Continuous Control

Krithika Govindaraj

1. **Introduction**

   In the previous lessons, we learned about Q-tables and dynamic programming for reinforcement learning. These methods however, had various drawbacks such as their applicability to continuous spaces where information cannot be stored in huge tables. Hence, neural networks or deep learning was introduced as a means of capturing this information in the form of weights. In order to derive policy directly from the learning we used policy based methods and finally combined the benefits of both policy based and value based methods using techniques like A2C, A3C or DDPG.

   The goal of this project is to train a double jointed arm to move along with the target using some of the techniques we learned from the policy-based methods and I will be implementing DDPG as it not only combines actor-critics, but is similar to DQN that we studied earlier. This gives us more options to experiment with the network and understand the working. Also, based on my own research most robotic arms work well with DDPG-HER, hence I decided to use this method.

2. **Implementation**

   DDPG consists of :

   1. Actor network: This is the network that implements the policy-based methods to derive the appropriate action.
   2. Critic network: This is the network that used the action form the actor network, along with the state of the environment to produce a value that acts as a baseline to reduce variance of policy-based actor network.
   3. Target Actor: Similar to DQN, in order to avoid the moving target issue, a target actor network is generated.
   4. Target Critic: Similar to DQN, in order to avoid the moving target issue, a critic network is generated.
   5. Replay buffer: Memory to store all experiences derived from the environment and randomly access it to reduce correlation during training.

3. **Experiment**

   3.1 Trial 1

```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 256        # minibatch size
GAMMA = 0.99            # discount factor
TAU = 1e-3              # for soft update of target parameters
LR_ACTOR = 1.5e-4       # learning rate of the actor
LR_CRITIC = 1.5e-3      # learning rate of the critic
WEIGHT_DECAY = 0.01     # L2 weight decay
fc1_units=400, fc2_units=300
```

Episode 10 Average Score: 0.12
Episode 20 Average Score: 0.14
Episode 30 Average Score: 0.10
Episode 40 Average Score: 0.08
Episode 50 Average Score: 0.07
Episode 60 Average Score: 0.06
Episode 70 Average Score: 0.06
Episode 80 Average Score: 0.06
Episode 90 Average Score: 0.05
Episode 100 Average Score: 0.05
Episode 110 Average Score: 0.05
Episode 120 Average Score: 0.03
Episode 130 Average Score: 0.03
Episode 140 Average Score: 0.03
Episode 150 Average Score: 0.02
Episode 160 Average Score: 0.02
Episode 170 Average Score: 0.02
Episode 180 Average Score: 0.02
Episode 190 Average Score: 0.02
Episode 200 Average Score: 0.02
Episode 210 Average Score: 0.02
Episode 220 Average Score: 0.02
Episode 230 Average Score: 0.02

No learning, need more exploration or model is not complex enough to pick nuances.

One possible reason for this difficulty is the distribution of the inputs to layers deep in the network may change after each mini-batch when the weights are updated. This can cause the learning algorithm to forever chase a moving target. This change in the distribution of inputs to layers in the network is referred to by the technical name “internal covariate shift.”

Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks.

Batch normalization provides an elegant way of reparametrization for almost any deep network. The reparametrization significantly reduces the problem of coordinating updates across many layers.

Changing the Adam Optimizer hyperparamters is not a good idea. I changed the weight decay to 0.01. it's better to leave it at its default, learning_rate = 0.001, beta1 = 0.9 beta2 =0.999 epsilon=1e-08.
The weight decay is for L2 regularization(i.e generalization) The value of 0.01 was too strong due to which the model was not updating properly.

3.2 Trial 2

BUFFER_SIZE = int(1e5) # replay buffer size
BATCH_SIZE = 256   # minibatch size
GAMMA = 0.99   # discount factor
TAU = 1e-3   # for soft update of target parameters
LR_ACTOR = 1e-4   # learning rate of the actor
LR_CRITIC = 1e-3   # learning rate of the critic
WEIGHT_DECAY = 0   # L2 weight decay

Added batch Normalization according to the paper:
Episode 10 Average Score: 0.12
Episode 20 Average Score: 0.14
Episode 30 Average Score: 0.81
Episode 40 Average Score: 1.08
Episode 50 Average Score: 1.07
Episode 60 Average Score: 1.06
Episode 70 Average Score: 1.06
Episode 80 Average Score: 1.06
Episode 90 Average Score: 1.15
Episode 100 Average Score: 1.17
Episode 110 Average Score: 1.15
Episode 120 Average Score: 1.13
Episode 130 Average Score: 1.13
Episode 140 Average Score: 1.13
Episode 150 Average Score: 1.12
Episode 160 Average Score: 1.12
Episode 170 Average Score: 1.15
Episode 180 Average Score: 1.17
Episode 190 Average Score: 1.18
Episode 200 Average Score: 1.20
Episode 210 Average Score: 1.22
Episode 220 Average Score: 1.25
Episode 230 Average Score: 1.25

Learning is very slow. Maybe needs more randomness to explore. Hence adding epsilon decay.

