Refining Learning Maps with Data Fitting Techniques: What Factors Matter?

X, X, X, X

X

Abstract. Cognitive models/Learning maps (skill graphs) have been identified to be possible to improve using some data mining techniques. However the factors that affect this improvements/refining process are not so clear. In an earlier paper we presented a method for improving these cognitive models. The purpose of this paper is to present the factors to consider when using our initial algorithm to refine learning maps. We present a simulation study that shows how important each of the factors is for this refinement process.

Keywords: Data Mining, LFA, Learning Maps, Cognitive Models

1 Introduction

Learning maps have been used as a tool to depict the set of skills in a cognitive domain and the relationship between these skills. A number of studies have been conducted to find and represent the relationships between the skills [4], [7], [9-10]. Tatsuoka introduced the Rule-space method for identifying skills/knowledge components in a given cognitive domain whereas [6] present another approach called the Attribute Hierarchy Method (AHM). The rule-space method (RSM) does not present the relationship of the skills/knowledge components as a hierarchy. However AHM, which is a variation of Tatsuokaôs RSM, considers the hierarchical relationship between the components [11]. Gierl [5] used the AHM approach to make inferences of studentsô cognitive assessment. None of these approaches dealt with methods for improving the item response models developed using the methods proposed. The Learning Factors Analysis (LFA) method by [2] was introduced to deal with this problem. In that paper three different operations for improving the predictive abilities of learning maps or cognitive models are introduced. In [1] an attempt was made to solve this problem by presenting the results of a number of experiments that showed that learning maps can be refined using just one (the merge operation) of the three possible of the LFA method. It was shown that there were significant improvements in RMSE for the best model chosen, starting off with a pre-defined learning map.

We realize that, to generalize the method for refining learning maps, there are a number of questions that still need to be answered. These include: ñWhat are the factors that can determine when a model can be best refined?ò and ñDo the number of skills, the number of items per skills, the number of levels in the skill hierarchy and

the number of data points have any effect in determining the best refined model?ò Whilst the LFA methods use a set of factors to determine whether to merge, add or split skills to generate better models from an existing one, all the factors used are based on expert knowledge and are independent of data. In order to answer the above questions, we present a number of simulation experiments.

2 Problem Statement

The LFA model uses three operations (splits, merges and adds) to refine knowledge components. In each of the operations, learning factors were included in the model refinement process. These factors did not include the number of skills in the model, the levels in the hierarchy of skills in the model, the number of items per skill and the guess and slip parameter values for the items. We hypothesize that these factors are important in generating an optimal model from a given learning map (pre-requisite skill hierarchy). Hence we set out in this paper to present a series of experiments that help in determining the impact of the above mentioned factors in refining a given learning map or cognitive model.

3 Methodology

To be able to answer the research questions, we started off with a 3-skill graph. We inserted a fake skill at different locations of the graph and run our evaluation code to determine when the original skill-graph is learned back and what factors determine when this occurs. We examine the following factors and determine which of these factors have the most impact on using the greedy algorithm presented in the earlier paper to refine a given model: guess and slip parameter values, the number of levels in the skill graph hierarchy and the number of data points (i.e. students and items) . For each randomly chosen skill graph we generate a set of simulated data, one each for the number of student and item pairs used. We then evaluate the models using Expectation Maximization to determine the factors that have the most impact. The section presents the random graph generation, Bayesian network creation, fake skill creation and the evaluation code.

3.1 Random Skill Hierarchy Generation.

To generate a skill graph randomly we start by choosing a random skill hierarchy. Our algorithm to generate the skill hierarchy takes a range of skills and a graph depth as input parameters. The output of the algorithm is a valid skill hierarchy where the number of vertices is within the skill range and the number of levels is within the depth range. We order our vertices from 1 to N and use the constraint that a vertex cannot have a directed edge pointing to a smaller numbered vertex. We also enforce the constraint that a vertex cannot have any self-edges.

To generate a random graph we choose a random number within the range of possible graphs. We then convert this number to binary form and add the correct number of leading zeroôs (we know the number of skills from the random number chosen). Then we simply insert the bits of the binary number into the varying spots of the matrix form of the graph in order.

The result is a directed acyclic graph with no self-edges. It will not necessarily be connected. The final step is to check if the graph is connected. If the graph is connected, we keep it; otherwise we discard it and repeat the generation process. This method allows us to instantly generate valid graphs. An example is shown in Table 1 and Fig 1 for a graph with three skills.

