Assignment 3
CP8319: Reinforcement Learning
Student Name: Albina Cako

Question 1.

Part a.

The optimal reward is 4.1.

As given, one episode is 5 actions. The maximum reward that can be achieved is 2, which is when we go from state 2 to state 1 (2→1). In order to achieve the highest score, we need to go to this maximum reward as many times as possible. Since we only can do 5 actions in a single trajectory, we can only do this twice.
In order to prove that 4.1 is the highest score we need to follow the following path:

0 → 2 → 1 → 2 → 1 → 0 => 0 + 2 + 0 + 2 + 0.1 = 4.1

As you can see, we start from state 0 (as we always have to start from state 0) and then we go straight to state 2. Then, we go to state 1 so we can achieve the max reward of 2, then we go back to state 2 in order to repeat the maximum reward one more time, before we go back to state 0. At the end we go to state 0 and get a reward of 0.1.

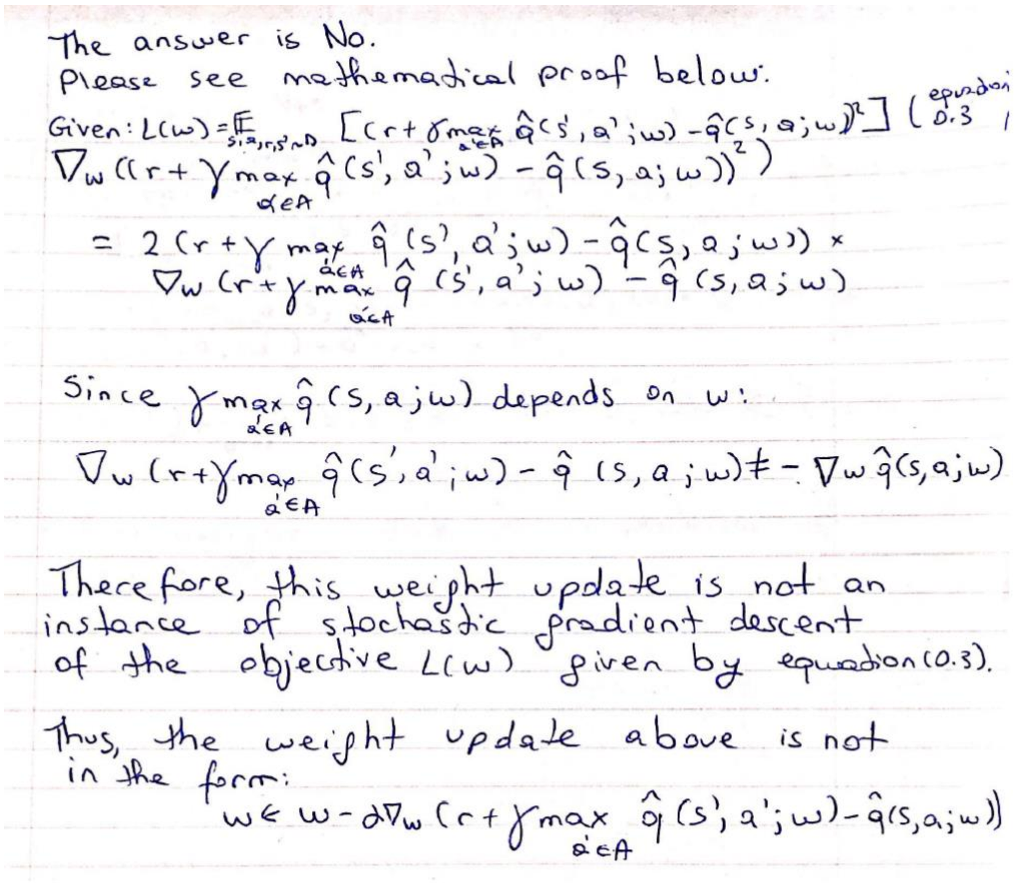
Question 2.

Part a.
A benefit of representing the Q function in this way, is in making a Q-Learning update. In this case, the Deep Q-Learning Network is taking a state as an input and is outputting the action values of the state for all the actions in that state at the same time. In this way, for each Q-Learning update we are only doing a single forward-pass. On the other hand, if we were to use both the action and the state as the input, then we would have to do O(|A|) forward-passes for each action. Therefore, it is more computationally efficient to have the Q function this way, as you are computing a vector from the state (vectorized version), rather than computing the scalar values separately.

Part b.

Yes, the test passes. I did not conduct any extra tests.

Part c.



The answer is No.
Please see mathematical proof below.

Given: $L(w) = \mathbb{E}_{s,a,r,s'\sim D}[(r+\gamma \max_{a'\in A} \hat{q}(s',a';w) - \hat{q}(s,a;w))^2]$ (equation 0.3)

$\nabla_w((r+\gamma \max_{a'\in A} \hat{q}(s',a';w) - \hat{q}(s,a;w))^2)$

$= 2(r+\gamma \max_{a'\in A} \hat{q}(s',a';w) - \hat{q}(s,a;w)) \times \nabla_w(r+\gamma \max_{a'\in A} \hat{q}(s',a';w) - \hat{q}(s,a;w)$

Since $\gamma \max_{a'\in A} \hat{q}(s,a;w)$ depends on w:

