Automated Student Model Improvement

Kenneth R. Koedinger
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
koedinger@cmu.edu

Elizabeth A. McLaughlin
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
mimim@cs.cmu.edu

John C. Stamper
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
john@stamper.org

ABSTRACT

Student modeling plays a critical role in developing and improving instruction and instructional technologies. We present a technique for automated improvement of student models that leverages the DataShop repository, crowd sourcing, and a version of the Learning Factors Analysis algorithm. We demonstrate this method on eleven educational technology data sets from intelligent tutors to games in a variety of domains from math to second language learning. In at least ten of the eleven cases, the method discovers improved models based on better test-set prediction in cross validation. The improvements isolate flaws in the original student models, and we show how focused investigation of flawed parts of models leads to new insights into the student learning process and suggests specific improvements for tutor design. We also discuss the great potential for future work that substitutes alternative statistical models of learning from the EDM literature or alternative model search algorithms.

Keywords

Student modeling, model discovery, model comparison.

1. INTRODUCTION

Student models drive the great many instructional decisions that automated tutors currently make, whether it is how to organize instructional messages, sequence topics and problems in a curriculum, adapt pacing to student needs, or select appropriate materials and tasks. Student models also appear critical to accurate assessment of self-regulated learning skills or motivational states. A better student model yields better instruction, which leads to improved learning. More accurate skill diagnosis leads to better prediction of what a student knows which provides better assessment. Better assessment leads to more efficient learning overall.

Cognitive Task Analysis (CTA) has been shown to lead to better instruction [2], and CTA is currently the best strategy for creating cognitive models. Unfortunately, CTA is an expensive and time intensive process that is fundamentally driven by human experts. The main goal of this research is to accelerate the process of improving student models. Educational data mining and machine learning techniques can be used to improve these models in an automated fashion. In this research, we combine an automated search algorithm with existing proposed student models in the DataShop repository in a crowd sourcing fashion. Using this process, we have been able to make improvements in the models that lead to important focused insights into ways to improve the instruction of the tutoring systems that use these models.

1.1 Background and Related Work

A number of studies have demonstrated how detailed CTA can result in dramatically better instruction [2; 7; 9]. Cognitive Task Analysis clearly works in creating cognitive models but as it currently exists, CTA has several limitations. First, CTA is more of an art than a science involving many subjective decisions. Second, the most successful CTA approaches are heavy in human effort. Approaches like structured interviews, think alouds, or developing cognitive model simulations all require high level of psychological expertise and significant time investment.

It may be possible to achieve similar outcomes using more automated techniques that utilize educational data mining and machine learning on large sets of student data. These techniques can reap many of the benefits of CTA, but with less effort and expertise than currently required. CTA typically produces a symbolic representation of a student model, for instance, a rule-based production system of the skills in a domain. An alternative is to use data and statistical inference to create a student model involving continuous parameters over latent variables with links to observed student performance variables.

In domains where cognitive models have been created, the learning curves derived from these models are a source for data-driven model revision. Others have applied learning curve analysis in the improvement of tutors. In one study of a constraint-based tutor for teaching database programming [10], learning curve analysis was used to analyze log data and make student model improvements. Creating models of student performance is useful both for student assessment and for better student instruction. With respect to assessment, use of such models in on-line systems might in fact be just as good at the job that standardized tests are intended to perform [5]. With respect to better instruction, such models are the basis for the kind of student-customized adaptive instruction that intelligent tutoring systems can provide [6]. Cognitive Tutors for mathematics are now in use in more than 2600 schools across the US for some 600,000 students per year. While these systems have been quite successful, there is room for improvement in the student models that drive their behavior.

The DataShop repository at the Pittsburgh Science of Learning Center (http://learnlab.org/datashop) provides a resource for educators and researchers to create, modify, and evaluate student models [8]. DataShop is an open data repository for educational data with associated visualization and analysis tools. DataShop has data from thousands of students derived from interactions with on-line course materials and intelligent tutoring systems. The data is fine-grained, with student actions recorded roughly every 20 seconds, and it is longitudinal, spanning semester or yearlong courses. As of December 2011, over 300 datasets are stored including over 70 million student actions, which equates to over 190,000 student hours of data. Most student actions are “coded” meaning they are not only graded as correct or incorrect, but are categorized in terms of the hypothesized competencies or “Knowledge Components” needed to perform that action.

In DataShop terminology, Knowledge Components (KCs) are used to represent pieces of knowledge, concepts or skills that students need to solve problems. When a specific set of KCs are mapped to a set of instructional tasks (usually steps in problems) they form a KC Model. A KC model is a specific kind of student model. DataShop provides an easy interface for exporting, modifying, and importing KC models, allowing researchers to evaluate and compare alternative KC models.

1.2 Student Model Improvement

A number of automated and hand search methods of exploring the cognitive model space have been proposed [1; 4; 10; 11; 13; 14]. These approaches create alternative models that are scored against existing models using one of several metrics for model prediction of student performance and how it changes over time. These include Akaike information criterion (AIC), Bayesian information criterion (BIC), and cross validation [12]. A statistical model is needed to make predictions about changes in student performance and DataShop uses an extension of item response theory that incorporates a growth or learning term [cf., 3; 14]. We refer to this model as the "Additive Factors Model" (AFM) [1; 13] and it is shown in Figure 1. In this statistical model, the discrete portion of the student model is represented by q_{jk}, the so-called "Q matrix" [16], which maps hypothesized difficulty or learning factors (the knowledge components or skills) to steps in problems. These factors are hypothesized causes for difficulty (β_k) or for learning improvement as students practice (γ_k). AFM gives a probability that a student i will get a problem step j correct based on the student's baseline proficiency (θ_i), the baseline difficulty (β_k) of the required KCs (q_{jk}), and the improvement (γ_k) in those KCs as the student gets practice opportunities (T_{ik}).

