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

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

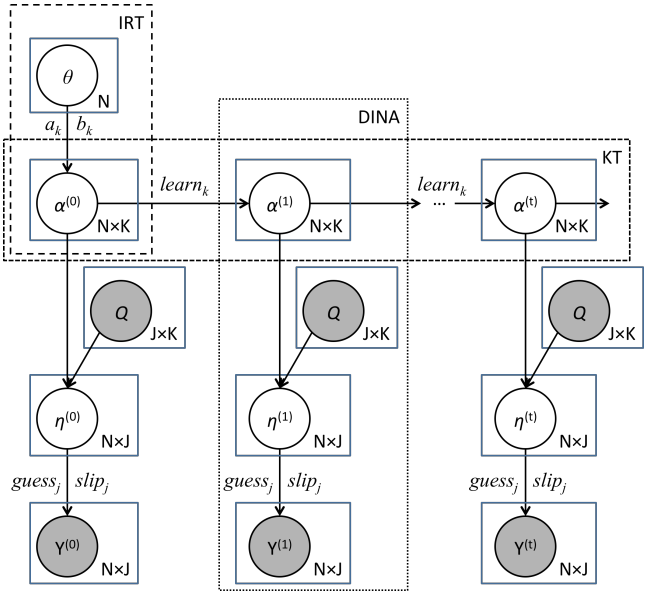
The two skill parameters determine the shape of the IRT curve. As a student's proficiency increases beyond the skill difficulty, the student's chance of performing correctly surpasses 50%. The skill discriminability reflects how fast the logit (log odds) increase or decrease when the proficiency changes. 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, α, 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 $\hat{\boldsymbol{a}}$ and difficulty $\hat{\boldsymbol{b}}$ 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 $\hat{\boldsymbol{a}}$, $\hat{\boldsymbol{b}}$ 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***	$\hat{\boldsymbol{a}}$ (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***	$\hat{\boldsymbol{b}}$ (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 $\hat{\boldsymbol{\theta}}$. 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)

	$\widehat{\boldsymbol{guess}}$	$\widehat{\boldsymbol{1-slip}}$	$\hat{\boldsymbol{\theta}}$
RMSE	0.0103	0.0196	0.9183
	$\hat{\boldsymbol{\alpha}}$		
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 $\boldsymbol{\alpha}$ 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 $\boldsymbol{\alpha}$ 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 $\boldsymbol{\alpha}$ 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.

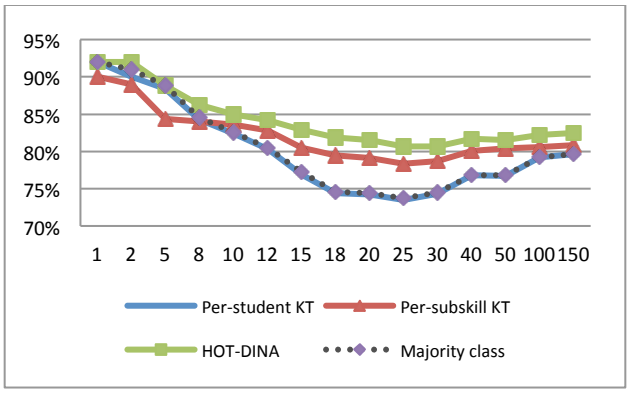


Figure 2. Accuracy on student's 1st, 2nd, 3rd, … test steps

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.

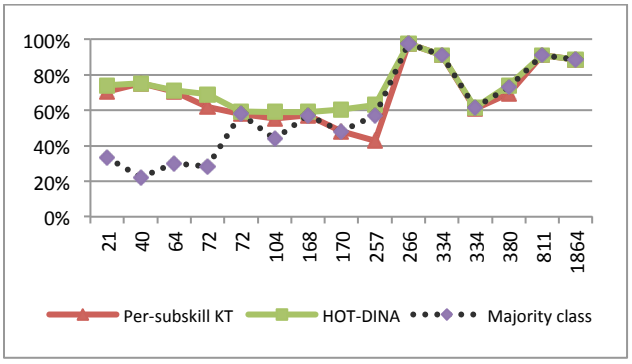


Figure 3. Skills sorted by amount of training data

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.

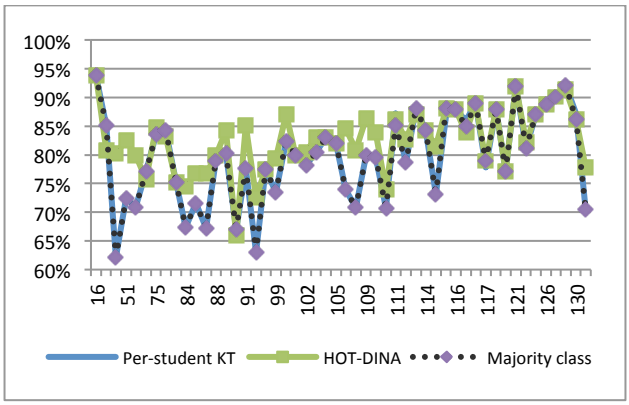


Figure 4. Students sorted by amount of training data

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.