What and When Do Students Learn?
Methods For Knowledge Tracing With Data-Driven Mapping of Items to Skills

José Pablo González-Brenes

CMU-LTI-13-010

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies.*

Keywords: cognitive diagnostic model, student model, knowledge tracing, mixed-membership model, sequence clustering

Abstract

Recent interest in online education, such as Massively Open Online Courses and intelligent tutoring systems, promises large amounts of data from students solving items at different levels of proficiency over time. Existing approaches for inferring students' knowledge from data require a cognitive model – a mapping between the tutor problems and the set of skills they require. This is a very expensive requirement, since it depends on expert domain knowledge. The success of previous methods in using student performance data to construct this mapping automatically has been limited in that they cannot handle data collected over time, or that they require expensive expert domain knowledge. This dissertation studies how to model students' time varying knowledge, without requiring expert domain knowledge. We introduce four novel methods:

- Dynamic Cognitive Tracing: an easily implemented prototype that jointly estimates cognitive and student models.
- Automatic Knowledge Tracing: a method to discover first a mapping of items to skills and then a student model separately.
- Topical Hidden Markov Model: a fully Bayesian method to discover both cognitive and student models jointly.
- ItemClue: a method to bias the estimation of Topical Hidden Markov Model to discover more interpretable cognitive models by grouping similar items together.

We evaluate our methods on synthetic data and real student data collected from students interacting with two commercial tutoring systems. Our automatic methods require much less human work and can achieve significantly higher accuracy at predicting future student performance than the models handcrafted by experts.

Acknowledgments

I want to thank my mentor, Jack Mostow, for enabling and trusting me to follow my intellectual interests. From Jack I learned many lessons that I know I will apply for the rest of my life. Jack is not only a great researcher, but also an inspirational, kind-hearted person that I look up to. I also want to thank my thesis committee, Kenneth Koedinger, Emma Brunskill and Michel Desmarais. They were very kind and generous with their suggestions that dramatically improved this thesis.

I am very fortunate to have crossed paths with so many kind people in my life. Fu-Ren Lin was a great advisor that introduced me to research. Robert Frederking, Alan Black and Carolyn Rosé were great mentors that gave me a lot of advise.

Sometimes the line between friends and colleagues is blurry. I want to thank Kriti, Ramnath, Mladen, Miro, Sourish, Wei, Yanbo, Jessica, Kaimin and Sivaraman for their friendship and helpful discussions. I am very appreciative to my good friends that I have met in Costa Rica, Pittsburgh, and Taiwan.

I am thankful to my family for always being there. Their support is unconditional. I am very grateful for having such a special life partner. I wouldn't have done it without you.

Gracias.

Contents

Chapter 1

Introduction

In many instructional settings, students are graded by their performance on instruments such as exams or homework assignments. Usually, these instruments are made of *items* – questions, problems, parts of questions – which are graded individually. Recent interest in online education, such as Massively Open Online Courses and intelligent tutoring systems, promises large amounts of data from students solving items at different levels of proficiency over time. A challenge in modern educational technology is to use item-level data to build *student models*, an estimate of the student's understanding of the subject matter [VanLehn, 1988]. Student models are important because they enable adapting the instruction to the student's needs [Corbett and Anderson, 1995].

For example, suppose we are interested in understanding a student's reading ability using data from a reading tutor that listens to children read aloud. Figure 1.1 shows sample data in this scenario. We follow the convention of using the term item for anything for which per-student grades are recorded [Winters et al., 2005]. The value of the input variable x_t is the identifier of the item, which in this case is the word read by the student at time step t. The target variable y_t is the performance of the student – in this case whether the tutor accepted the word read. The student attempts to read the words "smile because it happened," but misreads the word "because." The student model can be evaluated to predict future student performance.

Assessing knowledge from item-level data requires a *cognitive diagnostic model* – a mapping of items to skills [Corbett and Anderson, 1995]. Cognitive diagnostic models are often built manually by context experts and psychologists — an effort that can take years [Corbett, 2013,

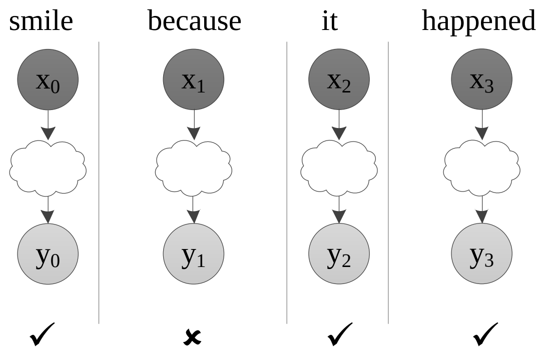


Figure 1.1: Reading tutor example of student modeling

Beck, 2007]. 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.

Methods to infer cognitive diagnostic models automatically have been restricted to static instruments administered to a student only once or twice. Current methods for discovering cognitive diagnostic models from student performance data are restricted due to (i) inability to handle *longitudinal data* – data collected over time, or (ii) not being fully automatic. For example:

- Matrix-based methods [Winters et al., 2005], such as Principal Component Analysis, Non-Negative Matrix Factorization and the Q-Matrix Method [Barnes, 2005], ignore the temporal dimension of the data. For a case where there are N students and M items, these methods model student performance in a $N \times M$ matrix. Such a matrix does not encode longitudinal data and is not able to represent that a student may encounter an item multiple times. Such matrix representations do not distinguish between bad performance at early time steps and bad performance after a lot of practice.

- On the other hand, semi-automatic approaches, such as Learning Factors Analysis [Cen et al., 2006], are designed for temporal data, but require an expert's cognitive model to improve upon.

In this dissertation we think of topics as the skills required to solve an item. Other authors have referred to skills as knowledge components [Cen et al., 2006] or topic skills [Desmarais, 2011]. Our contribution is a fully automatic approach to discover a cognitive model of longitudinal student data. Our goal is to discover student models, using an automatic clustering of items into skills.

1.1 What is an Item?

Previous work on fully automatic discovery of cognitive models has focused on static data – data without a temporal dimension. In this scenario, determining what is an item is straightforward: items are questions, problems or parts of questions [Winters et al., 2005]. However, dynamic data is often collected using an intelligent tutoring system, and the set of items presented to the student need not be determined a priori: a tutor may decide the items during the teaching activity. The methods we discuss in this thesis to discover cognitive diagnostic models assume each item has a unique identifier. We now discuss the design decisions we make to assign item identifiers.

We consider the format used by the PSLC DataShop[1] [Koedinger et al., 2010], a public repository for dozens of dynamic student-tutor datasets. The Datashop has developed a standard format to log dynamic student data. Although this format was not designed explicitly for automatic discovery of cognitive models, it is possible to use it for this purpose. The rich variety of tutors in the PSLC Datashop (dictation, geometry, algebra, physics, foreign language learning, etc.) proves that the format is general enough to support different tutors.

The PSLC Datashop is stored in relational database tables. It specifies columns for "problem name" and "step name." A Datashop problem is a task for a student to perform that typically involves multiple steps. A step is an observable part of solving to a problem. Step names are determined by the user interface available to the student for solving the problem. We consider three alternatives to model the identifier of items:

1. **Problem Name - Step Name:** Use the combination of problem and step name as identifiers. This strategy produces the most identifiers, and does not allow sharing steps across problems. This is the strategy used by Cen et al. [2006] to improve on existing cognitive models. This approach does not require an expert to name the items.

2. **Step Name:** Use the heuristic that items across different problems may use the same set of

[1]https://pslcdatashop.web.cmu.edu/

skills if they share the same step name. This approach requires an expert to name the items, but enables different problems to share steps, reducing the number of free parameters.

3. **Problem Name:** Use the problem names to identify items. In this case, the cognitive model discovered would be mapping the skills required to solve problems, not steps. This often produces the fewest identifiers of the strategies.

In this dissertation, we do not consider the step name strategy, building unique identifiers of step names, because it is not a fully automatic strategy.

1.2 Thesis Statement and Contributions

In this thesis we propose that data-driven methods for automatic skill discovery and tracing student knowledge can have performance comparable to manually engineered approaches that use domain knowledge. We quantify performance as how accurately the cognitive diagnostic model predicts future student performance.

1.3 Thesis Outline

The rest of this dissertation is organized as follows. Chapter 2 describes a feasibility study of the thesis statement. Chapter 3 describes our proposed methods to map items to skills. Chapter 4 describes a methodology to improve interpretability of the skills discovered. Chapter 5 describes preliminary follow-on work. Chapter 6 reviews related prior work. Chapter 7 provides some concluding remarks.

Chapter 2

Feasibility

In this chapter we describe our methodology to quickly prototype a system to support the thesis statement. The work described in this chapter extends previously published work:

- J. P. González-Brenes and J. Mostow. Dynamic Cognitive Tracing: Towards Unified Discovery of Student and Cognitive Models. In J. Stamper, K. Yacef, and O. Zaïane (editors). *Proceedings of the 5th International Conference on Educational Data Mining*, pages 49–56. Chania, Greece, 2012.

This chapter is organized as follows. Section 2.1 describes our approach, Dynamic Cognitive Tracing. Section 2.2 describes the experiments we conduct to evaluate Dynamic Cognitive Tracing.

2.1 Dynamic Cognitive Tracing

In this section we describe Dynamic Cognitive Tracing, a prototype of a method that is able to discover a cognitive diagnostic model and a student model simultaneously. The main advantage of Dynamic Cognitive Tracing is that it can easily be implemented using existing toolkits for Bayesian networks, such as the BNT toolkit [Murphy, 2001].

The rest of this section is organized as follows. Subsection 2.1.1 details our approach. Subsection 2.1.2 provides pointers on the training and inference algorithms used.

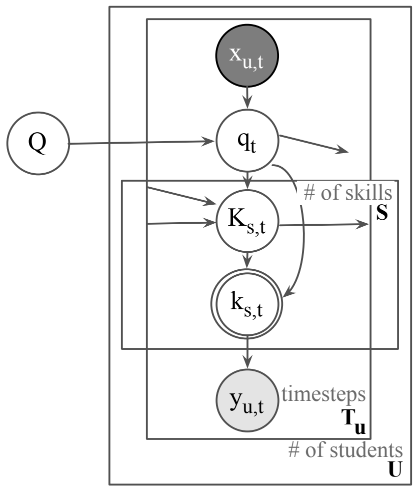
```
1. Draw $Q_x \sim$ Multinomial; $\mathbf{M}$ times
2. For each time-step $t \in \{0 \ldots T\}$:
    (a) Draw $x_t \sim$ Multinomial
    (b) For each skill $s \in \{0 \ldots \mathbf{S}\}$:
    (c) Set $q_{s,t} \leftarrow Q_{x_t}$
    (d) Draw $K_{s,t} \sim$ Binomial
    (e) Set $k_{s,t} \leftarrow K_{q_t} \cdot q_{s,t}$
    (f) $y_t \sim$ Binomial
```

Figure 2.1: Generative method of Dynamic Cognitive Tracing

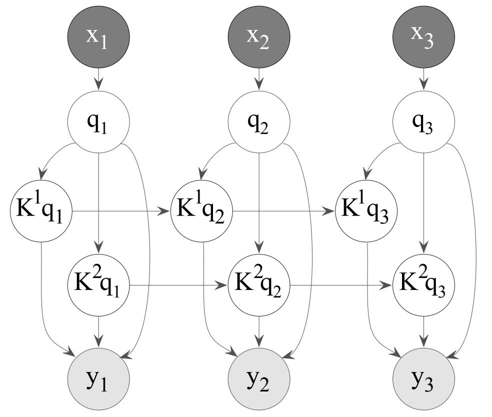
2.1.1 Model Specification

We formulate Dynamic Cognitive Tracing as a Bayesian Network. Bayesian Networks [Pearl, 1988], are a popular framework to reason using noisy information. Bayesian networks are directed acyclic graphical models where the nodes represent 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 2.2a. 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 2.2b. The description of the generative story of the variables is described in Figure 2.1. We follow the convention of using dark-gray to color variables that are observable during both training and testing. Variables visible during training only are colored in light gray. Latent variables, which are never observed, are denoted in white circles. The double-line around variables are used to indicate that their respective value is calculated deterministically given their parents.



(a) Plate diagram



(b) Unrolled example with two skills. For simplicity nodes k and q are drawn together

Figure 2.2: Dynamic Cognitive Tracing as a graphical model

The variables in Dynamic Cognitive Tracing are:

- $\mathbf{S}$ is the number of skills in the model.
- $\mathbf{M}$ is the number of items that the student can practice with the tutor. For example, in the case of a reading tutor, $\mathbf{M}$ is the vocabulary size. If the tutor is creating items dynamically, $\mathbf{M}$ is the number of templates from which items are being generated.
- x_t is the identifier of the item at time t.
- Q is an $\mathbf{M} \times \mathbf{S}$ matrix that maps items to skills. Each row Q_x is modeled as a multinomial representing the skills required for item x. For example, if $Q_{x_t} = [0.5, 0.5, 0, 0]$, we interpret item x_t to be a mixture of skills 1 and 2. In this example x_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 a skill drawn from the mixture Q_{x_t}. For example, $q_t = 1$ iff skill 1 is required for item x_t, $q_t = 2$ iff skill 2 is required, and so on. q_t is chosen deterministically as the row number x_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

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 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 x_t. Hence, its value is computed deterministically by applying dot product to the value of its parents: $k_{s,t}$ is true iff skill s is required ($q_t = s$), and the student has learned the skill ($K_{s,t} = 1$).
- y_t is the target variable that models performance. We represent performance as a binary variable (i.e., right or wrong), and we model it using a binomial distribution.

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

2.1.2 Training and Inference

We formulate Dynamic Cognitive Tracing as a directed graphical model (Bayesian Network). We leverage existing technologies to quickly implement a prototype of Dynamic Cognitive Tracing. We use the Bayesian Network Toolkit [Murphy, 2001] (BNT) for Matlab.

As described in Section 2.1.1, the knowledge of a skill is dependent on its value on the previous time step. This kind of dependency is called a Markov Chain. Therefore, in Dynamic Cognitive Tracing, the student knowledge of S skills is modeled using S layers of Markov Chains. Unfortunately, exact inference on layers of Markov Chains that produce a single output is intractable: the runtime complexity grows exponentially with the number of layers [Ghahramani and Jordan, 1997]. Hence, we limit our study to a small number of skills. In Chapter 3 we describe inference techniques that scale better using Gibbs Sampling.

The name Bayesian Network is a misnomer, because it does not require Bayesian Estimation, which uses random variables to model all sources of uncertainty (like the model's parameters). We use Maximum Likelihood Estimation to perform exact inference. BNT implements the Junction Tree algorithm [Jensen et al., 1989], an inference algorithm that generalizes the Forward-

Backward algorithm used in Knowledge Tracing and Hidden Markov Models [Rabiner and Juang, 1986]. To estimate the parameters of the model, we use the Expectation-Maximization (E-M) algorithm [Dempster et al., 1977]. Like all non-convex optimizers, E-M is not guaranteed to find the globally optimal solution.

2.2 Empirical Evaluation

In this section we describe experiments to validate Dynamic Cognitive Tracing. Section 2.2.1 describes experiments using synthetic data. Section 2.2.2 describes an experiment using real student data.

2.2.1 Synthetic data

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 from 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. We evaluate Dynamic Cognitive Tracing with the true number of skills.

Dynamic Cognitive Tracing aims to discover the skills automatically without supervision. We test whether the cognitive diagnostic model estimated by Dynamic Cognitive Tracing outperforms a cognitive diagnostic 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, the parameters are estimated using the same training set. Students in the testing and training sets do not overlap.

To generate the synthetic data sets, we use the generative story described in Figure 2.1, having each student encounter 25 items during training (sequence length = 25). In preliminary experiments, we noticed that by the 25th time step, most synthetic students learned. We sample randomly the sequence length of the test set, so we have a more balanced test set that has roughly the same number of correct and incorrect answers.

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 chapter we limit studying cognitive diagnostic 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 have zero *forget probability* (e.g., the transition probability from "knowing" to "not knowing" is zero).

We design the synthetic students to not have any initial knowledge. Beck and Chang [2007] 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.

