Modelling Learning in an Educational Game

Micheline Manske, Cristina Conati
Department of Computer Science, University of British Columbia,
Vancouver, BC, V6T1Z4, Canada
{manske, conati}@cs.ubc.ca

Abstract. We describe research on data-drive refinement and evaluation of a probabilistic model of student learning for an educational game on number factorization. The model is to be used by an intelligent pedagogical agent to improve student learning during game play. An initial version of the model was designed based on teachers' advice and subjective parameter settings. Here we illustrate data-driven improvements to the model, and we report results on its accuracy.

1. Introduction

A student model is one of the fundamental components of an intelligent learning environment [11], and much research has been devoted to creating student models for various types of computer based support. However, little work exists on student modelling for a relatively new type of pedagogical interaction, educational computer games (edu-games from now on). In this paper, we describe the design and evaluation of a student model to assess student learning during the interaction with Prime Climb, an edu-game for number factorization.

The main contribution of this work is a step toward providing intelligent computer based support to learning with edu-games. Providing this support is both extremely valuable and extremely challenging. It is valuable because, although there is overwhelming evidence that even fairly simple edu-games can be highly motivating, there is little evidence that these games, no matter how sophisticated they are, can actually trigger learning, unless they are integrated with ad hoc supporting activities [5,9,6]. This is because many students manage to successfully play these games without necessarily having to reason about the underlying domain knowledge. We argue that individualized support based on careful assessment of student learning during game playing can help overcome this limitation and make edu-games an effective new form of learning.

Providing this support is challenging because it requires careful tradeoffs between fostering learning and maintaining positive affective engagement. Thus, it is crucial to have accurate models of both student learning and affect. Creating these models is hard, however, because it necessitates understanding about cognitive and affective processes on which there is very little knowledge, given the relative novelty of games as educational tools. In [2] we present a model of student affect for the Prime Climb edu-game. Here we focus on the model of student learning. In particular, we describe the data-drive refinement and evaluation of an initial model based on expert knowledge and subjective judgements, previously described in [3].

There is increasing research in learning student models from data (e.g., [1,4,7]), but most of this research has focused on student models for more traditional ITS systems. An exception is [8], which describes a student model learned from data for a game designed to address common misconceptions about decimal numbers. The data used in [8] come from students' performance on a traditional test to detect decimal number misconceptions. Thus, the model parameters learned from these data, (e.g., the probability of an error of distraction (*slip*) or a

lucky guess), do not reflect the actual relationship between student performance and knowledge during game playing. This relationship is likely to be different than in traditional tests. Several studies have shown that students can be successful game players by learning superficial heuristics rather then by reasoning about the underlying domain knowledge. Furthermore, students may make more slips during game playing, because they are distracted by the game aspect of the interaction. In the work presented here, the data used to learn the student model comes from interaction with Prime Climb. Thus, the model parameters provide us with insights on how students learn and interact with this type of educational system, in itself a contribution given the relative lack of understanding of these mechanisms.

In the rest of the paper, we first introduce the Prime Climb game and an initial version of its student model (both described in more details in [3]). Next, we present a study to evaluate this model's accuracy. We then describe a data-drive refinement of the model, assess its accuracy and analyze the sensitivity to its various parameters. Finally, we introduce a further improvement with the modelling of common factoring, and compare the three student models.

2. The Prime Climb Game and Initial Student Model

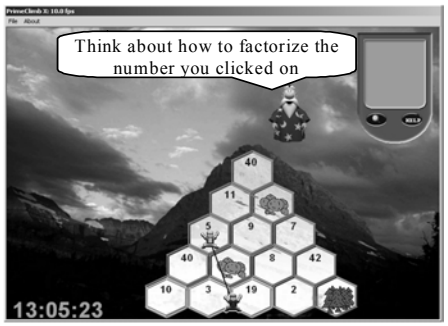


Figure 1a: The Prime Climb Interface

b: A factor tree displayed in the PDA

In Prime Climb (devised by the EGEMS group at the University of British Columbia) students in 6th and 7th grade practice number factorization by pairing up to climb a series of mountains. Each mountain is divided into numbered sectors (see Figure 1a), and players must try to move to numbers that do not share common factors with their partner's number, otherwise they fall. To help students, Prime Climb includes the Magnifying Glass, a tool that allows players to view the factor tree for any number on a mountain. This factor tree is shown in the PDA displayed at the top right corner of the game (see Figure 1b).

