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 a student's knowledge state 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 complex, opaque neural network models such as Deep Knowledge Tracing (DKT). Although DKT appears to be a powerful predictive model, little effort has been expended to dissect the source of its strength. We begin with the observation that DKT differs from BKT along three dimensions: (1) DKT is a neural network with many free parameters, whereas BKT is a probabilistic model with few free parameters; (2) a single instance of DKT is used to model all skills in a domain, whereas a separate instance of BKT is constructed for each skill; and (3) the input to DKT interlaces practice from multiple skills, whereas the input to BKT is separated by skill. We tease apart these three dimensions by constructing versions of DKT which are trained on single skills and which are trained on sequences separated by skill. Exploration of three data sets reveals that dimensions (1) and (3) are critical; dimension (2) is not. Our investigation gives us insight into the structural regularities in the data that DKT is able to exploit but that BKT cannot.

Keywords

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

1. INTRODUCTION

*Denotes equal contribution by authors

The optimization of the human learning is a recurring topic in educational research. Traditional human instructors monitor and assess a student's knowledge and adapt instructional activities to help the student achieve her goals. Assuming the knowledge in a domain has been decomposed in a hierarchy of skills, the sequence of learning activities becomes a scaffold for the learning process, helping the student to acquire prerequisite skills before moving to more complex skills in the hierarchy [1]. Therefore, in tailoring the sequence to the needs of the student it is essential to track, assess, and predict the student's changing knowledge state, thereby personalizing the design. In reality, with limited educational resources a 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 offer a more budget-friendly option in the long term. To be effective, automated tutoring systems should model the student's knowledge state, known as *knowledge tracing* [3], substituting the cues that a human instructor would use to assess the student with the student's 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.

1.1 Knowledge Tracing

A knowledge-tracing model tracks a student's evolving knowledge state as the student practices a sequence of problems [3]. The knowledge state is decomposed into a set of domain *skills* required to solve the specific problems that the student is attempting. Each problem is labeled with the corresponding skill required for that problem. The critical data to be modeled thus consist of a sequence of pairs, $\mathcal{D}_q = \{\ldots (X_{qt}, Y_{qt}) \ldots\}$, where X_{qt} is a categorical random variable indicating the specific skill required to be able to solve the problem presented to student q on trial t, and Y_{qt} is a binary random variable denoting the outcome of the trial, with $Y_{qt} \in \{\mathit{correct}, \mathit{error}\}$. Of course, most modern data sets have far richer information—the use of supporting materials or hints, response latencies, time between trials, number of attempts, the specific problems being attempted, etc. For the present research, we are not considering these additional sources of data.

In Bayesian knowledge tracing (BKT), the data are partitioned by skill, leading to a skill specific dataset,

$$\mathcal{D}_{qs} = \{(X_{qt}, Y_{qt}) | X_{qt} = s\}$$

in which the trial sequence is re-indexed for each skill s. BKT is a hidden Markov model that performs inference to determine a latent binary skill variable K_{qst}, denoting the knowledge state of student q on skill s at the start of trial t. The model for skill s has 4 parameters [5], $\boldsymbol{\theta}_s$, with the following interpretations in terms of the model:

$$\begin{aligned}\boldsymbol{\theta}_s = &\{P(K_{qs0}) = 1, \quad P(Y_{qt} = 1 | K_{qst} = 0),\\ &1 - P(Y_{qt} = 0 | Kqst = 1), P(K_{qst} = 1 | K_{q,s,t-1} = 0\}.\end{aligned}$$

In this form the model assumes no forgetting, i.e., the knowledge state K cannot transition from 1 to 0. Note that each skill is treated independently; cross-skill interactions are not modeled.

DKT [5, 7] is a recurrent neural network whose input layer is a representation of the previous trial, $(X_{q,t-1}, Y_{q,t-1})$ and whose output layer is a prediction, for every possible skill, of whether the student would answer problems of that skill correctly, i.e., $\forall s$, $P(Y_{q,t} | X_{q,t} = s)$.[1] Internally, DKT has a layer of recurrent hidden units that, through training, learn to hold the student's knowledge state in order to make predictions. Typically, the hidden layer contains LSTM units, often used to handle sequence processing tasks because of their ability to maintain state over time.