$$\ln\left(\frac{p_{ij}}{1-p_{ij}}\right) = \theta_i + \sum_{k=1}^{K} q_{jk}\beta_k + \sum_{k=1}^{K} q_{jk}\gamma_k T_{ik}$$

Figure 1. In the Additive Factors Model (AFM), the probability student i gets step j correct (p_{ij}) is proportional to the overall proficiency of student i (θ_i) plus for each factor or knowledge component k present for this step j (indicated by q_{jk}), add the base difficulty of that factor (β_k) and the product of the number of practice opportunities this student (i) has had to learn this factor (T_{ik}) and the amount gained for each opportunity (γ_k).

Previous efforts to evaluate cognitive models have used BIC as the evaluation criteria [1]. BIC reduces the chances of over-fitting the data by penalizing for increasing the number of parameters in the model. It is much faster to compute than cross validation and reasonably predicts the results of cross validation. When time is not an issue, cross validation is preferred. There is currently no consensus on how to perform the folding process in cross validation for student model comparison and we discuss three alternatives below (which are in use in DataShop).

2. THE CREATION AND EVALUATION OF STUDENT MODELS IN DATASHOP

Before discussing automated generation of student models, we first describe how DataShop supports researchers in creating and evaluating alternative knowledge component-based student models (represented as Q matrices). The log data collected in DataShop is composed of student attempts on problem steps in a given set of instruction. Each of these problem steps can be tied to one or more skills or knowledge components. This linking of problem steps to knowledge components is called a KC model in DataShop and represents a student model for that set of instruction. Researchers can export KC models from DataShop, modify them using Excel or another editor, and then import a new model into DataShop for comparison.

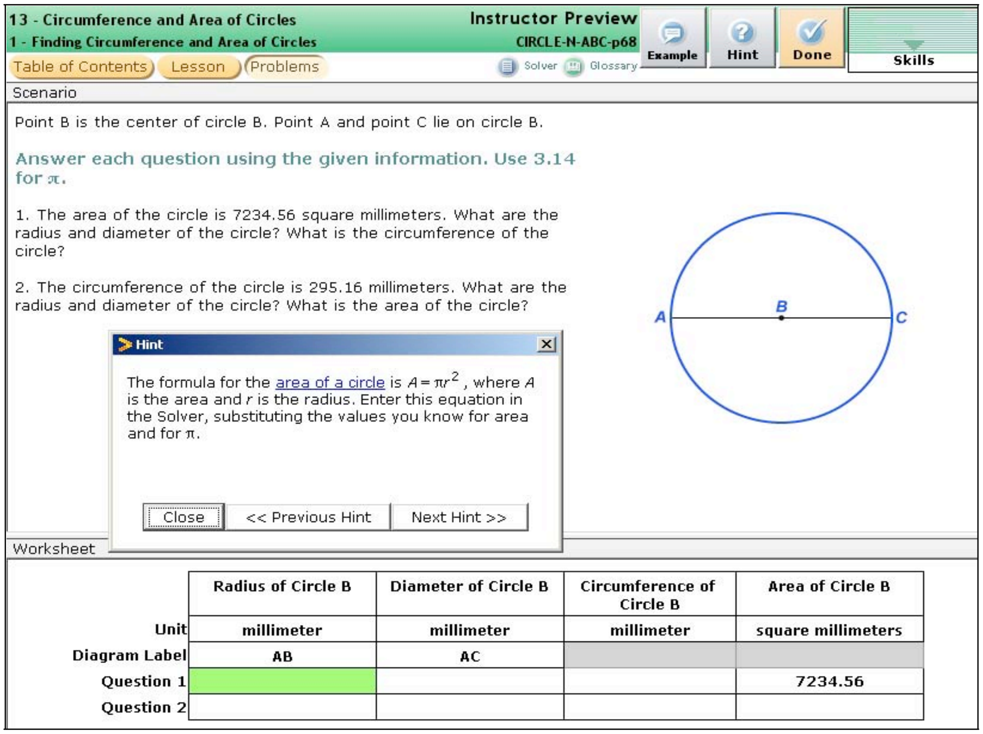
KC models in DataShop are fit to data using the AFM equation in Figure 1, and metrics for AIC, BIC, and three versions of cross validation are provided to evaluate and compare different models.

We illustrate the modification of a KC model to produce an improved model with implications for tutor redesign. The example data comes from a data set called Geometry9697 and can be found in the DataShop repository under Public Datasets. Figure 2 shows a screen shot of (a more recent version of) the tutor used in generating the data.

