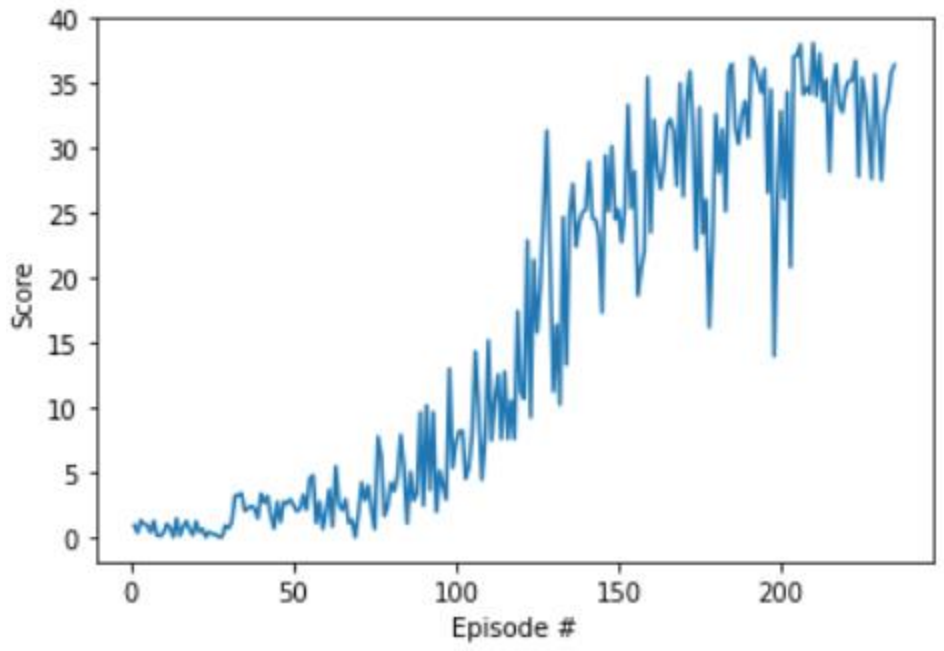
Report

The problem was solved in 235 episodes. The average score over 100 episodes was 30.11.



The major parameters are:

BUFFER_SIZE = int(1e6) # replay buffer size

BATCH_SIZE = 256   # minibatch size

GAMMA = 0.99   # discount factor

TAU = 1e-3   # for soft update of target parameters

LR_ACTOR = 1e-4   # learning rate of the actor

LR_CRITIC = 3e-4   # learning rate of the critic

WEIGHT_DECAY = 0   # L2 weight decay for Adam Optimizer

UPDATE_INTERVAL = 20   # Number of episodes elapsed between updates

UPDATE_FREQUENCY = 3   # Number of times parameters and policy will be updated from a sample of experiences in each update

max_t=2000 # maximum number of steps per episode

I use a Deep Deterministic Policy Gradient (DDPG) approach, which involves an actor network and a critic network.

The "Critic" estimates the value function. This could be the action-value (the Q value) or state-value (the V value). The "Actor" updates the policy distribution in the direction suggested by the Critic (such as with policy gradients). Both the Critic and Actor functions are parameterized with neural networks.

There are three other important features.

1.) Replay buffer - At each timestep the actor and critic are updated by sampling a minibatch uniformly from the buffer. Because DDPG is an off-policy algorithm, the replay buffer can be large, allowing the algorithm to benefit from learning across a set of uncorrelated transitions.

2.) Soft update - For both actor and critic, their target network weights are mixed with a small fraction of the regular network weights during each update. This can create faster convergence.

3.) Noise - A small amount of noise is added to each action in each time step. This can help with exploration. I found that Gaussian noise works much better than Ornstein-Uhlenbeck noise.

Since update interval and frequency were mentioned in the benchmark implementation, I experimented those as well as the batch size, which is the number of replay examples used in each update. The current configuration seems to generate the best results. Basically, I want to do a number of updates each time after a certain number of episodes.

Both actor and critic neural networks consist of multiple linear layers. Their structures mirror those in the Bipedal and Pendulum examples in the class's Github repository.

For future improvement, I will try adding noise to the parameters as well, which can lead to more consistent exploration and a richer set of behaviors (arXiv:1706.01905 [cs.LG]). I will also try priority experience replay: replaying important transitions more frequently can help agent learn more efficiently (arXiv:1511.05952v4 [cs.LG]). Another idea I want to explore further is adjusting the update frequency dynamically based on progress. The idea was tested but the result was so far inconclusive.