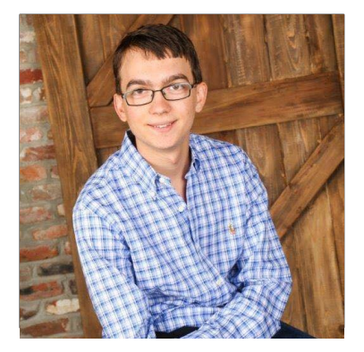
Department of Computer Science
Franklin College of Arts and Sciences

Center for Undergraduate Research Opportunities
UNIVERSITY OF GEORGIA

Analyzing Varying Complexity On Q-Learning



Bailey Nelson
CSCI 4960 Student
University of Georgia
bdn81550@uga.edu

Matthew Chapman
CSCI 6950 Student
University of Georgia
mdchap@uga.edu

Michael E. Cotterell
Lecturer, Department of Computer Science
University of Georgia
mepcott@uga.edu

For more information, please visit
http://baileydnelson.com/qlearning/
or scan the QR code to the right
→

Introduction

- Q-Learning is a Reinforcement Learning algorithm used to teach a computer agent how to act optimally in a complex environment. While playing the game, the computer agent needs to calculate multiple Q(s,a), i.e., measures of how well off they will be for taking a particular action a in state s, then pick the best one.

$$Q(s, a) \leftarrow Q(s,a) + \alpha(r + \gamma(\max(Q(s', a')) - Q(s, a))$$

  - $0 < \alpha < 1$ is the learning rate;
  - r is the reward for taking action a; and
  - $0 < \gamma < 1$ is a discount factor for utilizing future rewards.
- **Research Questions:** How effective can Q-Learning be as the difficulty of problems increase? What are the limitations?
- **Hypothesis:** Learning happens logarithmically; rapid initial growth followed by stagnation.

Game Setup

- Tools utilized: Python, PyGame, pandas, numpy, sklearn, scikit.
- In order to increase complexity, a change is made to the game. Instead of a game over occurring from hitting the walls or the body. There are additional blocks placed in the game at random places, making the board more difficult to navigate.

Moving Towards Food	Eating Food	Moving Away From Snake	Game Over
-0.2	1	0.1	-100

Table 1. The reward function. Chosen in a way that rewards the snake for making progress towards the food and penalizing death.

