Comparison of methods to trace multiple subskills: Is LR-DBN best?

Yanbo Xu
Carnegie Mellon University
RI-NSH 4105
5000 Forbes Ave, Pittsburgh, PA 15213
yanbox@cs.cmu.edu

Jack Mostow
Carnegie Mellon University
RI-NSH 4103
5000 Forbes Ave, Pittsburgh, PA 15213
mostow@cs.cmu.edu

ABSTRACT
A long-standing challenge for knowledge tracing is how to update estimates of multiple subskills that underlie a single observable step. We characterize approaches to this problem by how they model knowledge tracing, fit its parameters, predict performance, and update subskill estimates. Previous methods allocated blame or credit among subskills in various ways based on strong assumptions about their relation to observed performance. LR-DBN relaxes these assumptions by using logistic regression in a Dynamic Bayes Net. LR-DBN significantly outperforms previous methods on data sets from reading and algebra tutors in terms of predictive accuracy on unseen data, cutting the error rate by half. An ablation experiment shows that using logistic regression to predict performance helps, but that using it to jointly estimate subskills explains most of this dramatic improvement. An implementation of LR-DBN is now publicly available in the BNT-SM student modeling toolkit.

Keywords
Conjunctive knowledge tracing, Dynamic Bayes Nets, logistic regression

1. INTRODUCTION
Knowledge tracing (KT) [1] is widely used to update an intelligent tutor's estimate of the probability that a student has a given skill, based on the student's observable performance on steps that use the skill. KT does not in itself address the issue of how to update multiple subskills used in the same step. This paper compares various approaches to this "multiple subskills problem." Section 2 frames the space of prior (and new) methods. Section 3 describes a recent method named LR-DBN [2]. Section 4 compares LR-DBN against previous methods on two data sets. Section 5 concludes.

2. COMPARATIVE FRAMEWORK
As a framework to compare previous and proposed methods for tracing multiple subskills, we use four aspects to characterize them: how they **represent** the KT model, how they **fit** the model parameters to observations of multi-subskill steps, how they use the model to **predict** performance on such steps, and how they **update** estimates of the subskills based on observed performance.

2.1 Represent model
Previous solutions represent the student's knowledge at step *n* as a hidden state $K^{(n)}$ in a Hidden Markov Model (HMM), shown in Figure 1. It has knowledge parameters *already know* for the probability that the student knew the skill to start with, *learn* for the probability of the transition from not knowing the skill to knowing it, and *forget* (usually assumed to be 0) for the probability of the transition from knowing the skill to not knowing it. It also has performance parameters *guess* for the probability $P(C^{(n)} \mid not\ K^{(n)})$ of performing the step correctly despite lacking the skill, and *slip* for the probability $P(not\ C^{(n)} \mid K^{(n)})$ of performing the step incorrectly despite knowing the skill.

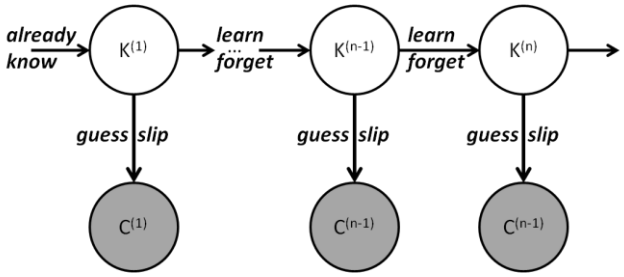

Figure 1: Single-skill knowledge tracing architecture

2.2 Fit parameters
One previous solution [3] tries to sidestep the problem by modeling each set of subskills as a distinct individual skill, e.g., computing the area of a circle embedded in a figure vs. by itself. However, modeling different sets of subskills as independent skills ignores transfer of learning between them.

Other previous solutions [4-6] simply treat each subskill used in a step as if it were entirely responsible for that step. They train a separate KT model for each subskill on observations of all the steps that use it. Thus the same observed step appears in the training data for every subskill that it uses. They simply estimate the model parameters for each subskill using the same training procedure as for conventional KT.

2.3 Predict performance
In standard "single-skill" KT, predicting performance $C^{(n)}$ at step *n* involving skill *j* is simple:

$$P\left(C^{(n)}\right) = P\left(K_j^{(n)}\right) \times (1 - slip_j) + \left(1 - P\left(K_j^{(n)}\right)\right) \times guess_j$$

Equation 1: Standard KT prediction

Previous solutions to the multiple-subskills problem predict performance on a step by combining in different ways the probabilities of correctly performing all the skills it requires. One way, based on an assumption that they are probabilistically independent, multiplies them [4, 6]:

$$P\left(C^{(n)}\right) = \prod_j P\left(K_j^{(n)}\right) \times (1 - slip_j) + \left(1 - P\left(K_j^{(n)}\right)\right) \times guess_j$$

