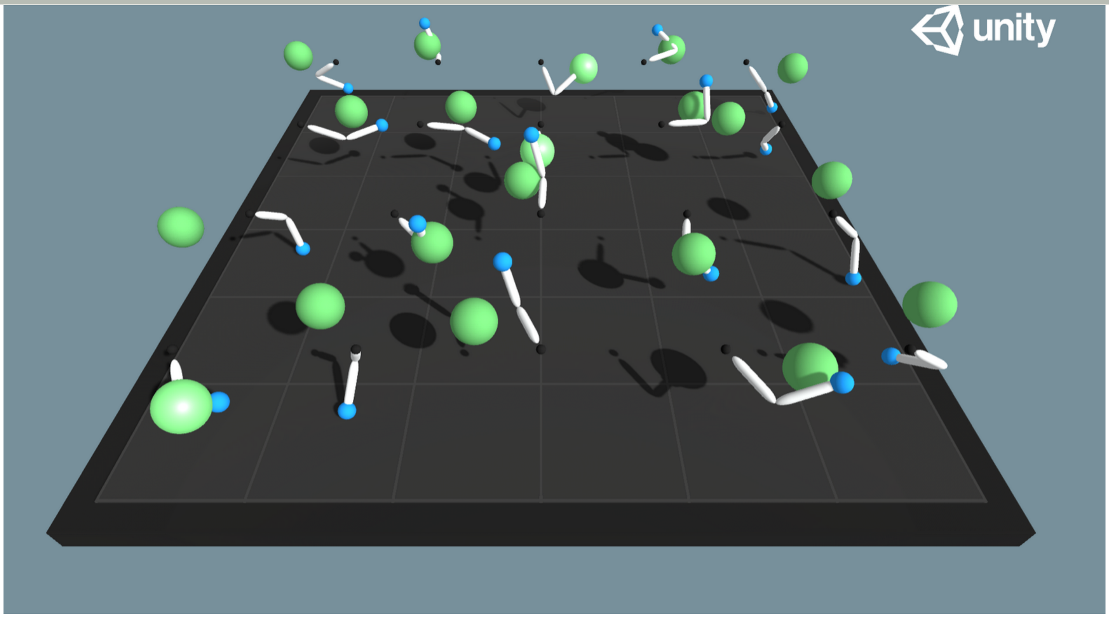
REPORT

PROJECT 2:CONTINUOUS CONTROL

AUTHOR NAME:VIGNESH.B.YAADAV



In this Project, a double-jointed arm has to 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. The observation space consists of 33 variables corresponding to the 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.

In this Project, well be using a DDPG(Deep Deterministic Policy Gradient action space) method for solving the project.

Intro into Continuous Control, we have come across many algorithms from DQN (Deep Q-Network) used as function approximation in a continuous space but DQN also has some constraints keeping up with the continuous space.

Actor-Critic method

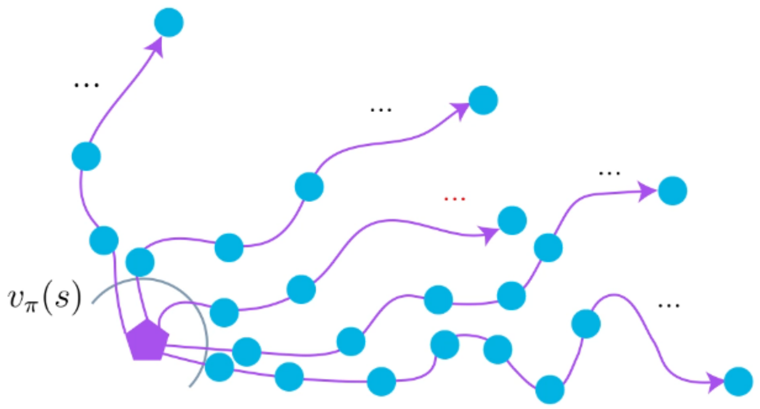
It is an intersection of Value-Based method and policy-based method.

