# Evaluating the Feasibility of Learning Student Models from Data *

**Anders Jonsson†, Jeff Johns†, Hasmik Mehranian‡, Ivon Arroyo§,**

**Beverly Woolf §, Andrew Barto†, Donald Fisher‡, Sridhar Mahadevan†**

† : Autonomous Learning Laboratory
Department of Computer Science
University of Massachusetts
Amherst MA 01003, USA

§ : Center for Knowledge Communication
Department of Computer Science
University of Massachusetts
Amherst MA 01003, USA

‡ : Human Performance Laboratory
Department of Mechanical & Industrial Engineering
University of Massachusetts
Amherst MA 01003, USA

## Introduction

Human teachers are expected to improve with experience; they should learn how to handle different types of students, and learn which methods work for which learning difficulties. But how can a computerized teacher improve its skills over time after working with hundreds of students? How can it recognize that certain problems are too difficult for a student or identify which hints should follow other hints? How can it become more skillful over time? Our research is directed toward providing answers to these questions, which may be divided into two distinct issues: 1) creation of student models from student log file data, and 2) creation of optimal policies that query the student model to determine appropriate responses to maximize student learning. This paper is an initial step toward addressing the first component: creating student models from behavioral data.

The worth and difficulties of using student log file data to improve tutoring systems has been discussed (Heiner, Beck, & Mostow 2004). Some ideas have been put forth about how to improve the model of the domain by taking into account students' behavior (Koedinger & Mathan 2004) and also on how to learn Bayesian networks to discover links between behavior and motivation/attitudes/goals (Arroyo & Woolf 2004; Conati, Gertner, & VanLehn 2002). Beck & Woolf (Beck & Woolf 2000a; 2000b; Beck, Woolf, & Beal 2000) demonstrated how past student data can be used to form a population student model. Some of these past student models have been fairly sophisticated, using dynamic Bayesian networks with hundreds of nodes to model the students. Bayesian networks are particularly appropriate given the level of uncertainty of a student's behavior (Conati, Gertner, & VanLehn 2002; Murray & VanLehn 2000; Reye 1998). The main limitation of this past work is that crucial parameters of these networks — such as the capability of a hint to support a student's skill transition from an unmastered to a mastered state — have been set by hand. Mayo & Mitrovic (Mayo & Mitrovic 2001) realized this limitation and constructed an English capitalization tutor (CAPIT) that learned the parameters of a dynamic Bayesian network from students' behavior. Their student model involved nodes that represented observable behaviors, such as the incorrect responses to a capitalization problem (constraint violation outcomes). CAPIT predicted performance on a problem based on the outcome of past problems involving similar constraints, or skills. They suggested that having hidden nodes of skill mastery was undesirable, because of the need to hand-code parameters such as the impact of hints on skills, and thus relied merely on observable data. Our work is a step forward in that it focuses on learning student models that include hidden skill mastery nodes, without hand-coding any parameters of the model. We use machine learning techniques that deal with missing data (Expectation-Maximization) to learn parameters of the network.

Ultimately, our goal is to develop an intelligent tutoring system that prepares high school students for standardized geometry tests. To meet this goal, we have developed a computerized tutor called Wayang Outpost (Beal *et al.* 2003) that trains students on geometry problems. Successful solution of these problems requires a variety of skills with different levels of difficulty. For instance, simple skills govern a student's ability to recognize fundamental geometry concepts (such as the identification of a rectangle), while complex skills are defined as compound concepts (such as finding the side of a 30-60-90 triangle given that the length of one other side is known). Both simple and complex skills

*We gratefully acknowledge support of this work from the National Science Foundation through two awards: 1) Learning to Teach: The Next Generation of Intelligent Tutor Systems, B. Woolf, P.I., with A. Barto, D. Fisher, S. Mahadevan and I. Arroyo, co-P.I.s, NSF REC ROLE, # 0411776, 08/15/04-08/14/07; and 2) AnimalWorld: Enhancing High School Women's Mathematical Competence, C. Beal, P.I., with B. Woolf & M. Royer, co-P.I.s, HRD-EHR Act for Women & Girls in Sem, 11/1/01-10/31/04. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the granting agencies.
Copyright © 2005, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