In this example, the best hand-generated model divides the ALT:COMPOSE-BY-ADDITION KC of the original in-use model into three KCs: *Subtract*, *compose-by-addition*, and *decompose*. The original ALT:COMPOSE-BY-ADDITION KC labels steps where the student must find the area of an irregular shape that may be the sum or difference of two regular shapes (e.g., what's left when a circle is cut from a square). This KC was targeted for improvement because, as shown in the top of Figure 3, it was found to have a non-smooth learning curve (a large difference between actual and predicted values) and although it is relatively difficult (26% error rate), the learning curve does not indicate any learning (the error rate does not go down with opportunities) and, correspondingly, the AFM slope estimate (γj) is zero. As described in [15], these features of a learning curve (not smooth, not low, and not declining) are indicators of a poorly defined KC. A KC may be improved by investigating the problem steps it labels, usually focusing on those where the error rate is much higher (or lower) than normal as in opportunities 12, 15-18, etc., shown in the curve at the top of Figure 3. The analyst seeks a feature of these problem steps that may change the difficulty of performing or learning that step (i.e., a difficulty or learning factor) that is not shared by the other problem steps. In this case, almost all of the hardest problem steps required students to identify the two regular shapes that make up a target irregular shape (i.e., to visually "decompose" and set subgoals to find the area of these regular shapes first). These problem steps were relabeled with a KC called *decompose*. In other problems, the ALT:COMPOSE-BY-ADDITION steps came after an explicit scaffold given to students to find the area of the regular shapes (e.g., a prompt to find the square and circle areas before finding the leftover). These remaining problem steps were relabeled *compose-by-addition*. We say that ALT:COMPOSE-BY-ADDITION was "split" by the decompose factor to produce a new "decompose" KC and a modified "compose-by-addition" KC with fewer steps associated with it.

By inspecting a subset of particularly easy problem steps, another factor was identified (repeated steps in the same problem) and these steps were labeled *Subtract*. That is, ALT:COMPOSE-BY-ADDITION was further split into a third set of steps. In sum, the new KC model splits the ALT:COMPOSE-BY-ADDITION KC in the Original model (which labels 20 steps) into three different KCs: *compose-by-addition* (6 steps), *decompose* (8 steps), and *Subtract* (6 steps).

The bottom of Figure 3 shows, for all three new KCs, the resulting learning curves and the parameter estimates for the difficulty intercepts (in both logit terms, β_k, and converted to a probability) and for the KC learning slopes (γ_k). Inspecting the empirical learning curves (red lines), all three look smoother than the original ALT:COMPOSE-BY-ADDITION, thus meeting the



Scenario

Point B is the center of circle B. Point A and point C lie on circle B.

Answer each question using the given information. Use 3.14 for π.

1. The area of the circle is 7234.56 square millimeters. What are the radius and diameter of the circle? What is the circumference of the circle?

2. The circumference of the circle is 295.16 millimeters. What are the radius and diameter of the circle? What is the area of the circle?

Hint

The formula for the area of a circle is $A = \pi r^2$, where A is the area and r is the radius. Enter this equation in the Solver, substituting the values you know for area and for π.

Close | << Previous Hint | Next Hint >>

Worksheet

	Radius of Circle B	Diameter of Circle B	Circumference of Circle B	Area of Circle B
Unit	millimeter	millimeter	millimeter	square millimeters
Diagram Label	AB	AC		
Question 1				7234.56
Question 2				

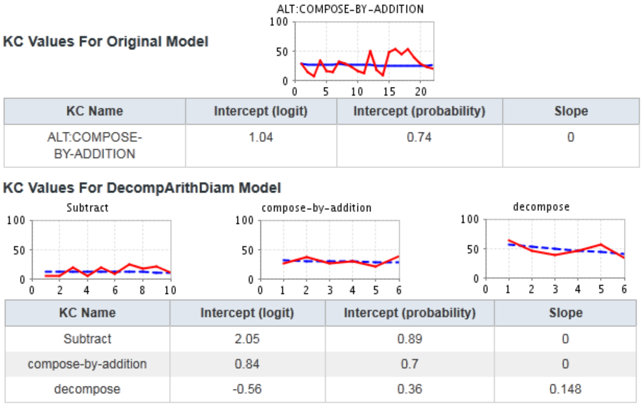
Figure 2. A screen shot of the problem Circle N present in the area unit of the Geometry Cognitive Tutor. In Question 1, students are given the area of the circle and must find the radius, diameter, and circumference. In Question 2, students are given the circumference and must find the radius, diameter, and area. The Hint message is in response to a student hint request for the highlighted cell.

“smoothness” criteria. The *decompose* KC learning curve appears to be declining in error rate and, correspondingly, the slope parameter is greater than zero indicating an improvement in the model. The *Subtract* KC curve is not declining but it is already at a low error rate from the start, so this also indicates an improvement in the model. Such a pattern indicates the KC is already known and mastered, therefore, little to no learning is expected. The story for the new *compose-by-addition* KC is mixed – while smoother, it is not declining and not already low. It might be possible to make further improvements in this KC.

The decompose KC results are quite different from the other two newly labeled KCs and the original KC (ALT:COMPOSE-BY-ADDITION) with a much higher initial error rate (57%) and a declining curve (intercept = .36, slope = .15). Given these results and the lack of mastery on the *decompose* KC after six opportunities, we recommend a higher concentration of decomposition problems with additional instructional aids such as worked examples, specific hints, and problems that isolate practice on this skill [15]. These results also indicate less practice is needed on the *Subtract* KC and corresponding problem steps could be reduced or even eliminated from the curriculum.

The statistical fit (shown in the blue lines in Figure 3) is based on the Additive Factors Model described above. Models are evaluated using AIC, BIC, and 10-fold cross validation. We report the root mean-square error (RMSE) averaged over the ten test sets in the cross validation.

For this dataset, the best models according to BIC and cross validation are ones that incorporate the distinction between unscaffolded (decompose) and scaffolded problem steps. More complexity, (e.g., models with 12 and 13 KCs) pays off relative to a simpler model with 10 KCs. But more complexity is not always better. The Original production rule model in the tutor had 15 skills, yet according to BIC and cross validation measures the simpler models with 12 and 13 skills are better predictors.



