, and simulated sequential data with size of 10,000.

4.2 Results

We used OpenBUGS [24] to implement the MCMC algorithm of HOT-DINA. We chose 5 chains starting at different initial points. We monitored the estimates of skill discrimination $\widehat{\boldsymbol{a}}$ and difficulty $\widehat{\boldsymbol{b}}$ to check their convergence, when all the chains appear to be overlapping each other. As a result, we ran the simulation for 10,000 iterations with a burn-in of 3000.

Table 5 reports the sample means and their 95% confidence interval for parameter estimates $\widehat{\boldsymbol{a}}$, $\widehat{\boldsymbol{b}}$ and $\widehat{\boldsymbol{learn}}$ respectively. We also report the Monte Carlo error (MC error) and sample standard deviation (s.d.) to assess the accuracy of the posterior estimates for each parameter. MC error, which is an estimate of the difference between the estimated posterior mean (i.e. the sample mean) and the true posterior mean, should be less than 5% of the s.d. in order to obtain an accurate posterior estimate.

Table 5. Estimates of skill-specific discrimination, difficulty, and learning rate (N = 100, T = 100, K = 4, J = 14)

k	***a***	$\widehat{\boldsymbol{a}}$ (95% C.I.)	s.d.	MC_error
1	1.50	1.33 (0.36, 2.43)	0.65	0.03216
2	1.20	1.23 (0.12, 2.43)	0.72	0.03561
3	1.90	1.85 (0.22, 2.73)	0.64	0.03146
4	1.00	0.98 (0.19, 2.12)	0.58	0.02870
k	***b***	$\widehat{\boldsymbol{b}}$ (95% C.I.)	s.d.	MC_error
1	-0.95	-0.95 (-2.15, -0.04)	0.50	0.02339
2	1.42	1.51(0.90, 2.21)	0.45	0.01936
3	-0.66	-0.69 (-1.81, -0.63)	0.42	0.01990
4	0.5	0.5 (0.05,1.18)	0.38	0.01691
k	***learn***	$\widehat{\boldsymbol{learn}}$ (95% C.I.)	s.d.	MC_error
1	0.8	0.81 (0.48, 0.99)	0.13	0.006599
2	0.6	0.60 (0.52, 0.70)	0.05	0.002132
3	0.5	0.57 (0.38, 0.84)	0.11	0.005432
4	0.3	0.29 (0.25, 0.33)	0.02	7.79E-04

We calculated Root Mean Squared Error (RMSE) of the estimates of the continuous variables $\widehat{\boldsymbol{guess}}$, $\widehat{\boldsymbol{1-slip}}$, and $\widehat{\boldsymbol{\theta}}$. We report the accuracy of recovering the true value of the latent binary variable $\boldsymbol{\alpha}$ in Table 6.

Table 6. Estimation RMSE of skill-specific guess, not slip, and student specific proficiency; Prediction accuracy of a student mastering a subskill (N = 100, T = 100, K = 4, J = 14)

	$\widehat{\boldsymbol{guess}}$	$\widehat{\boldsymbol{1-slip}}$	$\widehat{\boldsymbol{\theta}}$
RMSE	0.0103	0.0196	0.9183
	$\widehat{\boldsymbol{\alpha}}$		
Accuracy	99.38%		

From the results, we can see that the MCMC algorithm accurately recovered the parameters we used in generating the synthetic data for HOT-DINA. In addition to seeing how accurately it can estimate the parameters, we are also interested in finding out how many observations would be sufficient for the training algorithm to recover the hidden variables. Therefore, we conducted the study we now describe in Section 4.3.

4.3 Study Design

HOT-DINA requires data from enough students to rate the difficulty and discriminability of each skill, and data on enough skills to estimate the proficiency of each student. So we fixed the number of skills at K = 4, and varied the number of students N or the number of steps observed from each student T, to discover how many observations would be sufficient to estimate the parameters. In particular, we evaluated each model on how accurately it estimated the latent binary state $\boldsymbol{\alpha}$, which indicates if a student masters a skill. We generated the data by using the same parameters as in Section 4.1. Besides the general HOT-DINA model that accounts for multiple skills, we also studied the single-skill model by shrinking the number of skills J to equal K, and set **Q** as an identity matrix. Thus we specified the HOT-DINA model to be a HO-KT model alternatively.

We increased N, the number of students, from 10 to 1000, and T, the number of observations per student, from 5 to 100. Table

7 and Table 8 respectively show the accuracy of estimating the latent state α in HO-KT and HOT-DINA. Both tables show a trend of increasing accuracy when N or T increases (though at the cost of longer training time, roughly $O(N^2 \times T)$).

Table 7. Accuracy of estimating the latent binary states α with different N and T (K = J = 4)

N \ T	5	10	20	50	100
10	71.01%	80.81%	83.01%	93.11%	96.16%
20	72.32%	82.74%	86.52%	94.06%	97.33%
50	73.58%	83.79%	87.34%	95.27%	98.90%
100	77.55%	84.43%	88.08%	95.81%	**99.41%**
200	76.52%	84.02%	89.48%	97.26%	NA
500	78.13%	84.34%	92.50%	NA	NA
1000	80.10%	84.59%	NA	NA	NA

Due to the lack of sampling ability of OpenBUGS for high dimensional dynamic models, we have no available scores to show for N×T bigger than 10,000. We can see that the multiple skill model predicts better than the single-skill model because the average number of observations per skill in the former one is larger than the latter. As observed in both tables, it is more efficient to increase T, than N, to get a better estimate. Both of the models reach the best prediction accuracy score (> 99%) when N = 100 and T = 100. In order to obtain an accuracy > 90% for K = 4 skills, the least amount of data we need for HO-KT is N = 10 with T ≈ 50 observations as shown in Table 7, for HOT-DINA is N = 10 with T > 20 observations, as shown in Table 8.

Table 8. Accuracy of estimating the latent binary states α with different N and T (K = 4, J = 14)

N \ T	5	10	20	50	100
10	72.07%	75.57%	91.14%	96.90%	98.10%
20	74.32%	83.60%	91.56%	97.46%	98.53%
50	76.55%	84.71%	92.62%	97.52%	98.98%
100	77.80%	86.82%	93.83%	97.67%	**99.82%**
200	79.92%	88.78%	94.26%	99.41%	NA
500	82.15%	89.95%	98.61%	NA	NA
1000	83.58%	92.34%	NA	NA	NA

Next we apply the proposed model to real data logged by an algebra tutor. We evaluate the model fit and compare it against two baselines.

5. Evaluation on Real Data

We apply HOT-DINA to a real dataset from the Algebra Cognitive Tutor® [25]. Because of limited time, we chose a subset of the data, by crossing out the "isolated" algebra tutor steps. An "isolated" step here means a step that requires one skill all its own. We grouped the remaining steps that require the same multiple skills into one skill, resulting in J = 15 distinct skills that require K = 12 subskills. Following the study design in Section 4.3, we randomly chose N = 50 students with T = 100 in order to obtain enough data for the MCMC estimation.

Table 9. Data split of the Algebra Tutor data: training on I and IV, and testing on II and III

	Skill group A	Skill group B
Student group A	I	II
Student group B	III	IV

We split the 50 students into two groups of 25, and split the 15 skills into two groups of 8 and 7. As shown in Table 9, we combine data from I (student-group-A practicing on skill-group-A) and IV (student-group-B practicing on skill-group-B) to obtain the training data. Accordingly, we combined the data from II and III to obtain the test data. As a benefit of the data split, we are able to test the models on unseen students for the same group of skills, and also test on the unseen skills for the same group of students.

We compared HOT-DINA with the conjunctive minimum KT model [11] since it showed the best prediction accuracy among all the previous KT based methods [4]. It fits KT parameters by blaming each skill that is required at a step, predicts student's performance by the weakest skill, and updates only the weakest skill. Accordingly, we updated the most difficult skill in HOT-DINA as discussed in Section 3.2.3. As two baseline models, we fit per-skill KT and per-student KT. Comparing HOT-DINA with these two baselines also allows us to discuss some more interesting research questions later in this section.

Table 10 and Table 11 respectively show the models' prediction accuracy and log-likelihood on the test data. We report the majority class because of the unbalanced data. HOT-DINA beat the two baselines in predicting the student performance, and also obtained the maximum log-likelihood on the test data. The per-student KT model obtained the worst scores on both measures. It predicted student performance almost as poorly as majority class because it misclassified almost all the data in the minority class.

Table 10. Comparison of prediction accuracy on real test data

	Overall Accuracy	Accuracy on Correct Steps	Accuracy on Incorrect Steps
HOT-DINA	**82.48%**	96.63%	27.27%
Per-skill KT	80.87%	94.02%	29.60%
Per-student KT	79.63%	99.74%	1.20%
Majority class	79.60%	100.00%	0.00%

Table 11. Comparison of log-likelihood on real test data

	Log-likelihood
HOT-DINA	**-2021.04**
Per-skill KT	-2075.67
Per-student KT	-2464.74

We are also interested in three other hypotheses comparing HOT-DINA with KT. We describe them, test them, and show the results as follows.

1. HOT-DINA should predict early steps more accurately than KT since its estimate of *knew* reflects both skill difficulty and student proficiency, not just one or the other. In fact HOT-DINA beat KT throughout, as Figure 2 shows.

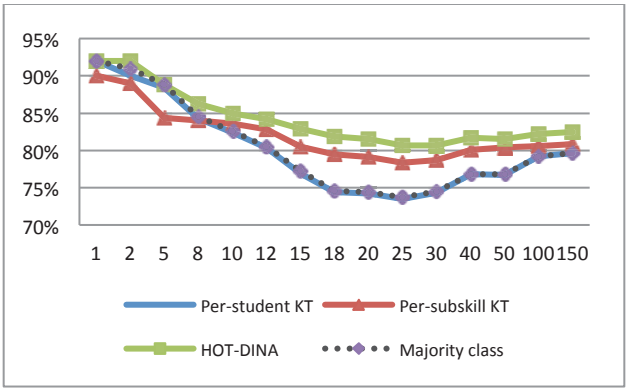


Figure 2. Accuracy on student's 1st, 2nd, 3rd, … test steps

2. HOT-DINA should beat KT on sparsely trained skills thanks to student proficiency estimates based on other skills. As Figure 3 shows, HOT-DINA tied or beat KT throughout.

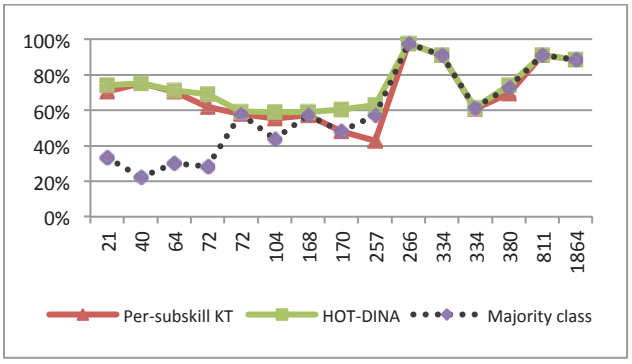


Figure 3. Skills sorted by amount of training data

3. HOT-DINA should beat KT on sparsely trained students thanks to skill difficulty and discriminability estimates based on other students. As Figure 4 shows, HOT-DINA beat KT throughout.

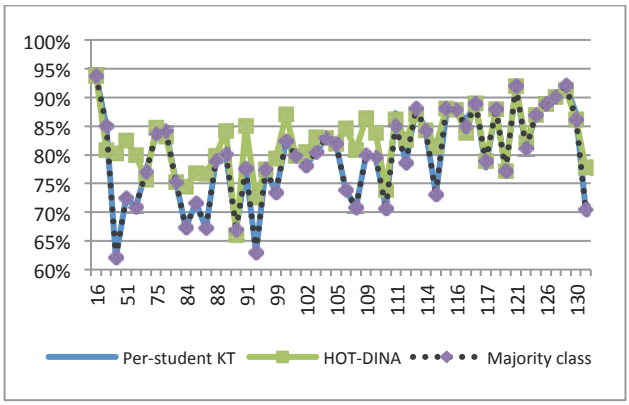


Figure 4. Students sorted by amount of training data

Thus, HOT-DINA outperformed the two baselines in model fit. It also beat them as specified by the three hypotheses above.

6. Contributions, limitations, future work

In this paper we make several contributions. We defined a 5-dimensional framework for student models. We showed how numerous student models fit into it. We described the new combination of IRT, KT, and DINA it suggests in the form of HOT-DINA. We specified how to train HOT-DINA by using MCMC, how to test it by predicting student performance, and how to update estimated skills based on observed performance.

HOT-DINA uses IRT to estimate *knew* based on student proficiency and skill difficulty. Thus it does not need training data on every <student, skill> pair, since it can estimate student proficiency based on other skills, and skill difficulty and discriminability based on other students. Likewise, it should estimate *knew* more accurately than KT for skills and students with sparse training data. HOT-DINA uses KT to model learning over time, and DINA to model combination of multiple skills underlying observed steps (unlike conventional KT and with fewer parameters than CKT [10] or LR-DBN [12]).

Tracing multiple skills underlying an observed step requires allocating responsibility among them for its success or failure. DINA simply conjoins them, a common method but inferior to others. Future work includes using the best known method [4], which we didn't use here because the logistic regression it performs is non-trivial to integrate with MCMC.

We evaluated HOT-DINA on synthetic and real data, not only showing that it predicts student performance better than previous methods, but analyzing when and why.

We reported a simulation study to test if training could recover model parameters, and to determine the amount of data needed. HOT-DINA requires data on enough students and skills to estimate their proficiency and difficulty, respectively. We explored how its accuracy varies with the number of test steps and the amount of training data per student and per skill. These analyses were correlational, based on variations that happened to occur in the training data. Future work should invest in the computation required to vary the amount of training data to establish its true causal effect on accuracy.

Evaluation on real data from an algebra tutor showed that HOT-DINA achieved higher predictive accuracy and log likelihood than KT with parameters fit per student or per skill. This evaluation was limited to a single data set and two baselines (not counting majority class). Future work should compare HOT-DINA to other methods – notably the Student Skill model [8], which is similar in spirit – and on data from other tutors.

We assumed that student proficiency is one-dimensional. Future work can test if k dimensions capture enough additional variance to make it worthwhile to fit k times as many parameters.

Finally, our choice of 5 dimensions is useful but limiting. Additional dimensions may provide useful finer-grained insights into the models covered by the current framework, and expand it to encompass other types of student models, e.g. where the cognitive model is unknown and must be discovered [18, 19].

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation through Grants 1124240 and 1121873 to Carnegie Mellon University. The opinions expressed are those of the authors and do not necessarily represent the views of the National Science Foundation or U.S. government. We thank Ken Koedinger for his algebra tutor data.

REFERENCES

[1] Zwicky, F. *Discovery, Invention, Research - Through the Morphological Approach*. 1969, Toronto: The Macmillian Company.

[2] Corbett, A. and J. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 1995. *4*: p. 253-278.

[3] Mostow, J., Y. Xu, and M. Munna. Desperately Seeking Subscripts: Towards Automated Model Parameterization. *Proceedings of the 4th International Conference on Educational Data Mining*, 283-287. 2011. Eindhoven, Netherlands.

[4] Xu, Y. and J. Mostow. Comparison of methods to trace multiple subskills: Is LR-DBN best? [Best Student Paper Award]. *Proceedings of the Fifth International Conference on Educational Data Mining*, 41-48. 2012. Chania, Crete, Greece.

[5] Hambleton, R.K., H. Swaminathan, and H.J. Rogers. *Fundamentals of Item Response Theory*. Measurement Methods for the Social Science. 1991, Newbury Park, CA: Sage Press.

[6] Pavlik Jr., P.I., H. Cen, and K.R. Koedinger. Performance factors analysis - a new alternative to knowledge tracing. *Proceedings of the 14th International Conference on Artificial Intelligence in Education (AIED09)*, 531-538. 2009.

[7] Pardos, Z. and N. Heffernan. Modeling individualization in a Bayesian networks implementation of knowledge tracing. *Proceedings of the 18th International Conference on User Modeling, Adaptation and Personalization*, 255-266. 2010. Big Island, Hawaii.

[8] Wang, Y. and N.T. Heffernan. The student skill model. *Intelligent Tutoring Systems - 11th International Conference*, 399-404. 2012. Chania, Crete, Greece. Springer.

[9] Cen, H., K.R. Koedinger, and B. Junker. Comparing Two IRT Models for Conjunctive Skills. *Ninth International Conference on Intelligent Tutoring Systems*, 796-798. 2008. Montreal.

[10] Koedinger, K.R., P.I. Pavlik, J. Stamper, T. Nixon, and S. Ritter. Avoiding problem selection thrashing with conjunctive knowledge tracing. In *Proceedings of the 4th International Conference on Educational Data Mining*. 2011: Eindhoven, NL, p. 91-100.

[11] Gong, Y., J.E. Beck, and N.T. Heffernan. Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. *Proceedings of the 10th International Conference on Intelligent Tutoring Systems*, 35-44. 2010. Pittsburgh, PA. Springer Berlin / Heidelberg.

[12] Xu, Y. and J. Mostow. Using logistic regression to trace multiple subskills in a dynamic Bayes net. *Proceedings of the 4th International Conference on Educational Data Mining*, 241-245. 2011. Eindhoven, Netherlands.

[13] de la Torre, J. and J.A. Douglas. Higher-order latent trait models for cognitive diagnosis. *Psychometrika* 2004. *69*(3): p. 333-353.

[14] Junker, B. and K. Sijtsma. Cognitive assessment models with few assumptions, and connections with nonparametric item response theory. *Applied Psychological Measurement*, 2001. *25*(3): p. 258-272.

[15] de la Torre, J. DINA Model and Parameter Estimation: A Didactic *Journal of Educational and Behavioral Statistics*, 2009. *34*(1): p. 115-130.

[16] Maris, E. Estimating multiple classification latent class models. *Psychometrika*, 1999. *64*(2): p. 197–212.

[17] Xu, Y. and J. Mostow. Using item response theory to refine knowledge tracing. In *Proceedings of the 6th International Conference on Educational Data Mining*, S.K. D'Mello, R.A. Calvo, and A. Olney, Editors. 2013, International Educational Data Mining Society: Memphis, TN, p. 356-357.

[18] González-Brenes, J.P. and J. Mostow. What and when do students learn? Fully data-driven joint estimation of cognitive and student models. In *Proceedings of the 6th International Conference on Educational Data Mining*, S.K. D'Mello, R.A. Calvo, and A. Olney, Editors. 2013, International Educational Data Mining Society: Memphis, TN, p. 236-239.

