Nanodegree: Deep Reinforcement Learning

Project: Continuous Control (By: Kranthi Kumar Talluri)

Section 1: Introduction

The project showcases the effectiveness of Actor-Critic methods, specifically Deep Deterministic Policy Gradient (DDPG), in learning appropriate policies for Reinforcement Learning tasks. The task involves a Reacher environment with continuous state space, where the agent’s goal, represented as a double-joined arm, is to maintain its position at a target location, which is an orbiting sphere. The agents aim to optimize cumulative rewards by associating states with actions through a policy. Optimal policy is achieved by using Deep Neural Network (DNN) as non-linear function approximator. Throughout the project, various files such as Jupyter Notebooks and code files are used to set up dependencies, define agent characteristics and store trained models.

A simple game is implemented here based on DNN to demonstrate its model free learning. The position of agent’s arm in Reacher environment must be in the optimal location to fetch a reward of +0.1. The observation space is 33 dimensions.

- The goal is to train an agent to achieve an average score of +30 over 100 consecutive episodes by maintaining its position for as many time steps as possible.
- The state space consists of 33 features, including the agent's velocity, position, rotation and angular velocities.
- The action space has a dimension of four, including the torque applicable to 2 joints. The entries are restricted to either -1 or 1.
- DDPQ, an off-policy learning algorithm, is used as the core learning algorithm. It learns the action-value function (q) through Actor-Critic network and uses a DNN as a function approximator.
- Code cell is provided with a random action agent, to learn how agent is controlled by Python API to receive feedback from environment.

Section 2: Implementation

Subsection 1: Algorithm

Deep Deterministic Policy Gradient (DDPG) is a RL technique which is a combination of both Policy gradients and Q-learning. This algorithm can specifically be used where the actions are continuous. Some of the main features are mentioned below:

- DDPG algorithm has two models namely the actor and the critic, hence this algorithm is also known as an actor-critic technique. The actor gives suggestions to perform action on the bases of current state whereas Critic helps to validate about how good is the taken action.

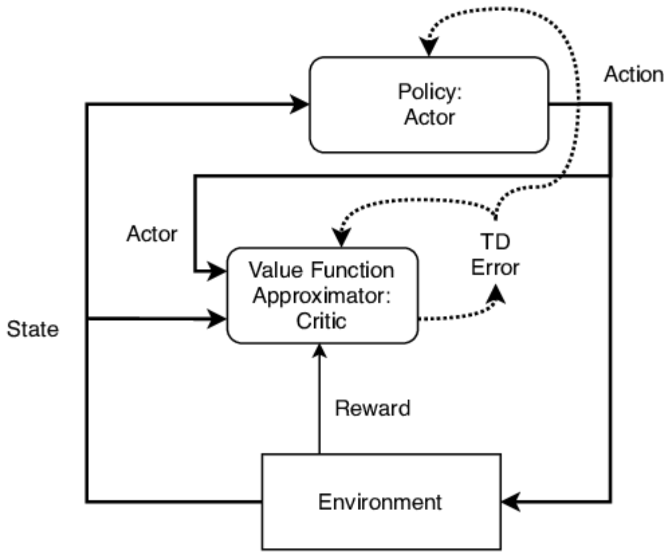


Fig: Actor-Critic workflow architecture for DDPG algorithm[source]

- The Actor network is a policy network which takes information of current game state as input and gives exact action to be taken as output rather than providing the probability distribution of the action.
- Critic network is a Q-value network that takes current state and action as inputs and helps to evaluate how good is that action based on Q-value score.
- DDPG is an off-policy method, which implies Actor and Critic are trained using experience of different policy instead of current policy. Because of which past experiences are efficiently learned by this approach.
- Overall, DDPG is an improvement over the basic actor-critic method, and it can efficiently handle continuous actions.

Subsection 2: Architecture of Neural Network

The actor network consists of 3 hidden layers, with 600, 400, and 200 nodes respectively. It takes game states as input and gives best action as output. The hidden layer is followed by an activation function named Rectified Linear Unit (ReLU) and for handling the continuous action space Tanh activation function is used at output layer.

Critic network consists of 2 hidden layers with 400 and 200 nodes respectively. Here also hidden layer uses ReLU as an activation function. The state and action from actor network are passed as input and Q-value score to evaluate how good is the action is obtained at output.

Subsection 3: Hyperparameters

Hyperparameters tuning plays a vital role in model performance. Among various important hyperparameters, the Learning rate (LR) plays a crucial role in achieving better performance for both Actor and Critic network. In can be seen from below figure that, Initially the performance of agent with DDPG algorithm is poor. Changing the learning rate parameter helped the agent to learn and show some better progress.

Initially LR 1e-4 was used for both the networks. The agent´s performance was bad as the learning was slow. Then, with LR 1.5e-4, the task was accomplished with an average score of **30.18** in **223 episodes**. Finally, with LR 2e-4 I was able to achieve decent exponential progress in learning with an average score of **30.02** in **178 episodes** as shown in the below table:

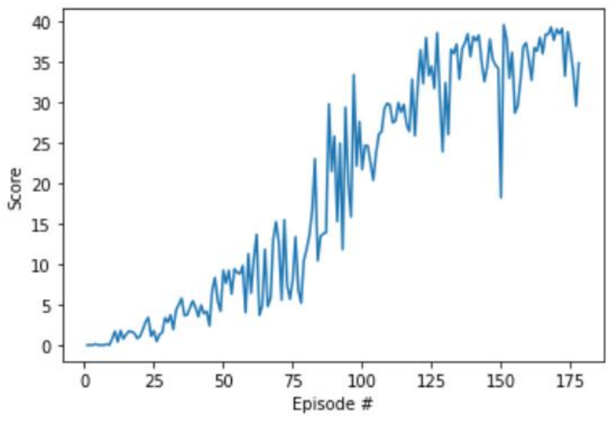
Number of Episodes	Average score
25	0.98
50	2.59
75	4.67
100	8.02
125	14.98
150	22.44
175	29.30
178	**30.02**

I could have tried with other different LR, but I was not able to access the GPU on the workspace. I was encountering issues loading the ML-Unity environment. So, with the help of local CPU I was able to achieve a good result in solving the task. Other hyperparameters are given in the below table.

Hyperparameters used	Assigned values
Learning rate Actor	2e-4
Learning rate Critic	2e-4
Relay buffer size	1e5
Batch size	128
Gamma (discount factor)	0.99
TAU	1e-3
Number of episodes	1000
Weight decay	1e-4

Section 3: Results

Below graph is plotted between the number of episodes on X-axis and score on Y-axis. After tweaking some parameters, the below results are obtained. The maximum number of episodes for solving is **178** episodes.



Conclusion and Ideas for Future Improvement

Continuous space problems can be solved efficiently using a policy-based method as it approximates the real work. The current problem solved here consists of one agent with double joined arm and the Reacher environment. As a future work it would be interesting to explore the performance of DDPQ algorithm in more critical environments or with increased arms. Deeper study on reward function designing could be done and creation of unity ML-agents simulation can be explored.