Does Deep Knowledge Tracing Model Interactions Among Skills?

Shirly Montero*
Dept of Computer Science
University of Colorado Boulder
shmo8450@colorado.edu

Akshit Arora*
Dept of Computer Science
University of Colorado Boulder
akshit.arora@colorado.edu

Sean Kelly
Woot Math
Boulder, Colorado
sean.kelly@wootmath.com

Brent Milne
Woot Math
Boulder, Colorado
brent.milne@wootmath.com

Michael Mozer
Dept of Computer Science
University of Colorado Boulder
mozer@colorado.edu

ABSTRACT

Personalized learning environments requiring the elicitation of the knowledge state of the learner have inspired researchers to propose distinct models to understand that knowledge state. Recently, the spotlight has shone on comparisons between traditional, interpretable models such as Bayesian Knowledge Tracing (BKT) and its variants, and complex, opaque neural network models such as Deep Knowledge Tracing (DKT). Although the performance of these models can be similar, BKT can be at a distinct disadvantage relative to DKT for example when it comes to exploiting inter-skill similarities. In this study, we explore the superiority of DKT by building a version of DKT that is trained in exactly the same manner as BKT: sequences of a single skill are input, and a different DKT model is trained for each skill. We built a recurrent neural networks (RNNs), in particular a Long Short Term Memory (LSTM), to model a learner's knowledge state using a dataset of grade-school students mastering ten core math skills via an adaptive learning environment (WootMath). We trained the network in two ways, either presenting a single sequence for each learner that involved trials of many different skills, or a separate sequence for each learner and each skill. Combining predictions across skills for each model, we estimate the prediction quality via the AUC. The results indicate that the inductive bias of BKT could be helpful for the model's performance. The BKT model has relatively few parameters to be tuned and in this way well constrained by the data.

Keywords

Personalized learning, Online education, Knowledge tracing, Deep learning, Sequential modeling

*Denotes equal contribution by authors

1. INTRODUCTION

The optimization of the learning process is a recurrent topic in educational research, and an indispensable part for any effective lesson design. Traditionally, the strategy is to assess the learner's knowledge at any particular moment chosen by the tutor, and when needed, the tutor adjusts the learning activities to help the learner achieve her learning goals. Assuming the domain knowledge has been decomposed in a hierarchy of skills, the sequence of learning activities become a scaffold for the learner's learning process, helping the learner to acquire prerequisite skills before moving to the higher level skills in the hierarchy [1]. Therefore, in tailoring the sequence to the needs of the learner it is essential to track, assess, and predict the learner's changing knowledge state making her optimal sequence a personalized design. In reality, with limited educational resources the standardized lesson design is more the norm than the exception. Nevertheless, automated tutoring/self-study designs have presented an interesting attempt to personalize learning, and a more budget-friendly option in the long term. Therefore, optimal automated tutoring systems should be able to model the learners knowledge-state i.e. knowledge tracing [3], substituting the cues that a human tutor would use to assess the learner with the learners performance along the sequence of formative and summative learning activities. However, knowledge tracing and the evaluation of the personalized learning environments remain a complex endeavor and the focus of interest for applied machine learning research.

Recently, the spotlight has shone on comparisons between traditional, interpretable models such as Bayesian Knowledge Tracing (BKT) and its variants, and complex, opaque neural network models such as Deep Knowledge Tracing (DKT) [4, 5, 6, 7, 8, 9, 10]. A vanilla BKT is at a distinct disadvantage relative to DKT when it comes to exploiting inter-skill similarities, integrating recency effects, contextualize trials and represent variations on the learner's abilities. Therefore, DKT on balance outperforms vanilla BKT. Efforts have been made to show that when you add extra machinery to BKT, it rises in performance to a comparable level with DKT [5]. But little has been done to examine what factors are contributing to the superiority of DKT. In this study, we explore that superiority by build-

ing a version of DKT that is trained in exactly the same manner as BKT: sequences of a single skill are input, and a different DKT model is trained for each skill, we call this model a DKT single skill(DKTSS). In additon, we compare the DKTSS performance with a regular DKT trained with sequences of interleaved skills, that we call DKT combined skill (DKTCS).

In principle, a typical DKT should do well because it's more flexible. For example, BKT assumes that once a student learns they stay in the 'knowing' state. In contrast, DKT could model forgetting. BKT is Markovian: it embodies the input sequence history in a single state variable. In contrast, DKT can in principle remember the last N trials and condition its prediction on a more complex state representation.

