ELEC-E8125 - Reinforcement learning
Assignment 1

Rui Qu
rui.qu@aalto.fi

September 29, 2019

Exercise Q1

The same model trained with 200 timesteps can balance the pole for 500 timesteps. To do the test, I use the first reward function in Question 3, it converges at 278 episode. When testing with 200 timesteps,

```
Average test reward: 757.7997934188896 episode length: 200.0
```

When testing with 500 timesteps,

```
Average test reward: 1529.501773236467 episode length: 500.0
```

The default value of maximum episode length was 200, after we change it to 500 the agent continues to learn to balance the pole and gets higher reward.

Exercise Q2

The difference comes from the fact that the initial position of the pole and the cart, they're different from each run. Moreover, it also depends on the initialization of the initial weight vector. If the weight vector and the initial position of the cart are set to a constant, the variance will decrease.

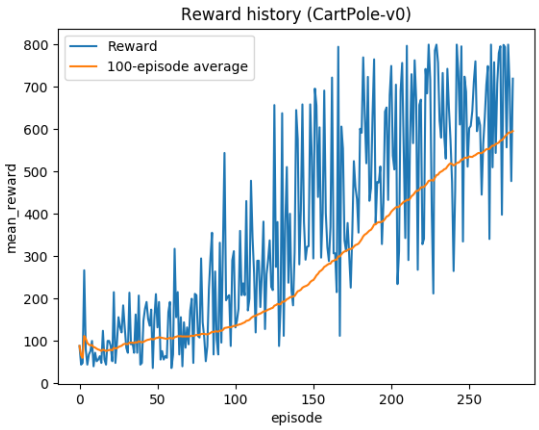
Exercise Q3

1. To give the agent extra incentive to balance the pole close to the center of the screen (close to x = 0).

```
1  def new_reward(state):
2      return min(4, np.abs(1/state[0]))
```

Parameter setting: $train\_episodes = 500, env.\_max\_episode\_steps = 200$

```
Episode 278 finished. Total reward: 720 (200 timesteps)
Looks like its learned. Finishing up early
```



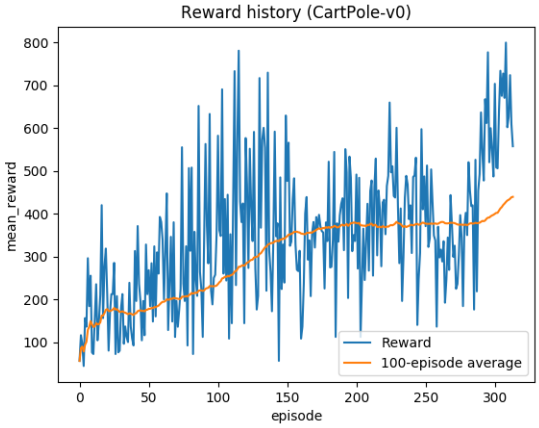
This reward function converges quickly. Since it starts from 0, the reward is already very high at the beginning and tends to stay in that position. We use the absolute value of the reverse position, which means that when the value is zero, the reward is highest and decreases as the position moves away from zero. However, we can only limit it to 4 at most, otherwise the reward will be too high to be stable.

2. To incentivize the agent to balance the pole in an arbitrary point of the screen (x Æ x0), with x0 passed as a command line argument to the script.

```
def new_reward(state):
    if state[0] == args.x_0:
         return 4
    else:
        return min(4, 1/np.abs(state[0]-args.x_0))
```

Parameter setting: $train\_episodes = 500, env.\_max\_episode\_steps = 200$

```
Episode 313 finished. Total reward: 558 (200 timesteps)
Looks like its learned. Finishing up early
```



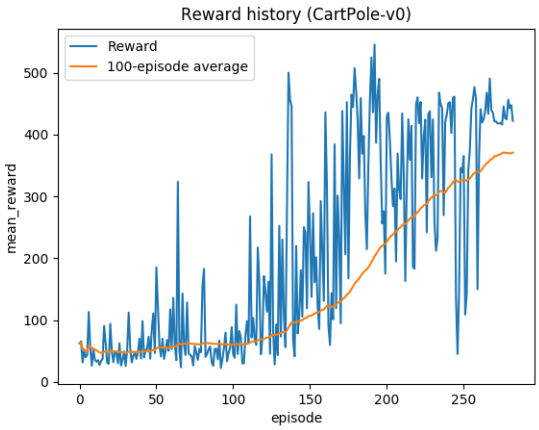
The reward function follows the similar logic as the first one, but the convergence time is longer. The model has also been tried several times. Any arbitrary taken is 1.5 and the test render shows that the point has arrived there. We also observed that the further away arbitrary point is from the center, the faster the agent will try to get there.

3. make the agent try to keep moving the cart as fast as possible, while still balancing the pole.

```
def new_reward(state):
    return 2+state[1]**2
```

Parameter setting: $train\_episodes = 500, env.\_max\_episode\_steps = 200$

```
Episode 282 finished. Total reward: 422 (200 timesteps)
Looks like its learned. Finishing up early
```



For this model, if the agent stays at a higher speed, we must give it more rewards. The above function penalizes the reward when the speed is near zero, and the reward is high when the speed is far from zero. However, we observed that the faster the agent moves, the more difficult it is to balance the pole, so the high speed is not very stable.