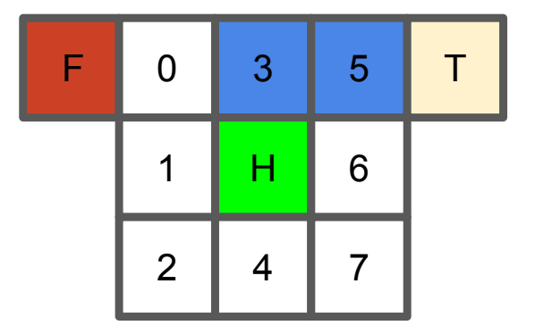
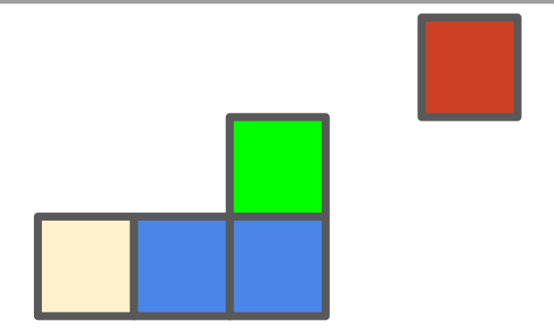
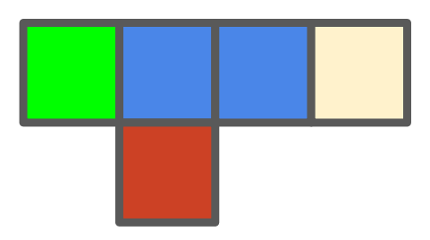
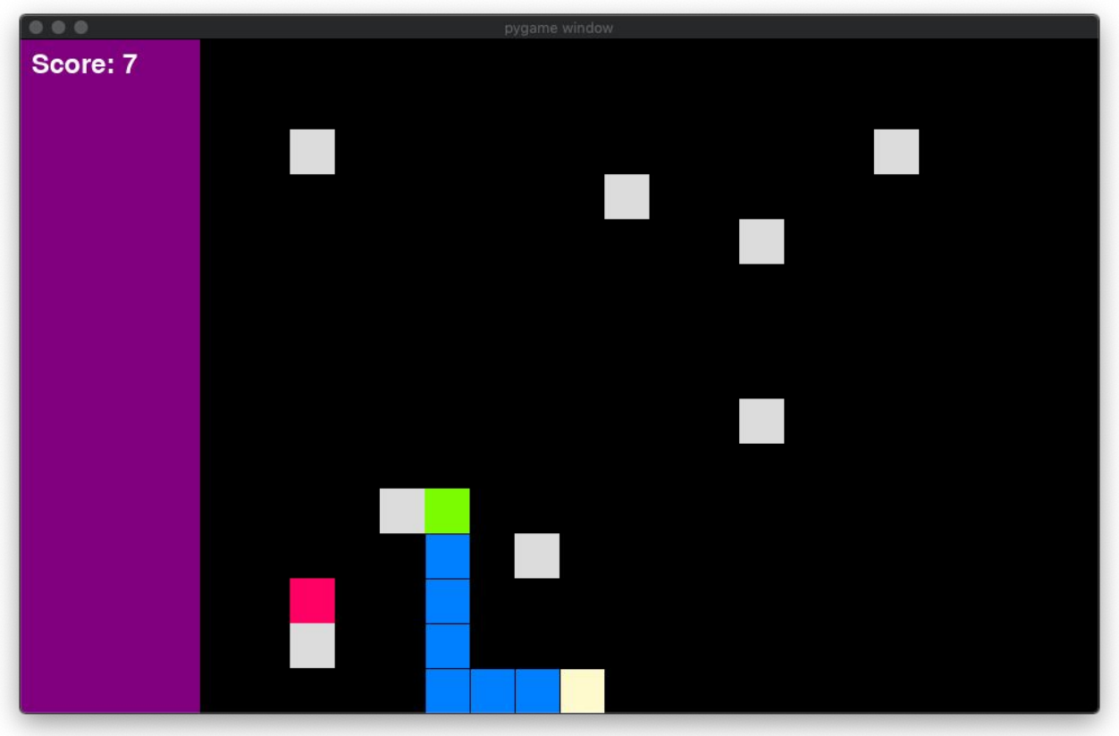
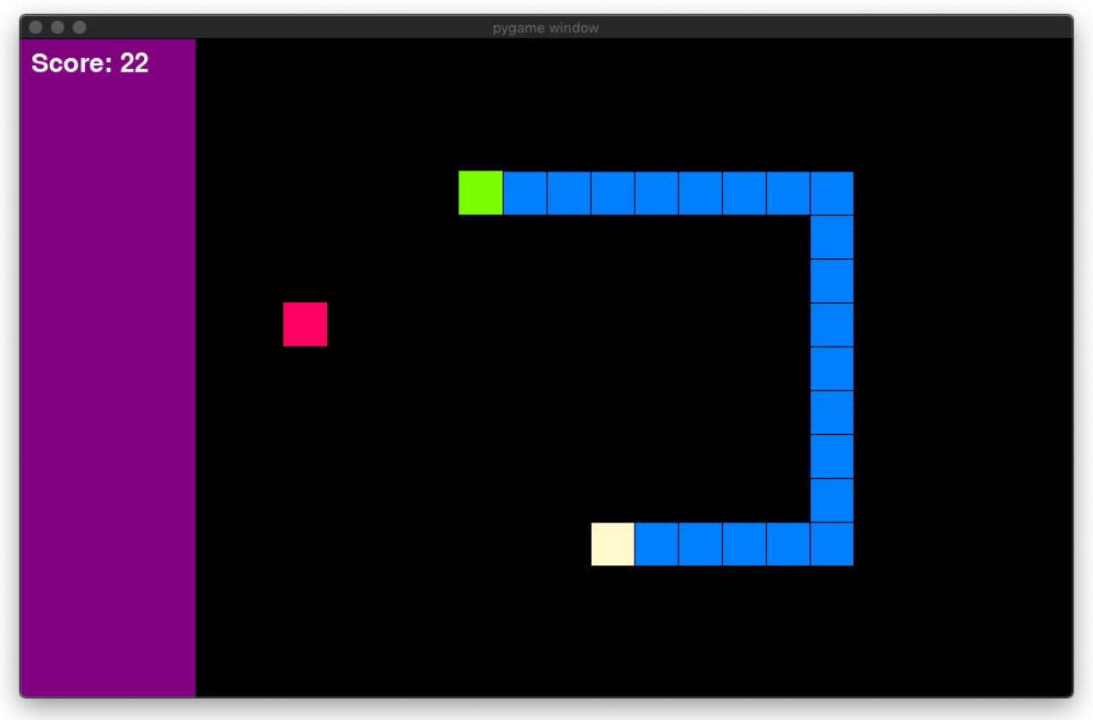
State		Value of Action			
Diagram	Bitmap	UP ⇧	DOWN ⇩	LEFT ⇦	RIGHT ⇨
	11 (direction) 01 (food location) 00101000 (surrounding)	N/A	0	0.109	-0.02
	00 00 00101000	0.11791	N/A	-0.5	0.0454636
	10 11 00000010	-3.21505	0.6458541	0	N/A