[19] González-Brenes, J.P. and J. Mostow. Dynamic cognitive tracing: towards unified discovery of student and cognitive models. *Proceedings of the Fifth International Conference on Educational Data Mining* 2012. Chania, Crete, Greece.

[20] Cen, H., K. Koedinger, and B. Junker. Learning factors analysis – a general method for cognitive model evaluation and improvement. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, 164-175. 2006. Jhongli, Taiwan.

[21] Fischer, G.H. The linear logistic test model. In G.H. Fischer and I.W. Molenaar, Editors, *Rasch Models: Foundations, Recent Developments, and Applications*, 131-155. Springer: New York, 1995.

[22] Wang, X., J.O. Berger, and D.S. Burdick. Bayesian analysis of dynamic item response models in educational testing. *Annals of Applied Statistics*, 2013. *7*(1): p. 126-153.

[23] Studer, C. *Incorporating Learning Over Time into the Cognitive Assessment Framework.* Unpublished PhD, Carnegie Mellon University, Pittsburgh, PA, 2012.

[24] Lunn, D., D. Spiegelhalter, A. Thomas, and N. Best. The BUGS project: Evolution, critique and future directions. *Statistics in Medicine*, 2009. *28*: p. 3049–306.

[25] Koedinger, K.R., R.S.J.d. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper. A data repository for the EDM community: the PSLC DataShop. In C. Romero, et al., Editors, *Handbook of Educational Data Mining*, 43-55. CRC Press: Boca Raton, FL, 2010.

The Sequence of Action Model: Leveraging the Sequence of Attempts and Hints

Linglong Zhu
Department of Computer Science
Worcester Polytechnic Institute
100 Institute Road, Worcester, MA
lzhu@wpi.edu

Yutao Wang
Department of Computer Science
Worcester Polytechnic Institute
100 Institute Road, Worcester, MA
yutaowang@wpi.edu

Neil T. Heffernan
Department of Computer Science
Worcester Polytechnic Institute
100 Institute Road, Worcester, MA
nth@wpi.edu

ABSTRACT

Intelligent Tutoring Systems (ITS) have been proven to be efficient providing student assistance and assessing their performance when they do their homework. Researchers have analyzed how students' knowledge grows and predict their performance from within intelligent tutoring systems. 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 and 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.

Keywords

Knowledge Tracing, Educational Data Mining, Student Modeling, Sequence of Action, Assistance Model, Ensemble.

1. INTRODUCTION

One of the student modeling tasks is to trace the student's knowledge by using student's performance. Corbett and Anderson (1995) put forward the well-known Knowledge Tracing (KT) based on their observation that the students' knowledge is not fixed, but is assumed to be increasing. KT model makes use of Bayesian network to model students' learning process and predicate their performance.

A variety of extensions of KT model are put forward in recent years. Baker, Corbett, and Aleven (2008) build a contextual guess and slip model based on KT that provides more accurate and reliable student modeling than KT. Pardos and Heffernan extends KT four parameters model to support individualization and skill specific parameters and get better prediction of students' performance. Qiu and Qi et al. find that forgetting is a more likely cognitive explanation for the over prediction of KT when considering the time students take to finish their tasks.

Alternative methods to KT model have been developed. For example, in order to generate adaptive instructions for students, Pavlik Jr., Cen, and Koedinger (2009) put forward the Performance Factor Analysis (PFA) model that can make predictions for individual students with individual skills. Gong, Beck, and Heffernan (2010) compared KT with PFA using multiple model fitting procedures and showed that there are no real differences in predictive accuracy between these two models.

However, little attention is paid to the data generated when students interact with computer tutors. Shih, Koedinger, and Scheines (2010) utilize Hidden Markov Model clustering to discover different strategies students used while working on a ITS and predict learning outcomes based on these strategies. Their work is based on a dataset that 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 results 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. In fact, the National Educational Technology Plan cited Feng, Heffernan, and Koedinger's work (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 (2011) built an Assistance Model (AM) and generated a performance table based on students' behavior of doing the previous question. Hawkins et al.(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 the first requesting a hint and then making 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 our previous work, we showed a Sequence of Action (SOA) model that made use of information about the action sequence of attempts and hints for a student in previous question better predicted the correctness of a current question.. We reported experimental results of an improvement upon the KT model. However, we later found a mistake in that experiment. So this paper serves as a correction of the previous results and as a formal presentation of the SOA model to the community. 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 does improve upon a

Published in CEUR-WS:
BKT20y workshop (Yudelson, González-Brenes and Mozer)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

standard Knowledge Tracing model, or even upon knowing number of hints and number of attempts alone.

The raw data and experiment result is available online: https://sites.google.com/site/assistmentsdata/projects/zhu2014.

1.1 The Tutoring System and Dataset

The data we used originated from the ASSISTments platform, 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. Figure 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."

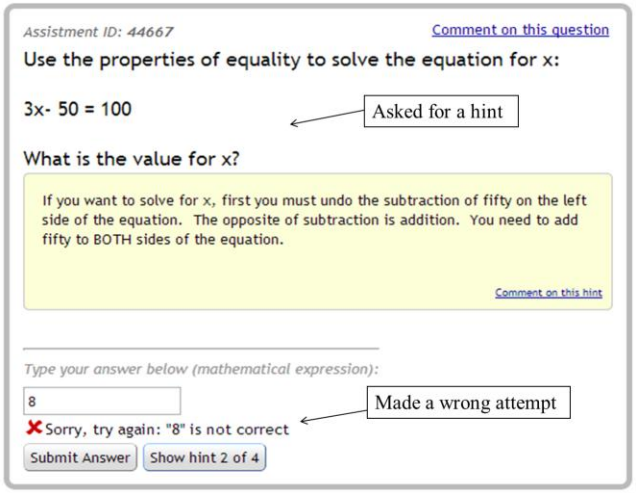


Figure 1. Assistance in ASSISTments. Which is first: asking for a hint or make an attempt?

Figure 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 this was wrong. Though Figure 1 shows the number of hints and attempts, interestingly you cannot tell whether the student asked a hint first or made an attempt first. This paper's argument is that information is very important.

ASSISTments records all the details about how a student does his or 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 problem, the time interval between an attempt, if he or she asks for a hint, the number of attempts a student makes, the number of hints a student asks for, as well as the answer and result for each attempt a student makes.

Figure 2 shows an example of a detailed sequence of action recorded by the system. The row in blue means that the answer is correct, the row in red means that the answer is wrong, and the row in orange means the student asked for a hint. We can see that this student answered correctly on his first attempt for the first problem PRAQM5U. The sequence of action is 'a' ('a' represents an attempt). For the second problem PRAQM2W, he asked three hints continuously before making the correct answer. The sequence of action is 'hhha' ('h' represents a hint). For the third problem PRAQM2F, he alternatively asked for hints and made attempts, and the sequence of action is 'hahaha'. For the last problem PRAQZPN, he made one wrong attempt before making the correct answer and its 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 class. 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 such problem logs where: 1) students quit the system immediately after they saw the question and the action logs were blank ,or 2) after they requested hints, but did not make any attempts and no answer was recorded.

Here we only consider the question pairs that have the same skill and skills having only one question were removed because they do not help in predicting. Questions of the same skills were sorted by start time in ASSISTments. We split equally 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.

2. INDIVIDUAL MODELS

2.1 KT

Knowledge Tracing (KT) is one of the most common methods that are used to model the process of student's knowledge gaining and to predict students' performance. The KT models is an Hidden Markov Model (HMM) with a hidden node (student

knowledge node) and an observed node (student performance node). It assumes that a skill has four parameters; two knowledge parameters and two performance parameters. The two knowledge parameters are: prior and learn. The prior knowledge parameter is the probability that a particular skill was known by the student before interacting with the tutor. The learn parameter is the probability that a student transits from the unlearned state to the learned state after each learning opportunity, i.e., after see a question. The two performance parameters are: guess and slip. Guess is the probability that a student will guess the answer correctly even if the skill associated with the question is in the unlearned state. Slip is the probability that a student will answer incorrectly even if he or she has mastered the skill for that question.

The goal of KT is to estimate the student knowledge from his or her observed actions. At each successive opportunity to apply a skill, KT updates its estimated probability that the student knows the skill, based on the skill-specific learning and performance parameters and the observed student performance (evidence). It is able to capture the temporal nature of data produced where student knowledge is changing over time. KT provides both the ability to predict future student response values, as well as providing the different states of student knowledge. For this reason, KT provides insight that makes it useful beyond the scope of simple response prediction.

2.2 Assistance Model

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

A parameter table was built in which rows represent the number of attempts a student required in the previous question and columns represent the number of hints the student asked for. Each cell contains the probability that the student will answer the current question correctly. The attempts are separated into three bins: one attempt, small number of attempts (2-5 times), and large numbers of attempts (more than five attempts). Hints are separated into four bins: no hint, small number of hints (1, 50%], large number of hints [50%, 100%), and all hints where students for all hints. Table 1 shows the parameter table gained from our dataset. As with Wang and Heffernan's experimental results, the parameter table confirms that students requiring more assistance to solve a problem probably have less corresponding knowledge.

Table 1. Assistance Model parameter table, average across six folds

	attempt= 1	**0<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

2.3 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 sequences 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 and we believe they were learning by themselves. In the data, there are 217 different sequences of actions. Intuitively, students' actions reflect their study attitude and this determines their performance. Based on the assumption that students who make more attempts 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 getting 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; and (5) Alternative, Hint First: the students asked for hint and made attempts alternatively and asked for a hint first. Table 2 shows the division and some examples of the action sequences in each category.

Table 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, …, aaaaaaaaaaa
All Hints/Bin 'h+'	ha, hha,…, hhhhhhha
Alternative, Attempt First/Bin 'a-mix'	aha, aahaaha,…, aahhhhaaa
Alternative, Hint First/Bin 'h-mix'	haa, haha,…, hhhhaha

Notice that each sequence ends with an attempt because in ASSISTments, a student cannot continue to next question unless he or she fills in the right answer of the current problem. In Table 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 or she types the correct answer and moves on to the next question.

2.3.1 Sequence of Action Tabling

After dividing all of sequence of actions into five categories, we use a Tabling method, which gets the next percent correct directly from the training data. For each fold, one table is generated by the tabling method by counting the number of total appearance and the number of next correct of each bin. After counting, a next correct percent is calculated by dividing *Next Correct Count* by *Total Count* of Bin.

Table 2. Next correct percent table of training group of fold 1

Bin Name	**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 shows the table computed for fold 1. Tables for other folds are similar. From Table 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 attitudes for study. They want to get some information from the hint based on which they try to solve the next 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 somewhat decides 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.

2.3.2 Logistic Regression

In this section, we are going to introduce the second part of the SOA model that makes use of a logistic regression model and information we get from the first part of SOA, i.e., tabling method.

Even though the next correct percentage we get from the tabling method indicates that the action of sequence can reflect the trend of next correct percentage, the table is very rough and is not intelligent enough to be used to predict students' performance. However, we can 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 totally 51 skills of the data. As mentioned in before, there are six folds and each fold has their 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. Parts of coefficients of the first fold are shown in Table 4.

Table 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

3. 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.

4. EXPERIMENTAL RESULTS

4.1 Compare AM, SOA and KT

To evaluate how well each of the individual models (SOA, AM, KT) and the combined 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 Area Under ROC Curve (AUC). Lower values for MAE and RMSE and higher values for AUC indicate better model fit.

Table 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**

Table 5 shows values of the three metrics from a six-fold across validation, which are calculated by averaging corresponding numbers obtained from each validation. As with Wang and Heffernan's results (Wang & Heffernan, 2011), the performance of linear regression combination of AM and KT, called as LR(AM, KT) is better than AM itself, which indicates information about the number of hints and attempts improves the prediction of KT model. Overall, the combination of any two models have higher prediction accuracy and this is especially true

that for the average ensemble of SOA and KT, called as AVG(SOA, KT), which has better accuracy than the other two combinations. Also, the linear regression of AM and KT has better prediction accuracy than linear regression combination of SOA and KT. However, from the two tailed paired t-test results shown in Table 6, the statistical difference between any two pairs of model combinations are not significant.

To examine whether there is significant difference between these models, we performed a 2-tailed paired t-test. The p values are shown in Table 6. We observe that most of the differences between two models are reliable, except for when we compare some AM and KT combined models with SOA and KT combined models. Both SOA and AM use the information about students' actions of hints and attempts. There might be a chance that SOA and LR(AM, KT) have some prediction overlap.

Table 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**

4.2 Further Analysis for SOA and KT

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

Tables 7 and 8 shows the student level result 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 8 shows the t-test p value for each pair of these three models, where the remaining degrees of freedom on all the tests is 65.

Table 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 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

Note that there is no significant difference of AUC between KT and SOA. We interpret these results by pointing out that RMSE and AUC are metrics that are optimized for measuring different things, and so this is quite possible.

Table 9 and 10 shows the skill level result across all 51 skills. From Table 9 we observe a very low AUC value for all the models, which indicates these models do not make a good classification at skill level. The t-test p value with remaining degrees of freedom 50 is shown in table 10.

Table 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 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 ensemble outperform KT in all of the three metrics. When we compare the ensemble model with SOA alone, the result is not so clear.

5. DISCUSSION AND FUTURE WORK

In this paper, we put forward a Sequence Of Action model that makes use of sequence of students attempts to answer questions and asking for hints. The SOA model consists of two parts. First, the sequence of students' actions are divided into five categories. A tabling method shows that students who only make attempts tend to answer the next question more correctly than students who only ask for hints. This could be caused by students who make more attempts are trying to figure problems out by themselves and it is an efficient way to master knowledge when they are told the steps to answer these questions by asking for hints. Second, we built a logistic regression model with next question correct percentage as dependent variable and skill_id, credits of sequence of action bins as independent variables.

We conducted six-fold cross validation experiments. The experimental result showed that SOA had reliably higher prediction accuracy than the Knowledge Tracing model and Assistance Model. The average combination of the SOA and KT had the highest prediction. In sum, the sequence of students' actions provided 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 experimental data we used is from ASSISTments, does SOA model still makes a big difference if use data from other intelligent tutor systems? How much can the performance of SOA model be improved if combined with other efficient prediction model such as PFA (Pavlik et al., 2009)? What is the SOA model's performance if we use a student action sequence of several previous questions when we train the model? How does SOA perform after individualization? These are some of the questions that still need to be explored.

6. 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 been paid to the temporally sequential actions of student when interacting with the tutoring system. 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.

We can think of several ways to improve upon this. First, our five bins that we put students into were somewhat arbitrary. There could be more bins or fewer. If we use more bins, we might have very different predictions. The downside is that for some of these bins we might not have enough data points to reliably fit the parameters. One way to make the model better might be to split the "All Hints" bin into one that has "Reached Bottom out Hint" and one that is "All hints excluding those that reached the bottom out." We could also try to pay attention to features like response time between hints or the response time after a hint in making an attempt.

According to our six-fold cross validation experiments and paired two-tailed t-test, both on student level and skill level, our Sequence of Action model had reliably higher prediction accuracy than KT and AM, the later uses the number of hints students ask for and the number of attempts students make. Furthermore, we combined SOA and KT using average and linear regression methods, and the ensemble model's prediction performance was much better than either SOA or KT. We also compared combination of SOA and KT with combination of AM and KT. The experimental result show that SOA contributes more useful information than AM alone, which indicates that the sequential information of action does convey more information about students' learning than the statistics information of actions students make.

7. ACKNOWLEDGMENTS

We acknowledge funding from NSF (#1316736, 1252297, 1109483, 1031398, 0742503), ONR's 'STEM Grand Challenges' and IES (# R305A120125 & R305C100024).

8. REFERENCES

Corbett, A. T., & Anderson, J. R. (1995). Knowledge tracing: modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction, 4*, 253–278.

Baker, R.S.J.d., Corbett, A.T. & Aleven, V. (2008). More Accurate Student Modeling Through Contextual Estimation of Slip and Guess Probabilities in Bayesian Knowledge Tracing. In: Wolf, B., Aimeur, E., Nkambou, R., Lajoie, S. (Eds.) *Intelligent Tutoring Systems. LNCS, 5091*, Springer Berlin. pp. 406-415.

Feng, M., Heffernan, N. & Koedinger, K.R. (2006a). Predicting state test scores better with intelligent tutoring systems: developing metrics to measure assistance required. In Ikeda, Ashley & Chan (Eds.). *Proceedings of the Eighth International Conference on Intelligent Tutoring Systems*. Springer-Verlag: Berlin. pp. 31-40.

Gong, Y., Beck, J. & Heffernan, N. (2010). Comparing Knowledge Tracing and Performance Factor Analysis by Using Multiple Model Fitting. In Aleven, V., Kay, J & Mostow, J. (Eds) *Proceedings of the 10th International Conference on Intelligent Tutoring Systems (ITS2010) Part 1*. Springer-Verlag, Berlin. pp. 35-44.

Hawkins, W., Heffernan, N., Wang, Y. & Baker, S,J,d.. (2013). Extending the Assistance Model: Analyzing the Use of Assistance over Time. In S. D'Mello, R. Calvo, & A. Olney (Eds.) *Proceedings of the 6th International Conference on Educational Data Mining (EDM2013)*. Memphis, TN. pp. 59-66.

Pardos, Z. & Heffernan, N. (2010). Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing. In Paul De Bra, Alfred Kobsa, David Chin, (Eds.) *The 18th Proceedings of the International Conference on User Modeling, Adaptation and Personalization*. Springer-Verlag. pp. 255-266.

Pavlik, P.I., Cen, H., Koedinger, K.R. (2009). Performance Factors Analysis - A New Alternative to Knowledge Tracing. In *Proceedings of the 14th International Conference on Artificial Intelligence in Education*. Brighton, UK. pp. 531-538.

Qiu, Y., Qi, Y., Lu, H., Pardos, Z. & Heffernan, N. (2011). Does Time Matter? Modeling the Effect of Time with Bayesian Knowledge Tracing In Pechenizkiy, M., Calders, T., Conati, C., Ventura, S., Romero , C., and Stamper, J. (Eds.) *Proceedings of the 4th International Conference on Educational Data Mining*. pp. 139-148.