Equation 2: Independent subskills performance prediction

The weakest-subskill alternative [5] takes their minimum:

$$P\left(C^{(n)}\right) = Min_j\, P\left(K_j^{(n)}\right) \times (1 - slip_j) + \left(1 - P\left(K_j^{(n)}\right)\right) \times guess_j$$

Equation 3: Weakest-subskill performance prediction

2.4 Update estimate

To update its estimate of a skill *j* based on the observed success of a step *n* that uses it, standard KT applies Bayes' rule:

$$P_{posterior}\left(K_j^{(n)}\right) = P\left(K_j^{(n)} \middle|\, C^{(n)}\right) = \frac{P\left(K_j^{(n)}\right) \times (1 - slip_j)}{P\left(K_j^{(n)}\right) \times (1 - slip_j) + \left(1 - P\left(K_j^{(n)}\right)\right) \times guess_j}$$

Equation 4: Standard KT skill update for successful step

Conversely, the standard update rule if the step fails is:

$$P_{posterior}\left(K_j^{(n)}\right) = P\left(K_j^{(n)} \middle|\, not\ C^{(n)}\right) = \frac{P\left(K_j^{(n)}\right) \times slip_j}{P\left(K_j^{(n)}\right) \times slip_j + \left(1 - P\left(K_j^{(n)}\right)\right) \times (1 - guess_j)}$$

Equation 5: Standard KT skill update for failed step

Either way, it estimates the probability of knowing the skill at the next step as either knowing without forgetting, or learning:

$$P\left(K_j^{(n+1)}\right) = P_{posterior}\left(K_j^{(n)}\right) \times (1 - forget_j) + \left(1 - P_{posterior}\left(K_j^{(n)}\right)\right) \times learn_j$$

Equation 6: Standard KT next-step update

When a step involves multiple subskills, previous methods use different ways to allocate responsibility among them for the observed success or failure of the step. The "full responsibility" approach applies these equations to all the subskills. The "update weakest subskill" approach simply applies the standard update equations above to whichever subskill in a step has the lowest probability, and leaves the others unchanged. Its "blame weakest, credit rest" variant credits the other subskills as correct even if the step failed.

Conjunctive knowledge tracing (CKT) [6] also predicts the probability of a step succeeding as a product of its subskill probabilities using Equation 2, and gives all of them full credit for success using Equation 4. However, rather than place full blame on each subskill for failure, CKT apportions blame among them differently. Instead of using Equation 5 to update each subskill based just on its own *guess* and *slip* probabilities, CKT takes into account those of the other subskills as well, as follows.

Bayes' rule says how to update a skill based on performance:

$$P_{posterior}\left(K_j^{(n)} \mid not\ C^{(n)}\right) = \frac{P\left(not\ C^{(n)} \mid K_j^{(n)}\right) \times P(K_j^{(n)})}{P(not\ C^{(n)})}$$

Equation 7: Bayes' rule for skill update

Conditioning on having skill *j* at step *n* reduces $P(K_j^{(n)})$ to 1, simplifying the numerator of Equation 9 to:

$$P\left(not\ C^{(n)} \mid K_j^{(n)}\right) = slip_j + (1 - slip_j) \times \prod_{i \neq j} \left[P(K_i^{(n)}) \times (1 - slip_i) + \left(1 - P(K_i^{(n)})\right) \times guess_i\right]$$

Equation 8: CKT subskills update for failed step

CKT computes the denominator by assuming independence:

$$P\left(not\ C^{(n)}\right) = 1 - P\left(C^{(n)}\right) = 1 - \prod_j \left[P(K_j^{(n)}) \times (1 - slip_j) + \left(1 - P(K_j^{(n)})\right) \times guess_j\right]$$

Equation 9: CKT prediction based on multiple subskills

Next, we introduce a different strategy of using logistic regression in KT to trace multiple subskills.

3. USING LOGISTIC REGRESSION TO TRACE MULTIPLE SUBSKILLS

We now describe two newer methods that trace multiple subskills using logistic regression. Previous KT methods fit their parameters independently using the same algorithm as for single-skill KT, thereby implicitly assigning full and equal responsibility to all the subskills in an observed step, and predict performance based on the weakest subskill or by multipying subskill estimates. Section 3.1 describes LR-DBN, which changes both these aspects by using Expectation Maximization (EM) [7] to fit parameters for multiple subskills simultaneously, and by using logistic regression to predict performance. As an ablation experiment to shed light on the relative impact of these two innovations, Section 3.2 introduces LR-DBN Minus, a hybrid method that fits the standard KT model just as previous methods do, but uses logistic regression to do prediction.

