Dynamic Cognitive Tracing: Towards Unified Discovery of Student and Cognitive Models

José P. González-Brenes and Jack Mostow
Project LISTEN
Language Technologies Institute
Carnegie Mellon University
{joseg, mostow}@cs.cmu.edu

ABSTRACT

This work describes a unified approach to two problems previously addressed separately in Intelligent Tutoring Systems: (i) Cognitive Modeling, which factorizes problem solving steps into the latent set of skills required to perform them [7]; and (ii) Student Modeling, which infers students' learning by observing student performance [9].

The practical importance of improving understanding of how students learn is to build better intelligent tutors [8]. The expected advantages of our integrated approach include (i) more accurate prediction of a student's future performance, and (ii) clustering items into skills automatically, without expensive manual expert knowledge annotation.

We introduce a unified model, Dynamic Cognitive Tracing, to explain student learning in terms of skill mastery over time, by learning the Cognitive Model and the Student Model jointly. We formulate our approach as a graphical model, and we validate it using sixty different synthetic datasets. Dynamic Cognitive Tracing significantly outperforms single-skill Knowledge Tracing on predicting future student performance.

1. INTRODUCTION

We propose Dynamic Cognitive Tracing as a method that estimates from performance data:

1. **A Student model.** The estimate of a student's knowledge of a skill in a given time.
2. **A Cognitive Model.** The skills a students require to solve a problem step.

Let's illustrate the student modeling problem with an example. Suppose we are interested in modeling data from a reading tutor that listens to children read aloud. Figure 1 shows sample data in this scenario. We follow the convention of referring to the scorable steps in an intelligent tutor task as "items" [27]. The input variable is the item id_t, which in this case is the word read by a student at time step t. The target variable p_t is the performance of the student– in this case whether the tutor accepted the word read. The student reads the words "smile because it" correctly, but misreads the word "happened". The student modeling problem is to predict future student performance.

Existing student modeling techniques require cognitive models, assignments of items to skills [9]. This is a very expensive requirement, since it often depends on expert domain knowledge [4]. For example, in our reading tutor scenario, it is not a trivial endeavor to cluster a dictionary of words into the set of skills needed to read them.

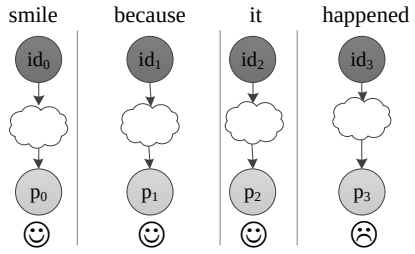


Figure 1: Reading tutor example of student modeling

Unfortunately, the success of existing methods for automatic construction of cognitive models has been limited [11]. Current methods for discovering cognitive models are restricted in that they cannot handle longitudinal data, or that they are not fully automatic. For example, Principal Component Analysis, Non-Negative Matrix Factorization [27] and the Q-Matrix Method [2] ignore the temporal dimension of the data. On the other hand, Learning Factors Analysis [7] is designed for temporal data, but it requires an expert's cognitive model. Our main contribution is a fully automatic approach to discover a cognitive model of longitudinal student data. Our goal is discovering student models, while simultaneously clustering similar items together.

The rest of this document is organized as follows. Section 2 reviews related prior work. Section 3 describes our approach, Dynamic Cognitive Tracing, to jointly learn a student model jointly with a factorization of items into skills. Section 4 evaluates performance using synthetic data. Section 5 provides some concluding remarks.

2. RELATION TO PRIOR WORK

In this section we study Dynamic Cognitive Tracing's relation with prior work. Section 2.1 surveys previous approaches to learn student models. Section 2.2 summarizes automatic approaches for cognitive model discovery.

2.1 Student Modeling

Corbett and Anderson [9]'s seminal paper introduced Knowledge Tracing as a way to model students' changing knowledge during skill acquisition. It uses (a) a cognitive model that maps a problem solving item to the skills required, and (b) logs of students' correct and incorrect answers as evidence of their knowledge on a particular skill. Reye [22] showed that there is an equivalent formulation of Knowledge Tracing as a Bayesian Network. Knowledge Tracing has enabled significantly faster teaching by Intelligent Tutors, while achieving the same performance on evaluations [8].

Knowledge Tracing, as well as Dynamic Cognitive Tracing, are non-convex problems. This means that the optimizer that estimates the parameters of the models might get stuck in local optima far away from the global optimum. Moreover, these formulations are also non-identifiable: There exist potentially many student models that may explain the data observed equally-well. In Knowledge Tracing, the main source of non-identifiability is the trade-off between the probability of a student's initial knowledge, and the probability of learning the skill [5]. To mitigate non-identifiability, recent work has proposed the use of Bayesian priors [5] or using contextual clues to estimate whether a student has guessed [1].