3.3 Trial 3:

BUFFER_SIZE = int(1e5) # replay buffer size
BATCH_SIZE = 128   # minibatch size
GAMMA = 0.99   # discount factor
TAU = 1e-3   # for soft update of target parameters
LR_ACTOR = 1e-4   # learning rate of the actor
LR_CRITIC = 1e-3   # learning rate of the critic
WEIGHT_DECAY = 0   # L2 weight decay
EPSILON = 1   #Epsilon Noise Parameter
EPSILON_DECAY = 1e-6   #Epsilon Decay Parameter

Episode 10 Average Score: 0.63
Episode 20 Average Score: 0.84
Episode 30 Average Score: 0.90
Episode 40 Average Score: 1.08
Episode 50 Average Score: 1.25
Episode 60 Average Score: 1.23
Episode 70 Average Score: 1.28
Episode 80 Average Score: 1.35
Episode 90 Average Score: 1.36
Episode 100 Average Score: 1.38
Episode 110 Average Score: 1.50
Episode 120 Average Score: 1.53
Episode 130 Average Score: 1.57
Episode 140 Average Score: 1.53
Episode 150 Average Score: 1.50
Episode 160 Average Score: 1.50
Episode 170 Average Score: 1.46
Episode 180 Average Score: 1.37
Episode 190 Average Score: 1.33
Episode 200 Average Score: 1.30
Episode 210 Average Score: 1.24

Learning improves initially but then decreases, due to noise induced uncontrollably. so we need to decide when to induce the noise.

3.4 Trial 4:

BUFFER_SIZE = int(1e5) # replay buffer size
BATCH_SIZE = 128   # minibatch size for memory
GAMMA = 0.99   # discount factor
TAU = 1e-3   # for soft update of target parameters
LR_ACTOR = 1e-4   # learning rate of the actor
LR_CRITIC = 1e-3   # learning rate of the critic

WEIGHT_DECAY = 0   # L2 weight decay
EPSILON = 1   #Epsilon Noise Parameter
EPSILON_DECAY = 1e-6   #Epsilon Decay Parameter
LEARNING_PERIOD = 20   # learning frequency
UPDATE_FACTOR   = 10   # how much to learn

Episode 10	Average Score: 0.44
Episode 20	Average Score: 0.43
Episode 30	Average Score: 0.70
Episode 40	Average Score: 0.77
Episode 50	Average Score: 0.94
Episode 60	Average Score: 1.04
Episode 70	Average Score: 1.11
Episode 80	Average Score: 1.22
Episode 90	Average Score: 1.31
Episode 100	Average Score: 1.38
Episode 110	Average Score: 1.53
Episode 120	Average Score: 1.67
Episode 130	Average Score: 1.74
Episode 140	Average Score: 1.83
Episode 150	Average Score: 1.93
Episode 160	Average Score: 2.05
Episode 170	Average Score: 2.21
Episode 180	Average Score: 2.24
Episode 190	Average Score: 2.28
Episode 200	Average Score: 2.40
Episode 210	Average Score: 2.48

Reward is better than before i.e quantity wise as well as constantly improving, but very slow learning. So increasing learning rate of actor and critic

3.5 Trial 5:

BUFFER_SIZE = int(1e5) # replay buffer size
BATCH_SIZE = 128   # minibatch size for memory
GAMMA = 0.99   # discount factor
TAU = 1e-3   # for soft update of target parameters
LR_ACTOR = 1.5e-4   # learning rate of the actor
LR_CRITIC = 1.5e-3   # learning rate of the critic
WEIGHT_DECAY = 0   # L2 weight decay
EPSILON = 1   #Epsilon Noise Parameter
EPSILON_DECAY = 1e-6   #Epsilon Decay Parameter
LEARNING_PERIOD = 20   # learning frequency
UPDATE_FACTOR   = 10   # how much to learn

Episode 10 Average Score: 0.44
Episode 20 Average Score: 0.43
Episode 30 Average Score: 0.70
Episode 40 Average Score: 0.77
Episode 50 Average Score: 0.94
Episode 60 Average Score: 1.04
Episode 70 Average Score: 1.11
Episode 80 Average Score: 1.22
Episode 90 Average Score: 1.31
Episode 100 Average Score: 1.38
Episode 110 Average Score: 1.53
Episode 120 Average Score: 1.67

Still slow learning, maybe increase the experience replay buffer size to get more data

3.6 Trial 6:

```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 256        # minibatch size for memory
GAMMA = 0.99            # discount factor
TAU = 1e-3              # for soft update of target parameters
LR_ACTOR = 1.5e-4         # learning rate of the actor
LR_CRITIC = 1.5e-3        # learning rate of the critic
WEIGHT_DECAY = 0        # L2 weight decay
EPSILON = 1               #Epsilon Noise Parameter
EPSILON_DECAY = 1e-6     #Epsilon Decay Parameter
LEARNING_PERIOD = 20    # learning frequency
UPDATE_FACTOR   = 10    # how much to learn
```

Episode 10 Average Score: 0.73
Episode 20 Average Score: 0.74
Episode 30 Average Score: 0.71
Episode 40 Average Score: 0.97
Episode 50 Average Score: 1.04
Episode 60 Average Score: 1.03
Episode 70 Average Score: 1.07
Episode 80 Average Score: 1.10
Episode 90 Average Score: 1.15
Episode 100 Average Score: 1.11

Good start, but slow after a point. Used nn.LayerNorm instead of nn.BatchNorm1d.

