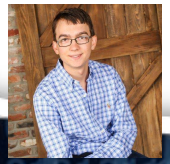
Department of Computer Science
Franklin College of Arts and Sciences
Center for Undergraduate Research Opportunities
UNIVERSITY OF GEORGIA

Analyzing Varying Complexity On Q-Learning

Bailey Nelson, Matthew Chapman, Dr. Michael Cotterell

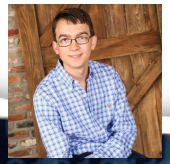
CURO Poster + Website

- Website: baileydnelson.com/qlearning
- Poster: baileydnelson.com/qlearning/poster



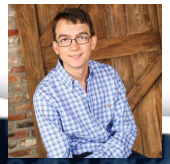
Scope of Research (Abstract)

- How does complexity affect Q-Learning algorithm?
- What are the limitations of Q-Learning?
  - Limitless learning?
- Learning aspect, want future students to be able to repeat experiment (website).



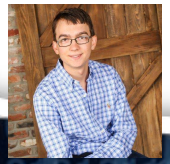
Machine Learning

- Computer agent learns!
- Like humans, computers love being rewarded.
- From “Execute this command because I tell you to” to “Obtain greatest reward”
- Very diverse field



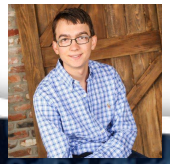
General Vocabulary

- Policy – How the agent decides to act
- Reward – Numerical value received by the agent form the environment. Chosen arbitrarily
- State – Current environment and all information needed to adequately explain environment
- Action – Choice made by agent

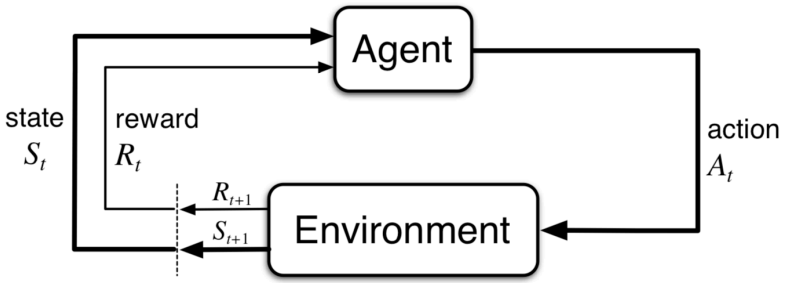


Introduction to Reinforcement Learning

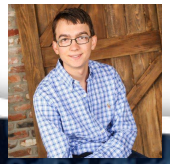
- Supervised Learning (learning from test data)
- Evolutionary strategies
  - Learn like humans, after many evolutions of policy
- Reinforcement Learning
  - Learns as actor interacts with environment



Q-Learning

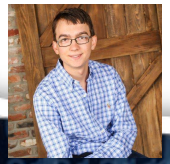


- Q-Table – how decisions are made
- Off Policy
- Previous knowledge of the current state is used to influence its next decision



Q-Learning

- $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max(a) Q(S_{t+1}, a) - Q(S_t, A_t)]$



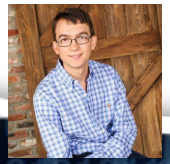
Example

Game Board:

Current state (*s*): 0 0 0
0 1 0

Q Table: γ = 0.95

	0 0 0 1 0 0	0 0 0 0 1 0	0 0 0 0 0 1	1 0 0 0 0 0	0 1 0 0 0 0	0 0 1 0 0 0
↑	0.2	0.3	1.0	-0.22	-0.3	0.0
↓	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
→	0.21	0.4	-0.3	0.5	1.0	0.0
←	-0.6	-0.1	-0.1	-0.31	-0.01	0.0



Example

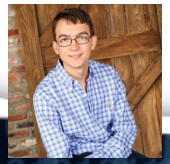
Game Board:

Current state (*s*): 0 0 0 / 0 1 0

Selected action (*a*): ⇨

Q Table: γ = 0.95

	0 0 0 1 0 0	0 0 0 0 1 0	0 0 0 0 0 1	1 0 0 0 0 0	0 1 0 0 0 0	0 0 1 0 0 0
⇧	0.2	0.3	1.0	-0.22	-0.3	0.0
⇩	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
⇨	0.21	0.4	-0.3	0.5	1.0	0.0
⇦	-0.6	-0.1	-0.1	-0.31	-0.01	0.0



Example

Game Board:

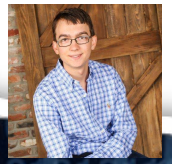
Current state (*s*): 0 0 0 / 0 1 0

Selected action (*a*): →

Reward (*r*): 0

Q Table: γ = 0.95

	0 0 0 1 0 0	0 0 0 0 1 0	0 0 0 0 0 1	1 0 0 0 0 0	0 1 0 0 0 0	0 0 1 0 0 0
↑	0.2	0.3	1.0	-0.22	-0.3	0.0
↓	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
→	0.21	0.4	-0.3	0.5	1.0	0.0
←	-0.6	-0.1	-0.1	-0.31	-0.01	0.0



Example

Game Board:

Current state (*s*): 0 0 0 / 0 1 0

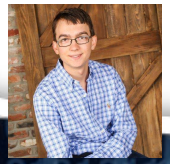
Selected action (*a*): ⇨

Reward (*r*): 0

Next state (*s'*): 0 0 0 / 0 0 1

Q Table: γ = 0.95

	0 0 0 1 0 0	0 0 0 0 1 0	0 0 0 0 0 1	1 0 0 0 0 0	0 1 0 0 0 0	0 0 1 0 0 0
⇧	0.2	0.3	1.0	-0.22	-0.3	0.0
⇩	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
⇨	0.21	0.4	-0.3	0.5	1.0	0.0
⇦	-0.6	-0.1	-0.1	-0.31	-0.01	0.0



Example

Game Board:

Current state (*s*): 0 0 0 / 0 1 0

Selected action (*a*): ⇨

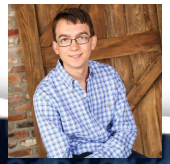
Reward (*r*): 0

Next state (*s'*): 0 0 0 / 0 0 1

max Q(s'): 1.0

Q Table: γ = 0.95

	0 0 0 1 0 0	0 0 0 0 1 0	0 0 0 0 0 1	1 0 0 0 0 0	0 1 0 0 0 0	0 0 1 0 0 0
⇧	0.2	0.3	1.0	-0.22	-0.3	0.0
⇩	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
⇨	0.21	0.4	-0.3	0.5	1.0	0.0
⇦	-0.6	-0.1	-0.1	-0.31	-0.01	0.0



Example

Game Board:

Q Table: $\gamma = 0.95$

	000 100	000 010	000 001	100 000	010 000	001 000
↑	0.2	0.3	1.0	-0.22	-0.3	0.0
↓	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
→	0.21	0.4	-0.3	0.5	1.0	0.0
←	-0.6	-0.1	-0.1	-0.31	-0.01	0.0

Current state (s): **0 0 0 / 0 1 0**

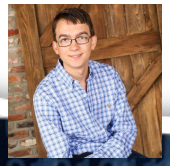
Selected action (a): →

Reward (r): **0**

Next state (s'): **0 0 0 / 0 0 1**

max Q(s'): **1.0**

→ New Q(s,a) = r + γ * maxQ(s') = 0 + 0.95 * 1 = **0.95**



Example

Game Board:

Q Table: $\gamma = 0.95$

	000 100	000 010	000 001	100 000	010 000	001 000
↑	0.2	0.3	1.0	-0.22	-0.3	0.0
↓	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
→	0.21	**0.95**	-0.3	0.5	1.0	0.0
←	-0.6	-0.1	-0.1	-0.31	-0.01	0.0

Current state (s): **0 0 0 / 0 1 0**

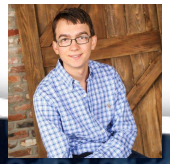
Selected action (a): →

Reward (r): **0**

Next state (s'): **0 0 0 / 0 0 1**

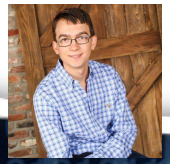
max Q(s'): **1.0**

New Q(s,a) = r + γ * maxQ(s') = 0 + 0.95 * 1 = **0.95**



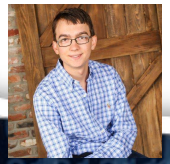
Choosing Reward

- ”Cobra Effect”
  - Government gave people money for dead cobras to help rid the area, people started breeding them.
  - **You get what you incentivize, may not be what you intend**



Example

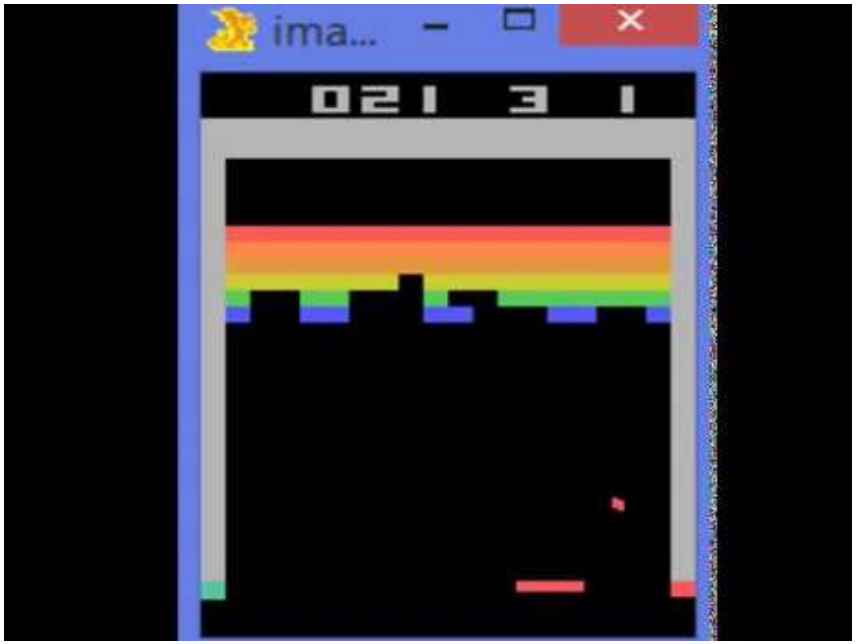
- Imagine you want to teach a robotic arm to stack blocks far away from the arms base. You may incentivize the arm based on how far away the block is from the arm. However the result could be as follows:

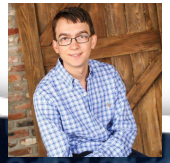



Speed 1x
Quality Auto

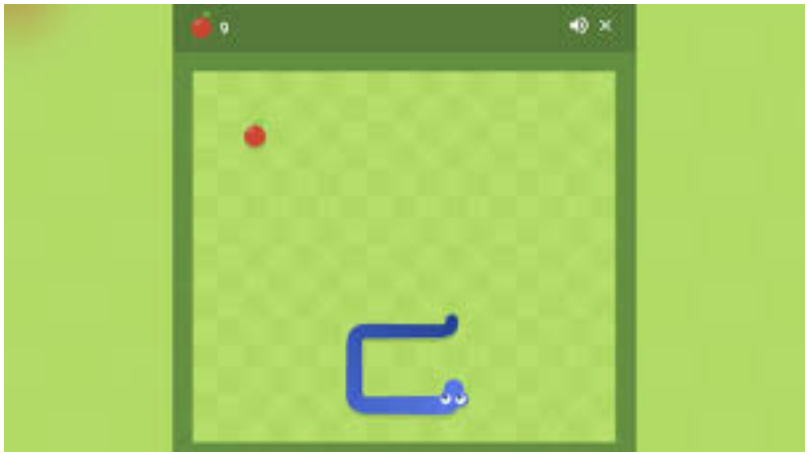
Q-Learning Application

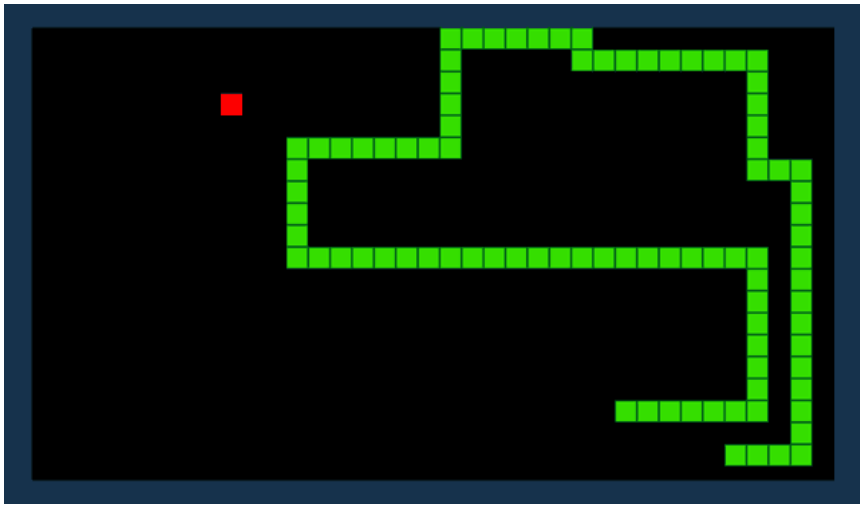
- Google DeepMind
  – "Deep Q-Learning"
    – Can play Atari 2600 games at expert levels

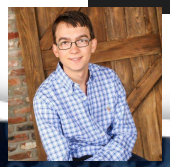




Snake

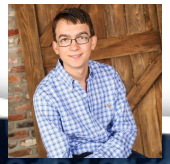




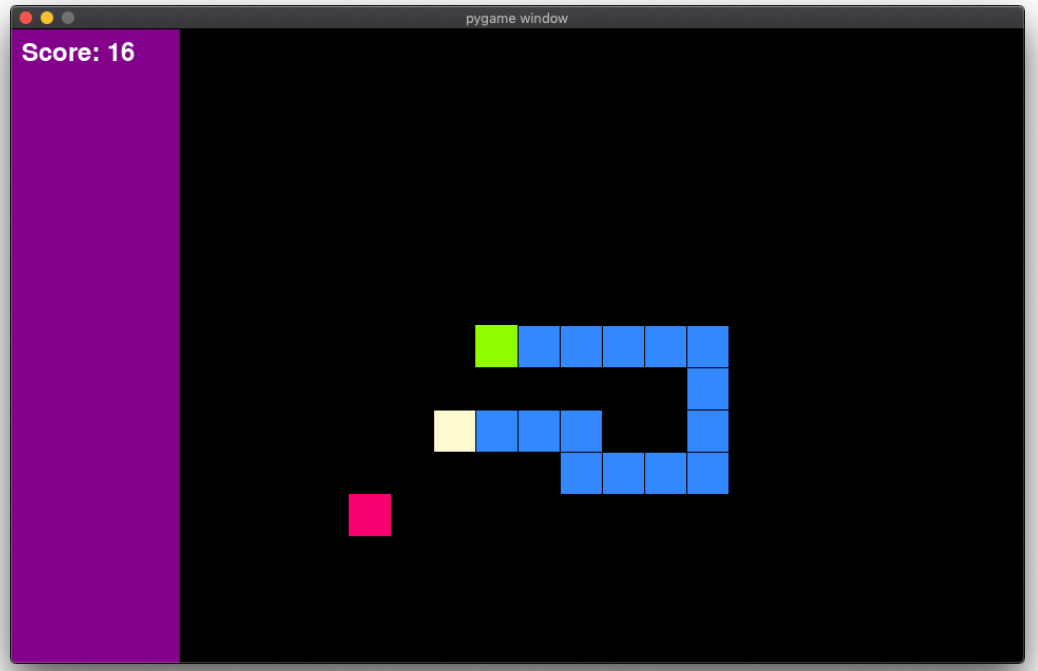


Why Snake?

- Simple. Most everyone knows how to play it/what it is
- Easy to increase complexity
- Quick to create and see results
- Inspiration from previous work

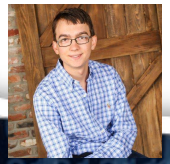


My Snake



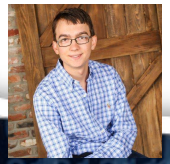
Elements:

Head: Green
Tail: White
Food: Pink
Body: Blue
Score: Length of snake



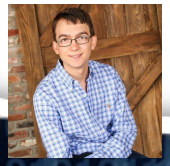
Software Stack

- Python (v3.8.1)
  - Pygame (v2.0.0)
    - Snake
  - Pandas (v1.0.1)
    - QTable
  - Matplotlib (v3.1.3)
    - Graphing/Analyzing



