Extending Knowledge Tracing to Allow Partial Credit: Using Continuous versus Binary Nodes

Yutao Wang and Neil Heffernan

Worcester Polytechnic Institute
{yutaowang,nth}@wpi.edu

Abstract. Both Knowledge Tracing and Performance Factors Analysis, are examples of student modeling frameworks commonly used in AIED systems (i.e., Intelligent Tutoring Systems). Both of them use student correctness as a binary input, but student performance on a question might better be represented with a continuous value representing a type of partial credit. Intuitively, a student who has to make more attempts, or has to ask for more hints, deserves a score closer to zero, while students who asks for no hints and just needs to make a second attempt on a question should get a score close to one. In this work, we present a simple change to the Knowledge Tracing model and a simple (non-optimized) method for assigning partial credit. We report our real data experiment result in which we compared the original Knowledge Tracing (OKT) model with this new Knowledge Tracing model that uses partial credit as input (KTPC). The new model outperforms the traditional model reliably. The practical implication of this work is that this new technique can be widely used easily, as it is a small change from the traditional way of fitting KT models.

Keywords: Knowledge Tracing, Intelligent Tutoring Systems, Student Responses, Partial Credit.

1 Introduction

In many important student models, such as the Knowledge Tracing model and the Performance Factor Analysis (Pavlik, Cen and Koedinger 2009), student performance is presented as a binary value of correct or incorrect. The amount of assistance a student needed to eventually get a problem correct is ignored in these models. Feng and Heffernan (2010) showed that we can predict student performance better by accounting for amount of assistance they received, but they did not provide the field with a model that could be used in "run time" to predict individual responses. Arroyo, et al.(2010) showed how to use this information to predict learning gains. Their work suggests that using hints and attempts to model student behavior online could be effective.

There is good work in the psychometric literature on using partial credit, which goes back 30 years. Psychometricians have shown that different multiple choice answers might worth different credits [6, 10]. For instance, choice A might be totally wrong but choice B is close, choice C is the correct answer.

More recently, a new type of partial credit is coming online. For instance, Attila and Powers (2010) at the Educational Testing Service showed they could better predict student GRE scores if they let students make multiple attempts. Their score on a question would go down by a third for each attempt (students could only make three attempts). Our work generalizes their work in two ways. First, we show how to incorporate the partial credit score into a model with learning (i.e., Knowledge Tracing) as their model did not model learning. Second, we show how to incorporate penalties for each hint student request.

In our previous work (Wang and Heffernan, 2010), we presented a naïve algorithm to assign partial credit, and showed it accounts for some variance in student knowledge. But in that work, we did not present a model that could do this task. In this paper we want to see if we can improve one of the dominant methods of student modeling (i.e., the Knowledge Tracing model) by relaxing the assumption of binary correctness: replacing the discrete performance node with continuous partial credit node.

In the next section, we describe our modification to the original Knowledge Tracing (OKT) model, and the method we use to make the correctness continuous. Section 3 describes the tutoring system and dataset used in our experiments and the experiment result. In Sections 5 and 6 we discuss our conclusions and future directions for our work.

2 Approach

2.1 Knowledge Tracing with Continuous Performance Node

The Knowledge Tracing model shown in Fig.1 has been widely used in ITS to model student knowledge and learning over time. It has become the dominant method of student modeling and many variants have been developed to improve its performance (Baker et al., 2010, Pardos and Heffernan 2010). Knowledge Tracing uses one latent and one observable dynamic Bayesian network to model student learning. As shown in Fig.1, four parameters are used for each skill, with two for student knowledge (initial knowledge and probability of learning the skill) and the other two for student performance (the probability of guessing correctly when the student doesn't know the skill and the probability of slipping when the student does know the skill).

The structure of the Knowledge Tracing model with a continuous performance node is the same as the original Knowledge Tracing model. The only difference is how we set up the "Student Performance" node. The idea is straight forward, yet there has never been positive result reported in this field. Some other Intelligent Tutoring System groups, such as LISTEN (http://www.cs.cmu.edu/~listen/) tried this approach before but failed for unknown reasons.

