Using The Q Learning Algorithm To Beat a Game

Mitchell Kolb
11670080
School of Electrical Engineering and Computer Science
Washington State University
Pullman, USA
mitchell.kolb@wsu.edu

Flavio Alvarez Penate
11689700
School of Electrical Engineering and Computer Science
Washington State University
Pullman, USA
f.alvarezpenate@wsu.edu

***Abstract*—Artificial Intelligence can be trained to find the optimal way to complete a task in a simulation of sorts. These simulations can include finding the fastest way to travel from one location to another, having a self-driving car, etc. When an AI is first introduced, it has no previous knowledge of the situation that it is in or the requirements expected of it. Hence, there's methods to train such AI through Q learning. Q learning is based on a reward and punishment system for good and bad behavior, respectively. This reward and punishment system is carefully selected among certain criteria that we want the AI to fit. After each iteration of the Q learning algorithm, the AI starts to be rewarded for good behavior and it looks for ways to reproduce the behavior that achieved this positive stimulant. After numerous iterations and time spent training the model, the Ai will have optimized behavior to achieve the tasks assigned to it. In order to prove the ability of Q learning in training AI, we will produce a video game that will train an AI to perform a certain task, such as avoiding certain objects, traveling towards a certain point, or going from one location to another in the fastest way possible through an obstacle course. This project's goal is not only to showcase the capabilities of Q learning, but to emphasize on its broad uses and the ability to be implemented into more than one situation. responses.**

Keywords—Q Learning, Artificial Intelligence, reward system, behavior, training.

I. Introduction

Training an Artificial Intelligence program to perform a task in the most efficient way possible can have many advantages. It could show the fastest route to get from point A to point B, teach a car how to drive itself, or show a non-player character in a video game that is controlled by the computer how to move so that it doesn't collide with walls, follows a certain path, or has the most effective combination of attacks, etc. The way that Q Learning starts out is with an artificial intelligence that knows nothing, rewarding good or correct behavior and punishing bad or failed behavior. For instance, if a non-player character is being trained using Q Learning to walk across a bridge, every movement that the artificial intelligence commits to that makes progress towards the other size is rewarded; meanwhile, any movement that results in getting off track, turning back, or falling off the bridge, is punished by the algorithm.

One of the difficulties that come with Q learning is deciding how to implement the reward system. While the punishment system can be easily defined as some action that we're trying to avoid, the reward system has to be carefully selected in order for the artificial intelligence to be able to make some progress through random actions when it begins. Hence, if one implements the only reward system for the bridge example as making it across the bridge, it is very likely that the artificial intelligence will never cross the bridge, since it is a much more distant goal than taking a step forward and rewarding each correct step individually. Thus, reward systems can be implemented for various aspects or types of behavior, each having a different reward value attached to them.

The objective of Q Learning is to find the policy that is optimal in the sense that the expected return over all successive time steps is the maximum achievable. Find optimal policy by learning the maximal values for each q pair. The Bellman optimality equation is used to calculate q*. The algorithm iteratively updates the q values for each state action pair using the bellman equation until the q function converges to q*. Once we have a optimal q function, q*, we can determine the optimal policy by applying a reinforcement learning algorithm to find the action that maximizes q* for each state.

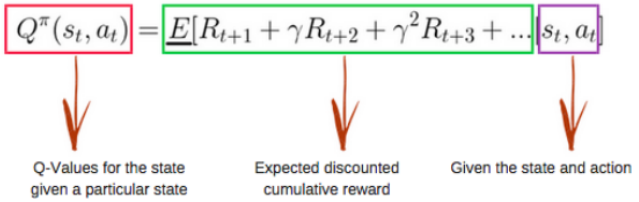
II. Literature Review

This project was inspired by a tutorial on an artificial intelligence lizard jumping tiles in a game that consists of tiles containing bugs, birds, or nothing in order to eat (or land on) the highest number of bugs possible. In this game, the artificial intelligence is using a reinforcement learning technique where it is rewarded one point for ending on a tile that has a single bug and ten points for ending on the maximum prize. Likewise, if the artificial intelligence ends on a tile that's empty, it loses one point, and if it ends on a tile that contains a bird, it loses ten points. Additionally, if the artificial intelligence stumbles upon the maximum reward or the maximum punishment, the game ends and that trial is over. The final goal of this game is to have the artificial intelligence lizard be able to reach the tile with the five bugs in the shortest number of moves possible, thus optimizing the q*. It uses a q table to store the q action pairs for every move that a particular lizard iteration takes. Over time it will use the data from this table to determine its moves and use it to choose smarter moves to maximize the most amount of points.

We were inspired to stick with this Q Learning model because we see lots of applications for this model outside of the lizard game. This model can be used for any game where you can let it use a set control scheme and reward system and let the model learn over many generations. We have looked into the math behind Q Learning and we found an article by Chathurangi Shyalika and she explains reinforcement learning and the six steps that make that work and the mathematical functions that allow Q Learning to take in data and store it to then determine in the future if the reward given by that decision is worth it.

III. Technical Plan

The techniques that we plan to use in the implementation of the Q Learning model is the Q-Table. This is a data structure that is used to calculate the maximum expected future rewards for every action at each state. The table mathematically shows a layout of the best decision to reach the goal. The Q-Table uses the Q Learning algorithm which uses part of the Bellman equation.



[2]https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/

This equation when used in conjunction with the Q-table can determine whether a certain action is "rewarding" or not by the resulting value. Using this equation over many different generations will allow the network to highlight the best decisions to maximize Q to get the highest rewards. Another part of our plan is to apply this to a game we will have to build from scratch. This game will be determined at a later date.

IV. References

[1] “Exploration vs. Exploitation - Learning the Optimal Reinforcement Learning Policy.” YouTube. YouTube, October 10, 2018. https://www.youtube.com/watch?v=mo96Nqlo1L8.

[2] freeCodeCamp.org. “An Introduction to Q-Learning: Reinforcement Learning.” freeCodeCamp.org. freeCodeCamp.org, August 9, 2018. https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/.

[3] “Playing Geometry Dash with Convolutional Neural Networks.” Accessed February 20, 2023. http://cs231n.stanford.edu/reports/2017/pdfs/605.pdf.

[4] “Q-Learning Explained - A Reinforcement Learning Technique.” YouTube. YouTube, October 5, 2018. https://www.youtube.com/watch?v=qhRNvCVVJaA&list=TLPQMjAwMjIwMjP5wYy7pRhaxg&index=2.

[5] “Q-Learning Explained - A Reinforcement Learning Technique.” YouTube. YouTube, October 5, 2018. https://www.youtube.com/watch?v=qhRNvCVVJaA.

[6] Shyalika, Chathurangi. “A Beginners Guide to Q-Learning.” Medium. Towards Data Science, July 13, 2021. https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c.