Other approaches to student modeling include Performance Factor Analysis [19, 14], which predicts student performance based on the item difficulty and student historical performances. Alternatively, Learning Decomposition [6], uses non-linear regression to determine how to weight different types of practice opportunities relative to each other. More recently, Tensor Factorization [25], has been used to the student modeling problem. It use recommender system techniques to learn student models. None of these techniques aim to discover cognitive models. Thai-Nghe et al. [25] make use of latent variables, but they argue that it is not possible to interpret their semantics. Their formulation is tied to specific students, and it is not clear how to generalize their approach to unseen students in the training set, or when students encounter only a very sparse set of items. We designed Dynamic Cognitive Tracing aiming to discover latent factors with the interpretation of Cognitive and Student Models.

Desmarais [11] argues that the construction of a cognitive model from data is highly desirable, not only to avoid the labor intensive task of specifying which skills are involved in which task, but because a data-driven approach might outperform human judgment. In the next subsection we study such approaches.

2.2 Automatic Discovery of Cognitive Models

Winters et al. [27] surveyed methods for automatic construction of cognitive models. Examples are matrix factorization techniques, such as Principal Component Analysis (PCA) and Non-Negative Matrix Factorization (NNMF). The theoretical relationships between different matrix factorization techniques has been studied in detail [24].

The Q-matrix algorithm [2, 3], is a hill-climbing method that creates a cognitive model linking skills and items directly from student response data. An alternative approach, Learning Factors Analysis [7], performs combinatorial search to evaluate and improve on existing cognitive models.

None of the techniques reviewed in this section take into account the temporal dimension of the data without human intervention. To the extent of our knowledge, we are the first ones to estimate a cognitive model completely automatically from data collected over time.

3. DYNAMIC COGNITIVE TRACING

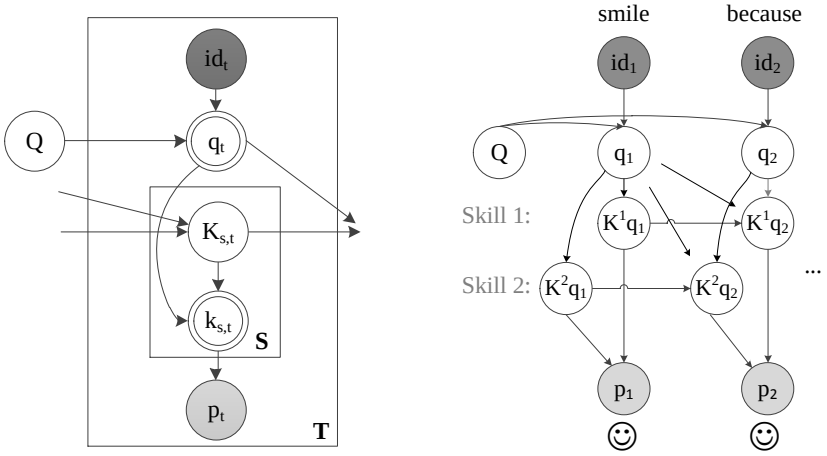
We now describe Dynamic Cognitive Tracing. Subsection 3.1 details our approach. Subsection 3.2 provides pointers on the training and inference algorithms used. Subsection 3.3 shows how Dynamic Cognitive Tracing relates two common techniques used in student modeling and in automatic generation of a cognitive model.

3.1 Model

We formulate Dynamic Cognitive Tracing as a Bayesian Network. Bayesian Networks [20], are a popular framework to reason using noisy information. Bayesian networks are directed acyclic graphical models where the nodes are variables and the edges specify statistical dependencies between variables.

Bayesian Networks are often described using plate diagram notation to show the statistical relationship between their random variables. The plate diagram of Dynamic Cognitive Tracing is shown in Figure 2(a). Instead of drawing a variable multiple times, we follow the convention of using a plate to group repeated variables. As an example, we unroll Dynamic Cognitive Tracing using two skills in Figure 2(b). The description of the generative story of the variables is described in Figure 3. We follow the convention of using dark-gray to color variables that are observable during both training and testing. Variables visible during testing only are colored in light gray. Latent variables, which are never observed, are denoted in white circles. The double-line around variables is used to indicate that their value is calculated deterministically given its parents. The variables in Dynamic Cognitive Tracing are:

- **S** is the number of skills in the model.
- **Ids** is the number of items that the student can practice with the tutor. For example, in the case of a reading tutor, **Ids** is the vocabulary size. If the tutor is creating items on the fly, **Ids** is the number of templates from where items are being generated.
- Q is an $\mathbf{Id} \times \mathbf{S}$ matrix that maps items to skills. Each row Q_{id} is modeled as a multinomial representing the skills required for item id. For example, if $Q_{id_t} = [0.5, 0.5, 0, 0]$, we interpret item id_t to be a mixture of skills 1 and 2. In this example id_t does not require skills 3 and 4. Q need not be hidden. If in fact Q is known, we can clamp the parameters to their known values.
- q_t is the skill for item id_t. For example, $q_t = 1$ iff skill 1 is required for item id_t, $q_t = 2$ iff skill 2 is required, and so on. q_t is chosen deterministically as the row number id_t of Q.
- $K_{s,t}$ indicates whether the student has the knowledge of skill s. Notice, there is a markovian dependency across time steps: if skill s is known at time $t-1$, it is likely to be known it at time t. Therefore, we also need to know which skills were active on the previous time step (i.e., $k_{s,t}$ depends on q_{t-1}). For simplicity, in this



(a) Plate diagram

(b) Unrolled example with two skills

Figure 2: Dynamic Cognitive Tracing as a graphical model

work we treat each K as a binary variable (whether the skill is known or not).