In this model, instead of assign the "*Guess*" and "*Slip*" parameters in a CPT table as the original Knowledge Tracing model, we assigned two Gaussian distributions for "*Guess*" and "*Slip*" with given standard deviations. Four parameters: *guess_mu*, *guess_sigma*, *slip_mu*, *slip_sigma*, are used to describe the two Gaussian distributions.

Similarly, when we predict student performance, we also get a Gaussian distribution with a mean value and a standard deviation value, in which the mean value will be the prediction and the standard deviation contains the information of how good the prediction is. In this work, we are not using the standard deviation of the prediction, but it has potential to be useful in the future to determine how confident we are in our prediction.

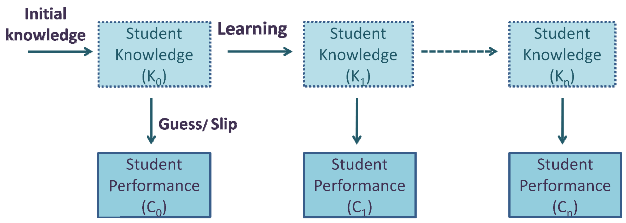


Fig. 1. Knowledge Tracing model

(Figure comes from Gong, Beck et al. 2010)

In our experiment, we used the Bayes Net Toolbox for Matlab developed by Murphy (2001) to implement Knowledge Tracing and the Expectation Maximization (EM) algorithm. The EM algorithm finds a set of parameters that maximizes the likelihood of the data. Since EM can be sensitive to initial conditions (Pardos and Heffernan 2010b), we report the initial settings. We used *initial knowledge* = 0.5, *learning* = 0.1, *guess_mu* = 0.1, *guess_sigma* = 0.02, *slip_mu* = 0.1, *slip_sigma* = 0.02 for the KTPC model, and *knowledge* = 0.5, *learning* = 0.1, *guess* = 0.1, *slip* = 0.1 for the OKT model.

2.2 Make the Correctness Continuous: Partial Credit

Partial credit can be assigned in different ways. In our experiment, we are using the algorithm that was mentioned in our previous poster [11] to make the correctness to be continuous. Since we never introduce the algorithm completely, it is described in this section in detail.

In a model with binary performance, a student would get a ‘1’ if he/she answered the problem correctly on the attempt without asking for a hint and ‘0’ otherwise. For the purpose of this paper we “made up” a scoring method that would give students a score between ‘0’ and ‘1’ according to how many attempts and how many hints they required to answer a question correctly based on intuition. We are well aware that this method could be optimized in lots of ways, for example, should each hint cost the same, or should the first hint cost less? As shown in our result, this simple method is effective and we leave to others different ways to optimize it.

Intuitively, the more hints that are asked for, the less likely it is that the student understands the skill, so we penalize a student for each hint asked for by what we call the hint penalty, which is 1 divided by the total number of hints available. For example, if there are 4 hints possible and a student asks for three of them and then gets the problem correct he/she would get a .25 score. In a similar manner, more attempts indicate a lower possibility of understanding the required skill, and we penalize each attempt. The size of the penalty depends upon whether the question type is “multiple choice” or “Fill in the Blank”. In our data set, we have about 80% questions that are “Fill in the Blank” questions, for which we picked a penalty 0.1 for each wrong attempt. For multiple choice questions with *x* choices, the penalty was computed by one over the number of remaining multiple choice options minus one. So a true false question will have a penalty of one if a student guessed wrong. If there were 4 choices, a student's first wrong attempt would get a penalty of 1/3, a second wrong attempt would get a penalty of ½, and a third wrong attempt would get a penalty of 1.

After computing hint penalty (*phint*) for each hint and attempt penalty (*pattempt*) for each attempt, we add them together to compute the total hint penalty (*total_phint*) and the total attempt penalty (*total_pattempt*) for this problem. If the number is less than zero we make it zero. The last column of Table 2 shows two examples of formula doing this calculation.