Table 1. Example Matrix. Matrix generated by the random number 5. A óYô represents that this cell is ignored because it must be a zero since a vertex cannot have directed edges pointing to vertices with larger numbers. An óXô represents that this cell is ignored because it must be a zero since a vertex cannot have self-edges.

Vertex / Vertex	1	2	3
1	X (0)	1	0
2	Y (0)	X (0)	1
3	Y (0)	Y (0)	X (0)

Fig. 1. Example Graph Generated

3.2 Create Bayesian Network.

The Bayesian network used for the analysis was generated from the skill graph selected from the previous step. To generate the items for the skills an item range is specified. A random number of items are chosen within the item range for each skill. In our experiments we restricted our range to be a single value so all skills will have an equal number of items. We set our Bayesian network up like knowledge tracing, where every skill has one or more items and every item has a guess and slip node [3]. An item must belong to exactly one skill. The skill nodes are latent nodes since we cannot observe whether or not a student knows the skill. Each item node is an observable node, which is a ó1ô if the student answered the item correctly and a ó0ô if the student did not answer the item correctly. Both the guess and slip nodes are also latent

nodes representing whether or not the student guessed or slipped on the item. A student is considered to have guessed when the student answered correctly but did not know the skill. A student is considered to have slipped when the student answered incorrectly but knew the skill. Using the previous skill graph example we added the item, guess, and slip nodes to the graph.

The final step to create the Bayesian network is to create the conditional probability tables (CPT) for the nodes. For our experiments we defined each skill node as AND nodes. This means that a student can only know a post-requisite skill if the student knows all of the prerequisite skills. Therefore if a student does not know one of the prerequisite skills then the student cannot know the post-requisite skill. If the student does know all the prerequisite skills (or there are no prerequisite skills), we pick a random probability that the student will know the post-requisite skill between 0.3 ï 0.7. Our guess and slip parameters have varying probabilities since that was one of the parameters we experimented with. All the item nodes have a deterministic (0% chance or 100% chance of correctness) CPT based off of the skill, guess, and slip nodes (which or not deterministic).

3.3 Creation of Fake skill

We exported our Bayesian network to Matlab and used Kevin Murphyôs Bayes Net Toolkit to generate the ground truth data. Once the ground truth data was generated we randomly generated ñfakeò skills from the original graph. A fake skill is generated by randomly choosing a real skill. Once a real skill is chosen, a random number of items are chosen from the real skill. These items are then detached from the real skill and attached to the fake skill. The fake skill is then randomly chosen to be either a parent or a child of the real skill. Fig 2 shows the creation of a fake skill.

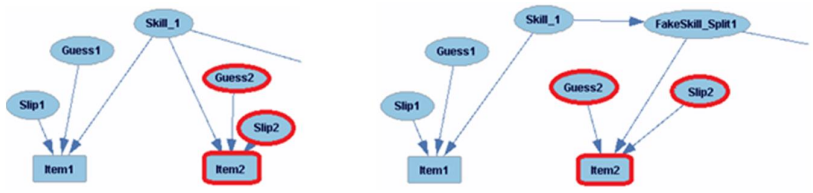

Fig. 2. Creation of Fake Skill. The left skill graph shows the original skill graph before the creation of the fake skill. The skill graph on the right shows the skill graph after the creation of the fake skill. The fake skill was created from Skill_1 where item 2 was removed from skill 1 and attached to the fake skill.

3.4 Evaluation

In order to evaluate our Bayesian Network we used a similar process as done in [1]. We use Expectation Maximization (EM) to learn parameters and fit our model. To

evaluate our model we used per student per item cross validation with 5 student folds and 3 item folds. Our student and item folds were chosen randomly for our evaluation. In [1] the item folds were chosen randomly but kept the same for each student. The only difference between the evaluation in and this experiment is that each student is assigned a different set of random item folds instead of all students having the same set of random item folds.

4 Experiments

4.1 Experiment 1

In this first experiment, we started with a set of 3-skill graphs. For each of the graphs, we insert a fake skill. We define a fake skill as one that is broken off of an existing skill. The fake skill has a random number of items chosen from the original skill and the fake skill is either a pre-requisite or post-requisite of the original skill. If the fake skill is a pre-requisite of the original skill, all the previous pre-requisites of the original skill become the pre-requisites of the new fake skill and the original skill becomes the post-requisite of the fake skill. The whole idea is to figure out if this fake skill will be easily identified and merged with the skill from which it was created from. This is to validate our merge operations and to determine what factors influence the determination of a better skill-model /skill map than the original.