What is the diff b/w value-based method vs policy-based method?
If an agent is set to use Deep Neural Network to estimate value function then its value-based.
If an agent uses a Deep Neural Network to approximate the optimal Policy then it's a Policy-based method. DQN is a value-based method.
In the value-based method, we try to approximate the optimal value for a given state "s" $V\pi(s) \rightarrow V^*(s)$
$Q\pi(s, a) \rightarrow Q^*(s, a)$.
Policy-based methods contain 2 types
-> Stochastic policy
-> Deterministic policy

In the value-based method, the actions are calculated using expectation returns.

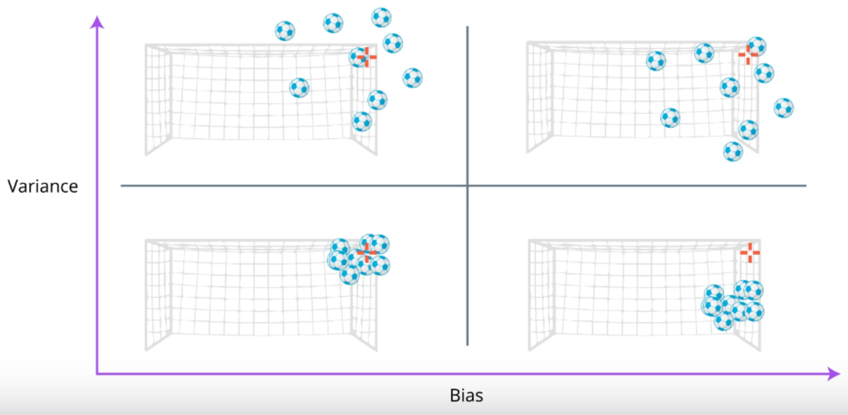
What's the best way to estimate value for Actor-Critic methods?

-> Montecarlo method:
In this method, all the rewards are added up with or without consideration of discounted return.
To calculate the value function we need to average the estimates. This method is unbiased but has a high variance.
-> TD Estimate:
In this method, we estimate the current state to the next state which is estimate over the estimate.
TD Estimate has a feature of bootstrapping were we leverage the estimate of the current state to calculate the next state. It has low variance but high bias.



$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$
$$q_\pi(s,a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$
$$a_\pi(s,a) = q_\pi(s,a) - v_\pi(s)$$

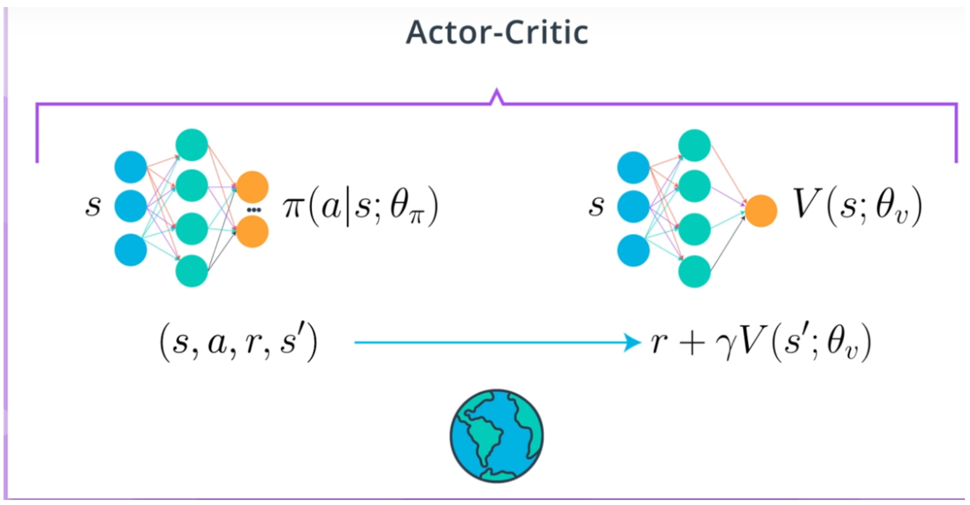
TD Estimate



How does a Basic Actor-Critic Agent work?

