Deep Knowledge Tracing

- Learning How Students Learn -

- Lisa Wang, Angela Sy, Larry Liu, Chris Piech -

Stanford University

Stanford ENGINEERING
Computer Science

Motivation

Massive open online courses (MOOCs) enable educators to reach millions of students by disseminating content through online classrooms. However, while these **online classrooms** cater to many students and instructors, they **are unable to provide individual feedback to students about their progress**. Research has found that feedback is an extremely critical component of student learning.

In our research, we **use deep learning to understand a student's learning trajectory as they solve open-ended problems**. With a robust understanding of student learning, we can **ultimately provide personalized automated feedback to students at scale**.

Definitions

Abstract Syntax Tree (AST): the specific ordering of code blocks that a student submits for a particular problem.

Trajectory: the ordered sequence of ASTs that a student submits for a particular problem leading up to their final solution.

Gold solution: the AST that most efficiently solves the problem (fewest blocks of code). For every problem, this also happens to be the most common solution provided by students

Data Overview

Project Statistics

Hour of Code Problems analyzed	9
Average Number of Unique Students	655,169
Average Student Accuracy	66.30%

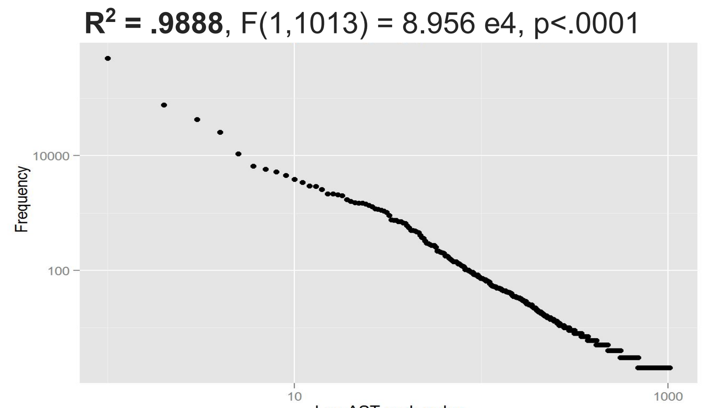
Hour of Code Problem 9

Number of Trajectories	12,324
Longest Trajectory Length (timesteps)	6
Number of ASTs	2,042

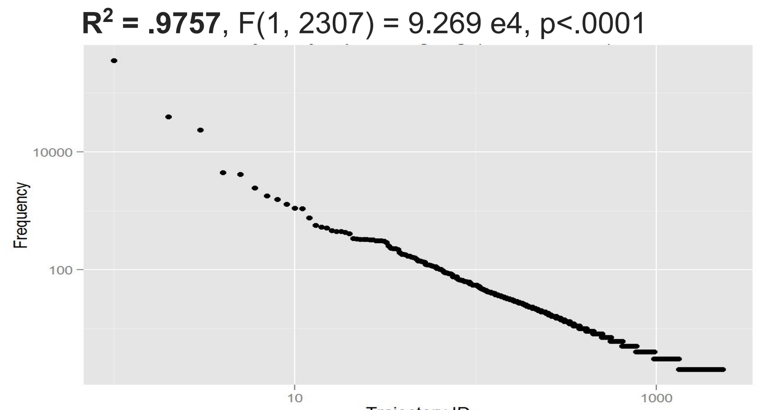
Data Distribution

The distribution of AST and Trajectory frequencies both follow Zipf's Law--that is, their frequency inversely correlated to its statistical rank.

