Sam Willenson (shw58)
ECE 5242

Project 4 Writeup (Reinforcement Learning)

Introduction

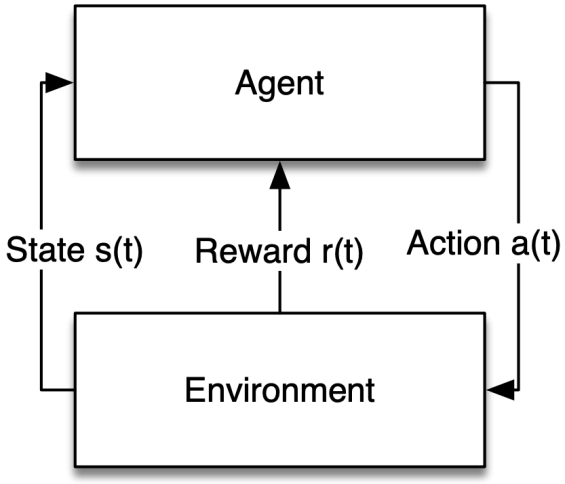
In this assignment, we focus on implementing Q-learning and REINFORCE algorithms to learn optimal policies for MDP environments. Focusing on value iteration and Q-learning, we see how fast our algorithms converge to the optimal Q values.

Problem Formulation

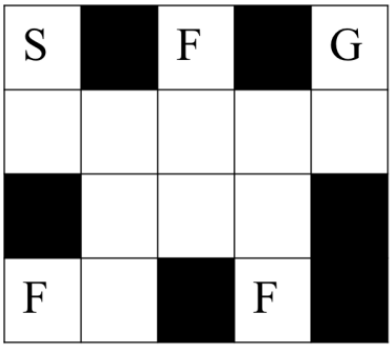
How can we implement value iteration to find an optimal policy and optimal Q values in a maze? How do we do this with Q-learning? How can we implement REINFORCE and Q-learning in the MountainCar environment?

Technical Approach (Part 1)

These reinforcement learning problems are modeled as Markov Decision Processes (MDPs), a cycle which can be seen in the figure below.



Part 1: Value Iteration



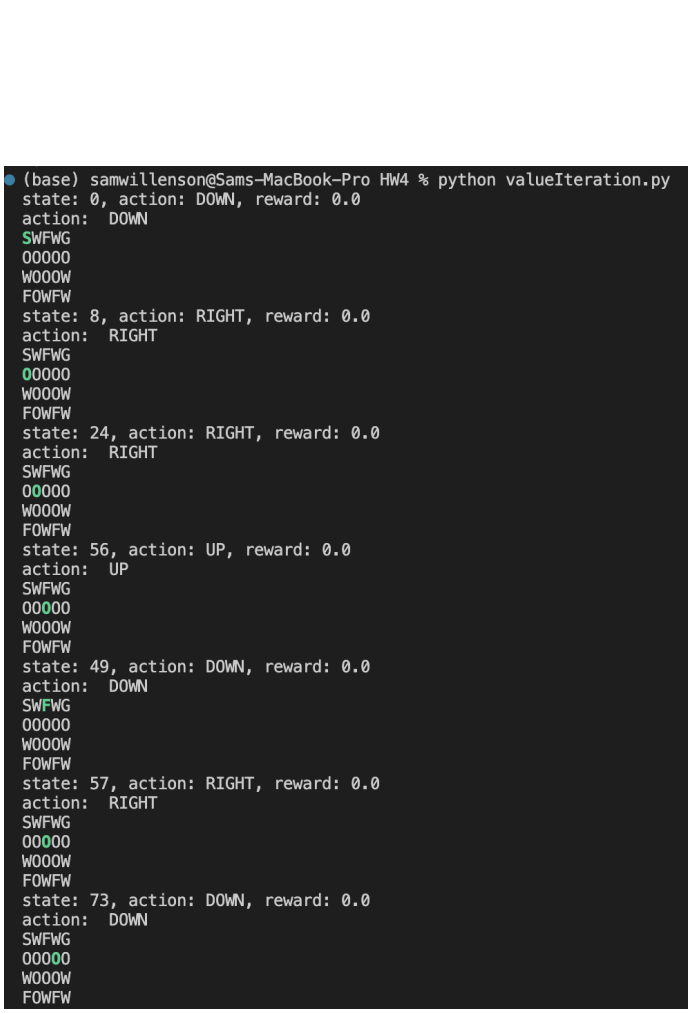
For the above maze, the goal in this step is to find the optimal path to the goal (S to G), with each F representing the location of a flag, where if the agent collects the flag it receives a reward.