We use E-M to learn the parameters of the models. We initialize the emission probabilities (slip and guess probabilities) of Dynamic Cognitive Tracing using the values of Knowledge Tracing with a single-skill model (S = 1). We initialize the other parameters of Dynamic Cognitive Tracing and Knowledge Tracing randomly. We use cross-validation to pick the best of 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 [Ghahramani and Jordan, 1997]. It measures how likely a model is to predict the test set. It penalizes high-confidence incorrect predictions more heavily. 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 $y_{i,t}$ 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:

$$\frac{\sum_i^I \sum_t^{T_i} \mathrm{p}(\hat{y}_{i,t} = y_{i,t} | \hat{y}_{i,t-1}, x_{i,t})}{\sum_i^I T_i}$$

- **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:

$$\frac{\sum_i^I \sum_t^{T_i} \delta(\mathrm{p}(\hat{y}_{i,t} = y_{i,t} | \hat{p}_{i,t-1}, x_{i,t}) > 0.5)}{\sum_i^I T_i}$$

In the next section, we report all of the different combinations of parameters we used.

2.2.1.1 Results

We created a total of 60 random synthetic datasets using the constraints explained in Section 2.2.1. All of them have four types of items (M $= 4$). We created twenty datasets with 2, 3 and, 4 skills (S $= 2, 3, 4$), respectively. As a proof of concept, we use Dynamic Cognitive Tracing with the correct number of skills.

In Figure 2.3, 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

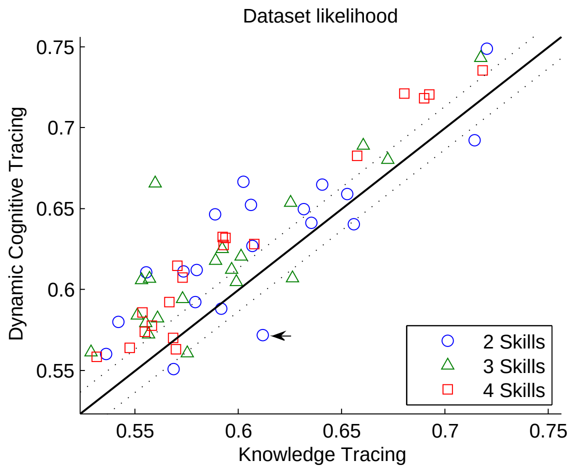


Figure 2.3: Average Likelihood of Dynamic Cognitive Tracing and single-skill Knowledge Tracing in 60 different data sets

Table 2.1: Dynamic Cognitive Tracing's worst performing dataset (highlighted in Figure 2.3)

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

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 diagnostic model with Dynamic Cognitive Tracing better than assuming a single skill model? Dynamic Cognitive Tracing should outperform a model that assumes that there are no distinguishable skills. Therefore, 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 Table 2.2, we aggregate the results of Figure 2.3. We report the mean performance of the parameters that generate the 60 synthetic data sets (True model), Dynamic Cognitive Tracing,

Table 2.2: 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	-

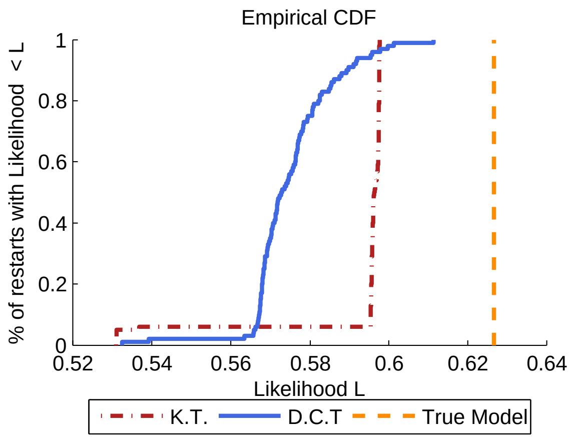


Figure 2.4: Cumulative Distribution Function of the Likelihood over 100 random initializations (using the dataset highlighted in Figure 2.3)

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 likelihood and classification accuracy similar to the True model and dominates Knowledge Tracing.

In Figure 2.3 the arrow points to the dataset that performs the worst compared to the single-skill Knowledge Tracing baseline. The likelihoods of the True model, Dynamic Cognitive Tracing, and single-skill Knowledge Tracing are 65%, 57%, and 61%, respectively. We try to understand when Knowledge Tracing outperforms Dynamic Cognitive Tracing, by studying this specific dataset. Table 2.1 shows the parameters of the student model. We notice that the learning and slip probabilities of the two skills are very similar. We run the E-M algorithm using 100 different random initializations for both Dynamic Cognitive Tracing and Knowledge Tracing. We use the same training set used for the highlighted dataset of Figure 2.3. To ensure more reliable results, we use a larger test set of 200 students (instead of 50 students). Figure 2.4 shows the Cumulative Distribution Function of the likelihood over 100 random initializations. For a specific likelihood ℓ on the horizontal axis, the vertical axis is the percentage of initializations with likelihood found at a value less than or equal to ℓ. Figure 2.4 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%. The inflection on the Knowledge Tracing line shows that it gets stuck in local optima in less than 5% of the restarts, and most initializations have likelihood 59.7%. On the other hand, for this dataset, the smooth curve of Dynamic Cognitive Tracing shows that it 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 only 4% of the initializations.

To give a taste of how Dynamic Cognitive Tracing works, we now show a sample cognitive diagnostic model. We refer to the cognitive diagnostic model from the ground truth and the one discovered by our data-driven Dynamic Cognitive Tracing as Q^* and $\hat{Q}$, respectively:

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

The estimated cognitive model has some uncertainty, but if we round $\hat{Q}$ to integer values, it matches Q^*. In the next chapter, we are interested in using Bayesian priors to encourage sparse entries in $\hat{Q}$ [Goldwater and Griffiths, 2007]. Bayesian estimation is not currently supported by the BNT toolkit in which we implemented our model.

In Figure 2.5 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 amounts of training data. For this, we experiment with 50, 100, 200, and 400 students. We observed that in the PSLC DataShop [Koedinger et al., 2010], 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 numbers of students. For all experiments here, we used four different types

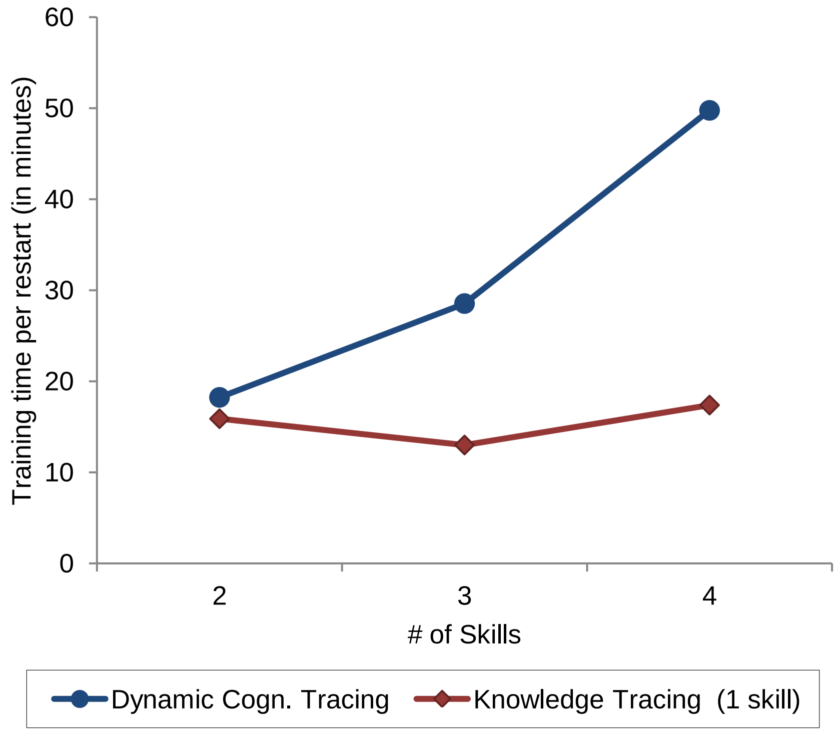


Figure 2.5: Time required (in mins.) to train a single restart

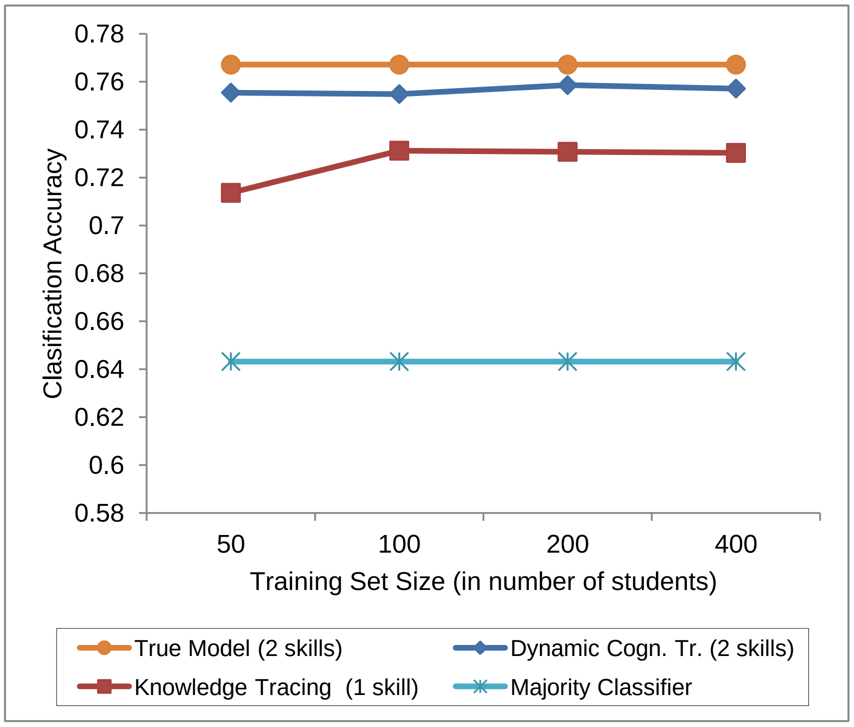


Figure 2.6: Classification accuracy using different training set sizes

Table 2.3: Model Comparison Over Number of Items

	$\mathbf{M} = 4$		$\mathbf{M} = 8$		M=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	-

Table 2.4: Model Comparison on Real-Student Data

	Acc.	Lik.
DCT (2 skills)	.81	.73
KT(1 skill)	.80	.71

of items ($\mathbf{M} = 4$), and two skills ($\mathbf{S} = 2$). In Figure 2.6, the True model line represents the classification accuracy of the model using the parameters from which the synthetic data was generated. The Knowledge Tracing line shows the performance of this 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 items into skills, the number of different items ($\mathbf{M}$) may have an impact on the performance of our approach. We create ten sets with 4, 8 and 16 item types respectively ($\mathbf{M} = 4, 8, 16$). All of them have two skills ($\mathbf{S} = 2$). In Table 2.3, we summarize the likelihood and the classification accuracy of different models. The True model's parameters achieve the highest likelihood, followed by our approach, which dominates Knowledge Tracing. This experiment provides evidence that discovering a cognitive diagnostic model using data can improve predictions of student performance.

2.2.2 Algebra Tutor data

We now describe how we validate Dynamic Cognitive Tracing on real student data. We use a dataset of students interacting with Bridge to Algebra Cognitive Tutor® [Koedinger et al., 2010]. We select the ten most popular problems of the dataset. This gives us a total of 85 students who attempted to solve these problems. We encode the identifier of items using the "problem name" column (see Section 1.1 for a discussion). We compare single-skill Knowledge Tracing, with

Dynamic Cognitive Tracing discovering two skills. We use 10 random restarts for each approach. We perform 10-fold cross-validation: we repeat the experiments ten time using nine folds to train the methods and the remaining fold to estimate performance. The folds are the same for the two approaches. Table 2.4 shows the mean classification accuracy, and the mean per-item likelihood across the folds.

We now test whether if estimating a 2-skill cognitive model with Dynamic Cognitive Tracing is better than assuming a single skill model. We compare the mean likelihood of Dynamic Cognitive Tracing ($\bar{x}_{\text{DCT}} = 72.95, s_{\text{DCT}} = 5.14$), with the mean likelihood of single-skill Knowledge Tracing ($\bar{x}_{\text{KT}} = 71.30, s_{\text{KT}} = 3.90$). 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 corresponding cross-validation folds (n=10). We reject the null hypothesis H_0 at the 95% confidence level. We conclude that discovering a 2-skill model with Dynamic Cognitive Tracing outperforms Knowledge Tracing with a single skill assumption in this dataset. The average improvement of using Dynamic Cognitive Tracing ($\bar{x}_{\text{DCT}} = 72.95$) over Knowledge Tracing ($\bar{x}_{\text{KT}} = 71.30$) is of 42% of the standard deviation ($s_{\text{KT}} = 3.90$) of the baseline.

2.3 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 [Corbett and Anderson, 1995], and (ii) Cognitive Modeling, which factors problem solving steps into the latent set of skills required to perform them [Cen et al., 2006].

We provide empirical results using synthetic data supporting the hypothesis that our unsupervised approach is better than assuming that all items come from the same skill. 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.

Chapter 3

Mapping Items to Skills

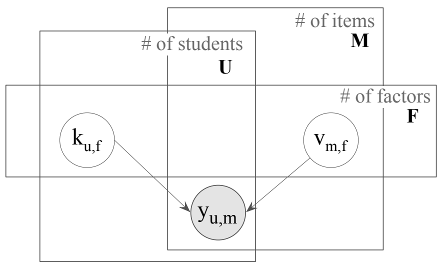
In this chapter we present two novel methods that discover a cognitive and student model from longitudinal data. We follow an evaluation methodology previously proposed [González-Brenes and Mostow, 2011] for comparing a method that discovers separate models to a method that discovers a joint model: Automatic Knowledge Tracing discovers a cognitive diagnostic model and then a student model, while Topical Hidden Markov Model (HMM) is a joint model.

The rest of this chapter is outlined as follows: Section 3.1 describes Automatic Knowledge Tracing; Section 3.2 describes Topical Hidden Markov Model; Section 3.3 describes empirical evaluation; Section 3.4 concludes.

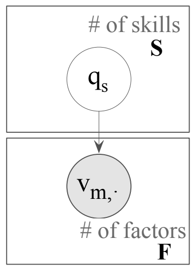
The description of Topical HMM in this chapter extends previously published work:

- J. P. González-Brenes and J. Mostow. Topical HMMs for Factorization of Input-Output Sequential Data. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *paper presented at Personalizing Education Workshop in Advances in Neural Information Processing Systems (NIPS'12)*, Lake Tahoe, CA, 2012

- J. P. González-Brenes and J. Mostow. What and When do Students Learn? Fully Data-Driven Joint Estimation of Cognitive and Student Models. In A. Olney, P. Pavlik, and A. Graesser, editors, *Proceedings of the 6th International Conference on Educational Data Mining*, pages 236–240, Memphis, TN, 2013



(a) Plate diagram of Matrix Factorization. $v_{m,f}, k_{u,f}$ are entries in the latent matrices $\vec{\mathbf{v}}$ and $\vec{\mathbf{k}}$, respectively, used to approximate $\vec{\mathbf{y}}$



(b) Plate diagram of clustering

Figure 3.1: Components of the Automatic Knowledge Tracing pipeline

3.1 Automatic Knowledge Tracing

Automatic discovery of cognitive diagnostic models on static data has used matrix factorization techniques, such as Non-Negative Matrix Factorization [Lee and Seung, 1999]. We now describe our attempt to use matrix factorization to trace student-varying knowledge of skills. We formulate Automatic Knowledge Tracing as a pipeline: we first use matrix factorization and clustering to discover a cognitive diagnostic model, and then we use Knowledge Tracing [Corbett and Anderson, 1995].

We first summarize matrix factorization, a family of algorithms that map both student and items to a joint latent factor space of dimensionality F inferred from performance patterns. Figure 3.1a shows Matrix Factorization using graphical model notation. Rather than repeat a variable, we follow the convention of using a plate to group repeated variables in Figure 3.1a. Variables visible only during training are in light gray. Latent variables, which are never observed, are denoted in white circles.