KC Values For Original Model

KC Name	Intercept (logit)	Intercept (probability)	Slope
ALT:COMPOSE-BY-ADDITION	1.04	0.74	0

KC Values For DecompArithDiam Model

KC Name	Intercept (logit)	Intercept (probability)	Slope
Subtract	2.05	0.89	0
compose-by-addition	0.84	0.7	0
decompose	-0.56	0.36	0.148

Figure 3. A knowledge component (KC) with a non-smooth learning curve (see top half of the figure) is replaced in an improved student model with three new KCs with smoother curves (see bottom half of the figure).

There are a couple instructional consequences of the fact that the decompose skill was confounded with performing scaffolded decomposition and, even, with simple subtraction. First, students were able to give the appearance of mastery because they were essentially given credit for this more complex skill when they successfully performed the two simpler skills. It is possible for students to graduate on the merged skill by only getting scaffolded decomposition and subtraction steps correct and never or rarely getting an unscaffolded decompose step correct (the tutor’s knowledge tracing algorithm allows for an occasional slip).

Second, because there was no differentiation of these skills, there was no way to provide any isolated or extra practice on the tough unscaffolded decompose skill. Thus, for both of these reasons, there was not enough practice of decompose.

We have illustrated how new KC models can be produced and evaluated within DataShop. This process has produced many different models across a variety of datasets. For instance, the Geometry9697 dataset had at least 10 KC models associated with it. These models range in number of KCs from 1 to 15 with a median of 12 KCs. The KC models vary on factors including the shape of a figure, the formula involved, whether a formula is applied forward or backward, whether or not this step repeats an analogous one in the same problem, etc.

We next present the use of the LFA algorithm to perform an automated search for better models. A key trick is to use, in a crowd-sourcing fashion, existing models entered by DataShop users as the basis for input to this search algorithm

3. APPLYING LEARNING FACTORS ANALYSIS (LFA) ACROSS DOMAINS

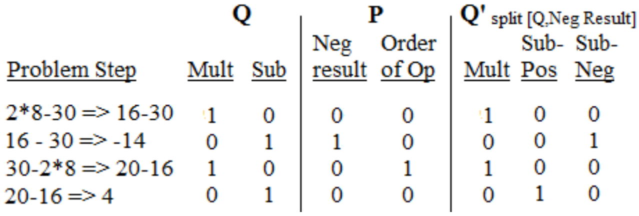
3.1 Adapting LFA for use in DataShop

Learning Factors Analysis (LFA) [1] is an algorithm that automatically finds better student models by searching through a space of KC models, represented as Q matrices, to find the one that best predicts student-learning data. The input to LFA includes a dataset of records that indicate a student, a step identifier (i.e., part of a problem or activity for which there is an observed and gradable student action), the order in which each student experiences each step, and whether the student was successful or not on the step (usually whether the students first action on a step is correct, that is, neither a hint request nor an incorrect action). The LFA input also includes a matrix, in the same form as the Q matrix described above, that indicates for each unique step (the rows) what candidate features or factors may affect student performance and learning on that step (the columns). This so-called P matrix is used, along with a set of operators, to determine the space of possible Q matrices that LFA searches over.

The output of LFA is a list of Q matrices (KC models) rank ordered (using either AIC or BIC) by how well they predict the student data. In the search process, new Q matrices are created from the current Q matrix by applying operators (split, merge, or add in the complete LFA) using a factor in the P matrix to modify some aspect of the current Q matrix.

Figure 4 provides a simple example of the search process beginning with the mapping of problem steps to Q and P matrices. In this example, a Q matrix with factors for multiplication (Mult) and subtraction (Sub) is modified by applying a split operator to a column in the Q matrix (Sub) using a column in the P matrix (Neg-result). The outcome is a new Q matrix (called Q' in Figure 4) that has the steps of Sub partitioned into two subsets (Sub-Pos and Sub-Neg) according to the values of Neg-result.

An important challenge in a broad application of LFA is determining how the P matrix gets created. In early applications of LFA, a human user (a single domain expert) created the P matrix. In the spirit of crowd sourcing, we have used the multiple hand-created KC models in DataShop as an alternative way to create the P matrix. The P matrix is automatically derived by combining all of the hypothesized KCs in the pre-existing KC models for that dataset. More specifically, the P matrix maintains the same rows (one for every step in every problem), but the columns (the hypothesized factors or KCs) are the union of all the columns in the pre-existing KC models (such that the number of columns is the sum of the number of columns across the existing models minus any duplicate columns).



	Q		P		Q' split [Q,Neg Result]		
Problem Step	Mult	Sub	Neg result	Order of Op	Mult	Sub-Pos	Sub-Neg
2*8-30 => 16-30	1	0	0	0	1	0	0
16 - 30 => -14	0	1	1	0	0	0	1
30-2*8 => 20-16	1	0	0	1	1	0	0
20-16 => 4	0	1	0	0	0	1	0

Figure 4. Example of a Q matrix and P matrix mapped to problem steps and the resulting Q' matrix when Sub in the Q matrix is "split" by Neg-result from the P matrix.

LFA implements a best-first search. In each iteration of the search, the best Q matrix so far (as measured by AIC or BIC) is chosen for expansion. The LFA search process begins with the simplest possible Q-matrix where all steps involve one and only one KC. New candidate Q matrices (KC models) are generated by applying operators to the columns in the P matrix and the selected Q matrix. In the version used here, we only used the split operator: LFA creates new candidate Q matrices by splitting KCs in the current Q matrix using relevant factors in the P matrix. The search stops either after a pre-set number of iterations or when model improvements diminish (see below).