1.1 Knowledge Tracing

An effective tutoring system possesses a mechanism to assess whether a learner has acquired a skill and how well the learner compares with an ideal model when applying that skill. In particular, knowledge tracing tracks the learner's changing knowledge state as it progresses along a sequence of learning activities[3]. The tutor system maintains an estimate of the probability that the learner has acquired each skill and based on history of outcomes it updates that estimate after each trial. Therefore, a model of the learner predicts her performance in the next trial for each step in the sequence. The primary data to build the model consists of a set of binary random variables representing the outcome of the trial for each learner $\{X_{lt}\}$ and the trial's label representing the trial itself or the needed skill to succeed in that trial $\{Y_{lt}\}$. In general, systems are much richer than just the primary data as they acquire other features e.g. time between trials, response time, use of supporting material or hints. The use of secondary data for refining the model is however, out of the scope of this study.

In BKT, the performance of a learner on a sequence of trials with same latent binary skill variable is modeled. Formally, BKT is a hidden Markov model that infers the value of the binary skill variable K_{si} for the sequence of responses for skill s, where i denotes the i-th trial in the sequence. The model depends on four parameters: $P(K_{s0} = 1)$, the probability that learner has mastered the skill before attempting the first trial; $P(X_{si} = 1|K_{si} = 0)$, the probability the learner transitions from the not-mastered to the mastered state after completing the i-th trial; the probability of correctly guessing the answer in the not-mastered state $P(Xsi = 1|Ksi = 0)$; and the probability of slipping or incorrectly answering when in a mastered state $P(Xsi = 0|Ksi = 1)$. In this form the model assumes no forgetting i.e. K cannot transition from 1 to 0 and each skill is treated independently so cross-skill dependencies are not modeled.

In a typical DKT [7], the skills are modeled combined. DKT is an approach that uses a recurrent neural network (RNN) "deep" in time, through a hidden layer of fully recurrent connections, that takes sequences of each learner's related tuples (X_{lt}, Y_{lt}) and returns a prediction of the outcome for each next trial. In particular, a Long Short-Term Memory (LSTM model) is a type of RNN that also contains combinations of gates that allow the network to learn remembering and forgetting information as needed. This makes LSTM

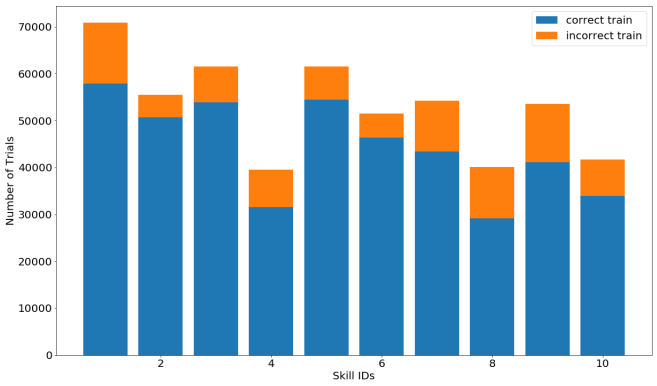

Figure 1: Distribution of the trials among the skills and the correctness of their outcomes used for training.

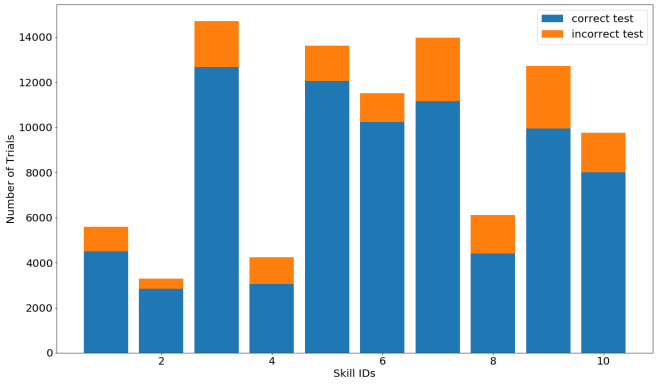

Figure 2: Distribution of the trials among the skills and the correctness of their outcomes used for evaluation.

intuitively suitable to handle predictions in a learner's sequential data. As pointed out elsewhere [5], RNNs can offer some advantages. For example, we can encode inter-skill similarity and the encoding can be easily modify to integrate secondary data when appropriate, both of which the traditional BKT model cannot handle. But it is the former property we are interested in exploring further.