Shih, B., Kenneth K., & Richard S. (2010). Unsupervised Discovery of Student Strategies. In Baker, R.S.J.d., Merceron, A., Pavlik, P.I. Jr. (Eds.) *Proceedings of the 3rd International Conference on Educational Data Mining*. pp. 201-210.

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

Using Similarity to the Previous Problem to Improve Bayesian Knowledge Tracing

William J. Hawkins
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609
whawkins90@gmail.com

Neil T. Heffernan
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609
nth@wpi.edu

ABSTRACT

Bayesian Knowledge Tracing (BKT) is a popular student model used extensively in educational research and in intelligent tutoring systems. Typically, a separate BKT model is fit per skill, but the accuracy of such models is dependent upon the skill model, or mapping between problems and skills. It could be the case that the skill model used is too coarse-grained, causing multiple skills to all be considered the same skill. Additionally, even if the skill model is appropriate, having problems that exercise the same skill but look different can have effects on student performance. Therefore, this work introduces a student model based on BKT that takes into account the similarity between the problem the student is currently working on and the one they worked on just prior to it. By doing this, the model can capture the effect of problem similarity on performance, and moderately improve accuracy on skills with many dissimilar problems.

Keywords

Student modeling, Bayesian Knowledge Tracing, Problem Similarity

1. INTRODUCTION

Bayesian Knowledge Tracing (BKT) [3] is a popular student model used both in research and in actual intelligent tutoring systems. As a model that infers student knowledge, BKT has helped researchers answer questions about the effectiveness of help within a tutor [1], the impact of "gaming the system" on learning [5], and the relationship between student knowledge and affect [9], among others. Additionally, it has been used in the Cognitive Tutors [6] to determine which questions should be presented to a student, and when a student no longer needs practice on a given skill.

However, BKT models are dependent upon the underlying skill model of the system, as a separate BKT model is typically fit per skill. If a skill model is too coarse-grained or too fine-grained, it can make it more difficult for a BKT model to accurately infer student knowledge [8].

Additionally, even when a skill model is tagged at the appropriate level, seeing similar problems consecutively as opposed to seeing dissimilar problems may have effects on guessing and slipping, two important components of BKT models. For example, if a student does not understand the skill they are working on, seeing a certain type of question twice or more consecutively may improve their chances of "guessing" the answer using a suboptimal procedure that would not work on other questions from the same skill.

Whether the skill model is not at the appropriate level or seeing consecutive similar questions helps students succeed without fully learning a skill, it may be important to take problem similarity into account in student models. In this work, we introduce the Bayesian Knowledge Tracing – Same Template (BKT-ST) model, a modification of BKT that considers problem similarity. Specifically, using data from the ASSISTments system [4], the model takes into account whether the problem the student is currently working on was generated from the same *template* as the previous problem.

The next section describes the ASSISTments system, its template system and the data used for this paper. Section 3 describes BKT and BKT-ST in more detail, and describes the analyses we performed on these models. The results are reported in Section 4, followed by discussion and possible directions for future work in Section 5.

2. TUTORING SYSTEM AND DATA

2.1 ASSISTments

ASSISTments [4] is a freely available web-based tutoring system used primarily for middle and high school mathematics. In addition to providing a way for teachers to assess their students, ASSISTments also assists the students in a few different ways: through the use of series of on-demand hint messages that typically end in the answer to the question (the "bottom-out hint"), "buggy" or feedback messages that appear when the student gives a common wrong answer, and "scaffolding" questions that break the original question into smaller questions that are easier to answer.

While teachers are free to author their own content, ASSISTments provides a library of approved content, which includes problem sets called *skill-builders*, which are meant to help students practice a particular skill. While most problem sets contain a fixed number of problems that must all be completed for a student to finish, a skill-builder is a special type of problem set that assigns questions in a random order and that is considered complete once a student answers three consecutive questions correctly on the same day.

Published in CEUR-WS:
BKT20y workshop (Yudelson, González-Brenes and Mozer)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

While requiring students to answer three consecutive questions correctly on the same day to complete a skill-builder ensures that they have some level of knowledge of the particular skill being exercised, it takes some students many problems to achieve this, meaning they may see the same problem more than once if the skill-builder does not contain enough unique problems.

To ensure this does not happen (or at least make it highly unlikely), ASSISTments has a templating system that facilitates creating large numbers of similar problems quickly. The content creator creates a question as normal, but specifies that it is a *template* and uses variables in the problem statement and answer rather than specific values. Then, they are able to generate 10 unique problems at a time from that template, where each problem is randomly populated with specific values as prescribed by the template. This is especially useful for skill-builders, whose problems should theoretically all exercise the same skill. Figure 1 shows an example of a template (a) and a problem generated from it (b).

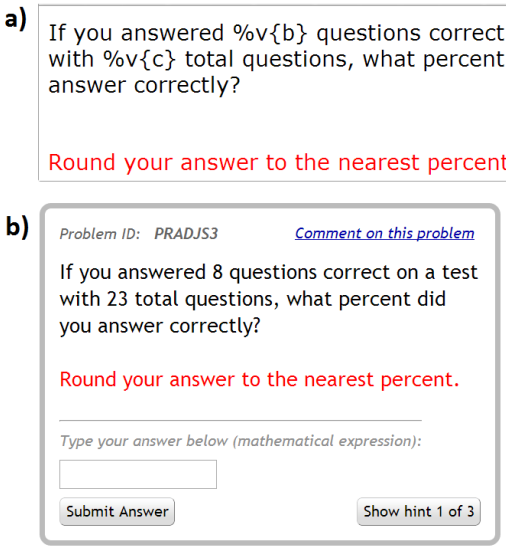


Figure 1. A template (top image) and a problem generated from it (bottom). The variables 'b' and 'c' in the template are replaced by '8' and '23' in the generated problem.

2.2 Data

In this work, we used ASSISTments skill-builder data from the 2009-2010 school year. This data set consists of 61,522 problem attempts by 1,579 students, spread across 67 different skill-builders. A (student, skill-builder) pair was only included if the student attempted three or more problems on that particular skill-builder, and a skill-builder was included if it was used by at least 10 students and at least one of them completed it.

3. METHODS

In this section, we begin by describing Bayesian Knowledge Tracing, and then move on to our modification of it, called Bayesian Knowledge Tracing – Same Template. Finally, we describe the analyses we performed using these two models.

3.1 Bayesian Knowledge Tracing

Bayesian Knowledge Tracing (BKT) [3] is a popular student model that uses a dynamic Bayesian network to infer student knowledge using only a student's history of correct and incorrect responses to questions that exercise a given knowledge component (or "skill").

Typically, a separate BKT model is fit for each skill. BKT models assume that there are only two states a student can be in for a given skill: the *known state* or the *unknown state*. Using a student's performance history on a given skill, a BKT model infers the probability that the student is in the *known state* on question *t*, $P(K_t)$.

Fitting a BKT model involves estimating four probabilities:

1. Prior Knowledge – $P(L_0)$: the probability the student knew the skill before answering the first question
2. Learn Rate – P(T): the probability the student will know the skill on the next question, given that they do not know the skill on the current question
3. Guess Rate – P(G): the probability the student will answer the current question correctly despite not knowing the skill
4. Slip Rate – P(S): the probability the student will answer the current question incorrectly despite knowing the skill

Note that forgetting is typically not modeled in BKT: it is assumed that once a student learns a skill, they do not forget it. An example of a BKT model, represented as a static unrolled Bayesian network, is shown in Figure 2.

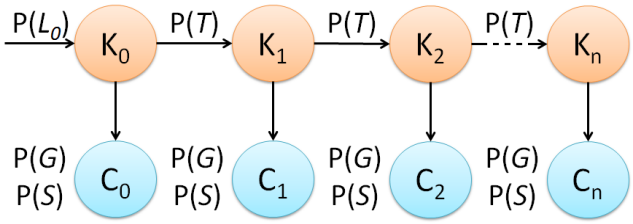


Figure 2. Static unrolled representation of Bayesian Knowledge Tracing. The K_t nodes along the top represent latent knowledge, while the C_t nodes represent performance.

3.2 Bayesian Knowledge Tracing – Same Template

The Bayesian Knowledge Tracing - Same Template (BKT-ST) model differs from the regular BKT model in one way: it takes into account whether the problem it's about to predict was generated from the same template as the previous problem the student worked on. This is modeled as a binary observed variable that influences performance.

This results in six parameters to be learned per skill: the initial knowledge rate, the learn rate, and two sets of guess and slip rates: one set for when the previous problem and current problem were generated from the same template (P(G|Same) and P(S|Same)), and one for when they aren't (P(G|Different) and P(S|Different)). The model is shown in Figure 3.

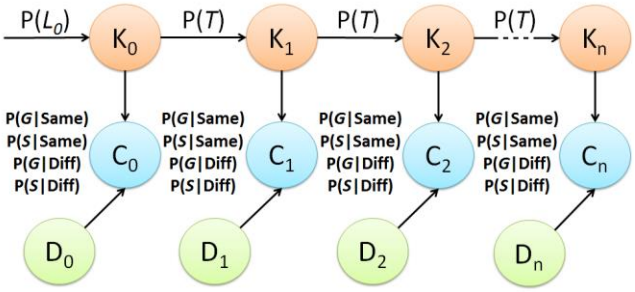


Figure 3. Static unrolled representation of Bayesian Knowledge Tracing – Same Template. The only difference from BKT is the presence of the D_t nodes, which represent whether the previous question was generated by the same template as the current one.

3.3 Analyses

The first analysis in this work simply considers how well the two models fit the data compared to each other overall. This is determined by fitting separate BKT and BKT-ST models for each skill and then predicting unseen student data using five-fold student-level cross-validation. Then, we evaluate each model's ability to predict next question correctness by computing the mean absolute error (MAE), root mean squared error (RMSE) and area under the curve (AUC) for each student and then averaging across students for each type of model. Finally, two-tailed paired t-tests are used to determine the significance of the differences in the metrics.

The second analysis considers what the metrics look like for each model based on how many templates were used for each skill-builder problem set. This is done by splitting the predictions made in the first analysis by how many templates were used in the corresponding skill-builder. We did this to see when it would be worth using BKT-ST over BKT.

Finally we consider the parameter values learned for the BKT-ST model to determine any effects that seeing problems generated by the same template consecutively has on guessing and slipping.

The BKT and BKT-ST models used in these analyses are fit using the Expectation-Maximization (EM) algorithm in the Bayes Net Toolbox for Matlab (BNT) [7]. The initial values given to EM for BKT were 0.5 for $P(L_0)$ and 0.1 for the other three parameters. This was also true for BKT-ST, except the slip rate was set to 0.2 when the current and previous problems were generated from the same template.

4. RESULTS

In this section, we first present the overall comparison of BKT and BKT-ST, then show how they compare to each other based on the number of templates used in each skill-builder. Finally, we examine the learned parameters for the BKT-ST model.

4.1 Overall

The overall results comparing BKT to BKT-ST are shown in Table 1.

Table 1. Overall results of fitting BKT and BKT-ST models.

	MAE	RMSE	AUC
BKT	0.3830	0.4240	0.5909
BKT-ST	0.3751	0.4205	0.6314

According to these results, BKT-ST outperforms BKT in all three metrics. Statistical tests confirmed that these results were reliable (MAE: $p < .0001$, $t(1578) = 9.939$; RMSE: $p < .0001$, $t(1578) = 4.825$; AUC: $p < .0001$, $t(1314) = -11.095$), though according to the values in the table, the only noticeable gain was in AUC.

4.2 By Number of Templates

Next, we considered how well each model did based on the number of templates a skill-builder contained. The results are shown in Figure 4.

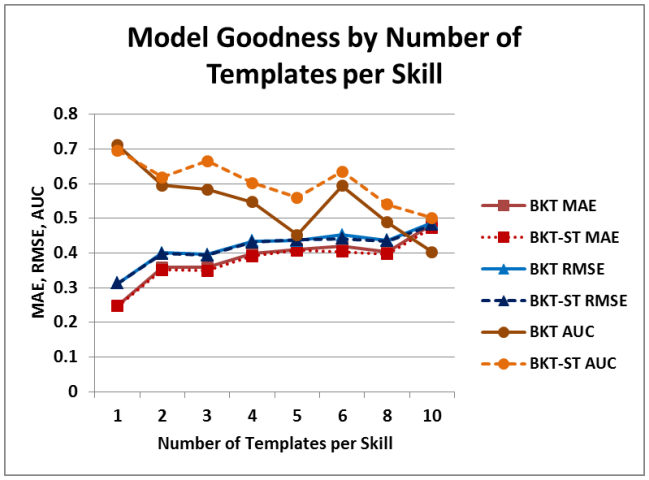


Figure 4. Graph of MAE, RMSE and AUC for the BKT and BKT-ST models, plotted against the number of unique templates per skill.

Interestingly, both BKT and BKT-ST decline rapidly in terms of model goodness as the number of templates per skill-builder increases. This is likely the case because those with more templates are more likely to have more than one skill being tested within them. Interestingly, although both models decline similarly in terms of MAE and RMSE, BKT-ST declines at a slower rate than BKT does in terms of AUC. In fact, BKT-ST outperforms BKT in terms of AUC for every group of skills with more than one template. When grouping the skills by the number of templates they had, BKT-ST achieved an AUC of at least 0.0236 better than BKT for each group that had more than one template, and achieved AUC values that were 0.1086 and 0.0980 better than BKT for skills with five and 10 templates, respectively. Additionally, while BKT performs worse than chance (AUC < 0.5) on skills with eight or more templates, BKT-ST never performs worse than chance.

4.3 Parameter Values

To analyze the parameters learned by BKT-ST, for each skill, we took the average value of each of the six parameters learned across the five folds from the overall analysis.

First, we computed the average value of each parameter across all 67 skills. These are shown in Table 2.

Table 2. Means and standard deviations of BKT-ST parameter values learned across 67 skill-builders

Parameter	Mean	SD
$P(L_0)$	0.6030	0.2617
P(T)	0.2966	0.2500

P(G|Different)	0.1880	0.1655
P(S|Different)	0.2941	0.1737
P(G|Same)	0.3337	0.2495
P(S|Same)	0.1514	0.0848

From the results in Table 2, it appears that on average, seeing consecutive questions generated from the same template both increases the guess rate ($p < .0001$, $t(66) = -4.516$) and decreases the slip rate ($p < .0001$, $t(66) = 7.186$).

Next, we examined how these parameters changed with respect to the number of templates used per skill-builder. The average values of the performance parameters (guess and slip rates for same and different templates) are shown in the graph in Figure 5. The results for skills with one template are omitted since the P(G|Different) and P(S|Different) parameters are meaningless in such cases.

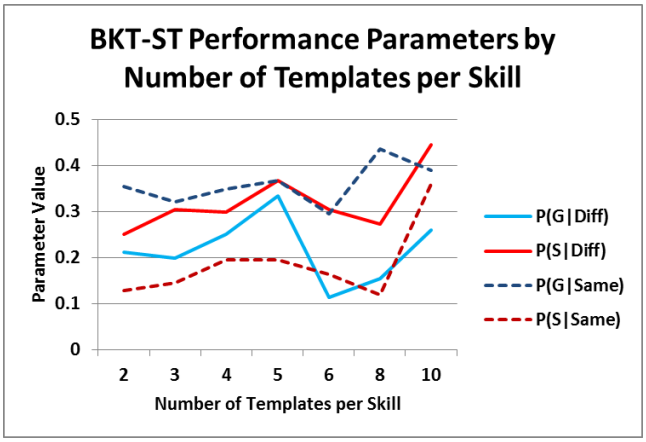


Figure 5. Average value of each performance parameter for the number of templates used per skill-builder.

Although there is no clear pattern for any of the four performance parameters shown in the graph, the average value of P(G|Same) is always higher than that of P(G|Different), and that of P(S|Same) is always lower than that of P(S|Different), with respect to the number of templates used per skill. This appears to reinforce the notion that seeing consecutive problems generated from the same template makes the latter easier to solve, whether this is due to the skill model being too coarse-grained or familiarity with a certain type of problem within a skill inflating performance.

5. DISCUSSION AND FUTURE WORK

From the results in this work, it appears that modifying Bayesian Knowledge Tracing to take similarity between consecutive problems into account moderately improves cross-validated predictive performance, especially in terms of AUC. Additionally, this work showed that seeing consecutive similar problems improves student performance by both increasing the guess rate – the probability of answering a question correctly despite not knowing the skill – and decreasing the slip rate – the probability of answering a question incorrectly despite knowing the skill. Regardless of the underlying reason for this, whether it is because the skill model is too coarse-grained or simply that familiarity with a type of problem within a skill improves performance, it appears important for student models to take the similarity of the problems students encounter into account when trying to model student knowledge.

One direction for future work would be to try going back further in the problem sequence to see how the similarity of problems earlier in a student's history affects their ability to answer the current problem. Additionally, it would be interesting to determine whether the effect changes in certain situations. For example, what is the effect of seeing two similar problems in a row, followed by one that is different from both?

Another area of interest would be to use a model that takes problem similarity into account when trying to predict a longer-term outcome, such as wheel-spinning [2], retention and transfer, as opposed to simply predicting next question correctness.

Finally, applying this model and others like it to other learning environments and skill models of various grain sizes would be helpful for understanding when it is useful. Presumably, if a skill model is at the appropriate grain size, the difference in predictive performance between BKT and BKT-ST would be reduced. The same would be true of systems that fall to one of two extremes: those whose problem sets are highly repetitive, and those whose problem sets have a rich variety of problems.

6. ACKNOWLEDGMENTS

We acknowledge funding from NSF (#1316736, 1252297, 1109483, 1031398, 0742503), ONR's 'STEM Grand Challenges' and IES (# R305A120125 & R305C100024).

7. REFERENCES

[1] Beck, J.E., Chang, K., Mostow, J., Corbett, A. Does help help? Introducing the Bayesian Evaluation and Assessment methodology. *Intelligent Tutoring Systems*, Springer Berlin Heidelberg, 2008, 383-394.

[2] Beck, J. E., and Gong, Y. Wheel-Spinning: Students Who Fail to Master a Skill. In *Artificial Intelligence in Education*, pp. 431-440. Springer Berlin Heidelberg, 2013.

[3] Corbett, A. and Anderson, J. Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. *User Modeling and User-Adapted Interaction*, *4*(4), 253-278.

[4] Feng, M., Heffernan, N.T., Koedinger, K.R. Addressing the assessment challenge in an Intelligent Tutoring System that tutors as it assesses. *User Modeling and User-Adapted Interaction*, *19*(3), 243-266.