Each student also has a pedagogical agent (Figure 1a) which provides individualized support, both on demand and unsolicited, when the student does not seem to be learning from the game (see [3] for more details on the agent's behaviours). To provide appropriate interventions, the agent must have an accurate model of student learning. However, this modelling task involves a high level of uncertainty because, as we discussed earlier, game performance tends to be a fairly unreliable reflection of student knowledge. We use Dynamic Bayesian networks (DBNs) to handle this uncertainty.

A DBN consists of *time slices* representing relevant temporal states in the process to be modelled. In Prime Climb, there is a DBN for each mountain that a student climbs (the *short-term student model*). A time slice is created in this network after every student action, to capture the evolution of student knowledge as the climb proceeds. Each short term model includes the following random binary variables:

- *Factorization (F) Nodes:* each factorization node F_x represents whether the student has mastered the factorization of number x down to its prime factors.
- *Knowledge of Factor Tree (KFT) Node:* models knowledge of the factor tree representation.

- *Click Nodes*: each click node C_x models the correctness of a student's click on number *x*.
- *Magnification (Mag) Nodes :* each Mag_x node denotes using the magnifying glass on number *x*.

The network for a given mountain includes F nodes for all its numbers, F nodes for their factors, and the KFT node. Click and Mag nodes are introduced in the model when the corresponding actions occur, and are immediately set to one of their values.

Figure 2 illustrates the structure that we used in the first version of the model to represent the relations between factorization and click nodes[1]. A key assumption underlying this structure, derived from mathematics teachers, is that knowing the prime factorization of a number influences the probability of knowing the factorization of its factors, while the opposite is not true. It is hard to predict if a student knows a number's factorization given that s/he knows how to factorize its non-prime factors.

To represent this assumption, F nodes are linked as parents of nodes representing their non-prime factors. The conditional probability table (CPT) for each non-root F node (e.g. F_x in Figure 2a) is defined so that the probability of the node being known is high when all the parent F nodes are true, and decreases proportionally with the number of unknown parents. The action of clicking on number *x* when the partner is on number *k* is represented by adding a click node C_x as parent of nodes F_x and F_k (see Figure 2b). Thus, evidence coming from click actions is represented in the diagnostic rather than causal direction. This structure prevents evidence on a number *x* from propagating upwards to the numbers that contain it as a factor (e.g. F_z in Figure 2b), thus respecting the insights provided by our teachers.

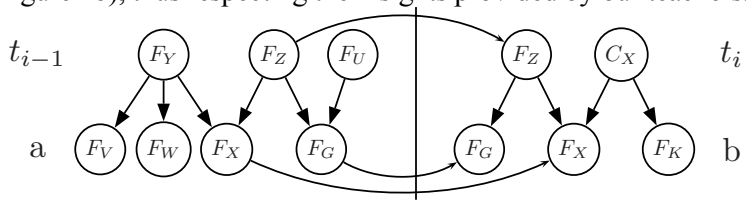


Figure 2. a: Factorization nodes, where Z=X*G and Y=V*W*X; **b:** Click action

Note that this model has two major limitations. The first is that it does not apportion blame for an incorrect click in a principled way. The two F nodes involved in a click should be conditionally dependent given the action, so that the node with the lower probability can be "blamed" more for the incorrect move. This dependency could be modelled by adding a link between the two F nodes (e.g, F_x and F_k in Fig. 2b), however this would increase the model's complexity so we chose not to. The second limitation is that the model does not include a node to explicilty represent knowledge of the common factor concept, which is a key component in playing the game successfully.

Although we were aware of these limitations, we wanted to investigate how far this relatively simple model would take us. In an initial study, the game with the agent giving help based on the above model generated significantly better learning than the game without agent [3]. However, the study was not designed to ascertain the role of the model in this learning. Hence, we ran a second study specifically designed to determine the model's accuracy.

2.1 Study for Model Evaluation

The study included data from 52 students in 6th and 7th grade. Each student played Prime Climb for approximately 10 minutes, with an experimenter as partner. All game actions were logged. Students were given identical pre and post-tests to gauge their factorization knowledge of 10 numbers frequently involved in the first two game levels, as well as their understanding

[1] We don't discuss the mechanisms to model learning through usage of the magnifying glass, because they are not involved in the model refinement process discussed here. See [3] for more details.