Table 1 shows the details of computing partial credit for scaffolding questions. Our dataset has a special type of feedback called scaffolding. Since it's only a small amount of our data this detail might not be that important. But for completeness, we wanted to describe this. (Please note that all of our code and data are available at *http://users.wpi.edu/~yutaowang/* so that others can attempt to improve upon our work). For those problems with scaffolding questions, if a student gets the original question wrong, the system will give the student a series of questions we call "scaffolding" that walk the student through the steps. Each of these scaffolding questions has hints and so can be scored with this partial credit function just like normal questions. The only question left is how to score the "original question". If a student gets a question wrong and is given three scaffolding questions, the total credit of the whole problem is computed by averaging the partial credit scores of the three scaffolding questions and penalized by 10% for answering the original question incorrectly. If a student got the original question wrong but then got all the scaffolding questions correct, he/she should get a score close to 1, which in our method would be 0.9. Again these parameters such as 0.9 are not optimized and could be learned from data in future work.

Table 1. The algorithm of computing partial credit

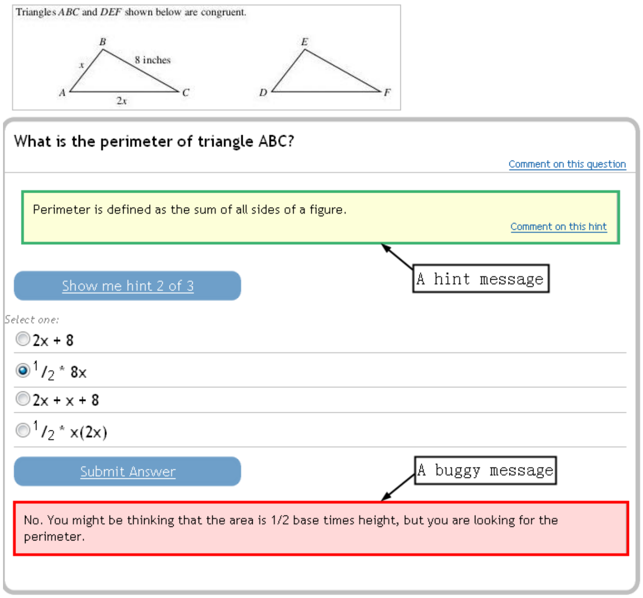
```
function pc = partial_credit(problem){

    if first attempt correct then
        return pc = 1
    else if problem has no scaffold then
        pc = 1 - #hint * phint – total_pattempt
        if pc<0 then return pc = 0
        return pc
    else
        for each scaffold question i in the problem do
            pc_scaffold(i)   =   partial_credit(scaffold(i))
        end for
        pc = 0.9 * average(pc_scaffold(i))
        return pc
}
```

The algorithm is used only for testing the effect of relaxing the assumption of binary correctness in a Knowledge Tracing model.

3 Evaluation

3.1 The Tutoring System and Dataset

Our dataset consisted of student responses from ASSISTments, a web based tutoring system for 7th-12th grade students that provides preparation for the state standardized test by using released math items from previous years' tests as questions. The tutorial helps the student learn the required knowledge by breaking the problem into sub questions called scaffolding or giving the student hints on how to solve the question. Fig.2. shows an example of a hint. A second type of assistance is presented if the student clicks on (or types in) an incorrect answer, at which point the student is given feedback that he/she answered incorrectly (sometimes, but by no means always, the student will get a context-sensitive message called "buggy message"). Examples can be seen at "tinyurl.com/buaesc2".


Triangles *ABC* and *DEF* shown below are congruent.

What is the perimeter of triangle ABC?

Perimeter is defined as the sum of all sides of a figure.

Select one:
- ○ $2x + 8$
- ● $^1/_2 * 8x$
- ○ $2x + x + 8$
- ○ $^1/_2 * x(2x)$

No. You might be thinking that the area is 1/2 base times height, but you are looking for the perimeter.