2. DATA SET

The data used was collected by Woot Math (Boulder, CO), an adaptive learning environment for mathematics, which focuses on helping students in grades 3-8 master core math concepts, beginning with rational numbers. The supplemental software delivers a personalized progression of interleaved video instruction and scaffolded problems to mimic the natural give and take between a learner and a tutor. The content within the environment is divided in units. Each unit is a collection of lessons related to a specific area of a subject from the elementary mathematics curriculum e.g fractions. Further, each lesson comprises several sets of problems and

Table 1: AUC results for training and test data using the implemented models.

Model	Training Sequences	Evaluation Sequences	Between-skill AUC		Within-skill AUC	
			train	test	train	test
Baseline	per skill	per skill	0.614	0.603	0.500	0.500
BKT	per skill	per skill	0.725	0.727	0.686	0.698
DKTSS-5	per skill	per skill	0.596	0.589	0.592	0.581
DKTSS-10	per skill	per skill	0.593	0.587	0.592	0.582
DKTCS	combined skills	combined skills	0.861	0.843	-	-

instructional content that focus on a particular aspect of a unit. Ultimately, each problem set is coupled to a skill.

The dataset consists of anonymized data capturing the state of the learning platform when the learner interacted with a particular labeled exercise. Although more secondary data features are available, we selected only the following as the primary features: an identifier tag, which is an unique identifier for a skill, and the correctness of the answer of binary outcome of the interaction. In order to decrease sparsity, we limited our data set to those learners who had at least 50% of their trials within ten most popular skills completed. This selection rendered a set of 625,619 trials and 11,659 number of learners. For the training phase, we split the data 80:20 leaving 9,327 learners for training the models and 2,332 learners for evaluating it. The distribution of the trials among the ten skills and the correctness of their outcomes are shown in Figure 1 for the data used during training and Figure 2 for the data used during the evaluation. It is important to notice that the learning trajectories are adaptive, and as consequence different learners have different sequence lengths.

2.0.1 Encoding for Deep Learning

The input vectors were built as one-hot encoding of the combination of the skill to which the trial belongs and whether the trial was answered correctly. In particular, X_{lt} has a length of twenty bits where the first ten bits representing each of the ten skills are reserved to encode correctly answered trials and the last ten bits representing again each of the ten skills are reserved to encode incorrectly answered trials. The input vectors were fed into the models in the order given by the sequence followed by each learner. Similarly, the output vectors $\{\hat{Y}_{lt}\}$ have lengths equal to twice the number of skills, where each entry represents the probability that the learner acquired that skill i.e. would correctly answer the next trial, given the performance history of the learner. The prediction of any Y_{lt} can then be read from the entry in $\hat{Y}_{lt}$ corresponding to the particular skill.

3. METHODS

Our training task was to predict the learner's performance in the next trial based on the learner's historic performance in a sequence of exercises.

3.1 BKT Model

As explained above, in BKT each skill is separately modeled. The model optimizes the cost using the four parameters of BKT explained above, predicting the outcome for each trial in the sequence of trials.

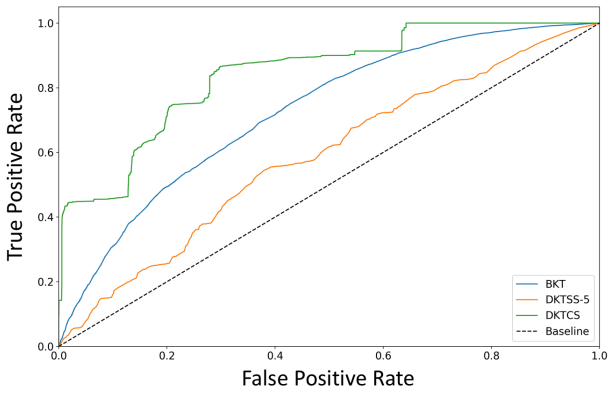

Figure 3: Evaluation of the models with a between-skill AUC.

3.2 DKT Model

For the DKT model, we implemented an off-the-shelf LSTM using the BasicLSTMCell from TensorFlow framework set to output the predictions for each next trial in the sequence. We used the AdamOptimizer and a hidden dimensionality of fifty for the DKTCS model with combined skills (ten skills), and five and ten hidden units respectively for the DKTSS-5 and DKTSS-10 models with separate skills (one skill per model).

