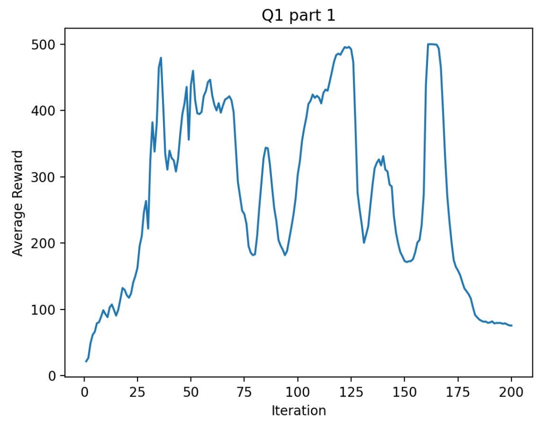
Q1.

The neural network model used for the policy generation and trained and forwarded to get an action.

It has 4 inputs of state and 2 output of action probabilities. It contains 256 hidden layers. PRELU is used with dropouts just as the previous assignment.

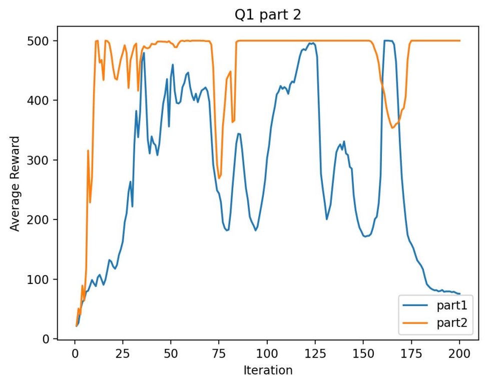
The loss function has been constructed differently for the following three parts.

Part 1.



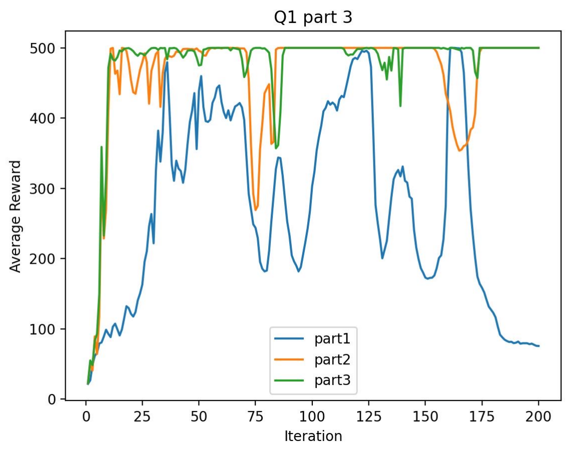
The average reward of the output of q1 shows not a stable behavior. It did learn by every iteration but it does not seem to converge just with 200 iterations and 500 episodes. However, it still shows that it learned from the iterated training.

Part 2.



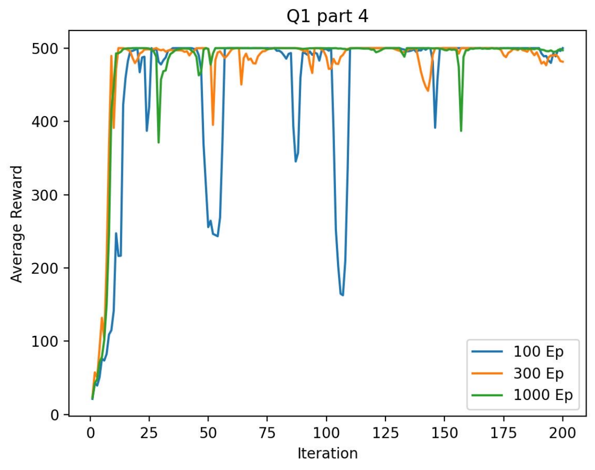
By only considering the forward return, the average reward converges much earlier and the output it much more stabilized than that of the part 1. It is because of the updated forward returns.

Part 3.



By incorporating the baseline with the forward return from part 2, the average reward is much more stabilized and shows lower variance than the part 2.

Part 4.



As episode increases, the average rewards converges earlier and become stabilized earlier. In addition, overall it had the lower variance over the entire iterations. It is because by having more episode per iteration, the policy gradients are optimized over more dataset so that the policy model pi can be optimized better. Moreover, the baseline itself could be improved by having a larger dataset to be averaged per iteration.

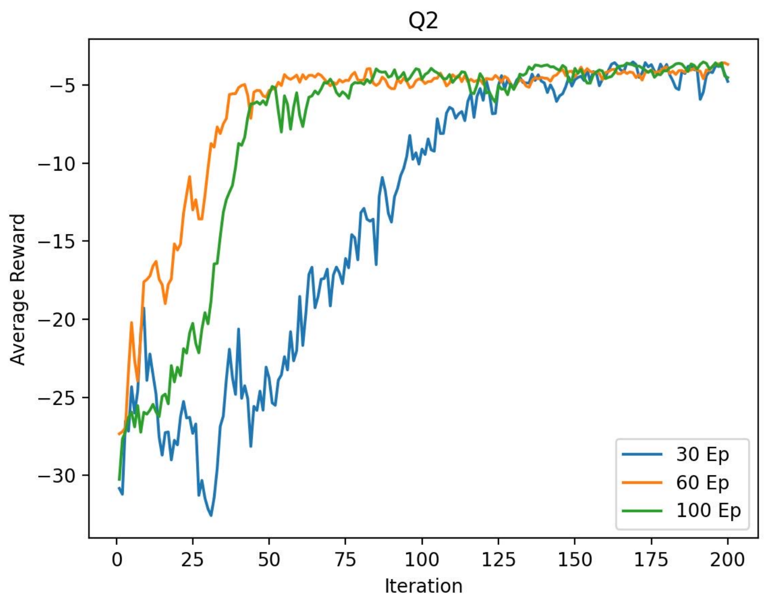
Q2.

The neural network model used for the policy generation and trained and forwarded to get an action.

It has 8 inputs of state and 2 output of action output. It contains 32 hidden layers. RELU is used as the activation function.

The output for action has been sampled from a multivariate normal distribution. It is to handle the dynamics of the link arm and its covariant matrix.

The loss function has been constructed from the policy gradient of the Vanilla Reinforce Algorithm forward return with baseline.



The trained output with three different number of episodes converges and shows to succeed the given task.

Thus, it can be said the 30 episode is the smallest possible number of episodes that succeeds.

However, it converges too late and it is still highly unstable in the later iteration so that the 60 episode can be the smallest possible number of episodes that stably succeeds, as it shows a stable average reward in the later iteration.

The gif file is included in the q2 folder with the rendered performance of one episode with the trained model with 30 episodes per iteration.