Actor-Critic Agent is an agent that uses function approximation to learn a policy and value Function to update the policy.
The actor-Critic method mainly consists of 2 neural networks 1. Actor-Network(Monte Carlo), 2. Critic network(TD Estimate)

A very basic Actor critic agent works as follows 1 takes in a state and outputs a probability distribution over actions, another network 2 takes in a state and outputs a state value function of policy π



Hears an intro on A3C (Asynchronous Advantage Actor-Critic):

As suggested by the name we are calculating Aπ and the critic will be learning to estimate Vπ.
A3C can use a single CNN with actor and critic sharing weights. Sharing whats is a more complex approach but using 2 networks speeds up the process.
As suggested by the name we are calculating Aπ and the critic will be learning to estimate Vπ.
A3C can use a single CNN with actor and critic sharing weights. Sharing whats is a more complex approach but using 2 networks speeds up the process.

What is On Policy VS Off Policy method?

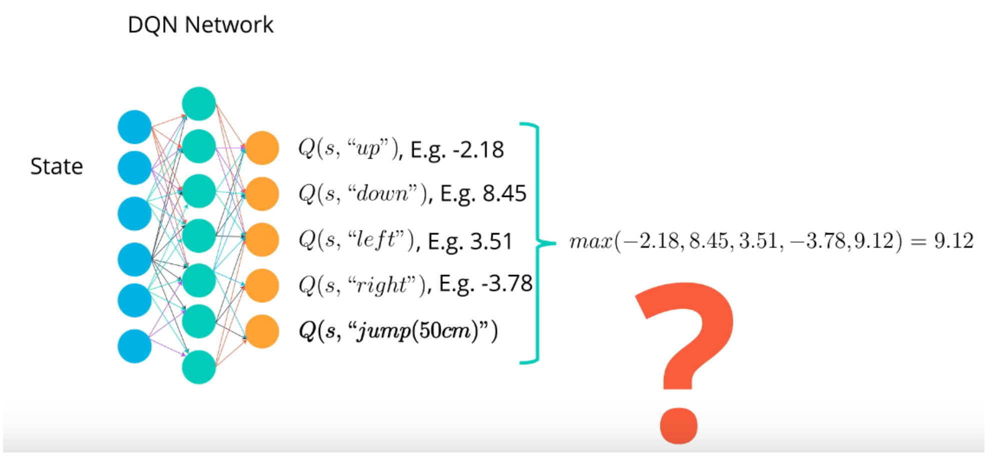
In On Policy, when policy used for interacting is the policy being learned. Eg: SARSA
In the Off Policy, when the policy is used for interacting with the environment is different from the policy being learned. Eg: Q-Learning

What is DDPG (Deep Deterministic Policy Gradient, Continuous control)?

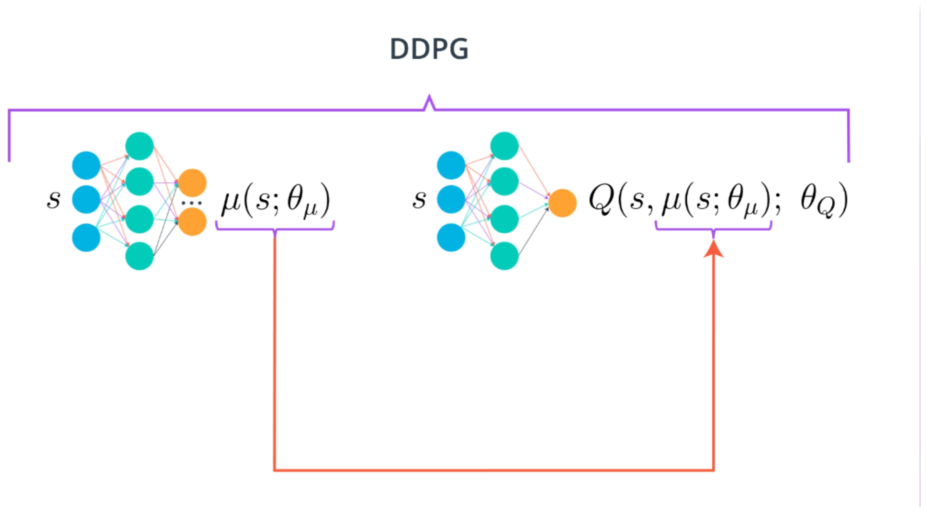
It's a different kind of Actor-Critic method, In DDPG the Critic is used to approximate the maximizer the vales of Q-vales of the next state.