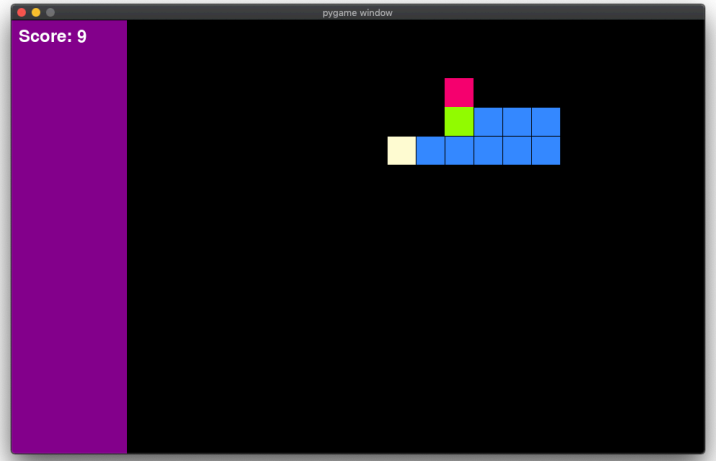
Step One: State Encoding

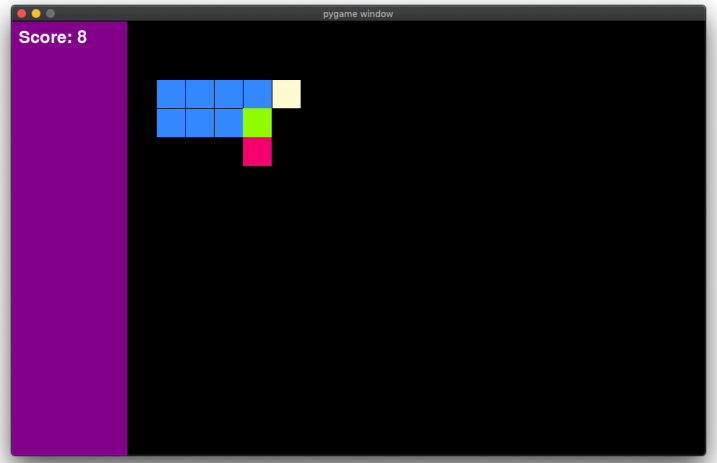
- Distance from food?
- Distance from food + body elements in between?
- Many many possibilities, must be cautious.
  - Bitmap of every square would lead to $2^{19+14=33}$ possible states

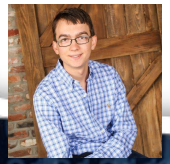


Desired Encoding

- Encode state in such a way that key attributes are encoded

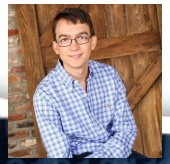






Decided Encoding

- 12-bit bitmap
  - (2^{12} possible states)
  - 2 bits: current direction
  - 2 bits: quadrant of food relative to head
  - 8 bits: safety value of each surrounding square from snake

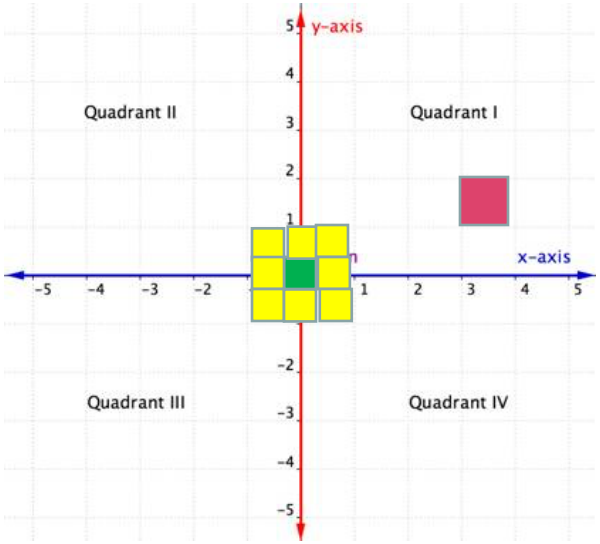


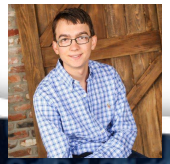
Encoding

Direction

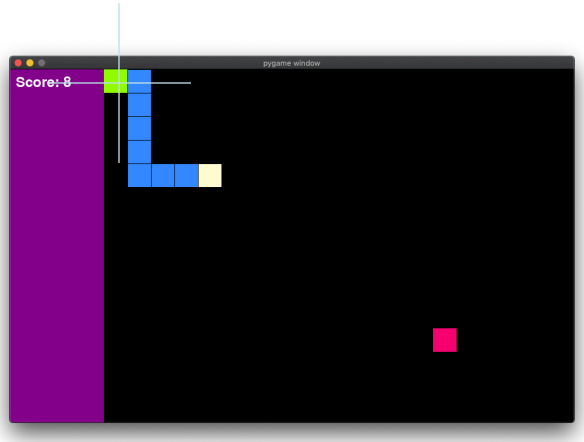
Direction	UP	LEFT	DOWN	RIGHT
Bit-Value	00	01	10	11

Quadrant

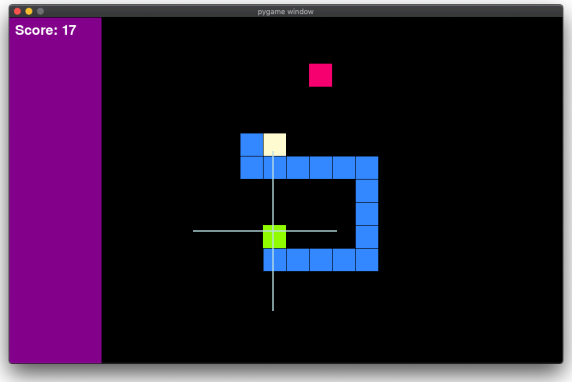




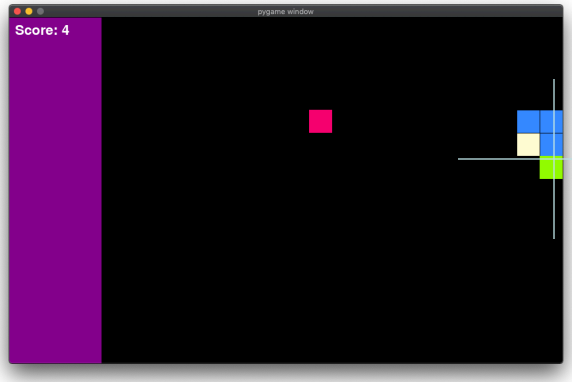
Examples



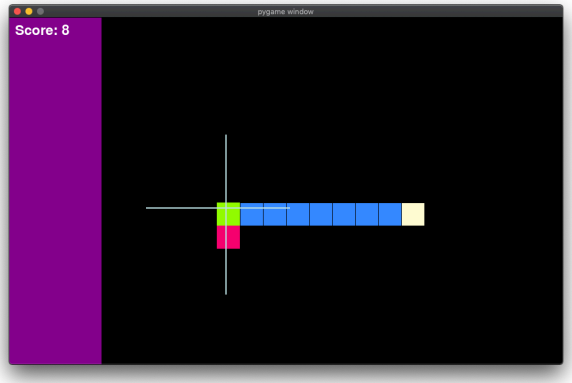
[01][11][11110111]