4.2 Analysis

We analyzed the results of the experiment and looked at how the number of students, number of items, guess/slip values, and the number of fake skills impacted RMSE of our predictions and the percent of correct graphs learned back. Fig 3 shows the relationship between the probability a student guesses/slipped and the RMSE as well as the percent of the correct skill graph being learned back. We paired guess and slip values to lower the number of variables in our experiment. Our guess/slip pairings are as follows {(0, 0), (0.1, 0.08), (0.3, 0.16), (0.5, 0.25)}. It shows that the higher chance the student has to guess the answer the less accurate and harder it is to learn back the true original graph. The percent of graphs learned back with a guess/slip probability of 0 is significantly better than the percent of graphs learned back with a guess probability of .5 ($p < .001$). A realistic guess probability is around 0.14 calculated in [8]. At this point the percentage of graphs learned is somewhere between 0.25 and 0.33. These are not great percentages to learn back a correct graph under realistic guess and slip values. Not much can be done to lower the guess probability on typical questions middle school math students would see. However more student data can be used to increase model performance.

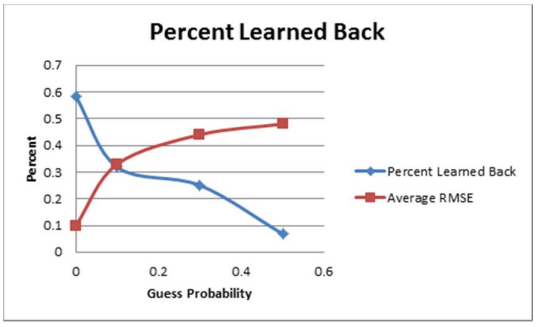

Fig. 3. Effect of guess/slip on learning back the original graph.

The guess/slip probability is the biggest factor that affects model accuracy followed by the number of students. Table 2 shows how both the guess/slip probability and the number of students affects the percentage of correct graphs learned back and average RMSE. A cell is broken up into two columns where the first column in the cell is the percentage of correct graphs learned back and the second column in the cell in the average RMSE value.

Table 2. Student/Guess Impact on Evaluation

	Students							
Guess	**50**		**100**		**150**		**200**	
	P_{LB}	**RMSE**	**P_{LB}**	**RMSE**	**P_{LB}**	**RMSE**	**P_{LB}**	**RMSE**
0	0.33	0.09	0.64	0.08	0.7	0.1	0.67	0.02
0.1	0.25	0.36	0.33	0.33	0.33	0.32	0.38	0.31
0.3	0.25	0.46	0.33	0.44	0.08	0.44	0.38	0.43
0.5	0.08	0.49	0.08	0.48	0.08	0.48	0	0.46

For a guess probability of 0.3, the percentage of correct graphs (P_{LB}) increases from 25% for 50 students to 38% for 200 students ($p = 0.2$). This shows that under a realistic worst case guess probability, increasing the number of students can increase the percentage of correct skill graphs learned back. The number of fake skills seems to have little effect on RMSE, however a large effect on learning back the correct graph. With more than one fake skill the percentage of correct graphs drops significantly from 0.6 for 1 fake skill to 0 for 3 fake skills ($p < .002$) for guess values of 0.1. This can be seen for the three points with an x-axis value of 0.1 in Fig 4. We excluded the number of items from our analysis since there were no strong trends.

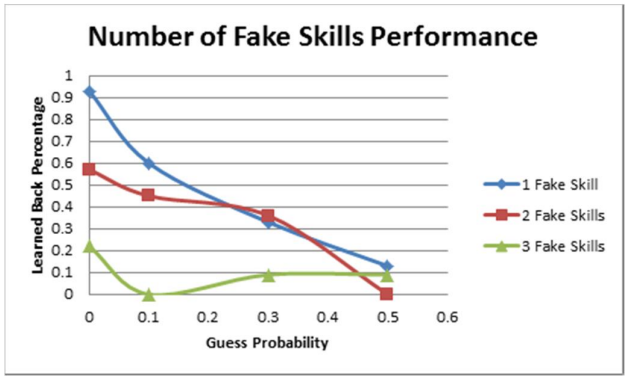

Fig. 4. Effect of Number of Fake Skills on model improvements

4.3 Experiment 2

In experiment 1 multiple randomly chosen graphs were used as the ground truth. In this experiment we chose to try each possible 3-skill graph to see if the graph structure had an effect on whether or not the correct skill graph was learned back. The methodology was the same as experiment 1 except instead of randomly choosing graphs we ran each of the four graphs for each possible number of students and items per skill. Fig 5 shows all four possible 3-skill graphs. After determining that the major factor impacting performance were guess/slip values, a reasonable pair of values were chosen for the guess and slip values (guess=0.1 and slip=0.08). Additionally we fixed the number of fake skills to one in order to reduce the variability of the factors.

