Assignment 2 - Deep Q-Networks

Part II

Implementing DQN

Deep Q-network (DQN) is a reinforcement learning algorithm that utilizes a deep neural network to solve tasks through trial and error.

Experience Replay:

We store agent experiences in a replay buffer, then randomly sample batches from the buffer to train the network. Doing so gives us the benefit of…

- Breaking correlations in data
- Learn from all past policies (catastrophic interference)

We implement a Target Network, in which we calculate target Q values (i.e. predicted Q-values) We sync this network with the policy network every 3-5 episodes.

Target Network:

- Stabilizes learning
- Reduces the effects of the moving target problem

We will test the implemented DQN on the following environments:

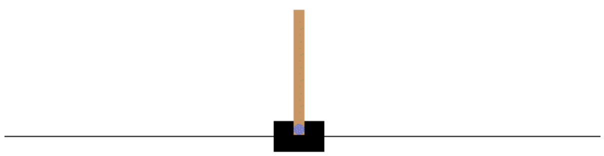
Warehouse Robot (assignment 1):

Goal: Robot is able to successfully pick up and deliver the package in the correct locations with the highest reward possible
Observation Space: A 6x6 array representing the respective locations of obstacles, the package, the dropoff point, and our agent.
Action Space: agent can go up, down, left, or right corresponding to integers 0-3.
Rewards:

- Every timestep: -1 to incentivize speed
- Drop off with no package: -100
- Drop off wrong location: -10
- Drop off with package in correct location: +100
- Pick up on wrong location: -10
- Pick up on correct location: +40

- Running into an obstacle: -20
- Trying to escape bounds: -25

Cartpole-V1:



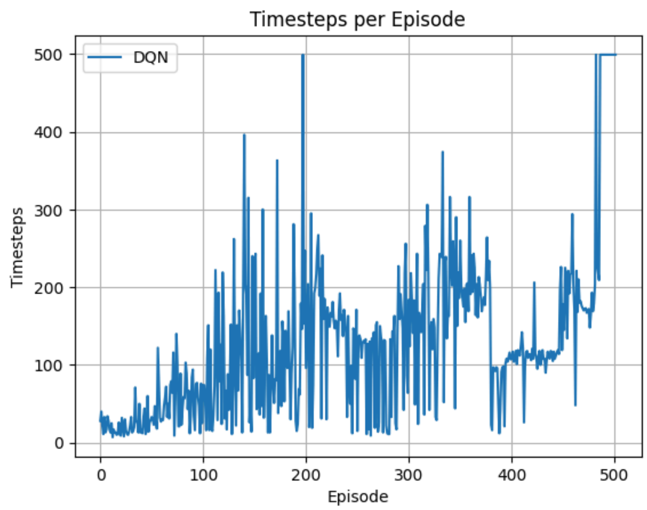
Goal: Balance the Pole by moving to the left/right of the cart

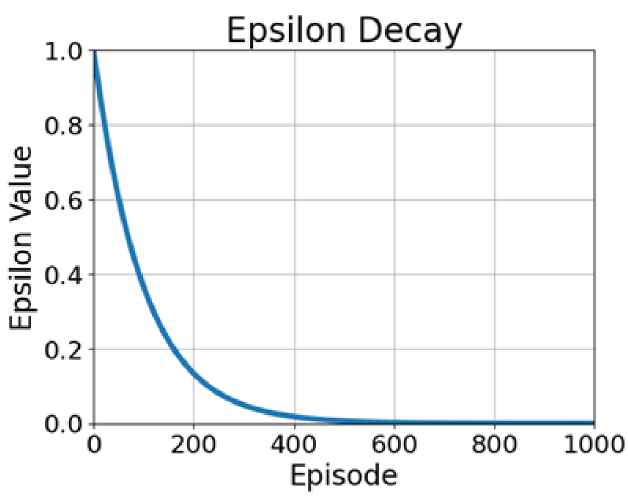
Observation Space: an array of size 4 with values corresponding to positions and velocities of the cartpole

Num	Observation	Min	Max
0	Cart Position	-4.8	4.8
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -0.418 rad (-24°)	~ 0.418 rad (24°)
3	Pole Angular Velocity	-Inf	Inf

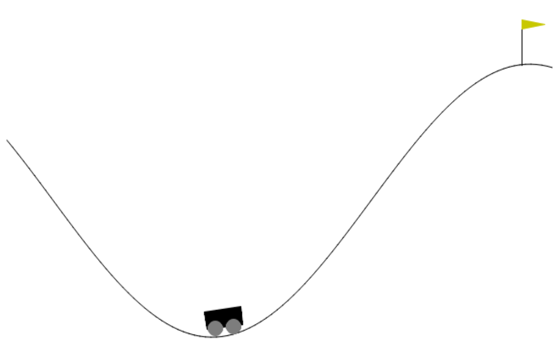
Action Space: Integers 0 and 1 which correspond to the following actions
0: push the cart to the left
1: push the cart to the right

Rewards: a reward of +1 for every timestep taken, including the termination step, is allotted. The threshold for rewards is 475.