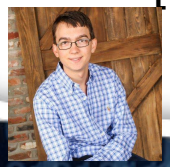
[00][00][00001001]



[10][01][00010111]



[01][11][00000010]

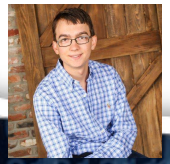


Step Two: Defining Reward

- Eat food = +1
- Die = -100
- Increased distance from food = -0.2
- Decreased distance from food = +0.1

Takeaways:

- Don’t die
- Get closer to the food

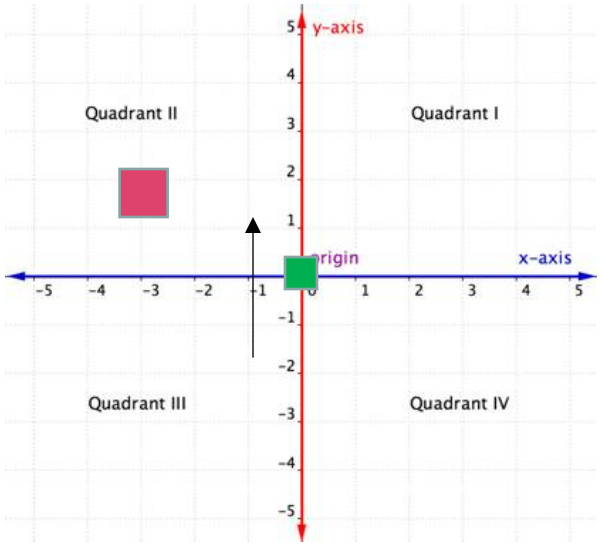


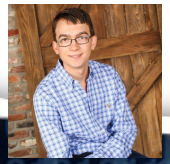
Sample Q-Table

	UP	DOWN	LEFT	RIGHT
000100000000	0.451250	NaN	-0.004920	0.000000
010100000000	0.132099	0.00	0.000000	NaN
001000000000	-0.020000	NaN	0.337603	-0.020000
111000000000	-0.002597	-0.02	NaN	-0.020000
100100000000	NaN	0.00	-0.007449	0.016829
...	...	...	...	...
000010010000	0.000000	NaN	0.000000	0.010000
110100010110	0.000000	-0.02	NaN	0.000000
100100000110	NaN	0.00	0.000000	0.006256
100000000111	NaN	-0.02	0.000000	0.000000
100000001111	NaN	-0.02	0.000000	0.000000

[69 rows x 4 columns]

- Why NaN?
- Random values?

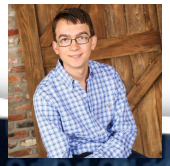




Step Three: Training

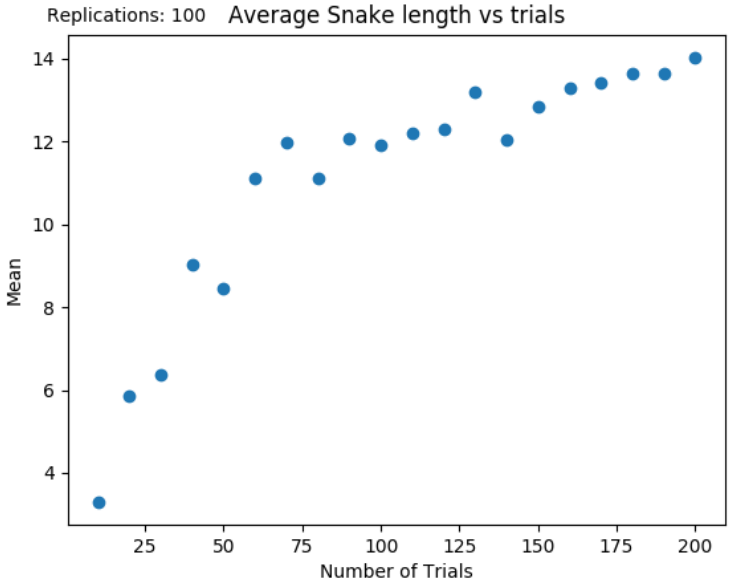
- Training:
  - 10 – 200 trials jumping by 10
  - Trial = training game

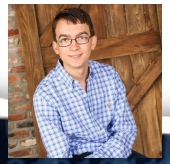
After training, “real” game is played and recorded. This is repeated for 100 replications. Each trial has



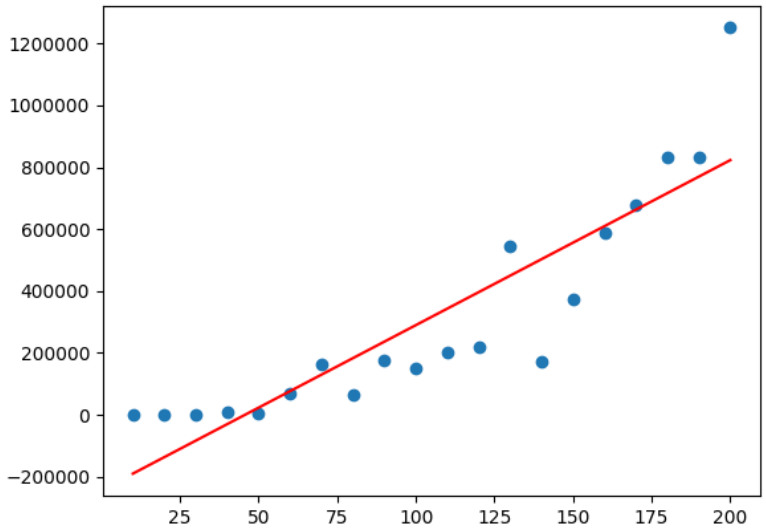
Analysis

- Logarithmic?

