CS230

Deep Learning Knowledge Tracing for Designing Adaptive Assessments for the GMAT

Chris Wang
Department of Computer Science
Stanford University
chrwang@stanford.edu
Mentored by Sherry Ruan, Alex Kolchinski

Abstract

With increased software for education, being able to adapt assessments for individual students has become increasingly important. When presented with a dataset of students, each with sequences of questions and answers, we implement an RNN with LSTM to predict each student's performance on our entire set of questions, especially question they have not seen before. Neural networks can help capture latent elements of the student's knowledge in layers, and thus make better predictions regarding their ability.

1 Introduction

Being able to model student knowledge and ability based upon performance on assessments can help make predictions regarding student performance and develop curriculum that adapts to each individual student, creating interesting new innovations in the education space. For our model, we focus on multiple choice questions with a simple correct or incorrect response for each student. Our goal is to predict whether a student will answer any given question within our defined set correctly given a sequence of questions the student has answered thus far. Our approach uses deep knowledge tracing using a recurrent neural network (RNN) with LSTM, and applied our model to a new data set of student GMAT assessment data to analyze its performance.

2 Related work

Current knowledge tracing models are divided into Bayesian Knowledge Tracing Models (BKT), and Deep Knowledge Tracing Models. Bayesian tracing models traditionally models student mastery with a Hidden Markov Model to update latent variables each time a student answers a question or demonstrates the skill in question. This model does a good job of storing student performance, but depends on explicit topics for each question and misses out on relationships between topics and concepts. Piech et al. introduces a deep learning knowledge (DKT) model that applies recurrent neural networks and long short-term memory to this same problem. His team applies a simple version of this model to student data sets, and achieves improved results. We continue to build upon this model by incorporating additional assessment data and playing around with topics on a new data set.

3 Dataset and Features

The data set of GRE student information is provided by TAL Education. It associates student IDs with question IDs, answer choices, time taken to answer a question, and the time and data each question

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

the question was answered. In total, we had over 8,000,000 data points, each of which was a student and question pair.

To process the data set into a usable CSV file, we extracted the important information for each question. The sequences of question and answer pairs for each individual is extracted. We also tried including more information about the concepts covered in each question to get a broader understanding of each student's skill-set by replacing the question id with an encoding based .

Our data set was split 75 percent for the training set and 25 percent for the test set. The reason why we wanted to maintain this amount for the test set was to ensure that we had a sizable number of longer sequences to test our model on, with varying students and performance levels. Although we had 8 million data points, this actually only yielded a few thousand student sequences.

Below, this figure is am example of the data we used, before processing:

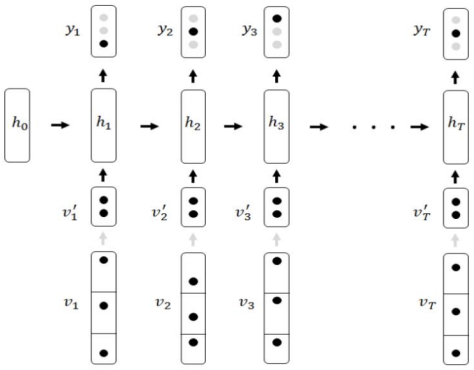
user id	practice id	question id	timestamp	correct or no	practice moc	user answer ID
1310344	35938_44_2(	1	1500954942	0	1	["26537"]
1310344	35940_44_2(	2	1500955028	0	1	["26531"]
1310344	35941_44_2(	5534	1500955075	1	1	["1061"]
1310344	35943_44_2(	4	1500955121	0	1	["26526"]
1310344	35944_44_2(	5	1500955195	0	1	["26522"]
1310344	35946_44_2(	6	1500955310	0	1	["26518"]
1310344	35947_44_2(	5539	1500955395	1	1	["1046"]
1310344	35970_44_2(	8	1500955504	0	1	["26512"]

4 Methods

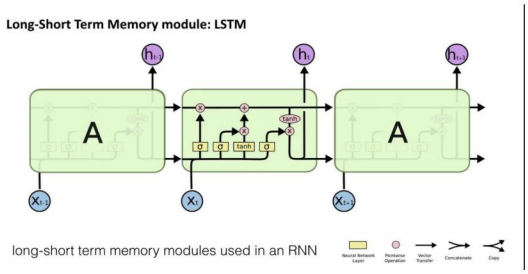
The model used was a DKT model using an RNN with LSTM. LSTM features "gates" to prevent against vanishing and exploding gradients. The processed question data was encoded into a question id for each question. We attempted our model with two different methods: one with solely question ids, and one with topics for each question encoded, which replaced the question ids. The goal of the latter method was to capture similarity between problems and prevent our tensors from being too sparse since we did not have complete coverage of all questions, although this reduced the size of the data set significantly because not all questions were labeled with topics.

In either case, the model takes an input sequence of time series vectors and maps them to an output sequence of time series vectors through hidden states which store information from the preceding data. The latent units store values until cleared by a forget gate in an LSTM.