Performing matrix factorization on matrix $\mathbf{Y}$ requires estimating $\mathbf{K}$ and $\mathbf{V}$ such that:

$$\hat{y}_{u,m} = g(\vec{\mathbf{v}}_m \cdot \vec{\mathbf{k}}_u) \tag{3.1}$$

here $\vec{\mathbf{v}}_m, \vec{\mathbf{k}}_u \in \mathbb{R}^F$. For a given item m, the elements of $\vec{\mathbf{v}}_m$ measure the extent to which the item requires latent abilities (factors); for a given student u, the elements of $\vec{\mathbf{k}}_u$ measure the extent to which the student u has the latent abilities; g is a link function. Singh and Gordon [2008] showed that different matrix factorization algorithms can be derived with a specific choice of g, under certain constraints. For example, Conventional Matrix Factorization [Koren et al., 2009] uses a linear link function and Binary Matrix Factorization [Zhang et al., 2007] uses a logistic link function.

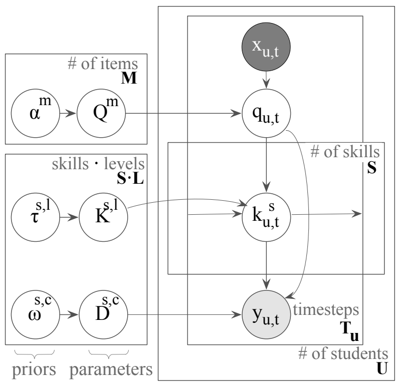
3.1.1 Model Specification

Algorithm 1 Automatic Knowledge Tracing

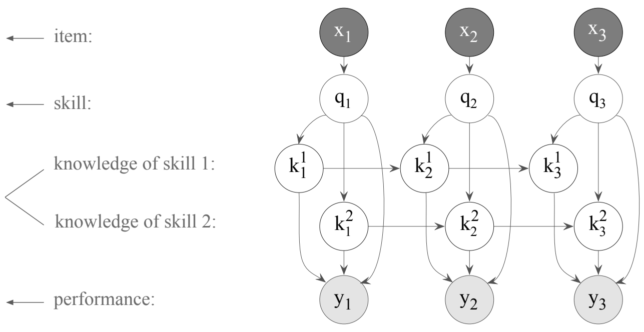
Require: A sequence of item identifiers $x_{1...T_u}$ for $\mathbf{U}$ users, number of skills $\mathbf{S}$, number of student states $\mathbf{L}$, number of items $\mathbf{M}$

```
 1: function AUTOMATIC KNOWLEDGE TRACING($\{y_{u,1} \dots y_{u,T_u} : u \in 1 \dots U\}, S$)
 2:     // First part of analysis does not consider time:
 3:     $\mathbf{Y}' = \varnothing$
 4:     for each user $u$ do
 5:         for each time step $t$ do
 6:             if $y_{x_{u,t}} = \varnothing$ then //  Take first observation of item
 7:                 $\mathbf{Y}'_{u,x_{u,t}} \leftarrow y_{u,t}$
 8:     // Discover skills:
 9:     $\langle \mathbf{K}, \mathbf{V} \rangle = \mathrm{matrix\_factorization}(\mathbf{Y}', \mathrm{S})$
10:     $\langle \mathbf{v} \rightarrow \mathbf{q} \rangle = \mathrm{cluster}(\vec{v_1}, \dots, \vec{v_M}, S)$ // Map each factor $v$ into a cluster $q$
11:     // Do conventional Knowledge Tracing:
12:     for each skill $s \leftarrow 1 \dots S$ do
13:         $y'' \leftarrow \varnothing$
14:         for each user $u \leftarrow 1 \dots U$ do
15:             for each time $t \leftarrow 1 \dots T_U$ do
16:                 if $\langle v_{x_{u,t}} \rightarrow s \rangle \in \langle \mathbf{v} \rightarrow \mathbf{q} \rangle$ then
17:                     $y''_{s,u} \leftarrow \mathbf{Y}'_{u,x_{u,t}}$
18:         $\mathrm{train\_hmm}(\mathbf{y''_{s,u}})$
```

Algorithm 1 describes the details of the Automatic Knowledge Tracing algorithm we propose: Let U be the number of students, and M be the number of items. Lines 2–7 build a $U \times M$ matrix $\mathbf{Y}'$ where each entry is the performance of a student solving an item. For simplicity, in case a student solves an item multiple times, we only take into account the first encounter of the item. We do not explore using other descriptive statistics such as the last encounter, the average or the median. Line 9 performs matrix factorization to approximate the entries of the matrix $\mathbf{Y}$. Line 10 uses a clustering algorithm to group items with similar factors together. A graphical



(a) Plate diagram of Topical Hidden Markov Models



(b) Unrolled example of Topical HMM with two skills ($\mathbf{S} = 2$), for a single user ($\mathbf{U} = 1$) with three time steps ($\mathbf{T_1} = 3$). Student indices, parameters (Q, K, D) and priors (α, τ, ω) are omitted for clarity

Figure 3.2: Topical HMM as a graphical model

model representation of the clustering is in Figure 3.1b, where we assume that the factors are observed. Lines 11–18 perform Knowledge Tracing by training a Hidden Markov Model on the sequences of the student performance practicing a skill.

3.2 Topical Hidden Markov Model

We formulate Topical Hidden Markov Model (Topical HMM) as a mixed membership model: we use soft classification of items into skills using latent variables. For example, in Latent Dirichlet Allocation [Blei et al., 2003], a popular mixed-membership model, a multinomial encodes a document as a mixture of topics. Similarly, Topical HMM uses a multinomial to encode an item as a mixture of skills.

We describe the formulation of Topical HMM and its assumptions in Section 3.2.1, and the Blocked Gibbs Sampler algorithm we use for inference in Section 3.2.2.

3.2.1 Model Specification

Graphical models are described using plate diagram notation and a generative story; we show these formalisms for Topical HMM in Figure 3.2 and Algorithm 2, respectively. We unroll Topical HMM using two skills in Figure 3.2b. The hyper-parameters in Topical HMM are:

- $\mathbf{S}$ is the number of skills in the model.
- $\mathbf{U}$ is the number of users (students).
- $\mathbf{T_u}$ is the number of time steps student u practiced.
- $\mathbf{M}$ is the number of items. For example, in the case of a reading tutor, $\mathbf{M}$ may represent the vocabulary size. If the tutor is creating items dynamically, we assume that $\mathbf{M}$ can be calculated as the number of templates from which items are being generated.
- $\mathbf{L}$ is the number of levels a student can master. For example, if we consider that students can be novice, medium or expert, we would use $\mathbf{L} = 3$. We only use two levels of mastery (knowledge or not) following previous convention [Corbett and Anderson, 1995].

The discrete variables are:

- $x_{u,t}$ is the item the student u practiced at time t.
- q_t is the skill required for item $x_{u,t}$. For example, $q_t = 1$ iff skill 1 is required for item $x_{u,t}$, $q_t = 2$ iff skill 2 is required, and so on. The value of q_t is being drawn by the multinomial $Q^{x_{u,t}}$, which allows for soft membership of skills.
- $k^s_{u,t}$ is a variable that takes values from $1 \dots L$ that represents the level of knowledge for skill s. There is a Markovian dependency across time steps: if skill s is known at time $t-1$, it is likely to be known at time t. We impose a deterministic constraint that the knowledge of a skill can only change its value while the student exercises the skill. Thus, we assume there is no transfer between skills [Koedinger and Roll, 2012].
- $y_{u,t}$ is the target variable that models performance. It is only observed during training.

Since we take a fully Bayesian approach, we model parameters as random variables:

- $Q^{x_{u,t}}$ is a multinomial representing the skills required for item $x_{u,t}$. For example, if $Q_{x_{u,t}} = [0.5, 0.5, 0, 0]$, we interpret item $x_{u,t}$ to be a mixture of skills 1 and 2, and not needing skills 3 and 4. Therefore, Topical HMM is able to assign multiple skills per item. Xu and Mostow [2012] compared the multiple strategies used in the literature to model multiple skills per item. They argue that considering multiple skills simultaneously works better than previous work that trains each skill separately and assigns full responsibility according to a function (typically by using either multiplication, the minimum or the maximum). Topical HMM is similar to LR-DBN [Xu and Mostow, 2012] in that it considers multiple skills simultaneously: Topical HMM considers items a convex combination of skills, while LR-DBN considers items a log-linear combination of skills. A convex combination of skills means that items are represented as a linear combination of skill weights where all weights are non-negative and sum to one. Unlike LR-DBN, Topical HMM allows Q to be be hidden. However, if an expert designs a cognitive diagnostic model, Q can be clamped to the values assumed by the expert.

- $K^{s,l}$ is the probability distribution for transitioning from knowledge state l of skill s at one time step to a knowledge state at the next time step.
- $D^{s,l}$ is the emission probability of the performance for skill s and knowledge state l.

For simplicity, the distribution we use to model the priors α, τ, ω of the parameters is the Dirichlet. The Dirichlet distribution is conjugate to the multinomial distribution, which means that the posterior of the distribution is also a Dirichlet – and therefore it is easy to calculate. The values of the priors are assumed to be given.

Topical HMM assumes that the knowledge of a skill cannot change while the skill is not being practiced, and that the knowledge of skills do not change while the skill is not being practiced (there is no transfer of learning between skills). These assumptions make possible an efficient sampler.

3.2.2 Blocked Gibbs Sampling Inference

We use Gibbs Sampling to infer the posteriors over random variable values. We will provide just a summary of Gibbs Sampling [Besag, 2004, Resnik and Hardisty, 2010]: we sample each hidden node conditioned on its Markov blanket (parents, children, and co-parents). After a burn-in period, if some loose requirements are met, samples will converge to the true joint distribution.

Algorithm 2 Generative story of Topical HMM

```
Require: A sequence of item identifiers $x_{1...T_u}$ for $\mathbf{U}$ users, number of skills $\mathbf{S}$, number of student states $\mathbf{L}$, number of items $\mathbf{M}$

function TOPICAL HMM($\{x_1 \ldots x_{T_u} : u \in 1 \ldots U\}, \mathbf{S}, \mathbf{L}, \mathbf{M}$)
    // Draw parameters from priors:
    for each skill $s \leftarrow 1$ to $\mathbf{S}$ do
        for each knowledge state $l \leftarrow 1$ to $\mathbf{L}$ do
            Draw parameter $K^{s,l} \sim$ Dirichlet($\tau^{s,l}$)
            Draw parameter $D^{s,l} \sim$ Dirichlet($\omega^{s,l}$)
    for each item $m \leftarrow 1$ to $\mathbf{M}$ do
        Draw $Q^m \sim$ Dirichlet($\alpha$)
    // Draw variables from parameters:
    for each student $u \leftarrow 1$ to $\mathbf{U}$ do
        for each timestep $t \leftarrow 1$ to $\mathbf{T}_u$ do
            Draw skill $q_{u,t} \sim$ Multinomial($Q^{x_{u,t}}$)
            for $s \leftarrow 0$ to $\mathbf{S}$ do
                if $s = q_{u,t}$ then
                    // knowledge state could change:
                    $k'' \leftarrow k^s_{u,t-1}$ // previous knowledge
                    Draw $k^s_{u,t} \sim$ Multinomial($K^{s,k''}$)
                else
                    // knowledge state can't change:
                    $k^s_{u,t} \leftarrow k^s_{u,t-1}$
            $q' \leftarrow q_{u,t}$ // current skill
            $k' \leftarrow k^{q_{u,t}}_{u,t}$ // current knowledge state
            Draw performance $y_{u,t} \sim$ Multinomial($D^{q',k'}$)
```

We follow a fully Bayesian treatment: given that the model's parameters themselves are unknown quantities, we treat them as random variables. Therefore, to train the model, we also use Gibbs Sampling to infer the posterior over the parameter values. In our experimental evaluation, we never use data from the test set to update parameters. We now describe how we sample the posterior of random variables values.

3.2.2.1 Pointwise Sampling of Parameters

Because we model priors using the Dirichlet distribution and parameters with the multinomial distribution, the posterior distribution of the parameters is a Dirichlet distribution parameterized by the observed evidence plus the priors' hyperparameters [Resnik and Hardisty, 2010]. For example, let's consider how to sample parameter Q^m. Let δ be the Kronecker's function that is 1 iff its arguments are true. For notational convenience, let's define the S dimensional vector $\tilde{\boldsymbol{\alpha}}^{\boldsymbol{m}}$ such that each entry $\tilde{\alpha}^m_{(s)}$ is the empirical count of item m being assigned to skill s:

$$\tilde{\alpha}^m_{(s)} = \sum_u \sum_t \delta(q_{u,t} = s, x_{u,t} = m) \tag{3.2}$$

Then, to sample a new value of Q^m we draw:

$$Q^m \sim \text{Dirichlet}(\underbrace{\tilde{\boldsymbol{\alpha}}^{\boldsymbol{m}}}_{\text{empirical count}} + \underbrace{\boldsymbol{\alpha}^{\boldsymbol{m}}}_{\text{prior}}) \tag{3.3}$$

3.2.2.2 Pointwise Sampling of *q* nodes

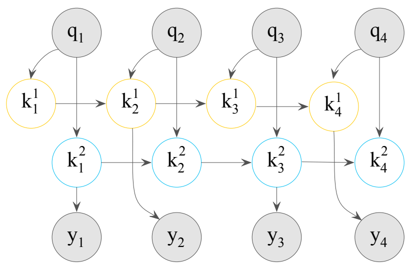
We now show how to sample the skill nodes. Suppose we are using Topical HMM to model two skills: multiplication and subtraction. Imagine a student who is an expert at subtraction, but a novice in multiplication. If this student gets an item wrong, it is likely that the item is testing a skill the student does not know, in this case, multiplication.

More formally, to sample skills, we need to condition on the parameters Q, D, and the student knowledge at the current time step k_t. We sample a value $\mathbf{q}$ proportionally to:

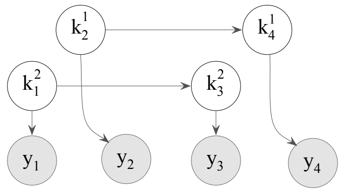
$$p(q_t = \mathbf{q}) \propto \sum_s \underbrace{p(y_t|k^s, D^{\mathbf{q},k^s_t})}_{\text{empirical count}} \underbrace{p(\mathbf{q}|Q^{x_t})}_{\text{prior}} \tag{3.4}$$

3.2.2.3 Blocked Sampling of *k* nodes

Pointwise Gibbs Sampling requires that all of the sampled conditional probabilities should be non-zero [Goldwater, 2007]. Unfortunately, this requirement is not met in Topical HMM: in line 19 of Algorithm 2, the `if` statement operationalizes the assumption that the knowledge of a skill can only be assessed while the student is exercising the skill. This deterministic constraint effectively sets some conditional probabilities to zero. Therefore, we need to use a blocked sampler instead of a pointwise sampler: our algorithm samples all the knowledge nodes simultaneously as a block, updating all the deterministic values simultaneously.



(a) Markov blanket for nodes knowledge nodes $k_1 \dots k_4$



(b) Markov blanket of nodes $k_1 \dots k_4$ after removing nodes that do not change their value because of deterministic constraints

Figure 3.3: Sample Markov blanket for the nodes k of a 2-skill Topical HMM when $q_1 = 1, q_3 = 1$ and $q_2 = 2, q_4 = 2$. Parameter nodes are omitted for clarity.

Let's illustrate our strategy with an example. Gibbs sampling, like the Expectation Maximization algorithm [Dempster et al., 1977], iterates through all the latent random variables of the model, and assumes that all other random variables are observed. As an example, when we are sampling the knowledge random variables k^s, we can assume that all other random variables are observed. Figure 3.3a shows a two-skill Topical HMM for the purpose of sampling k: it assumes that the skill nodes are observed. In this example, skill 1 is used at time 1 and 3, and skill 2 is used at time 2 and 4 ($q_1 = 1, q_3 = 1$ and $q_2 = 2, q_4 = 2$). When the skills are observed, some dependencies can be removed because they are outside of the Markov blanket. If we remove nodes that are not allowed to change their value across time (because the skill is not being used), we end up with only the dependencies of Figure 3.3b: when the skill nodes are observed, Topical HMM is equivalent to an HMM per skill. For example, in the figure, a two-skill Topical HMM reduces to two HMMs. Sampling the latent states of a HMM using Gibbs sampling is straightforward, as we can use the Forward-Backward probabilities [Besag, 2004].