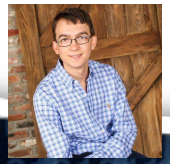




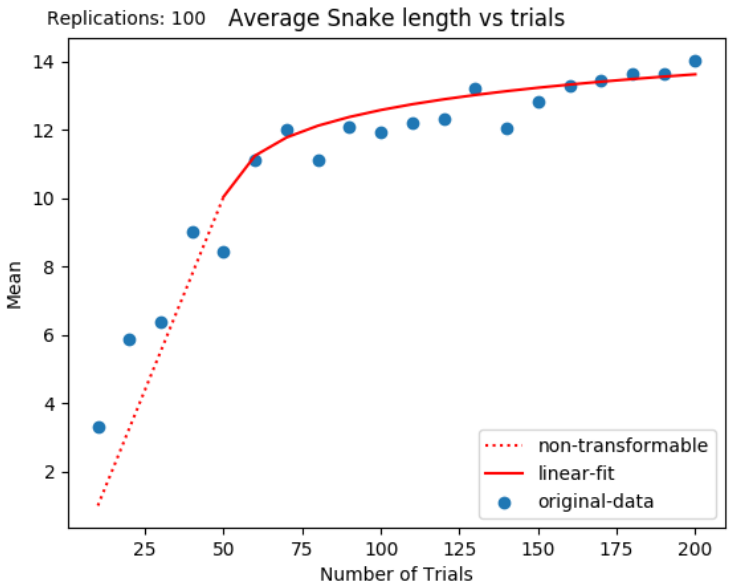
Linear Regression



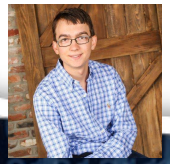
- Can’t perform linear regression on non-linear data
- Exponentiate logarithmic data to convert to linear data



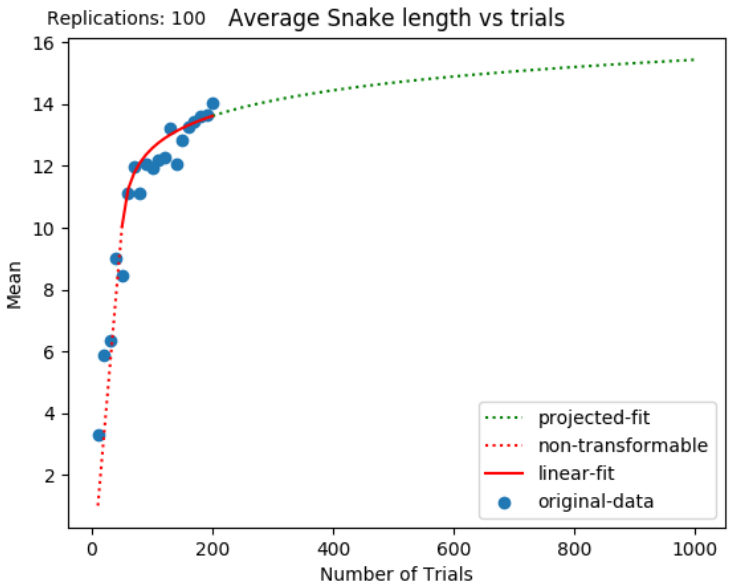
Logarithmic Regression



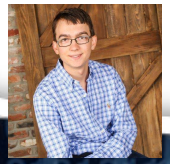
- Take natural log (ln) of linear line to convert back to logarithmic line
- R^2 of .77



Extrapolated Data

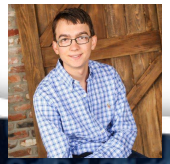


- Not collected data!
- How trendline would continue
- Future work?