3.1 LR-DBN

LR-DBN is a recent but published method [2, 8] to trace multiple subskills, so we summarize it here only briefly in terms of the four aspects discussed in Section 2.

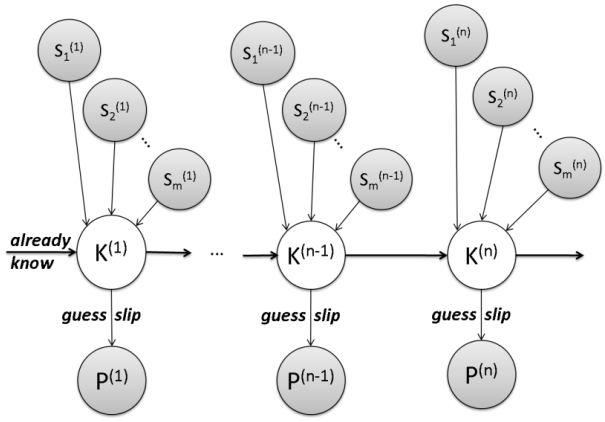

Figure 2: Knowledge tracing with logistic regression

Represent model: Like standard KT, LR-DBN represents the knowledge for step *n* as a hidden knowledge state $K^{(n)}$ in a dynamic Bayes net. However, as Figure 2 illustrates, LR-DBN

adds a layer of observable states $S_j^{(n)}$ as indicator variables to represent whether step *n* involves subskill *j*: 1 if so, 0 if not.

LR-DBN uses logistic regression to model the initial hidden knowledge state at step *0* and the transition probabilities from step *n-1* to *n* as follows:

$$P(not\ K^{(0)}) = \frac{exp\ (-\sum_{j=1}^{m} \beta_j^{(0)}\ S_j^{(0)})}{1 + exp\ (-\sum_{j=1}^{m} \beta_j^{(0)} S_j^{(0)})}$$

$$P(not\ K^{(n)} \mid not\ K^{(n-1)}) = \frac{exp\ (-\sum_{j=1}^{m} \beta_j\ S_j^{(n)})}{1 + exp\ (-\sum_{j=1}^{m} \beta_j\ S_j^{(n)})}$$

$$P(not\ K^{(n)} \mid K^{(n-1)}) = \frac{exp\ (-\sum_{j=1}^{m} \gamma_j\ S_j^{(n)})}{1 + exp\ (-\sum_{j=1}^{m} \gamma_j\ S_j^{(n)})}$$

Equation 10: Logistic regression to combine subskills

These three conditional probabilities for each subskill replace the KT knowledge parameters *already know*, *learn*, and *forget***,** but LR-DBN retains KT's *guess* and *slip* parameters at each step.

Fit parameters: LR-DBN uses Expectation Maximization (EM) [7] to fit parameters for all subskills together.

Predict performance: LR-DBN uses logistic regression in a Dynamic Bayes Net to combine multiple subskills more flexibly than using Equation 2 to multiply their probabilities or Equation 3 to take their minimum, but it uses Equation 1 to predict expected performance based on estimated knowledge, *guess*, and *slip*.

Update estimates: LR-DBN uses the same Bayes rule as single-skill KT to update its estimate of the hidden knowledge state in Equation 4, Equation 5, and Equation 6.

3.2 LR-DBN Minus

LR-DBN Minus is a hybrid of LR-DBN and standard KT. It combines KT's single-skill fitting process with LR-DBN's update and prediction based on logistic regression. The key is to convert the probability of knowing a subskill into a coefficient in logistic regression. LR-DBN uses logistic regression to model the transition probabilities between knowledge states, as well as the relation of the knowledge state at each step to the subskills it involves [2]. Thus, given the set of subskills $\{S_j^{(n)}\}$ used at step n, a set of coefficients $\{\beta_j^{(n)}\}$ exists such that

$$P(K^{(n)}) = 1 - sigmoid\left(\sum_{j=1}^{m} \beta_j^{(n)} S_j^{(n)}\right)$$

Equation 11: Logistic regression for the knowledge state

If we assume step *n* requires only a single subskill *i*, then $S_j^{(n)} = 0$ for all the j's such that j$\neq i$, and transformations between the probabilities $P\left(K_i^{(n)}\right)$ and the coefficients $\beta_i^{(n)}$ are:

$$P\left(K_i^{(n)}\right) = 1 - sigmoid\left(\beta_i^{(n)}\right)$$

$$\beta_i^{(n)} = logit(1 - P\left(K_i^{(n)}\right))$$

Equation 12: Transformation between probabilities and logistic regression coefficients

To update the estimates, we need to distribute the update at each step that is calculated either from Equation 4 or Equation 5 to the subskill coefficients. We assume that the coefficient for each subskill changes by the same amount $\Delta\beta$ when updated:

$$P_{posterior}\left(K^{(n)}\right) = 1 - sigmoid(\sum_{j=1}^{m} (\beta_j^{(n)} + \Delta\beta)\ S_j^{(n)})$$