In general, to sample the knowledge nodes on a S-skill Topical HMM, we build S HMMs, removing the knowledge nodes of skills that are not being exercised. We build the s$^{\text{th}}$ HMM using all the nodes k_t^s such that $q_t =$ s.

3.2.2.4 Pointwise sampling of y nodes

During sampling, the value of the performance y_t is drawn proportionally to the emission probability D, conditioning on each skill $s = q_t$ and knowledge state $l = k_t^{q_t}$:

$$y_t \propto D^{q_t, k_t^{q_t}} \tag{3.5}$$

Since Gibbs sampling requires a large number of samples to estimate the posterior of a random variable, during evaluation we also experimented with calculating the posterior of the random variable y_t directly, marginalizing over the student knowledge and the skills, as follows:

$$p(y_t = \mathbf{y}) = \sum_s \sum_l p(q_t = s) p(y_t = \mathbf{y} | k^{q_t} = l) \tag{3.6}$$

However, we did not find any significant differences between the predictive performance of sampling (Equation 3.5) and marginalizing skills and knowledge (Equation 3.6). Therefore, for our experiments we only report the results of sampling.

3.3 Evaluation

We now describe the experiments we conducted with data collected from real students (Section 3.3.1), and from synthetic students (Section 3.3.2).

3.3.1 Predicting Future Student Performance of Real Students

We evaluate cognitive diagnostic model strategies by how accurately they predict future student performance. We operationalize predicting future student performance as the classification task of predicting which students correctly solved the items in a held-out set. There is no consensus in the Educational Data Mining community on how to set up such experiments for this task. For example, previous work has used as test set the last question of each problem set [Pardos and Heffernan, 2010], or the second half of the encounters [Xu and Mostow, 2011]. In this chapter, we are focusing on predicting performance on unseen students, which is arguably a harder task. To make predictions on the development and test set we use the history preceding the time step we want to predict. For example, to predict on the third time step, we make use of all the data up to the second time step. Because the model does not update its beliefs with a new observation, our implementation makes very inefficient use of memory. For example, we process a student with three observations like three pseudo-students. The first pseudo-student has only one observation, the second has two, and the third has three. To speed up computations and save memory, in our experiments we predict up to the 150th time step in the development set, but we predict up to the 200th time step in the test set.

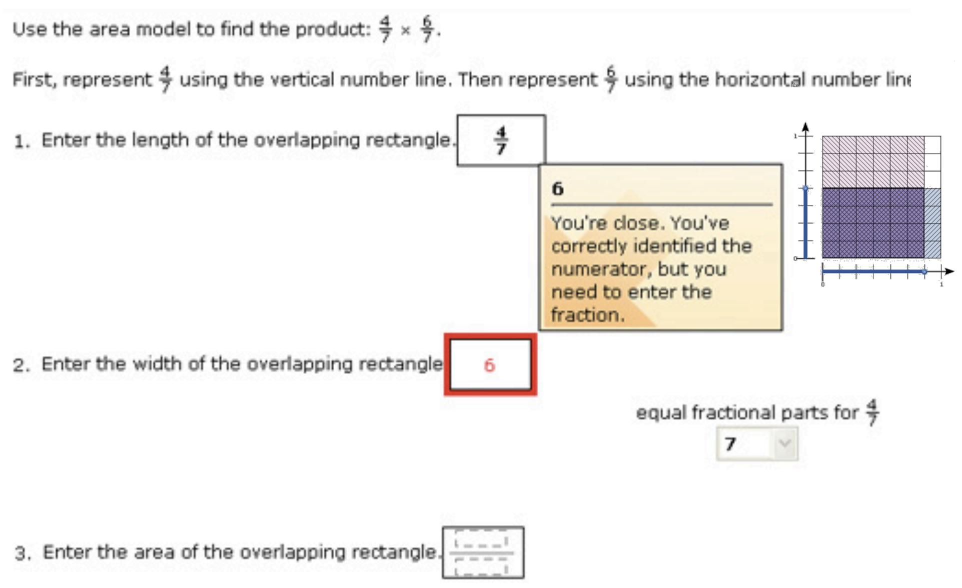
We evaluate the models' predictions using a popular machine learning metric, the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve. An AUC of 1 represents a perfect classifier; an AUC of 0.5 represents a useless classifier. AUC estimates can be interpreted as the probability that the classifier will assign a higher score to a randomly chosen positive example than to a randomly chosen negative example. The AUC metric is especially useful when a dataset has one class that is much larger than the other.

For our experiments with Automatic Knowledge Tracing and Topical HMM, we initialize the models randomly and then collect 2,000 samples of the Gibbs Sampling Algorithm described in Section 3.2.2. We discard the first 500 samples as a burn-in period. To infer future student performance, we use the last 1,500 samples given by Equation 3.5 on page 27.

The rest of the section is organized as follows: Section 3.3.1.1 describes the two datasets we use in our analysis; Section 3.3.1.2 describes our use of Bayesian priors; Section 3.3.1.3 describes experiments showing the effect of changing the number of skills in the models; Section 3.3.1.4 reports the computation time required to run our models; Section 3.3.1.5 compares Topical HMM and Automatic Knowledge Tracing to other models. Section 3.3.1.6 describes the cognitive diagnostic model discovered by Topical HMM.

3.3.1.1 Datasets

We now describe the two datasets of real students interacting with an intelligent tutoring system we used for our analysis. The format of these datasets follows the PSLC Datashop format [Koedinger et al., 2010]. Each problem is divided into a sequence of one or more steps, and it is at this level that we do our analysis: We consider the different steps as the items the student is solving. To provide an item identifier, we use the strategy of combining problem and step


Use the area model to find the product: $\frac{4}{7} \times \frac{6}{7}$.

First, represent $\frac{4}{7}$ using the vertical number line. Then represent $\frac{6}{7}$ using the horizontal number line

1. Enter the length of the overlapping rectangle. $\frac{4}{7}$

6

You're close. You've correctly identified the numerator, but you need to enter the fraction.

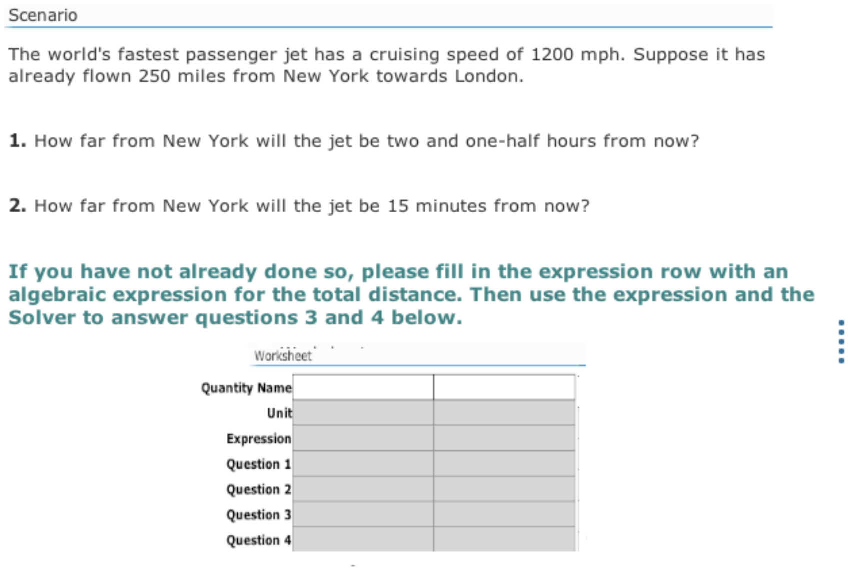
2. Enter the width of the overlapping rectangle 6

equal fractional parts for $\frac{4}{7}$

7

3. Enter the area of the overlapping rectangle.

(a) Bridge to Algebra Tutor®


Scenario

The world's fastest passenger jet has a cruising speed of 1200 mph. Suppose it has already flown 250 miles from New York towards London.

1. How far from New York will the jet be two and one-half hours from now?

2. How far from New York will the jet be 15 minutes from now?

If you have not already done so, please fill in the expression row with an algebraic expression for the total distance. Then use the expression and the Solver to answer questions 3 and 4 below.

Worksheet

Quantity Name		
Unit		
Expression		
Question 1		
Question 2		
Question 3		
Question 4		

(b) Algebra I Linear Tutor

Figure 3.4: Screenshots of two intelligent tutoring systems

name discussed in Section 1.1 (page 3). Students do not follow the curriculum in the same order; the tutor decided which items the students solved in which order. Figure 3.4 shows modified screenshots of the tutors, modified to fit on paper.

The datasets already have expert cognitive diagnostic models engineered using domain knowledge. Unfortunately we do not have an estimate of how long it took an expert to build a cognitive diagnostic model for this datasets, but if we make a conservative estimate that on average each item can be read and classified into the skills it requires in twenty seconds, it would take a human up to 27 hours of work to process the 5,000 items which appear in one of the datasets. Corbett [2013] privately gave a time estimate of the effort to build the items and the cognitive diagnostic model: creation and refinement took over four years for a team of two cognitive scientists and a teacher; but presumably building the item bank is much more time consuming than constructing the cognitive diagnostic model.

The expert models includes some items that use multiple skills. Appendix 7.1 lists the skills that experts defined.

We use a development set to tune hyper-parameters (value of the priors) and select the number of skills to model the data. The test set was only evaluated once just before writing this chapter—no tuning was done on the test set. We learn the parameters of the model exclusively on the training set. We only report results on the development or test set.

Carnegie Learning Bridge to Algebra Cognitive Tutor® Data Set We use data collected by the Carnegie Learning Bridge to Algebra Cognitive Tutor® [Koedinger et al., 2010]. This tutor aims to improve students' foundational skills and prepares them for an Algebra I class.

This Bridge to Algebra® dataset includes data from 123 students, each of whom answered an average of 340.73 items (standard deviation 102, minimum 48, maximum 562, median 341), for a total of 41,910 observations. The data is very unbalanced in terms of performance — over 80% of items were answered correctly.

We split the data into three sets of non-overlapping students: a training set with 97 students, and development and test sets with 13 students each. The entire data set contains data from 893 different problems. We encode items using a unique identifier for every step within a problem, adding up to 5,233 different items.

Carnegie Learning Algebra I Data Set We use data collected by the Carnegie Learning Algebra I Cognitive Tutor. Carnegie Learning Algebra I is designed as a first-year Algebra course for core instruction.

This Algebra I dataset includes data from 205 students, each of whom answered an average of 373.37 items (standard deviation 286, minimum 3, maximum 989, median 314), for a total of 73,181 observations. This dataset is also very unbalanced — over 77% of items were answered correctly.

The data set contains data from 283 different problems. We encode items using a unique identifier for every step within a problem. For example, we encode step "s1" within problem "p1" as item "p1_s1". We end up with 3,081 different items.

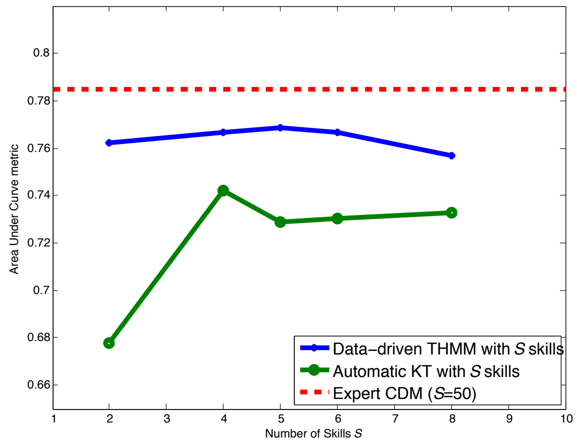
3.3.1.2 Bayesian Priors setup

We now describe the values we used for the priors' hyper-parameters α, τ and ω.

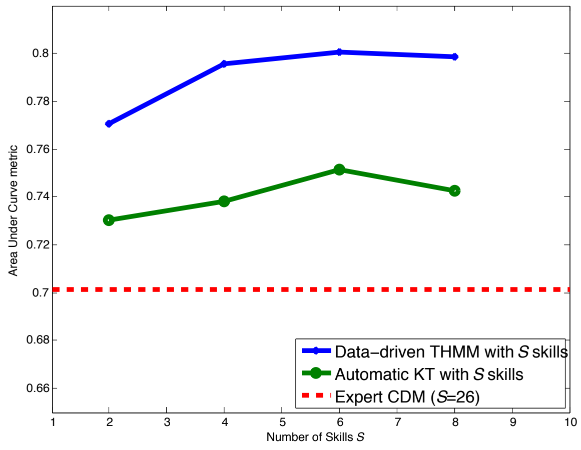
- **Sparse cognitive model**. For our experiments with Topical HMM, we encourage sparsity on the cognitive diagnostic model parameter (Q), motivated by the assumption that each item uses only a few skills. When the value of the hyper-parameter of a Dirichlet prior is below one, the samples are sparse multinomials. We set $\alpha = 0.1$.
- **Practice helps learning, and there is no forgetting**. For our experiments with Automatic Knowledge Tracing and Topical HMM, we tune the effect of students transitioning to a level of better performance, and not going back to the previous level. We tune for the combinations of magnitudes of this effect for $\tau = 10, 100$ and $\omega = 10, 100$.

3.3.1.3 Effect of different number of skills

We now evaluate the behavior of running Automatic Knowledge Tracing (Section 3.1) and Topical HMM (Section 3.2) with a different number of skills (S). Figure 3.5 shows the number of skills on the horizontal axis, and the AUC metric on the vertical axis. We compare the performance of Topical HMM discovering a cognitive diagnostic model from data, with an expert, manually designed model, on the task of predicting future student performance.



(a) Bridge to Algebra Tutor® dataset



(b) Algebra I Linear Tutor dataset

Figure 3.5: Development set performance for different number of skills

Each point in the graph is the best model discovered by our algorithms in the development set, from all the different combinations of prior strengths tried (described in Section 3.3.1.2) and random initializations. In the Bridge to Algebra® dataset, the cognitive diagnostic model hand-crafted by experts obtains an AUC of 0.78, while the data-driven Topical HMM performs its best at $S = 5$ with an AUC of 0.76. Automatic Knowledge Tracing performs its best at $S = 4$ with an AUC of 0.74. On the other hand, in the Algebra I dataset, the fully automatic methods outperform the expert models. The expert scores an AUC of 0.7, while Automatic Knowledge Tracing scores an AUC of 0.75 at $S = 6$. Topical HMM discovers the best cognitive diagnostic model with an AUC of 0.8 with $S = 6$.

3.3.1.4 Computation time

We now compare the time it takes to compute the cognitive diagnostic models described in the previous section on an Intel® Xeon® 3Ghz computer with 16Gb of RAM. Figure 3.6 shows the number of skills computed on the horizontal axis, and the number of computation hours on the vertical axis. The computation time takes into consideration learning the parameters of the model and doing predictions.

The higher performance of Topical HMM over Automatic Knowledge Tracing comes with some computational cost. Although for 2 skills both approaches take about an hour, for 6 skills Auto-

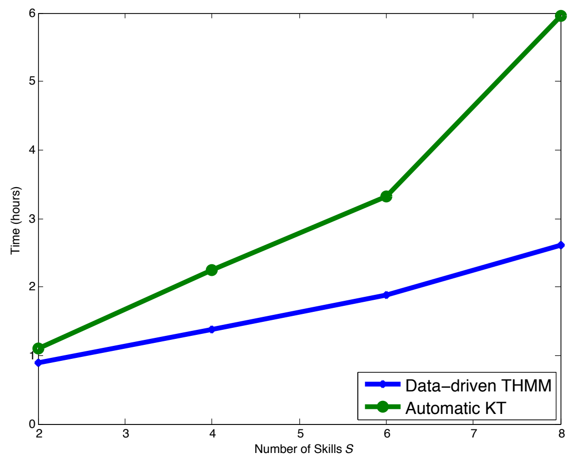


Figure 3.6: Time for training and prediction of the development set

matic Knowledge Tracing requires over 2 hours, while Topical HMM requires 6 hours of computation. Although theoretically Topical HMM should grow in time linearly to the number of skills, in practice we do not see this behavior. We hypothesize that this is due to high memory consumption that may require operating system use of virtual memory.

3.3.1.5 Comparison with other approaches

We compare the performance of these methods:

- **HMM**. Can we find evidence of multiple skills? Topical HMM should perform better than a cognitive model that assigns all of the items to a single skill. We run Bayesian Knowledge Tracing[1] with a cognitive diagnostic model that has only one skill in total. This approach is equivalent to a single Bayesian HMM.