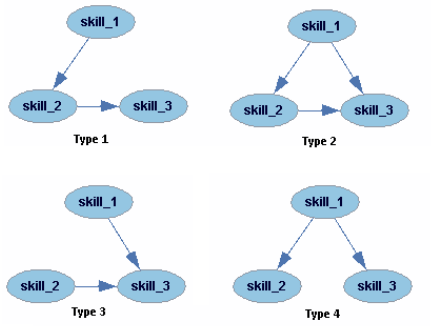

Fig. 5. Different Graph types for experiment 2

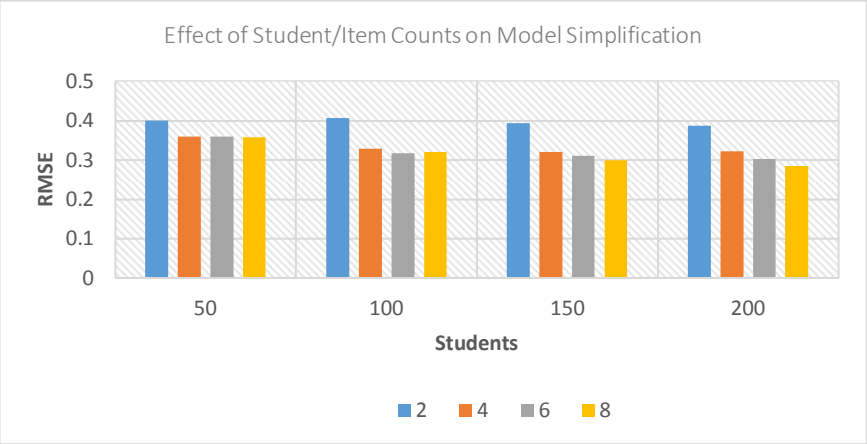

Fig. 6. Effect of students/items on the model simplification.

The general observation from this experiment is clear from Fig 6 above. As the number of data points increases, the level of accuracy in recovering the original graph increases. This is in spite of the fact that the location of the fake skill was not fixed. Moreover, for any given number of students, an increase in the number of items results in a slight decrease in RMSE and hence better chance of learning back the original graph. This experiment shows that the data points (i.e. student and item numbers) have an impact on improving on the determination of the best model from a given model.

4.4 Experiment 3

In this experiment we fixed all variables except for the number of students and the number of items per skill. We wanted to see how stable our search was and how well it performed for a small example with reasonable parameter values. We fixed guess at 0.10 and slip at 0.08 with three skills and one fake skill. For the fake skill we took the first half of items from the original skill. We ran our algorithm for 50, 100, 150, and 200 students for 2 and 8 items per skill. For each pair of parameters we ran the experiment 10 times with different random seeds and took an average of the number times the correct graph was learned back. Fig 7 shows the results of this experiment. We found that the results are very stable for graphs that had two items per skill. The results were less stable for graphs with eight items per skill although the percent of graphs learned back was much batter. The graphs that had two items per skill were learned back correctly 8% of the time, where graphs with eight items per skill were learned back correctly 43% of the time, which is a significant improvement ($n=40$, $p<.001$).

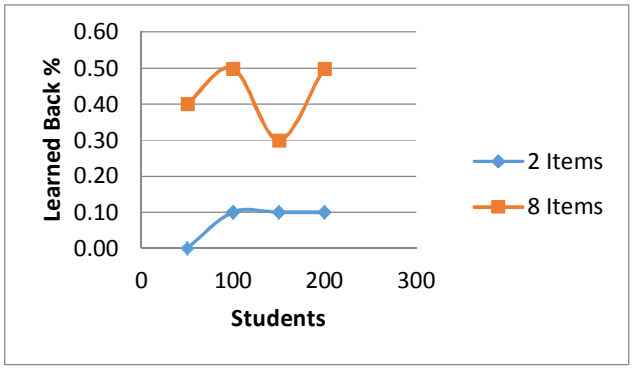

Fig. 7. Percent of graphs learned back for student ranges 50-200 and 2+8 items per skill.