Table 2. An example Q-Table. The first entry shows an in-depth example of the state is encoded. The columns show the Q-Value for taking that action at that state. The highlighted column denotes the action that would be chosen.



Screenshot 2. An example from the second level of complexity. The gray blocks represent the blocks that will cause a game over if touched.



Screenshot 1. An example from the base level snake game.

Methodology

- For this experiment, the snake trained for 100 replications and the number of trials went from 10 to 200 by 10.
- Each trial is a training game for the snake and each replication is the snake training for the number of trials before playing a game where the data is recorded.
- The data analyzed is a 95% confidence interval of the average score achieved over the replications.
- The hypothesis was the data would be logarithmic. This means the agent would learn and improve very rapidly at first, but the improvements would quickly diminish regardless of the length of training.
- The score of the game is directly determined by the number of food eaten.

Results

- Once the data was collected, the data was "linearized" by multiplying the data by Euler's number.
- Figure 1 shows the linearized data from the first complexity of the game.
- Once the data is linearized, a Linear regression is performed to model the linear relationship on the transformed data.
- The red line shows the linear regression performed on the data.
- Once the linear line of best fit is obtained, it can be turned into a logarithmic line by multiplying all of the data on that line by the natural log (ln). This converts the linear data back to logarithmic data.
- Now the logarithmic regression can be viewed compared to the original data to determine the fit of the logarithmic line.
- Figure 3 undergoes the same transformation.
- The R^2 value for figure 2 is 0.7716
- The R^2 value for figure 3 is 0.6694