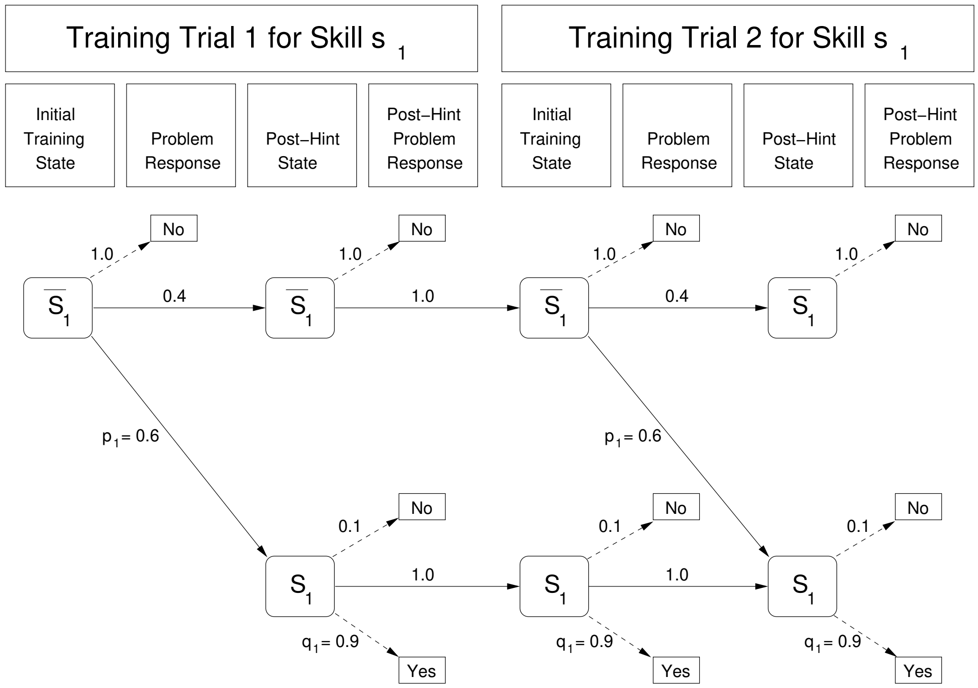


Figure 1: The first and second training sessions for skill $s_1$

are used in the process of solving a problem. If we can identify the set of skills required to solve the test problems, student modeling consists of estimating a student's level of mastery of the set of skills. This model can be used to decide which problem to present to a student next. We can use the model to predict whether a student will answer a problem correctly, and update our estimate based on the student's actual answer. Learning in the tutor takes place through the use of hints. A hint describes the information associated with a single skill and shows the application of the skill to the solution of the problem.

We evaluate the feasibility and accuracy of automatically learning parameters for Bayesian student models with latent variables (skill mastery) from behavioral data. To accomplish this goal, we present simulated student behavioral data generated from a model of the training of one-skill problems, described in Section 2. In Section 3, we introduce a dynamic Bayesian network to model the problem-solving activity of students, including hidden (latent) nodes of skill mastery. We explain and describe the results of using parameter learning techniques that deal with missing data to learn probability distributions over hidden nodes (the mastery of skills), guesses, slips, and the capability of hints to make the student master skills, among others. In Section 4, we compare the learned parameters to the true originating ones, and end with a discussion of the feasibility of using machine learning to learn sophisticated student models with hidden variables from actual student behavioral data.