The search process results in the creation of machine-generated KC models that are usually more predictive (as measured by cross validation) than any of the starting models. Because the current implementation is only using LFA's split operator, which only generates single KC codes for each problem step, it is possible a starting model that includes multiple KCs per step could be better than any LFA model generated. Although cross validation is arguably the best way to test the predictive efficacy of a model, it is too computationally expensive to run inside the LFA search. Instead we have used BIC and, more recently, AIC as the heuristic to guide the search. After the search is complete, we test the best models using cross validation.

3.2 Method: Apply LFA across 11 datasets

In order to examine the LFA search process across a variety of datasets, we used eleven datasets representing five domains (geometry, algebra, fractions, English articles, and statistics) from the DataShop repository. Each dataset had from 1-16 KC models previously created by content specialists or researchers and most (65%) of these models coded a single KC per step. The number of knowledge components within models ranged from 1-48 and the number of student users ranged from 41-318. In addition to a variety of dataset characteristics, we have a wide group of researchers/authors represented: Lovett (statistics), Wiley (English articles), Booth (equation solving), Lomas (fractions), Koedinger (symbolization), Stamper, Ritter and Koedinger (geometry area).

To each dataset, we applied a version of LFA that: 1) used only the split operator, 2) started with a Q matrix with a single KC labeling all problem steps, and 3) started with a P matrix made up of the union of all existing KC models. We ran the search process twice on each dataset, once using BIC as the search heuristic and once using AIC. The search continues until a streak of 5 iterations does not produce a model with an improved heuristic value. We compared the two best models from each of the AIC and BIC runs (4 models total) with all the existing models using root mean square error (RMSE) as determined by 10-fold cross validation. Three different cross validations were run: 1) student stratified (SSCV), 2) item stratified (ISCV), and 3) no stratification (NSCV). For student stratified and item stratified cross validation, students or items (i.e., problem steps), respectively, were randomly chosen for the folds. No stratification cross validation selects the 10 folds randomly from the dataset as a whole, irrespective of student or item. We chose item stratified cross validation as the primary metric (the one used in Tables), because we are concerned with improving tutors and item stratified cross validation corresponds most closely with a key tutor decision of what next problem to select.

3.3 Results: Better Models Found

Table 1 summarizes the results. Analysis of the datasets using RMSE from a 10-fold item stratified cross validation (ISCV) shows a machine-generated model is the best predictor of student performance across all eleven datasets and all four domains. This can be seen in Table 1 by noting that the RMSE values in the Best-LFA column (representing the results of machine-generated models) are all lower than those in the columns for the best model found by hand (Best-hand) and for the original model in use by the tutoring system or game (Orig-in-use).

The results from both SSCV and NSCV were mostly similar in that in 10 of 11 datasets, an LFA model was best. For the DFA-318 dataset, the Best-hand model was better on SSCV and NSCV (but not on ISCV). That model involved some steps being coded by more than one KC – such multi-KC coding is not a model that the LFA version we used (which only incorporates the split operator) can produce, however, a version with LFA's add operator could, in principle, produce such a model.

Besides reducing prediction error, the LFA-discovered models tend to better capture student learning in that the typical slopes on learning curves are steeper. The last columns in Table 1 show a comparison of the median learning slope (γ_k) for the Best-hand and Best-LFA models. In all but a couple of cases, the median learning slope is larger for the Best-LFA model than it is for the Best-hand model (e.g., 0.11 vs. 0.07 for Geometry9697).

4. INTERPRETING STUDENT MODEL IMPROVEMENTS

We have shown how an automated process, such as an LFA search, discovers new and improved student models. But, are these improvements substantial and interpretable? Can they be used to propose plausible improvements in a tutoring system?

One method to evaluate newly discovered machine models is to investigate when and how much each KC in the original model is changed in the transformations that produced the best-hand and best-machine models. By isolating improvement in knowledge components, areas of student difficulty can be uncovered and automated systems can be redesigned to more efficiently address student learning by focusing better instruction and more practice on more difficult skills and less practice on easier skills.

We use the Geometry9697 dataset as an example of a proposed strategy for interpreting LFA results toward model and tutor improvement. That strategy starts with inspecting the impact of model improvements on specific aspects of the original model. A key observation is that while the overall prediction error (RMSE) reductions in Table 1 may seem small, LFA is likely to make significant changes in the KC model only in a few isolated places, that is, only for some of the original KCs. Although those changes may be practically significant, they are obscured in the overall RMSE change given much of the model remains the same.

Thus, we suggest trying to identify which of the KCs in a base model are most substantially changed in the creation of a new model. One way to do so is to compute the reduction in RMSE between models for each of the KCs in the base model.

Table 1. The root mean square error (RMSE) for the best KC models as determined by item stratified cross validation.

Dataset	Content area	RMSE			Median Learning slope (logit)	
		Orig in-use	Best-hand	Best-LFA	Best-hand	Best-LFA
Geometry9697	Geometry area	0.4129	0.4033	0.4011	0.07	0.11
Hampton 0506	Geometry area	NA	0.4022	0.4012	0.03	0.04
Cog Discovery	Geometry area	NA	0.3250	0.3244	0.16	0.16
DFA-318	Story problems	0.4461	0.4407	0.4405	0.07	0.17
DFA-318-main	Story problems	0.4376	0.4287	0.4266	0.09	0.17
Digital game	Fractions	0.4442	0.4396	0.4346	0.17	0.14
Self-explanation	Equation solving	NA	0.4014	0.3927	0.01	0.04
IWT 1	English articles	0.4262	0.4110	0.4068	0.10	0.12
IWT 2	English articles	0.3854	0.3854*	0.3806	0.12	0.16
IWT 3	English articles	0.3970	0.3965	0.3903	0.05	0.15
Statistics-Fall09	Statistics	0.3648	0.3527	0.3353	**	0.09

NA: Original models (or statistics on them) were not available in some cases.
* IWT 2 dataset only has 1 model, therefore the original-in-use and best-hand models have the same RMSE.
**The best-hand model for Statistics-Fall09 dataset has only 1 KC.