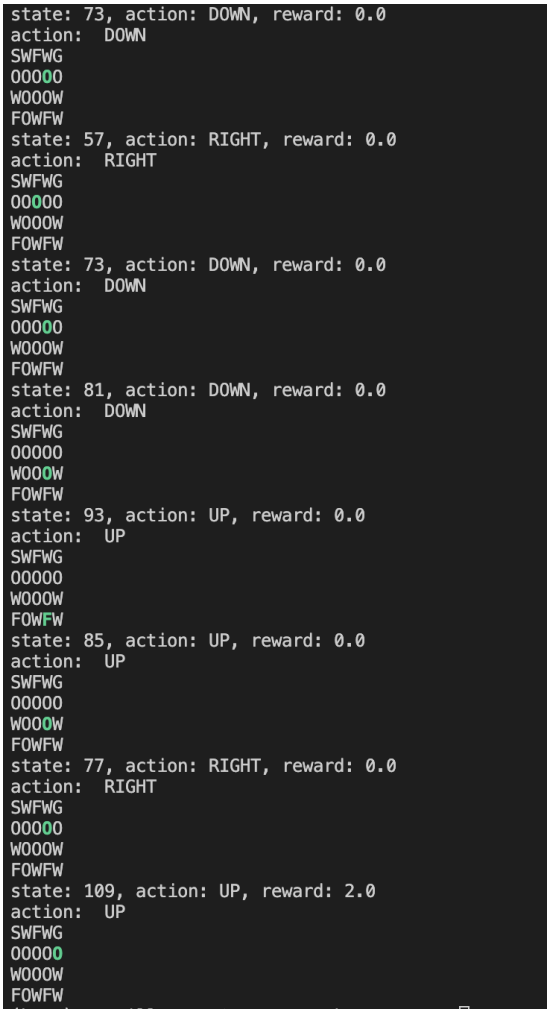
$$v_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_*(s') \right], \forall s \in \mathcal{S}$$

For each state I assign a value to v, updating it with the above equation. I make my discount factor 0.9, and use a theta = 1.6 (the smallest number I could use while still achieving convergence).

Manually looking at this maze, I can see that there are four possible paths to take:
No flag route: 6 steps (Down, Right, Right, Right, Right, Up)
1 flag route: 8 steps (Down, Right, Right, Up, Down, Right, Right, Up)
2 flag route: 12 steps (Down, Right, Right, Up, Down, Right, Down, Down, Up, Up, Right, Up)
3 flag route: 18 steps (Down, Right, Down, Down, Left, Right, Up, Up, Right, Up, Down, Right, Down, Down, Up, Up, Right, Up)

Results (Part 1)





Above is my optimal policy (left image first then right image, state 73 on the top of the right image is repeated from the left for clarity of order). We can see that the optimal policy is following the two flag route of 12 steps. Here we see 14 steps, but this happens because the down step (at the top of the right image) slips and goes to the left, requiring 2 extra steps to make up for this slip.

Technical Approach (Part 2)

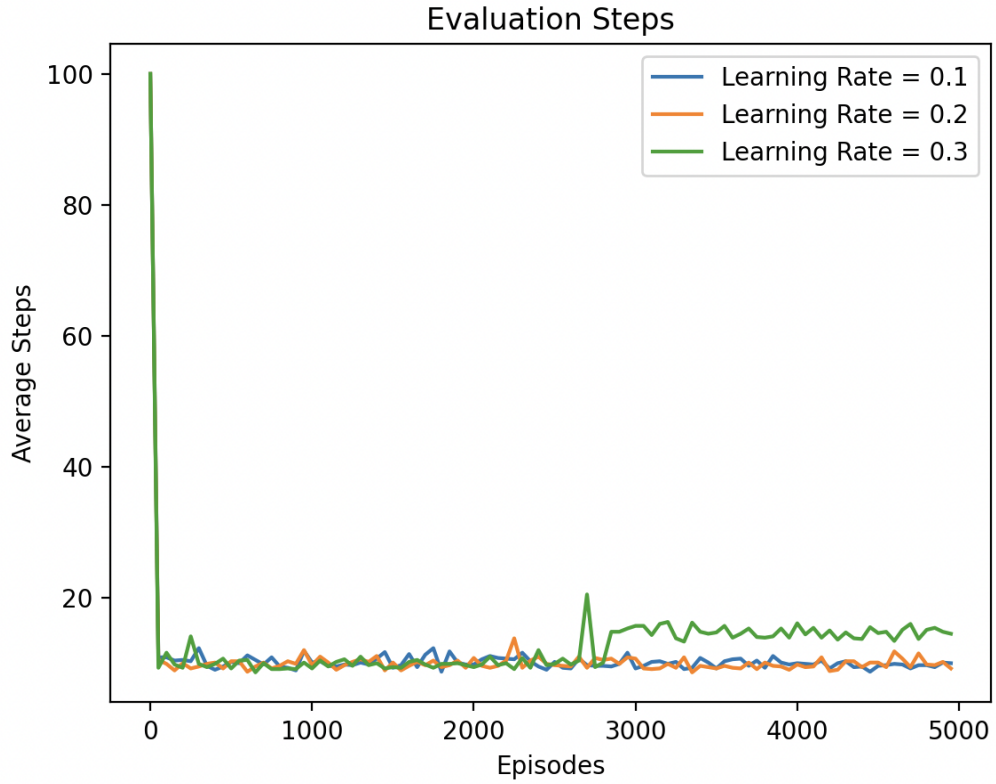
In this part we focus on Q learning, implementing this on the maze environment. In Q learning, we use a function approximator to estimate the action-value function.

