CS230

# Deep Knowledge Tracing and Engagement with MOOCs

**Kritphong Mongkhonvanit**
Graduate School of Education
Stanford University
kritphon@stanford.edu

**Klint Kanopka**
Graduate School of Education
Stanford University
kkanopka@stanford.edu

**David Lang**
Graduate School of Education
Stanford University
dnlang86@stanford.edu

## Abstract

MOOCs and online courses have notoriously high attrition [1]. One challenge is that it can be difficult to tell if students fail to complete because of disinterest or because of course difficulty. Starting from the Deep Knowledge Tracing framework, we account for student engagement by including course interaction covariates. With these, we find that we can predict a student's next item response with over 88% accuracy. Using these predictions, targeted interventions can be offered to students and targeted improvements can be made to courses.

## 1 Introduction

Massively Open Online Courses (MOOCs) offer the possibility of providing tremendous learning opportunities to students in a scalable form. However, because this population of students is substantially different in terms of the degree of interest, background, and other characteristics, a MOOC offers relatively few opportunities for assessment and personalization. As such the only way to assess a student's performance is often after they have passed or failed a class. In this paper we propose an early warning system that will predict subsequent course performance by incorporating video interaction features from a course as well as item data. This is an incremental but important step in identifying points where interventions could meaningfully improve students' learning outcomes.

The input to our model is a time series of feature vectors associated with an individual student's experience in an introductory MOOC on statistical learning. Each vector at "time" $t$ is associated with the $t^{th}$ question (questions will be henceforth referred to as *items* ) in the MOOC. The features for $x_t$ are the item correctness and seven features related to course structure and the student's interaction with the course. The first five are related directly to how the student engaged with the video associated with that question and include playback speed, whether or not the video was paused, fast forwarded or rewound and whether or not the video was completed. Playback speed is a value between 0.5 and 2.0, while the other variables are dichotomously coded. The final two features are a dichotomous variable for whether or not the item was attempted (defined as submitted with an answer selected) and whether or not the item was a part of an end-of-unit quiz. If the item was a quiz item, the video features are set to the average across all relevant features for the videos associated with the items on that quiz. These are fed into an RNN with $tanh$ activations and a sigmoid output layer, where the final output corresponds to the predicted probability of answering item $t + 1$ correctly.

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

| anon_name | feature | index1 | index2 | ··· | index103 |
|---|---|---|---|---|---|
| 38fqh9dy | items attempted | 103 | ··· | | |
| 38fqh9dy | correct | 0 | 1 | ··· | 1 |
| 38fqh9dy | playback_speed | 1 | 1.25 | ··· | 0 |
| 38fqh9dy | pauses | 1 | 0 | ··· | 0 |
| 38fqh9dy | seek_back | 0 | 0 | ··· | 0 |
| 38fqh9dy | seek_forward | 0 | 0 | ··· | 0 |
| 38fqh9dy | video_completed | 0 | 0 | ··· | 0 |
| 38fqh9dy | attempt | 1 | 1 | ··· | 0 |
| 38fqh9dy | quiz | 0 | 0 | ··· | 0 |

Figure 1: Example entry in the data set for one student

Our works is rooted in two fields in the realm of learning analytics. The first, Bayesian Knowledge Tracing (BKT), is the most common approach used for predicting subsequent performance in a class. BKT [2] models a student's knowledge of a concept as a binary indicator and predicts that if a student has that knowledge, a student will get subsequent items correct. While there have been extensions of this model, critics contend that BKT doesn't allow for multidimensional items (items that depend on knowledge in more than one topic) nor continuous latent traits (knowledge that is not all or nothing and can be partially mastered)[3].

Neural networks have further been used to extend learning in online environments in several key ways. Some work in this space have suggested that Neural Networks can accurately predict student dropout in MOOCs using sentiment analysis and discussion forum posts [4]. Piech first used RNNs to predict student performance as well as cluster items by skill domain in a technique he called Deep Knowledge Tracing (DKT)[5]. The original paper only uses item correctness and an item classification as input features. The advantages for this method include accuracy of prediction, flexibility of input features and an ability to identify latent categories in items without explicit identification. In an extension of DKT, other work has found that course interactions beyond strict item correctness and category are also predictive of student performance [6]. As such we thought a natural extension would be to combine DKT with course interaction covariates.