[5] Gong, Y., Beck, J., Heffernan, N., Forbes-Summers, E, The impact of gaming (?) on learning at the fine-grained level. in *Proceedings of the 10th International Conference on Intelligent Tutoring Systems*, (Pittsburgh, PA, 2010), Springer, 194-203.

[6] Koedinger, K.R., Anderson, J.R., Hadley, W.H., Mark, M.A. (1997). Intelligent Tutoring Goes To School in the Big City. *International Journal of Artificial Intelligence in Education*, *8*(1), 30-43.

[7] Murphy, K. The bayes net toolbox for matlab. *Computing science and statistics*, *33*(2), 1024-1034.

[8] Pardos, Z. A., Heffernan, N. T., & Anderson, B., Heffernan, C. L. Using Fine-Grained Skill Models to Fit Student Performance with Bayesian Networks. *Proceedings of the Workshop in Educational Data Mining held at the 8th Interna-*

tional Conference on Intelligent Tutoring Systems. (Taiwan, 2006).

[9] San Pedro, M., Baker, R.S.J.d, Gowda, S.M., Heffernan, N.T. Towards an Understanding of Affect and Knowledge from Student Interaction with an Intelligent Tutoring System. In Lane, H.C., Yacef, K., Mostow, M., Pavlik, P. (Eds.) AIED 2013. LNCS, vol. 7926/2013, pp.41-50. Springer-Verlag, Berlin Heidelberg.

Is this Data for Real?

Rinat B. Rosenberg-Kima
University of California, Berkeley
rosenbergkima@berkeley.edu

Zachary Pardos
University of California, Berkeley
pardos@berkeley.edu

ABSTRACT
Simulated data plays a central role in Educational Data Mining and in particular in Bayesian Knowledge Tracing (BKT) research. The initial motivation for this paper was to try to answer the question: *given two datasets could you tell which of them is real and which of them is simulated?* The ability to answer this question may provide an additional indication of the goodness of the model, thus, if it is easy to discern simulated data from real data that could be an indication that the model does not provide an authentic representation of reality, whereas if it is hard to set the real and simulated data apart that might be an indication that the model is indeed authentic. In this paper we will describe initial analysis that was performed in an attempt to address this question. Additional findings that emerged during this exploration will be discussed as well.

Keywords
Bayesian Knowledge Tracing (BKT), simulated data, parameters space.

1. INTRODUCTION
Simulated data has been increasingly playing a central role in Educational Data Mining [1] and Bayesian Knowledge Tracing (BKT) research [1, 4]. For example, simulated data was used to explore the convergence properties of BKT models [5], an important area of investigation given the identifiability issues of the model [3]. In this paper, we would like to approach simulated data from a slightly different angle. In particular, we claim that the question,"*given two datasets could you tell which of them is real and which of them is simulated?",* is interesting as it can be used to evaluate the goodness of a model and may potentially serve as an alternative metric to RMSE, AUC, and others. We would like to start approaching this problem in this paper by comparing simulated data to real data with Knowledge Tracing as the model.

Knowledge Tracing (KT) models are widely used by cognitive tutors to estimate the latent skills of students [6]. Knowledge tracing is a Bayesian model, which assumes that each skill has 4 parameters: two knowledge parameters including initial (prior knowledge) and learn rate, and two performance parameters including guess and slip. KT in its simplest form assumes a single point estimate for prior knowledge and learn rate for all students, and similarly identical guess and slip rates for all students. Simulated data has been used to estimate the parameter space and in particular to answer questions that relate to the goal of maximizing the log likelihood (LL) of the model given parameters and data, and improving prediction power [7], [8], [9].

In this paper we would like to use the KT model as a framework for comparing the characteristics of simulated data to real data, and in particular to see whether it is possible to distinguish between the real and sim datasets.

2. DATA SETS
To compare simulated data to real data we started with 2 real dataset generated from the assisstment software[1] (specifically, datasets G6.207-exact.txt with 776 students and G6.259-exact.txt with 212 students) from a previous BKT study [10]. Both of the datasets consist of 6 questions in linear order where all students answer all questions. Next, we generated synthetic, simulated data using the best fitting parameters that were found for the real data as the generating parameters. By this we generated a simulated version of dataset G6.207 and a simulated version of dataset G6.259 that had the exact same number of questions, number of students, and was generated with what appears to be the best fitting parameters. The specific best fitting parameters that were found for each dataset and were used to generate the simulated data are presented in table 1.

Table 1. Best fitting parameters for each dataset. These parameters were used to generate the simulated datasets.

	N	Prior	Learn	Guess	Slip
G6.207	776	.453	.068	.270	.156
G6.259	212	.701	.044	.243	.165

3. METHODOLOGY
We are interested to find out whether it is possible to distinguish between the simulated data and the real data. The approach we took was to calculate LL for the gird of all the parameters space (prior, learn, guess, and slip). We hypothesized that the LL pattern of the simulated data and real data will be different across the parameters space. For each of the matrices we conducted a grid search with intervals of .04 that generated 25 intervals for each parameter and 390,625 total combinations of prior, learn, guess, and slip. For each one of the combinations LL was calculated and placed in a four dimensional matrix. We used fastBKT [11] to (a) calculate the best fitting parameters of the real datasets, (b) generate simulated data, and (c) calculate the LL of the parameters space. Additional code in Matlab and R was generated to put all the pieces together[2]. In particular, we calculated the LL for all the combinations of two parameters where the other two parameters were fixed to the best fitting value. In an additional analysis, we let all parameters be free and took the average LL for all combinations of two parameters, collapsed over the space of the other two parameters not visualized. The motivation for this was to visualize the error space interactions in the four dimensions of the model.

[1] Data can be obtained here: http://people.csail.mit.edu/zp/

[2] Matlab and R code will be available here:
[2] Matlab and R code will be available here:
http://myweb.fsu.edu/rr05/

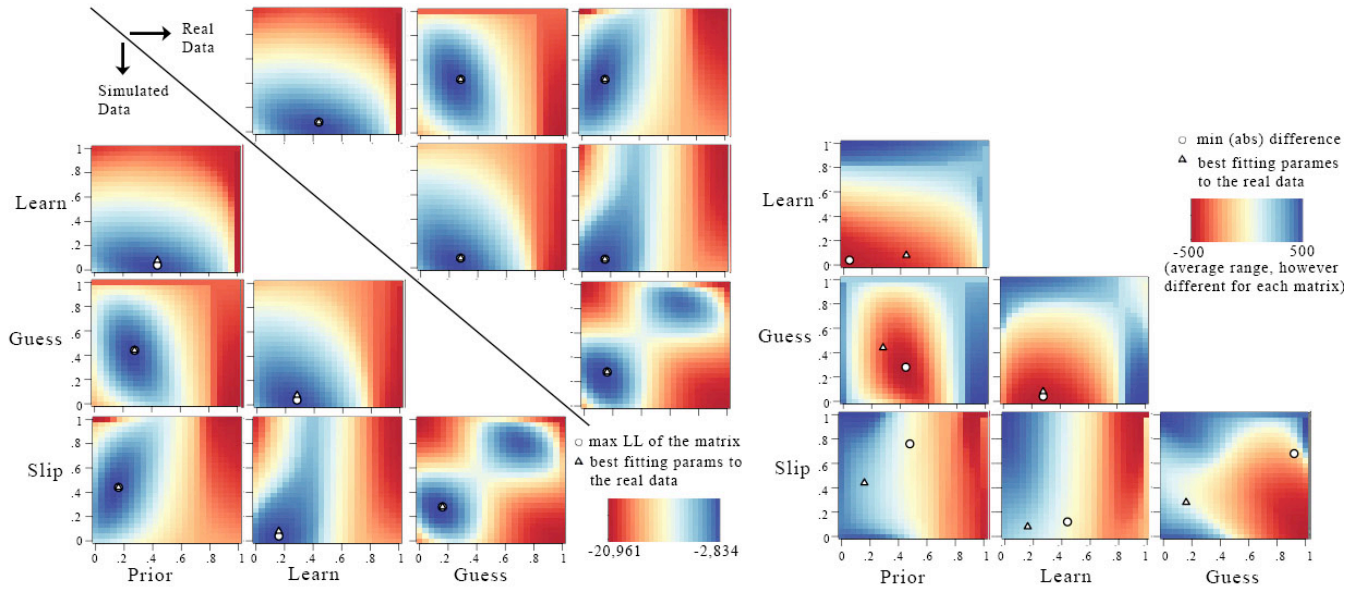


Figure 1.a (left). Heat maps of LL of real assistment dataset G6-207 (k=776 students) and a corresponding simulated data that was generated with the best fitting parameters of the real dataset. The two parameters not in each figure were fixed to the best parameters. Blue areas indicate high LL, and red areas indicate lower LL. Circles indicate maximum LL of the given matrix, and triangles indicate the best fitting parameters to the real data (that were also used to generate the simulated data). In this case the triangles and circles fit the same point.

Figure 1.b (right). Heat maps of delta LL between real dataset G6-207 and the corresponding simulated data that was generated with the best fitting parameters of the real dataset. The two parameters not in each figure were fixed to the best parameters. Blue areas indicate high difference between the real and sim LL, and red areas indicate lower difference. Circles indicate minimum absolute delta of the given matrix, and triangles indicate the best fitting parameters to the real data.

4. DOES THE LL OF SIM vs. REAL DATA LOOK DIFFERENT?

Our initial thinking was that as we are using a simple BKT model, it is not authentically reflecting reality in all its detail and therefore we will observe different patterns of LL across the parameters space between the real data and the simulated data. The LL space of simulated data in [5] was quite striking in its smooth surface but the appearance of real data was left as an open research question.

4.1 Does the LL of sim vs. real data looks different across two parameters grids?

First, we calculated the LL over all the combinations of two parameters for dataset G6.207 where the other two parameters were fixed to the best fitting value. For example, when we calculated LL for the combination of slip and prior (top right figure in figure 1.a), we fixed learn and guess to be .068 and .270 accordingly. To our great surprise, when we plotted heat maps of the LL matrices of the real data and the simulated data (Figure 1.a - real data is presented in the upper triangle and simulated (sim) data is presented in the lower triangle) we received what appears to be identical matrices (for example, the upper right heat map is the (slip x prior) LL matrix of the real data, whereas the lowest left heat map is the (slip x prior) LL matrix of the sim data).

The extent of the similarity between the matrices was surprising and in order to get a better picture of the differences between them we plotted heat maps of the deltas between the real data and the simulated data (LL_RealData-LL_SimData) for each matrix. Even though the matrices appear to be identical, as can be seen in Figure 1.b, there is in fact a difference between the LL of the matrices although it is not a big difference compared to the values of LL. Another surprising finding was that the LL of the real data was in many cases higher than the LL of the sim data. We expected that the model would better explain the sim data as there should not be additional noise as expected in reality, and therefore the LL of the sim data should be higher, yet the findings were not consistent with this expectation.

Another interesting finding was that the location of the ground truth (the triangle) in most of the cases resulted in smaller delta between the real and the sim data although not in all cases (e.g., guess x slip). Note that the circles in Figure 1.b indicate the minimum absolute difference in LL between the real and the sim data, and this point is usually not located at the exact ground truth (except for learn x guess).

Another interesting finding can be seen in Figure 1.a - slip vs. guess. Much attention has been given to this LL space which revealed the apparent co-linearity of BKT with two primary areas of convergence, the upper right area being a false, or "implausible" converging area as defined by [3]. What is interesting in this figure is that despite what appears to be two global maxima, the point with the best LL in this dataset is in fact the lower region for both sim and real data.

Next we conducted the same analysis with the second dataset.

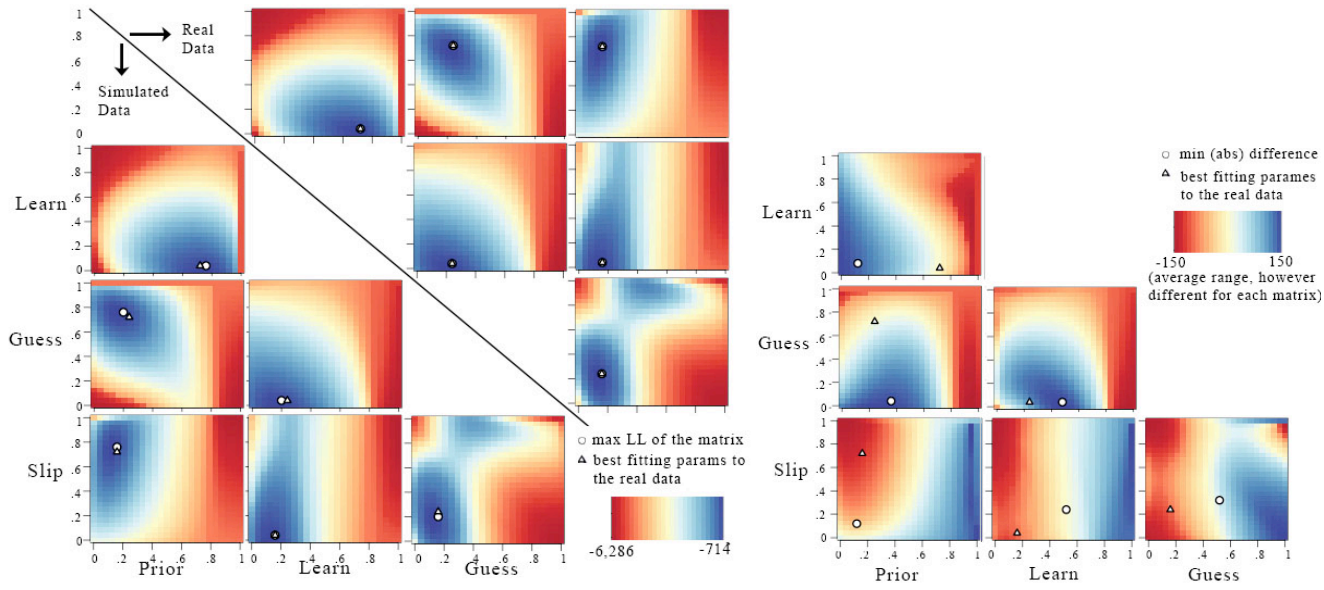


Figure 2.a (left) Heat maps of delta LL between real dataset G6-259 (k=212 students) and the corresponding simulated data that was generated with the best fitting parameters of the real dataset. The two parameters not in each figure were fixed to the best parameters. Blue areas indicate high difference between the real and sim LL, and red areas indicate lower difference. Circles indicate maximum LL of the given matrix, and triangles indicate the best fitting parameters to the real data.

Figure 2.b (right). Heat maps of delta LL between real assistment dataset G6-259 and the corresponding simulated data that was generated with the best fitting parameters of the real dataset. The average is across the two parameters not in each figure. Blue areas indicate high difference between the real and sim LL, and red areas indicate lower difference. Circles indicate minimum absolute delta of the given matrix, and triangles indicate the best fitting parameters to the real data.

Even though the G6-259 dataset was significantly smaller than the first dataset, we received very similar results to the first dataset with surprisingly similar heat maps for the sim and real data (see Figure 2.a). Like in the first dataset, notice that even though the LL heat maps look very similar, there is a difference in the delta heat maps (see Figure 2.b). Nevertheless, there is an interesting difference between the two datasets. Concretely, unlike the bigger dataset (G6-207), in G6-259 the LL of the sim data was actually higher than the real data in most cases.

4.2 What if we average LL over 2 parameters across all the combinations of the other 2 parameters?

We were interested to find out how will the heat maps look like if we do not fix the other two parameters to be best fit, but rather average the LL across the entire space of the other two parameters. For example, to calculate the matrix of guess and slip we practically calculated a matrix of guess and slip LL for each combination of learn and prior (25 x 25 = 625 matrices) instead of only one matrix for the best fit learn and prior. Then, we took the average of all these matrices for each combination of guess and slip (see Figure 3.a). The results are both surprising and interesting. As far as (guess x slip), we no longer receive the two maximum (global and local) that we received when learn and prior where fixed to best fit parameters. Another interesting finding is the relationship between the average maximum across the other two parameters and the overall best fit parameters for given two parameters. For example, if we look at the heat map of matrix (learn x prior) we can see that there is not a big difference between the average maximum point (white circle) and the overall best fit parameters (white triangle). This may indicate that changing guess and slip will not affect the value of learn and prior that maximizes the LL, therefore might suggest independency. If we look at (guess x learn), we see that changes in prior and slip will again not have an impact on the best fit value of guess, however, they will affect the value of learn. Then again, if we look at the heat map of (prior x guess), we will see that both prior and guess are sensitive to changes in learn and slip. Yet again, the extremely surprising part of these results is that the sim data appear to be almost identical to the real data. It is possible to see from Figure 3.b though that indeed there are differences between the simulation data and the real data and like before, the LL of the real data is higher than that of the sim data in the larger dataset.

Like for the fixed matrices, we received similar LL matrices for the smaller dataset (G6-259) (see table 4.a). In addition, as before, the LL of the sim data for this dataset was higher than that of the real data (the opposite direction of the larger dataset G6-207). Another interesting finding for this dataset can be seen in the (guess x slip) matrices (4.b). Notice that while the sim data converged to the lower point of the blue area, the real data converged to the higher point. Nevertheless, this only happened in the averages matrices and not in the fixed ones.

