Using Q-Learning to Personalize Pedagogical Policies for Addition Problems

Amanda Shen
Cortney Weintz
Takara Truong

I. Introduction

- The prevalence of COVID-19 over the past year has illuminated the need for effective digital education tools. With students studying from home, teachers have struggled to provide their students with adequately challenging coursework.
- Our project aims to solve this issue in the context of math. More specifically, our goal is to encourage thoughtful learning by supplying students with personalized two-number addition problems that take time to solve but that we expect the student can still answer correctly.
- Our solution is to model the process of selecting a math problem to give a student as a Markov Decision Process (MDP) and then use Q-learning to determine the best policy for arriving at the most optimally challenging two-number addition problem for that student.

II. Features

- A Problem generator produces all possible combinations of two three-digit integer addition problems and bins the problems based on a set of features. The math problems are characterized by four features, Table 1.
- For two three-digit addition problems, there are 1 million total possible problems that can be given. Our feature extractor bins these problems into 49 distinct states.

II. Data

- For this project, we create a simulated student for each group member and aptly name them: Amanda, Cortney, Takara.
- Each member solved four randomly generated problems for each state (mentally). A distribution of response time and probability of correctness was created for each state from which the simulation sampled from.

Table 1. State Representation Examples

State Representation				Example Math Problems		
number 1 digit count	number 2 digit count	# of carry operations	# of zeros			
1	1	1	0	1+9	2+8	9+6
2	2	1	2	10+90	30+70	70+50
2	3	1	0	11+119	15+793	23+291
3	3	3	0	111+889	116+888	898+114

III. Model

MDP Formulation:

- $start\_state()$: problem of lowest degree difficulty which is a zero added to a constant.
- $action(s)$: set of valid actions from the current difficulty to the next. For each feature, we can increase/decrease by 1 or stay. Aside from the lowest/highest difficulty, each state has 4x3=12 actions.
- $successor(s, a)$: gets the next state, s', from the current state and action
- $reward(s, a, s')$: The difference between the current response time and the following. If the user is wrong, the difference returned is enforced to be negative as a penalty.

IV. Experiments & Results

- The first experiment aimed to answer how quickly (number of iterations) the algorithm can give an appropriate level problem for simulated each student.
- The second experiment aimed to answer whether pre-training on Cortney and Takara would make the process faster for Amanda.

Table 2. Results

	Iterations (mean ± std)		
	Amanda	Cortney	Takara
Experiment 1	162 ± 134	230 ±2 05	247 ± 236
Experiment 2	35 ± 44	-	-

VI. Discussion

- Experiment 1 demonstrates that the algorithm can give appropriately difficult problems; however, it takes many iterations to arrive there. Experiment 2 shows that we can speed the process up by pre-training.
- This observed improvement is reasonable since there are similar trends in what students find difficult. For instance, three-digit problems usually take longer to solve than two-digit problems.
- Inspection of student data shows that everyone has unique strengths and weaknesses. This is a likely why it takes several iterations for Amanda to adapt to the pre-trained Q-table from Cortney and Takara.

VII. Future work

- The project can be easily scaled towards different types of problems such as subtraction, multiplication, division and could be expanded towards solving algebraic problems or word-based math problems.
- Future work may also use deep-Q learning to alleviate the need for feature engineering.

Stanford University

Stanford | ENGINEERING
Computer Science

Amanda Shen
Stanford University
Computer Science
amshen@stanford.edu

Cortney Weintz
Stanford University
Computer Science
cweintz@stanford.edu

Takara Truong
Stanford University
Computer Science
takaraet@stanford.edu