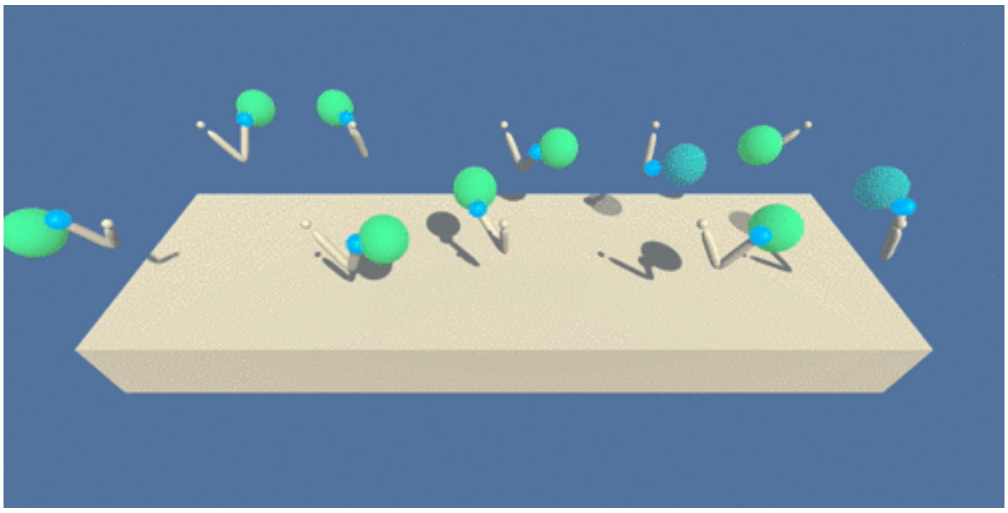
Environment Introduction

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 your agent is to maintain its position at the target location for as many time steps as possible.

The observation space consists of 33 variables corresponding to position, rotation, velocity, and angular velocities of the arm. Each action is a vector with four numbers, corresponding to torque applicable to two joints. Every entry in the action vector should be a number between -1 and 1.

The version of environment chosen for the project is the version contains 20 identical agents, each with its own copy of the environment.



Solving the Environment

The algorithm used here is a Deep Deterministic Policy Gradient (DDPG). A DDPG is composed of two networks: one actor and one critic. During a step, the actor is used to estimate the best action, the critic then use this value as in a DDQN to evaluate the optimal action value function. Both actor and the critic are composed of two networks. On local network and one target network.

The first trial was done on the agent and model provided in ddpg-pendulum project with little modification of hyperparameters. As explained in the guidelines, the code was modified to work for 20 agents. Moreover, gradient clipping was also applied when training the critic network. Though it solves the environment within 150 episodes (average score 30 in last 100 episodes) but still un-stability was seen during the training phase. One major breakthrough was while noticing the replay buffer size as originally 1e4 can only hold experiences from 5 episodes. This becomes problem if the agent performance goes down over episodes it was not possible to recover from the limited number of episodes experience stored in the replay buffer. With slight increasing the size of reply buffer to 5e4 it was seen that the agent trains really smoothly within 150-200 episodes.

- Architecture of Actor Network
    - input size = 33
    - output size = 4
    - 2 hidden layers and one output layer
    - each hidden layer has 128 hidden units and is followed by a ReLU activation layer
    - output layer is followed by a tanh activation layer
- Architecture of Critic Network
    - input size = 4
    - output size = 1
    - 2 hidden layers and one output layer
    - each hidden layer has 128 hidden units and is followed by a ReLU activation layer
    - output layer is followed by a linear activation unit
- Training hyperparameters
    - Buffer size : 500,000
    - Batch size : 128
    - γ : 0.99
    - τ : 0.001
    - learning rate actor : 0.001
    - learning rate critic : 0.001
    - weight decay : 0
    - learning performed per each environment step : 1

Plot of rewards

Ideas for future Work

- Prioritized experience replay: Prioritized experience replay selects experiences based on a priority value. This can improve learning by increasing the probability that rare and important experience vectors are sampled.
- Other algorithms: Solving the same environment with other algorithms like PPO, A3C or D4PG.