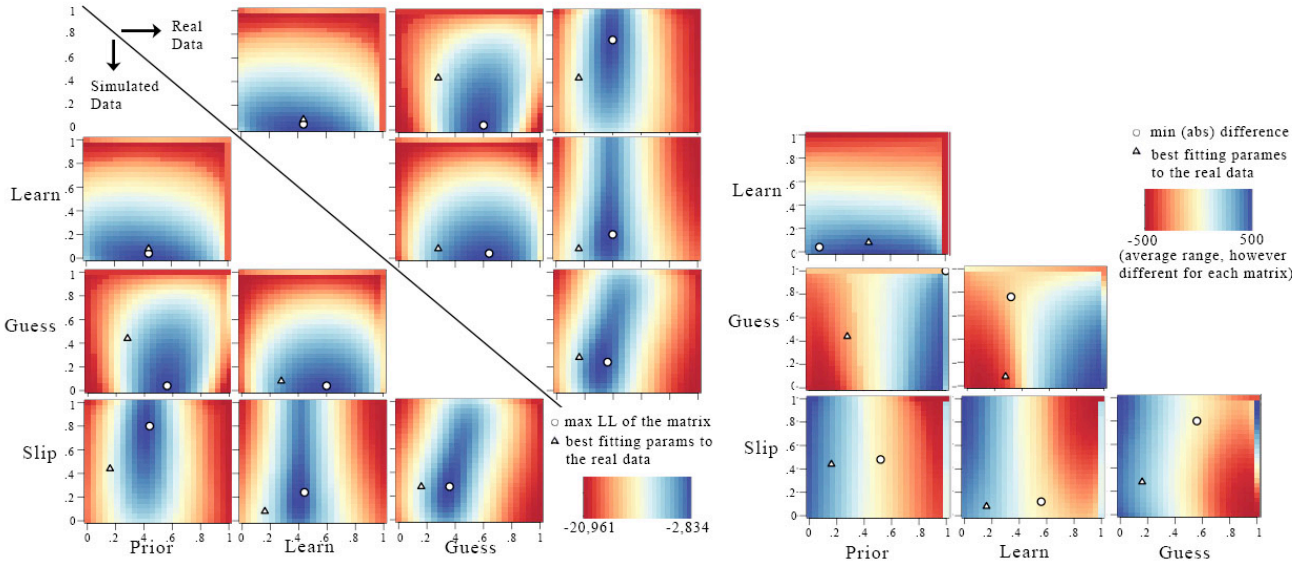


Figure 3.a (left). Heat maps of average LL of real assistment dataset G6-207 (k=776 students) and a corresponding simulated data that was generated with the best fitting parameters of the real dataset. The average is across the two parameters not in each figure. Blue areas indicate high LL, and red areas indicate lower LL. Circles indicate maximum LL of the given matrix, and triangles indicate the best fitting parameters to the real data (that were also used to generate the simulated data).

Figure 3.b (right). Heat maps of delta LL between real assistment dataset G6-207 and the corresponding simulated data that was generated with the best fitting parameters of the real dataset. The average is across the two parameters not in each figure. Blue areas indicate high difference between the real and sim LL, and red areas indicate lower difference. Circles indicate minimum absolute delta of the given matrix, and triangles indicate the best fitting parameters to the real data.

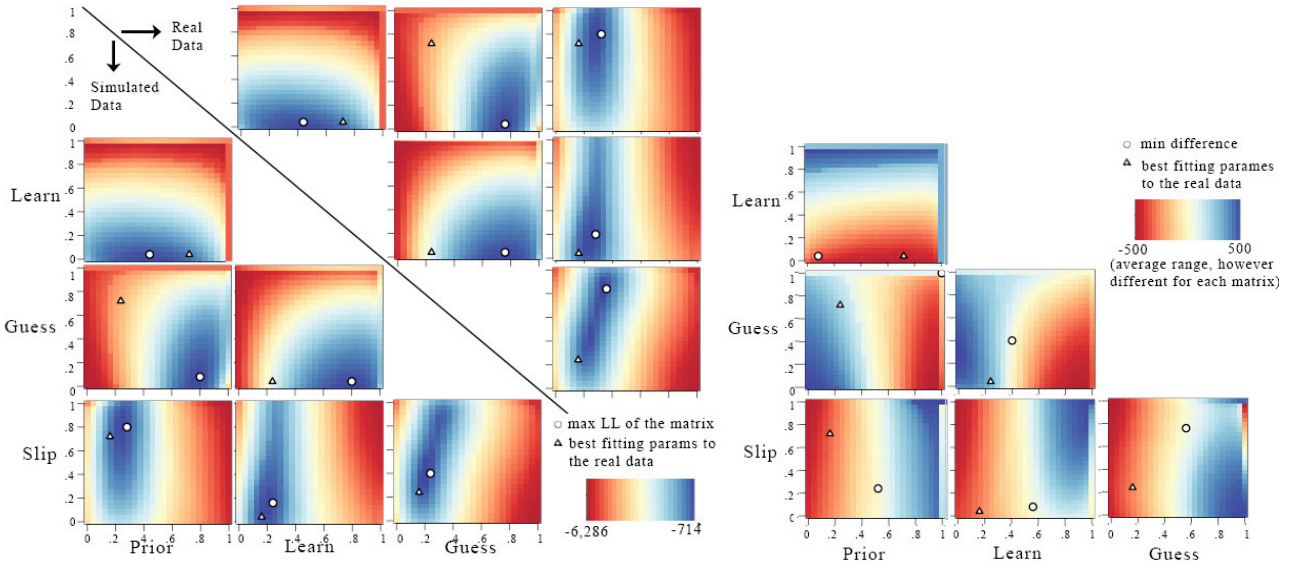


Figure 4.a (left). Heat maps of average LL of real assistment dataset G6-259 (k=212 students) and a corresponding simulated data that was generated with the best fitting parameters of the real dataset. The average is across the two parameters not in each figure. Blue areas indicate high LL, and red areas indicate lower LL. Circles indicate maximum LL of the given matrix, and triangles indicate the best fitting parameters to the real data (that were also used to generate the simulated data).

Figure 4.b (right). Heat maps of delta LL between real assistment dataset G6-259 and the corresponding simulated data that was generated with the best fitting parameters of the real dataset. The average is across the two parameters not in each figure.

5. DISCUSSION AND FUTURE WORK

The initial motivation of this paper was to find whether it is possible to discern a real data from a sim data. If for a given model it is possible to tell apart a sim data from a real data then the authenticity of the model can be questioned. This line of thinking is in particular typical of simulation use in Science context, where different models are used to generate simulated data, and then if a simulated data has a good fit to the real phenomena at hand, then it may be possible to claim that the model provides an authentic explanation of the system [12]. We believe that it may be possible to generate a new matric for evaluating the goodness of a model by comparing a simulated data from this model to real data.

In this work we explored similarities between simulated and real data. Nevertheless, we are yet to answer the question "is this data for real?". In other words, what we still did not do in this work is come up with an algorithm that can take a dataset and determine whether it is real or simulated. Another way to think of it is to come out with an algorithm that can tell us whether it is possible to discern real and simulated data and use it as an indication of the goodness of the model. We found differences between the real and sim data, but are they strong enough to be noticed by such algorithm in a consistent way? In future work we plan to further investigate this question by creating a training set of multiple real datasets and sim datasets and use machine learning techniques to extract a learning algorithm from this training dataset that can take as input a dataset and determine whether it is real or sim. We argue that if such algorithm can be found, it is an indication that the underlying model can be improved. In future work we also plan to compare different variations of the KT model and contrast their resulting simulated data with real data. In particular we plan to generate a more complex set of simulated data that is based on a more complex model (e.g., different learning rate for different types of questions), and then use it as "real" data with the (wrong) assumption that the model is simple (standard BKT model) to simulate a scenario where the real data is indeed grounded in more complex model than our assumptions and see what results would a learning algorithm that uses this "real" data in comparison to a sim data will yield.

In addition, this paper raises interesting questions that we did not think of while trying to answer our initial question. For example, it seems like there is potential to dive deeper into the average LL (Figures 3&4) and find more about the relationships and dependencies between the different parameters. Another question that emerged is how could it be that the simulated data had lower LL than the real data in the bigger dataset yet lower in the smaller dataset? Further analysis is needed to answer these questions.

Last but not least, given the remarkable resemblance between the sim data and the real data, these initial findings provide an indication that the BKT model is a model with a very strong hold in reality.

6. REFERENCES

[1] R. S. Baker and K. Yacef, "The state of educational data mining in 2009: A review and future visions," *J. Educ. Data Min.*, vol. 1, no. 1, pp. 3–17, 2009.

[2] M. C. Desmarais and I. Pelczer, "On the Faithfulness of Simulated Student Performance Data.," in *EDM*, 2010, pp. 21–30.

[3] J. E. Beck and K. Chang, "Identifiability: A fundamental problem of student modeling," in *User Modeling 2007*, Springer, 2007, pp. 137–146.

[4] Z. A. Pardos and M. V. Yudelson, "Towards Moment of Learning Accuracy," in *AIED 2013 Workshops Proceedings Volume 4*, 2013, p. 3.

[5] Z. A. Pardos and N. T. Heffernan, "Navigating the parameter space of Bayesian Knowledge Tracing models: Visualizations of the convergence of the Expectation Maximization algorithm.," in *EDM*, 2010, pp. 161–170.

[6] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User Model. User-Adapt. Interact.*, vol. 4, no. 4, pp. 253–278, 1994.

[7] S. Ritter, T. K. Harris, T. Nixon, D. Dickison, R. C. Murray, and B. Towle, "Reducing the Knowledge Tracing Space.," *Int. Work. Group Educ. Data Min.*, 2009.

[8] R. S. d Baker, A. T. Corbett, S. M. Gowda, A. Z. Wagner, B. A. MacLaren, L. R. Kauffman, A. P. Mitchell, and S. Giguere, "Contextual slip and prediction of student performance after use of an intelligent tutor," in *User Modeling, Adaptation, and Personalization*, Springer, 2010, pp. 52–63.

[9] R. S. Baker, A. T. Corbett, and V. Aleven, "More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing," in *Intelligent Tutoring Systems*, 2008, pp. 406–415.

[10] Z. A. Pardos and N. T. Heffernan, "Modeling individualization in a bayesian networks implementation of knowledge tracing," in *User Modeling, Adaptation, and Personalization*, Springer, 2010, pp. 255–266.

[11] Z. A. Pardos and M. J. Johnson, "Scaling Cognitive Modeling to Massive Open Environments (in preparation)," *TOCHI Spec. Issue Learn. Scale*.

[12] U. Wilensky, "GasLab—an Extensible Modeling Toolkit for Connecting Micro-and Macro-properties of Gases," in *Modeling and simulation in science and mathematics education*, Springer, 1999, pp. 151–178.

The Effect of Variations of Prior on Knowledge Tracing

Matti Nelimarkka
School of Information, UC Berkeley
102 South Hall
Berkeley, California 94720-4600
Helsinki Institute for Information Technology HIIT,
Aalto University
PO Box 15600
Aalto, Finland 00076
matti.nelimarkka@hiit.fi

Madeeha Ghori
Department of Electrical Engineering and
Computer Sciences, UC Berkeley
387 Soda Hall
Berkeley, California 94720-17761
madeeha.ghori@berkeley.edu

ABSTRACT
Knowledge tracing is a method which enables approximation of a student's knowledge state using a Bayesian network for approximation. As the applications of this method increase, it is vital to understand the limits of this approximation. We are interested how well knowledge tracing performs when students' prior knowledge on the topic is extremely high or low. Our results indicate that the estimates become more erroneous when prior knowledge is extremely high ($prior = 0.90$).

Keywords
bayesian knowledge tracing, personalization, prior, parameter estimation

1. INTRODUCTION
The Bayesian Knowledge-Tracing (BKT) algorithm was developed in 1995 in an effort to model students' changing knowledge state during skill acquisition [5]. The idea is to interpret students' knowledge – a hidden variable – based on observed answers to a set of questions. The algorithm tracks the change in this probability distribution over time using a simple Bayes' net. The model is often presented as four parameters: prior, learn, guess and slip (see Figure 1). *Prior* refers to the probability that the student knows the material initially, before acquiring any skills, *learn* indicates that the student did not have the skill initially but acquired it through doing the exercise, *guess* refers to accidentally answering the question correct and *slip* to answering accidentally wrong.

Knowledge tracing is the most prominent method used to model student knowledge acquisition and is used in most intelligent learning systems. These systems have been said to be outperforming humans since 2001 [3] and have been used in the real world to tutor students [4]. For these reasons it is important to fully understand the strengths and limitations of knowledge tracing before applying it more widely in the classroom. As the parameters of the model are now known, there is a need to estimate these parameters from the given data. Previous research has demonstrated that the accuracy of parameter estimation – and therefore knowledge tracing – can be improved by applying different heuristics [17, 13] or methods [16, 18] including personalizing the model for each user [20, 8] or by extending the data used for analysis [15, 6, 1].

Our work starts from a different premise: how robust is the BKT approach to variation in the parameter space? Our special interest is in the *prior* variable, which correlates to a student's knowledge of the topic before answering a question. In any classroom, MOOC or otherwise, some students will come in with a better understanding of the material than others. Therefore it is important to study the effectiveness of knowledge tracing on parameter estimation when prior is extremely high or low.

If knowledge tracing models are inaccurate in modelling students of a certain prior parameter, then smart tutors and other systems designed to help those students learn will be less effective. Especially if the students being modelled inaccurately are those students doing poorly in the class, as the smart tutors exist to help them the most.

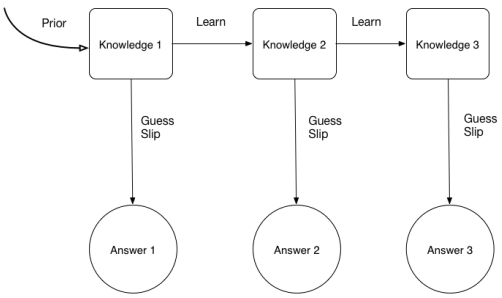


Figure 1: The model of knowledge tracing

Published in CEUR-WS:
BKT20y workshop (Yudelson, González-Brenes and Mozer)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

2. PREVIOUS WORK

For the purposes of this work, here we shortly summarize three methods previously applied to improve the prediction capabilities of BKT models. However, these methods are insufficient to address the practical problem described above, resulting in a need for our own experiment.

2.1 Individualization

Yudelson et al. [20] experimented with individualization by bringing student-specific parameters into the BKT algorithm on a larger scale. They split the usual skill-specific BKT parameters into two components: one skill-specific and one student-specific. They then built several individualized BKT models and added student-specific parameters in batches, examining the effect each addition had on the model's performance. They found that student-specific prior parameters did not provide a vast improvement. However, student-specific learning provided a significant improvement to the model's prediction accuracy.

Pardos and Heffernan furthered the experiment by developing a method of formulating the individualization within the Bayes' Net framework [11]. Especially interesting in terms of our work is the difference prior values and methods suggested for this individualization. Pardos observes that models taking student spesific priors based on students' prior knowledge clearly outperform traditional knowledge trace approach. This is a contrast Yudelson et al.'s findings [20] but it still underscores the importance of individualization in the BKT algorithm.

Related to individualization per user, there have been discussion on using different values per resources. It can be argued that different exercises teach different topics [7, 14]. This can be further used to individualize the model for different topics, an approach which has gained initial support on empirical studies [14].

2.2 Enhancing the data

The second approach to improve these methods is related to enhanching the data used for prediction. In its most simple form, this can be done by adding additional relevant data, such as data from past years, to the analysis [15]. Others have explored the possibility of adding more data to the general domain-related knowledge on the models, and suggest that these indeed improve the estimates [6].

However, the current direction in enhanced data relates to information available on user interaction – especially in MOOC environments where it is possible to access this kind of data. To illustrate, Baker, Corbett, and Aleven [1] explore interactions with the learning system and other non-exercise related data, such as time spent on answering and asking help, to determine the difference between slips and guesses.

We applaud these efforts and acknowledge that data other than just student responses may indeed help to detect both the cases where initial knowledge (prior) is high and when it is low, instead of tweaking the EM algorithm further.

2.3 Improving the methods

There are several heuristics currently used to enhance the BKT algorithm. One such heuristic involves expecting the sum of slip and guess to be less than or equal to 1 [17]. Other work determined that one's starting estimated parameters could affect where the algorithm converged to. In order to improve the accuracy of the convergence, it was suggested that starting parameters be selected from a Dirichlet distrbution derived from the data set [2, 13].

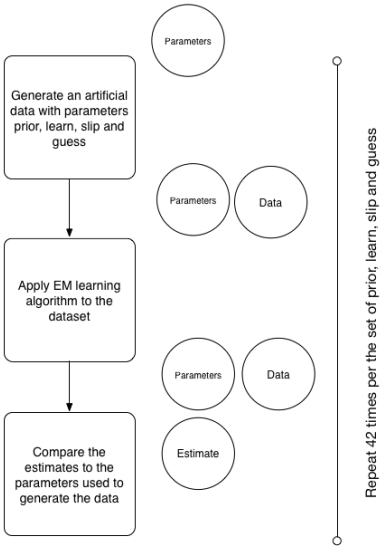


Figure 2: The approach used in this study

There have also been efforts to explore other machine learning methods on educational data. Initial trials born in the KDDCup competition use a medley of random forests and other machine learning algorithms but these methods have proven largely unsuccessful [16, 18].

The knowledge tracing community, while accepting the validity of some of these heuristics [9, 12], has criticized their inability to provide any insight into the student learning model. Individualization, however, has the potential to improve the BKT algorithm while also providing a pedagogical explanation for said improvements.

3. METHODOLOGY

We began by generating datasets with specific known initial parameters in order to simulate groups of students at different knowledge levels. We then ran expectation maximization (EM) on these datasets and allowed knowledge tracing to calculate its own estimated parameters. We then compared these estimated parameters to the original ones used for generation to determine if the accurency of the parameter estimation depends on the initial parameters.

Table 1: Ground Truth Parameter Sets

	prior	learn	guess	slip
Set 1.1 ... 1.6	0.15	0.10	0.10	0.05
Set 2.1 ... 2.6	0.30	0.10	0.10	0.05
Set 3.1 ... 3.6	0.15	0.20	0.10	0.05
	⋮			
Set 48.1 ... 48.6	0.90	0.20	0.20	0.10

3.1 Generating the Data

As our goal was to determine how the prior ground truth affects parameter estimation, we varied the prior used to synthesize the data sets. We used six different priors $(0.15, 0.30, \ldots, 0.75, 0.9)$, and two variations on learn, slip and guess[1] each (see Table 1); total of 48 variations of these parameters. Each of these data sets consists of 10,000 students and 20 observations per student. To increase the variation, we generated 6 datasets per condition. This kind of simulated approach has been previously used to evaluate the success of Bayesian machine learning methods [8].

3.2 Analysis Procedure

For each data set, we estimated the parameters using the *expectation maximization fitting* (EM) algorithm using the fastHMM implementation [10]. The parameter estimation was conducted using a grid search with ten parameters, and the best fitting model was selected using the log likelihood.

Using our 288 data sets, we can compare the estimates and ground truths for each parameter and analyze the accuracy of the estimates. We apply the standard methods of root-mean-square error (RMSE) and other visualizations to do our analysis. Using RMSE, we will be able to see if certain ground truths lend themselves to more accurate estimations.