- **Student Performance**. What is the effect of students' individual abilities? We predict that the likelihood of answering item at time t correctly is the percentage of items answered correctly up to time $t-1$. Intuitively, this is the student's "batting average".

- **Random cognitive diagnostic model**. Does the cognitive diagnostic model matter? We create a random cognitive diagnostic model with five skills and assign items randomly to one of five categories. We then train Topical HMM to learn the student model (transition

[1]Although Knowledge Tracing is often called "Bayesian" for historical reasons, we are truly using a contemporary Bayesian treatment of the parameters.

and emission probabilities), without updating the cognitive diagnostic model.

- **Item difficulty**. What is the classification accuracy of a simple classifier? We use a classifier that predicts the likelihood of answering item x as the mean performance of students in the training data on item x. Note that this classifier does not create a cognitive diagnostic model.

- **Manual cognitive diagnostic model**. How accurate are experts at creating cognitive diagnostic models? We use Topical HMM with the cognitive diagnostic model designed manually by an expert using domain knowledge. We initialize the parameter Q of Topical HMM with the expert model and do not update its values. In the case that the expert decided that an item uses multiple skills, we assign uniform weight to each skill even though the experts assumed a conjunctive model. Topical HMM cannot handle a conjunctive cognitive diagnostic model. Knowledge Tracing with multiple skills is an open problem. For example, LR-DBN [Xu and Mostow, 2011] is the state-of-the-art for modeling multiple skills, but we were not able to use it because it is not able to make predictions on unseen students.

- **Automatic Knowledge Tracing cognitive diagnostic model**. We initialize Automatic Knowledge Tracing with the best model discovered using the development set. For Matrix Factorization we use previously published code [Thai-Nghe et al., 2010], and adapted it to use batch gradient descent. For simplicity, we fix the number of latent factors in matrix factorization and the k in K-means clustering to be the same. For items in the test set that are not seen during training, we assign them to random skills.

- **Topical HMM cognitive diagnostic model**. We initialize Topical HMM with the best model discovered using the development set.

Figures 3.7 and 3.8 show the AUC of the different methods on the Bridge to Algebra® and Algebra I datasets, respectively. The error bars show the 95% confidence intervals calculated with an implementation of the Logit method[2] [Qin and Hotilovac, 2008]. The Logit method calculates confidence intervals on non-independent points of the ROC curve.

In both datasets, the random cognitive diagnostic model performs significantly better than chance. We hypothesize that this occurs because we try multiple random restarts and hyper-parameter

[2]http://www.subcortex.net/research/code/area_under_roc_curve

configurations in the development set and we choose the best one for the test set.

In both datasets, Topical HMM outperforms all other approaches. The difference between the expert model and Topical HMM is statistically significant in the Algebra I dataset, because the confidence intervals do not overlap. In the Bridge to Algebra® dataset, the AUC of both Topical HMM and the expert model is 0.77. The automatic approach has the same performance as the expert, manual model, but requires much less human effort. Because the confidence intervals do not overlap, we can conclude with 95% confidence that our data-driven models are significantly more accurate than assuming a cognitive diagnostic model with a single skill (HMM), using the student performance (Student Perf.), or assigning items to skills randomly (Random). The confidence intervals for Automatic Knowledge Tracing, the item difficulty classifier, the manually engineered cognitive diagnostic model and Topical HMM overlap, and have AUC scores of 0.73, 0.75, 0.77 and 0.77 respectively. In Figure 3.9, we plot the ROC curve of the three best classifiers.

In the Algebra I dataset, Topical HMM discovers the most predictive cognitive diagnostic model with an AUC of 0.77 ± 0.01 (with 95% confidence), while the model handcrafted by experts only achieves an AUC of 0.67 ± 0.02; thus, the automatic approach is 14% better than the manual model on this measure. Automatic Knowledge Tracing also performs better than the expert (AUC= 0.73), but worse than the item difficulty (AUC=0.74). Again, our fully data-driven methods outperform the baselines of a cognitive diagnostic model with a single skill (HMM), using the student performance (Student Perf.), or assigning items to skills randomly (Random).

3.3.1.6 Discovered Model

A limitation of Automatic Knowledge Tracing is that it can assign only one skill per item, while Topical HMM can assign multiple skills per item. We now compare the cognitive diagnostic models found by the experts and Topical HMM. To count the number of skills used by Topical HMM, we take the last sample of the parameter Q, and count the number of entries for every item that has probability higher than a threshold of 0.05.

Figures 3.10 and 3.11 show a histogram for the Bridge to Algebra® and Algebra I datasets. In both datasets, the experts assign most items to a single skill for around 80% of the time. On the

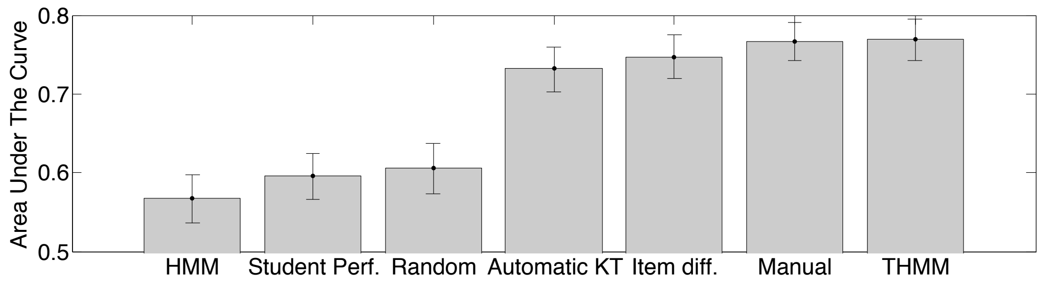


Figure 3.7: Test set AUC performance of different models for the Bridge to Algebra® dataset

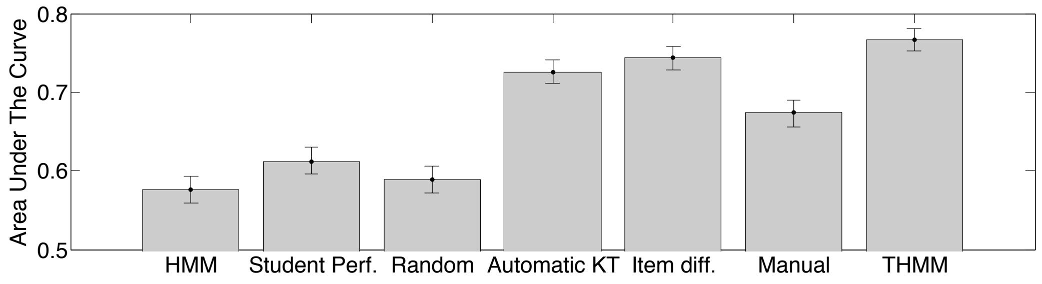


Figure 3.8: Test set AUC performance of different models for the Algebra I dataset

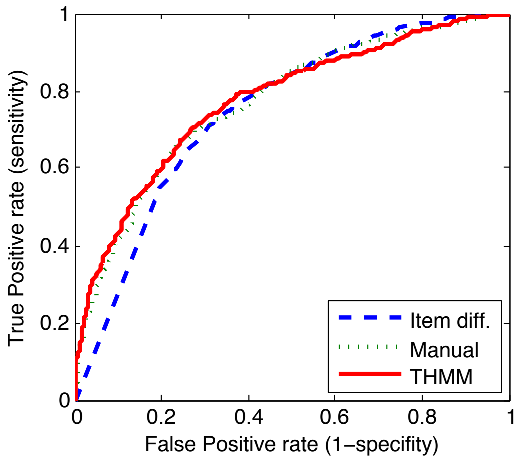


Figure 3.9: Test set ROC of the best models of the Bridge to Algebra® dataset

Table 3.1: Learned Parameters with Topical HMM

skill	Learn	Forget	L0	Guess	Slip
0	.70	.27	.13	.11	.06
1	.75	.21	.18	.10	.04
2	.79	.20	.43	.08	.06
3	.85	.12	.44	.08	.06
4	.91	.05	.91	.01	.55

other hand, for Topical HMM the mode for both datasets is 2 skills per item, with a frequency of about 40%.

In the table of Appendix B we show the mapping of items to skills in the the Bridge to Algebra® dataset. We show this dataset, because it has human readable item names. In table 3.1, we show the learned parameters of the student model.

3.3.2 Understanding Topical HMM with Synthetic Students

We now investigate how well Topical HMM recovers the true parameters from data. For this purpose, we constructed a synthetic data set with 300 students and 30 items. We randomly assigned the items to two skills. Each student solves the items in the same order.

We want synthetic data to be plausible; for example, the probability of answering an item correctly if guessed should be lower than the probability of answering an item correctly if known. Therefore, we hand-crafted some of the behavior of the synthetic students:

- The *initial knowledge*, the probability a student knows a skill before practicing it, is 0.3.
- The *learning rate*, the probability of transitioning from not knowing a skill to knowing it, is 0.15.
- The *performance of mastered skill*, the probability of applying a skill correctly if the student has mastered the skill, is 0.95.

We did not tune these values, and we did not use this knowledge to bias our inference procedure—in this experiment we set up informationless Bayesian priors: $\alpha = -1$, $\beta = 1$, $\tau = 1$, $\omega = 1$.

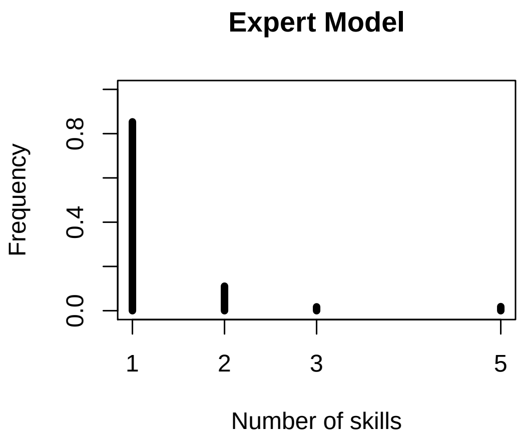
Figure 3.15 marks an ‘x’ on the true parameters from which the synthetic data was generated. We ran 100 different random restarts of the Topical HMM’s Gibbs Sampler. After a burn-in period of 400, for each random restart we collected 1,000 different samples. We then plot the frequency of different points in the parameter space, using a contour plot. For the parameters that determine the initial knowledge and learning rate, most of the samples are in the vicinity of the true parameter value. However, for determining the emission probability – the performance of answering correctly a skill, given that the skill is mastered – the sampler gets stuck in two local optima, each one centered at the value of the parameter value. This experiment shows that Topical HMM recovers the true parameters occasionally. Future work may investigate what are conditions when Topical HMM is able to recover the true parameters from data.

3.4 Conclusion

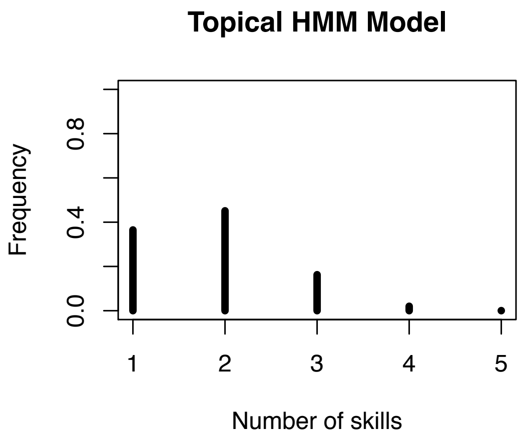
We propose two novel methods, Automatic Knowledge Tracing and Topical HMM, and test them on real student data and synthetic data. A difficulty of modeling real student data is that it may contain many sparse items, students, observations and skills.

We present a blocked Gibbs Sampling procedure that allows efficient inference. Unlike prior methods, Topical HMM and Automatic Knowledge Tracing discover cognitive diagnostic models that trace knowledge of students through time without engineered domain knowledge. To our knowledge, we are the first [González-Brenes and Mostow, 2013] to present an automatic approach to discover a cognitive diagnostic model solely from data collected over time from student data.

Previous work on cognitive diagnostic models from static data was successful in distinguishing between broad areas (i.e., French and math), but not finer distinctions within an area [Winters et al., 2005, Desmarais, 2011]. Given that we were able to discover five different skills within an Algebra data set, we are optimistic about this line of research.

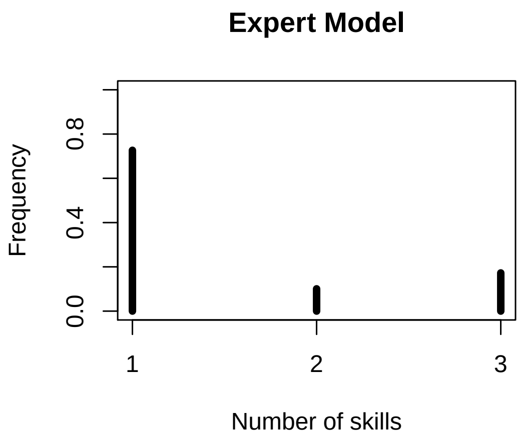


(a) Expert model

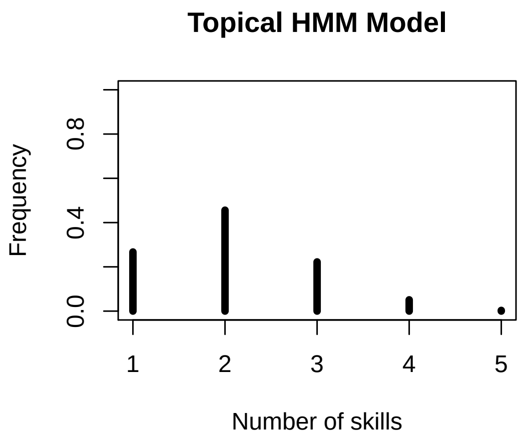


(b) Topical HMM

Figure 3.10: Histogram of number of skills mapped to an item in the Bridge to Algebra® dataset



(c) Expert model



(d) Topical HMM

Figure 3.11: Histogram of number of skills mapped to an item in the Algebra I dataset

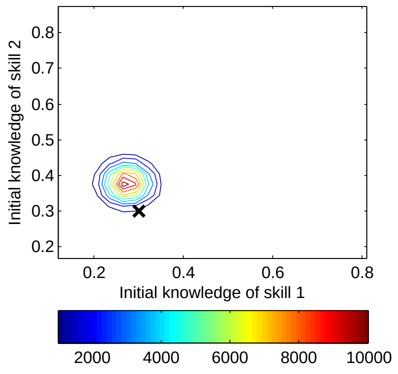


Figure 3.12: K_0

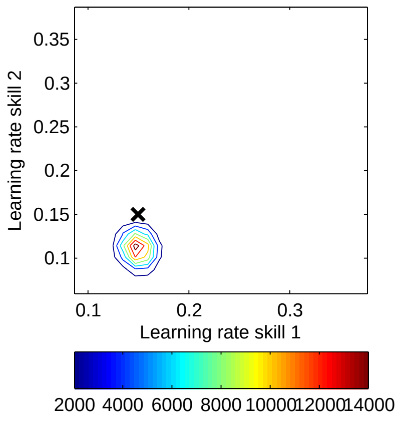


Figure 3.13: K

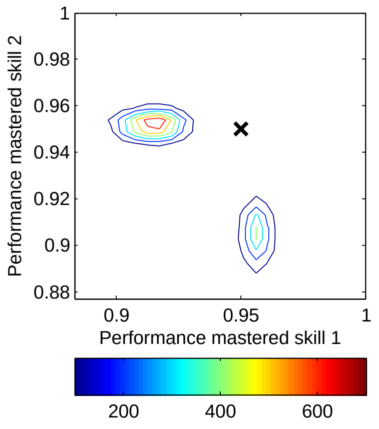


Figure 3.14: D

Figure 3.15: Distribution of estimated parameters using Topical HMM. The ‘x’ indicates the true value that we used to generate the synthetic data. The contour represents the frequency of values sampled out of 100,0000 samples generated by Topical HMM.

Chapter 4

Interpretability

Topical HMM and Automatic Knowledge Tracing use only student performance data to discover a cognitive diagnostic model – a mapping from items to skills – from tutorial data. Both methods are non-convex and non-identifiable: different initializations may make them find different cognitive diagnostic models, which may be equally predictive. Because these methods do not consider similarity between items, some of these cognitive diagnostic models are likely less interpretable than others.