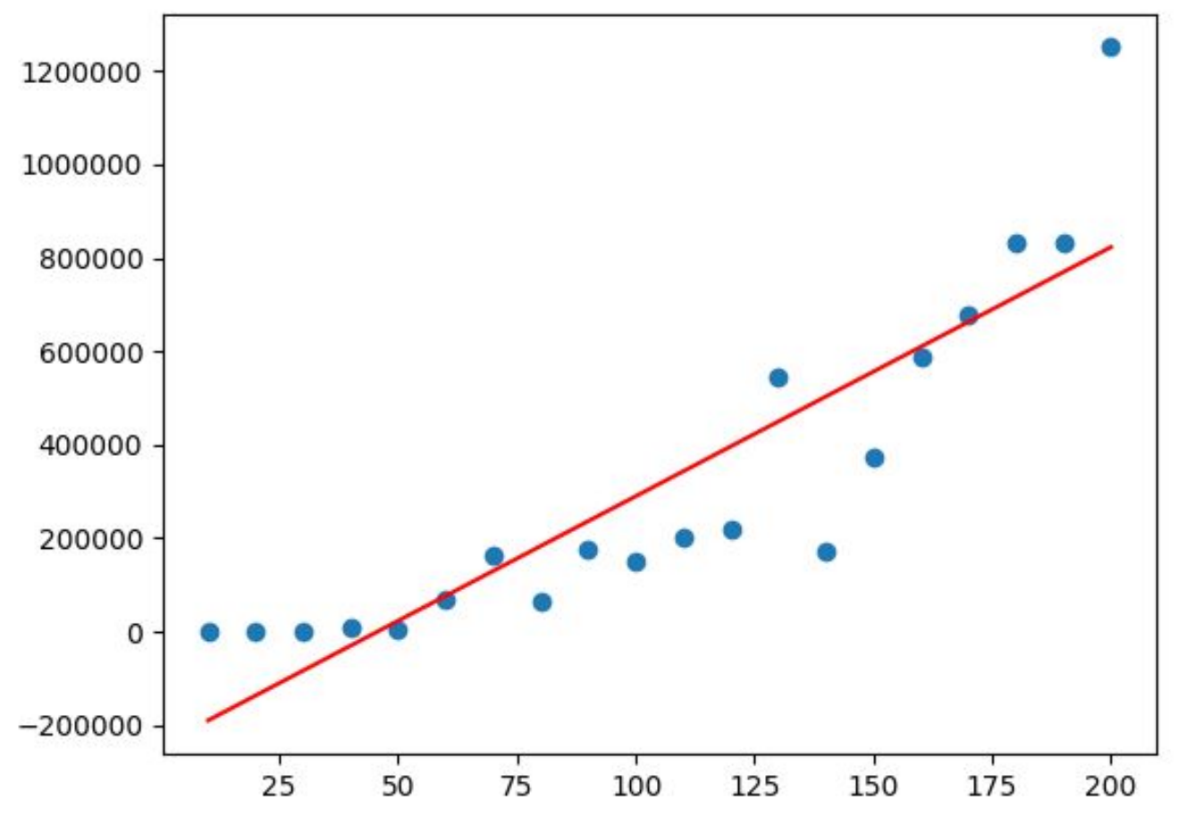


Figure 1. Linearized data with line of best fit in red.