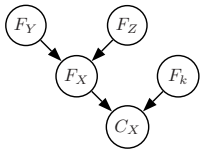
of the common factoring concept. We used the post-test answers to evaluate the model's assessment after game play (as explained in section 3.1). Despite an effort to fine-tune the model using data from the study, its accuracy was no better than chance (50.8%). This is not surprising, given the model limitations described above. The fact that agent condition showed significantly better learning indicates that even hints based on an almost random model are better than no hints at all. However, the fact that there was still large room for improvement in the post-tests of the agent-condition suggests that a more accurate student model may yield even more substantial learning gains. Thus, we set to improve our model to incrementally address the two limitations discussed earlier. This process resulted in two new versions of the model, both with parameters learned from data, which we illustrate in the following sections.

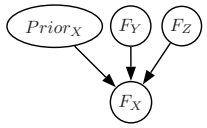
3. New Model – Causal Structure

One of the limitations of the original model is that it did not correctly apportion blame for incorrect moves. The new model uses a causal structure over click nodes to fix this problem. Each click node is added as child of the two F nodes involved in the click (see Figure 3a in contrast to Figure 2b). Thus, these nodes become conditionally dependent given a click and share the blame for an incorrect action proportionally to their probability.

F_x	F_k	P(Click=C)
K	K	*1-α*
K	U	*e_guess*
U	K	*e_guess*
U	U	*guess*

a





b

Prior	F_z	F_y	$P(F_x=K)$
K	K or U		1
U	K	K	*max*
	K	U	*max*/2
	U	K	*max*/2
	U	U	0

Figure 3. a: Click configuration at time t_i.; **b:** Roll-up on node F_x at time t_{i+1} when node F_x has two parents. K: known, U: unknown, C: correct

The three parameters needed to specify this configuration are α*, e_guess,* and *guess* (Figure 3a). The α parameter represents the probability of making an incorrect move despite knowing the factors of the relevant numbers, because of either a slip or lack of understanding of the common factoring concept. The *guess* parameter represents the probability of a correct move when both the numbers involved are unknown. The *e_guess* (educated guess) parameter is introduced to represents the possibility that it is easier to guess correctly when knowing the factorization of one of the numbers.

To reduce the computational complexity of evaluating the short-term model, at any given time we maintain at most two time slices in the DBN. This requires a process known as *roll-up*, i.e. saving the posterior probabilities of the slice that is removed (e.g., slice in Figure 3a) into the new slice that is created (e.g., slice in Figure 3b). Posterior probabilities of root nodes in the removed slice are simply saved as priors of the corresponding nodes in the new slice. For non-root nodes the process is more complicated, and requires different approaches for various network configurations [3,10]. The approach proposed here is as follows: for every non-root F node that needs to be rolled up (e.g. F_x in Figure 3a) we introduce an additional *Prior* node in the new time slice (e.g. $Prior_x$ in Figure 3b), and give it as a prior the posterior of the F node in the previous time slice.

The CPT for the F node in the new slice (see table for F_x in Figure 3b) is set up such that knowing the factorization in the previous time slice implies knowing the factorization in the current slice (i.e. we do not model forgetting). Otherwise, the probability of the node being known is 0 when all the parent F nodes are unknown, and increases proportionally with the number of known parents to a maximum of *max*, the probability that the student can infer the factorization of *x* by knowing the factorization of its parent nodes.

We now describe how we learn the parameters *α, e_guess, guess,* and *max* from data from the user study described in the previous section.

3.1 Setting Parameters from Data

When all the nodes involved in a given CPT are observable, the CPT values can be learned from frequency data. F nodes are not usually observable, however, we have pre and post-test assessment on 10 of these nodes for each of our 52 students. If we consider data points in which pre and post-test had the same answer, we can assume that the value of the corresponding F nodes remained constant throughout the interaction (i.e. no learning happened), and can use these points to compute the frequencies for the CPT entries involving *α, guess,* and *e_guess*. We found 58 such data points in our log files, yielding the frequencies in Table 1.

