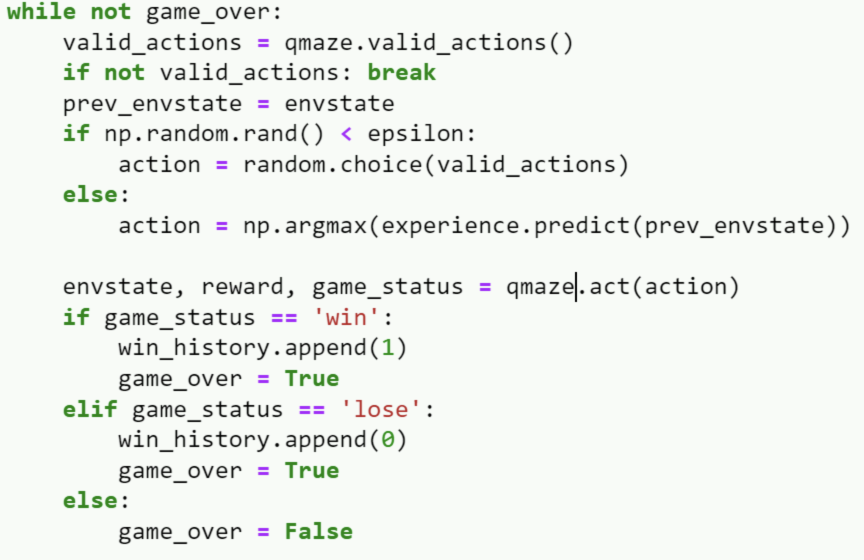
Branden McCusker

December 11 2022

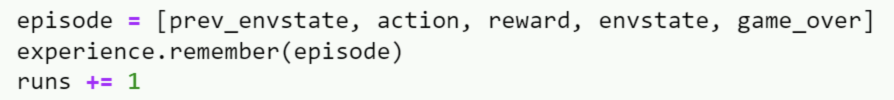
Design Defense

The neural network implemented within the Treasure Hunt Game takes very similar processes to how a human may interact with the scenario in real life. To clarify exactly what is going on in this program; The pirate needs to get the treasure, which in other words mean Point A needs to get to Point B, which is unknown. There are obstacles in the way, although the pirate does not know where they are until it is reached, which the pirate must then start over with the knowledge that this obstacle exists at the location. It is the neural networks job to not only reach the goal, but optimize the path taken until the fastest route is found.



This code shows the process of the pirate starting at the beginning position and “randomly” choosing the next point of movement. If the pirate finds the treasure or loses by touching an obstacle, the program will sort them into such categories for learning.

This process can be easily replicated in real life, disregarding minor human errors such as not always moving at a constant speed. Similar to the actual program, we can have a 8x8 grid where a person starts on one square in the corner. They move one square at a time, and they must find where the goal is. If they touch an obstacle, they must start over. Eventually, this person will find the goal, and over time they will also discover the locations of all obstacles, and finally calculate the fastest route to reach the goal.

Because of this, neural networks are very similar to humans. Given that an algorithm has infinite amount of time to solve the problem, trial and error is the sure way to end up with the most optimal route. This program uses reinforcement learning, which is very similar to how a normal person may tackle this scenario. I added deep Q-learning through “experience”, which allows the algorithm to make educated guesses on the next possible best solution before actually running it. This allows the program to find the most optimized route faster than without experience.

This code shows how the network remembers each important aspect of the previous run and implements the knowledge into each run afterwards.