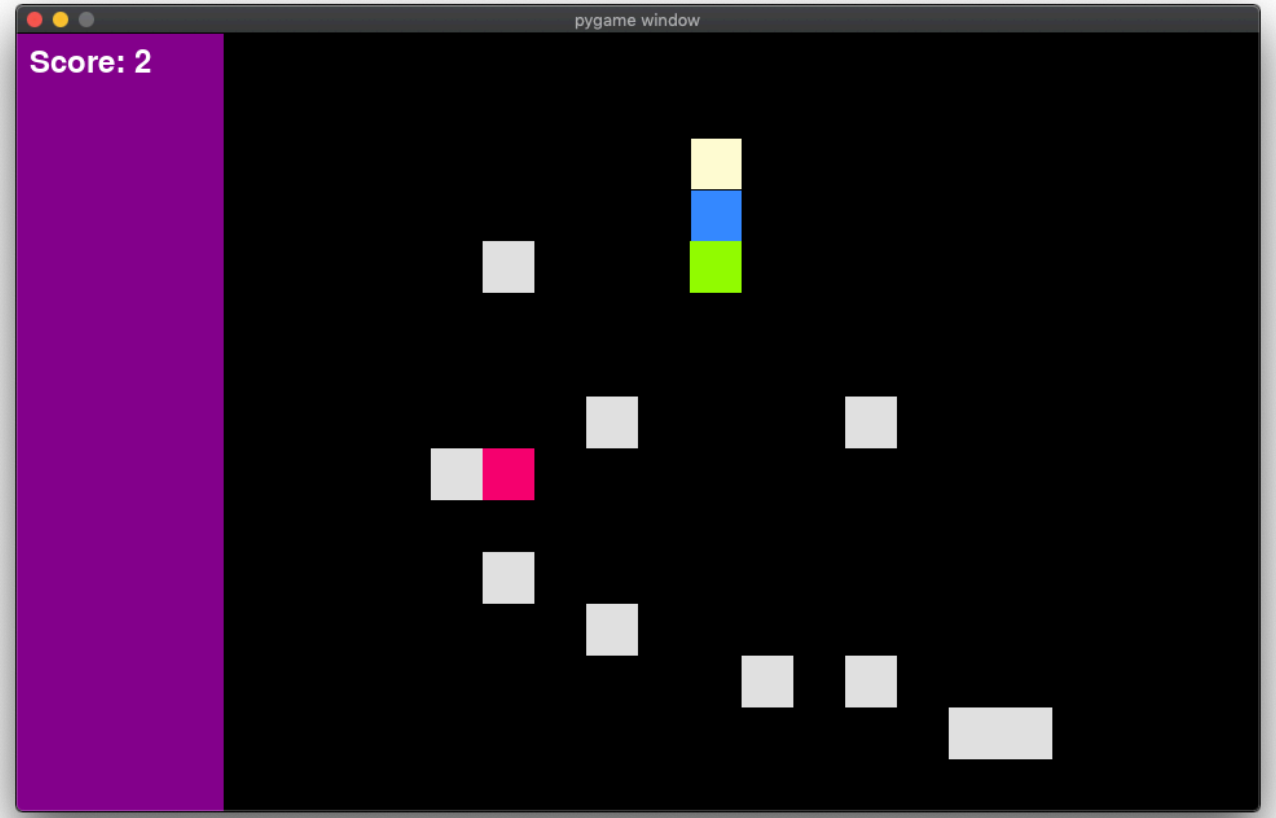


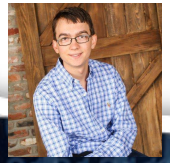
How to Make Snake Complex?

- More obstacles
- Multiple food blocks
  - Find food in order
  - Poison food
- Moving food



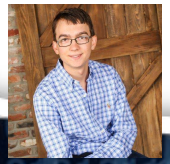
More Obstacles!





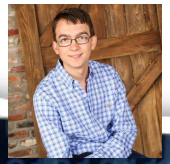
How is State Encoding Affected?

- Very little
- Obstacles are considered bad blocks (such as wall or body)



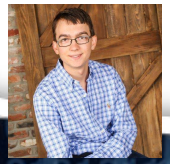
How is Reward Affected?

- Not at all
- Touching obstacle = death = -100

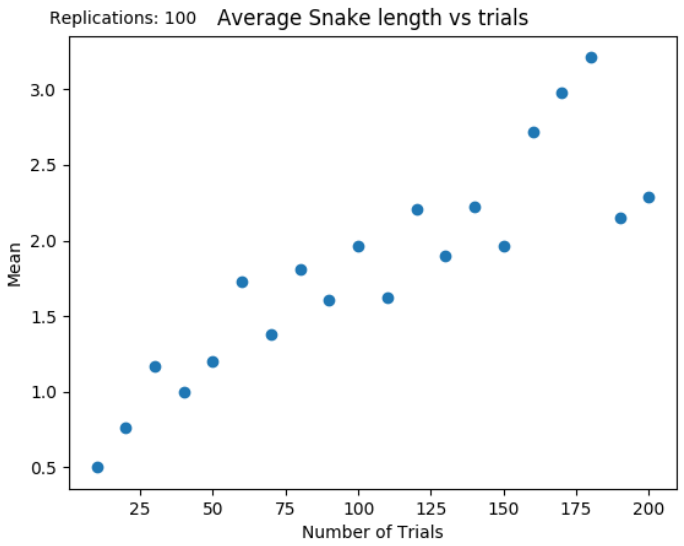


Training

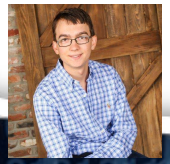
- Exactly the same as the first iteration in order to remain consistent