- $k_{s,t}$ is a binary variable that represents if the skill is known and required by the item id_t. Hence, its value is computed deterministically by applying a dot product to its parents: $k_{s,t}$ is true iff skill s is required ($q_t = s$), and the student has learned the skill ($K_{s,t} = 1$).
- p_t is the target variable that models performance. It is only observed during training.
  - For discrete grades (i.e., right or wrong), a Binomial distribution or logistic regression can be used. The use of logistic regression in Bayesian Networks has been studied in the context of mixture of experts [16], and more recently for the multiple subskill problem in student modeling [28]. In this paper we use the Binomial approach.
  - For continuous grades, (i.e., $0 \sim 100$) linear regression can be used.

Our main contribution is unsupervised estimation of the cognitive model Q from longitudinal data, while simultaneously estimating the student model parameters. In the next subsection we study how to learn the parameters of Dynamic Cognitive Tracing, as well as how to perform inference on it.

3.2 Training and Inference

Dynamic Cognitive Tracing is formulated as a directed graphical model (Bayesian Network). We leverage existing technologies to quickly implement a prototype of Dynamic Cognitive Tracing. We used the Bayesian Network Toolkit [18] (BNT) for Matlab.

As described in the previous subsection, the knowledge of a skill is dependent of its value on the previous time step. This kind of dependency is called a Markov Chain. Therefore, in Dynamic Cognitive Tracing, the student knowledge

1. Draw $Q_{id} \sim$ Multinomial; **Ids** times
2. For each time-step $t \in \{0 \dots T\}$:
   (a) Draw $id_t \sim$ Multinomial
   (b) For each skill $s \in \{0 \dots \mathbf{S}\}$:
   (c) Set $q_{s,t} \leftarrow Q_{id_t}$
   (d) Draw $K_{s,t} \sim$ Binomial
   (e) Set $k_{s,t} \leftarrow K_{q_t} \cdot q_{s,t}$
   (f) $p_t \sim \mathcal{N}(k_{1,t}, k_{2,t}, \dots, k_{S,t})$, for continuous p, or for binary variables either $p_t \sim logistic(k_{1,t}, k_{2,t}, \dots, k_{S,t})$, or $p_t \sim$ Binomial

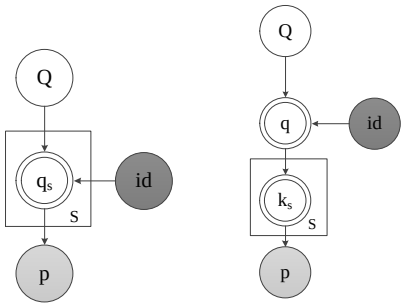
Figure 3: Generative story of Dynamic Cognitive Tracing

of S skills is modeled using S layers of Markov Chains. Unfortunately, this is not scalable, because exact inference on layers of Markov Chains that produce a single output is untractable: the runtime complexity grows exponentially on the number of layers [12]. Hence, we limit our study to a small number of skills. In future work we will implement inference techniques that scale better, like Gibbs Sampling.

The name Bayesian Network is a misnomer, because it does not require to use Bayesian Estimation, as in fact, we used Maximum Likelihood Estimation to perform exact inference. BNT implements the Junction Tree algorithm [15], an inference algorithm that generalizes the the Forward-Backward algorithm that is used in Knowledge Tracing and Hidden Markov Models [21]. To estimate the parameters of the model, we use the Expectation-Maximization (E-M) algorithm [10]. Like all non-convex optimizaters, E-M is not guaranteed to find the globally optimal solution.

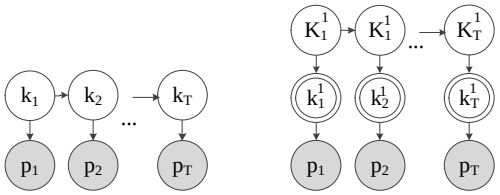
3.3 Unifying Perspective

We now discuss how Dynamic Cognitive Tracing generalizes two common techniques for cognitive and student modeling.