Equation 13: Update coefficients in LR-DBN Minus

Then the update of each subskill becomes:

$$P_{posterior}\left(K_j^{(n)}\right) = 1 - sigmoid((\beta_j^{(n)} + \Delta\beta)\ S_j^{(n)})$$

Equation 14: Update subskills in LR-DBN Minus

Next we still use the standard KT Equation 6 to update subskills at step *n+1*. Now we have successfully transformed LR-DBN to upate on standard KT parameters. Note that we replace the separate $guess_j$ and $slip_j$ parameters for each subskill *j* with uniform *guess* and *slip* for all the steps. The reason is that LR-DBN combines subskills to estimate the probability of the student knowing a step and then uses *guess* and *slip* to predict performance. In contrast, previous methods apply $guess_j$ and $slip_j$ to each subskill *j* before combining them to predict performance on the step.

4. EXPERIMENTAL EVALUATION

To compare LR-DBN and LR-DBN Minus to previous methods for tracing multiple subskills, we fit seven models to real data, summarized in Table 1: LR-DBN, LR-DBN Minus, CKT, and three variants of standard KT distinguished by how they update estimated skills: "full responsibility," "blame weakest, credit rest," and "update weakest subskill," with majority class as an additional baseline. Sections 4.1 and 4.2 describe our data and results.

Table 1: Summary of models compared

<table>
<tr><th>Models</th><th>Fit</th><th>Predict</th><th>Update</th></tr>
<tr><td>LR-DBN</td><td>Train subskills together. Logistic regression assigns responsibility.</td><td rowspan="2">Logistic regression on subskill estimates.</td><td>Update subskills together. Logistic regression assigns responsibility.</td></tr>
<tr><td>LR-DBN Minus</td><td rowspan="4">Train subskills separately. Assign each one full responsibility.</td><td rowspan="2">Update subskills separately, each with full responsibility.</td></tr>
<tr><td>CKT</td><td>Multiply subskill estimates.</td></tr>
<tr><td>Full responsibility</td><td rowspan="2">Minimum of subskill estimates.</td></tr>
<tr><td>Blame weakest, credit rest</td><td rowspan="2">Update only the weakest subskill.</td></tr>
<tr><td>Update weakest subskill</td><td></td></tr>
<tr><td>Majority class</td><td>Identify larger class</td><td>Majority class</td><td>No update</td></tr>
</table>

4.1 Data sets

We train and test the models on real data from two tutors used at schools. One data set is from children using Project LISTEN's Reading Tutor [9] at primary schools during the 2005-2006 school year. To model their oral reading fluency, we define performance $C^{(n)}$ as whether the Reading Tutor scored a text word as read fluently at step *n*, i.e., read without help or hesitation and recognized by the automated speech recognizer. We assume that whether a student read a word fluently depended on whether the student knew the requisite subskills, namely the grapheme-to-phoneme mappings in the word. Due to the large amount of data (1,792,103 read words from 275 students), we randomly selected 20 children who read a total of 80,268 words (3,972 distinct word types) with 320 unique grapheme-phoneme mappings. To counteract the prevalence of high-frequency words like *the*, we include at most the first 20 of each student's encounters of a word in the training data, leaving 24,145 read words. We do not limit the test data, so it includes 40,867 words.

The other data set [10] came from 123 high school students working on a geometry area unit of the Bridge to Algebra Cognitive Tutor®. The model for each student includes the same 50 subskills, and predicts whether the student will perform a step correctly. Again we include at most the first 20 of each student's encounters of a step in the training data, leaving 11,730 algebra practice steps, but 22,737 steps of test data.

The data sets from both tutors are unbalanced. The Reading Tutor scored 68.84% of the words in the training set as fluent, and 74.31% of the words in the test set. The Algebra Tutor rated 74.22% of the steps in the training data as correct, and 84.63% of the steps in the test data.

We fit each model separately for each student, as opposed to training a single model on the data for all the students. One reason is computational expedience: unlike methods that fit a separate model for each subskill, LR-DBN fits a single model for all the subskills, which involves processing much more data at a time. Training this model on all the students' data at once would be computationally unwieldy. The other reason is to compare methods fairly. Except for LR-DBN, it is feasible to train a single model of a subskill on the data for all the students, and in fact we tried it, but the resulting model does not perform as well as training a separate model for each student.

For all the methods, we fit the model for each student to the first half of the student's steps, and test it on the second half. We report average per-student accuracy on the unseen test data, weighting its mean and variance by per-student sample size to derive 95% confidence intervals. We use paired T-tests, paired by student, to rate LR-DBN's accuracy against each other method.