## 2 Dataset and Features

We collected item response and video interaction data for 12,007 students in a Stanford statistical learning MOOC [7]. In this course, students interacted with 77 videos which were associated with 103 individual items. From the videos, we took clickstream data and reduced it five video interaction features. Playback speed was coded in terms of a multiplier that ranged from 0.5 to 2.0. Pauses, seek back, seek forward and video completed were all dichotomous features coded as 0 if they did not occur during a video or 1 if they did occur during the video. If the student submitted an answer to a question, the "attempt" feature was coded 1 (and 0 otherwise) and the "quiz" feature was coded as 1 if the item was part of a summative (end of unit) quiz. Item correctness was also dichotomously coded with a 0 if incorrect and a 1 if correct. One uniquely appealing feature about this course compared other candidate MOOCs we had access to was that the structure of the course was such that almost every item was embedded into the videos in a "check for understanding" format. As such, it was trivial to map each item to its related video content. Figure 1 shows an example entry for an example student. The first line, "items attempted," was used by the code segment that read in the .csv of data and constructed the vectors for our model to produce appropriately padded vectors of consistent shape. Our model was ultimately evaluated with an 80/20 train/test split because we did not have an overwhelmingly large amount of training data and also wanted to experiment with training and evaluating on a subgroup of the most engaged students.

## 3 Methods

The model consists of a recurrent layer with 128 hidden units and $tanh$ activation. At the last time step, the model passes the activations through a dense layer with a sigmoid activation to produce the probability that the next response will be correct. The model is constructed is such a way that the exact

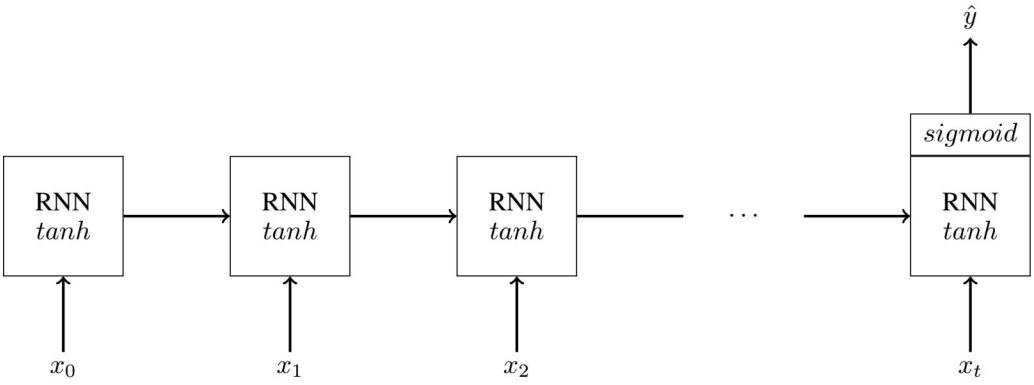


Figure 2: The structure of the model

| Inputs \ Type | GRU | LSTM | Simple RNN |
|---|---|---|---|
| Response only | 0.8726 | 0.8826 | 0.8668 |
| Response and clickstream | 0.8818 | 0.8830 | 0.8834 |

Figure 3: Comparison of the accuracy of different variants of the model

kind of RNN cells used can be changed easily. This was done so we could evaluate simple RNNs, GRUs and LSTMs in our architecture search. The implementation is also capable of automatically adapting the dimensions of the RNN layer to match the input data set.

All of the RNN models we investigated are designed to work on sequenced data, with each one feeding information about previous time steps into each subsequent iteration. By design, LSTMs should have the largest capacity for memory, followed by GRUs, followed by simple RNNs. This comes at the cost of computational efficiency, however. While the gated networks ought to outperform the simple RNN [8], we evaluated all three.

For our cost function, we used binary cross-entropy loss with L2 regularization,

$$J = -\frac{1}{m}\sum_{i=1}^{m}(ylog(\hat{y}) + (1-y)log(1-\hat{y})) + \frac{\lambda}{2m}\sum_{l=1}^{L}||w^{[l]}||_F^2,$$

because our model is performing a binary classification task (correct or incorrect) with estimated probabilities. L2 regularization was selected because it smoothed oscillations in the training loss. We had seen other RNN implementations using gradient clipping, but our model did not seem to need/benefit from that so it was left out. Our model also uses the Adam optimization algorithm, an optimization algorithm that uses the running averages of the previous gradients to adjust parameter updates during training. In addition, we implemented a variable learning rate that was automatically reduced during training if the validation loss failed to decrease after a few epochs.

The actual model itself was built from scratch using NumPy[9], scikit-learn[10], TensorFlow[11], and Keras [12].