Table 1: Parameter estimates from click frequencies

Parameter	Freq	Points
a	0.23	44
e_guess	0.75	12
guess	0	2

As Table 1 shows, the frequency for the *α* parameter is based on 44 points, thus we feel confident fixing its value at 0.23. However, because we have far fewer points for the *e_guess* and *guess* parameters we must estimate these parameters in another manner. Similarly, we cannot use frequencies to set the *max* parameter as we do not have data on *Prior* nodes, which represent the (possibly changing) student knowledge at any given point in the interaction.

To select ideal values for *e_guess, guess* and *max* we attempt to fit the data to the answers students gave on post-tests. We fix the parameters to a specific triplet, feed each student's log file to the model, and then compare the model's posterior probabilities over the 10 relevant F nodes with the corresponding post-test answers. Repeating this for our 52 students yields 520 <*model prediction, student answer*> pairs for computing model accuracy. Since it would be infeasible to repeat this process for every combination of parameter values, we select initial parameter values by frequency estimates and intuition. Next we determine whether the model is sensitive to any of the three parameters, and if so, try other parameter settings. The values used initially for *e_guess* were {0.5,0.6,0.7}, chosen using Table 1 as starting point. For *guess* there are too few cases to base the initial values on frequencies, so we rely on the intuition that they should be less than or equal to the *e_guess* values, and thus use {0.4,0.5,0.6}. For *max* we use {0,0.2,0.4}. We try all 27 possible combinations of these values and chose the setting with the highest model accuracy.

To avoid over fitting the data, we perform 10-fold cross-validation by splitting our 520 data points to create 10 training/test folds. For each fold, we select the parameter triplet which yields the highest accuracy on the 90% of the data that forms that training set, and we report its accuracy on the 10% in the test set. We then select the parameter setting with the best training set performance across folds.

As our measure of accuracy, we chose (sensitivity + specificity)/2 [12]. Sensitivity is the percentage of known numbers that the model classifies as such; specificity is the percentage of unknown numbers classified as such. Thus, we need a threshold that allows us to classify model probabilities as *known* or *unknown*. To select an adequate threshold, we picked several different threshold values, and computed the average model accuracy on training set across all 10 folds and 27 parameter settings. The threshold yielding the highest average accuracy was 0.8 (see Table 2). Note that the standard deviation across folds is low, indicating that we are not over fitting the data.

Using a threshold of 0.8, the setting with best performance across all 10 folds (highest accuracy in all but one of the folds) was 0.5 for both *e_guess* and *guess* and 0 for *max*. The fact that the two guess parameters are high confirms previous findings that students can often perform well in educational games through lucky guesses or other heuristics not requiring correct domain knowledge.

Table 2: Average training set accuracy across folds by threshold

Threshold	Accuracy	Std. Dev.
0.4	0.624	0.010
0.5	0.697	0.009
0.65	0.753	0.007
0.8	0.772	0.007
0.95	0.725	0.006

The fact that they are equal indicates that there is no substantial difference in the likelihood of a lucky guess given different degrees of domain knowledge. The setting of 0 for *max* indicates that the teacher-suggested relation between knowing the factorization of a number and knowing the factorization of its non-prime factors may be too tenuous to make a difference in our model (more on this in the next section).

Using these settings, our model achieves an average test set accuracy of 0.776, with a sensitivity of 0.767, and a specificity of 0.786. This is a substantial improvement over the 0.508 accuracy of the old model.

3.2 Model Sensitivity to Individual Parameters

To investigate how sensitive our model is to each parameter, we fix two of the parameters and calculate the standard deviation of the model's accuracy across all three values of the third. This yields an average standard deviation of 0.002 for *e_guess*, 0.005 for *guess*, and 0.002 for *max*, indicating low sensitivity to small changes in these parameters. To rule out the possibility that the three values we initially chose for each parameter were not ideal, we try more extreme values (0.3 and 0.1 for *guess* and *e_guess*; 0.6 and 0.8 for *max*). All of them yielded worse accuracy, indicating that the model is sensitive to larger changes in these parameters. Slight variation of the α parameter also produced little change in accuracy, with more extreme values (0.1 and 0.5) decreasing accuracy. These results indicate that we were able to identify adequate value ranges for the parameters in our new model configuration, and that the model is not sensitive to small changes of these parameters in the given ranges. They also suggest that we could select a value slightly higher than 0 for the *max* parameter if we want to maintain the teacher-suggested relationship among F nodes in the model, or we can choose to ignore these relationships if we need to improve the efficiency of model update.