MountainCar-v0:



Goal: accelerate the car to reach the goal state at the top of the right hill

Observation Space:

Num	Observation	Min	Max	Unit
0	position of the car along the x-axis	-1.2	0.6	position (m)
1	velocity of the car	-0.07	0.07	velocity (v)

Action Space: 3 actions corresponding to the following integers
0: accelerate to the left
1: don't accelerate
2: accelerate to the right
Rewards: The goal is to reach the flag placed on top of the right hill as quickly as possible, as such the agent is penalized with a reward of -1 for each timestep to incentivize speed.

3.
Grid World:
Cart Pole:
Mountain Car:

4.
Grid World:
Cart Pole:
Mountain Car:
5.

Part III

Improving DQN

One of the improved versions of DQN that we can implement is DDQN, or double DQN. in which we utilize the

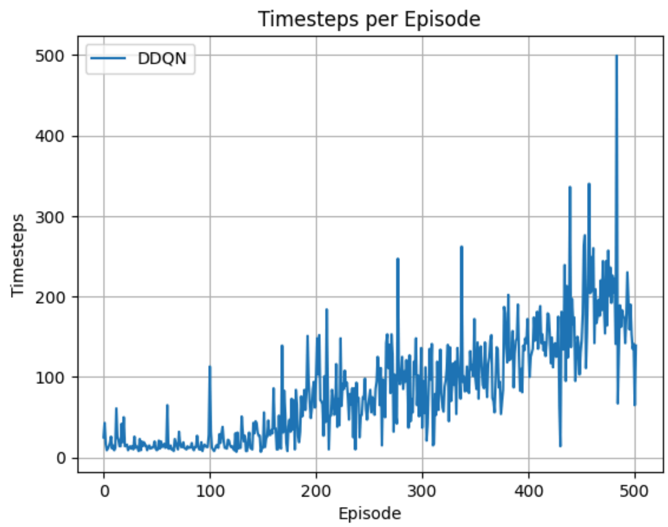
Benefits to DDQN:

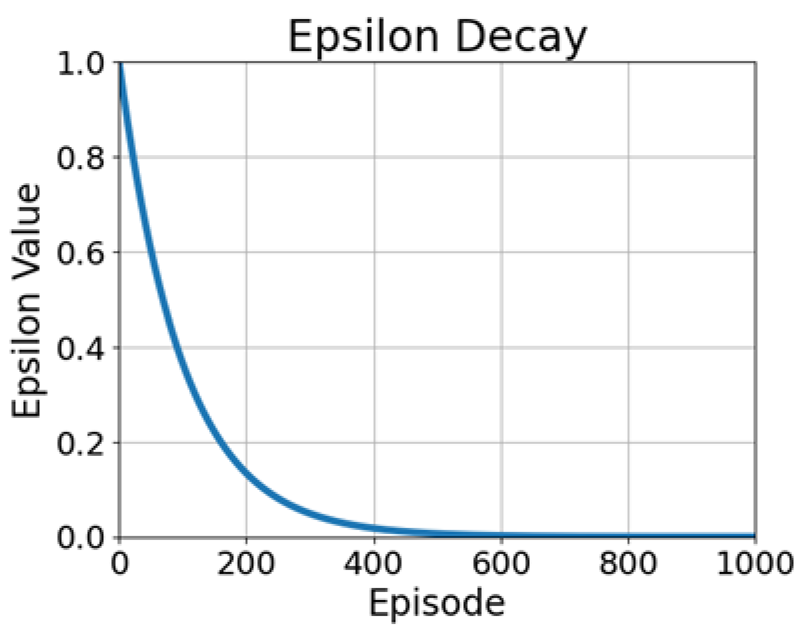
- Helps reduce overestimation bias, through the use of the second network.
- Utilizes the DQN algorithm without additional networks/parameters

DDQN on Warehouse Robot:

DDQN on Cartpole-V1:

On Cartpole-v1, DDQN significantly reduces the Overestimation bias present within regular DQN, resulting in a more steady trend. However, it has a slower growth rate comparatively and takes longer to train





DDQN on MountainCar-v0:

References:

https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf
https://www.gymlibrary.dev/environments/classic_control/cart_pole/
https://www.gymlibrary.dev/environments/classic_control/mountain_car/
https://www.youtube.com/watch?v=gOV8-bC1_KU
https://www.youtube.com/watch?v=fevMOp5TDQs