Continuous Control Project Report

Udacity Nano Degree Program

Student:

Nikhil Masinete

Course:

Deep Reinforcement Learning

Introduction

Reinforcement Learning is a branch of machine learning where a software agent is taught to how to take actions in an environment in order to maximize its cumulative reward. This theory is fundamentally based on Markov Decision Process. This concept resembles the way living beings live. The following are the terms which are frequently used in this report.

Agent: An agent is the smart being which is trained to achieve tasks given to it.

Environment: An environment is a situation where the agent lives.

State: A state is a quantified description of a situation.

Action: An action is the way how the agent should respond in each state.

Reward: A reward is the outcome given to the agent for taking a particular action in each state. A reward is positive or negative based on the action taken was desirable or undesirable.

State-Action value function (Q function): A state action value function is a matrix that stores the expected reward when a particular action is taken in each state.

Policy: The policy is the strategy that the agent employs to determine the next action based on the current state.

Episode: All states that come in between an initial-state and a terminal state.

Challenge

The challenge is to place an end detector of a robotic arm in a moving target space. In this environment, a double-jointed arm can move to target locations. A reward of +0.1 is provided for each step that the agent's hand is in the goal location. Thus, the goal of the agent is to maintain its position at the target location for as many time steps as possible.

Algorithm

To solve this challenge Deep Deterministic Policy Gradients algorithm was used. The reason for emergence of this algorithm is due to limitation of the Deep Q Learning model. DQL algorithm is best suited for environments where the environment has a low discrete action space. In this scenario the action space is a continuous one. So DDPG was required. The DDPG algorithm consists of an actor and a critic. An actor decides the which action to be taken for a given state. A critic is good at estimating the worth of action taken and able to predict the future rewards. Generally policy gradient methods is used as an actor and Monte-carlo estimates are used to determine the future rewards. The combination of these two models is DDPG algorithm. In solving the current environment the following hyper parameters were used.

Buffer_size = 1e6

Batch_size = 128

Gamma (discount factor) = 0.95

Tau (update factor) = 1e-3

Learning_rate_actor = 1e-4

Learning_rate_critic = 1e-3

Update_times = 1

Update_every = 5

Epsilon = 1.0

Epsilon_decay = 1e-6

Sizes of hidden layers = 400, 300

Result

The training on a single agent was solved in 461 episodes. But when it was trained on twenty agents it was solved in 31 episodes. That is due to the knowledge sharing between the agents.

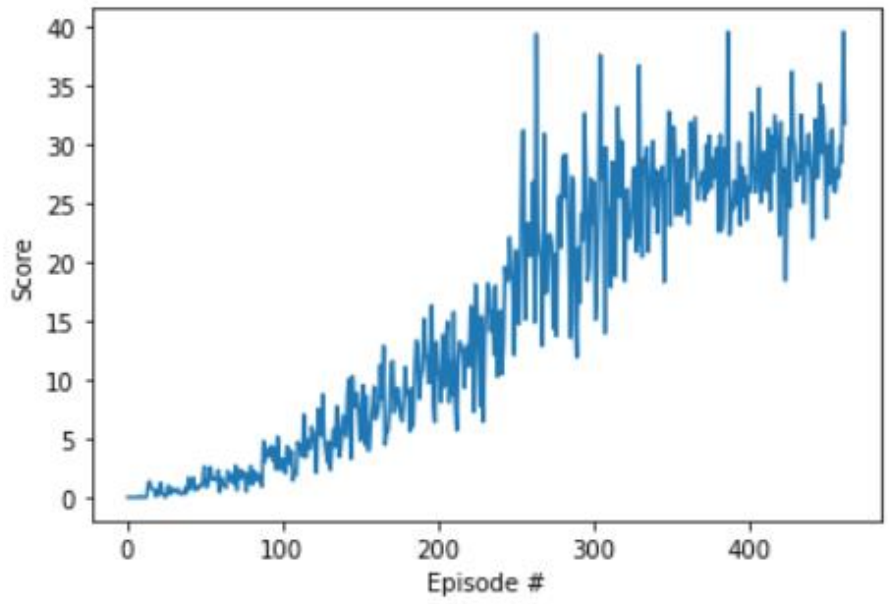

Figure 1: Result of training of a Single agent

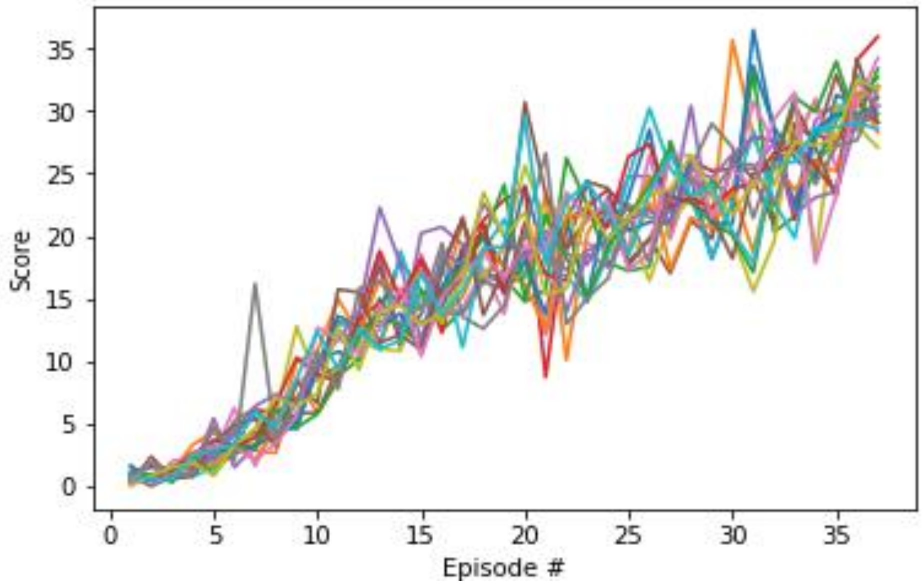

Figure 2: Result of training of Multi agent

Future Improvements

The performance can be improved by implementing Cross Entropy Method and REINFORCE.

References

1. Continuous control with Deep Reinforcement Learning, Google DeepMind.