4. RESULTS

First, let us explore the parameter estimation in detail. The avarage RMSE measurement in the data (Table 2) indicate that the prediction quality decreases as the prior increases; there is also increase of variance of the RMSE. This indicates that the predictions with higher priors are first more erronous and second, they converge in a larger area, resulting in variance. To confirm our observations, we conducted a Wilcox-Mann-Whitney test to explore if the computed RMSEs differented in statistically significant manner. As shown in Table 3, both the RMSEs computed from the data sets with priors 0.15 and 0.90 statistically differ significantly from the other datasets ($p < 0.05$). Therefore we conclude that the EM algortihm performs badly when prior is high.

To further understand this phenomena, we explore the estimates per parameter. The errors per parameter are shown in the Figure 3. The mean estimates are rather constantly close by the zero, though a higher prior does affect variance. As ground truth prior increases, the variance of guess and learn increases while the variance of prior decreases. In theory, a lesser variance on the prior prediction should imply a more accurate prior estimate. However, as we saw in Table 2, this is not actually the case. The prior estimate gets less accurate as the value of the ground truth prior increases. In Figure 3 we can see again some of the results we saw in Table 2: the prediction accuracy decreases when prior is 0.6 and continues to decrease as prior increases.

[1]Variations were 0.10 and 0.20 for learn and guess, and 0.05, 0.10 for slip.

Ground truth prior	mean RMSE	var RMSE
0.15	0.056639	0.000594
0.30	0.069073	0.001137
0.45	0.070005	0.000584
0.60	0.074044	0.001874
0.75	0.075946	0.002229
0.90	0.085257	0.004876

Table 2: The mean and variance of the root-mean-square errors per prior

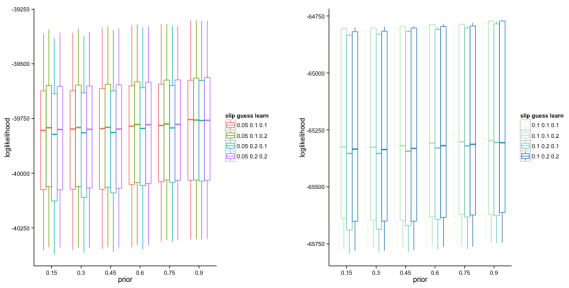


Figure 4: Log likelihoods with different parameters

Figure 4 shows that the log likelihood for each of the parameter combinations we analyzed. We see a slight, but non-significant increase in the log likelihoods, suggesting that the model is performing better – even while our RMSE error indicator demonstrates otherwise. It is also noteworthy to observe that that when slip is 0.10, all log likelihoods range between -65500 and -65250 but when slip is 0.05, all log likelihoods range between -40000 and -35750, indicating that the slip value had a dramatic effect on the model estimation accurancy.

5. IMPLICATIONS

Our findings indicate that there are higher errors in the parameter estimations when prior is high (0.90). This is probably due to the lack of evidence available for the HMM to attribute to the learn and guess parameters. One approach to examine the impact of these errors is to examine the students' subjective experience in different conditions [19]. As our data is syntetic, we can not measure the time consumed by students due to errors, as examined by Youdelson & Koedinger [19]. Instead we explore the difference on the number of questions students' need to answer to achieve mastery learning – for our purposes knowledge above 95 % and assuming that the students answer each question correctly.

Examining the case of high prior knowledge, and when the true learning was 0.1, we observed that majority of students needed to answer over 5 times to achieve mastery (or: from the 168 predicted value sets available, only 24 achieved mastery), and for the high learning (0.2) the situation was not

Table 3: Significant differences between the RMSEs

	0.15	0.30	0.45	0.60	0.75	0.90
0.15	1	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001
0.30		1	0.347	0.614	0.967	0.014
0.45			1	0.660	0.125	0.081
0.60				1	0.744	0.035
0.75					1	0.007
0.90						1

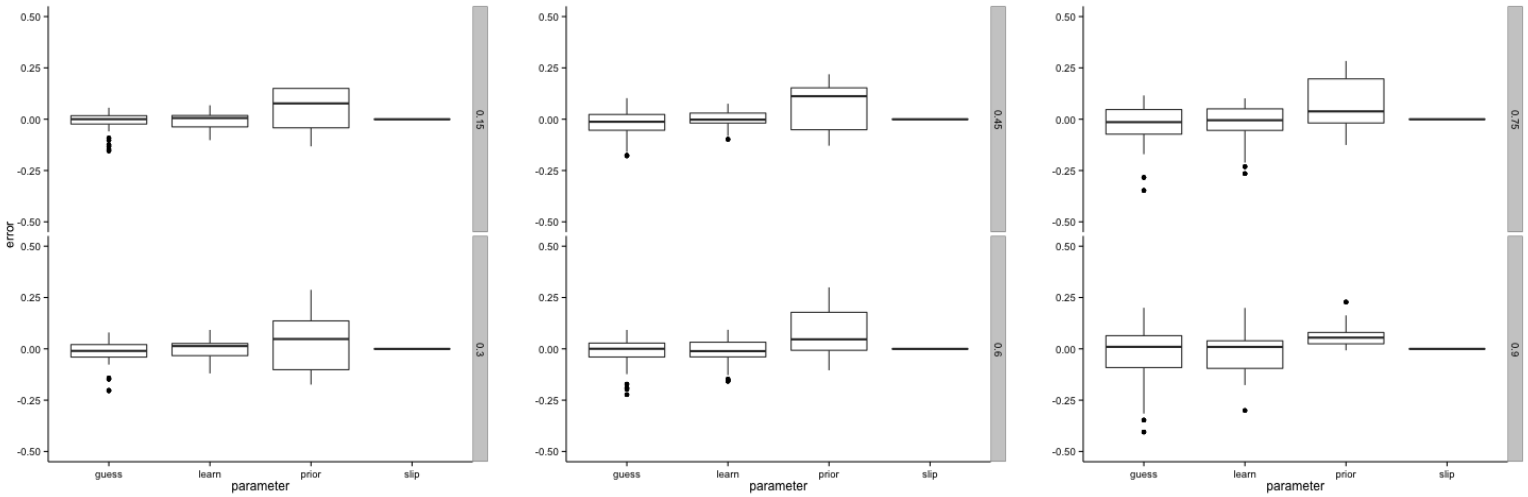


Figure 3: Predicting parameters with different values of prior

significantly better – there 56 values achieved mastery with 5 responses. This indicates that the impact indeed was significant in terms of impact to students learning and highlights the importance of this study.

6. CONCLUSIONS

We started this study with the motivation to explore how well the knowledge tracing method performs when the prior is high or low; this performance has practical implications when applying this approach in a heterogenius classroom where students arrive with highly different knowledge of the domain. We studied this empirically by generating 288 different synthetic datasets and explored the difference between the predicted parameters and the parameters used to generate the dataset.

Our results indicated a slightly increased in the estimation error when prior was 0.90, which we mostly attribute to higher error in learn and guess parameters. This observation was statistically significant and most likely due to the fact that students with higher priors produce less information to be used by the HMM to estimate the guess and learn parameters.

We explored the influence these errors had on the propability of knowledge and observed that these errors significantly reduced the speed students achieved mastery learning. This result therefore implies that more work needs to be done to detect those with high prior knowledge to cater their learning needs.

Acknowledgments

This work was conducted during UC Berkeley School of Information class “INFO290: Machine learning in education“ instructed by Zach Pardos. We thank the support of the course staff and peers on the presentation.

References

[1] RyanS.J.d. Baker, AlbertT. Corbett, and Vincent Aleven. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In BeverleyP. Woolf, Esma AÃŕmeur, Roger Nkambou, and Susanne Lajoie, editors, *Intelligent Tutoring Systems*, volume 5091 of *Lecture Notes in Computer Science*, pages 406–415. Springer Berlin Heidelberg, 2008.

[2] Joseph E Beck and Kai-min Chang. Identifiability : A Fundamental Problem of Student Modeling. pages 137–146, 2007. doi: 10.1007/978-3-540-73078-1_17.

[3] Albert Corbett. Cognitive computer tutors: Solving the two-sigma problem. In *User Modeling 2001*, volume 2109 of *Lecture Notes in Computer Science*, pages 137–147. Springer Berlin Heidelberg, 2001.

[4] Albert Corbett, Megan McLaughlin, and K Christine Scarpinatto. Modeling student knowledge: Cognitive tutors in high school and college. *User modeling and user-adapted interaction*, 10(2-3):81–108, 2000.

[5] Albert T Corbett and John R Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4): 253–278, 1994.

[6] Albert T Corbett and Akshat Bhatnagar. Student modeling in the act programming tutor: Adjusting a procedural learning model with declarative knowledge. *COURSES AND LECTURES-INTERNATIONAL CENTRE FOR MECHANICAL SCIENCES*, pages 243–254, 1997.

[7] Tanja KÃďser, Severin Klingler, AlexanderGerhard Schwing, and Markus Gross. Beyond knowledge tracing: Modeling skill topologies with bayesian networks. In Stefan Trausan-Matu, KristyElizabeth Boyer, Martha Crosby, and Kitty Panourgia, editors, *Intelligent Tutoring Systems*, volume 8474 of *Lecture Notes in Computer Science*, pages 188–198. Springer International Publishing, 2014.

[8] Z. A. Pardos and N. T. Heffernan. Navigating the parameter space of Bayesian Knowledge Tracing models Visualizations of the convergence of the Expectation

Maximization algorithm. *In Proceedings of the 3rd International Conference on Educational Data Mining*, 2010.

[9] ZA Pardos and NT Heffernan. Using HMMs and bagged decision trees to leverage rich features of user and skill from an intelligent tutoring system dataset. *Journal of Machine Learning Research W & CP*, 2010. URL http://people.csail.mit.edu/zp/papers/pardos_JMLR_in_press.pdf.

[10] Z.A. Pardos, M.J. Johnson, and et al. Scaling cognitive modeling to massive open environments. *TOCHI Special Issue on Learning at Scale*, (in preparation).

[11] ZacharyA. Pardos and Neil T. Heffernan. Modeling individualization in a bayesian networks implementation of knowledge tracing. In Paul Bra, Alfred Kobsa, and David Chin, editors, *User Modeling, Adaptation, and Personalization*, volume 6075 of *Lecture Notes in Computer Science*, pages 255–266. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-13469-2.

[12] Pardos, Zachary A, Sujith M. Gowda, Ryan S.J.d. Baker, and Neil T. Heffernan. The sum is greater than the parts. *ACM SIGKDD Explorations Newsletter*, 13(2):37, May 2012. ISSN 19310145. doi: 10.1145/2207243.2207249. URL http://dl.acm.org/citation.cfm?id=2207249 http://dl.acm.org/citation.cfm?doid=2207243.2207249.

[13] Dovan Rai, Yue Gong, and Joseph E Beck. Using dirichlet priors to improve model parameter plausibility. *International Working Group on Educational Data Mining*, 2009.

[14] Leena Razzaq, Neil T Heffernan, Mingyu Feng, and Zachary A Pardos. Developing Fine-Grained Transfer Models in the ASSISTment System. *Technology, Instruction, Cognition & Learning*, 5(3):1–16, 2007.

[15] Steven Ritter, Thomas K Harris, Tristan Nixon, Daniel Dickison, R Charles Murray, and Brendon Towle. Reducing the knowledge tracing space. *International Working Group on Educational Data Mining*, 2009.

[16] A Toscher and Michael Jahrer. Collaborative filtering applied to educational data mining. *Journal of Machine Learning Research*, 2010.

[17] Brett van De Sande. Properties of the Bayesian Knowledge Tracing Model. *Journal of Educational Data Mining*, 5(2):1–10, 2013.

[18] Hsiang-Fu Yu, Hung-Yi Lo, Hsun-Ping Hsieh, Jing-Kai Lou, Todd G McKenzie, Jung-Wei Chou, Po-Han Chung, Chia-Hua Ho, Chun-Fu Chang, Yin-Hsuan Wei, et al. Feature engineering and classifier ensemble for kdd cup 2010. *JMLR: Workshop and Conference Proceedings*, 1, 2010.

[19] Michael V Yudelson and Kenneth R Koedinger. Estimating the benefits of student model improvements on a substantive scale. In *Proceedings of the 6th International Conference on Educational Data Mining*, 2013.

[20] Michael V Yudelson, Kenneth R Koedinger, and Geoffrey J Gordon. Individualized bayesian knowledge tracing models. In *Artificial Intelligence in Education*, pages 171–180. Springer, 2013.

A Brief Overview of Metrics for Evaluation of Student Models

Radek Pelánek
Masaryk University Brno
pelanek@fi.muni.cz

ABSTRACT

Many different metrics are used to evaluate and compare performance of student models. The aim of this paper is to provide an overview of commonly used metrics, to discuss properties, advantages, and disadvantages of different metrics, and to summarize current practice in research papers. The paper should serve as a starting point for workshop discussion about the use of metrics in student modeling.

1. INTRODUCTION

A key part of intelligent tutoring systems are models that estimate the knowledge of students. To compare and improve these models we use metrics that measure quality of model predictions. Metrics are also used (sometimes implicitly) for parameter fitting, since many fitting procedures try to optimize parameters with respect to some metric.

At the moment there is no standard metric for model evaluation and thus researchers have to decide which metric to use. The choice of metric is an important step in the research process. Differences in predictions between competing models are often small and the choice of metric can influence the results more than the choice of a parameter fitting procedure. Moreover, fitted model parameters are often used in subsequent steps in educational data mining and thus the choice of metric can indirectly influence many other aspects of the research.

However, despite the fact that the choice of metric is important and that there is no clear consensus on the usage of performance metrics, the topic gets very little attention in most research papers. Most authors do not provide any rationale for their choice of metric. Sometimes it is not even clear what metric is exactly used, so it may be even difficult to use the same metric as previous authors. The main aim of this paper is to give an overview of performance metrics relevant for evaluation of student models and to explicitly discuss points that are in most papers omitted.

2. OVERVIEW OF METRICS

To attain clear focus we discuss only models that predict probability of a correct answer. We assume that we have data about n answers, numbered $i \in \{1, \ldots, n\}$, correctness of answers is given by $c_i \in \{0, 1\}$, a student models provides predictions $p_i \in [0, 1]$. A model performance metric is a function $f(\vec{p}, \vec{c})$. Note that the word "metric" is here used in a sense "any function that is used to make comparisons", not in the mathematical sense of a distance function. Since we are interested in using the metrics for comparison, monotone transformations (square root, logarithm, multiplication by constant) are inconsequential and are used mainly for better interpretability (or sometimes rather for traditional reasons).

2.1 Mean Absolute Error

This basic metric consider the absolute differences between predictions and answers: $MAE = \frac{1}{n} \sum_{i=1}^{n} |c_i - p_i|$. This is not a suitable performance metric, because it prefers models which are biased towards the majority results. As a simple illustration, consider a simulated student which answers correctly with constant probability 0.7. If we compare different constant predictors with respect to this metric, we get that the best model is the one which predicts probability of correct answer to be 1. This is clearly not a desirable result. As this example illustrates, the use of MAE can lead to rather misleading conclusions. Despite this clear disadvantage, MAE is sometimes used for evaluation (although mostly in combination with other metrics, which reduces the risk of misleading conclusions in published papers).

2.2 Root Mean Square Error

A similar metric is obtained by using squared values instead of absolute values: $RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (c_i - p_i)^2}$. Note that from the perspective of model comparison, the important part is only the sum of square errors (SSE). The square root in RMSE is traditionally used to get the result in the same units of as the original "measurements" and thus to improve interpretability of the resulting number. In the particular context of student modeling and evaluation of probabilities, this is not particularly useful, since the resulting numbers are hard to interpret anyway. In order to get better interpretability researchers sometimes use R^2 metric: $R^2 = 1 - \sum_{i=1}^{n} (c_i - p_i)^2 / \sum_{i=1}^{n} (c_i - \bar{c})^2$. With respect to comparison of models, R^2 is equivalent to RMSE since here again the only model dependent part is the sum of square errors. In the context of the standard linear regression (where it is

Published in CEUR-WS:
BKT20y workshop (Yudelson, González-Brenes and Mozer)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

most commonly used) R^2 has a nice interpretation as "explained variability". In the case of logistic regression (which is more similar to student models) this interpretation does not hold and different "pseudo R^2" metrics are used (e.g., Cox and Snell, McFadden, Nagelkerke). Thus a disadvantage of R^2 is that unless the authors are explicit about which version of R^2 they use (usually they are not), a reader cannot know for sure which metric is reported.

In educational data mining the use of RMSE metric is very common (it was also used as a metric in KDD Cup 2010 focused on student performance evaluation). In other areas, particularly in meteorology, mean square error (RMSE without the square root) is called the Brier score [1]. The Brier score is often decomposed into additive components (e.g., reliability and refinement) which provide further insight into the behaviour of the predictor. Moreover, in an analogy to AUC metric and ROC curve (described below), this metric can be interpreted as area under Brier curves. These methods may provide interesting inspirations for student modeling.

2.3 Metrics Based on Likelihood

The likelihood of data (the answers) given a model (predicted probabilities) is $L = \prod_{i=1}^{n} p_i^{c_i} \cdot (1-p_i)^{(1-c_i)}$. Since we are indifferent to monotonic transformations we typically work with the numerically more stable logarithm of the likelihood $LL = \sum_{i=1}^{n} c_i \log(p_i) + (1-c_i)\log(1-p_i)$. This metric can also be interpreted from information theoretic perspective as measure of data compression provided by a model [4]. The log-likelihood metric can be further extended into metrics like Akaike information criterion (AIC) and Bayesian information criterion (BIC). These metrics penalize large number of model parameters and thus aim to avoid overfitting. In the context of student modeling it is typically much better to address the issue of overfitting by cross-validation. Since AIC and BIC provide a faster way to assess models than cross-validation, they may be useful as heuristics in some algorithms (e.g., learning factor analysis), but they are not serious contenders for proper model comparison.

MAE, RMSE and LL have all the form of "sum of penalties for individual errors" and differ only in the function which specifies the penalty. For RMSE and LL values of penalty functions are quite similar, the main difference is in the interval [0.95, 1], i.e., in cases where the predictor is confident and wrong. These cases are penalized very prohibitively by LL, whereas RMSE is relatively benevolent. In fact the LL metric is unbounded, so single wrong prediction (if it is too confident) can ruin the performance of a model. This property is usually undesirable and an artificial bound is used. This corresponds to basically forcing a possibility of a slip and guess behaviour into a model. After this modification the penalties for RMSE and LL are rather similar. Nevertheless, the LL approach "penalize mainly predictions which are confident and wrong" is reasonable thus it is rather surprising that this metric is used only marginally in evaluation of student models (it is used mostly in connection with AIC or BIC).