Fig. 2. Assistance in ASSISTment

The data we analyzed was drawn from ASSISTments. It comes from 72 twelve-through fourteen-year old 8th grade students in a school district of the Northeast United States. There were 106 skills (e.g., area of polygon, Venn diagram, division, etc.) that students were working on. The data consisted of 52,529 log records during the period Jan 2009-Feb 2009 where each log record is similar to one row in Table 2, which shows the details of one problem done by one student. We use the same data format as the KDD

Cup 2010: Educational Data Mining Challenge (https://pslcdatashop.web.cmu.edu/ KDDCup/FAQ/#data-format). Table 2 shows an example of the type of data we used. There are in total 12 columns, the first 9 columns in the table are straight from the KDD Cup data format (https://pslcdatashop.web. cmu.edu/KDDCup/rules_data_format.jsp), and we added three extra columns, which are used for partial credit. In particular, column 10 "Number of Choices (if Multiple Choice)" was added to describe if the problem is multiple choice problem or not, and how many choices there are. Total number of hints available for the problem is put in column 11, to help compute the partial penalty per hint. The last hint always gives away the answer, so if a student asked for all of the hints, their score should be zero. This column allows us to give a bigger penalty for hints if the number of total available hints is small. Column 12 is for showing how we compute the partial credit score, a continuous value between 0 and 1 that the student would get given the data log. Note that the original KT model will only use the 7th column, "Error Rate", as model input; while the KT with partial credit model will only use the 12th column, "Partial Credit". The 7th column is generated as 1 if the student answered the problem correctly, otherwise 0.

Table 2. An example of a few rows of data, showing how we calculate partial credit

1.Row	**2.Student**	**3.Problem**	**4.Step**	**5.Incorrects**	**6.Hints**	**7.Error Rate**
1	S01	WATERING _VEGGIES	(WATERED-AREA Q1)	0	0	0
2	S01	WATERING _VEGGIES	(TOTAL-GARDEN Q1)	2	1	1

8.Knowledge component	**9.Opportunity Count**	**10.Number of Choices (If Multiple Choice)**	**11.Total Hints Available**	**12.Partial Credit**
Circle-Area	1	4 Choice Multiple Choice	2	1
Rectangle-Area	1	Fill in the Blank	3	1-2*0.1-1*1/3=0.46

3.2 Results

To evaluate how well the new model fits the data, we used the Root Mean Squared Error (RMSE) to examine the predictive performance on an unseen test set. Lower values for RMSE indicate better model fitting. There were randomly 2,313 student data in the test set and 3,297 students in the training set.

Table 3 shows the result of the comparison of the two different models, the original Knowledge Tracing(OKT) model and the Knowledge Tracing with partial credit(KTPC) model.

We compared the RMSE in predicting the partial credit performance and in predicting the traditional binary performance respectively. The Knowledge Tracing with partial credit model has lower RMSE value in both situations. The lower left column shows that KTPC does a great job in predicting partial credit scores, which is expected. The top left cell shows that OKT can do some reasonable job of predicting partial credit scores. The more interesting result is the right column, which shows that OKT has higher RMSE than the KTPC in predicting binary performances.

Table 3. Original KT (OKT) vs KT with partial credit (KTPC)

Model	**RMSE**	
	Partial Credit	**Binary Performance**
OKT	0.4128	0.4637
KTPC	0.2824	0.4572

We determined whether the difference between these two models is statistically reliable by computing the RMSE value for each student to account for the non-independence of student actions, and then compared these two models using a two tailed paired t-test.

The t-test *p* value of the RMSE between using the original Knowledge Tracing model and the Knowledge Tracing with partial credit model to predict the partial credit is 0. The *p* value between using the original Knowledge Tracing model and the Knowledge Tracing with partial credit model to predict the binary performance is $p<.001$. The degree of freedom of the t-test is 2,312 (since we are doing a student level t-test, the degree of freedom is the same as the number of students in the test set). Thus, the Knowledge Tracing with partial credit model is statistically reliably better at predicting student performance than the original Knowledge Tracing model.

4 Conclusions and Future Work

In this paper, we extended Bayesian Network student modeling to include continuous performance node. The effectiveness is demonstrated by incorporating a partial credit algorithm that assigns continuous performance given detailed student responses. Experiment results show that relaxing the assumption of binary correctness in student modeling can help improve predictions of student performance. This also proves that our intuition based heuristic for partial credit might be broadly applicable.