Why not use DQN it approximates over continuous spaces?

DQN network takes in a state and gives an estimated return action-value function its quite good at giving out action-value function for n-actions space, but not capable of outputting a continuous range of action this is where DDPG can overcome it.


DQN Network
State
$Q(s, \text{“}up\text{”})$, E.g. -2.18
$Q(s, \text{“}down\text{”})$, E.g. 8.45
$Q(s, \text{“}left\text{”})$, E.g. 3.51
$Q(s, \text{“}right\text{”})$, E.g. -3.78
$Q(s, \text{“}jump(50cm)\text{”})$
$max(-2.18, 8.45, 3.51, -3.78, 9.12) = 9.12$
?

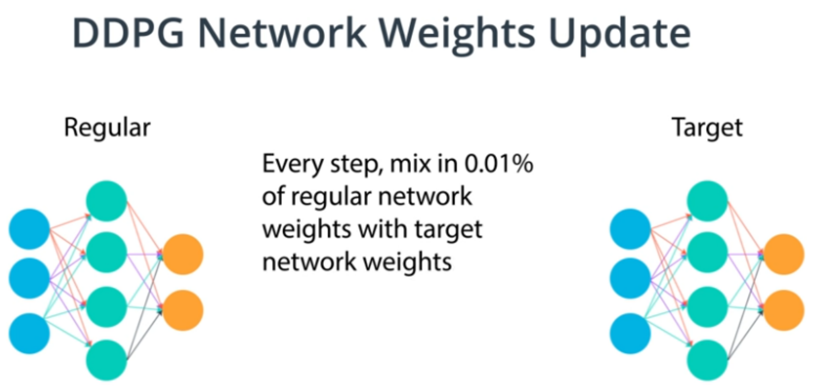
DQN network takes in a state and gives an estimated return action-value function its quite good at giving out action-value function for n-actions space, but not capable of outputting a continuous range of action this is where DDPG can overcome it.



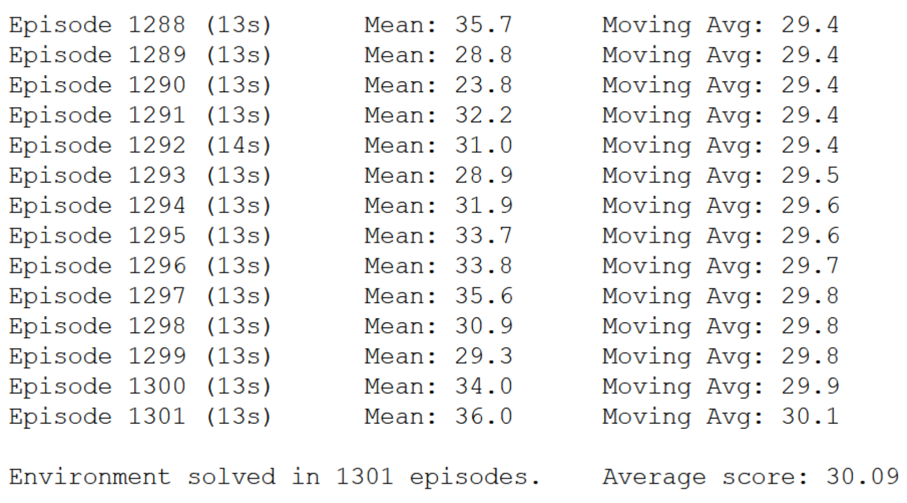
DDPG has 2 Neural networks 1. Actor, 2.Critic. The actor approximates the optimal Policy deterministically. The critic gets to evaluate the optimal policy by using the actor's best-believed action, we use this actor again to calculate the new target value based on it.