But how can we operationalize interpretability? In their seminal work, Chi et al. [1981] suggest that novice students categorize items by surface features, such as "words in problem text." On the other hand, experts categorize items based on abstract features of the solution, aiming to group items that require the same principle together, such as "conservation of momentum". We assume that if we can explain the skills in terms of cohesive clusters of shallow features, we can improve on interpretability.

In this chapter we focus on using surface features to increase interpretability of cognitive diagnostic models. We aim to discover more interpretable cognitive diagnostic models at the cost of using some domain knowledge. We propose a method that we call ItemClue that builds a Bayesian prior that biases similar items to be clustered together in Topical HMM. ItemClue uses a distance function to quantify the similarity between a pair of items, and then uses the output of a clustering algorithm as a prior to bias the estimation of Topical HMM towards clusterings that group similar items together.

Algorithm 3 ItemClue Prior

```
Require: Item text $x_1, x_2, \dots x_M$, distance function $\mathrm{distance}(\cdot,\cdot)$, number of clusters $S$, intensity $a$
 1: function ITEMCLUEPRIOR($x_1, x_2, \dots x_M, d, S, \mathbf{a}$)
 2:     for each text $i \leftarrow 1 \dots t_M$ do
 3:         for each text $j \leftarrow 1 \dots t_M$ do
 4:             $\mathbf{D}_{i,j} \leftarrow d(\mathrm{Preprocess}(t'_i), \mathrm{Preprocess}(t'_j))$
 5:     $\langle x_i \rightarrow$ cluster $c_i \rangle \leftarrow \mathrm{Relational\_KMeans}(\mathbf{D}, \mathrm{S})$ // Map each item $x_i$ into a cluster $c_i$
 6:     for each mapping $x_i \rightarrow c_i$ do
 7:         for each skill $s \leftarrow 1 \dots S$ do
 8:             if $s = c_i$ then
 9:                 $\alpha^i_{(s)} \leftarrow a$
10:             else
11:                 $\alpha^i_{(s)} \leftarrow 1$
        return $\boldsymbol{\alpha}$
```

4.1 ItemClue

We now describe ItemClue, a method to group items according to a distance function using Topical HMM. Algorithm 3 describes how to build an ItemClue prior for Topical HMM: It requires items $x_1 \dots x_M$, a distance function to specify the similarity of the items, the number of clusters S, and a bias intensity parameter of how much the item similarity should influence discovery of cognitive diagnostic models. Lines 2-4 build a similarity matrix comparing each item to each other using Euclidean distance. We describe the preprocessing of items in detail in Section 4.2, which returns a vector of feature values. Two alternatives of how to preprocess items are described in Algorithms 4 and 5. Line 5 uses Relational K-Means [Szalkai, 2013], an extension of the K-Means algorithm that works with an arbitrary similarity matrix, to group similar items together into S clusters. Finally, lines 6–11 build a prior that can be used in Topical HMM.

4.2 Shallow Features

Previous methods to discover interpretable cognitive diagnostic models have been proposed by clustering patterns in the correct answer [Li et al., 2013] or in the text of the item [Karlovčec et al., 2012]. These approaches require feature engineering and some domain knowledge. We operationalize these strategies, and use two distance functions with ItemClue to measure item similarity:

- **Similarity of the correct answer**: We aim to achieve interpretability by grouping items with similar correct responses. Algorithm 4 specifies the preprocessing we use, which reproduces previous work [Li et al., 2013]: first we tokenize an item's correct answer, so that the whole and fractional parts of a number are replaced by N. For example -20.5 is replaced by $-N.N$. We normalize variables to be represented as v. We then extract letter bigrams and trigrams from the tokens. We only take into consideration whether the n-gram is present or absent. Table 4.1 shows Li et al. [2013]'s example of features we use. For simplicity, if a question has more than one correct response, we pick one randomly. Unlike Li et al. [2013]'s work, we do not apply Principal Component Analysis to the features extracted. Our rationale is that the factors discovered by Principal Component Analysis often do not exhibit a conceptual meaning and are hard to interpret.

- **Similarity of the item's text**: We aim to achieve interpretability by grouping items with similar text of the questions. For example, we would be clustering on the lexical level on items' text, such as "How far will you have flown in two hours?". Algorithm 5 shows the preprocessing we perform: we replace entities such as numbers and proper names (places or people), we lemmatize the tokens, for example "flown" turns into "fly", and we remove stop words. Lastly, we count the number of times a word appears.

Algorithm 4 Preprocessing to cluster by correct algebraic response. This preprocessing was first proposed by Li et al. [2013].

```
Require: Item id $x$
1: function CORRECT_ANSWER($x$)
2:     $x' \leftarrow$ Get_Correct_Answer($x$)
3:     $x' \leftarrow$ Rename_Variables($x'$)
4:     $x' \leftarrow$ Replace_Numbers($x'$)
5:     return is-n-gram-present?($x'$)
```

4.3 Evaluation

We evaluate only on data from the Algebra I tutor described in section 3.3.1.1, because this is the only dataset for which we have both the item text and the correct response. For natural

Table 4.1: An example list of correct answers, how they are preprocessed, and the features that are extracted for ItemClue. This table and the preprocessing performed were first proposed by Li et al. [2013].

	Item	Features extracted													
Answer	Preprocessed	-N	Nv	v=	=N	-Nv	Nv=	v=N	v+	+N	N=	Nv+	v+N	+N=	N=N
-3x = 6	$-Nv = N$	✓	✓	✓	✓	✓	✓	✓							
2y+5=7	$Nv + N = N$		✓		✓				✓	✓	✓	✓	✓	✓	✓

Algorithm 5 Preprocessing to cluster by item text

```
Require: Item id $x$
1: function PREPROCESS_TEXT($x$)
2:     $x' \leftarrow$ Get_Item_Text($x$)
3:     $x' \leftarrow$ Replace_Named_Entities($x$)
4:     $x' \leftarrow$ Lemmatize($x'$)
5:     $x' \leftarrow$ Remove_Stop_Words(x')
6:     return count-words($x'$)
```

language processing, we use the Stanford Natural Language Processing Toolkit[1], which includes a Named Entity Recognition parser [Finkel et al., 2005, Angeli et al., 2012], and a Part-Of-Speech tagger [Toutanova et al., 2003, Toutanova and Manning, 2000], useful for lemmatization. The list of stop words we used is from the United States Patent Full Text Database[2].

We show the effect of the number of clusters on ItemClue on clustering by correct algebraic response. Figure 4.1 shows on the horizontal axis the number of clusters, and on the vertical axis the within-cluster scatter. The scatter is simply the sum of all of the pairwise distances between elements, with each pair of elements counted once. The within-cluster scatter is the sum of the scatters for each cluster. We compute the within-cluster scatter from the best of three random initializations of the clustering algorithm. As the number of clusters increases, the within-cluster scatter decreases monotonically, except for 8 clusters, presumably due to not finding an appropriate initialization.

We evaluate the effect of increasing interpretability by forcing Topical HMM to group similar items together and we build an ItemClue prior at different strengths. We evaluate cognitive diagnostic model strategies by how well they predict future student performance. We operationalize predicting future student performance as the classification task of predicting whether students

[1]`http://nlp.stanford.edu/software/`
[2]`http://www.uspto.gov/patft/help/stopword.htm`

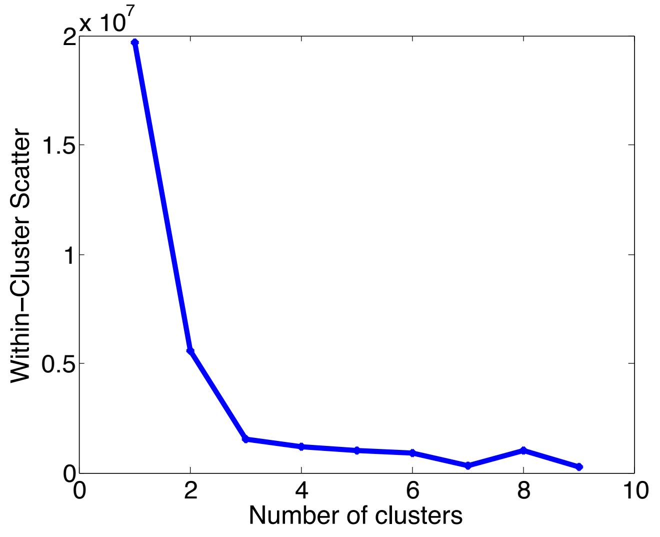


Figure 4.1: Effect of strength of ItemClue prior

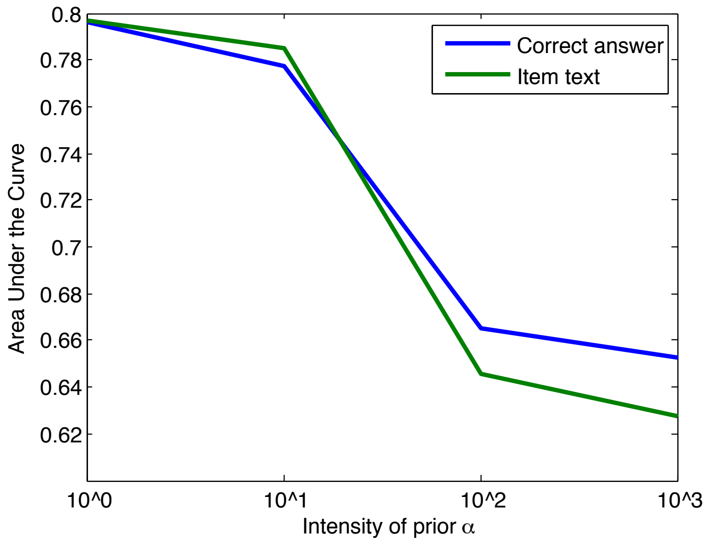


Figure 4.2: Effect of strength of ItemClue prior

correctly solved the items in a held-out set. We use the history preceding the time step we want to predict. To speed up computations, we predict up to the 150th time step. Because we are studying different hyper parameter strengths, we evaluate only on the development set.

Figure 4.2 shows on the horizontal axis the intensity a used to generate a prior with ItemClue, and on the vertical axis the AUC of Topical HMM with ItemClue clustering by the similarity of the correct answer or of the item's text. We set the number of skills to six, which gives best performance on the development set (see Figure 3.5). We try different hyper-parameters and initializations, but report only the best values. The AUC when not using ItemClue (intensity $a = 10^0$) is approximately 0.8, and it decreases when using a stronger prior ($a = 10^3$): it drops to 0.65 and 0.62 to cluster by correct answer and text, respectively.

At the strongest prior ($a = 10^3$), Topical HMM is using only the prior to discover the item to skill mapping. The clusters discovered are:

- **Positive integer constants**, abstracted as N.
- **Plotting operations**, e.g. Series1PlotPoint2, are responses that require students to manipulate a plot.
- **Simple (1 operator) constant expressions**, e.g. $N \cdot N$, $N + N$.
- **Complex (2+ operator) constant expressions**, e.g. $N - N \cdot N$, $N + N \cdot N$.
- **Simple (1 operator) variable expressions**, e.g. $N/N \cdot x$, $Nx + N$.
- **Complex (2+ operator) variable expressions**, e.g. $Nx(x + N)$, $x(N.N) + N$.

Although the item to skill mapping is done automatically, the interpretations of what each skill is requires human analysis. Future work may look at how much human effort is required to analyze different cognitive diagnostic models. The six clusters discovered are much coarser than the ones discovered by the expert, and may suggest that some of the expert skills could be merged.

When the prior is lower, and Topical HMM does not make use of ItemClue to bias estimation, the clusters discovered are hard to interpret, mainly because an item often is assigned to multiple skills at different strengths. See Figures 3.10 and 3.11 for more detail. Although multiple skills improve accuracy, they hurt interpretability. Future work may look at using a penalty for multiple skills.

4.4 Conclusion

ItemClue increases the cohesiveness of clusters according to a distance function that quantifies similarity; however, it decreases the predictive performance of Topical HMM. The fact that surface features hurt the predictive performance of a cognitive diagnostic model in our experiments requires further research, as previous work [Li et al., 2013] has used surface features to build a cognitive diagnostic model. Future work could reproduce such work, ablating the surface features to understand the relationship of performance and surface features.

Chapter 5

Follow-On Work

To spare researchers from falling in the same pitfalls we did, in this chapter we discuss strategies that have not yielded positive results yet in our preliminary experiments. Section 5.1 describes our attempt to increase the number of parameters of Topical HMM by modeling the hierarchical structure of problems and steps. Section 5.2 describes our attempt at reducing the number of parameters of Topical HMM by tying parameters together. Section 5.3 reports the results of these attempts.

5.1 More parameters: Hierarchical Topical HMM

The methods we discuss in this thesis to discover cognitive diagnostic models assume that each item has a unique identifier. But many intelligent tutor systems, such as the ones that logged the data in the PSLC Datashop, store more fine-grained data. Often tutors decompose items into problems which are tasks for a student to perform that typically involves multiple steps.

We hypothesize that steps within a problem require a small number of skills, and that performance can be improved by modeling explicitly the relationship between problems and steps. Therefore, we propose Hierarchical Topical HMM to allow for a problem identifier, and a step identifier, instead of only using an item. We aim to improve the interpretability of the models discovered by Topical HMM, and to account for the different levels of granularity data that can be studied. We describe Hierarchical Topical HMM as a graphical model in Figure 5.1 and its generative story in Algorithm 6. We analyze items at two levels of granularity, at the problem level $\ddot{t}$ and at a step level t. The hyper-parameters of Hierarchical Topical HMM are similar to Topical HMM:

- $\mathbf{S}$ is the number of skills in the model.
- $\mathbf{U}$ is the number of users (students).
- $\ddot{\mathbf{T}}_{\mathbf{u}}$ is the number of problems student u practiced.
- $\mathbf{T}_{\mathbf{u}}(\ddot{t})$ is the number of time steps student u practiced problem $\ddot{t}$.
- $\ddot{\mathbf{M}}$ is the number of problems.
- $\mathbf{L}$ is the number of levels a student can master. For example, if we consider that students can be novice, medium or expert, we would use $\mathbf{L} = 3$.

The discrete variables account for problems and steps:

- $\ddot{x}_{u,\ddot{t}}$ is the problem the student u practiced at time $\ddot{t}$.
- $x_{u,\ddot{t},t}$ is the step number t the student u practiced while solving problem $\ddot{x}_{u,\ddot{t}}$.
- $r_{u,\ddot{t},t}$ is the skill mixture required for problem $\ddot{x}_{u,\ddot{t}}$.
- $q_{u,\ddot{t},t}$ is the skill required for item $x_{u,\ddot{t},t}$.
- $k^s_{u,\ddot{t},t}$ is a variable with values from $1 \ldots L$ that represents the level of knowledge for skill s. There is a Markovian dependency across time steps: if skill s is known at a point of time, it is likely to be known at the next point of time. We model a deterministic constraint that the knowledge of a skill can change its value only while the student exercises the skill. Since we need to take into account problems and steps to consider the previous time point, we defined the function previous to aid in computing the previous time step:

$$\text{previous}(\langle \ddot{t}, t \rangle) = \begin{cases} \langle \ddot{t}, t-1 \rangle, & \text{if } t > 1 \\ \langle \ddot{t}-1, T(\ddot{t}-1) \rangle & \text{if } t = 1 \end{cases} \tag{5.1}$$

Here $T(\ddot{t}-1)$ is the number of steps of the problem number $\ddot{t}-1$. Let's illustrate this with an example. Suppose we are interested in calculating the previous time step of the first step of the second problem. If the first problem has 3 steps, then $\text{previous}(\langle 2, 1 \rangle) = \langle 1, 3 \rangle$.