3.7 Trial 7:

BUFFER_SIZE = int(1e5) # replay buffer size
BATCH_SIZE = 256   # minibatch size for memory
GAMMA = 0.99   # discount factor
TAU = 1e-3   # for soft update of target parameters
LR_ACTOR = 1.5e-4   # learning rate of the actor
LR_CRITIC = 1.5e-3   # learning rate of the critic
WEIGHT_DECAY = 0   # L2 weight decay
EPSILON = 1   #Epsilon Noise Parameter
EPSILON_DECAY = 1e-6   #Epsilon Decay Parameter
LEARNING_PERIOD = 20   # learning frequency
UPDATE_FACTOR   = 10   # how much to learn

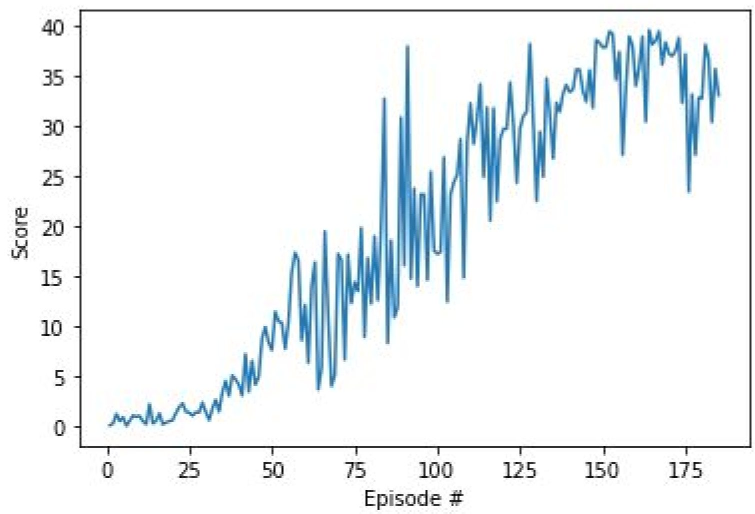
Episode 10	Average Score: 0.52
Episode 20	Average Score: 0.39
Episode 30	Average Score: 0.60
Episode 40	Average Score: 0.80
Episode 50	Average Score: 1.03
Episode 60	Average Score: 1.45
Episode 70	Average Score: 1.65
Episode 80	Average Score: 1.79
Episode 90	Average Score: 1.98
Episode 100	Average Score: 2.14
Episode 110	Average Score: 2.48
Episode 120	Average Score: 2.83
Episode 130	Average Score: 3.40
Episode 140	Average Score: 3.77
Episode 150	Average Score: 4.23
Episode 160	Average Score: 4.48
Episode 170	Average Score: 4.88
Episode 180	Average Score: 5.15
Episode 190	Average Score: 5.36
Episode 200	Average Score: 5.63
Episode 210	Average Score: 5.90
Episode 220	Average Score: 6.00
Episode 230	Average Score: 5.86
Episode 240	Average Score: 5.75

Better results, but learning is still slow. Making the learning rate of both actor and critic the same could help synchronize the experiences and learning of both networks.

3.8 Trail 8:

```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 256        # minibatch size for memory
GAMMA = 0.99            # discount factor
TAU = 1e-3              # for soft update of target parameters
LR_ACTOR = 1.5e-4         # learning rate of the actor
LR_CRITIC = 1.5e-4        # learning rate of the critic
WEIGHT_DECAY = 0        # L2 weight decay
EPSILON = 1             #Epsilon Noise Parameter
EPSILON_DECAY = 1e-6     #Epsilon Decay Parameter
LEARNING_PERIOD = 20    # learning frequency
UPDATE_FACTOR   = 10    # how much to learn
```

Episode 10 Average Score: 0.62
Episode 20 Average Score: 0.62
Episode 30 Average Score: 0.94
Episode 40 Average Score: 1.48
Episode 50 Average Score: 2.45
Episode 60 Average Score: 4.05
Episode 70 Average Score: 4.94
Episode 80 Average Score: 6.05
Episode 90 Average Score: 7.38
Episode 100 Average Score: 8.76
Episode 110 Average Score: 11.03
Episode 120 Average Score: 13.80
Episode 130 Average Score: 16.66
Episode 140 Average Score: 19.47
Episode 150 Average Score: 22.36
Episode 160 Average Score: 24.76
Episode 170 Average Score: 27.46
Episode 180 Average Score: 29.40
Environment Solved in Episode 185 Average Score: 30.21



4. **Future Work**

Using TD estimation in the critic network and using roll-out of length 5 has shown to produce good results based on some papers I have read by helping to improve the learning. Using a parallel learning technique such as training the 20 double-jointed arms could improve learning time.