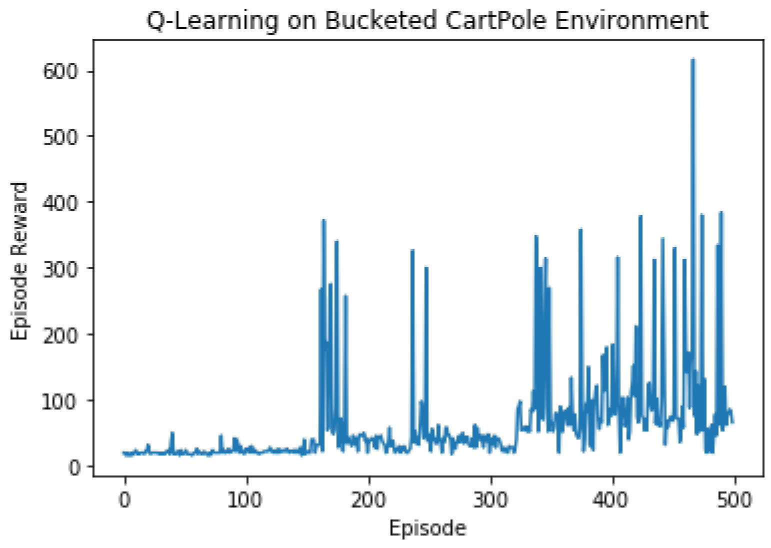
Unless cited, code is my own. I also referenced some tutorials on evolutionary neural networks and keras documentation as writing a neural network from scratch was time consuming. ChatGPT and geeksforgeeks were used in plotting and Python help.

1) Create a Q-learning agent that learns to solve the "Cart Pole" environment. The agent should balance the pole for 100 time steps.

How will you handle the continuous state space?

I discretized the state space into 3 buckets each for cart position and velocity and 6 buckets for pole angle and pole velocity. This allowed me to “fake” a discrete space for solving.



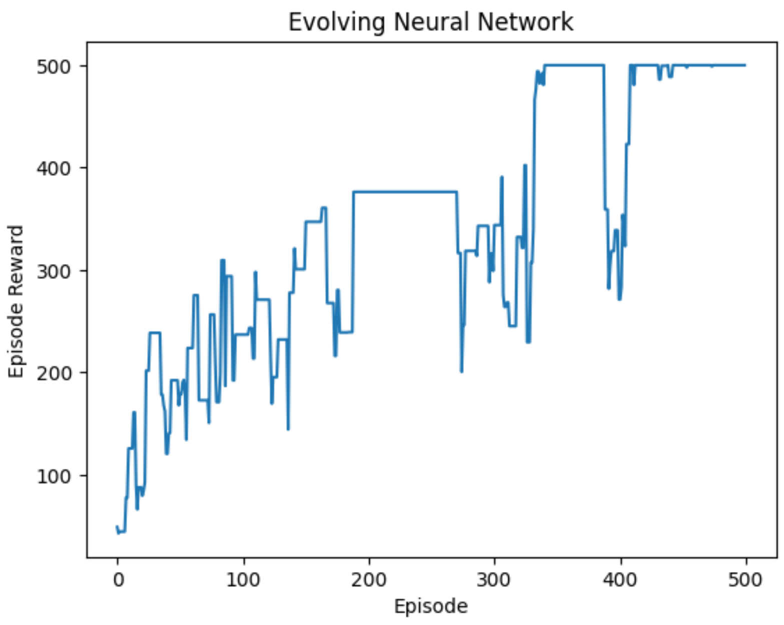
2) Evolve a neural network to solve the previous task.

What will you use for your evaluation function?

The network is evaluated by running the cartpole simulation ten times and averaging the reward across all runs.

What mapping should the network learn?

The network learns the mapping of a point in the observation space to an action. The actions are chosen based on the highest q value of the actions which incorporates the reward. Hence the network improves over time.



Algorithm Description

Q learning: I first discretized the state space so there were 3 buckets each for cart position and velocity and 6 buckets for pole angle and pole velocity. Then following the same format as standard q learning, I learned for a specified number of epochs and in each I reset the environment, and ran the following for 500 timesteps: I picked an action using either the greedy approach on the discretized state space or a random sample, then stepped in the environment, discretized the state based on the step, find the best action using the q table for that state, and then update the q table using the policy. Then update the state and repeat. It's very similar to standard q learning except the state had to be discretized.

Evolving a neural network

I first created an initial network using some data from a random agent. Then I made a copy of the network and randomly modified some of the weights. Then both of these networks were run in 10 training cycles and the rewards were averaged. The better performing network was kept, and the cycle repeated. This is a very basic form of an evolutionary network where the initial population is 1. A true evolutionary network would start with more than one initial network in the population.

Comparisons

The Q Learning algorithm performed better with more and more episodes as it better learned the best actions to take while taking very little time to do so. With the specific bucketing I used there were only a few states so this probably helped. If I had increased the number of buckets the algorithm would've been more computationally expensive but performed better. The q learner also learns with each iteration. Q learning is also a lot easier to implement than the neural network solution.

The evolved neural network took a very long time to finish but performed much better sooner and there was less variation when compared to the q learner. Here however, the method of mutation was slower and with an initial population of 1, sometimes the networks evolved in the wrong direction but they managed to keep a score above 100 after the first few episodes. If the initial population of networks contained more than 1, the reward would have consistently improved as only the higher performing networks would have been kept. I would also have only evaluated fitness on the newly created networks instead of all of them to speed up slightly, but this would still be slower than Q learning. The tradeoff is that the neural network can’t change with each iteration as it can’t learn online like Q learning does with each iteration.

Both perform decently in the CartPole Environment but the evolved neural network learns in fewer episodes but is computationally more expensive.

Sources:
https://nandakishorej8.medium.com/part-2-evolutionary-algorithms-for-reinforcement-learning-solving-openais-cartpole-318aaef8a6eb
https://theobservator.net/neural-network-for-open-ai-cartpole-v1-challenge-with-keras/