(a) PPCA (b) Dynamic Cognitive Tracing

Figure 4: Two-skill models with one time step



(a) Knowledge Tracing (b) Dynamic Cognitive Tracing

Figure 5: Unrolled graphical model representation of one-skill student models

Cognitive models have been built by matrix factorization techniques [27]. Probabilistic Principal Component Analysis (PPCA) [26] is an example of such matrix factorization techniques. It is a formulation of the Principal Component Analysis algorithm using graphical models. The main advantages of this approach over conventional PCA, is that it can handle missing data, and it provides a probabilistic interpretation of the underlying factors.

In Figure 4(a) we show the graphical model representation of PPCA when explicitly formulated to handle missing data. If the variable p is continuous, it is modeled with a Gaussian. If the variable p is discrete, it is model with a Binomial, using a logistic link function. Discrete PCA is also known in the literature as Logistic PCA [23]. Figure 4(b) shows the simplified Dynamic Cognitive Tracing with two skills, when there is no temporal information available. The structure of both graphical models is very similar: in both cases, the performance is explained by latent variables that represent the skills. The main difference is that Dynamic Cognitive Tracing takes into account the knowledge of the skill estimated from the student model: the performance is explained by the latent knowledge of the skills. We hypothesize that the advantage of our approach lies in the fact that it is not limited to a single timestep like PPCA is. We expect that item-performance data to be very noisy, and that the temporal information would be useful to model skill acquisition.

Figure 5(a) shows the graphical model representation of Knowlege Tracing with a single skill model, which is just a Hidden Markov Model. Figure 5(b) shows the unrolled single-skill Dynamic Cognitive Tracing ($\mathbf{S} = 1$) counterpart. In this case the structure of Dynamic Cognitive Tracing is equivalent to Knowledge Tracing.

4. EMPIRICAL EVALUATION

In this section, we report results of using Dynamic Cognitive Tracing to predict future student performance using synthetically generated datasets. In the context of this paper, we decouple the problem of discovering the assignments of items to skills and the problem of discovering the number of skills. For our experiments, we assume the number of skills is known. In a real scenario, where the number of skills is unknown, it could be estimated by using cross-validation using a held-out set. We report our results using Dynamic Cognitive Tracing using the true number of skills.

Dynamic Cognitive Tracing aims to discover the skills automatically without supervision. We compare if the cognitive model estimated by Dynamic Cognitive Tracing outperforms a cognitive model that assigns all of the items to a single skill. Therefore, as a baseline, we compare against Knowledge Tracing using a single skill.

In all comparisons between Knowledge Tracing and Dynamic Cognitive Tracing, their parameters are estimated using the same training set. The testing and training sets do not overlap students.

4.1 Experimental setup

In this section, we describe the synthetic data sets generation criteria and the evaluation metrics. To generate the synthetic data sets, we use the generative story described in Figure 3, having each student encounter 25 items during training (sequence length = 25). In preliminary experiments, we noticed that by the 25^{th} time step, most synthetic students learned. To have a more balanced test set that has roughly the same number of correct and incorrect answers, the sequence length of the test set is sampled randomly.

We want synthetic data to be plausible; for example, the probability of answering an item correctly by guessing should be lower than the probability of answering an item correctly due to knowledge. Therefore, the synthetic datasets follow these constraints:

- The *learning probability*, the probability of transitioning from not knowing a skill, to knowing it, lies in $[0.01 \ldots 0.45]$.
- The *guess probability*, the probability of answering correctly, given that the student does not know the skill, lies in $[0.01 \ldots 0.30]$.
- The *slip probability*, the probability of answering incorrectly, given that the student knows the skill, lies in $[0.01 \ldots 0.30]$.

Note that these constraints are only exercised for generating the data. None of our models make use of this prior knowledge. For simplicity, in this paper we limit studying cognitive models that have only one skill active per item, but Dynamic Cognitive Tracing does not make use of this information. We constrain the models to not learn the "forget probability" (e.g., the transition probability from "knowing" to "not knowing" is zero).

Knowledge Tracing can sometimes provide bad parameter estimates. Beck and Chang [5] argued that when Knowledge Tracing performs badly, it is often because of incorrect estimation of the initial knowledge of the students (initial probabilities). We want to make sure that our results are better than Knowledge Tracing because of the strengths of Dynamic Cognitive Tracing, not because Knowledge Tracing got stuck in an "unlucky" local optimum. Therefore, we constrain all of the students to not have any initial knowledge in our experiments.

E-M is used to learn the parameters of the models. Knowledge Tracing and Dynamic Cognitive Tracing are initialized with random parameters, however, the emission probabilities (slip and guess probabilities) of Dynamic Cognitive Tracing are initialized using a single-skill model. We experiment running E-M using five different random initializations.

Unless noted otherwise, each dataset is divided in three parts: (i) a training set with 200 students, (ii) a development set with 50 students, used to choose the best out of five random initializations of the E-M algorithm, and (iii) a test set with 50 students. Students do not overlap among the sets.