At each time step for a student, an encoded question and answer vector is passed into the sequence, which has stored the information from earlier encodings in a hidden state. At each time step, an output vector is also produced which represents that student's predicated probabilities for answering each question in our set of question correctly, given the information we have had up to that time step.



This first diagram represents a simple RNN for a time series. Notice the encoded vectors, v' going in at each time step, while h represents the hidden states and y represents the output at each time step.



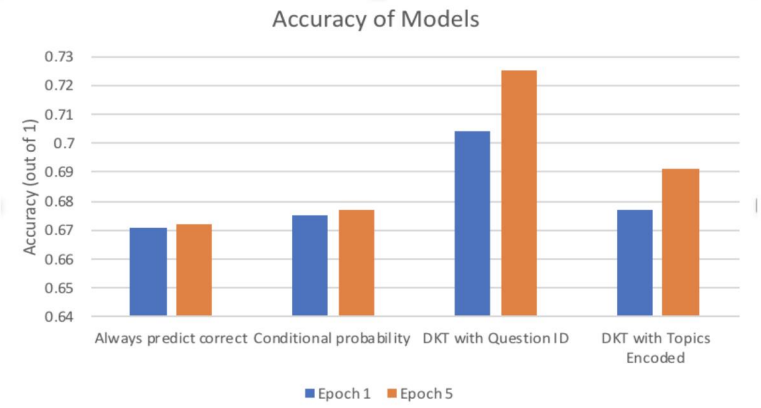
Zooming in to a particular time step, notice the specific hyperparameters and how they relate to each other in producing the output vector and next hidden state.

5 Experiments/Results/Discussion

For my hyperparameters, the goal was to achieve a balance between reasonable training time on the GPU and accurate results. The batch size was chosen to be 32, while the learning rate was chosen to be 0.0005, as this was found to be a good balance between convergence and reasonable amount of time needed. The one hyperparameter that was tuned a lot was maximum sequence length. Beyond this length, the sequence would be truncated. Truncating too early would mean effectively eliminating any meaningful result achieved, while truncating too late would result in sparse tensors (not all sequences had similar lengths) and very lengthy training periods.

We measured the performance of out model through accuracy and AUC. We compared the results of our model with the baseline models of always predicting correct and a baseline conditional probability model. The below table and graph depicts a summary of our results, with a focus on the test set.

	Epoch 1, Test		Epoch 5, Test		Training
Method	Accuracy	AUC	Accuracy	AUC	Accuracy
Always predict correct	0.671	N/A	0.672	N/A	0.67
Conditional probability	0.675	0.8054	0.677	0.811	0.687
DKT with Question ID	0.7043	0.8343	0.7254	0.844	0.9834
DKT with Topics Encoded	0.6772	0.8192	0.691	0.8225	0.9651



A major challenge encountered throughout this project was the sparsity of questions. In other words, there were a large amount of data points, but a lot of them were on different questions. This made our tensors very sparse. A potential solution was using the topic data, since each question had multiple topics, many of which overlapped. However, this actually resulted in the opposite effect because it reduced the number of data points we had significantly.

Overall, the results we were able to achieve had slightly higher than a baseline conditional probability model. This was not as high as anticipated, but due to the factors of data sparsity and missing topics, seemed to be as high as possible given our data set. There is certainly lots of inherent unpredictability in determining whether a student will answer a question correctly. Compared to other models, another

reason why our model does not perform significantly above baseline is that in absolute terms, our baseline is already at 67 percent. Regardless of baseline, it may be harder to improve accuracy as the baseline gets closer to 100 percent, or all questions being answered correctly.

6 Conclusion/Future Work

A more complete data set with data labels and additional features for each question, as well as a sophisticated way to encode them into vectors (which would necessitate a new architecture for input into our model) may yield more accurate results. Natural language processing can be used for the questions and answers to better gauge question characteristics without needing labels. This could also assist the application of the model to non-binary tasks, such as open ended question answering or even programming assignments.

Beyond improvements to the model itself, the results from such a model can be applied to help create adaptive assessments for students. Based on predictions for student knowledge, we can make recommendations that best improve the student's overall likelihood of solving a randomly selected question correctly.

7 Code

A sample of the code repository is at: https://github.com/chriswang1999/adaptiveassessment. The data is not in the repository, we would prefer not to share it.

8 Acknowledgements

I would like to acknowledge Sherry Ruan and Alex Kolchinski for being my mentors throughout the course of this project.

References

Libraries: Tensorflow, scikit-learn

C. Piech, et. al. Deep Knowledge Tracing, Stanford University, 2015

M. Yudelson, et. al. Individualized Bayesian Knowledge Tracing Models. Carnegie Mellon University, 2016

M. Khajah, R. Lindsay, M. Mozer. How Deep is Knowledge Tracing, cs.AI, 2016