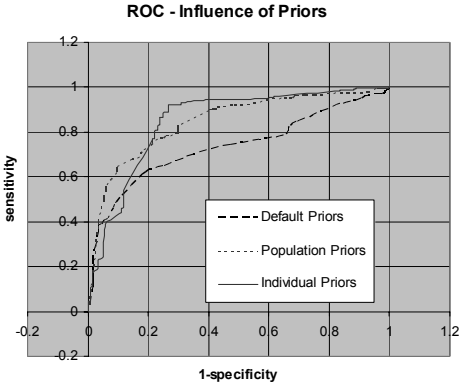


Figure 4: ROC curves comparing priors influence on sensitivity and specificity.

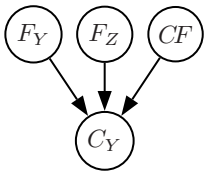
Finally, we analyzed the sensitivity of the model to the initial prior probability of F nodes. All results presented thus far have used *population* priors derived from frequencies over all students' pre-tests. We tried two more settings: (i) *Default*, which gives a prior of 0.5 for each F node; (ii) *Individual*, with priors derived from each student's pre-test answers. As the Receiver-Operator Curve (ROC) in Figure 4 show, population priors and individualized priors do better than default priors at most thresholds. However, the model can still have good performance even when accurate priors are not available (maximum accuracy is 0.717 for default, 0.776 for population, and 0.828 for individualized).

Although this new model has shown significant gains in accuracy, we wanted to see whether we could get further improvements by addressing the second limitation of the original model: omitting the concept of common factoring. We discuss its addition in the next section.

4. Modelling Common Factoring Knowledge

Because the model discussed above does not model common factor knowledge, when a student makes an incorrect move despite knowing the factorization of both numbers involved, the

model can only infer that the student either made a slip or does not know the concept of common factors. This limits the system's capability to provide precise feedback based solely on model assessment. However, modelling common factor knowledge increases model complexity. To see how much we can be gained from this addition, we generated a new model that includes a common factor node (CF) as a parent of each click action (Figure 5). Note that the CPT entry corresponding to an incorrect action when all the parent nodes are known now isolates the probability of a slip. As before, the *guess* and *edu-guess* parameters in the CPT reflect potential differences in the likelihood of a lucky guess given different levels of existing knowledge.



CF	Fy	Fz	P(Click=C)
K	K	K	1-*slip*
K	K	U	*e_guess*
K	U	K	*e_guess*
K	U	U	*guess*
U	K or U		*guess*

Figure 5: Click configuration with common factor node

We use the same process described in the previous sections to set the parameters in the new model. Optimal threshold is again 0.8, while optimal parameter setting is 0.2 for *slip*, 0.6 for *e_guess* and *guess*, and 0 for *max*, showing good consistency with parameters in the model without CF node. Like that model, the new model is also not very sensitive to small changes in the parameters. Its average test set accuracy with population priors across all folds is 0.768 (SD 0.064) over F nodes and 0.654 for CF node (SD 0.08).

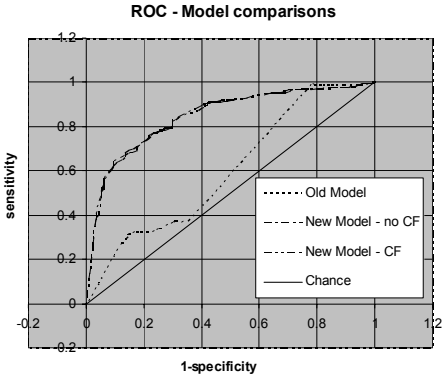


Figure 6: ROC curve comparisons of the three models and chance.

Figure 6 compares the accuracy of the three models and of a baseline chance model in assessing number factorization knowledge. As we can see, the accuracy of the assessment on number F nodes does not change considerably between the CF and no CF version. Furthermore, the assessment accuracy over CF is not very high. This may suggest that the addition of the CF node would not substantially increase the model's capability to support precise didactic interventions, and thus may not be worth the potential delays in model updates due to larger CPTs. However, two factors speak to the contrary. The first is that we have not seen these delays in our test runs. The second is that our current data may not be sufficient for accurate parameter learning in this more complex model, as it is suggested by the larger standard deviation of accuracy across folds compared to the no CF version. We plan to gather more data and see if that improves accuracy in the CF assessment of the model.