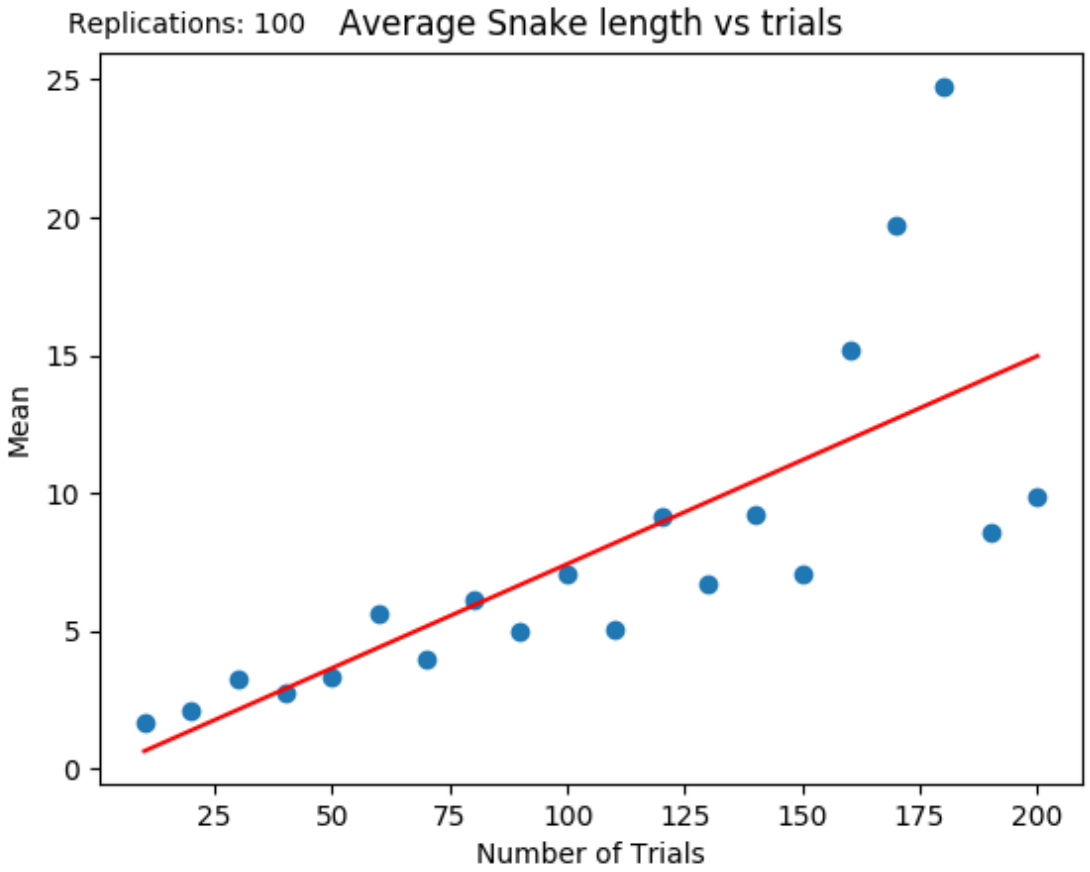
Raw Data

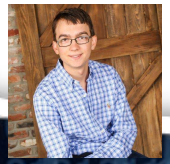


- Still logarithmic?
- Slower growth
- Smaller numbers



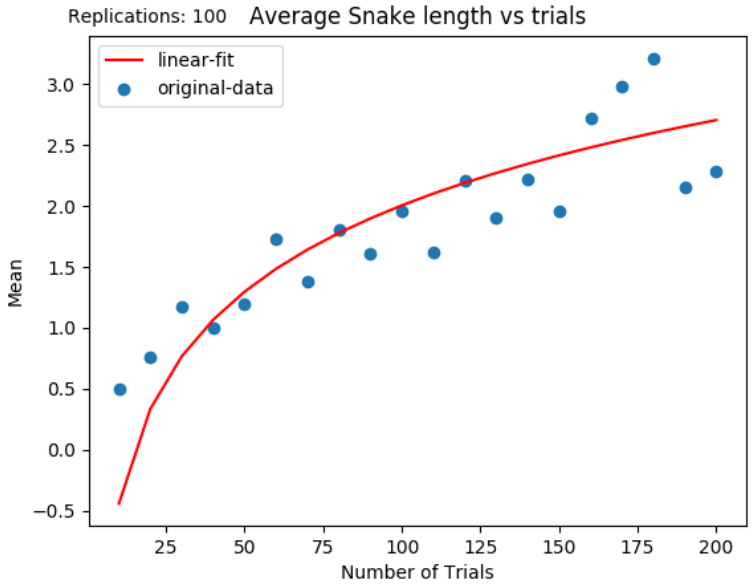
Linear Regression

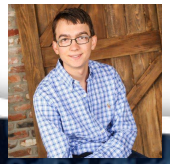




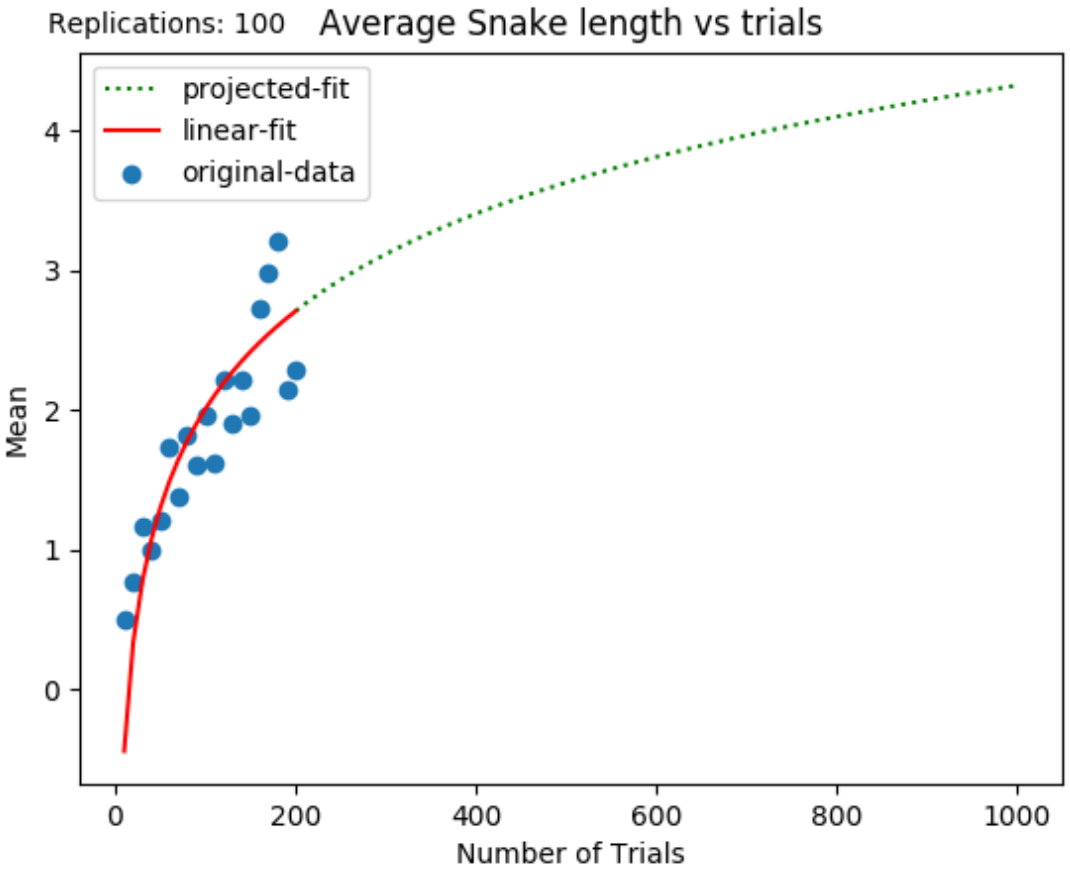
Logarithmic Regression

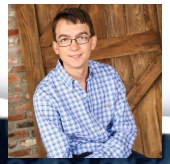
- R^2 of .67





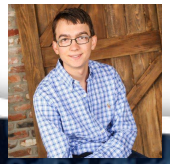
Extrapolated Data





Conclusions

- Original Questions:
  - How does complexity affect Q-Learning algorithm?
    - Complexity is harder
  - What are the limitations of Q-Learning?
    - Computational power: logarithmic lines are monotonically increasing; as long as the agent can train, it will learn



Acknowledgements

- CURO, for introducing me to research and giving me a platform to display my research.
- Matthew Chapman, I couldn't have finished this project without all of the explanations and hours long brainstorming sessions.
- Supa Mike, thank you for teaching me the countless new concepts that I could apply for this project, and keeping me on track with my research.
- UGA CS, for offering this opportunity to do research and follow a topic I was interested in.