As originally implemented, DKT makes three assumptions that distinguish it from BKT:

1. All skills are interleaved in a single sequence over time, and predictions are made for each trial in the sequence. In contrast, BKT assumes that skills are presented in separate sequences. We will refer to this distinction as *combined sequence* (*CS*) versus *separate sequences* (*SS*).

2. All skills are learned by a single model that combines information across skills. In contrast, BKT assumes that a separate model is trained on each skill, and thus the parameters for different skills do not interact. We refer to this distinction as *combined model* (*CM*) versus *separate models* (*SM*).

3. DKT is of course based on a neural network, whereas BKT is a probabilistic model. The neural network has far greater flexibility. For example, BKT assumes that once a student learns they stay in the 'knowing' state. In contrast, DKT can model forgetting. To illustrate another difference, DKT can in principle remember the last n trials and condition its prediction on this complex state representation, whereas BKT is Markovian—it embodies the input history in a single binary state variable.

[1]The inputs and outputs of DKT can be representations of either *skills* or *problems*. For example, DKT could represent $4+3$ and $7+2$ as two distinct problems or it could represent them as the skill *single-digit addition*. Because BKT operates with the level of representation being skills and we wish to compare DKT to BKT, our implementation of DKT does the same: its representation of the current trial is a skill index and the correctness of the response; its representation of the output is one prediction per skill index.

Assumption 1 is conditioned on assumptions 2 and 3; assumption 2 is conditioned on assumption 3. Our goal is to tease apart these assumptions and examine them individually, allowing us to determine which assumptions are most responsible for the improvements in performance that DKT achieves over BKT. In addition to the standard form of BKT and DKT, we introduce two new variants of DKT: one that drops assumption 1, and one that drops assumptions 1 and 2. For the sake of understanding the relationship among the four models, we relabel the standard forms of BKT and DKT, obtaining the following progression of models:

- **DKT-CM-CS**: The standard form of DKT, which is a single neural network that learns all skills (the combined model or CM) and its input sequence consists of the interlaced sequence of trials across all skills (the combined sequence or CS). This model incorporates assumptions 1-3.
- **DKT-CM-SS**: DKT minus assumption 1. This variant is trained on a separate sequence for each skill. A single model is still used to predict for all skills (the combined model or CM) but the input is separated by skill (the separate sequences or SS).
- **DKT-SM-SS**: DKT minus assumptions 1 and 2. This variant trains a different model for each skill (separate models or SM) and because each skill is fed into a different model, it is necessary to separate the sequences by skill (SS).
- **BKT-SM-SS**: The standard form of BKT. We augment the name with SM to remind the reader that a separate instantiation of the model is constructed for each skill, and with SS to indicate that sequences are separated by skill and fed into the corresponding model. This model drops all three assumptions of DKT.

Pairwise comparisons among models allow us to examine individual assumptions: DKT-CM-CS and DKT-CM-SS differ only in assumption 1; DKT-CM-SS and DKT-SM-SS differ only in assumption 2; and DKT-SM-SS and BKT-SM-SS differ only in assumption 3. By examining the performance differences between each pair, we can determine the value of each assumption.

1.2 Related Work

Recent studies compare traditional models such as Bayesian Knowledge Tracing (BKT) and its variants against complex neural network models such as Deep Knowledge Tracing (DKT) [4, 5, 6, 7, 8, 10, 11, 12]. The basic BKT (or BKT-SM-SS) is at a distinct disadvantage relative to the standard DKT (or DKT-CM-CS) when it comes to exploiting inter-skill similarities, integrating recency effects, contextualizing trials and representing variations on the student's abilities. Therefore, DKT on balance outperforms basic BKT. Efforts have been made to show that when additional machinery is added 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.

2. METHODOLOGY

2.1 Data sets

We examined three data sets which vary in the number of students and the number of skills. Two data sets, ASSISTments 09-10(b) and KDD Cup 2010, are well studied in the educational data mining literature. The ASSISTments data set is generated from an online grade school mathematics tutor. The 09-10(b) version of the data were cleaned by Xiong et al. [11] to remove repeated multiple skill problems which, in the original data base, were duplicated for each component skill, and when left in the data set give an advantage to DKT over BKT. ASSISTments 09-10(b) consists of 4217 students and 124 skills. The KDD Cup 2010 data is from the 2005-2006 Cognitive Algebra Tutor [9]. These data consist of 574 students and 100 skills. Both data sets were obtained from a GitHub repository of one of the authors [13]. The data in the repository are divided into a training and a test set.