Table 1: Conditional probabilities of the skill model.

| Problem | $p_i$ | $q_i$ |
|---|---|---|
| 1 | 0.6 | 0.9 |
| 2 | 0.5 | 0.8 |
| 3 | 0.4 | 0.7 |

## Generative skill model

Eventually, we want to train a student model on actual student data. However, to test the feasibility of estimating skill mastery levels, we tested our approach on simulated data first. We developed a simple training model consisting of ten trials during which a student is trained on three skills: $s_1$, $s_2$, and $s_3$. The motivation for this model came from a state-based approach to skill acquisition (Fisher 1996). We assume that students do not have skills $s_1$, $s_2$ or $s_3$ mastered at the onset of learning. We train only on problems that require one skill. If a student answers a problem incorrectly during learning, a hint is shown. As a result of seeing the hint, the student masters the skill with probability

$$P(s_i = \text{mastered} \mid s_i = \text{not mastered}, h_i = \text{shown}) = p_i,$$

where $h_i$ is the hint associated with skill $s_i$, $i = 1, 2, 3$. After the hint is shown, a new problem that requires the same skill is presented to the student. Let $o_i$ be the student's answer to problem $i$, which depends on whether the student successfully applies skill $s_i$. The conditional probability of successfully applying skill $s_i$, given that it is mastered, is:

$$P(o_i = \text{correct} \mid s_i = \text{mastered}) = q_i.$$

In this model, we are not concerned with the precise sequence of training trials but with the number of times each skill is trained. We generated data for 36 possible combinations of the three skills trained during 10 trials with the parameter values subjective set according to Table 1.

Figure 1 illustrates the first and second training sessions for skill $s_1$. In the figure, $S_1$ denotes the event that $s_1$ is mastered, and $\bar{S}_1$ denotes the event that $s_1$ is not mastered. We can calculate marginal probabilities of a student answering correctly after being trained on the skill once or twice. Following the same logic, we can calculate that probability for any number of training sessions for each skill.

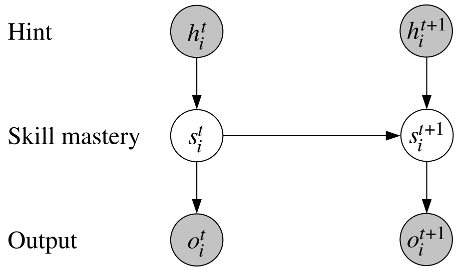


Figure 2: The DBN model of a one-skill problem

## Student model

There are several things to consider when selecting an appropriate model of student learning. Many factors influence a student's behavior during interaction with a tutor, such as attitude, knowledge, fatigue, and interest. These characteristics are largely unobservable and can only be inferred from observations. A suitable model should deal with unobservable information and perform inference. There is also a range of possible observations that can be made about a student, such as the answer to a specific problem, the time it took to give an answer, and the student's facial expression. A model has to specify which information to include and which to ignore, both in terms of unobservable characteristics and observations.

The goal of an intelligent tutoring system is to increase a student's knowledge of a specific subject matter. As a student interacts with a tutor, the student's knowledge level changes as s/he learns about the subject through the problems and hints presented by the tutor. A student's attitude and fatigue level also varies throughout the tutoring session. A suitable model should describe how a student's knowledge and behavior changes over time. Another useful property of a student model is to predict a student's behavior based on current estimates of the student's characteristics.

We modeled student learning with a dynamic Bayesian network, or DBN (Dean & Kanazawa 1989), which describes how the values of a set of variables evolve over time. The variables can be either observed or unobserved, and several fast algorithms can be used to perform inference. In addition, a DBN facilitates modeling conditional independence between variables, which helps speed up learning and inference. For these reasons, we believe that the DBN is exceptionally well-suited to model student learning.

To start simple, we decided to restrict the unobservable characteristics to a student's knowledge of the subject matter. We also included observations about a student's answers to problems and whether or not a student was presented with hints. In the simple case we are studying, each problem is associated with a single skill that the student is supposed to master. Intuitively, the answer that a student provides to a problem depends on the student's current mastery of the associated skill. The student's mastery of the skill as a result of working on a problem depends on the previous level of mastery and whether or not the student saw a hint.