2.4 Area Under an ROC Curve

Another popular metric is based on the receiver operating characteristics (ROC) curve. If we want to classify predictions into just two discrete classes (correct, incorrect), we need to select a threshold for the classification. For a fixed threshold we can compute standard metrics like precision, recall, and accuracy. If we do not want to use a fixed threshold, we can use the ROC curve, which summarises the behaviour of the prediction model over all possible thresholds. The curve has "false positive rate" on x-axis and "true positive rate" on the y-axis, each point of the curve corresponds to a choice of a threshold. Area under the ROC curve (AUC) provides a summary performance measure across all possible thresholds. It is equal to the probability that a randomly selected correct answer has higher predicted score than a randomly selected incorrect answer. The area under the curve can be approximated using a A' metric, which is equivalent to the well-studied Wilcoxon statistics [2]. This connection provides ways to study statistical significance of results (but requires attention to assumptions of the tests, e.g., independence).

The ROC curve and AUC metric are successfully used in many different research areas, but their use is sometimes also criticised [3], e.g., because the metric summarises performance over all possible thresholds, even over those for which the classifier would never be used in practice. From the perspective of student modeling the main reservation seems to be that this approach focuses on classification and considers predictions only in relative way – note that if all predictions are divided by 2, the AUC metric stays the same.

In the context of student modeling we are usually not interested in classification, we are often interested directly in absolute values of probabilities and we need these values to be properly calibrated. The probabilities are often compared to a fixed constant (typically 0.95) as an indication of a mastered skill and the specific value is meant to carry a certain meaning. Probabilistic estimates can be also used to guide the behaviour of a system to achieve suitable challenge for students, e.g., by choosing question of right difficulty or modifying difficulty by number of options in multiple choice questions.

Nevertheless, despite this disadvantage, AUC is widely used for evaluation of student models, often as the only metric. It seems that in some cases AUC is used as the only metric for final evaluation, but the parameter fitting procedure uses (implicitly) different metric (RMSE or LL). Particularly in cases of brute force fitting this approach seems strange and should be at least explicitly mentioned.

3. REFERENCES

[1] G. W. Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[2] J. Fogarty, R. S. Baker, and S. E. Hudson. Case studies in the use of ROC curve analysis for sensor-based estimates in human computer interaction. In *Proc. of Graphics Interface 2005*, pages 129–136, 2005.

[3] J. M. Lobo, A. Jiménez-Valverde, and R. Real. AUC: a misleading measure of the performance of predictive distribution models. *Global ecology and Biogeography*, 17(2):145–151, 2008.

[4] M. S. Roulston and L. A. Smith. Evaluating probabilistic forecasts using information theory. *Monthly Weather Review*, 130(6), 2002.

A Comparison of Error Metrics for Learning Model Parameters in Bayesian Knowledge Tracing *

Asif Dhanani† Seung Yeon Lee† Phitchaya Mangpo Phothilimthana† Zachary Pardos
University of California, Berkeley
{asifdhanani, sy.lee, mangpo, pardos}@berkeley.edu

ABSTRACT

In the knowledge-tracing model, error metrics are used to guide parameter estimation towards values that accurately represent students' dynamic cognitive state. We compare several metrics, including log-likelihood (LL), RMSE, and AUC, to evaluate which metric is most suited for this purpose. In order to examine the effectiveness of using each metric, we measure the correlations between the values calculated by each and the distances from the corresponding points to the ground truth. Additionally, we examine how each metric compares to the others. Our findings show that RMSE is significantly better than LL and AUC. With more knowledge of effective error metrics for learning parameters in the knowledge-tracing model, we hope that better parameter searching algorithms can be created.

1. INTRODUCTION

In Bayesian Knowledge Tracing (BKT), one of the essential elements is the error metric that is used for learning model parameters: prior, learn, guess, and slip. Choice of a type of error metric is crucial because the error metric takes a role of guiding the search to the best parameters. The BKT model can be fit to student performance data by using a method which finds a best value calculated from the error metric such as log-likelihood (LL), root-mean-squared error (RMSE), or area under the ROC curve (AUC).

As a modeling method, grid search/brute force [1] is often used to find the set of parameters with optimal values of the error metric, and Expectation Maximization (EM) algorithm [5] is also commonly used to choose parameters maximizing the LL fit to the data. Many studies have compared different modeling approaches [1, 4]. However, the findings are varied across the studies, and it has still been unclear which method is the best at predicting student performance [2].

Pardos and Yudelson compares different error metrics to investigate which one has the most accuracy of estimating the moment of learning [6]. Our work extends this comparison by looking closer into the relationship between three popular error metrics: LL, RMSE, and AUC, and particularly elucidating the relationship to one another closer to the ground truth point.

*For more details of this work, please refer to the full technical report [3].

†Asif Dhanani, Seung Yeon Lee, and Phitchaya Mangpo Phothilimthana contributed equally to this work and are listed alphabetically.

2. METHODOLOGY

To assess whether LL, RMSE, or AUC is the best error metric to use in parameter searching for the BKT model, we needed datasets with known parameter values in order to compare these with the parameter values predicted by using different error metrics. Therefore, we synthesized 26 datasets by simulating student responses based on diverse known ground truth parameter values.

Correlations to the ground truth. For each dataset, we evaluated LL, RMSE, and AUC values on all points over the entire prior/learn/guess/slip parameter space with a 0.05 interval. On each point, we calculated students' predicted responses (probability that students will answer questions correctly). We then used these predicted responses with the actual responses to calculate LL, RMSE, and AUC for all points. To determine which error metric is the best for this purpose, we looked at the correlations between values calculated from error metrics (i.e. LL, RMSE, and AUC) and the euclidean distances from the points to the ground truth. We applied logarithm to all error metrics other than LL in order to compare everything on the same scale. Finally, we tested whether the correlation between the values calculated by any particular error metric and the distances is significantly stronger than the others' by running one-tailed paired t-tests comparing all three metrics against one another.

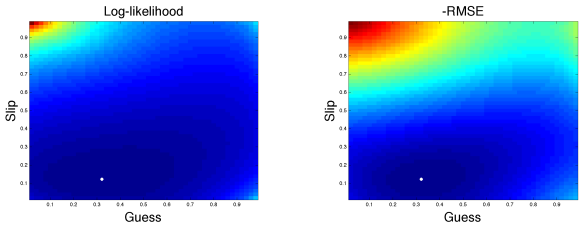
Distributions of values. We visualized the values of LL and -RMSE of all points over the 2 dimensional guess/slip space with a 0.02 interval while fixing prior and learn parameter values to the actual ground truth values. Using the guess and slip parameters as the axes, we visualize LL and -RMSE values by color. The colors range from dark red to dark blue corresponding to the values ranging from low to high.

Direct comparison: LL and RMSE. We plotted LL values and RMSE values of all points against each other in order to observe the behavior of the two metrics in detail. We then labeled each data point by its distance to the ground truth with a color. The range of colors is the same as used in the previous method.

Published in CEUR-WS:
BKT20y workshop (Yudelson, González-Brenes and Mozer)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

Comparision	Δ of correlations	t	p-value
RMSE > LL	0.0408	8.9900	<< 0.0001
RMSE > AUC	0.0844	2.7583	0.0054
LL > AUC	0.0436	1.4511	0.0796

Figure 1: T-test statistics



(a) LL Heatmap (b) -RMSE Heatmap

Figure 2: LL and -RMSE values when fixing prior and learn parameter values and varying guess and slip parameter values. Red represents low values, while blue represents high values. The white dots represent the ground truth.

3. RESULTS

Correlations to the ground truth. The average LL, RMSE, and AUC correlations were 0.4419, 0.4827, and 0.3983 respectively. We define that an error metric A is *better* than B if the correlation between values calculated by an error metric A and the distances to the ground truth is higher than that of B. By this definition, RMSE was better than LL on all 26 datasets and better than AUC on 18 of 26 datasets. This is validated by the one-tailed paired t-test shown in Figure 1 revealing RMSE as statistically significantly better than both LL and AUC.

Distributions of values. Figure 2 shows the heat maps of LL and RMSE on a representative dataset. If we follow the gradient from the lowest value to the highest value in the LL heat map, we see that it is very high at the beginning (far from the ground truth) and is very low at the end (close to the ground truth). Conversely, in the -RMSE heat map, the change in the gradient is low. Additionally, notice that the darkest blue region in -RMSE heat map is smaller than that in LL heat map. This suggests that we may be able to refine the proximity of the ground truth better with RMSE.

Direct comparison: LL and RMSE. Figure 3 shows a LL vs -RMSE graph from the most representative dataset. As expected, LL values and RMSE values correlate logarithmically. Additionally, a secondary curve, which we will refer to as the *hook*, is observed in varying sizes among datasets. The hook converges with the main curve when the -RMSE and LL values are both sufficiently high and the points are very close to the ground truth.

Before this point, when we look at a fixed LL value with varied RMSE values, most points in the hook have higher -RMSE values and are closer to the ground truth than do the points in the main curve. However, this same pattern is not seen for a fixed RMSE value with varied LL values. After the curve and hook converge, we can infer that both RMSE and LL will give similar estimates of the ground truth. However, for a portion of the graph before this point, RMSE is a better predictor of ground truth values.

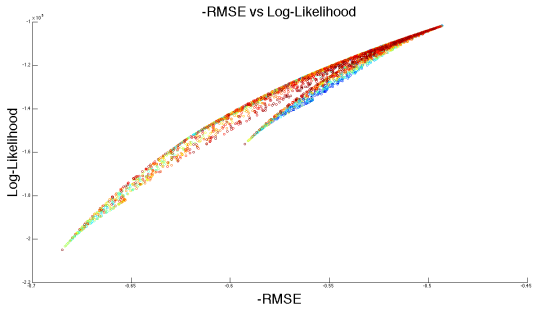


Figure 3: LL vs -RMSE of dataset 25 when prior = 0.564, learn = 0.8, guess = 0.35 , and slip = 0.4

4. CONCLUSION

In our comparison of LL, RMSE, and AUC as metrics for evaluating the closeness of estimated parameters to the true parameters in the knowledge tracing model, we discovered that RMSE serves as the strongest indicator. RMSE has a significantly higher correlation to the distance from the ground truth on average than both LL and AUC, and RMSE is notably better when the estimated parameter value is not very close to the ground truth. The effectiveness of teaching systems without human supervision relies on the ability of the systems to predict the implicit knowledge states of students. We hope that our work can help advance the parameter learning algorithms used in the knowledge tracing model, which in turn can make these teaching systems more effective.

5. REFERENCES

[1] R. Baker, A. Corbett, S. Gowda, A. Wagner, B. MacLaren, L. Kauffman, A. Mitchell, and S. Giguere. Contextual slip and prediction of student performance after use of an intelligent tutor. In *User Modeling, Adaptation, and Personalization*, volume 6075 of *Lecture Notes in Computer Science*. 2010.

[2] R. S. Baker, Z. A. Pardos, S. M. Gowda, B. B. Nooraei, and N. T. Heffernan. Ensembling predictions of student knowledge within intelligent tutoring systems. In *Proceedings of the 19th International Conference on User Modeling, Adaption, and Personalization*, 2011.

[3] A. Dhanani, S. Y. Lee, P. Phothilimthana, and Z. Pardos. A comparison of error metrics for learning model parameters in bayesian knowledge tracing. Technical Report UCB/EECS-2014-131, EECS Department, University of California, Berkeley, May 2014.

[4] Y. Gong, J. Beck, and N. Heffernan. Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. In *Intelligent Tutoring Systems*, volume 6094 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010.

[5] Z. Pardos and N. Heffernan. Modeling individualization in a bayesian networks implementation of knowledge tracing. In *User Modeling, Adaptation, and Personalization*. 2010.

[6] Z. A. Pardos and M. V. Yudelson. Towards moment of learning accuracy. In *Proceedings of the 1st AIED Workshop on Simulated Learners*, 2013.

Prediction of Student Success Using Enrolment Data

Nihat Cengiz
Epoka University
Department of Computer Engineering
Rr. Tiranë-Rinas,Km. 12
1039 Tirana, Albania
ncengiz@epoka.edu.al

Arban Uka
Epoka University
Department of Computer Engineering
Rr. Tiranë-Rinas,Km. 12
1039 Tirana, Albania
auka@epoka.edu.al

ABSTRACT

Predicting the success of students as a function of different predictors has been a topic that has been investigated over the years. This paper explores the socio-demographic variables like gender, region lived and studied, nationality and high school degree that may influence success of students. We examine to what extent these factors help us to predict students' academic achievement and will help to identify the vulnerable students and their need for extra tutoring or similar supportive services at an early time.

We analyzed the data of the Epoka University students that have been enrolled from 2007 to 2013. The sample includes 1211 undergraduate students where 716 did and were supposed to complete the three-year bachelor studies in the past six semesters.

Based on the data mining techniques the most important predictors for student success were the students' high school GPA and gender. For students with high school grades below average, females were found to have a higher percentage of success than boys. No significant correlation was found between the students' success and the demographic information.

Keywords

Academic achievement, influence, classification tree, outcome

1. INTRODUCTION

Increasing the student graduation and decreasing the dropout rates is a long term goal of the higher education institutions. From the students' perspective, a timely and successful graduation is vital as these two factors would strongly affect their employability rate. Employability rate has become an indicator in determining the ranking of higher education institution (HEI), thus HEIs are focusing more on increasing this rate [2].

Many of the students studying at the university face several difficulties during the first year and thus the performance of the first year has been identified as an important predictor of timely graduation rate. In terms of keeping the students in the university, the retention rate is a factor that has been studied extensively. Mallincrodt and Sedlacek (1987) found that freshman class attrition rate were greater than the other academic years with numbers running up to 30%.[3] Therefore most researchers targeted the first year students. An early identification of the students at high risk of failing will enable a timely intervention with the necessary measures by the educators that would increase the graduation rate. Preventing students' failure depends on the identification of the factors affecting success.

Here in this work we will analyze whether the background information has any effect on the success rate of regular students. The only data we collected during the registration period of Epoka University based on the registration form. The content of this form determined by the local authorities and University Administration. In this study we tried to get answers if we can use this data to predict student success. The main objective of our study is to determine the factors that may affect the study outcomes in Epoka University.

2. DATA AND METHODOLOGY

Epoka University student management system does not provide data in the format ready for a direct statistical analysis and modeling. Therefore a data preparation and cleaning were undertaken to prepare database for modeling.

Table Descriptive statistics – Study outcome (716 students)

Descriptive						
		count		%		
	Domain	FAIL	PASS	FAIL	PASS	Total
GENDER	M	221	189	53.9	46.1	57.3
	F	78	228	25.5	74.5	42.7
COUNTRY	ALB	238	372	39.0	61.0	85.2
	TUR	35	14	71.4	28.6	6.8
	KOS	14	17	45.2	54.8	4.3
	OTH	12	14	46.2	53.8	3.6
NATIONALITY	ALB	256	382	40.1	59.9	89.1
	OTH	43	35	55.1	44.9	10.9
REGION	CITY	262	372	41.3	58.7	88.5
	VILL.	37	44	45.7	54.3	11.3
HS_GPA	UPPER	48	224	17.6	82.4	38.0
	INTER.	89	113	44.1	55.9	28.2
	LOWER	160	77	67.5	32.5	33.1

2.1. Data and Methodology

Outcome that we used in our analysis is for the outcome of the student at the end of three-year study. We measured only outcomes, labeled as: Pass and Fail. Students labeled 'Pass' successfully completed the program at the end of three years. Students labeled as 'Fail' include the withdrawn students from the

program voluntarily or by the academic registry for not fulfilling the regulations. Those students who stayed on the program until the end of the study but scored less than the graduation grade (2.00) were also allocated into this category.

The data set with numeric continuous variable such as secondary school grade (HS GPA) was converted into a categorical variable with only three levels A (UPPER), B (INTERMEDIATE) or C (LOWER) denoting grades above 9 out of 10, grades between 8 and 9 and grades less than 8 respectively. Other variables (nationality, citizenship, and region) were classified upon major groups.

In this study we conducted three main types of data mining approaches. Descriptive approach which concerns the nature of the dataset such as the frequency table and the relationship between the attributes obtained using cross tabulation analysis. Predictive approach which is conducted by using four different classification trees and a comparison between these and Logistic regression to confirm the accuracy of the predictors.

Classification tree models can handle a large number of predictor variables, are non-parametric, can capture nonlinear relationships and complex interactions between predictors and dependent variable.[1]

Before generating the classification trees we classified the variables according to the study outcome, i.e. whether students are eligible to be graduated or not. We used attribute selection to rank the variables by their importance for further analysis. Then we generated the classification trees in four different growing methods.

2.2. Summary Data Description

We carried out a cross-tabulation for each variable and the study outcome after cleaning the data as shown in the table above. Table shows that the majority of the successful students are female (over 57%) which is the result of the fact that 74.5% of the female students successfully completed the study. This suggests that female students are more likely to succeed than their male classmates. In terms of country and nationality it is clearly seen that Albanian population is leading the group.

An expected result has been observed in secondary school degrees. We can say that high school degree graduation ratio is directly proportional to the university graduation ratio. While 82% of upper students were able to complete the study on time 56% of intermediate and 32% of lower group students were able to complete.

2.3. Decision Trees

Although the results of the attribute selection suggests continuing analysis with only the subset of predictors, we included all available predictors in our classification trees but only 2 variables were used in the diagrams: HS_GPA and GENDER. Even though some variables may have little significance to the overall prediction outcome, they can be essential to a specific record [1].

Almost all growing methods, (CHAID, exhaustive CHAID, CRT and QUEST) generated exactly the same trees. The largest successful group consists of 272 (38%) students. HS_GPA of this group is over 90%. The largest unsuccessful group contains 237 students (33% of all participants). They have a HS_GPA less than 80%. The next largest group considered also as unsuccessful students are male students having lower HS_GPA.

As the cross-validation estimate of the risk (0.309) indicates that the successful or unsuccessful students are predicted with an error of 30.9% of the cases which means the risk of misclassifying a student is approximately 31%. This result is consistent with the results in the CHAID classification matrix. The Overall percentage shows that the model only classified correctly 70% of students. The classification tables, however, reveal one potential problem with this model: for unsuccessful students, it predicts as successful for only 65.9% of them, which means that 34% of failing students are inaccurately classified with the passing students.