We report the performance of our models using two metrics:

- **Average Per-item Likelihood.** Likelihood is a common metric to evaluate models that find latent structure [12]. It measures how likely a model is to predict the test set. It penalizes more heavily incorrect predictions with high-confidence. More formally, let I be the number of students in the test set, let $\hat{p}_{i,t}$ be the estimated performance of student i at time t, let $p_{i,t}$ be the real performance of the student and let T_i be the number of time steps for student i. Then we compute the per-item likelihood as:

$$\frac{\sum_{i}^{I}\sum_{t}^{T_i} \text{pr}(\hat{p}_{i,t} = p_{i,t} | \hat{p}_{i,t-1}, id_{i,t})}{\sum_{i}^{I} T_i}$$

- **Classification Accuracy.** Classification accuracy measures how often the predicted performance matches the actual performance. Formally, let $\delta(\cdot)$ be the Indicator function that returns 1 iff its argument is true, and 0 otherwise. We compute the accuracy as:

$$\frac{\sum_{i}^{I}\sum_{t}^{T_i} \delta(\text{pr}(\hat{p}_{i,t} = p_{i,t} | \hat{p}_{i,t-1}, id_{i,t}) > 0.5}{\sum_{i}^{I} T_i}$$

In the next section, we report all of the different parameter combinations of parameters we used to experiment. We did not perform any additional tuning besides the one reported in the next section.

4.2 Results

We create a total of 60 random synthetic datasets using the constraints explained in Section 4.1. All of them have four types of items (**Ids** = 4). We created twenty datasets with 2, 3 and 4 skills (**S** = 2, 3, 4), respectively.

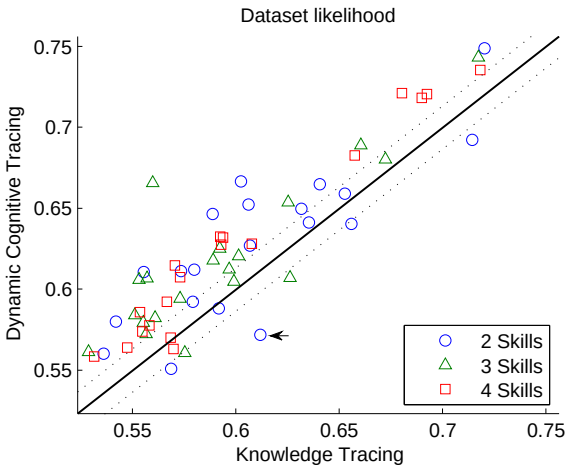


Figure 6: Average Likelihood of Dynamic Cognitive Tracing and single-skill Knowledge Tracing in 60 different data sets

Table 1: Dynamic Cognitive Tracing's worst performing dataset (highlighted in Figure 6)

	Skill 1	Skill 2
Learning probability:	.35	.30
Slip probability:	.09	.08
Guess probability:	.02	.11

In Figure 6, the horizontal axis denotes the Likelihood of single-skill Knowledge Tracing. The vertical axis is the Likelihood of Dynamic Cognitive Tracing. The solid line divides the datasets in which Dynamic Cognitive Tracing performed better than Knowledge Tracing (upper left corner) and the ones in which it performed worse (lower right corner). The dotted lines represent the confidence interval for the mean of the Likelihood of Knowledge Tracing. Dynamic Cognitive Tracing performs as well or above the baseline in a total of 52 (87%) of the datasets.

Is estimating a cognitive model with Dynamic Cognitive Tracing better than assuming a single skill model? We compare the mean Likelihood of Dynamic Cognitive Tracing ($\bar{x}_{\text{DCT}} = 62.34, s_{\text{DCT}} = 5.13$), with the mean Likelihood of single-skill Knowledge Tracing ($\bar{x}_{\text{KT}} = 59.97, s_{\text{KT}} = 5.18$). The null hypothesis is that the mean Likelihood of both models is the same ($H_0 : \mu_{DCT} = \mu_{KT}$). We perform a two-tailed t-test, pairing on the datasets (n=60). We reject the null hypothesis H_0 with confidence $p < 0.05$. We conclude that Dynamic Cognitive Tracing outperforms Knowledge Tracing with a single skill assumption.

In Figure 6 the arrow points to the dataset that performs the worst compared to the single-skill Knowledge Tracing baseline. The Likelihood of the true model is 65%, of Dynamic Cognitive Tracing is 57%, and of single-skill Knowledge Tracing is 61%. We now investigate why Knowledge Tracing outperforms Dynamic Cognitive Tracing on this specific dataset. Table 1 shows the parameters of the student model. We notice that both skills' learning and slip probabilities are very similar. We run the E-M algorithm using 100 different random initializations for both Dynamic Cog-