Figure 2 shows two time-slices of the DBN that we designed to model a student's mastery of a one-skill problem. At time $t$, the variable $s_i^t$ is the current estimate of a student's mastery of skill $s_i$, $h_i^t$ indicates whether or not a hint is being presented, and $o_i^t$ is the student's answer to the problem. Shaded nodes indicate that the associated variable is observed. We assume that a student either masters or does not master the skill, so all variable have discrete values. The student's answer to the problem at time $t$ ($o_i^t$) depends on the student's mastery of the skill at time $t$ ($s_i^t$). The student's mastery of the skill at time $t+1$ ($s_i^{t+1}$) depends on the mastery at time $t$ ($s_i^t$) and whether or not a hint is presented at time $t+1$ ($h_i^{t+1}$).

## Experimental results

The DBN shown in Figure 2 was used for all experiments. Each variable in this network is modeled using a binomial random variable. A latent skill variable $s_i^t$ is either mastered or not mastered. A hint $h_i^t$ is an observed variable that is either shown or not shown. Finally, the observation variable $o_i^t$ is either correct or incorrect.

Multiple experiments were conducted to determine if the student model parameters could be learned from data. There are several reasons why learning the parameters is preferable to hardcoding them. If the model accurately describes the underlying generative process, then learning the parameters results in a model that best fits the data. Furthermore, each problem in the tutor will have different dynamics depending on the number of skills involved and the inherent difficulty of the problem. Automating the parameter estimation process captures an individual problem's dynamics.

Simulated data was generated for three one-skill problems according to the parameter values in Table 1. The goal of these experiments was to determine the requirements for parameter estimation in order to recover the true model. Once the limitations of this method are well understood (i.e. amount of data required to recover the model), we plan to use the same technique on actual log data of high school students using the geometry tutor.

The experiments represent varying levels of problem difficulty, where Problem 3 is harder than Problem 2 which is harder than Problem 1. The increased difficulty comes in the form of lower probabilities of a student getting the problem correct given the skill is mastered as well as lower probabilities of a hint changing a skill from not mastered to mastered. For all tests, the probability of getting a problem

Table 2: Parameters learned for Problem 1 versus the true generative model. Before learning, the parameters were either initialized completely randomly or partially randomly with some of the skill transition parameters set close to the true value.

| Model Parameters | All Random | Partially Random | True |
|---|---|---|---|
| $P(s_1^1 = \text{mastered} \mid h_1^1 = \text{not shown})$ | 0 | 0 | 0 |
| $P(o_1^t = \text{correct} \mid s_1^t = \text{mastered})$ | .91 | .90 | .9 |
| $P(o_1^t = \text{correct} \mid s_1^t = \text{not mastered})$ | 0 | 0 | 0 |
| $P(s_1^{t+1} = \text{mastered} \mid s_1^t = \text{mastered}, h_{t+1} = \text{shown})$ | 1.0 | 1.0 | 1 |
| $P(s_1^{t+1} = \text{mastered} \mid s_1^t = \text{mastered}, h_{t+1} = \text{not shown})$ | .99 | 1.0 | 1 |
| $P(s_1^{t+1} = \text{mastered} \mid s_1^t = \text{not mastered}, h_1^{t+1} = \text{shown})$ | .59 | .59 | .6 |
| $P(s_1^{t+1} = \text{mastered} \mid s_1^t = \text{not mastered}, h_1^{t+1} = \text{not shown})$ | .03 | .01 | 0 |

correct given the skill is not mastered is zero. This simplification, which will be relaxed in future experiments, means the student cannot guess the solution without having knowledge of the requisite skill.

Each dataset consisted of 360 instances (ten trials from each of the 36 possible combinations of skills) of a simulated student answering a problem anywhere from one time to eight times. If a simulated student is given a problem multiple times, there is more evidence for learning the transition probability $P(s_i^{t+1} \mid s_i^t,\ h_i^{t+1})$. For each of the three experiments, the simulated students were assumed to not have the skill mastered on the first time slice. If the student answered incorrectly, then a hint was shown before the student attempted to solve the problem again.

