Report

CRITERIA

Learning Algorithm

The report clearly describes the learning algorithm, along with the chosen hyperparameters. It also describes the model architectures for any neural networks.

1) Learning algorithm

For the continuous task in this project, DDPG algorithm is used, which is considered an actor-critic algorithm. In DDPG algorithm, there still remain some important characteristics in DQN algorithm such as 1**) the critic's** implementing the network structure similar to the Q-Network in DQN and **following the same paradigm as in Q-learning** for training and 2) implementing a **replay buffer** in order to get experiences from the agent. In this project, the actor implements a current policy to try optimally mapping states to a desired action (to be used for the critic's input). With the gradients from maximizing the estimated Q-value from the critic, the actor is trained. In this way, **the critic and actor are trained interactively**.

Here, **Ornstein-Uhlenbeck noise**(to encourage exploration during training) is added to the actor's selected actions. Unlike for discrete action spaces (where exploration is done via epsilon-greedy or Boltzmann exploration, for instance), exploration is done adding noise to the action itself.

(Reference: T. Lillicrap et al., Continuous control with deep reinforcement learning (2015).)

Additionally, **Epsilon decay** is implemented to decrease an average scale of noise applied to actions.

On the actor's last output, the **Tanh activation** is used, which ensures that the range of every entry in the action vector is between -1 and 1.

For an optimizer, **Adam** is used for both actor and critic networks - the critic Adam's L2 weight decay is set 0.

Using a **soft update** strategy, target networks are updated, consisting of slowly blending in your regular network weights(actually trained, most up to date) with your target network weights(used for prediction to stabilize strain).

Lastly, a **learning interval** is defined to speed up the learning process – performing the learning step after a specific learning interval is passed.

2-1) Actor Network's architecture (mapping states to actions)

fc layer 1 with Batch Normalization and ReLU (input: 33 units (state_size), output: 400 units)

fc layer 2 with ReLU (input: 400 units, output 300 units)

fc layer 3 with Tanh (input: 300 units, output: 4 units (action_size))

2-2) Critic Network's architecture (mapping (state, action) pairs to Q-values)

fcs layer 1 with Batch Normalization and ReLU (input: 33 units (state_size), output: 400 units)

fc layer 2 with Concatenate[(output from layer 1, action), axis=1] & ReLU

(input: 404 units(400 + action_size), output 300 units)

fc layer 3 (input: 300 units, output: 1 unit)

3) Parameters used in DDPG algorithm

Maximum episodes: 10000

Maximum steps per episode: 1000

First target average score: 30.0

4) Parameters used in DDPG Agent

BUFFER_SIZE = 1e+6   # replay buffer size

BATCH_SIZE = 128   # minibatch size

GAMMA = 0.99   # discount factor

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

LR_ACTOR = 2e-4   # learning rate of the actor

LR_CRITIC = 2e-4   # learning rate of the critic

WEIGHT_DECAY = 0   # L2 weight decay

OU_MU = 0.0   # Ornstein-Uhlenbeck noise parameter

OU_SIGMA = 0.2   # Ornstein-Uhlenbeck noise parameter

OU_THETA = 0.15   # Ornstein-Uhlenbeck noise parameter

EPSILON_DECAY = 1e-6   # Decay value for epsilon (epsilon -> epsilon - EPSILON_DECAY)

LEARN_EVERY = 10 # time-step interval for learning

NUM_LEARN = 5 # number of learning with sampleing memory

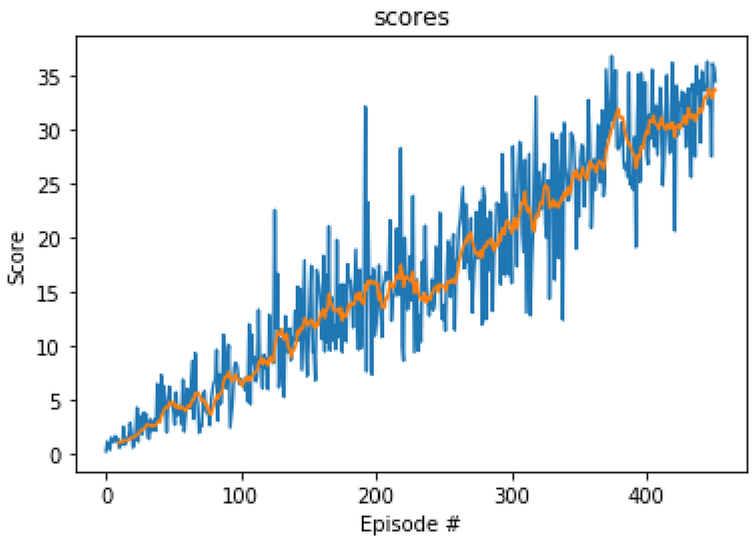
Plot of Rewards

A plot of rewards per episode is included to illustrate that either:

- *[version 1]* the agent receives an average reward (over 100 episodes) of at least +30, or
- *[version 2]* the agent is able to receive an average reward (over 100 episodes, and over all 20 agents) of at least +30.

The submission reports the number of episodes needed to solve the environment.

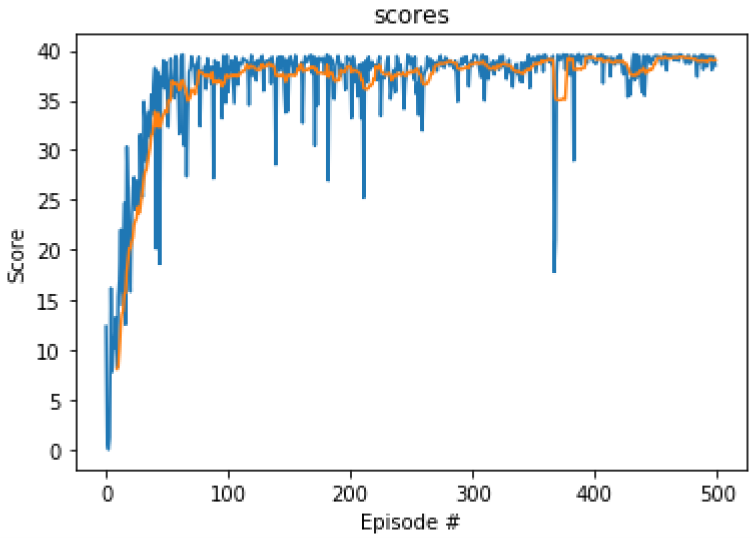
```
Episode 100	Average Score: 4.02
Episode 200	Average Score: 11.82
Episode 300	Average Score: 17.25
Episode 400	Average Score: 26.05
Episode 452	Average Score: 30.05
Environment solved in 352 episodes!	Average Score: 30.05
```



Additional training (from the above – loading the checkpoints of the above)

```
agent = Agent(state_size=state_size, action_size=action_size,
score = ddpg(target_mean_scores_deque=40.0, train_mode=True)
```

```
Episode 100	Average Score: 30.39
Episode 200	Average Score: 37.95
Episode 300	Average Score: 37.82
Episode 400	Average Score: 38.26
Episode 500	Average Score: 38.83
Episode 572	Average Score: 38.53
```



Ideas for Future Work

The submission has concrete future ideas for improving the agent's performance.

1) Utilizing Crawler-DDPG, PPO, TRPO, D4pG, A2C, A3C, Q-Prop, et cetera.

2) Implementing prioritized replay.

3) Optimizing hyperparameters more extensively.