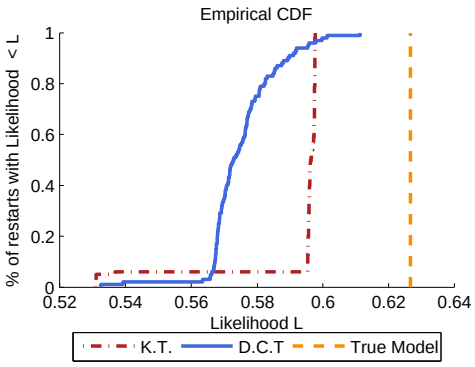


Figure 7: Cumulative Distribution Function of the Likelihood over 100 restarts (using the dataset highlighted in Figure 6)

Table 2: Model Comparison Over Number of Skills

	2 skills		3 skills		4 skills	
	Acc.	Lik.	Acc.	Lik.	Acc.	Lik.
True model	.75	.64	.75	.61	.76	.62
DCT	.74	.63	.73	.62	.73	.62
KT(1 skill)	.71	.61	.69	.59	.70	.60
Majority	.63	-	.66	-	.67	-

nitive Tracing and Knowledge Tracing. We use the same training set used for the highlighted dataset of Figure 6. To ensure more reliable results, we use a larger test set of 200 students (instead of 50 students). Figure 7 shows the Cumulative Distribution Function of the Likelihood over 100 random initializations. For a specific Likelihood ℓ in the horizontal axis, the vertical axis is the percentage of initializations with Likelihood found at a value less than or equal to ℓ. Figure 7 shows that the Likelihood of the true model is 62.6%. The best Likelihood of Dynamic Cognitive Tracing is 61.1%, and of single-skill Knowledge Tracing is 59.7%. Knowledge Tracing gets stuck in local optima in less than 5% of the restarts. On the other hand, for this dataset, Dynamic Cognitive Tracing gets stuck in local optima 99% of the time. While there is a Dynamic Cognitive Tracing solution that outperforms Knowledge Tracing, the E-M algorithm found it in 4% of the initializations.

In Table 2, we aggregate the results of Figure 6. We report the mean performance of the parameters that generate the 60 synthetic data sets (True model), Dynamic Cognitive Tracing, single-skilled Knowledge Tracing (KT), and the classifier that always predicts the majority class (Majority). We present the mean Classification Accuracy and the mean Likelihood. Dynamic Cognitive Tracing has a similar Likelihood and Classification Accuracy to the True Model and dominates Knowledge Tracing.

Let's study a sample cognitive model estimated using Dynamic Cognitive Tracing. Here Q^* is the True Model's cognitive model from which the synthetic data was generated. An estimate $\hat{Q}$, learned from data using our approach is:

$$Q^* = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \hat{Q} = \begin{pmatrix} 0.85 & 0.15 \\ 0.43 & 0.57 \\ 0.27 & 0.73 \\ 0.91 & 0.09 \end{pmatrix}$$

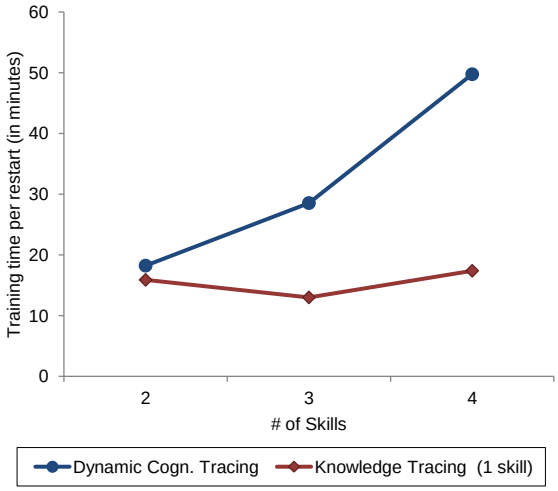


Figure 8: Time required (in mins.) to train a single restart

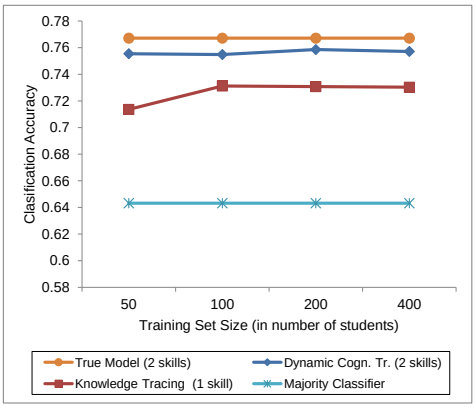


Figure 9: Classification accuracy using different training set sizes

The estimated cognitive model has some uncertainty, but if we round $\hat{Q}$ to integer values, it matches Q^*. In future work, we are interested in using Bayesian priors to encourage sparse entries in $\hat{Q}$ [13]. Bayesian estimation is not currently supported by the BNT toolkit in which we implemented our model.

In Figure 8 we show how long it took to perform a single restart of Dynamic Cognitive Tracing and Knowledge Tracing. Although Dynamic Cognitive Tracing achieves better accuracy, its exact inference implementation does not scale well with the number of skills.