The model parameters were learned using Expectation-Maximization, or EM (Dempster, Laird, & Rubin 1977). This algorithm is used to learn a joint probability distribution when some of the data is hidden. EM alternates between an E-step and an M-step. In the E-step, the expected values for the hidden values are calculated based on the observed data and the current estimate of the model parameters. The model parameters are then updated in the M-step to maximize the joint likelihood given the statistics learned in the E-step. Convergence to a locally optimal solution is guaranteed.

To determine the impact of parameter initialization, we conducted two separate tests. The first test was done using completely random intialization of the seven free model parameters. The second test randomly initialized four of the seven parameters. The remaining three parameters not randomly initialized were assigned the following values at the beginning of EM:

$$P(s_i^{t+1} = \text{mastered} | s_i^t = \text{mastered}, h_i^{t+1} = \text{shown}) = .99$$

$$P(s_i^{t+1} = \text{mastered} | s_i^t = \text{mastered}, h_i^{t+1} = \text{not shown}) = .99$$

$$P(s_i^{t+1} = \text{mastered} | s_i^t = \text{not mastered}, h_i^{t+1} = \text{not shown}) = .01$$

The results are shown in Tables 2, 3, and 4. The learned parameters, which are in the second and third columns of the tables, are averaged over five trials. The last column in the tables shows the actual values used to generate the simulated data. For each test, EM is able to recover the model parameters within an acceptable margin of error. The results are consistent across Problem 1, Problem 2, and Problem 3; therefore, problem difficulty does not seem to have an effect on the quality of parameter estimation.

In particular, the parameters learned using partially random initialization achieve values very close to the true model. While this is not surprising, it does suggest having enough sequential data is important to effectively learn the skill transition matrix $P(s_i^{t+1} \mid s_i^t,\ h_i^{t+1})$. To attain this data, students need to be given the same problem (or similar versions of that problem) multiple times. We further examined this constraint by varying the number of problems the simulated students saw in the training set. Students were given Problem 3 anywhere from only once up to a maximum of eight times. EM was able to learn better (i.e. closer to the true value with less variance) parameters if the simulated student saw more problems. The results from this experiment are shown in Figure 3 for parameters $p_3$ and $q_3$. The resulting values of $p_3$ and $q_3$ were averaged across 20 trials, each involving 360 students. All model parameters were randomly initialized. The error bars in Figure 3 represent ± one standard deviation from the mean.

## Conclusion

In this paper, we proposed a model of student behavior based on hidden skills. The skills correspond to geometry concepts which a student applies to a problem. The model, which is formulated as a dynamic Bayesian network, is flexible enough to include additional factors that are known to affect student performance (i.e. student attitude and interest) in the future. We chose to keep the model simple at this point to facilitate the evaluation of learning the model parameters from data.

Experiments were conducted using simulated data for one-skill problems. Since the skills are latent variables, we used the Expectation-Maximization algorithm to learn the model parameters. Results were presented for three problems with varying degrees of difficulty and different parameter initializations. We conclude from these experiments that the model parameters can be learned for one-skill problems provided enough sequential data exists to learn the effect of a hint. This *sequential data* condition has an important implication for mining student models from log data. It is only when students see enough problems of a similar kind that mining algorithms such as EM can learn parameters that are important to the student model, such as the capability of a

Table 3: Parameters learned for Problem 2 versus the true generative model.

| Model Parameters | All Random | Partially Random | True |
|---|---|---|---|
| $P(s_2^1 = \text{mastered} \mid h_2^1 = \text{not shown})$ | 0 | 0 | 0 |
| $P(o_2^t = \text{correct} \mid s_2^t = \text{mastered})$ | .82 | .80 | .8 |
| $P(o_2^t = \text{correct} \mid s_2^t = \text{not mastered})$ | 0 | 0 | 0 |
| $P(s_2^{t+1} = \text{mastered} \mid s_2^t = \text{mastered}, h_{t+1} = \text{shown})$ | 1.0 | 1.0 | 1 |
| $P(s_2^{t+1} = \text{mastered} \mid s_2^t = \text{mastered}, h_{t+1} = \text{not shown})$ | .98 | 1.0 | 1 |
| $P(s_2^{t+1} = \text{mastered} \mid s_2^t = \text{not mastered}, h_2^{t+1} = \text{shown})$ | .48 | .49 | .5 |
| $P(s_2^{t+1} = \text{mastered} \mid s_2^t = \text{not mastered}, h_2^{t+1} = \text{not shown})$ | .06 | .03 | 0 |