More specifically, for each data point labeled by a base model KC, we find the RMSE based on the predictions of the base model and compare it with the RMSE based on the predictions of the new model. We compute the percent reduction in RMSE ((base - new)/base).

Table 2 summarizes the results of this analysis on the Geometry9697 dataset by comparing each pair of the Original, best hand, and best LFA models in terms of the Original model KCs. As anticipated, the last row shows that for most of the Original KCs (12 of 15), there is little to no improvement in the best hand and best LFA models. However, for three of the KCs, there are large reductions in prediction error. The TRIANGE-SIDE KC has the largest RMSE reduction from Original to LFA model (11.1%); however, the decrease is mostly caught in the improvements made to the best hand-model from the Original model (10.0%). The COMPOSE-BY-ADDITION KC is also improved (as described in section 2) just in the original to hand-model transition. The CIRCLE-RADIUS KC, on the other hand, realizes an almost 6% reduction of RMSE from the Original model to the best hand-model, and then another sizeable reduction of almost 4% from best-hand model to the best LFA model. This discovery of LFA represents a genuine machine-based discovery not directly anticipated by human analysts.

Table 2. Improvement in knowledge components in the Geometry9697 dataset measured by the percent reduction of root mean squared error (RMSE) from item-stratified cross validation.

Original model KCs	% reduction in RMSE orig->hand	 hand->LFA	 orig-LFA
CIRCLE-RADIUS	5.8%	4.0%	9.5%
COMPOSE-BY-ADDITION	5.2%	0.3%	5.5%
TRIANGLE-SIDE	10.0%	1.2%	11.1%
Range of the 12 other KCs	-.5 to 3.4%	-.3 to 1.0%	-.2 to 3.1%

A closer look at the CIRCLE-RADIUS KC from the Original shows it is coded as three separate KCs in the best hand-model: (1) circle area, (2) circle-diam-from-given, and (3) circle-diam-from-subgoal. In all three of these KCs, computing a radius is the target skill but how it is computed depends on what component measure is provided. For example, in the Circle-N problem (see Figure 2), area is given in the first row of the table (and in the text as Question 1) and students must compute the remaining values including radius; this is labeled as the circle-area KC. In the circle-diam-from-subgoal KC, circumference is given (row 2 or Q2) and in the circle-diam-from-given KC, diameter is given. As can be seen in Figure 5, the LFA model further changes these same KCs by either combining (e.g., circle-diam-from-given combines with three other KCs to form Geometry) or splitting (e.g, circle-area splits to form a reduced version of itself and a new KC called radius-from-area). Circle-diam-from-subgoal remains the same from best hand to best machine. We use the circle-area split as an illustration of how the machine-model uncovered a useful improvement in the original model that was not anticipated by humans generating hand models.

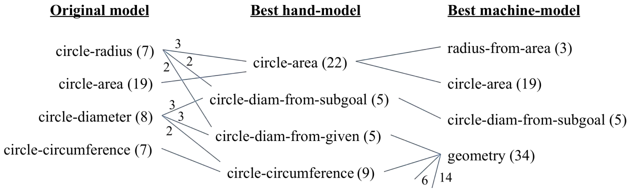


Figure 5. The splitting and combining of circle-radius and other related hypothesized knowledge components in going from the original-model to the best hand model to the best machine model.

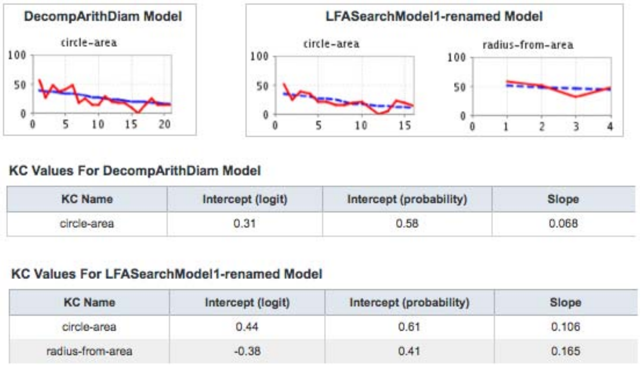
In Figure 5, we see the circle-area KC in the best hand-model has 22 problem steps but after the LFA algorithm is applied this one KC is divided into two KCs – one with 19 problem steps and the other with 3 problem steps. What is unique about these three problem steps that they split from the original 22 to form a "new" KC?

A careful examination of the three problem steps in the new radius-from-area KC reveals a backward strategy is necessary for a correct solution (e.g., finding radius when area of circle is given) unlike the other nineteen problem steps, which require a forward strategy. Although using a backward strategy is not uncommon in the dataset (about 27% of the problem steps require it), none of the other KCs were split between backward and forward by the search algorithm. In fact, eight of thirteen KCs in the best hand-model label backward steps but only circle-area benefits from being split into forward and backward versions.

Figure 6 is analogous to Figure 3 and shows the effect of the LFA discovery to split circle-area in the best hand-model (called DecompArithDiam) into circle-area (labeling fewer steps) and radius-from-area. Both resulting learning curves are smoother than the original and they have steeper slopes (.105 and .165, respectively, as compared to .068).

