Deep Knowledge Tracing on Fractions

Shirly Montero Quesada
shirly.monteroquesada@colorado.edu

Akshit Arora
akshit.arora@colorado.edu

Sean Kelly
sean.kelly@wootmath.com

1 Abstract

Human learning is a complex process and its efficiency depends on many factors. However, understanding the knowledge state of the learner has been traditionally one of the common interests of researchers in this area. We implemented a vanilla recurrent neural network(RNN) with logistic units and a Long Short Term Memory (LSTM) to model the learner's knowledge state by predicting the probability that a learner would correctly answer a problem in our set given a sequence of prior responses. Both of our models performed better than a simple predictor based on the frequency of correctness in our dataset.

2 Introduction

A recurrent topic in educational research is the optimization of the learning process, usually an indispensable part for any effective lesson design. Traditionally, the strategy is to assess the learner's knowledge at any particular moment that the tutor chooses to evaluate the design, and when needed, the tutor adjusts the learning activities to help the learner achieve her learning goals. The sequence of activities become a scaffolding for the learner's learning process making her optimal sequence a personalized design. In reality, there are limited educational resources, and standardized lesson designs are more the norm than the exception. Nevertheless, automated tutoring/self-study designs have presented for a couple of decades an interesting and somewhat cheaper-in-the-long-term attempts to personalize learning. Therefore, it is of general interest to be able to model the learners knowledge-state also known as knowledge tracing, substituting the verbal/non-verbal cues a human tutor would use to assess the learner with the learners performance during the sequence of formative (and eventually summative) activities. Yet, knowledge tracing remains a difficult job but the focus of interest for applied machine learning research. In this work, we explore an RNN and a LSTM to model student learning.

3 Related work

Some of the approaches used earlier to model the learner are Bayesian Knowledge Tracing (BKT), Deep Knowledge Tracing (DKT), Partially Observable Markov Decision Processes (POMDPs), and Performance Factor Analysis (PFA) . Piech et al. used a vanilla RNNs with sigmoid units and in particular LSTM [2, 3] for DKT, and demonstrated better performance than the traditional Bayesian knowledge tracing approach [1].

4 Approach

4.1 Data

The database used was facilitated by Wootmath (Boulder, CO) and consists of anonymized data capturing the state of the learning platform when the learner interacted with a particular labeled question. The questions are part of learning tasks which in turn are part of units that cover topics from the elementary mathematics curriculum e.g fractions. The labels are the common core state standards (CCSSM) assigned to each of the tasks. Although more data features are available, initially, we used the following:

- question's identifier. An unique identifier for a question. Since the learning trajectories are adaptive, different learners have different sequence lengths. We limited our data set to those learners with at least 50% of the 10 most popular tasks completed. This selection reduced the number of questions to 612 and the number of labels to four. In addition, the questions can be presented in a different relative order, therefore we chose to feed the entire sequences to the network.

- correctness of the answer (incorrect/correct). Since the learner can have two tries, we encoded these states as ternary vector of 612 in length with values 0.3, 0.66, 1 representing incorrect, correct second try and correct first try respectively and the index representing the question's identifier.

- question's common core state standards (CCSSM) label. These were encoded as the correctness i.e. a ternary vector of length four with values 0.3, 0.66, 1 representing incorrect, correct second try and correct first try respectively and the index representing the question's CCSSM label. This vector was concatenated with the correctness to form the input vector of length 616.

The input vectors were feed into the network in sequences matching each learner's trajectory, padding when necessary for a fixed-length input sequence. As output we had a prediction across the questions and CCSSM labels in the dataset from which we can extract the probability that the learner would correctly answer the last question of the sequence.

4.2 RNN Model

Architecture

We implemented a vanilla RNN with a logistic activation using the BasicRNNCell from TensorFlow, as depicted in Figure 1. The RNN had 616 hidden units, which is the size of our input vectors. In addition, we also ran an LSTM using BasicLSTMCell from Tensorflow using the same number of hidden units.

