Analyzing Varying Complexity on Q-Learning

Bailey Nelson
Department of Computer Science
University of Georgia
Athens, Georgia
Bdn81550@uga.edu

ABSTRACT
Artificial Intelligence (AI) and machine learning (ML) are growing in both importance and popularity. New feats are constantly being accomplished through the use of AI; a quick google search shows plenty of tutorials and articles related to creating and using a neural network for any sort of task. A popular ML algorithm is Q-Learning. Q-Learning is a reinforcement learning algorithm that can be used to teach an AI. It uses information from the current state and environmental information to choose the next state that results in the most reward. I look to view how the algorithm scales with the complexity of the game it is learning, in this case, Snake. As humans, we can have up to hundreds of hours of experience to pull from when learning how to play a game, but a computer must learn it all from scratch. Imagine if you were dropped in a game of Snake without any idea of rules or goal, the only thing known is how to move and what reward is associated with which move. That is how the AI starts. In order to give the AI all possible controls and knowledge of its state, Snake will be recreated using Python and the module PyGame. In addition, there will be supplemental blog posts at major milestones in the project. This will allow other students to follow my steps in order to learn more about neural networks and Q-Learning.

CCS Concepts
• **Computing methodologies~Reinforcement learning** • **Computing methodologies~Q-learning** • Computing methodologies~Neural networks

Keywords
Q-Learning; Machine Learning; Neural networks; Reinforcement learning; AI; Python; PyGame; Snake; Artificial Intelligence;

1. INTRODUCTION
1.1 The Machine Learning

In order to gain familiarity with machine learning algorithms and view the abilities of machine learning, I will be creating snake with varying levels of complexity. Through (re)creating this game, I will analyze how the complexity change will impact how the ML algorithm runs. Snake is considered a very basic game, but additions can be made to make it more complicated. The Q-learning algorithm that will be used uses the current state and information from the environment in order to choose the next choice that will give the most reward [1, 2, 3]. The algorithm intends to "learn" what to do by using a reward system that is assigned to each state. The AI then attempts to choose the choice that will result in the greatest reward. At first, the AI acts randomly and erratically to get sample data on what states it could change and what rewards it gets. As it acts randomly and die, it will eventually learn how to play the game the correct way.

1.2 Modifying Complexity

The goal of this project is to view how Q-Learning is impacted by increased complexity. The game of Snake is considered very simple, there is a snake with a body, and an apple. I plan to add elements such as obstacles such as rocks and pits. I also plan to add diversions such as numbered apples to be collected in a specific order.

1.3 The Documentation

While this project focuses on how the machine learning algorithm will help play the game, I am also focused on allowing others to recreate my efforts. My journey through machine learning will be documented through blog posts relating to progress, and tutorials. These will cover initial setup, creating the game, implementing the AI, and how the AI and game are modified in as the complexity changes. They will also cover my conclusions and finding at the end of each major milestone as well as the overall conclusion.

1.4 The Significance

This project is significant because for two reasons. It will be used for my own personal analysis of machine learning algorithms. Also, with my beginner-friendly follow up, anyone will be able to begin working on machine learning algorithms using python. I hope that my work can help other students understand that capabilities of Q-Learning and reinforcement learning.

2. Related Work
2.1 How to Teach an AI to Play Games

Mauro Comi [1] (2018) introduces Q-Learning and teaches an AI through the Q-Learning algorithm to play the game of snake using PyGame and Python. He then visualizes the progress made by the AI through a graph.

2.2 AI Learns to Play Snake Using Genetic Algorithm and Deep Learning

The YouTube channel Code Bullet [4] (2017) makes a Snake Game and uses the Genetic Algorithm to train the AI. The video documents his philosophy in how the AI can "think" and the rest of the video is watching how the AI evolves.

3. Method and Design (1/7 – 4/28)
3.1 Deliverable 1: Setup Post (1/21)
This deliverable consists of completing the setup as well as posting the setup process including what is necessary. This includes: Python, PyGame, pip, gym, and possibly others. This deliverable also includes setting up the blog that will house my posts for the project. This first post will serve as proof of concept that the blog works. For now, I plan on posting this blog on my Raspberry Pi but will likely include them on GitHub at the end.

3.2 Deliverable 2: Initial Basic Snake Prototype (1/31)

This deliverable consists of having an initial prototype of a working snake game in Python. A finished blog post is not required. This does consist of a working snake game with the following elements: score counter, movement, and food.