## 4 Experiments/Results/Discussion

While conducting our hyperparameter search, we primarily experimented with the number of epochs, the type of recurrent layer, the number of hidden units, the learning rate and the regularization parameter. Throughout the training, accuracy was our primary evaluation metric. We chose to keep the parameters associated with the Adam optimizer constant during this search. Since our model trained relatively quickly, we chose to experiment with many hyperparameters across all three architectures before finally settling on the simple RNN. For all three architectures, 128 hidden units was a better balance of speed and accuracy than 64 or 256 hidden units. Similarly, we tried a variety

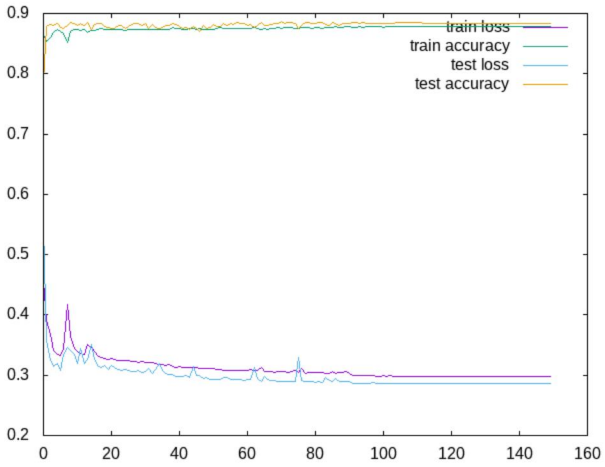


Figure 4: Training and test accuracy and loss for the final model (128 hidden unit RNN)

of values for the learning rate and regularization parameter before settling on the simple RNN with variable learning rate and a regularization parameter of $\lambda = 0.01$, which yielded a test accuracy of $88.4\%$.

In our research, we found that a simple RNN produced the highest accuracy for our dataset. This was not what was expected, as conventional wisdom seems to be that LSTMs outperform other types of RNNs. One potential explanation is that handling longer memory was not as important in this dataset as we would have expected. Past work suggests that there might be transition points in which a student's behavior changes during a course. Past studies of this particular course have shown that students tend to disengage after obtaining a passing mark in the course [13], further reducing the need for the advanced memory capabilities of gated models.

Another finding from our dataset is that the gains from including our engagement covariates are relatively small. Subsequent analysis of our datasets found that by the end of the course, nearly three quarters of students had no video interactions but still actively attempted items. This suggests that our gains from including the additional covariates are likely understated. Future work will focus on expanding this approach to additional courses in search of a generalizable model.

## 5 Conclusion/Future Work

For the course we investigated, a simple RNN and binarized engagement covariates can predict correct item responses with over 88% accuracy. The model we have presented here shows promise for implementation in live MOOCs as a tool to target students for support interventions and as a post hoc diagnostic for targeting segments of the course for improvement in future iterations. The performance of the simple RNN was something of a surprise, so we recommend still considering GRUs and LSTMs for future work.

For future work, there are a few directions we would like to explore. First, we would use the data from more courses to examine the problem in different contexts. If different courses behave similarly, it might suggest that a common architecture could be recommended to MOOC designers who wish to employ this type of modeling. Second, our project worked with a streamlined feature set, because we asked questions like, "did the student complete the video," and fed the model a binarized result. A larger and more sophisticated set of features might provide additional predictive power for the model.

## 6 Contributions

David Lang was primarily responsible for obtaining and cleaning the data set and the generation of summary statistics. Klint Kanopka and Kritphong Mongkhonvanit wrote the underlying Deep Knowledge Tracing model from scratch after experimenting with an Open Source implementation

that did not meet our needs and would have required extensive porting. Kritphong Mongkhonvanit was in charge of monitoring the model on FarmShare and drove the hyperparameter search. All parties worked collaboratively on the poster, document, research, planing and intermediate work.

## References


[1] Brent J Evans, Rachel B Baker, and Thomas S Dee. Persistence patterns in massive open online courses (moocs). *The Journal of Higher Education*, 87(2):206–242, 2016.

[2] Ryan SJ d Baker, Albert T Corbett, and Vincent Aleven. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *International Conference on Intelligent Tutoring Systems*, pages 406–415. Springer, 2008.

[3] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*, pages 505–513, 2015.

[4] Devendra Singh Chaplot, Eunhee Rhim, and Jihie Kim. Predicting student attrition in moocs using sentiment analysis and neural networks. In *AIED Workshops*, 2015.

[5] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dicstein. 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*, pages 505–513. Curran Associates, Inc., 2015.

[6] Tsung-Yen Yang, Christopher G Brinton, Carlee Joe-Wong, and Mung Chiang. Behavior-based grade prediction for moocs via time series neural networks. *IEEE Journal of Selected Topics in Signal Processing*, 11(5):716–728, 2017.

[7] Stanford Center for Advanced Online Reasearch and Learning. Stanford carol mooc data: Statistical learning, 2017.

[8] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[9] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[10] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[11] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[12] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[13] David Lang, Alex Kindel, Ben Domingue, and Andreas Paepcke. Making the grade: How learner engagement changes after passing a course. *Grantee Submission*, 2017.