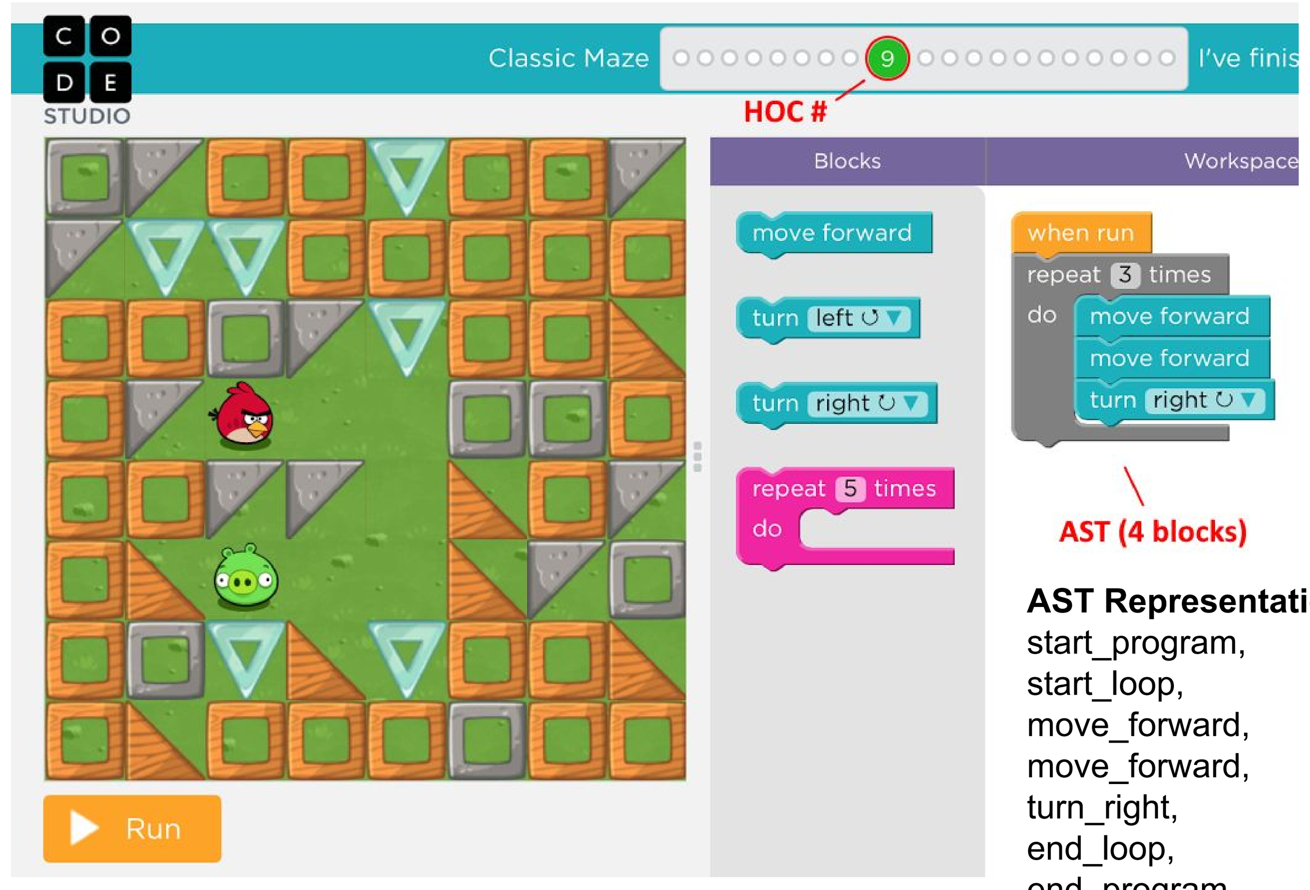
AST Linear model: log(y) = -1.75log(x) + 12.36
R^2 = .9888, F(1,1013) = 8.956 e4, p<.0001



Trajectory Linear model: log(y) = -1.16log(x)+9.29
R^2 = .9757, F(1, 2307) = 9.269 e4, p<.0001