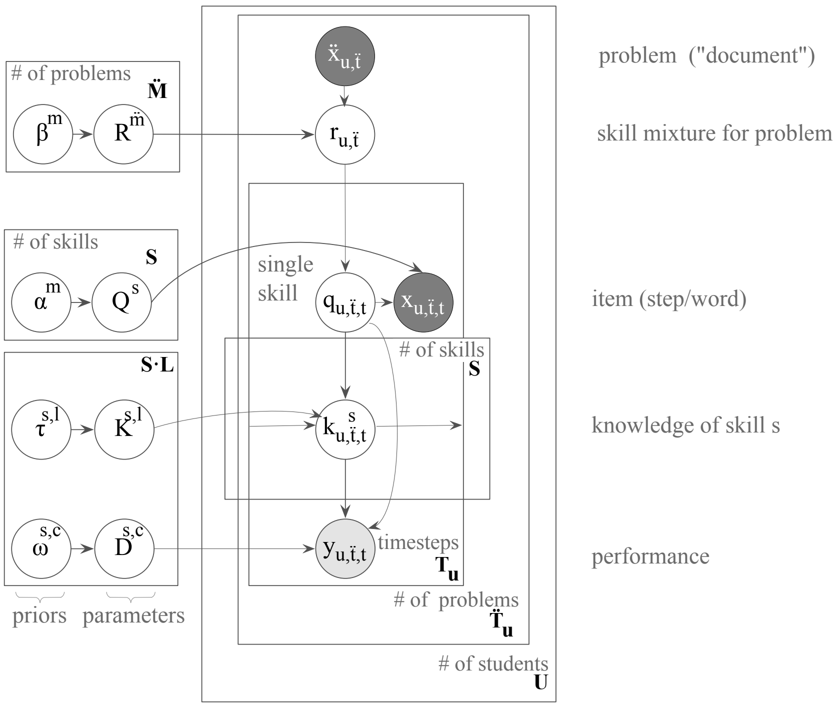


Figure 5.1: Hierarchical model

- $y_{u,\ddot{t},t}$ is the target variable that models performance. It is observed only during training.

Since we take a fully Bayesian approach, we model parameters as random variables.

- $R^{\ddot{x}_{u,\ddot{t}}}$ is a multinomial representing the skills required for problem $r_{u,\ddot{t}}$.

- $Q^{q_{u,\ddot{t},t}}$ is a multinomial representing the items that use skill $q_{u,\ddot{t},t}$.

Parameters K and D are like in Topical HMM.

- $K^{s,l}$ is the probability distribution for transitioning from knowledge state l of skill s at one time step to a knowledge state at the next time step.

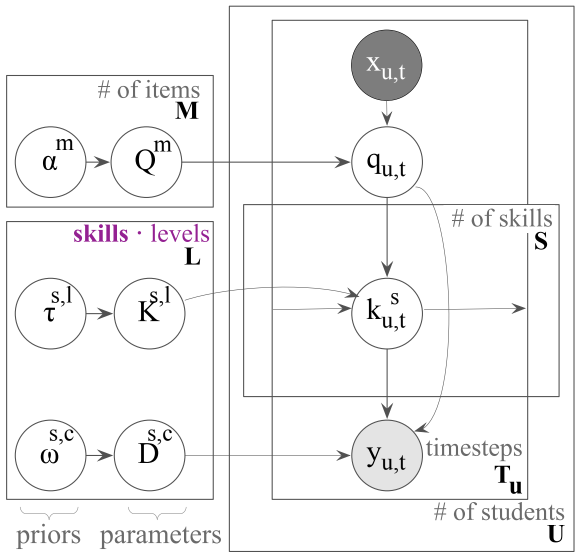
- $D^{s,l}$ is the emission probability of the performance for skill s and knowledge state l.

We use Dirichlet priors α, τ, ω for the parameters.

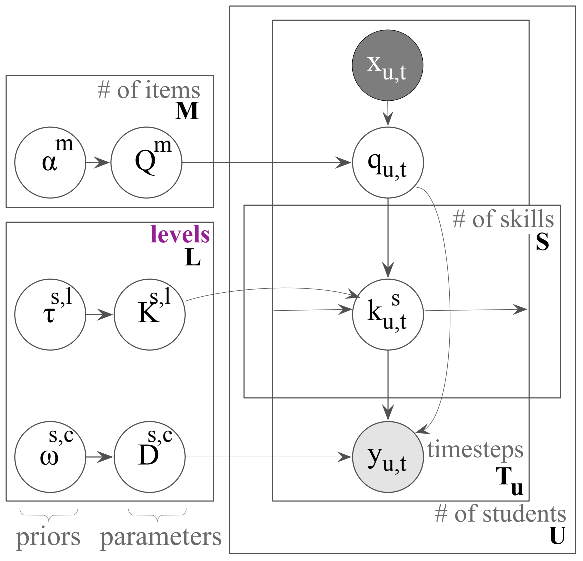
Algorithm 6 Generative story of Hierarchical Topical HMM

Require: A sequence of problem identifiers ($\ddot{x}$) and step identifiers (x) for $\mathbf{U}$ users, number of skills ($\mathbf{S}$), number of student states ($\mathbf{L}$), number of items ($\mathbf{M}$)

```
 1: function HIERARCHICAL_TOPICAL_HMM($\ddot{x}, x, \mathbf{S}, \mathbf{U}, \mathbf{L}, \mathbf{M}$)
 2:     // Draw parameters from priors:
 3:     for each skill $s \leftarrow 1$ to $\mathbf{S}$ do
 4:         for each knowledge state $l \leftarrow 1$ to $\mathbf{L}$ do
 5:             Draw parameter $K^{s,l} \sim$ Dirichlet($\tau^{s,l}$)
 6:             Draw parameter $D^{s,l} \sim$ Dirichlet($\omega^{s,l}$)
 7:     for each problem $m \leftarrow 1$ to $\ddot{\mathbf{M}}$ do
 8:         Draw $R^m \sim$ Dirichlet($\beta$)
 9:     for each step $m \leftarrow 1$ to $\mathbf{M}$ do
10:         Draw $Q^m \sim$ Dirichlet($\alpha$)
11:     // Draw variables from parameters:
12:     for each student $u \leftarrow 1$ to $\mathbf{U}$ do
13:         for each problem $\ddot{t} \leftarrow 1$ to $\ddot{\mathbf{T}}_u$ do
14:             for each step $t \leftarrow 1$ to $\mathbf{T}_u(\ddot{t})$ do
15:                 Draw skill $r_{u,\ddot{t}} \sim$ Multinomial($R^{\ddot{x}_{u,\ddot{t}}}$)
16:                 Draw skill $q_{u,\ddot{t},t} \sim$ Multinomial($r_{u,\ddot{t}}$)
17:                 Draw step $x_{u,\ddot{t},t} \sim$ Multinomial($Q^{q_{u,\ddot{t},t}}$ )
18:                 for $s \leftarrow 0$ to $\mathbf{S}$ do
19:                     if $s = q_{u,t}$ then
20:                         // knowledge state could change:
21:                         $k'' \leftarrow k^s_{u,\text{previous}(\langle \ddot{t},t \rangle)}$ // previous knowledge
22:                         Draw $k^s_{u,t} \sim$ Multinomial($K^{s,k''}$)
23:                     else
24:                         // knowledge state can't change:
25:                         $k^s_{u,t} \leftarrow k^s_{u,\text{previous}(\langle \ddot{t},t \rangle)}$
26:             $q' \leftarrow q_{u,\ddot{t},t}$ // current skill
27:             $k' \leftarrow k^{q'}_{u,\ddot{t},t}$ // current knowledge state
28:             Draw performance $y_{u,\ddot{t},t} \sim$ Multinomial($D^{q',k'}$)
```



(a) Plate diagram of Topical Hidden Markov Models



(b) Plate diagram of Topical Hidden Markov Models with Parameters Tied. Unlike Topical HMM, the parameters K and D are the same for all of the skills.

Figure 5.2: Comparison of Topical HMM and Tied Topical HMM

5.2 Fewer Parameters: Tied Parameters Topical HMM

We attempted to reduce the number of parameters of Topical HMM by tying the transition and emission probabilities in order that all skills have the same learning, guess and slip parameters. Figure 5.2 compares Topical HMM with Tied Topical HMM – the difference between the two models is that the parameters and priors are not skill specific.

5.3 Results and Conclusion

In our preliminary results, the Gibbs Sampler we formulated for Hierarchical Topical HMM and Tied Topical HMM failed to learn useful parameters, as they both diverged. The alternatives to Topical HMM in this chapter are overparameterized or underparameterized for the datasets on which we tested them. This is, these methods have too many or too few parameters to model real student data. Future work might use Hierarchical Topical HMM with a non-random initialization. Another possibility is to initialize the model with a simpler alternative, such as Tied Topical HMM or Topical HMM. An interesting opportunity is to explore alternative techniques to training Hierarchical Topical HMM, such as Variational Expectation Maximization [Beal and Ghahramani, 2003] or Particle Filtering [Doucet et al., 2000].

Chapter 6

Relation to Prior Work

This section is organized as follows: Section 6.1 describes prior work on automatic discovery of cognitive diagnostic models. Section 6.2 describes prior work on student modeling. Section 6.3 describes prior work on temporal data mining.

6.1 Automatic Discovery of Cognitive Diagnostic Models

We now describe automatic methods that discover cognitive diagnostic models from performance data and other sources.

6.1.1 Performance data

Student performance data has been used to discover cognitive diagnostic models. For example, in psychometrics, the branch of psychology and education concerning educational statistics, matrix factorization methods have been applied to static assessment instruments such as a single exam, or a homework assignment to discover a cognitive diagnostic model. A survey of previous approaches to automatic discovery of cognitive diagnostic models can be found elsewhere [Winters et al., 2005]; popular approaches include Item Response Theory [Rasch, 1961], and matrix factorization techniques such as Principal Component Analysis, Non-Negative Matrix Factorization [Desmarais, 2011, Winters et al., 2005], and the Q-Matrix Method [Barnes et al., 2005].

These methods can help explain what skills students have mastered, but they ignore the temporal dimension of data. Unlike Topical HMM, these approaches do not discover a clustering *per se*: they predict student performance as a combination of latent user traits, and latent item difficulty traits (skills). Latent item traits (or skills) are a linear combination of performance on items. How to apply matrix factorization techniques to students and items with little or no past performance data is an active line of research [Zhang et al., 2011]. Topical HMM models uncertainty in the cognitive diagnostic model and therefore does not require an ad-hoc solution for predicting on unseen students. Clustering of performance data [Chiu et al., 2009] without dimensionality reduction has been used to provide a cognitive diagnostic model for discovering a student's static knowledge using Deterministic-Input Noisy-AND (DINA) models [Junker and Sijtsma, 2001].

Learning Factors Analysis [Cen et al., 2006] uses temporal data, but requires engineered knowledge to improve upon. The knowledge provided to Learning Factors Analysis can be encoded with an ItemClue prior, and future work may compare Learning Factors Analysis with ItemClue. To our knowledge, our work in Topical HMM and Automatic Knowledge Tracing is the first to evaluate a cognitive diagnostic model for tracing student knowledge completely automatically.

6.1.2 Other Approaches

Not all methods rely on using performance data to discover interpretable cognitive diagnostic models. For example, supervised learning has been used to group the text associated with the item [Karlovčec et al., 2012]. To improve the interpretability of the skills discovered by Topical HMM, we make use of an existing method that uses unsupervised learning. This approach requires extensive feature engineering to cluster items' answers [Li et al., 2013]. An alternative line of work has used simulated students to build a cognitive diagnostic model [Li et al., 2011].

6.2 Student Modeling

We now review previous student modeling techniques that relate to our methods. Knowledge Tracing [Corbett and Anderson, 1995] is a popular method to model students' changing knowledge during skill acquisition. It requires (a) a cognitive diagnostic model that maps each item to the skills required, and (b) logs of students' correct and incorrect answers as evidence of their knowledge of particular skills. Knowledge Tracing can be formulated as a graphical model [Reye, 2004]: a student's items that belong to the same skill are grouped into a single sequence of steps, and a HMM is trained for each sequence. The observable variable is the performance of the student solving the item, and the hidden state is a binary latent variable that represents whether the student knows the skill. Topical HMM generalizes Knowledge Tracing when the cognitive diagnostic model is known, and partitions the items into a non-overlapping set of skills (each item uses exactly one skill). Knowledge Tracing has enabled significantly faster teaching by intelligent tutors, while achieving the same performance on evaluations [Corbett, 2001].

Attempts to include time using tensor factorization – matrices with more than two dimensions – to model student learning have been limited, because these methods do not scale in the number of dimensions. To overcome this limitation, Thai-Nghe et al. [2010] only consider the average data from the last few time steps, instead of considering the whole student performance sequence. Topical HMM and Automatic Knowledge Tracing are able to use the whole sequence of student performance.

6.3 Temporal Models

We formulate Topical HMM as a hierarchical Bayesian model in which each item is modeled as a mixture over an underlying set of skills. Our formulation of Topical HMM is related to Input-Output HMM [Bengio and Frasconi, 1995] and Factorial HMM [Ghahramani and Jordan, 1997]. We now briefly discuss these approaches.

The Input-Output HMM, as well as the conventional HMM, is tractable only with a relatively small number of states: to represent b bits of information about the history of a time sequence, an Input-Output HMM would need 2^b distinct states [Ghahramani and Jordan, 1997]. A Factorial HMM works around this exponentially large number of states with a distributed state representation that can achieve the same task with b binary state variables. However, Factorial HMMs only model an output sequence. Topical HMM combines concepts from both Input-Output HMMs and Factorial HMMs: it uses a distributed state representation and is able to map input sequences to output sequences.

Chapter 7

Conclusion

In this thesis we operationalize the question of what and when students learn as discovering cognitive and student models, respectively. We propose four methods:

- Dynamic Cognitive Tracing, an easily implemented model that can be programmed in a Bayesian network toolkit, but can handle only a few items and skills.
- Automatic Knowledge Tracing, a pipeline that discovers a single-skill cognitive diagnostic model first, and then estimates the student model.
- Topical Hidden Markov Models, a joint model that discovers simultaneously a mixture of items to skills, and the student model.
- ItemClue, an approach that leverages previous literature, to bias Topical HMM parameter discovery towards interpretable models.

We provide empirical evidence for the thesis statement that it is possible to trace student knowledge with a data-driven cognitive diagnostic model that does not rely on expert domain knowledge.

In all of our experiments, Topical HMM outperformed Automatic Knowledge Tracing, at the cost of more computational expense. The relative performance of a data-driven model and an expert model seems dependent on the domain. For example, in the Bridge to Algebra Tutor® dataset, our automatic approaches did as well as the expert model. However, on the Algebra I dataset, Topical HMM significantly outperformed the expert cognitive diagnostic model engineered with domain knowledge.

The contributions of this dissertation are the following:

- Fully data-driven methods that discover how to factor items into skills and when students master them.
- Fully data-driven approaches to model skill acquisition through time.
- A novel approach to modeling items as a mixture of skills.
- An automatic approach for reasoning on entire student performance sequences, without the need to group the information into skills.
- A principled approach that relies on Bayesian priors to bias estimation of cognitive diagnostic models to more interpretable solutions.
- A simple cognitive diagnostic model discovered using a method developed by Li et al. [2013]. The model has six skills and it groups items' correct responses either as plotting operations or as whether they contain numbers or variables and on the number of operators used.
- Empirical evaluation comparing cognitive diagnostic models handcrafted by experts versus discovered automatically.
- An efficient Gibbs sampler method to perform inference on Topical HMM.

7.1 Future work and limitations

In this thesis, we have operationalized a skill as a grouping – either through hard or soft clustering – of items that have similar response patterns by students. Although our models are able to answer what and when students learn, we are not able to answer *how* students are learning. For example, our methods are agnostic as to whether students are doing addition by mental calculation or by counting fingers. Future work could leverage recent work on brain-computer interfaces to extend our methods.

We formulate the discovery of a cognitive model as a dimensionality reduction problem. A limitation in this approach is that we cannot distinguish between two items with the same identifiers. Therefore, our approaches are restricted in their ability to model skills that change with context.

We describe Topical HMM, a novel model to discover a mixture of skills for items using an efficient Gibbs Sampler that learns from data. However, our Gibbs Sampler is only efficient for batch processing of data, and it is not scalable for online inference. Efficient updates of a multi-skill model are not trivial because the assignment of item to skill may change when further evidence is presented. This may cause the paradoxical result [Hooker et al., 2009] that the estimate of a student's ability in some skill may decrease after answering a question correctly. For this reason, Topical HMM might be suited for guiding tutoring activities, but not high-stakes assessments.