3.3 Training

We trained the network in two ways, the DKTSS single-skill model, for which the input are sequences for each learner and each separate skill and the DKTCS combined-skill model presenting a single sequence for each learner that involved trials of interleaved skills. When evaluating the models with the testing data, the DKTSS models were presented with the data filtered for the particular skill. The DKTCS model was evaluated with the entire testing data with combined skills.

4. RESULTS AND DISCUSSION

We estimated the diagnostic quality of the models via the AUC. There are a two methods in which we can compute a single AUC value to facilitate the comparison of the models. One method is to compute first the AUC per skill and take the mean for all the skill. We named this the within-skill AUC. The other method is to compute one AUC for all the skills combined. We named this the between-skill AUC. In general, the between-skill AUC is larger than the within-skill AUC but weighing the computation on all trials uniformly, rewarding those that are more accurate. In addition, this

method allows for inferring the difficulty of one skill in comparison with others. The within-skill AUC weighs the skills uniformly; moreover, it disregards whether the accuracy on the prediction of the trials is the result of many correct trials in contrast than just a few correct trials. We summarize the results using both methods in Table 1.

DKTSS did not perform better than baseline or the BKT model when computing the between-skill AUC 3. Although, it does better than baseline when the within-skill AUC is computed instead. This result was consistent for both DKTSS-5 and DKTSS-10, which indicates that the problem is not in the architecture of the network not having enough free parameters to train. Note that even for the smallest network, five hidden units is 5x as much as the one state variable in BKT. In addition, the results show that there is no overfitting since test performance is well matched to training performance. Therefore, we believe that the DKTSS is having trouble learning the relative performance of different skills: it seems to be a calibration issue of one skill relative to another.

5. CONCLUSIONS

In this study, we attempted to understand the factors that boost the performance of the DKT when compared with a BKT. We built a DKT and trained it in exactly the same manner as BKT with sequences of a single skill, and a different DKT model is trained for each skill. The results indicate that the inductive bias of BKT could be helpful for the model's performance. The BKT model has relatively few parameters to be tuned and in this way well constrained by the data.

The typical BKT and DKT models differ in two aspects: (1) BKT is presented with sequences from a single skill; DKT is presented with interleaved skill sequences. (2) BKT is trained separately on each skill; DKT is trained on all skills at once.

Logically, these two respects are independent, and we could investigate the two differences separately.

6. ACKNOWLEDGMENTS

The authors wish to thank Dr. Mohammad Khajah for many useful discussions.

7. REFERENCES

[1] L. W. Anderson and D. R. Krathwohl, editors. *A Taxonomy for Learning, Teaching, and Assessing. A Revision of Bloom's Taxonomy of Educational Objectives*. Allyn & Bacon, New York, 2 edition, December 2001.

[2] T. Barnes, M. Chi, and M. Feng, editors. *Proceedings of the 9th International Conference on Educational Data Mining, EDM 2016, Raleigh, North Carolina, USA, June 29 - July 2, 2016*. International Educational Data Mining Society (IEDMS), 2016.

[3] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, Dec 1994.

[4] Y. Gong, J. E. Beck, and N. T. Heffernan. Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. In *Proceedings of the 10th International Conference on Intelligent Tutoring Systems - Volume Part I*, ITS'10, pages 35–44, Berlin, Heidelberg, 2010. Springer-Verlag.

[5] M. Khajah, R. V. Lindsey, and M. Mozer. How deep is knowledge tracing? In Barnes et al. [2].

[6] A. Lalwani and S. Agrawal. Few hundred parameters outperform few hundred thousand? pages 448–453, 2017.

[7] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 505–513, 2015.

[8] Y. Qiu, Y. Qi, H. Lu, Z. A. Pardos, and N. T. Heffernan. Does time matter? modeling the effect of time with bayesian knowledge tracing. In M. Pechenizkiy, T. Calders, C. Conati, S. Ventura, C. Romero, and J. C. Stamper, editors, *Proceedings of the 4th International Conference on Educational Data Mining, Eindhoven, The Netherlands, July 6-8, 2011*, pages 139–148. www.educationaldatamining.org, 2011.

[9] L. Wang, A. Sy, L. Liu, and C. Piech. Deep knowledge tracing on programming exercises. In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, L@S '17, pages 201–204, New York, NY, USA, 2017. ACM.

[10] Y. Zhang, R. Shah, and M. Chi. Deep learning + student modeling + clustering: a recipe for effective automatic short answer grading. In Barnes et al. [2], pages 562–567.