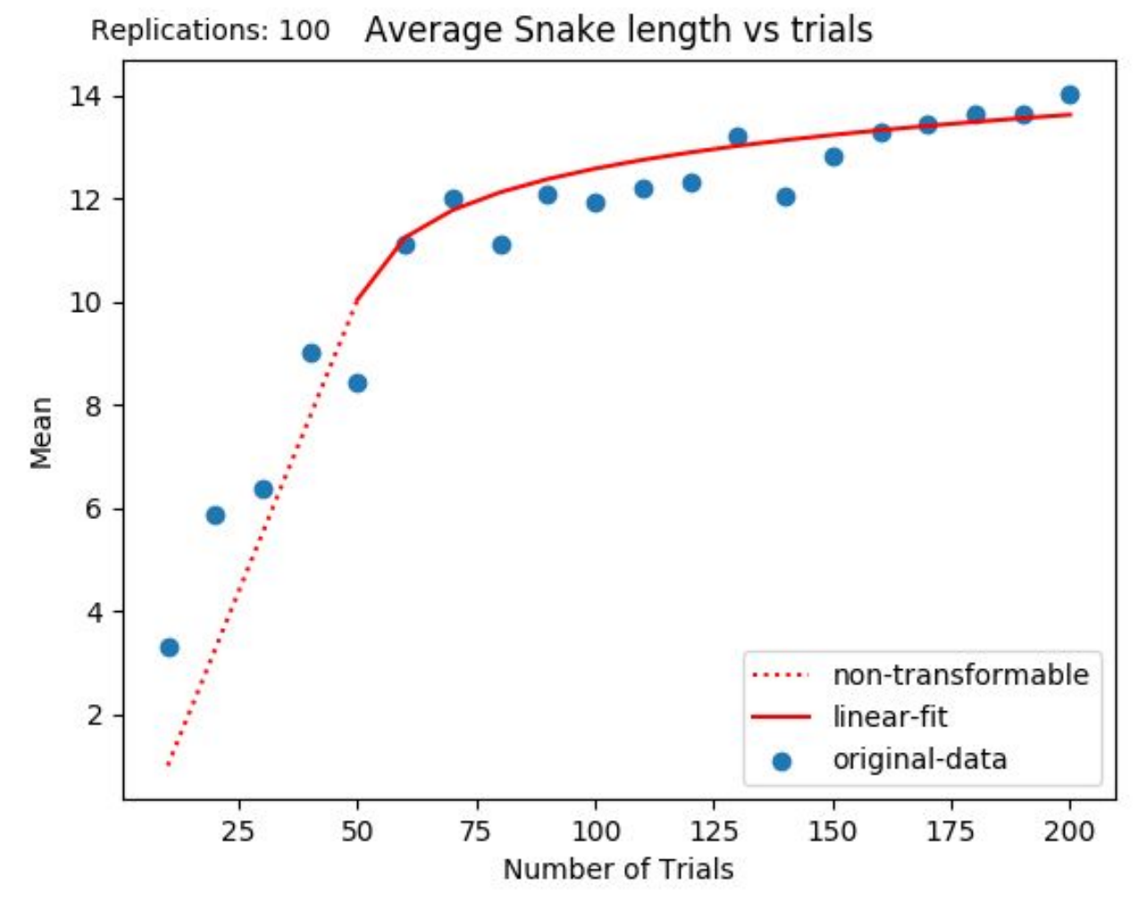


Figure 2. First complexity data with line of best fit.