5. Discussion and Future Work

Although even simple games like Prime Climb are extremely motivating for students, as we observed during our studies, there is currently very little evidence that simple or complex edu-games trigger learning. Usually this is not because of poor design, but because it is difficult to introduce intervention elements that make students reflect on domain knowledge without interfering with engagement. An accurate model of student learning is essential for balancing the trade-off between fostering learning and engagement in an educational game.

In this paper, we presented research to improve a model of student learning during the interaction with Prime Climb, an edu-game for number factorization. The model is to be used by a pedagogical agent that generates tailored interventions to trigger student reasoning when the student seems not to be learning well from the game. We discussed how we substantially improved the accuracy of an initial model by (i) changing the causality of the dependencies between knowledge and evidence nodes; (ii) learning model parameters from data. We also described a third version of the model that includes a common factor node to increase the specificity of the didactic advice that the model can support.

The next step in this research is to explore whether we can further increase model accuracy by (1) obtaining data to refine the part of the model that includes information on usage of the Magnifying Glass [3]; (2) including in the model the Prime Climb agent's interventions, which are currently not considered because we wanted to ascertain model accuracy before adding agent actions that relied on the model.

We also plan to run ablation studies to verify what impact the model accuracy has on overall effectiveness of the pedagogical agent. Finally, we wish to explore the scalability of our approach to modelling learning in more complex games and skills.

Acknowledgments

This research has been sponsored by an NSERC PGS-M scholarship. We thank Heather Maclaren helping with the user study, and Giuseppe Carenini for his help with data analysis

References

[1] Beck, J., P Jia and J. Mostow. Assessing Student Proficiency in a Reading Tutor That Listens. User Modeling 2003: pp. 323-327.

[2] Conati, C. and H. Maclaren. Data-driven Refinement of a Probabilistic Model of User Affect. To appear in User Modeling 2005.

[3] Conati, C. and X. Zhao. Building and Evaluating an Intelligent Pedagogical Agent to Improve the Effectiveness of an Educational Game. Intelligent User Interfaces 2004. pp. 6-13.

[4] Croteau, E. A., N. T. Heffernan and K. R. Koedinger. Why Are Algebra Word Problems Difficult? Using Tutorial Log Files and the Power Law of Learning to Select the Best Fitting Cognitive Model. Intelligent Tutoring Systems 2004. pp. 240-250.

[5] Klawe, M. When Does The Use Of Computer Games And Other Interactive Multimedia Software Help Students Learn Mathematics? NCTM Standards 2000 Technology Conference, 1998.

[6] Leemkuil, H., T. De Jong, R. deHoog, and N. Christoph. KM Quest: A collaborative Internet-based simulation game. Simulation & Gaming, 2003, 34(1).

[7] M. Mayo and A. Mitrovic. Optimising ITS Behaviour with Bayesian Networks and Decision Theory. International Journal of Artificial Intelligence in Education 2001. 12, pp 124-153.

[8] Nicholson, A.E., T. Boneh, T.A. Wilkin, K. Stacey, L. Sonenberg, V. Steinle: A Case Study in Knowledge Discovery and Elicitation in an Intelligent Tutoring Application. Uncertainty in Artificial Intelligence 2001.

[9] Randel, J.M., B.A. Morris, C.D. Wetzel, and B.V. Whitehill, The effectiveness of games for educational purposes: A review of recent research. Simulation & Gaming, 1992, 23(3).

[10] Schafer, R. And T. Weyrath. Assessing Temporally Variable User Properties with Dynamic Bayesian Networks. User Modeling 1997.

[11] VanLehn, K. Student modeling. Foundations of Intelligent Tutoring Systems. M. Polson and J. Richardson. Hillsdale, NJ, Lawrence Erlbaum Associates. (1988). pp. 55-78.

[12] VanLehn, K. and Z. Niu Bayesian student modeling, user interfaces and feedback: A sensitivity analysis. International Journal of Artificial Intelligence in Education, 2001. 12, pp. 154-184.