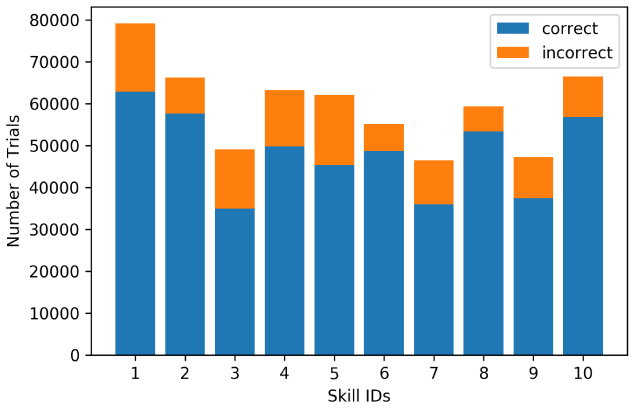

Figure 1: Example distribution of the trials in Woot Math dataset among the skills and the correctness of their outcomes used for training.

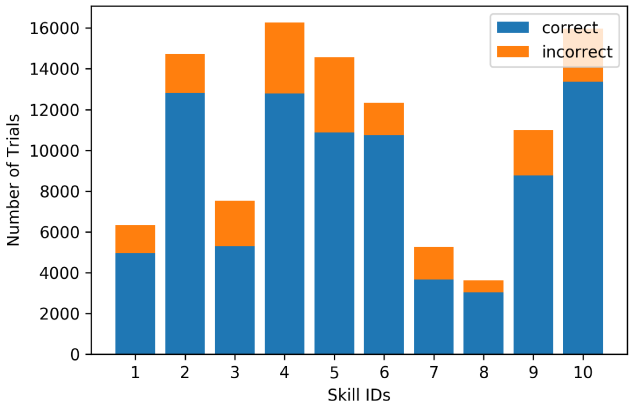

Figure 2: Example distribution of the trials in Woot Math dataset among the skills and the correctness of their outcomes used for evaluation.

The third data set was collected by Woot Math, a Boulder Colorado start up that develops adaptive learning environments for mathematics. The focus is on helping students in grades 3-8 master core math concepts, beginning with rational numbers. The Woot Math software delivers a personalized progression of interleaved video instruction and scaffolded problems to mimic the natural give-and-take between a student 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 instructional content that focus on a particular aspect of a unit. Ultimately, each problem set is coupled to a skill. It is worth noting that the learning trajectories are adaptive, and as consequence different students have different numbers of trials in a lesson.

That dataset consists of anonymized data capturing the state of the learning platform when the student 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 students who had at least 50% of their trials within ten most popular skills completed. This selection rendered a set of 625,619 trials from 11,659 students, with exercises drawn from among 10 skills. The data were split by student to obtain an 80:20 ratio of training to testing examples (9,327 students for training, 2,332 students for testing). The distributions of trials among the ten skills and the correctness of their outcomes are shown in Figure 1 for data used during training and Figure 2 for data used during the evaluation.

2.2 Data encoding

For DKT, input vectors are a one-hot encoding of the previous trial, specified by the conjunction of (a) the skill to which the trial belonged, and (b) whether or not the trial was answered correctly. Thus, if there are n_s skills in total, then the input vector has $2n_s$ units, exactly one of which would be turned on for any input. The input vectors are fed into the models in a sequence sorted by the temporal order of the trials. The output from DKT is a vector with n_s elements, each element s being the model's estimate of the probability that the student had acquired skill s given the performance history of the student. This probability also specifies how likely the student will be to answer the next trial correctly if it is a problem requiring skill s.

2.3 Model implementation

To implement DKT methods, we modified the source code used by Xiong et al. [11], as obtained from one of the author's GitHub repository [13]. The modifications pertained the way the data were to be fed to the model when single skill sequences were used. For DKT, we ran five replications of training the neural net with different random weight initializations each replication. The same training and test split was used for all five replications.