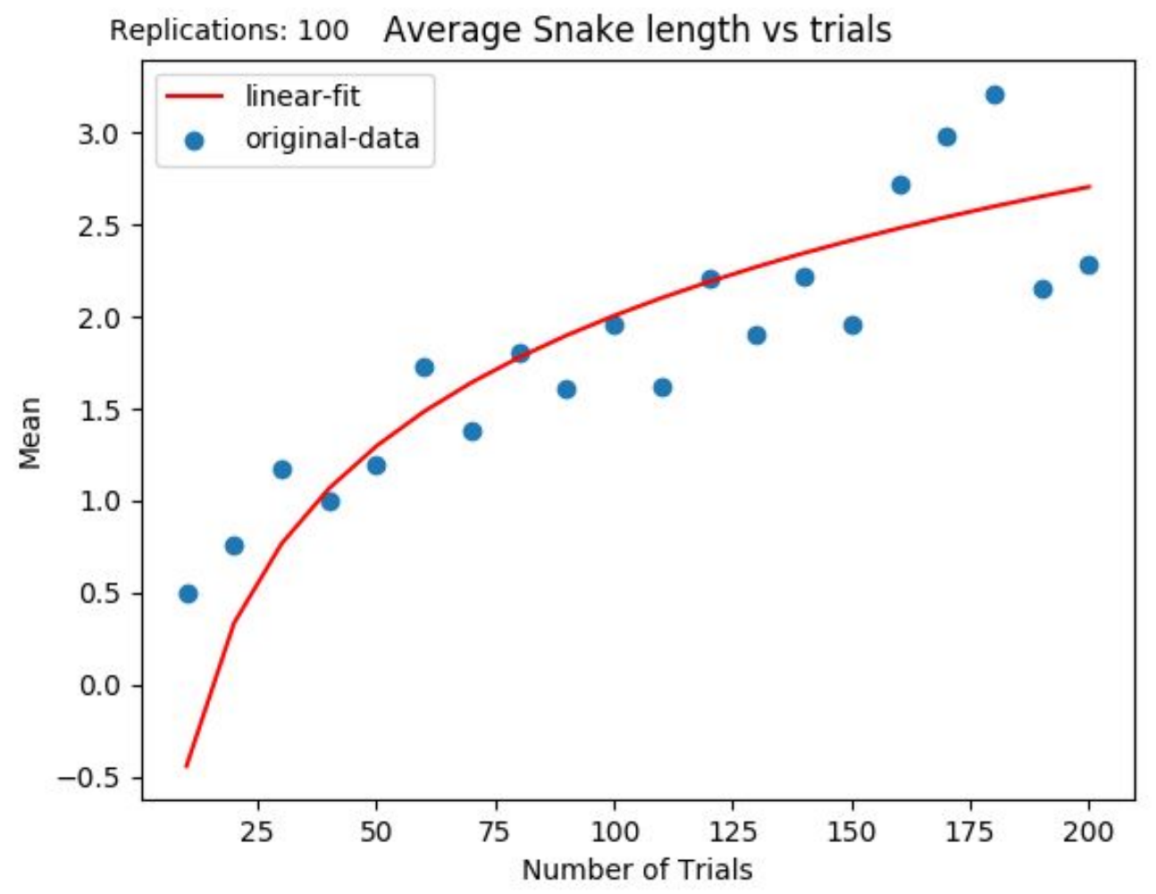


Figure 3. Data observed from second level of complexity.

Discussion

- Looking back at figure 2, the snake was initially able to learn quickly. From no knowledge of good moves, to only 20 training sessions, the snake could get a score of, on average, 5.85.
- Rapid growth can be seen until around 100 training sessions. After 100, it can be seen that the growth has stagnated.
  - From 20 training sessions to 100 training sessions, the growth is 103.76%.
  - Even after another 80 training sessions, the average score increases from 11.92 (at 100 training sessions) to 13.63 (at 180 training sessions). A growth of only 14.3%.
- The R^2 value related to figure 2 is 0.7716. This suggests a strong relationship between the model and the actual data. The R^2 value of .6694 is slightly weaker, but still a good indicator of fit.
- The max score in the first iteration is approximately 14, the max score in the second iteration is approximately four. This illustrates how a new complexity can affect the ability of the agent.
  - Perhaps a new state encoding could further help the snake avoid the new obstacles.
- In the original iteration, the only obstacles the snake could run into are their own tail or the wall. The snake couldn't accidentally hit its tail until it already achieved a score of five. In the second iteration, the food is now placed in a maze of walls. It can very easily game over on the way to the food.
- Project can be expanded on:
  - What happens at next complexity?
  - How does the state encoding affect the learning?
- Difficulties with website:
  - Teaching an unknown audience with unknown prior knowledge about a complex algorithm.
  - Rewriting code from scratch, but recreating initial mistakes to illustrate common errors is difficult.

Conclusion

- The results of the experiments coincide with the hypothesis. There was rapid growth at the beginning, but the growth wasn't continuously linear.
- This would suggest that while Q-Learning can be effective at getting initial results of learning. However, as the problem complexity increases, the algorithm is less effective at getting optimal results.
- To answer the research questions:
  - Q-Learning follows the same logarithmic curve regardless of complexity, but with the same state encoding, it isn't as effective.
  - The main limitation is computational power. Since the curve appears to be logarithmic, it is monotonically increasing, enough training will always cause an increase in score.