Performance on the circle-area backward problems (called radius-from-area) is lower (54%) than performance on the circle-area forward problems (80%), which is perhaps not a surprise. However, in this context, it is surprising that the difference in backward vs. forward performance on the other KCs is small and statistically negligible. For example, pentagon area does not split from apothem or side (backward = 66%, forward = 62%), parallelogram area does not split from parallelogram side (backward = 89%, forward = 91%), trapezoid area does not split from base or height (backward = 54%, forward = 55%), and triangle area does not split from base or height (backward = 68%, forward = 78%).

In addition to not finding a backward split for any of the area formulas for other shapes, there was no forward-backward split for other circle formulas (circumference = pi*diameter and diameter = 2*radius). A unique feature of working the circle area formula backwards is that it requires a square root operation to find the radius (after dividing the given area by pi). Thus, the uncovered knowledge component appears to be about learning when and how to employ a square root operation rather than about a general ability to apply a backward strategy, for instance, by using algebra (as suggested in the hint in Figure 2). Note, the need to employ a square root is also required in backward application of the square area formula, however, this tutor unit did not have any such problems where only the area of a square is given and the side must be found.



KC Values For DecompArithDiam Model

KC Name	Intercept (logit)	Intercept (probability)	Slope
circle-area	0.31	0.58	0.068

KC Values For LFASearchModel1-renamed Model

KC Name	Intercept (logit)	Intercept (probability)	Slope
circle-area	0.44	0.61	0.106
radius-from-area	-0.38	0.41	0.165

Figure 6. Learning curves and model values of circle area knowledge component in best hand-model and after splitting to circle area and radius-from-area KCs in best LFA model.

The consequences for tutor changes based on this automated discovery of LFA are significant. We recommend the tutor distinguish area-to-radius problems while merging all other forward/backward area combinations (e.g., pentagon side and pentagon area). The current unit in the Geometry Tutor only has three problem steps associated with the area-to-radius KC, not enough to lead to mastery on a difficult skill. Thus, more such problems should be created. The skill tracking, which is done with skill bars in the tutor, should be modified to maintain the forward-backward distinction for the circle area formula, but merge the forward-backward distinctions for other formulas (including the other circle formulas for circumference-diameter and the diameter-radius relationships). This change should substantially reduce the time students currently take to separately (and unnecessarily) demonstrate mastery of both the forward and backward versions of each of the area formulas. This time could in turn be used to make sure that students get enough practice in cases where their prior algebra background appears insufficient for backward application of formulas, namely in use of the square root operation. Further, given the need for more square root practice, square-area-to-side problems should also be added.

5. CONCLUSION AND FUTURE WORK

This paper demonstrates an automated technique for the discovery of better student models using input from previously generated models. LFA discovered better student models in 11 of 11 cases as measured by item stratified cross validation and 10 of 11 cases by student stratified and non-stratified cross validation.

Although the reduction in overall error (RMSE) appears rather small, we demonstrated how this small error reduction is a consequence of most of the discovered model being essentially the same as the original. There are a few isolated changes that are statistically large and, more importantly, practically significant for tutor redesign.

A related, more nuanced point is that even in cases where there is no significant improvement in prediction error, the new model may be better in parsimony (i.e., by reducing the number of KCs). Parsimony improvements in student models are important scientifically because they simplify explanations and suggest broader transfer of learning. They are also practically important in that tutors with a more compact student model will save and focus student time.

Student models are critical to effective adaptive instruction. Different kinds of student model changes (e.g, original KCs splitting or not splitting) suggest specific tutor redesigns. A number of instructional design changes are suggested when an original KC is split into one or more new KCs. First, the skill bars and knowledge tracing need to be changed to include the new KCs and to be sure that students master all of them. Second, often a newly discovered KC will occur in too few of the existing problems and thus new problems need to be created. Third, the distinction discovered in the KC split (e.g., use of square root) may be better highlighted in the various forms of instruction that tutor may employ including worked examples, error feedback messages, and next-step hint messages. Similarly, a different set of instructional design changes are suggested when a number of original KCs are merged (i.e., are not split) into a single KC. First, the skill bars and knowledge-tracing model need to be changed to eliminate unnecessary distinctions. With corresponding knowledge tracing parameter changes, the number of practice problems a student needs to master the merged KC will be significantly reduced relative to the greater number previously needed to master each of the separate KCs. Second, the commonality discovered in the KC merge (e.g., the learning obstacle for most area operations is learning to retrieve and map the right formula, not learning how to apply it in a forward or backward direction) can be highlighted in the various forms of instruction that tutor employs. Doing so may better encourage the desired generalization and greater transfer of learning.

The general LFA algorithm includes an “add” operation that can produce multiple KC codes for a single problem step. For efficiency reasons (for larger datasets, LFA ran for multiple days) and to simplify interpretation of the results, we did not use the add operator in the version of LFA employed here. However, in the case that a new KC is added, there are instructional implications beyond those indicated above for split operations. Namely, there is the possibility of inventing new tasks that isolate a KC that might only occur with other KCs in the current set of tasks. Doing so has been demonstrated to yield significant improvements in student learning [7]. Future work should explore the use of the add operator in the LFA algorithm and, more generally, needed algorithm improvements to increase efficiency.

Some other automated techniques discover models that are difficult or impossible to understand, either toward deriving insights into student learning or making practical improvements in instruction. The output of LFA is more interpretable and convertible to tutor changes than these alternative “black box” machine learning methods that may produce Q matrices (or other latent variable representations) without consistent application of analyst-derived codes or without code labels at all. Even so, the output of LFA is complex and not trivial to interpret. Thus, we recommend a strategy (illustrated in Table 2) for isolating the practically significant student model improvements that LFA discovers. Such improvements in the student model have direct implications for many aspects of tutor design including problem development, knowledge tracing, problem selection and sequencing, skill bar display, instructional hint and hint message content. Using data to automatically improve student models and, in turn, improve instructional systems is a tremendous opportunity for educational data mining, especially as we accumulate large datasets and relevant techniques in repositories like DataShop.