For initializing weights in all the DKT methods, we used random uniform weights in the range $[-.05, +.05]$. All DKT models had a single hidden layer. DKT-SM-SS used 10 LSTM units for all datasets. For DKT-CM-SS and DKT-CM-CS, we used one hidden layer with 50 LSTM units for the Woot Math dataset and 200 LSTM units for the other two datasets, due to the fact that they contain more skills. Additionally, for all DKT models, we used drop-out on the hidden layer with keep probability of 0.6.

Rather than run BKT on ASSISTments and KDD, we report the results from Xiong et al. [11]. Our own implementation of BKT was used to obtain performance estimates for the Woot Math data.

3. RESULTS

We estimate the discriminative performance of each model—its ability to predict when a student will answer correctly or incorrectly—using the signal detection AUC (area under the curve) measure. There are a two methods by which AUC can be computed. One method, within-skill AUC, involves separating the test data for all students by skill and computing an AUC value for each skill and then computing the mean across skills. The other method, between-skill AUC, involves combining data from all students and all skills and computing a single AUC score. In general, the between-skill AUC is larger than the within-skill AUC for two reasons. First, it incorporates the degree to which models are successful at predicting relative performance among skills. Second, the between-skill AUC weighs all trials equally, whereas the within-skill AUC de-emphasizes skills with many trials. In our work, we compute between-skill AUC, both because it is sensitive to aspects of the data we care about and it matches the methodology used by Xiong et al. [11].

Table 1 shows a summary of results for the three data sets (rows of the table) and the four models (columns 4-7 of the table). From left to right, BKT-SM-SS is the basic BKT model, for which a separate model is trained per skill and the sequences are separated by skill. DKT-SM-SS is an implementation of DKT in which a separate model is constructed for each skill and the sequences are separated by skill; this procedure is analogous to the manner in which BKT is trained, except the model is a neural network instead. DKT-CM-SS involves a single combined model trained on all skills, but the sequences fed to the model are separated by skill. Finally, DKT-CM-CS is the standard implementation of DKT in which a combined model is trained on all skills and the input sequences combine skills to obtain an interleaved trial history.

3.1 Interleaved- vs. blocked-skill sequences

DKT-CM-CS and DKT-CM-SS differ only in the manner in which the student sequences are parsed. The combined sequences interleave various skills; the separate sequences are blocked or filtered by skill. For example, 1-3-3-2-2-1-1-2-3 is an interleaved sequence, and {1-1-1, 3-3-3, 2-2-2} are the set of blocked sequences. In both cases, the sequence order corresponds to temporal order of the trials. Our results show a win for DKT-CM-CS for ASSISTments and KDD. In these cases, DKT is able to leverage the interaction among skills. One likely form of interaction that the model exploits is the fact that strong students perform well on all skills, weak students perform more poorly on all skills. Consequently, there should be an inter-skill correlation for a given student. To elaborate, consider the sequence of trials with two skills, 1-1-1-2-2-2. If the student performs extraordinarily well on the 1-1-1 sequence, this observation should be predictive of better-than-average performance on 2-2-2. We suspect that adding IRT-like student ability parameters to DKT might eliminate the difference between the combined- and separate-sequence versions of DKT.

For the Woot Math data set, there was no benefit to combining. We hypothesize that the reason for this finding is that there are only 10 skills, and the breakdown by skill is fairly coarse. Because the skills have little in common, there is less likely to be transfer from one skill to another, and therefore predicting performance on one skill would not benefit from knowing performance on another skill. (Similarly, you wouldn't expect, say, someone's driving ability to predict their juggling ability.)

3.2 Combined-skill vs. separated-skill models

Both DKT-CM-SS and DKT-SM-SS are trained on sequences blocked by skill. They differ in that DKT-CM-SS is trained on all skills at once. Thus its parameters are shared across skills. In contrast, a separate instance of DKT-SM-SS is trained for each skill. Thus, its parameters are not shared across skills. In both cases, AUCs are computed by pooling data across skills and computing a single AUC—the between-skill AUC we referred to earlier.