4.2 Results

Table 2 and Table 3 list all seven methods in decreasing order of their binary predictive accuracy on the test data. LR-DBN dramatically outpredicts all the other methods. LR-DBN's overall accuracy on the Reading Tutor data is 13% higher than majority class, vs. only 1% for the next method. For the Algebra Tutor data, LR-DBN is the only method that beats the majority class, by 7%. That is, on both data sets, LR-DBN has only half the error rate of the next best method.

For unbalanced data, accuracy on the minority class can be especially important. The minority class in our tutor data represents negative student outcomes to remediate by means of practice and instruction. LR-DBN beats every other method on the minority class by over 20% absolute in both data sets.

Table 2: Mean per-student accuracy on Reading Tutor data (95% confidence interval in parentheses) compared to LR-DBN is significantly (p<.01) worse if underlined, or *better if italicized*.

Models	Accuracy	Accuracy Within Positive Class	Accuracy Within Negative Class
LR-DBN	87.31% (±1.90%)	91.17% (±2.80%)	75.80% (±12.53%)
Update weakest subskill	74.53% (±4.55%)	*95.06%* (±2.73%)	15.15% (±5.29%)
Majority class	74.31%	*100.00%*	0.00%
LR-DBN Minus	74.11% (±5.05%)	90.71% (±7.89%)	26.09% (±11.61%)
Blame weakest, credit rest	73.90% (±4.59%)	92.36% (±3.86%)	20.52% (±6.43%)
CKT	72.79% (±3.99%)	89.47% (±3.52%)	24.52% (±7.76%)
Full responsibility	66.20% (±5.39%)	72.30% (±10.24%)	48.53% (±12.87%)

Table 3: Mean per-student accuracy on Algebra Tutor data is significantly (p<.001) worse than LR-DBN's where underlined; *italicized values* are significantly better.

Models	Accuracy	Accuracy Within Positive Class	Accuracy Within Negative Class
LR-DBN	91.99% (±2.00%)	96.5% (±1.30%)	72.3% (±7.80%)
Majority class	84.63%	*100.00%*	0.00%
CKT	84.38% (±1.14%)	*99.03%* (±0.26%)	20.44% (±3.11%)
Full responsibility	84.27% (±1.13%)	95.65% (±0.88%)	34.55% (±4.60%)
LR-DBN Minus	83.92% (±1.17%)	97.23% (±0.62%)	25.80% (±3.84%)
Blame weakest, credit rest	80.38% (±1.13%)	90.70% (±0.72%)	35.28% (±3.14%)
Update weakest subskill	79.59% (±1.19%)	91.13% (±0.69%)	29.20% (±2.76%)

What does comparison to LR-DBN Minus reveal about the relative contributions of the fitting and update procedures? LR-DBN Minus uses the same fitting procedure as conventional knowledge tracing, but uses logistic regression to update estimates. It performs substantially worse than LR-DBN, and comparably to the other methods. We conclude that LR-DBN's accuracy benefits more from its fitting procedure than from using logistic regression to combine estimates of hidden subskills.

Why does LR-DBN outpredict the other methods? Possible reasons include the strong assumptions that it avoids, but which they make implicitly by fitting and updating subskill estimates separately, multiplying them to predict performance on a step, and assigning each subskill full responsibility for the step's outcome. Inspection of Table 1 reveals that this last assumption is the only one they all have in common, implicating it as the likeliest culprit.

Predictive accuracy is just one way to evaluate student models. A more sensitive metric is model fit as measured by data likelihood, penalized by model complexity. Table 4 and Table 5 list the complexity-penalized model fits of the methods on the two data sets in increasing order, as scored by the Akaike information criterion (AIC) [11] and Bayesian information criterion (BIC) [12], defined respectively as:

$$AIC = 2k - 2\ln(L)$$

$$BIC = -2\ln(L) + k \cdot ln(n)$$

Equation 15: Formulas for calculating AIC and BIC

Table 4: Complexity-adjusted Reading Tutor training data fit

Models	AIC	BIC	*k*
LR-DBN	75,054.52	231,226.89	19,300
LR-DBN Minus	120,259.60	239,606.70	12,840
CKT	145,779.60	383,730.20	25,600
Full responsibility			
Blame weakest, credit rest			
Update weakest subskill			

Table 5: Complexity-adjusted Algebra Tutor training data fit

Models	AIC	BIC	*k*
LR-DBN	60,545.20	201,052.44	19,065
LR-DBN Minus	43,195.94	143,962.30	12,546
CKT	67,303.94	264,885.00	24,600
Full responsibility			
Blame weakest, credit rest			
Update weakest subskill			

Both AIC and BIC measure model fit as log-likelihood of the training data, *ln(L)*, penalized by model complexity (number of parameters, *k*). BIC also penalizes the number of observations, *n*. We calculate the number of parameters per student as follows:

LR-DBN fits the 3 groups of coefficients for each of the subskills and one intercept in Equation 10, plus two shared parameters, *guess* and *slip*. For the Reading Tutor data set, this number totals 3 × (320 + 1) + 2 = 965, multiplied by the 20 children in the data sample. For the Algebra Tutor data set, it totals 3 × (50 + 1) + 2 = 155, multiplied by 123 students.