Table 4: Parameters learned for Problem 3 versus the true generative model.

| Model Parameters | All Random | Partially Random | True |
|---|---|---|---|
| $P(s_3^1 = \text{mastered} \mid h_3^1 = \text{not shown})$ | 0 | 0 | 0 |
| $P(o_3^t = \text{correct} \mid s_3^t = \text{mastered})$ | .74 | .71 | .7 |
| $P(o_3^t = \text{correct} \mid s_3^t = \text{not mastered})$ | 0 | 0 | 0 |
| $P(s_3^{t+1} = \text{mastered} \mid s_3^t = \text{mastered}, h_{t+1} = \text{shown})$ | 1.0 | 1.0 | 1 |
| $P(s_3^{t+1} = \text{mastered} \mid s_3^t = \text{mastered}, h_{t+1} = \text{not shown})$ | .97 | 1.0 | 1 |
| $P(s_3^{t+1} = \text{mastered} \mid s_3^t = \text{not mastered}, h_3^{t+1} = \text{shown})$ | .36 | .38 | .4 |
| $P(s_3^{t+1} = \text{mastered} \mid s_3^t = \text{not mastered}, h_3^{t+1} = \text{not shown})$ | .06 | .03 | 0 |

hint to make a student transition from non-mastery to mastery of a skill. In general, this work shows that mining student models from data seems plausible under certain conditions.

In the future, we will scale up the simulation to include problems involving multiple skills and allow for students correctly guessing the answer without having the required skill. This constraint needs to be relaxed since problems in the geometry tutor and the SAT exam are multiple choice. In conjunction with these experiments, we will determine how well this model explains actual data of high school students who have used the geometry tutor. Lastly, we plan to use the learned student models to help choose an appropriate action for the tutor. We are exploring an optimization framework to select tutoring strategies based on student model predictions.

## References


Arroyo, I., and Woolf, B. 2004. Inferring unobservable learning variables from students' help seeking behavior. *Analyzing Student-Tutor Interaction Logs to Improve Educational Outcomes: Workshop Proceedings of ITS-2004*.

Beal, C.; Arroyo, I.; Royer, J.; and Woolf, B. 2003. Wayang outpost: An intelligent multimedia tutor for high stakes math achievement tests. *American Educational Research Association annual meeting*.

Beck, J., and Woolf, B. 2000a. Reasoning from data rather than theory. *Proceedings of the Thirteenth International Florida Artificial Intelligence Research Symposium*.

Beck, J., and Woolf, B. 2000b. High-level Student Modeling with Machine Learning. *Proceedings of the International Conference on Intelligent Tutoring Systems*, 5: 584–593.

Beck, J.; Woolf, B.; and Beal, C. 2000. ADVISOR: A machine learning architecture for intelligent tutor construction. *Proceedings of the National Conference on Artificial Intelligence*, 17: 552–557.

Conati, C.; Gertner, A.; and VanLehn, K. 2002. Using bayesian networks to manage uncertainty in student modeling. *Journal of User Modeling and User-Adapted Interaction*, 12(4): 371–417.

Dean, T., and Kanazawa, K. 1989. A model for reasoning about persistence and causation. *International Journal of Computational Intelligence*, 5(3): 142–150.

Dempster, A.; Laird, N.; and Rubin, D. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of Royal Statistical Society Series B*, 39: 1–38.

Fisher, D. 1996. State models of skill acquisition: Optimizing the training of older adults. *Aging and skilled performance: Advances in theory and applications*: 17–44.