We do not observe a significant difference in performance between DKT-CM-SS and DKT-SM-SS. On KDD they perform almost identically. On ASSISTments, DKT-SM-SS does slightly better. And on Woot Math, DKT-CM-SS does slightly better. In principle, training a combined model on all skills will be beneficial if different skills are learned in a similar fashion, i.e., if the time course of learning skill s_1 is related to the time course of learning skill s_2. When there is similarity across skills, there can be inter-skill transfer in modeling the temporal dynamics of learning. However, the benefit of this transfer should diminish as data sets get larger. With a large enough data set for skill s_1, the weak inductive bias of s_2 provides little benefit. We suspect that the reason for observing no benefit by training a single model on all skills is that our data sets are relatively large. It is possible on much smaller data sets, we would observe a benefit of using data from skill s_1 to constrain predictions on skill s_2.

3.3 Neural network vs probabilistic model

DKT-SM-SS and BKT-SM-SS are trained in exactly the same way: each model has distinct parameters for each skill, and data from one skill is not used to inform performance on other skills. The models differ in that DKT-SM-SS is an intrinsically flexible neural network with hundreds of parameters, whereas BKT-SM-SS has 4 parameters. By restricting our neural network to model only single skills we are taking out of the equation the possibility of exploiting inter-skill similarities, leveling the playing field for the more restricted BKT model. Nonetheless, the results indicate better performance of the neural net than the probabilistic model on all three data sets. This is consistent with the neural net being more flexible in characterizing the time course of learning.

Table 1: Test set performance (AUC) for four models. Standard deviations (N =5) are in parenthesis.

Dataset	# Students	# Skills	BKT-SM-SS	DKT-SM-SS	DKT-CM-SS	DKT-CM-CS
ASSISTments 09-10(b)	4217	124	0.630	0.733 (0.0003)	0.726 (0.0008)	0.809 (0.0021)
KDD	574	100	0.620	0.771 (0.0003)	0.764 (0.0013)	0.818 (0.0025)
Woot Math	11659	10	0.727	0.745 (0.0007)	0.760 (0.0005)	0.745 (0.0032)

BKT-SM-SS embodies a strongly restricted model of learning. For example, BKT-SM-SS assumes that the probability of learning on trial t_1 is identical to the probability of learning on t_2, for any t_1 and t_2. In contrast, DKT-SM-SS might discover that if a student does not learn early on, they are not likely to learn later on.

4. CONCLUSIONS

Our goal in this research is to understand the factors that contribute to the strong performance of DKT. We explored three factors that differentiate DKT and BKT, and we developed a continuum of 4 models which, when paired, allowed us to evaluate one factor at a time. Our three key findings are as follows. First, DKT benefits from being presented with a sequence of interleaved skills. We hypothesize that this benefit is due to being able to estimate strength of a student based on their performance on one skill and then use this estimate to predict performance on another skill. Second, DKT does not benefit per se by learning about multiple skills at once versus learning about a single skill. We speculate that the reason for this finding is that we have relatively large data sets, and the inductive bias provided by one skill offers little leverage in modeling other skills. Third, DKT shows a large benefit by being a flexible model that does not incorporate a strong theory of human learning, as does BKT. This is perhaps our most significant finding, as it suggests that the simple all-or-none learning-without-forgetting theory that BKT posits is too simplistic.

5. ACKNOWLEDGMENTS

The authors wish to thank Dr. Mohammad Khajah for many useful discussions. This research was supported by the National Science Foundation awards EHR-1631428 and SES-1461535.

6. 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? In X. Hu, T. Barnes, A. Hershkovitz, and L. Paquette, editors, *Proceedings of the 10th International Conference on Educational Data Mining*, EDM '17, 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] J. Stamper, A. Niculescu-Mizil, S. Ritter, G. Gordon, and K. Koedinger. Algebra i 2005-2006. Challenge data set from kdd cup 2010 educational data mining challenge. Find it at http://pslcdatashop.web.cmu.edu/kddcup/downloads.jsp, 2010.

[10] 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.

[11] X. Xiong, S. Zhao, E. V. Inwegen, and J. Beck. Going deeper with deep knowledge tracing. In Barnes et al. [2], pages 545–550.

[12] 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.

[13] S. Zhao. 2016-edm. https://github.com/siyuanzhao/2016-edm.