We now try to simulate the effect of different amount of training data. For this, we experiment with 50, 100, 200 and 400 students. We observed that in the PSLC DataShop [17], a repository for student data sets, it is common for smaller datasets to have data from at least 50 students. We assess the performance of our approach using ten synthetic training sets with different number of students. For all experiments here, we used four different types of items (**Ids** = 4), and two skills (**S** = 2). In Figure 9, the "True model" line represents the classification accuracy of the model using the parameters from where the synthetic data was generated. The Knowledge Tracing line shows the performance of this

Table 3: Model Comparison Over Number of Items

	Ids = 4		**Ids** = 8		**Ids**=12	
	Acc.	Lik.	Acc.	Lik.	Acc.	Lik.
True model	.77	.66	.70	.60	.73	.61
DCT	.76	.65	.67	.58	.66	.57
KT(1 skill)	.73	.63	.64	.56	.66	.57
Majority	.66	-	.59	-	.58	-

approach, using a single skill. The results suggest that the approaches compared can achieve good performance even on a smaller datasets.

Since we are actually clustering similar items into skills, the number of different items (**Ids**) may have an impact on the performance of our approach. We create ten sets with 4, 8 and 16 item types respectively ($\mathbf{Id} = 4, 8, 16$). All of them have two skills ($\mathbf{S} = 2$). In Table 3, we summarize the Likelihood and the Classification Accuracy of different models. The true model's parameters achieve the highest likelihood, followed by our approach, that dominates Knowledge Tracing.

5. CONCLUSION

We propose Dynamic Cognitive Tracing as a novel unified approach to two problems previously addressed separately in Intelligent Tutoring Systems: (i) Student Modeling, which infers students' learning by observing student performance [9], and (ii) Cognitive Modeling, which factorizes problem solving steps into the latent set of skills required to perform them [7].

We provide empirical results using synthetic data supporting that our unsupervised approach is better than assuming that all items come from the same skills. Dynamic Cognitive Tracing significantly outperforms Knowledge Tracing using a single skill assumption.

We used the Bayesian Networks Toolkit to quickly prototype our approach. However, our prototype is limited in that (i) the inference algorithm used by the toolkit leads to complexity exponential in the number of skills, and (ii) the optimization algorithm gets stuck in local optima. We recommend implementing Dynamic Cognitive Tracing using approximate inference as future work.

For simplicity, in this paper we limited our study to synthetic data of items that require a single skill. However, our formulation is capable of discovering items that require multiple skills. It is an empirical question that we leave for future work to understand how well Dynamic Cognitive Tracing performs in this context.

We are also interested in comparing Dynamic Cognitive Tracing to other automatic methods that produce cognitive models from data, such as matrix factorization techniques [27]. An interesting alternative we leave unexplored is finding a cognitive model by first clustering items into skills, and then using Knowledge Tracing with the discovered cognitive model. However, it is not clear how to learn the skill clustering from data that comes at different points of time. For example, it is not obvious how PCA could be applied to temporal data. To our knowledge, we are the first ones to propose a fully-unsupervised method that combines student modeling with discovering a cognitive model.

Acknowledgements

This work was supported in part by the Institute of Education Sciences, U.S. Department of Education, through Grant R305A080628 to Carnegie Mellon University. José was partially supported by the Costa Rican Ministry of Science and Technology (MICIT). The opinions expressed are those of the authors and do not necessarily represent the views of the Institute or U.S. Department of Education. We thank the educators, students, and LISTENers who helped generate, collect, and analyze our data, and the reviewers for their helpful comments.

References

[1] R. Baker, A. Corbett, and V. Aleven. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In B. Woolf, E. Aïmeur, R. Nkambou, and S. Lajoie, editors, *Intelligent Tutoring Systems*, volume 5091 of *Lecture Notes in Computer Science*, pages 406–415. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-69130-3. URL `http://dx.doi.org/10.1007/978-3-540-69132-7_44`.

[2] T. Barnes. The q-matrix method: Mining student response data for knowledge. In J. Beck, editor, *Proceedings of AAAI 2005: Educational Data Mining Workshop*, pages 978–980, Pittsburgh, PA, 2005.

[3] T. Barnes, D. Bitzer, and M. Vouk. Experimental analysis of the q-matrix method in knowledge discovery. In M.-S. Hacid, N. Murray, Z. Ras, and S. Tsumoto, editors, *Foundations of Intelligent Systems*, volume 3488 of *Lecture Notes in Computer Science*, pages 11–41. Springer Berlin / Heidelberg, 2005. ISBN 978-3-540-25878-0. URL `http://dx.doi.org/10.1007/11425274_62`.

[4] J. Beck. Difficulties in inferring student knowledge from observations (and why you should care). In *Educational Data Mining: Supplementary Proceedings of the 13th International Conference of Artificial Intelligence in Education*, pages 21–30, Marina del Rey, CA, 2007.