Heiner, C.; Beck, J.; and Mostow, J. 2004. Lessons on using its data to answer educational research questions. *Analyzing Student-Tutor Interaction Logs to Improve Educational Outcomes: Workshop Proceedings of ITS-2004*.

Koedinger, K., and Mathan, S. 2004. Distinguishing qualitatively different kinds of learning using log files and learning curves. *Analyzing Student-Tutor Interaction Logs to Improve Educational Outcomes: Workshop Proceedings of ITS-2004*.

Mayo, M., and Mitrovic, A. 2001. Optimising its behaviour

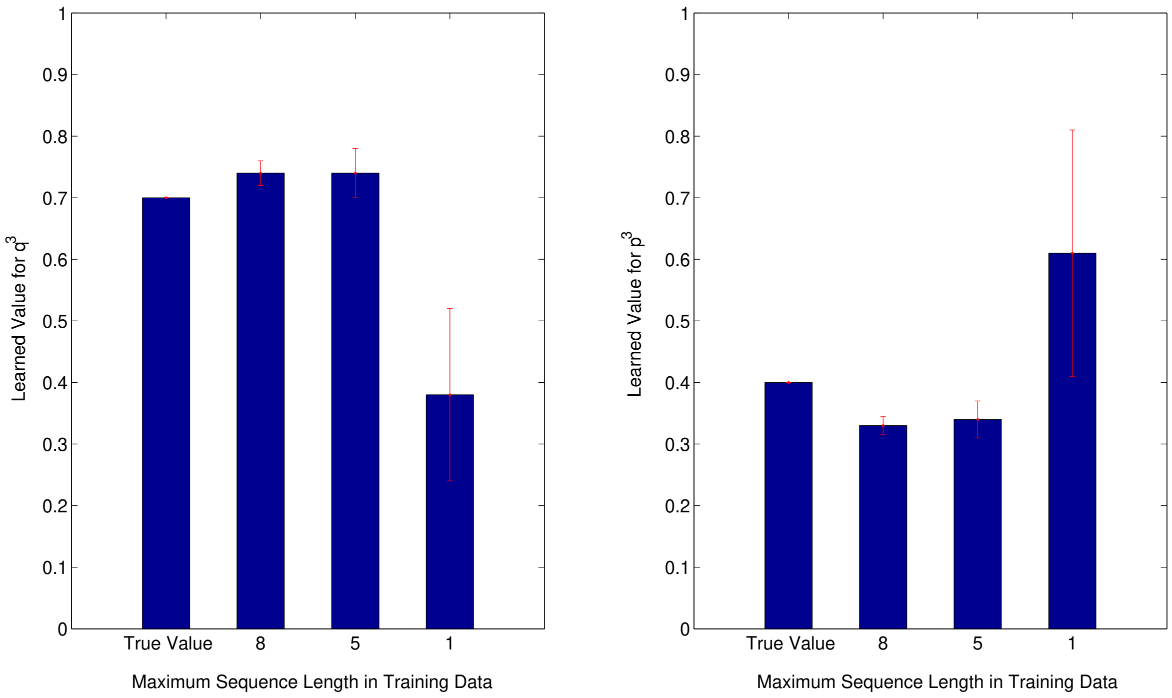


Figure 3: Parameter estimation results for $p_3$ and $q_3$ where students in the training set answered anywhere from one to eight problems. The learned parameters are closer to the true model with less variance when more sequential data is provided.


with bayesian networks and decision theory. *International Journal of Artificial Intelligence in Education*, 12: 124–153.

Murray, R., and VanLehn, K. 2000. DT Tutor: A decision-theoretic, dynamic approach for optimal selection of tutorial actions. *Proceedings of the International Conference on Intelligent Tutoring Systems*, 5: 153–162.

Reye, J. 1998. Two-phase updating of student models based on dynamic belief networks. *Proceedings of the International Conference on Intelligent Tutoring Systems*, 4: 274–283.