We evaluate our methods on predicting student performance, not on how well they improve tutoring. Future work may evaluate the models by the expected time gained by students [Lee and Brunskill, 2012, Yudelson et al., 2013]. We do not evaluate the human computer interaction issues of discovering an interpretable cognitive diagnostic model. For example, an interface that allows the researcher to graphically compare the accuracy of the cognitive diagnostic models and the item to skill mappings discovered at different levels of interpretability would be useful to researchers. Such an approach could build on recent work on using graphical user interfaces for topic modeling in documents [Hinneburg et al., 2012].

Future work may apply Topical HMM to other domains than education: a general problem in recommender systems is accurate prediction of responses associated with dyadic data, such as sparse matrices of movie-viewer ratings, word-document counts, or product-customer purchases. In this thesis, we focus on student-item performance data collected at different time points from students as they learn new material, but we are excited to explore other domains to apply the models we propose.

Appendices

Appendix A

Expert Cognitive Diagnostic Models

In this appendix we list the skills defined by experts to illustrate how different the cognitive models designed by experts and by automatic methods can be.

A.1 Bridge to Algebra Cognitive Tutor®

1. Enter given side
2. Enter given area
3. Enter given circle diameter
4. Enter given circle radius
5. Enter given measurement
6. Enter given parallelogram base
7. Enter given parallelogram height
8. Enter given parallelogram non
9. Enter given rectangle length
10. Enter given rectangle width
11. Enter given square side length
12. Enter given trapezoid height
13. Enter given trapezoid longer base
14. Enter given trapezoid non base side
15. Enter given trapezoid shorter base
16. Enter given triangle base
17. Enter given triangle height
18. Enter given triangle non base side
19. Find added area
20. Find circle area in context
21. Find circle area out of context
22. Find circle circumference in context

23. Find circle circumference out of context
24. Find individual area
25. Find individual area in context
26. Find individual area out of context
27. Find parallelogram area in context
28. Find parallelogram area out of context
29. Find parallelogram perimeter in context
30. Find parallelogram perimeter out of context
31. Find rectangle area in context
32. Find rectangle area out of context
33. Find rectangle perimeter in context
34. Find rectangle perimeter out of context
35. Find square area in context
36. Find square area out of context
37. Find square perimeter in context
38. Find square perimeter out of context
39. Find subtracted area
40. Find trapezoid area in context
41. Find trapezoid area out of context
42. Find trapezoid perimeter in context
43. Find trapezoid perimeter out of context
44. Find triangle area in context
45. Find triangle area out of context
46. Find triangle perimeter in context
47. Find triangle perimeter out of context
48. Know to pose component areas
49. Know to use component areas
50. Successfully pose component areas

A.2 Algebra I Data Set

1. Changing axis bounds
2. Changing axis intervals
3. Correctly placing points
4. Entering a given
5. Entering the slope
6. Entering the y
7. Find X any form
8. Find X any slope
9. Find X negative slope
10. Find X positive slope
11. Find Y negative slope
12. Find Y positive slope
13. Find Y Simple
14. Identifying units
15. intercept
16. Labelling point of intersection
17. Setting the slope
18. Setting the y
19. Using difficult numbers
20. Using large numbers
21. Using simple numbers
22. Using small numbers
23. Write expression any form
24. Write expression mx
25. Write expression negative slope
26. Write expression positive slope

Appendix B

Learned Cognitive Diagnostic Models

In this appendix we list the item to skill mapping discovered by Topical HMM. The automatic method identifies 5 skills, in contrast to the dozens identified by the expert (see previous appendix). Here, we show a random sample of 10 items assigned to each skill. We show the name of the item, and the cluster in which it was assigned. In contrast to the expert model, the output of an automatic method is difficult to understand. An item may be assigned to multiple skills.

Table B.1: Cognitive Diagnostic Model Learned by Topical HMM on the Bridge to Algebra® dataset

Skill 1	Skill 2	Skill 3	Skill 4	Skill 5
parallelogram-1-Q1-base-gn	trapezoid-1-Q2-perimeter-gn	trapezoid-1-Q1-side-2-gn	square-1-Q1-side-gn	trapezoid-2-Q1-shorter-base-gn
rectangle-2-Q1-base-gn	parallelogram-1-Q1-base-gn	square-1-Q1-perimeter-gn	rectangle-1-Q1-height-gn	square-1-Q1-side-gn
rectangle-2-Q1-area-gn	trapezoid-1-Q1-area-gn	triangle-1-Q1-base-gn	trapezoid-1-Q2-longer-base-gn	square-1-Q1-area-gn
area-operation	rectangle-1-Q1-base-gn	triangle-2-Q1-base-gn	rectangle-1-Q2-area-gn	trapezoid-2-Q1-area-gn
circle-1-Q1-circumference-gn	trapezoid-1-Q1-area-gn	square-1-Q1-area-gn	rectangle-1-Q1-base-gn	rectangle-1-Q1-area-gn
rectangle-1-Q1-area-gn	parallelogram-1-Q1-base-gn	parallelogram-1-Q1-base-gn	circumference-gn	area-operation
rectangle-1-Q1-height-gn	trapezoid-2-Q1-area-gn	area-operation	trapezoid-1-Q1-area-gn	trapezoid-1-Q1-area-gn
rectangle-1-Q2-area-gn	trapezoid-1-Q1-longer-base-gn	rectangle-1-Q1-base-gn	rectangle-1-Q2-height-gn	area-operation
rectangle-1-Q1-base-gn	triangle-1-Q2-perimeter-gn	rectangle-1-Q1-height-gn	triangle-1-Q2-base-gn	square-1-Q1-side-gn
parallelogram-1-Q2-side-gn	trapezoid-2-Q1-shorter-base-gn	rectangle-1-Q1-base-gn	rectangle-1-Q2-base-gn	area-operation

Bibliography

This bibliography includes back-references to the pages where a reference was cited, except for the references given in full at the start of Chapters 2 and 3.

G. Angeli, C. D. Manning, and D. Jurafsky. Parsing time: learning to interpret time expressions. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 446–455, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=2382029.2382092`. 44

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. 2

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`. 54

M. J. Beal and Z. Ghahramani. The Variational Bayesian EM Algorithm for Incomplete Data: With Application to Scoring Graphical Model Structures. In J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith, and M. West, editors, *Bayesian Statistics*, volume 7, pages 453–464. Oxford University Press, 2003. 53

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. 2

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. 10

Y. Bengio and P. Frasconi. An input output HMM architecture. In D. S. Touretzky, M. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 427–434, Denver, CO, 1995. MIT Press. 56

J. Besag. An Introduction to Markov Chain Monte Carlo Methods. In M. Johnson, S. Khudanpur, M. Ostendorf, and R. Rosenfeld, editors, *Mathematical Foundations of Speech and Language Processing*, volume 138 of *The IMA Volumes in Mathematics and its Applications*, pages 247–270. Springer New York, 2004. ISBN 978-1-4612-6484-2. URL http://dx.doi.org/10.1007/978-1-4419-9017-4_11. 23, 26

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3: 993–1022, Mar. 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944937. 21

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. 2, 3, 17, 55

M. T. Chi, P. J. Feltovich, and R. Glaser. Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5(2):121 – 152, 1981. URL http://www.sciencedirect.com/science/article/pii/S0364021381800298. 41

C.-Y. Chiu, J. A. Douglas, and X. Li. Cluster analysis for cognitive diagnosis: Theory and applications. *Psychometrika*, 74(4):633–665, 2009. 55

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. 56

A. Corbett. personal communication, April 2013. 1, 30

A. Corbett and J. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, 1995. ISSN 0924-1868.

doi: 10.1007/BF01099821. URL `http://dx.doi.org/10.1007/BF01099821`. 1, 17, 19, 22, 55

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)*, 39(1):1–38, 1977. 9, 26

M. Desmarais. Conditions for Effectively Deriving a Q-Matrix from Data with Non-negative Matrix Factorization. In M. Pechenizkiy and T. Calders and C. Conati and S. Ventura and C. Romero and J. Stamper, editor, *Proceedings of the 4th International Conference on Educational Data Mining*, pages 169–178, Eindhoven, Netherlands, 2011. URL `http://educationaldatamining.org/EDM2011/wp-content/uploads/proc/edm2011_paper35_full_Desmarais.pdf`. 2, 38, 54

A. Doucet, N. d. Freitas, K. P. Murphy, and S. J. Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, UAI '00, pages 176–183, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-709-9. URL `http://dl.acm.org/citation.cfm?id=647234.720075`. 53

J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. URL `http://dx.doi.org/10.3115/1219840.1219885`. 44

Z. Ghahramani and M. Jordan. Factorial Hidden Markov Models. *Machine learning*, 29(2): 245–273, 1997. 8, 11, 56

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

S. J. Goldwater. *Nonparametric bayesian models of lexical acquisition*. PhD thesis, Brown University, Providence, RI, USA, 2007. AAI3271977. 25

J. P. González-Brenes and J. Mostow. What System Differences Matter? Using ℓ_1/ℓ_2-regularization to Compare Dialogue Systems. In *Proceedings of the 12th Annual SIGdial*

Meeting on Discourse and Dialogue, Portland, OR, 2011. 18

J. P. González-Brenes and J. Mostow. Topical HMMs for Factorization of Input-Output Sequential Data. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *paper presented at Personalizing Education Workshop in Advances in Neural Information Processing Systems (NIPS'12)*, Lake Tahoe, CA, 2012.

J. P. González-Brenes and J. Mostow. What and When do Students Learn? Fully Data-Driven Joint Estimation of Cognitive and Student Models. In A. Olney, P. Pavlik, and A. Graesser, editors, *Proceedings of the 6th International Conference on Educational Data Mining*, pages 236–240, Memphis, TN, 2013. 38

A. Hinneburg, R. Preiss, and R. Schröder. TopicExplorer: Exploring document collections with topic models. In *Proceedings of the 2012 European conference on Machine Learning and Knowledge Discovery in Databases - Volume Part II*, ECML PKDD'12, pages 838–841, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-33485-6. URL `http://dx.doi.org/10.1007/978-3-642-33486-3_59`. 59

G. Hooker, M. Finkelman, and A. Schwartzman. Paradoxical results in multidimensional Item Response Theory. *Psychometrika*, 74(3):419–442, 2009. 59

F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. Bayesian updating in recursive graphical models by local computations. *Computational Statistical Quarterly*, 89(4):269–282, 1989. 8

B. W. Junker and K. Sijtsma. Cognitive assessment models with few assumptions, and connections with nonparametric item response theory. *Applied Psychological Measurement*, 25(3): 258–272, 2001. 55

M. Karlovčec, M. Córdova-Sánchez, and Z. Pardos. Knowledge component suggestion for untagged content in an intelligent tutoring system. In S. Cerri, W. Clancey, G. Papadourakis, and K. Panourgia, editors, *Intelligent Tutoring Systems*, volume 7315 of *Lecture Notes in Computer Science*, pages 195–200. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-30949-6. doi: 10.1007/978-3-642-30950-2_25. URL `http://dx.doi.org/10.1007/978-3-642-30950-2_25`. 42, 55

K. R. Koedinger and Roll. Learning to think: Cognitive mechanisms of knowledge transfer. In *The Oxford Handbook of Thinking and Reasoning*, Oxford Library of Psychology, pages 789–806. OUP USA, 2012. ISBN 9780199734689. 22

K. R. Koedinger, R. S. J. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper. A data repository for the EDM community: The PSLC DataShop. In C. Romero, S. Ventura, M. Pechenizkiy, and R. Baker, editors, *Handbook of Educational Data Mining*, pages 43–55, Boca Raton, FL, 2010. CRC Press. 3, 14, 16, 28, 30

Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37, Aug. 2009. ISSN 0018-9162. URL `http://dx.doi.org/10.1109/MC.2009.263`. 20

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, Oct. 1999. ISSN 0028-0836. URL `http://dx.doi.org/10.1038/44565`. 19

J. I. Lee and E. Brunskill. The impact on individualizing student models on necessary practice opportunities. In K. Yacef, O. R. Zaïane, A. Hershkovitz, M. Yudelson, and J. C. Stamper, editors, *Proceedings of the 5th International Conference on Educational Data Mining*, pages 118–125, Chania, Greece, 2012. www.educationaldatamining.org. URL `http://educationaldatamining.org/EDM2012/uploads/procs/Full_Papers/edm2012_full_11.pdf`. 59

N. Li, W. W. Cohen, K. R. Koedinger, and N. Matsuda. A machine learning approach for automatic student model discovery. In M. Pechenizkiy and T. Calders and C. Conati and S. Ventura and C. Romero and J. Stamper, editor, *Proceedings of the 4th International Conference on Educational Data Mining*, pages 31–40, Eindhoven, Netherlands, 2011. www.educationaldatamining.org. ISBN 978-90-386-2537-9. 55

N. Li, W. Cohen, and K. Koedinger. Discovering student models with a clustering algorithm using problem content. In S. D'Mello and R. A. Calvo, editors, *Proceedings of the 6th International Conference on Educational Data Mining*, Memphis, TN, 2013. 42, 43, 44, 47, 55, 58

K. Murphy. The bayes net toolbox for matlab. *Computing science and statistics*, 33(2):1024–1034, 2001. 5, 8

Z. A. Pardos and N. T. Heffernan. Modeling individualization in a bayesian networks implementation of knowledge tracing. In *Proceedings of the 18th international conference on User Modeling, Adaptation, and Personalization*, UMAP'10, pages 255–266, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-13469-6, 978-3-642-13469-2. URL `http://dx.doi.org/10.1007/978-3-642-13470-8_24`. 27

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. 6

G. Qin and L. Hotilovac. Comparison of non-parametric confidence intervals for the area under the ROC curve of a continuous-scale diagnostic test. *Stat Methods Med Res*, 17(2):207–21, 2008. 34

L. Rabiner and B. Juang. An introduction to Hidden Markov Models. *ASSP Magazine, IEEE*, 3 (1):4–16, 1986. 9

G. Rasch. On general laws and the meaning of measurement in psychology. In *Proceedings of the Fourth Berkeley symposium on mathematical statistics and probability*, volume 4, pages 321–333. University of California Press Berkeley, CA, 1961. 54

P. Resnik and E. Hardisty. Gibbs sampling for the uninitiated. Technical Report UMIACS-TR-2010-04, University of Maryland, 2010. 23, 25

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`. 56

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`. 19

B. Szalkai. Generalizing k-means for an arbitrary distance matrix. *CoRR*, abs/1303.6001, 2013. URL `http://arxiv.org/abs/1303.6001`. 42

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. URL `http://www.sciencedirect.com/science/article/pii/S1877050910003194`. 34, 56

K. Toutanova and C. D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics*, EMNLP '00, pages 63–70, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. URL `http:`

//dx.doi.org/10.3115/1117794.1117802. 44

K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. URL http://dx.doi.org/10.3115/1073445.1073478. 44

K. VanLehn. Student Modeling. In M. C. Polson and J. J. Richardson, editors, *Foundations of intelligent tutoring systems*, Hillsdale, NJ, USA, 1988. L. Erlbaum Associates Inc. ISBN 0-805-80053-0. 1

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. 1, 2, 3, 38, 54

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. 27, 34

Y. Xu and J. Mostow. Comparison of methods to trace multiple subskills: Is LR-DBN best? In K. Yacef, O. R. Zaïane, A. Hershkovitz, M. Yudelson, and J. C. Stamper, editors, *Proceedings of the 5th International Conference on Educational Data Mining*, pages 41–48, Chania, Greece, 2012. www.educationaldatamining.org. URL http://educationaldatamining.org/EDM2012/uploads/procs/Full_Papers/edm2012_full_17.pdf. 23

M. Yudelson, K. R. Koedinger, and G. J. Gordon. Individualized bayesian knowledge tracing models. In *Artificial Intelligence in Education - 16th International Conference (AIED 2013)*, pages 171–180, Memphis, TN, 2013. Springer. 59

L. Zhang, D. Agarwal, and B.-C. Chen. Generalizing matrix factorization through flexible regression priors. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 13–20, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0683-6. URL http://doi.acm.org/10.1145/2043932.2043940. 55

Z. Zhang, T. Li, C. Ding, and X. Zhang. Binary matrix factorization with applications. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, ICDM '07, pages 391–400, Washington, DC, USA, 2007. IEEE Computer Society. URL `http://dx.doi.org/10.1109/ICDM.2007.99`. 20