[5] J. Beck and K.-m. Chang. Identifiability: A fundamental problem of student modeling. In C. Conati, K. McCoy, and G. Paliouras, editors, *User Modeling 2007*, volume 4511 of *Lecture Notes in Computer Science*, pages 137–146. Springer Berlin / Heidelberg, 2007. URL `http://dx.doi.org/10.1007/978-3-540-73078-1_17`.

[6] J. Beck and J. Mostow. How who should practice: Using learning decomposition to evaluate the efficacy of different types of practice for different types of students. In B. Woolf, E. Ameur, R. Nkambou, and S. Lajoie, editors, *Intelligent Tutoring Systems*, volume 5091 of *Lecture Notes in Computer Science*, pages 353–362. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-69130-3.

[7] H. Cen, K. Koedinger, and B. Junker. Learning factors analysis: A general method for cognitive model evaluation and improvement. In M. Ikeda, K. Ashley, and

T.-W. Chan, editors, *Intelligent Tutoring Systems*, volume 4053 of *Lecture Notes in Computer Science*, pages 164–175. Springer Berlin / Heidelberg, 2006. URL http://dx.doi.org/10.1007/11774303_17.

[8] A. Corbett. Cognitive computer tutors: Solving the two-sigma problem. In M. Bauer, P. Gmytrasiewicz, and J. Vassileva, editors, *User Modeling 2001*, volume 2109 of *Lecture Notes in Computer Science*, pages 137–147. Springer Berlin / Heidelberg, 2001. ISBN 978-3-540-42325-6. URL http://dx.doi.org/10.1007/3-540-44566-8_14.

[9] A. Corbett and J. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[10] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[11] M. Desmarais. Conditions for effectively deriving a q-matrix from data with non-negative matrix factorization. best paper award. In *EDM*, pages 41–50, Eindhoven, Netherlands, 2011.

[12] Z. Ghahramani and M. Jordan. Factorial hidden markov models. *Machine learning*, 29(2):245–273, 1997.

[13] S. Goldwater and T. Griffiths. A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P07-1094.

[14] Y. Gong, J. Beck, and N. Heffernan. Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. In V. Aleven, J. Kay, and J. Mostow, editors, *Intelligent Tutoring Systems*, volume 6094 of *Lecture Notes in Computer Science*, pages 35–44. Springer Berlin / Heidelberg, 2010. ISBN 978-3-642-13387-9.

[15] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. Bayesian updating in recursive graphical models by local computations. *Computational Statistical Quarterly*, (4): 269–282, 1990.

[16] M. Jordan and R. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2): 181–214, 1994.

[17] K. Koedinger, R. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper. *A data repository for the EDM community: The PSLC DataShop*. CRC Press, Boca Raton, FL, 2010.

[18] K. Murphy. The bayes net toolbox for matlab. *Computing science and statistics*, 33(2):1024–1034, 2001.

[19] P. Pavlik, H. Cen, and K. Koedinger. Performance factors analysis–a new alternative to knowledge tracing. In *Proceeding of the 2009 conference on Artificial Intelligence in Education: Building Learning Systems that Care: From Knowledge Representation to Affective Modelling*, pages 531–538. IOS Press, 2009.

[20] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. ISBN 0-934613-73-7.

[21] L. Rabiner and B. Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[22] J. Reye. Student modelling based on belief networks. *Int. J. Artif. Intell. Ed.*, 14:63–96, January 2004. ISSN 1560-4292. URL http://dl.acm.org/citation.cfm?id=1434852.1434856.

[23] A. Schein, L. Saul, and L. Ungar. A generalized linear model for principal component analysis of binary data. In C. M. Bishop and B. J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, pages 14–21, Key West, FL, 2003.

[24] A. P. Singh and G. J. Gordon. A unified view of matrix factorization models. In *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*, ECML PKDD '08, pages 358–373, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-87480-5. URL http://dx.doi.org/10.1007/978-3-540-87481-2_24.

[25] N. Thai-Nghe, L. Drumond, A. Krohn-Grimberghe, and L. Schmidt-Thieme. Recommender system for predicting student performance. *Procedia Computer Science*, 1 (2):2811 – 2819, 2010. ISSN 1877-0509. doi: 10.1016/j.procs.2010.08.006. URL http://www.sciencedirect.com/science/article/pii/S1877050910003194.

[26] M. Tipping and C. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[27] T. Winters, C. Shelton, T. Payne, and G. Mei. Topic extraction from item-level grades. In J. Beck, editor, *American Association for Artificial Intelligence 2005 Workshop on Educational Datamining*, Pittsburgh, PA, 2005.

[28] Y. Xu and J. Mostow. Using logistic regression to trace multiple subskills in a dynamic bayes net. In M. Pechenizkiy, T. Calders, C. Conati, S. Ventura, C. Romero, and J. Stamper, editors, *Proceedings of the 4th International Conference on Educational Data Mining*, pages 241–245, Eindhoven, Netherlands, 2011.