2.4. Logistic regression

The Variables not in the Equation table in block 0 shows that four of the five variables are individually significant predictors of whether a student is successful or not. Region is not a significant predictor. The variables not in the Equation table in block 1 shows that only high school grade point average and gender are significant predictors, but not the other variables. This result also confirms why these two were the only variables used in decision trees

3. CONCLUSIONS

This study examines the background information from enrolment data that impacts upon the study outcome programs at the Epoka University. Based on results, the classification accuracy from the classification trees was significantly high 71% in all tree methods. Although all the variables except the region individually significant predictors as described in attribute selection trees displayed only two variables Gender and secondary school degree. This outcome is also confirmed by the logistic regression. Block 0 classification implied that all except region were good predictors (p<,001) but block 1 classification highlighted that only gender and secondary school degree were significant.

4. REFERENCES

[1]. Kovačić, Z.J. 2010, Early Prediction of Student Success: Mining Students Enrolment Data, proceedings of Informing Science & IT Education Conference (InSITE) 2010, Open Polytechnic, Wellington, New Zealand

[2]. Bratti, M., McKnight, A., Naylor, R., & Smith, J. (2004): Higher Education Out-comes, Graduate Employment and University Performance Indicators. In: Journal of the Royal Statistical Society, 167(3), pp 475-496.

[3]. Mallinckrodt, B., & Sedlacek, W. E. (1987). Student retention and the use of campus facilities by race. NASPA Journal, 24, 28-32.

Expanding Knowledge Tracing to Prediction of Gaming Behaviors

Sarah E Schultz
Worcester Polytechnic Institute
100 Institute Rd
Worcester, MA
seschultz@wpi.edu

Ivon Arroyo
Worcester Polytechnic Institute
100 Institute Rd
Worcester, MA
iarroyo@wpi.edu

ABSTRACT
Knowledge tracing has been used to predict students' knowledge and performance for almost twenty years. Recently, researchers have become interested in looking at students' behaviors, especially those considered gaming behaviors. In this work, we attempt to leverage a variation of knowledge tracing to predict gaming behaviors without damaging the prediction of performance. We compare the predictions of this model to those of knowledge tracing and a separate engagement tracing model.

Keywords
Knowledge tracing, affect, engagement, gaming, behavior

1. INTRODUCTION
When Corbett and Anderson first published the knowledge tracing model in 1995, they claimed that their goal was "to implement a simple student modeling process that would allow the tutor to […] tailor the sequence of practice exercises to the student's needs" [1]. While knowledge tracing is generally able to predict students' performance "quite well," it does not take into account the possibility of disengagement. Traditionally, knowledge tracing is used with the probability of transition from a learned to an unlearned state set at 0, so students who become disengaged are not presumed to be forgetting the skill. When the forgetting transition is allowed, models such as knowledge tracing can become confounded, mistaking disengagement for unlearning, as illustrated in Figure 1.

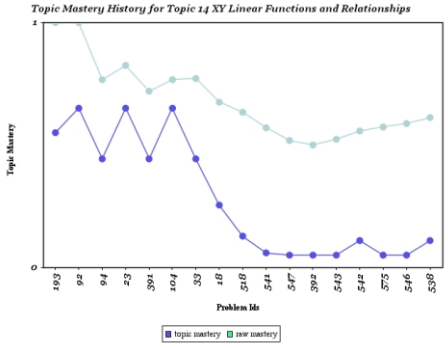


Figure 1- Bayesian Knowledge Estimation of a student on one skill (bottom line)

Figure 1 suggests that this student was un-learning, while after looking at the logs in detail, it was clear that, after the 7th problem, the student was just clicking through all the available multiple-choice answers without attempting to answer correctly.This type of behavior is defined by Baker et al as "gaming the system" [2] and is considered to be an indicator of disengagement or negative affect. Some work has been done in modeling engagement and affect in Intelligent Tutoring Systems [3], but relatively little research has focused on combining these methods with ways of tracking knowledge, such as knowledge tracing, in order to create a model that can predict both student performance and disengaged behavior and intervene appropriately.

2. PREVIOUS WORK

2.1 Bayesian Knowledge Tracing
Corbett and Anderson's Bayesian Knowledge Tracing (BKT) [1] (Figure 2) is a hidden Markov model. At each time step there is a latent node, knowledge, and an observed node, performance. The parameters for this model are $P(L_0)$, the probability that a student already knows the skill; P(T), the probability of learning the skill from one time-step to the next; P(G), the probability that a student who does not know the skill but correctly guesses; and P(S), the probability that a student who does know the skill slips and gets the answer incorrect. As mentioned in the introduction, P(F), forgetting, is traditionally set at 0, however for this work we allow forgetting in order to see if looking at behavior affects the amount of forgetting that students appear to do.

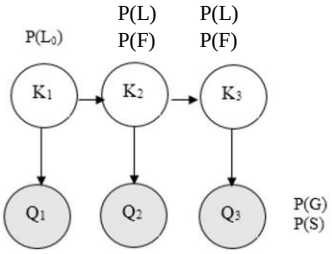


Figure 2- Bayesian Knowledge Tracing

2.2 HMM-IRT
In 2006, Johns and Woolf proposed the Dynamic Mixture Model (DMM) for predicting student knowledge and engagement [4]. They used a hidden Markov model like BKT for tracing engagement, but paired it with an Item Response Theory-like model for predicting knowledge. Rather than predicting knowledge at each time step, there is a single knowledge node for every skill and students' performance relies on that in addition to their engagement state. This allowed more accurate knowledge predictions than IRT alone, as disengagement, indicated by gaming behaviors, could explain away some incorrect attempts, rather than attributing those to knowledge.

Published in CEUR-WS:
BKT20y workshop (Yudelson, González-Brenes and Mozer)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

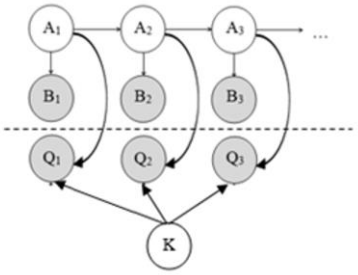


Figure 3- Dynamic Mixture Model

2.3 The KAT Model

In our previous work [5], we proposed the knowledge and affect tracing (KAT) model (Figure 5), which combines two hidden Markov models, BKT and the engagement tracing piece of DMM. As in DMM, affect influences performance. This model was able to predict both performance and behavior better than the dynamic mixture model, but did not predict performance as well as standard BKT, perhaps due to over-parameterization [5].

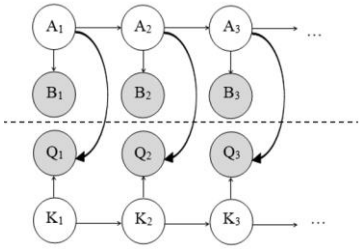


Figure 4- The KAT Model

3. THE KTB MODEL

We propose the "Knowledge Tracing with Behavior" (KTB) model. This model has only one latent node, which we call "knowledge"-- although in reality is a combination of both knowledge and engagement-- and two observables, performance and gaming behaviors. This model is shown in Figure 5.

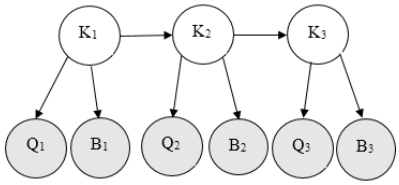


Figure 5- KTB Model

This model has fewer parameters than the dynamic mixture model or KAT model, but still can predict both performance and disengaged behavior of the students.

The variable called Gaming Behavior (B) is defined as either gaming or normal. See our definition for "gaming" in this context in our previous work [5].

4. BAYESIAN ENGAGEMENT TRACING

Since the performance prediction of the KTB model can be compared to that of Bayesian Knowledge Tracing, it is necessary to have a model of engagement tracing to compare the behavior predictions. To that end, we include a model of "Bayesian Engagement Tracing" (BET) in this work, which is the same as the HMM part of Johns and Woolf's model or the engagement piece of the KAT model, but not connected to any other model (top part of figure 4).

5. DATASETS AND METHODS

The data and methods used in this work was the same as that used in [5]. The data came from two tutors for middle and high school mathematics, ASSISTments and Wayang Outpost. For details, please see [5] in the main conference proceedings.

6. RESULTS AND ANALYSIS

While KT and KTB both outperform KAT and DMM in all predictions, in seven of the nine knowledge components, KTB was better able to predict performance than standard knowledge tracing, although the only significant difference between the two was in the ASSISTments skill "Circle Graph" ($p=0.03$). Interestingly, the Bayesian engagement tracing model was better able to predict students' behavior than KTB in eight of the nine knowledge components, although the differences are again not significant, except in two cases, "Box and Whisker," and "Triangles" ($p=0.02$).

7. DISCUSSION

We have proposed a new model, knowledge tracing with behavior, which can predict both student performance and behavior, and have shown that it can do so at least as well as BKT and a separate Bayesian engagement tracing, at predicting future behaviors (correctness at responding math problems and gaming behaviors). KTB seems to stop the false forgetting effect that is recorded by KT when forgetting is not allowed to be zero.

ACKNOWLEDGEMENTS

This research is supported by the Office of Naval Research, STEM Challenge Award, # N0001413C0127US. We also acknowledge funding from NSF (#1316736, 1252297, 1109483, 1031398, 0742503), and IES (# R305A120125 & R305C100024). Any opinions or conclusions expressed are those of the authors, not necessarily of the funders.

REFERENCES

[1] Corbett, A.T., Anderson, J.R., "Knowledge tracing: Modeling the acquisition of procedural knowledge." *User Modeling and User-Adapted Interaction*, 1995, 4, p.253-278.

[2] Baker, R.S., Corbett, A.T., Koedinger, K.R., Wagner, A.Z. (2004) Off-Task Behavior in the Cognitive Tutor Classroom: When Students "Game The System". In *Proceedings of ACM CHI 2004: Computer-Human Interaction*, 383-390.

[3] Beck, J.E. "Engagement tracing: using response times to model student disengagement." *Proceedings of AIED conference*, 2005. p. 88-95. IOS Press

[4] Johns, J. and Woolf, B.P. "A Dynamic Mixture Model to Detect Student Motivation and Proficiency." *Proceedings of AAAI Conference*, 2006, 1, p. 163-168.

[5] Schultz, S. and Arroyo, I. "Tracing Knowledge and Engagement in Parallel in an Intelligent Tutoring System." To appear in *Proceedings of the 7th Annual International Conference on Educational Data Mining*, 2014

Evaluating Student Models

Adaeze Nwaigwe
University of Maryland University College
3501 Unversity Blvd East
Adelphi, MD 207831 412 608 8747
adaeze.nwaigwe@faculty.umuc.edu

ABSTRACT
We use the Additive Factors Model to drive the evaluation of the student model of an Intelligent Tutoring System. Using data from the Andes Physics Tutor, applying the simple location heuristic and implementing the Additive Factors Model tool in the Pittsburgh's Science of Learning Center's DataShop, we discover possible ways to improve the student model of the Andes Intelligent Tutor.

Keywords
Student modeling, learning curves, additive factors model.

1. INTRODUCTION
The quality of student models drive many of the instructional decisions that automated tutoring systems make, whether it is what feedback to provide, when and how to sequence topics and problems in a curriculum, how to adapt pacing to the needs of students and even what problems and instructional materials are necessary [1]. We used the Additive Factors Model (AFM) tool in the Pittsburgh's Science of Learning Center's (PSLC) DataShop to identify areas for improvement in the curriculum for the ANDES Intelligent Tutoring System.

1.1 BACKGROUND
Learning curves derived from student models drive evaluation, revision and improvement of the Intelligent Tutor. The AFM is a statistical algorithm which models learning and performance by using logistical regression performed over the "error rate" learning curve data [1]. If a student is learning the knowledge component (KC) or skill being measured, the learning curve is expected to follow a so-called "power law of practice" [2]. If such a curve exists, it presents evidence that the student is learning the skill being measured or conversely, that the skill represents what the student is learning.

While use of learning curves is now a standard technique for assessing the cognitive models of Intelligent Tutors, the technique requires that a method is instated for attributing blame to skills or KCs. This simply means that each error a student makes must be blamed on a skill or set of skills. Four different heuristics for error attribution have been proposed and tested. These heuristics are guided by whether the method is driven by location – the simple location heuristic (LH), the model-based location heuristic (MLH); or by the temporal order of events – the temporal heuristic (TH), the model-based temporal heuristic (MTH); and whether the choice of the student model is leveraged (MLH, MTH) [3].

2 EVALUATING THE STUDENT MODEL

2.1Adapting the Andes Log data for the AFM Algorithm
The log data used for this work was obtained from the Andes Intelligent Tutor [4] and encompassed four problems in the area of electric field, across 102 students. The data was collected in Spring 2005 at the US Naval Academy during its regular physics class and as part of the PSLC's LearnLab facility that provides researchers, access to run experiments in or perform secondary analyzes of data collected from one of seven available technology-enhanced courses running at multiple high school and college sites (see http://learnlab.org).

Prior to using the AFM tool on the dataset, the simple location heuristic (LH) was applied to error transactions in the Andes log data which had missing KCs. That is, when the Andes failed to assign blame to a KC on an error transaction, the LH will select the first correctly implanted KC in the same location as the error. The LH was applied to about 44% of the original data. Table 1 depicts a summary of the LH data.

2.2 Generating Model Values using AFM
The Datashop's AFM algorithm was used to compute statistical measures of goodness of fit for the model - Akaike Information Criterion (AIC) and Bayesian Information criterion (BIC), as well as to generate learning curves for the Andes log data.

3 RESULTS AND DISCUSSION
We found that there were 5 groups of KCs – "Low and Flat", "No learning", "Still high", "Too Little data" and "Good". The "Low and Flat" group indicated KCs where students likely received too much practice. It appears that although students mastered the KCs they continued to receive tasks for them. It may be better to reduce the required number of tasks or change Andes' knowledge tracing parameters so that students get fewer opportunities with these KCs. The "Still high" group suggests KCs, which students continued to struggle with. Increasing opportunities for practice for these KCs might improve the student model. The "No learning" group indicated KCs where the slope of the predicted learning curve showed no apparent learning. A step towards improving the student model could be to explore whether each of these KCs can be split into multiple KCs. The new KCs may better reflect the variation in difficulty and transfer of learning that may be happening across problem steps, which are currently labeled by each KC. The KCs in the "Too Little data" group seem to be KCs for which students were exposed to insufficient practice opportunities for the data to be meaningful. For these KCs, adding

Published in CEUR-WS:
BKT20y workshop (Yudelson, González-Brenes and Mozer)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

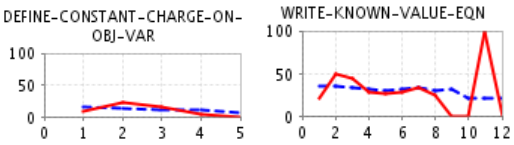
more tasks or merging similar KCs might provide data that is interpretable. The KCs that appeared "Good" may reflect those in which there was substantial student learning. Table 2 shows the different group of KCs, their frequencies and AIC and BIC scores. Figures 1a – 1d show the different groups of KCs. Intercept (logit) and intercept (probability) both indicate KC difficulty. Higher intercept values indicate more difficult KCs. The slope parameter indicates the KC learning rate. Higher values suggest students will learn such KCs faster.

Table 1. LH Data Summary

Number of Students	102
Number of Unique Steps	125
Total Number of Steps	5,857
Total Number of Transactions	71,300
Total Student Hours	107.02
# of Knowledge Component Model	34

Table 2. KC Groups and Statistical Scores

Low and Flat	No Learning	Still High	Too Little data	Good
2	2	4	24	2
# of Knowledge Components				34
AIC				6532.75
BIC				7668.14



KC Name	Intercept (logit)	Intercept (probability)	Slope
define-constant-charge-on-obj-var	1.77	0.85	0.120
write-known-value-eqn	0.63	0.65	0.037

Figure 1a – "Good"

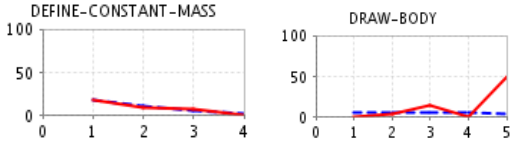
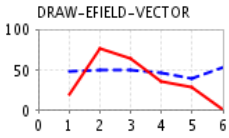
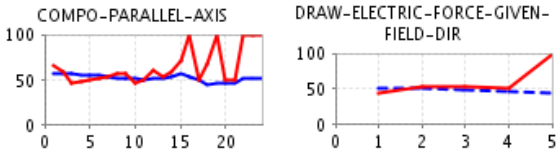


Figure 1b – "Low and Flat"



KC Name	Intercept (logit)	Intercept (probability)	Slope
draw-efield-vector	0.06	0.52	0.000

Figure 1c – "No Learning"



KC Name	Intercept (logit)	Intercept (probability)	Slope
compo-parallel-axis	-0.28	0.43	0.000
draw-electric-force-given-field-dir	-0.01	0.50	0.000

Figure 1d – "Still High"

4 CONCLUSION AND FUTURE WORK

This paper presented how the AFM can be used to evaluate the student model of the Andes Physics Tutor. Refining four of the five groups of KCs identified, might improve the Andes student model. A further approach would to use Learning Factors Analysis [1] algorithm to automatically find better student models by searching through a space of KC models. The next step is to explore these options and measure their effect.

5 ACKNOWLEDGMENTS

Our thanks to the Pittsburgh Science of Learning Center for providing the analysis tool for this work, to Bob Hausmann and Kurt VanLehn for dataset access.

6 REFERENCES

[1] Koedinger, K.R., McLaughlin, E.A., Stamper, J.C. 2012 Automated Student Model Improvement. Proceedings of the 5th International Conference on Educational Data Mining, Chania, Greece, pp. 17–24.

[2] Mathan S. & Koedinger K. 2005. Fostering the Intelligent Novice: Learning From Errors With Metacognitive Tutoring. Educational Psychologist. 40(4), pps. 257–265.

[3] Nwaigwe, A. & Koedinger, K.R. 2011. The Simple Location Heuristic is Better at Predicting Students' Changes in Error Rate Over Time Compared to the Simple Temporal Heuristic. Proceedings of the 4th International Conference on Educational Data Mining. Eindhoven, Netherlands.

[4] VanLehn, K., Lynch, C., Schultz, K., Shapiro, J. A., Shelby, R. H., Taylor, L., et al. 2005. The Andes physics tutoring system: Lessons learned. International Journal of Artificial Intelligence and Education, 15(3), 147-204.