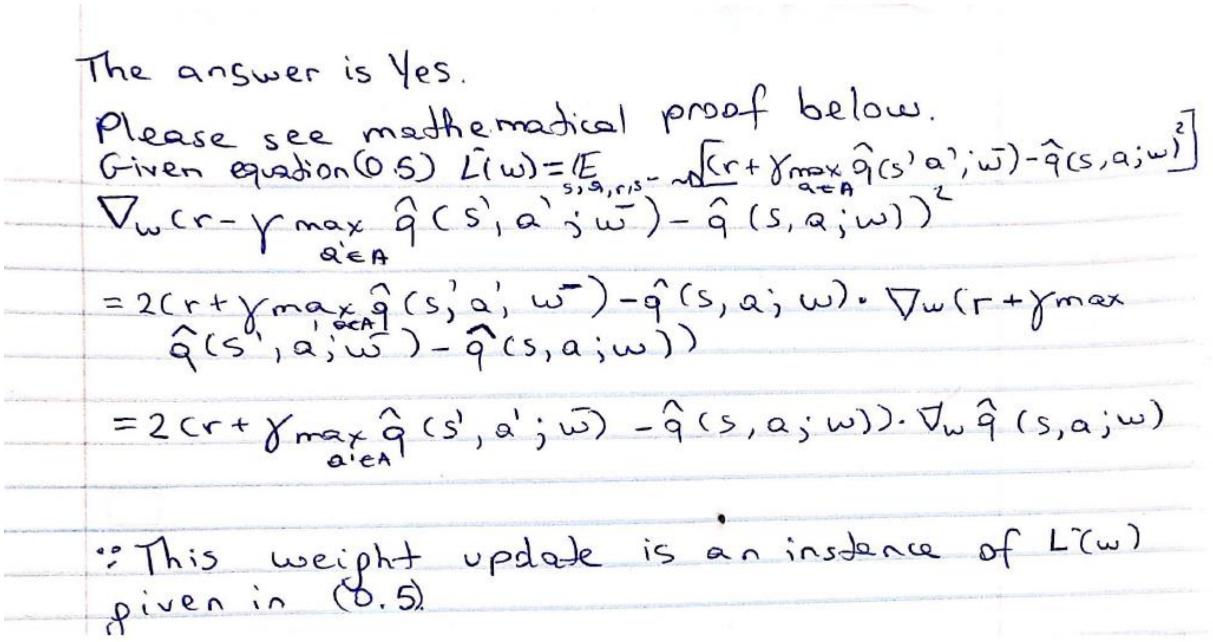
$\nabla_w(r+\gamma \max_{a'\in A} \hat{q}(s',a';w) - \hat{q}(s,a;w) \neq -\nabla_w \hat{q}(s,a;w)$

Therefore, this weight update is not an instance of stochastic gradient descent of the objective $L(w)$ given by equation (0.3).

Thus, the weight update above is not in the form:

$w \leftarrow w - \alpha\nabla_w(r+\gamma \max_{a'\in A} \hat{q}(s',a';w) - \hat{q}(s,a;w))$

Part d.



The answer is Yes.
Please see mathematical proof below.

Given equation (0.5) $L^{-}(w) = \mathbb{E}_{s,a,r,s'\sim D}[(r+\gamma \max_{a'\in A} \hat{q}(s',a';w^-) - \hat{q}(s,a;w)^2]$

$\nabla_w(r-\gamma \max_{a'\in A} \hat{q}(s',a';w^-) - \hat{q}(s,a;w))^2$

$= 2(r+\gamma \max_{a'\in A} \hat{q}(s',a',w^-) - \hat{q}(s,a;w) \cdot \nabla_w(r+\gamma \max \hat{q}(s',a;w^-) - \hat{q}(s,a;w))$

$= 2(r+\gamma \max_{a'\in A} \hat{q}(s',a';w^-) - \hat{q}(s,a;w)) \cdot \nabla_w \hat{q}(s,a;w)$

∴ This weight update is an instance of $L^{-}(w)$ given in (0.5)

Question 3.

Part a.

Yes, I did reach the optimal reward of 4.1 in the test environment. Please see the graph attached below:

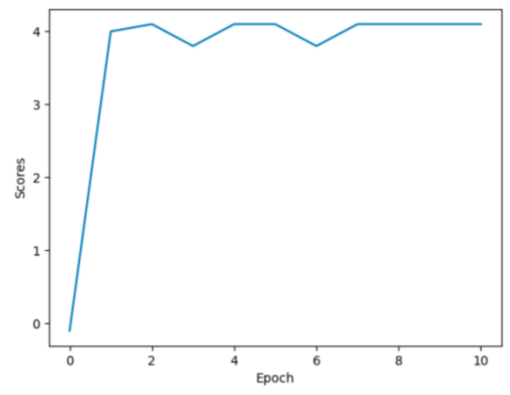


Figure 1. Plot of performance of the linear approximation in the test environment using Pytorch

Part b.

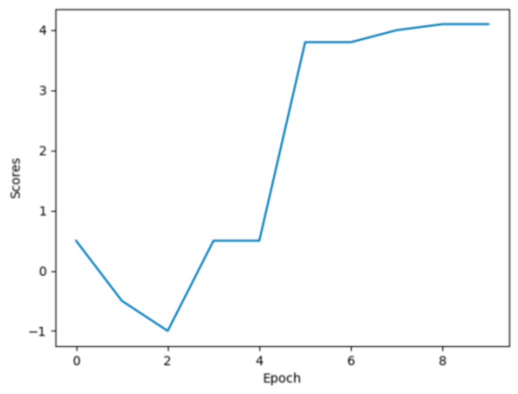


Figure 2. Plot of performance of the nature DQN in the test environment using Pytorch

Both the model in the linear approximation and nature Deep Q-Network (DQN) performed the same. They both reached the optimal policy of 4.1. However, the training time in the nature was longer as it took about 30 seconds, while in the linear approximation it was only about 10 s. Thus, the DQN performed approximately 3 times slower than the linear approximation. This could be due to the fact that the DQN had to go through the convolutional layers, which could take a longer time to compute.