REINFORCEMENT LEARNING WITH AI - PHASE 1
GROUP PROJECT
CPSC 4371 FALL 2019

Group Members: Sarah Baker, Caleb Lipko, Kiarra Jackson, Eli Taylor
Instructor: Doerschuk

REINFORCEMENT LEARNING

Reinforcement learning teaches the agent how to take appropriate action in a given situation by trial and error. If an action produces a good result the agent will be positively rewarded, and if the end result is poor the agent will be rewarded negatively. This method keeps the agent on the path of attempting to reach for positive rewards and prevent repeating past mistakes. The reinforcement learning algorithm searches over the space of input states for the action to take to maximize the reward using sensors. The algorithm also determines states, actions, rewards, and policy. In this case, the algorithm would determine the following:

States: The position of the curve starts and finishes.
Actions: Move left, Move Right
Rewards: Positive closer to the top of the mountain, Negative further from the top of the mountain

PROBLEM

Mountain Car is a classic reinforcement learning problem. In this environment, a small car with a weak engine is placed in the middle of a valley between two steep hills. The agent needs to move the car at the right velocity at its current state to reach a flag at the top of the right hill. The car is constrained by the steepness of the hills, gravity, and its weak engine. The problem is: How do we use our knowledge in reinforcement learning in tandem with OpenAI GYM to get the car to learn how to reach the flag at the right hill?

METHODOLOGY:

In order to implement a solution to our problem, our agent must implement the Q-Learning algorithm to reach the flag. Q-Learning is a method of machine learning that stores the expected rewards in a Q-table. The object in question will use the Q-table to determine the best reward using its current status. The Q-Learning update is when the reward values in the Q-table are updated to produce better results.

DATA:

State Space:

The state-space of the mountain car is all points on the valley and hills between -1.2 and 0.6 and multiplied by its particular velocity. Since the state space is between -1.2 and 0.6 at different velocities, that means the state size is too large to be able to efficiently determine the outcome of this project. For simplicity, the state space will be discretized for this project.

Action Space:

0	1	2
Move left	No-Op	Move Right

SOLUTION:

In order to solve the problem of the Mountain Car using Reinforcement Learning, we used code from the blog, Towards Data Science "Getting Started with Reinforcement Learning and OpenAI Gym". The program runs 5,000 episodes of the Mountain Car learning to reach the top of the hill. The program from the blog returns the average reward per seed and animation for the last few episodes. MountainCar-v0 also requires a change in state size by discretizing the state space. We also added onto the code to find which hyperparameters would give us the best outcome of rewards for 3 separate seeds (0, 1, 2) of the environment.

The Code:

The agent is first created in the Mountain-Car environment and invokes reset() to start the car in a random state in the environment. Then the Q-Learning() function is created. In the Q-Learning( ) function we first 1) determine the size of discretized state space, 2) initialize the Q-table, 3)Initialize variables to track rewards, 4) Calculate episodic reduction epsilon (because we are using greedy epsilon) 5)Run the Q-Learning algorithm 6)Track rewards, and finally 7) Use grid search to find the best parameters for the agent to be most successful for 3 separate seeds.

The Q-Learning() function contains 6 parameters:

Parameter name	**Parameter Type**	**Parameter used**	**Description**
env	Environment	MountainCar-v0	The OpenAI Environment of MountainCar
learning	double	0.2, 0.4, 0.6	Learning rate used in the Q-Learning algorithm
discount	double	0.2, 0.5, 1.0	Used for adjusting Q-value for current stated
epsilon	double	1.0, 0.8, 0.6	Used for finding best reward given the current state
min_episodes	int	0	The minimum amount of episodes used for episodic reduction
episodes	int	3000	Amount of episodes being run

DATA:

The data received back from the code gives the seed number, the associated starting space, and then the average reward at the end of each 5000th episode. After the return of the average reward, the data includes which hyperparameter was used for that particular set of episodes.

```
Episode 5000 Average Reward: -198.929
0.2 0.5 1.0
LEARNING RATE | DISCOUNT FACTOR | EPSILON
-
```

Example of returned data for seed 1

CONCLUSION:

In conclusion, we have demonstrated how reinforcement learning can be used to solve the OpenAI Gym Mountain Car problem. To solve this problem, it was necessary to discretize our state space and make some small modifications to the Q-learning algorithm. Q-learning is a powerful technique believed by many researchers as the most promising lead towards artificial intelligence. Using this technique we found that within the given 5,000 episodes the mountain car was able to reach the top of the hill much of the time. The average rewards always ranged between (negative) 168-200 in our testings. Each reward would add -1 each time it successfully reaches the top of the hill, therefore, resulting in the negative reward numbers. There are 28 different combinations per seed and you will see a change in the rewards between each of these combinations. In the first seed (seed [0]), we can see it get down to -172 in one of the instances. For seed 2 (seed [1]) we can see it get as low as -168, and finally, in the 3rd and final seed (seed[2]) we can see it get as low as -176. The Q-Learning update is when the reward values in the Q-table are updated to produce better results. Through reinforcement learning, we saw how the weak engine that was not able to get the mountain car up the hill was made possible. With the proper Q-Learning, Q-table, random seeding, and grid search it was able to make it to the top of the hill with its momentum of moving back and forth. This is just one of the many environments available to users in Open AI Gym. By continually modifying and building on the Q-learning algorithm, it should be possible to solve any of the environments available to users of OpenAI Gym. Nevertheless, as with everything, the first step is learning the basics. This is what we have succeeded in doing throughout this project.

REFERENCES:

Code was largely borrowed from:
https://towardsdatascience.com/getting-started-with-reinforcement-learning-and-open-ai-gym-c289aca874f