A notable innovation here is a simple form of crowd sourcing of the work done by data analysts using DataShop. A significant limitation of LFA recognized in prior work [1] is the need for the P matrix input to the algorithm, that is, the human coding needed to produce the various difficulty and learning factors on problem steps that are the basis for hypothesizing knowledge components. This limitation is addressed here by taking advantage of the DataShop facilities for creating and storing hand-built KC models and the fact that analysts have been using those facilities to create models.

Our simple version of combining human and machine intelligence toward better model discovery could be applied more broadly beyond DataShop data or student modeling. The general idea involves a web-site with these components: 1) users can modify an existing model (e.g., add new features based on feature engineering) and see whether their new model yields better predictions (e.g., DataShop's KC model leaderboard display), 2) a machine algorithm that collects features across all human-entered models (e.g., a simple union of all features), and 3) a machine algorithm that searches over the space of features to identify more predictive models (e.g., the LFA algorithm). We leave it for future research to test this proposed generalization of the approach presented here.

ACKNOWLEDGMENTS

Our thanks to Hui Cheng for running the search algorithm, to DataShop (http://pslcdatashop.org) for dataset access, and to the National Science Foundation (#SBE-0354420) for funding of the Pittsburgh Science of Learning Center.

6. REFERENCES

[1] Cen, H., Koedinger, K. R., & Junker, B. (2006). Learning Factors Analysis: A general method for cognitive model evaluation and improvement. In M. Ikeda, K.D. Ashley, T.-W. Chan (Eds.) *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, 164-175. Berlin: Springer-Verlag.

[2] Clark, R.E., Feldon, D., van Merriënboer, J., Yates, K., & Early, S. (2008). Cognitive task analysis. In Spector, J. M., Merrill, M.D., van Merriënboer, J., & Driscoll, M.P. (Eds.) *Handbook of research on educational communications and technology* (3rd ed.). Mahwah: Lawrence Erlbaum.

[3] Draney, K.L., Pirolli, P., & Wilson, M. (1995). A measurement model for complex cognitive skill. In P. Nichols, S.F. Chipman, & R.L. Brennan (Eds.), *Cognitively diagnostic assessment* (pp. 103–126). Hillsdale: Erlbaum.

[4] Falmange, J.C., Koppen, M., Villano, M., Doignon, J.P. & Johannesen, L. (1990). Introduction to knowledge spaces: How to build, test, and search them. *Psychological Review*, 97, 201-224.

[5] Feng, M., Heffernan, N.T., & Koedinger, K.R. (2009). Addressing the assessment challenge in an online system that tutors as it assesses. User Modeling and User-Adapted Interaction: *The Journal of Personalization Research (UMUAI). 19*(3), pp. 243-266.

[6] Koedinger, K. R. & Aleven, V. (2007). Exploring the assistance dilemma in experiments with Cognitive Tutors. *Educational Psychology Review, 19* (3): 239-264.

[7] Koedinger, K.R. & McLaughlin, E.A. (2010). Seeing language learning inside the math: Cognitive analysis yields transfer. In S. Ohlsson & R. Catrambone (Eds.), *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*. (pp. 471-476.) Austin, TX: Cognitive Science Society.

[8] Koedinger, K.R., Baker, R.S.J.d., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J., (2011) A Data Repository for the EDM community: The PSLC DataShop. In Romero, C., Ventura, S., Pechenizkiy, M., Baker, R.S.J.d. (Eds.) *Handbook of Educational Data Mining*. Boca Raton, FL: CRC Press.

[9] Lee, R. L. (2003). Cognitive task analysis: A meta-analysis of comparative studies. Unpublished doctoral dissertation, University of Southern California, Los Angeles, California.

[10] Martin, B., & Mitrovic, A. (2004). Evaluating Intelligent Education Systems with Learning Curves in *Proceedings of the workshop on evaluation at AH2004*, Eindhoven, pp. 179-188.

[11] Martin, B., Mitrovic, T., Mathan, S., & Koedinger, K.R. (2011). Evaluating and improving adaptive educational systems with learning curves. *User Modeling and User-Adapted Interaction* (2011) 21:249–283.

[12] Model Values. In PSLC DataShop. Retrieved 4/10/12 from https://pslcdatashop.web.cmu.edu/help?page=modelValues

[13] Pavlik Jr., P.I., Cen, H., Koedinger, K.R.: Learning Factors Transfer Analysis: Using Learning Curve Analysis to Automatically Generate Domain Models. In: Barnes, T., Desmarais, M., Romero, C., Ventura, S. (eds.) *Proceedings of the the 2nd International Conference on Educational Data Mining*, Cordoba, Spain, pp. 121-130 (2009).

[14] Spada, H., & McGaw, B. (1985). The assessment of learning effects with linear logistic test models. In S. E. Embretson (Ed.), *Test design: Developments in Psychology and Psychometrics*.

[15] Stamper, J. & Koedinger, K.R. (2011). Human-machine student model discovery and improvement using data. In J. Kay, S. Bull & G. Biswas (Eds.), *Proceedings of the 15th International Conference on Artificial Intelligence in Education*, pp. 353-360. Berlin: Springer.

[16] Tatsuoka, K.K. (1983) Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of Educational Measurement*, 20, 345-354.