Project criteria

The goal of this project was to train robotic arm to maintain contact with the sphere in the environment. The agents must get an average score of +30, over 100 consecutive episodes.

Implementation

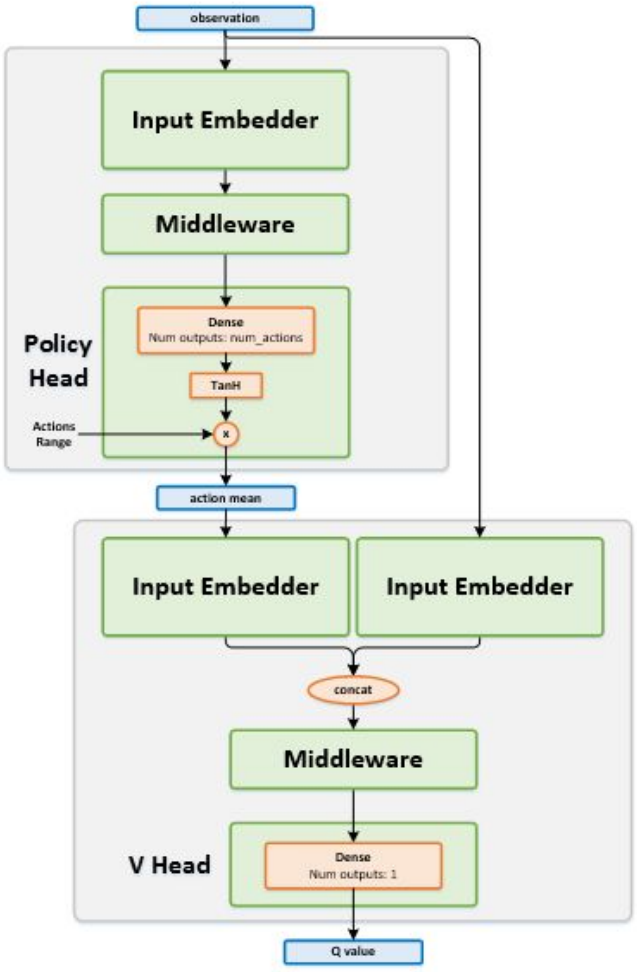
The algorithm that I have used is an implementation of Deep Deterministic Policy Gradient with Replay Buffer. I have also found that features such as gradient clipping and learning every 10 episodes but 10 times helped to speed up training process and made it more stable.

Original paper:

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

Helpful explaination:
https://towardsdatascience.com/deep-deterministic-policy-gradients-explained-2d94655a9b7b

DDPG is bringing the best from worlds of Policy methods and Value methods by introducing Actor and Critic networks focused on different sides of the problem. Actor learns the action based on observation, while Critic network learns the Q value of the state and the action produced by Actor. With the addition of target Actor and target Critic networks, which are slightly delayed copies of originals, we balance instabilities and allow step by step movement of the entire system towards the solution of the environment.



Reacher environment specific values:

- state_size = 33
- action_size = 4

Actor network:

- Hidden: (state_size, 400) - BatchNorm1d - ReLU -
- Hidden: (400, 300) - ReLU -
- Output: (300, action_size) - TanH

Critic network:

- Hidden: (state_size, 400) - BatchNorm1d - ReLU -
- Hidden: (400 + action_size, 300) - ReLU -
- Output: (300, action_size) - Linear

Hyperparameters:

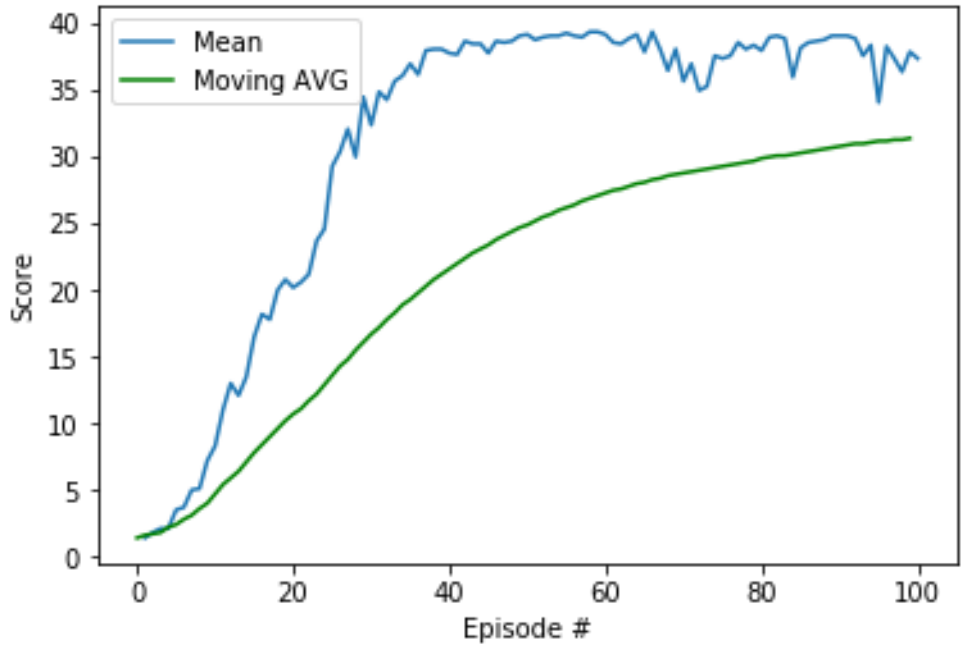
```
BUFFER_SIZE = int(1e6)          # replay buffer size
BATCH_SIZE = 128                # minibatch size
GAMMA = 0.99                    # discount factor
TAU = 0.0025                    # for soft update of target parameters
LR_ACTOR = 2e-4                   # learning rate of the actor
LR_CRITIC = 5e-4                  # learning rate of the critic
WEIGHT_DECAY = 0.0              # L2 weight decay

LEARN_EVERY = 20                # learning time step interval
LEARN_NUM = 10                  # num of learn per learn every

GRAD_CLIPPING = 1.0             # clamp gradient for critic local netowork

EPSILON       = 1.0             # for epsilon in the noise process (act step)
EPSILON_DECAY = 1e-6            # epsilon decay per learn
EPSILON_MIN = 0.1               # epsilon min
```

Results



The agents were able to solve task in 100 episodes with a final moving average score of 31.4.

Improvements

Improving the DDPG algorithm by applying Priority Experience Replay would be the next logical step. This addition greatly boosts stability and speed of learning.

I could also try:

- Experimenting with different RL algorithms may yield more stable training process (D3PG, D4PG, A3C, PPO) as this is one of the main flaws of DDPG.
- Improving results tuning the hyperparameters may decrease instability of the DDPG algorithm.
- Different Actor and Critic network structures could help with more efficient feature extraction.