4.5 Experiment 4

We ran experiment 4 to confirm that the number of students has an impact on the recoverability of the original graph, fixing all other parameters at reasonable values and varying the number of students. For this experiment, guess and slip values were set at 0.1 and 0.08 respectively. We used graph type 4 (fig. 6), set the number of items to 4 and fake skills at 1, varying the location of the fake skill. The student numbers were varied from 10 to 100. For each student number, the evaluation was run 10 times. The results, in fig 8, show that as we intuitively assumed, the number of students has a huge impact on the algorithmôs ability to learn back the true graph. The results show that as the number of students increases the probability of a skill graph being learned back increases whiles at the same time the RMSE reduces. These results, we found, are significant with p-values below 0.01. This finding confirms that student numbers is an important factor that needs to be considered when refining learning maps.

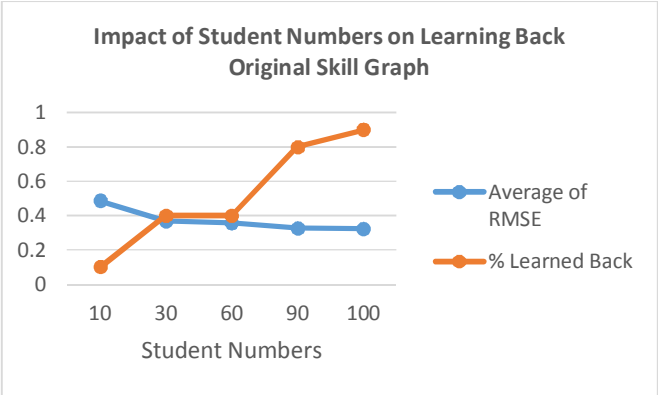

Fig. 8. Impact of Student Numbers

5 Conclusion

Many learning maps/cognitive models are built from expert knowledge. With the production of lots of educational data on student performance, it has become imperative to find data centered methods of improving upon these expert-designed learning maps. In our earlier studies we designed and presented an algorithm for simplifying/improving the predictive accuracy of these models. In this paper we have presented a number of factors that influence the data centered model improvement process we initially published. We have shown with our simulation studies that the guess/slip values, number of items per skill, the number of students and the number of fake skills in the graph affect the simplification of the skill models. We also explored many parameters to see how much data is needed to recover the true learning maps. For future work we plan to continue to evaluate our algorithm on larger examples to see how well our algorithm can scale up and test it on well-known real data sets.

6 References

1. Adjei, S. A., Selent, D., Heffernan, N. T., Broadus, A, Kingston, N. (2013) Refining Learning Maps with Data Fitting Techniques: Searching for Better Fitting Learning Maps, International Conference on Learning Sciences (Submitted)
2. 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.
3. Corbett, Albert T., and John R. Anderson.(1994) "Knowledge tracing: Modeling the acquisition of procedural knowledge." *User modeling and user-adapted interaction* 4.4 (1994): 253-278.
4. Embretson, S. E. (1998). A cognitive design system approach to generating valid tests: Application to abstract reasoning. Psychological Methods, 3(3), 380-396.
5. Gierl, M. J., Leighton, J. P., & Hunka, S. M. (2007). Using the attribute hierarchy method to make diagnostic inferences about examinees' cognitive skills.
6. Leighton, J. P., Gierl, M. J., & Hunka, S. M. (2004). The attribute hierarchy method for cognitive assessment: A variation on Tatsuoka's rule-space approach. *Journal of Educational Measurement*, 41, 205-236.
7. Mislevy, R. J., Steinberg, L. S., & Almond, R. G. (2002). Design and analysis in task-based language assessment. Language Testing, 19(4), 477-496.
8. Pardos, Zachary A., and Neil T. Heffernan.(2010) "Navigating the parameter space of Bayesian Knowledge Tracing models: Visualizations of the convergence of the Expectation Maximization algorithm." In *EDM*, pp. 161-170. 2010.
9. Sheehan, K. M. (1997). A tree-based approach to proficiency scaling and diagnostic assessment. Journal of Educational Measurement, 34(4), 333-352.
10. Tatsuoka, K. K. (1983), Rule space: an approach for dealing with misconceptions based on item response theory. *Journal of Educational Measurement*, 20: 345ï 354.
11. Tatsuoka, K. K. (1995). Architecture of knowledge structures and cognitive diagnosis. In P. D. Nichols, S. F. Chipman & R. L. Brennan (Eds.), Cognitively diagnostic assessment (pp. 327-359). Hillsdale, NJ: Erlbaum