Code.org Example Exercise



Hour of Code Problem 9

Deep Knowledge Tracing and RNNs

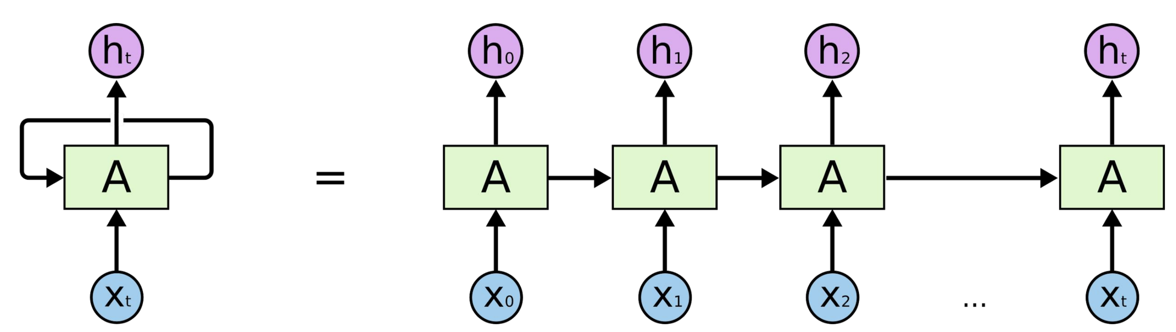
Knowledge tracing: predicting a student's next interaction based on observations of their previous interactions, (Corbett et al.).

Deep knowledge tracing: Knowledge tracing using artificial neural networks (deep learning). We explore how well DKT performs on a richly structured dataset, i.e. where the solution space is unbounded.

Recurrent Neural Networks (RNN)

RNNs are suited to predict outcomes based on a sequence of inputs. A general RNN takes as input a time series of vectors x_0 through x_t and outputs a time series of vectors h_0 through h_t.

RNNs are ideal models for our knowledge tracing task which tracks a student's learning over multiple timesteps because RNNs maintain information on the history of a student's learning, rather than a mere snapshot of a student's most current knowledge.



Unrolled RNN Architecture

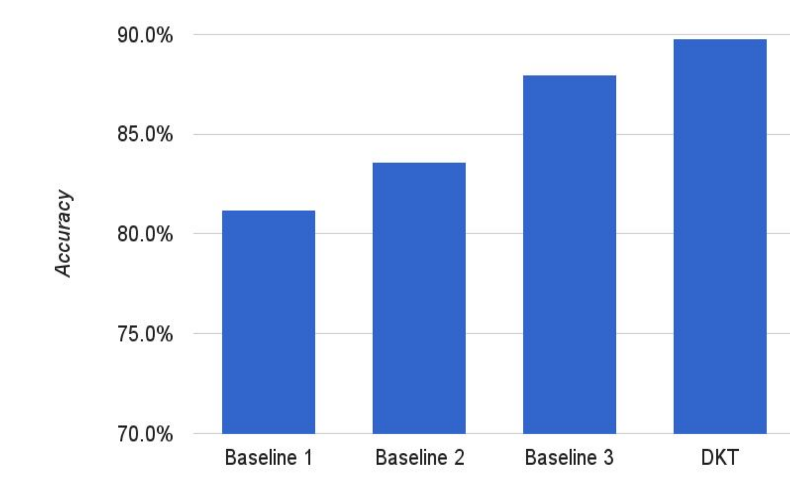
(Image Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Task 1: Predict Performance on Next Exercise

Given their performance (correct or incorrect) on the first *t* exercises, predict whether the student will solve the next exercise correctly.

Input x_t: A one-hot encoding representing whether the student successfully solved the exercise at time step *t*
Output y_t: Predicted probability of the student answering each of the exercises correctly at time step *t+1*.

Method	Acc
Baseline 1: Always predict correct	81.2%
Baseline 2: Use Historical Probability	83.6%
Baseline 3: Imitate Outcome of Previous Question	88.0%
DKT	92.0%



Task 2: Predict the Next Move

Predict the next step a student takes in their problem solving path, given their previous steps on the same coding exercise.

Method 1: Using AST IDs as inputs
Input x_t: AST ID at time step *t*
Output y_t: Predicted AST ID at time step *t+1*

Method 2: Using program embeddings as inputs
A program embedding summarizes the features of a program based on the actual code. We derived the embeddings using a separate RNN by predicting the next command within the program.

Input x_t: Program embedding of AST at time step *t*
Output y_t: Predicted AST ID at time step *t+1*

Promising advantages of using program embeddings:

- **Flexible:** Same code gets the same embedding, independent from specific exercise.
- **Scalable:** All embeddings have the same vector length.