LR-DBN Minus fits 2 parameters (*already know* and *learn*) per subskill, plus 2 shared parameters (*guess* and *slip*). This number of parameters per student totals (2 × 320) + 2 = 642 for the Reading Tutor and (2 × 50) + 2 = 102 for the Algebra Tutor.

The other methods fit 4 parameters (*already know*, *learn*, *guess* and *slip*) per subskill for each student, totaling 4 × 320 = 1280 for the Reading Tutor, and 4 × 50 = 200 for the Algebra Tutor.

Thus compared to previous methods, LR-DBN has about 1 less parameter per subskill, and LR-DBN Minus about 2 less.

What about the number *n* of observations? LR-DBN uses one observation per step to fit all the subskill parameters. In contrast, the other methods fit each subskill separately, assigning it full responsibility for every step that uses it, as if observing it separately for each subskill. Counting such duplicate observations as separate, they use three times as many Reading Tutor observations as LR-DBN, and twice as many Algebra Tutor observations.

Table 6 and Table 7 show the average log-likelihood of steps in the training and test data. All the methods except LR-DBN share the same likelihood on the training data because they fit parameters in the same way (as shown in Table 1).

Table 6: Average log-likelihood for the Reading Tutor data

Models	On training data	On unseen test data
LR-DBN	-0.7549	-0.3555
CKT	-1.9586	-1.1330
Full responsibility		-1.2230
Blame weakest, credit rest		-1.4944
LR-DBN Minus		-1.5690
Update weakest subskill		-1.6665

Table 7: Average log-likelihood for the Algebra Tutor data

Models	On training data	On unseen test data
LR-DBN	-0.9555	-0.1503
CKT	-0.7717	-0.2082
Full responsibility		-0.2065
Blame weakest, credit rest		-0.2529
LR-DBN Minus		-0.2364
Update weakest subskill		-0.2816

Normally one might expect log-likelihood to be lower for test data than training data, by an amount reflecting the degree of overfitting. However, the models assign higher likelihood to correct steps because, as Section 4.1 mentioned, they are more common than incorrect steps in the training data, and this difference is more pronounced in the test data. Its likelihood is therefore *higher*, and hence is not a direct gauge of overfitting.

Table 6 and Table 7 reveal that LR-DBN's log-likelihood is by far the highest on unseen test data from both tutors, consistent with how dramatically it outpredicts the other methods, even though they have higher log-likelihood on the training data from the Algebra Tutor. This reversal from training to test data suggests that the other methods might overfit that training data.

In summary, LR-DBN has a smaller number *k* of parameters than the other methods (except for LR-DBN Minus), a smaller number *n* of observations (counting duplicate observations as distinct), and higher likelihood on Reading Tutor training data, where it achieves the lowest AIC and BIC scores. Most important, LR-DBN far surpasses all the other methods in accuracy and log-likelihood on unseen test data from both tutors.

5. IMPLEMENTATION

To make LR-DBN publicly available[1], we added it to the Bayes Net Toolkit for Student Modeling (BNT-SM) [13]. BNT-SM inputs a data set and a DBN student model (not only the simple one used in standard knowledge tracing), specified in XML. It generates and executes BNT code to train and test the model, and outputs Excel files containing the parameter estimates and inference results. BNT is an open-source Matlab package[2] that supports many learning and inference algorithms for both static and dynamic Bayes models. BNT-SM hides most of the BNT coding details, freeing users to focus on constructing the student models rather than on programming them.

Using BNT-SM consists of four phases [13]:

1. Specify the data source in an XML specification.
2. Specify the DBN structure in XML.
3. Specify and initialize parameters in XML.
4. Call RunBnet.m in Matlab.

To fit LR-DBN on the Reading Tutor data with 320 subskills, we specify the structure shown in Figure 2 to BNT-SM in XML, as shown in the APPENDIX.

6. CONCLUSIONS

This paper makes multiple contributions to knowledge tracing:

First, we present a framework to characterize previous and new methods for tracing multiple subskills by how they (1) model knowledge tracing, (2) fit its parameters, (3) predict performance, and (4) update subskill estimates.

Second, we use data sets from reading and algebra tutors to compare LR-DBN against previous methods in terms of AIC, BIC, and predictive accuracy on unseen data, and show that LR-DBN performs significantly better on both data sets on all three metrics, cutting the best previous prediction error rate in half.

