Predicting programming success using Deep Learning

Ben Stenhaug stenhaug@stanford.edu
Raejoon Jung raejoon@stanford.edu
Neeraj Mathur mathurn@stanford.edu

Introduction

A valuable method of education is to let students attempt a task in an environment and then help them when their struggle starts to become unproductive. In statistical terms, a struggle is unproductive if there is a high likelihood that a student will not accomplish the task within T tries where T is a parameter that determines how fast the struggling student should be assisted. This is a perfect modeling task for deep learning.

Our deep learning modeling goal is to **predict if a student will fail to accomplish the coding task within T tries**.

Dataset

Data is provided by code.org (https://code.org/research). We focus on data from a single exercise where students guide the blue arrow to the green heart using code blocks:

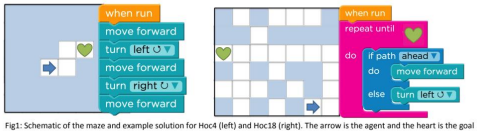


Fig1: Schematic of the maze and example solution for Hoc4 (left) and Hoc18 (right). The arrow is the agent and the heart is the goal

Students can submit their code as many times as they would like. Each code submission is represented as an Abstract Syntax Tree (AST).

	Hoc4	Hoc18
Unique # of ASTs	3,629	34,901
Unique # of trajectories	55,354	66,822
# of Students	403,022	263,569

Approach

Key Learnings & Decisions

1. RNN is used as code submissions are sequential
2. Generated embeddings using model from paper #1. This model is having a LSTM layer and output of this layer is used as embeddings.
3. Each student has completed the homework with different steps, Input matrics is padded so all students had same number of "attempts"
4. We noticed data imbalance thus Cut off to max timestep of 10 is applied

Data Preparation

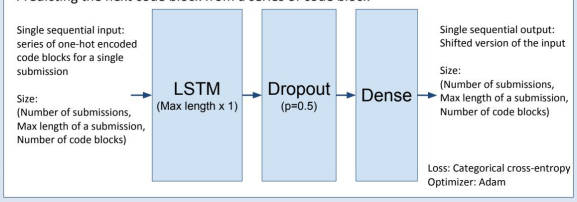
1. Input: ASTs represented in matrices where each row is a code block
2. Output: Matrix where row is a student and columns are timesteps and cells denote success within T timesteps

Development

1. Used R for generating input metrics and output labels from the data (JSON) provided by code.org
2. Coded model in Python using Keras with TensorFlow backend.

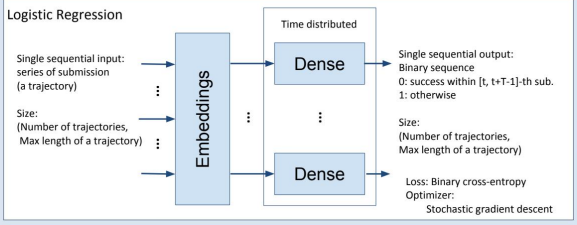
Embeddings generation model

Predicting the next code block from a series of code block

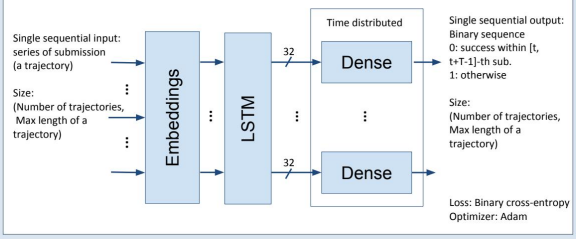


What is our Baseline?

Logistic Regression



Neural Network Architecture



Results

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression (w/o Embeddings)	0.9276	0.9125	0.9579	0.9344
Logistic Regression (w/ Embeddings)	0.8557	0.8714	0.8627	0.8665
RNN (w/o Embeddings)	0.9010	0.8768	0.9499	0.9112
RNN (w/ Embeddings)	0.9255	0.9147	0.9515	0.9325

Table 1: Results for Hoc4 (window length: 2)

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression (w/o Embeddings)	0.7625	0.6944	0.8627	0.7628
Logistic Regression (w/ Embeddings)	0.6894	0.6370	0.7512	0.6823
RNN (w/o Embeddings)	0.7778	0.7876	0.7142	0.7424
RNN (w/ Embeddings)	0.7784	0.7656	0.7523	0.7523

Table 2: Results for Hoc18 (window length: 64)

Discussion

- It is critical to pay attention to the **distribution of the data** — what to do with the few students that make many attempts
- Predicting the next block to generate **code embeddings** may not be useful because students tend to use different logic in coding during learning (in contrast to NLP)
- **Recall** is key metric because intervention is not costly and having a student give up is
- The **baseline** performs incredibly well. We hypothesize that this is the result of the simplicity of the coding exercise
- Predicting for longer **window length** is harder since it needs to predict outcomes in further future.
- Determining the **optimal window length** is an interesting educational question that corresponds to how long to let a student struggle
- Open question on whether **unit of analysis** should submission or student

Future Directions

- Use on more complicated block structures
- Use on non-block code including a variety of languages
- Use for different purposes: computer science, data science, etc.

References

1. Piech, Chris, et al. "Autonomously generating hints by inferring problem solving policies." Proceedings of the Second (2015) ACM Conference on Learning@ Scale. ACM, 2015.
2. Wang, Lisa, et al. "Learning to represent student knowledge on programming exercises using deep learning." Proceedings of the 10th International Conference on Educational Data Mining; Wuhan, China. 2017.
3. Vihavainen, Arto. "Predicting Students' Performance in an Introductory Programming Course Using Data from Students' Own Programming Process." Advanced Learning Technologies (ICALT), 2013 IEEE 13th International Conference on. IEEE, 2013.