Training

Our dataset was produced by sampling the Wootmath database as decribed above. That operation rendered 1255 students. We split the data 90:10 leaving 1130 for training and 125 students for testing. As introduced above, the input vectors were feed into the RNN as part of sequence and per step we obtain a prediction across the questions and CCSSM labels as a probability that the learner would correctly answer any particular question in the dataset (any of the 612). In the forward propagation, we used the sigmoid_cross_entropy_with_logits() from tensor flow to compute the error per input vector and used the reduce_mean() at the end of the sequence to back propagate the error on the output (pooled with last question prediction's error). For optimization we used two shelf-optimizers: MomentumOptimizer and AdagradOptimizer form TensorFlow (learning rates 0.1 and 1).

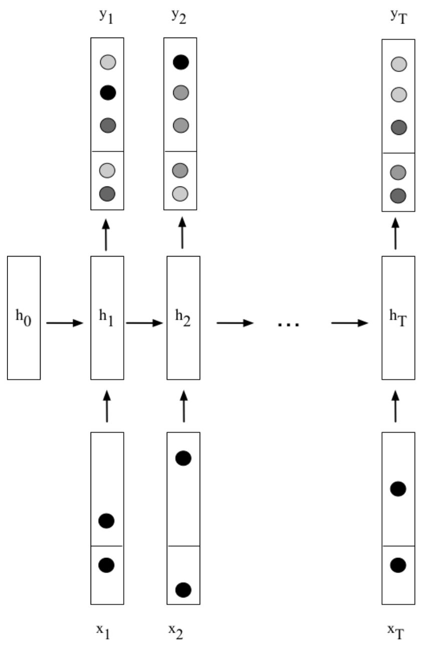

Figure 1: A representation of our recurrent neural network modified from [2]. The input vectors are a concatenation of 612 questions and four standard labels for learner's skills. The output vectors are the probability of correctness across the set of questions and labels.

	Stupid predictor (frequency of correctness)	DKT - RNN MomentumOptimizer	DKT-LSTM AdagradOptimizer
question level	0.564	0.625 (0.117)	0.655 (0.049)
CCSSM label level	0.621	0.994 (0.009)	0.997 (0.002)

Table 1: AUC results for the implemented models, standard deviations s = 3 in parenthesis.

4.3 Results and Discussion

The area under the curve (AUC) was used to evaluate the performance of our model. For a baseline, we computed a stupid predictor by using

- the observed correctness frequency for the last question on any particular sequence across the dataset (AUC = 0.564)
- the observed correctness frequency for the last CCSSM label on any particular sequence across the dataset (AUC = 0.621)

The vanilla RNN performed similarly to the LSTM, however, the LSTM render more robust results as shown by the standard deviation on Table 1. Both models seem to be better than the stupid predictor used for the baseline.

5 FUTURE WORK

As presented here, our models were a vanilla RNN and an LSTM with a ternary input vector. We used two different optimizers and tuned the nets with a few parameter. However, more tuning can be achieved with other techniques e.g. dropout. In addition, a follow up of this work would be to determine the added value of the ternary feature encoding the state of a correct answer on the second try. The effect of the second try can be measured by running the model with binary inputs, that is, if a question was correctly answer regardless of the try or if the answer was incorrect even after the second try.

Our input vector was the result of concatenating the questions and the CCSSM labels. That vector can be expanded to include other features from the fields available:

- The time stamp could be used as a good predictor that counts for memory effects if there is long times in between exercises.
- The time duration could map to a students hesitation, which in turn could be a consequence of a weak foundation on the particular skill.
- Information about previously completed units could point to skill relations that have not been annotated.
- More of the structured data gathered regarding the students interactions with the application can provide some more predictive power e.g. if and how they modeled the answer the provided.

Ideally, we would like to identify the fields and combinations thereof that produce an improvement on the prediction.

ACKNOWLEDGMENTS

The authors wish to thank Dr. Mohammad Khajah, Dr. Michael Mozer, and Dr. Brent Milne.

REFERENCES

[1] M. Khajah, R. V. Lindsey, and M. C. Mozer. How deep is knowledge tracing? 04 2016.

[2] 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, eds., *Advances in Neural Information Processing Systems 28*, pp. 505–513. Curran Associates, Inc., 2015.

[3] 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, pp. 201–204. ACM, New York, NY, USA, 2017. doi: 10.1145/3051457.3053985