Third, we introduce the hybrid LR-DBN Minus method, which fits the same standard KT model as previous methods, but uses logistic regression to predict student performance.

Fourth, by comparing LR-DBN Minus to LR-DBN, we show that using logistic regression to predict performance suffices to beat previous methods, but that using logistic regression EM to jointly estimate subskills accounts for most of LR-DBN's superior performance.

Finally, in order to amplify the impact of this work, we have made LR-DBN publicly available and easy to extend to other student modeling with dynamic Bayes nets, by incorporating it into the latest version of the BNT-SM student modeling toolkit [13] used in previous studies of knowledge tracing [e.g., 14].

This work has several limitations for future work to address.

First, LR-DBN has so far been applied just to simple knowledge tracing of multiple subskills, but it can apply to any DBN. Future work could use LR-DBN to improve other DBN student models, for example to measure more accurately the scaffolding and learning effects of tutor help [14].

Second, LR-DBN needs 5.5 hours on average per student to fit and update; the other methods take less than 1 hour to fit a single set of parameters for all the students and subskills, and 2-5 minutes to update. Future work may train LR-DBN faster or develop other methods that are faster to train. Such work might adapt two previous types of cognitive diagnosis models that operate on static data and have statistical learning algorithms, both EM and MCMC [15]. NIDA (Noisy Inputs, Deterministic "And" gate) models [16] resemble CKT because it applies *guess* and *slip* to individual subskills before combining them conjunctively. DINA (Deterministic Inputs, Noisy "And" gate) models [17] resemble LR-DBN because it combines subskills (with logistic regression) before applying *guess* and *slip* to the resulting knowledge state. Extending either type of model to apply to knowledge tracing may improve LR-DBN itself.

Finally, although LR-DBN traces multiple subskills better than previous methods, it (like them) must be told which steps use which subskills. Future work may infer this information automatically [18].

ACKNOWLEDGMENTS

This work was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305A080628 to Carnegie Mellon University. The opinions expressed are those of the authors and do not necessarily represent the views of the Institute or U.S. Department of Education. We thank Ken Koedinger for his CKT implementation and algebra tutor data, and children, schools, and LISTENers for our Reading Tutor data.

[1] At http://www.cs.cmu.edu/~listen/BNT-SM

[2] At http://code.google.com/p/bnt

APPENDIX

To use LR-DBN in BNT-SM, we first specify its data source:

```
<multi_subskill>yes</multi_subskill>
<input>
   <evidence_train>evidence.train.xls</evidence_train>
   <evidence_test>evidence.test.xls</evidence_test>
</input>
<output>
   <param_table>param_table.xls</param_table>
   <inference_result>inference_result.xls</inference_result>
   <inference_result_header>inference_result.xls</inference_re
sult_header>
   <log>log.txt</log>
</output>
```

To add logistic regression to standard knowledge tracing, we represent the ***320*** subskills as a single ***multi*** node ***kc***, which transits to the latent node ***knowledge*** within a step. The hidden state of ***knowledge*** transits both to the output ***fluent*** within the current step and to the ***knowledge*** state at the next step:

```
<nodes>
   <node>
         <id>1</id>
         <name>kc</name>
         <type>multi</type>
         <values>320</values>
         <latent>no</latent>
         <prefix_field>kc</prefix_field>
         <within>
               <transition>knowledge</transition>
         </within>
         <between></between>
   </node>
   <node>
         <id>2</id>
         <name>knowledge</name>
         <type>discrete</type>
         <values>2</values>
         <latent>yes</latent>
         <field> knowledge</field>
         <within>
         <transition>fluent</transition>
         </within>
         <between>
         <transition>knowledge</transition>
         </between>
   </node>
   <node>
         <id>3</id>
         <name>fluent</name>
         <type>discrete</type>
         <values>2</values>
         <latent>no</latent>
         <field>fluent</field>
         <within></within>
         <between></between>
   </node>
</nodes>
```

