Reacher Environment – Continuous Control action space

The problem:

This is the second project of Udacity Reinforcement Learning Nanodegree program. Now, the challenge is to solve a Continuous Control Unity Environment: Reacher. 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 problem is considered solved when the agent gets an average score of +30 over 100 consecutive episodes.

Learning method - Continuous action space problem:

To solve the environment a DDPG (Deep Deterministic Policy Gradient) method was applied here. DDPG is a Reinforcement Learning approach that fits well on Continuous Control environments such as Reacher. It's an actor-critic algorithm, applicable to continuous action spaces.

Like DQN, the critic estimates the Q-value function using off-policy data and the recursive Bellman equation. The actor is trained to maximize the critic's estimated Q-values by back-propagating through both networks.

The use of two networks:

- Critic that measures how good the action taken is (value-based)
- Actor that controls how our agent behaves (policy-based)

The Critic: evaluate state-value function using the TD Estimate. Using the Critic we will calculate de Advantage function and train the action using this value.

The Actor: takes in the current environment state and determines the best action to take, the critic plays the evaluation role by taking in the environment state and action and returning an action score.

The Actor-Critic learning algorithm is used to represent the policy function independently of the value function. The policy function structure is known as the actor, and the value function structure is referred to as the critic. The actor produces an action given the current state of the environment, and the critic produces a TD (Temporal-Difference) error signal given the state and resultant reward. If the critic is estimating the action-value function Q(s, a), it will also need the output of the actor. The output of the critic drives learning in both the actor and the critic. Critic evaluates state-value function using the TD Estimate. Using the Critic we will calculate de Advantage function and train the action using this value.

DDPG algorithm incorporate both replay buffer and separate target network concepts from DQN for increase learning stability.

DDPG add noise to network parameters for better exploration. Parameter noise helps algorithms more efficiently explore the range of actions available to solve an environment.

Hyperparameters:

```
BUFFER_SIZE = int(1e5)
BATCH_SIZE = 256
GAMMA = 0.99
TAU = 1e-3
LR_ACTOR = 2e-4
LR_CRITIC = 2e-4
WEIGHT_DECAY = 0
```

The Network:

2 fully connected hidden layers with ReLU activations [Size: 400 / 400+action_size(Critic) and 300]
Batch normalization between the hidden layers

Tanh activation for Actor output and ReLU for Critic

Future Accomplishments:

A good next accomplishment here is to solve the same environment using other approach such as PPO for example. Other good idea is to solve the second version of this problem that has 20 agents instead of only one, using methods like PPO, A3C, and D4PG.

References:

https://blog.openai.com/better-exploration-with-parameter-noise/

https://arxiv.org/pdf/1509.02971.pdf

Udacity Reinforcement Learning Nanodegree Program