Interesting aspects of DDPG are :
-> Replay Buffer
-> Softupdate

In DDPG we have two copies of network weights for each network local
actor, target actor, local critic, target critic, but in DDPG targets are updated
using soft update strategy its slowely blending of regual network weights with
target


DDPG Network Weights Update
Regular
Target
Every step, mix in 0.01% of regular network weights with target network weights

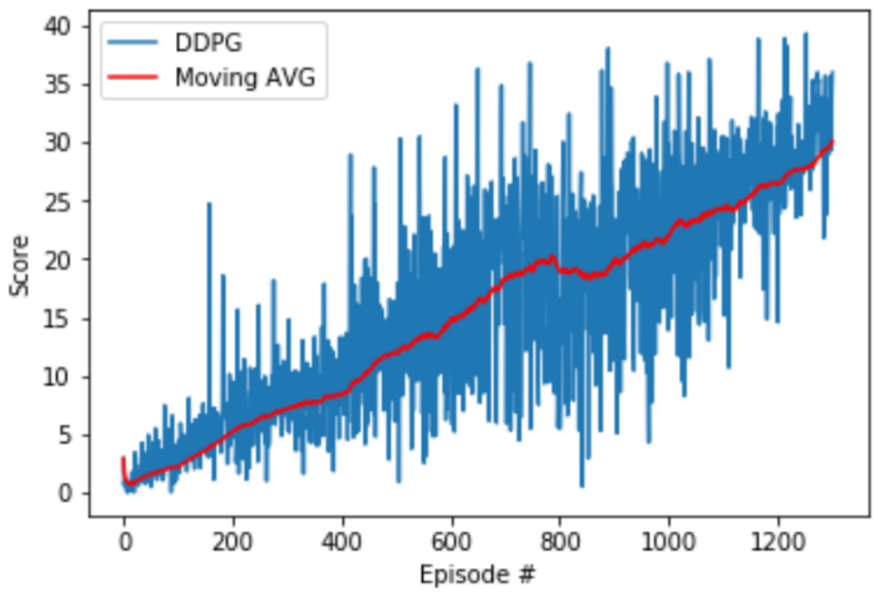
Model architecture
The Actor network consist of three fully connected layer with batchnormalization applied at the first layer. The network maps states to actions. It uses ReLU as activation function except the last layer where it use tanh.
The critic network also consist of three fully connected layer with batchnormalization applied at the first layer. The network maps maps (state,action) pairs to Q-values. It uses ReLU as activation function in the first two layers and no activation function for the last Hyperparameters:
fc1_units=400 # Number of nodes in the first hidden layer
fc2_units=300 # Number of nodes in the second hidden layer
BUFFER_SIZE = int(1e6) # 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-3 # learning rate of the actor
LR_CRITIC = 1e-3 # learning rate of the critic
WEIGHT_DECAY = 0 # L2 weight decay
LEARN_EVERY = 20 # learning timestep interval
LEARN_NUM = 10 # number of learning passes
GRAD_CLIPPING = 1.0 # gradient clipping
EPSILON = 1.0 # for epsilon in the noise process
EPSILON_DECAY = 1e-6 3 epsilon decay



```
Episode 1288 (13s)      Mean: 35.7      Moving Avg: 29.4
Episode 1289 (13s)      Mean: 28.8      Moving Avg: 29.4
Episode 1290 (13s)      Mean: 23.8      Moving Avg: 29.4
Episode 1291 (13s)      Mean: 32.2      Moving Avg: 29.4
Episode 1292 (14s)      Mean: 31.0      Moving Avg: 29.4
Episode 1293 (13s)      Mean: 28.9      Moving Avg: 29.5
Episode 1294 (13s)      Mean: 31.9      Moving Avg: 29.6
Episode 1295 (13s)      Mean: 33.7      Moving Avg: 29.6
Episode 1296 (13s)      Mean: 33.8      Moving Avg: 29.7
Episode 1297 (13s)      Mean: 35.6      Moving Avg: 29.8
Episode 1298 (13s)      Mean: 30.9      Moving Avg: 29.8
Episode 1299 (13s)      Mean: 29.3      Moving Avg: 29.8
Episode 1300 (13s)      Mean: 34.0      Moving Avg: 29.9
Episode 1301 (13s)      Mean: 36.0      Moving Avg: 30.1

Environment solved in 1301 episodes.    Average score: 30.09
```

The Score and the moving average has been plotted out


DDPG
Moving AVG
Score
Episode #