Then we define and set initial values of the LR-DBN parameters. We specify the input node ***kc*** as ***root*** to have no parents and no parameters, the latent node ***knowledge*** as ***softmax*** to have a multinomial logit function, and the output node ***fluent*** to have a simple ***discrete*** conditional probability table, with random initial parameter values in LR-DBN's EM fitting algorithm:

```
<eclasses>
   <eclass>
         <id>1</id>
         <formula>P1(kc)</formula>
         <type>root</type>
   </eclass>
   <eclass>
         <id>2</id>
         <formula>P2(knowledge </formula>
         <type>softmax</type>
         <cpd>
               <eq>P2(T)</eq>
               <init>rand</init>
               <param>L0</param>
               <eq>P2(F)</eq>
               <init>1-P1(T)</init>
               <param>null</param>
         </cpd>
   </eclass>
   <eclass>
         <id>3</id>
         <formula>P3(fluent| knowledge </formula>
         <type>discrete</type>
         <cpd>
               <eq>P3(T|F</eq>
               <init>rand</init>
               <param>guess</param>
               <eq>P3(F|T)</eq>
               <init>rand</init>
               <param>slip</param>
               <eq>P3(F|F)</eq>
               <init>1-P3(T|F)</init>
               <param>null</param>
               <eq>P3(T|T)</eq>
               <init>1-P3(F|T)</init>
               <param>null</param>
         </cpd>
   </eclass>
   <eclass>
         <id>4</id>
         <formula>P4(knowledge| knowledge)</formula>
         <type>softmax</type>
         <cpd>
               <eq>P4(T|F </eq>
               <init>rand</init>
               <param>learn</param>
               <eq>P4(F|T)</eq>
               <init>rand</init>
               <param>forget</param>
               <eq>P4(F|F </eq>
               <init>1-P4(T|F)</init>
               <param>null</param>
               <eq>P4(T|T)</eq>
               <init>1-P4(F|T)</init>
               <param>null</param>
         </cpd>
   </eclass>
</eclasses>
```

REFERENCES

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

[2] Xu, Y. and J. Mostow. *Using Logistic Regression to Trace Multiple Subskills in a Dynamic Bayes Net*. in *Proceedings of the 4th International Conference on Educational Data Mining* 2011. Eindhoven, Netherlands.

[3] Cen, H., K. Koedinger, and B. Junker. *Learning Factors Analysis – A General Method for Cognitive Model Evaluation and Improvement*. in *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*. 2006. Jhongli, Taiwan.

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

[5] Gong, Y., J. Beck, and N.T. Heffernan. *Comparing Knowledge Tracing and Performance Factor Analysis by Using Multiple Model Fitting Procedures*. in *Proceedings of the 10th International Conference on Intelligent Tutoring Systems*. 2010. Pittsburgh, PA: Springer Berlin / Heidelberg.

[6] Koedinger, K.R., et al., *Avoiding Problem Selection Thrashing with Conjunctive Knowledge Tracing*, in *Proceedings of the 4th International Conference on Educational Data Mining*. 2011: Eindhoven, NL. p. 91-100.

[7] Dempster, A., N. Laird, and D. Rubin, *Maximum Likelihood from Incomplete Data via the EM Algorithm.* Journal of the Royal Statistical Society Series B (Methodological), 1977. **39(1)**: p. 1-38.

[8] Xu, Y. and J. Mostow. *Logistic Regression in a Dynamic Bayes Net Models Multiple Subskills Better! [Best Poster Nominee]*. in *Proceedings of the 4th International Conference on Educational Data Mining* 2011. Eindhoven, Netherlands.

[9] Mostow, J. and G. Aist, *Evaluating tutors that listen: An overview of Project LISTEN*, in *Smart Machines in Education*, K. Forbus and P. Feltovich, Editors. 2001, MIT/AAAI Press: Menlo Park, CA. p. 169-234.

[10] Koedinger, K.R., et al., *A Data Repository for the EDM community: The PSLC DataShop*, in *Handbook of Educational Data Mining*, C. Romero, et al., Editors. 2010, CRC Press: Boca Raton, FL. p. 43-55.

[11] Akaike, H., *A new look at the statistical model identification*. IEEE Transactions on Automatic Control, 1974. **19**(6): p. 716–723.

[12] McQuarrie, A.D.R. and C.-L. Tsai, *Regression and Time Series Model Selection*. 1998: World Scientific.

[13] Chang, K.-m., et al., *A Bayes Net Toolkit for Student Modeling in Intelligent Tutoring Systems*, in *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, K. Ashley and M. Ikeda, Editors. 2006: Jhongli, Taiwan. p. 104-113.

[14] Beck, J.E., et al. *Does help help? Introducing the Bayesian Evaluation and Assessment methodology*. in *9th International Conference on Intelligent Tutoring Systems*. 2008. Montreal.

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

[16] Hartz, S., *A Bayesian framework for the unified model for assessing cognitive abilities: Blending theory with practicality*. 2002, University of Illinois at Urbana-Champaign: Unpublished doctoral dissertation.

[17] Torre, J.d.l. and J. Douglas, *Higher-order latent trait models for cognitive diagnosis.* Psychometrika, 2004. **69**: p. 333-353.

[18] González-Brenes, J.P. and J. Mostow. *Dynamic Cognitive Tracing: Towards Unified Discovery of Student and Cognitive Models*. in *Proceedings of the Fifth International Conference on Educational Data Mining*. 2012, in press. Chania, Crete, Greece.