One topic we are interested in exploring is other partial credit schemes, for example, a method to refine the algorithm to generate partial credits that can better fit student data and more accurately infer student knowledge. Also, since we observed some abnormal parameters in the performance parameters (guess/slip), we are interested in finding out why the parameters are so different compare to normal Knowledge Tracing model.

5 Contributions

Moving from binary performance to continuous performance could make Intelligent Tutoring Systems more flexible. In this paper, on one hand, we extended the Knowledge Tracing framework to include a continuous performance node. This allows the Knowledge Tracing model to combine with all possible continuous performances such as essay score, speech recognition score. On the other hand, we presented an understandable and easy to refine algorithm to assign partial credit according to detailed student responses. This algorithm is one of many possible ways to convert student detailed responses into a continuous value.

The model presented in this paper enhanced student model accuracy by improving upon the classic Knowledge Tracing model. The result shows that the new model makes statistical reliably improvement in predicting both students' partial credit performances and binary performances. Also, freely available code is shared online, which could be useful for researchers that are trying to do the same task.

Acknowledgements. This research was made possible by the U.S. Department of Education, Institute of Education Science (IES) grants #R305K03140 and #R305A070440, the Office of Naval Research grant # N00014-03-1-0221, NSF CAREER award to Neil Heffernan, and the Spencer Foundation. All the opinions, findings, and conclusions expressed in this article are those of the authors, and do not reflect the views of any of the funders.

References

1. Arroyo, I., Cooper, D.G., Burleson, W., Woolf, B.P.: Bayesian Networks and Linear Regression Models of Students' Goals, Moods, and Emotions. In: Handbook of Educational Data Mining, pp. 323–338. CRC Press, Boca Raton (2010)
2. Attali, Y., Powers, D.: Immediate feedback and opportunity to revise answers to open-end questions. Educational and Psychological Measures 70(1), 22–35 (2010)
3. Baker, R.S.J.d., Corbett, A.T., Gowda, S.M., Wagner, A.Z., MacLaren, B.A., Kauffman, L.R., Mitchell, A.P., Giguere, S.: Contextual Slip and Prediction of Student Performance after Use of an Intelligent Tutor. In: De Bra, P., Kobsa, A., Chin, D. (eds.) UMAP 2010. LNCS, vol. 6075, pp. 52–63. Springer, Heidelberg (2010)
4. Corbett, A., Anderson, J.: Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. User Modeling and User-Adapted Interaction 4, 253–278 (1995)
5. Feng, M., Heffernan, N.: Can We Get Better Assessment from a Tutoring System Compared to Traditional Paper Testing? Can We Have Our Cake (Better Assessment) and Eat It too (Student Learning during the Test)? In: Aleven, V., Kay, J., Mostow, J. (eds.) ITS 2010, Part II. LNCS, vol. 6095, pp. 309–311. Springer, Heidelberg (2010)
6. Masters, G.N.: A rasch model for partial credit scoring. Psychometrica 47, 149–174 (1982)
7. Pardos, Z.A., Heffernan, N.T.: Modeling individualization in a bayesian networks implementation of knowledge tracing. In: De Bra, P., Kobsa, A., Chin, D. (eds.) UMAP 2010. LNCS, vol. 6075, pp. 255–266. Springer, Heidelberg (2010a)
8. Pardos, Z.A., Heffernan, N.T.: Navigating the parameter space of Bayesian Knowledge Tracing models: Visualization of the convergence of the Expectation Maximization algorithm. In: Proceedings of the 3rd International Conference on EDM (2010b)
9. Pavlik, P.I., Cen, H., Koedinger, K.: Performance Factors Analysis – A New Alternative to Knowledge. In: Proceedings of the 14th International Conference on Artificial Intelligence in Education, pp. 531–538 (2009)
10. Tang, K.L.: Polytomous item response theory (IRT) models and their applications in large-scale testing problems: Review of the literature. Educational Testing Service Technical Report (1996), `http://www.ets.org/Media/Research/pdf/RM-96-08.pdf`
11. Wang, Y., Heffernan, N.T., Beck, J.E.: Representing Student Performance with Partial Credit. In: Proceedings of the 3rd International Conference on Educational Data Mining, Pittsburgh, PA (2010)