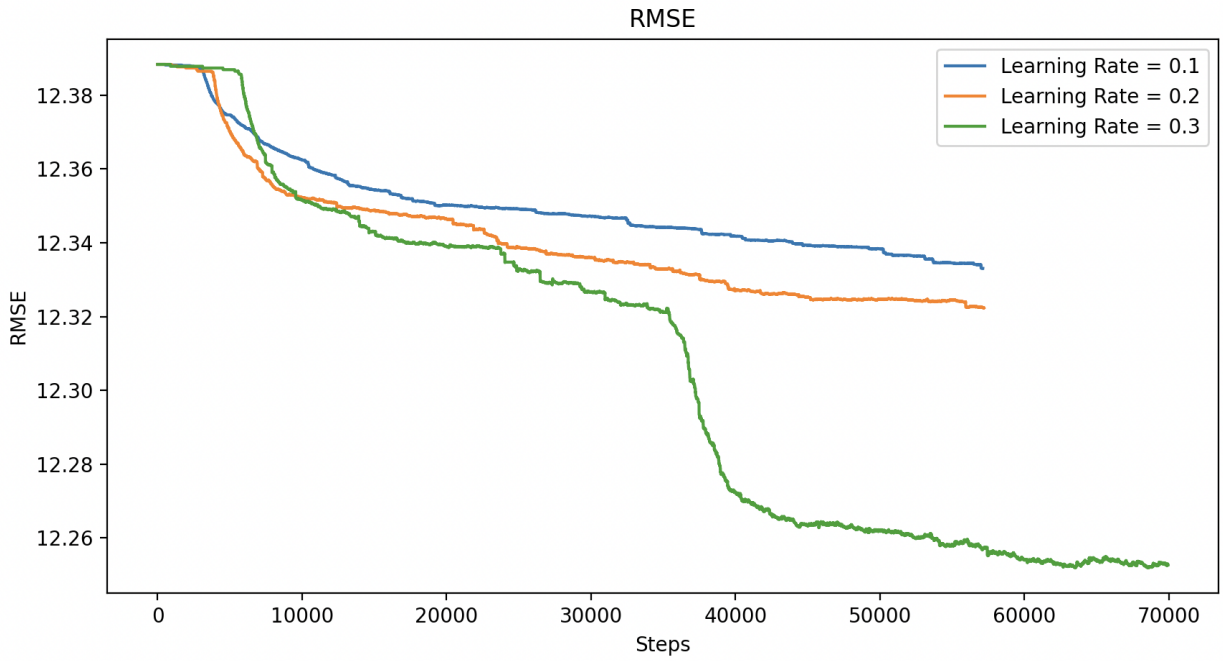
We want to find a Q-function that satisfies the Bellman Equation:

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s',a') | s, a\right]$$

In my code I vary the learning rate to see how that affects my learned Q values. I use an ε-greedy approach where with probability ε, a randomly selected action is performed, and with probability 1 - ε, a greedy action is performed. I experiment with learning rates of 0.1, 0.2, and 0.3.

Results (Part 2)


Evaluation Steps
Learning Rate = 0.1
Learning Rate = 0.2
Learning Rate = 0.3
Average Steps
Episodes
100
80
60
40
20
0
1000
2000
3000
4000
5000


RMSE
Learning Rate = 0.1
Learning Rate = 0.2
Learning Rate = 0.3
RMSE
Steps
12.38
12.36
12.34
12.32
12.30
12.28
12.26
0
10000
20000
30000
40000
50000
60000
70000

Discussion

My algorithm is working as expected in part 1, finding an optimal path through the maze. It takes the 2 flag approach and finds its way to the goal, dealing with the slips accordingly when they occur.

My results from part 2 show that as I increase the learning rate from 0.1 to 0.3 , my RMSE error reaches lower values. In terms of steps, they are all fairly similar - though a learning rate of 0.3 trends slightly up at around 3000 episodes. The rewards are all very similar, but the average learning rate = 0.3 jumps up to 2 at around 3000 episodes, which coincides with the amount of steps taken for this learning rate.