3.3 Deliverable 3: Basic Polished Snake Game (2/7)

This deliverable consists of the "finished" version of a simple snake game that is fully functional and playable. This will be accompanied with a blog post about the creation of the game. This blog post is not only a place to post source code, but also a place to detail design decisions.

3.4 Deliverable 4: AI Integration (2/21)

This deliverable consists of completing AI integration with the Snake game. This consists of the AI being able to play the game using the Q-Learning algorithm. A complete blog post is not required.

3.5 Deliverable 5: Conclusion Post (2/28)

This deliverable consists of the conclusion post from the snake game. This includes a wrap up, what I observed, what I learned, how the AI performed, and final thoughts. This includes data sets related to the AI with graphs visualizing the AI improvement over time. This deliverable also contains the setup for the next game, brick break.

3.6 Deliverable 6: Snake Revamp (3/6)

This deliverable consists of upping the complexity of the Snake game. The first complexity change will be to add traps and obstacles. Pits and rocks will be randomly added to the grid. If the Snake goes over a pit, they will die. If they Snake attempts to go over a rock, they are pushed to the left or the right based on their body orientation.

3.7 Deliverable 7: AI Revamp (3/13)

This deliverable consists of revamping the AI in order to deal with the new obstacles that were added in deliverable 6. This involves having the AI take into account the new additions of rocks and pits.

3.8 Deliverable 8: Conclusion Post (3/20)

This deliverable consists of a blog post related to adding the developments form deliverable 6, as well as how the AI was changed in order to adapt to the new challenges. Finally, the effectiveness of the AI from deliverable 7 is analyzed in how it performed in the state of increased complexity.

3.9 Deliverable 9: Snake Revamp (3/27)

This deliverable consists of a second revamp of the complexity of the Snake game. On top of the obstacles added in deliverable 6, another layer of complexity will be added. Now, there will be multiple numbered apples on the screen at once. The snake has to get these apples in the correct order, or the score goes down.

3.10 Deliverable 10: AI Revamp (4/3)

This deliverable consists of revamping the AI in order to deal with the new layer of complexity that was added in deliverable 9. The AI will now take into account which apple is the next apple to get while avoiding the other apples.

3.11 Deliverable 11: Conclusion Post (4/10)

This deliverable consists of a blog post related to adding the extra apples to the game, as well as the changes that were made to the AI. The second blog post is related to how the AI was able to handle the added complexity.

3.12 Deliverable 13: Conclusion Post & Overall Thoughts (4/24)

This deliverable consists of the final blog post relating to my findings over the course of the project. This will compare how the AI performed at the basic snake game and how it performed in the last iteration of the snake game.

4. Significance and Conclusion

This project can help show how complexities can affect the Q-Learning algorithm. There are already many examples of Q-Learning being used to play games [1, 4, 5], but these examples focus on how the Q-Learning algorithm impacts one single iteration of a game. I haven't found an example where the complexity of the game changes over time. This research can show if the Q-Learning algorithm scales up with growing complexity.

5. REFERENCES

Comi, Mauro. "How to Teach AI to Play Games: Deep Reinforcement Learning." *Medium*, Towards Data Science, 24 Mar. 2019, towardsdatascience.com/how-to-teach-an-ai-to-play-games-deep-reinforcement-learning-28f9b920440a.

[1] Comi, Mauro. "How to Teach AI to Play Games: Deep Reinforcement Learning." *Medium*, Towards Data Science, 24 Mar. 2019, towardsdatascience.com/how-to-teach-an-ai-to-play-games-deep-reinforcement-learning-28f9b920440a.

[2] Chanda, Kaustav. "Q-Learning in Python." *GeeksforGeeks*, 7 Feb. 2019, www.geeksforgeeks.org/q-learning-in-python/.

[3] G. Li, R. Gomez, K. Nakamura and B. He, "Human-Centered Reinforcement Learning: A Survey," in *IEEE Transactions on Human-Machine Systems*, vol. 49, no. 4, pp. 337-349, Aug. 2019.
doi: 10.1109/THMS.2019.2912447

[4] Bullet, Code, director. *Ai Learns to Play Snake Using Genetic Algorithm and Deep Learning*. *YouTube*, YouTube, 7 Dec. 2017, www.youtube.com/watch?v=3bhP7zulFfY.

[5] D, Soren. "Teaching a Neural Network to Play a Game Using Q-Learning." *Practical Artificial Intelligence*, 4 Sept. 2017, www.practicalai.io/teaching-a-neural-network-to-play-a-game-with-q-learning/.