Continuous Control Report

I used Deep Deterministic Policy Gradients (DDPG) to solve this Reacher Environment with 20 agents. As the scripts take the environment information of number of agents as an input, this can be applied to the environment with 1 agent as well.

Learning Algorithm

The final agents are trained with DDPG.

The environment has a state space of 33 dimensions and a 4-dimenional action space with continuous values. We cannot use value-based method such as DQN to solve this problem since the action space is continuous. So I pick DDPG for this environment.

DDPG is an off-policy Actor-Critic algorithm. Its actor learns a deterministic policy while its critic estimates the Q-value of the states and actions. Both networks maintain a separate target network to help stabilize the training. As the policy is deterministic, we also introduce noise in training to do exploration. To make the agent more robust, I also added gradient clipping with a max norm of 1 in the critic network.

Some hyperparameters used:

- Experience replay buffer size: 100000
- Training batch size: 128
- Reward discounting Gamma: 0.99
- Soft update Tau: 0.001
- Learning rate - Actor: 0.0001
- Learning rate - Critic: 0.001
- Weight decay for L2 regularization: 0
- Noise: Ornstein-Uhlenbeck process with mu=0, theta=0.15 and sigma=0.2

Here are the details of the Actor and Critic networks.

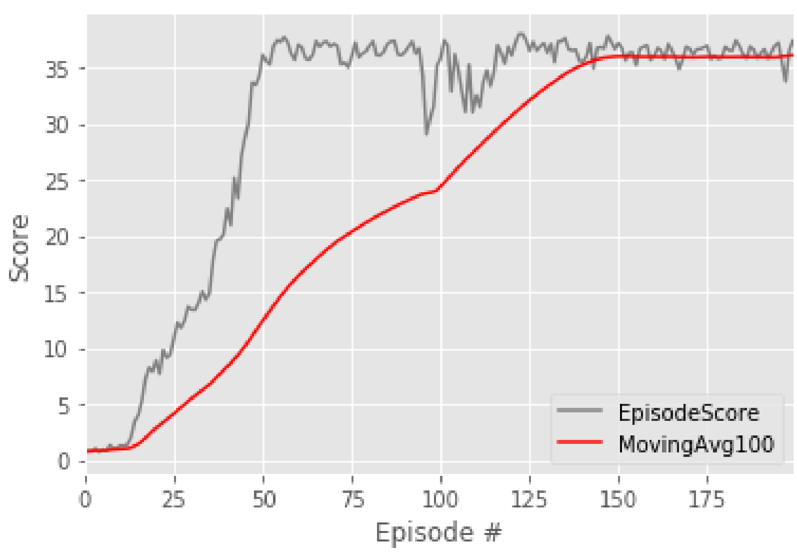
```
Actor(
  (fc1): Linear(in_features=33, out_features=256, bias=True)
  (fc2): Linear(in_features=256, out_features=4, bias=True)
)

Critic(
  (fcs1): Linear(in_features=33, out_features=256, bias=True)
  (fc2): Linear(in_features=260, out_features=256, bias=True)
  (fc3): Linear(in_features=256, out_features=128, bias=True)
  (fc4): Linear(in_features=128, out_features=1, bias=True)
)
```

Model Performance

the environment is considered solved when the average reward across all 20 agents over 100 episodes is 30+. The trained agents can solve it within 120 episodes with the architecture mentioned above. The average 100-episode reward is about 36 after 200 episodes of training. Here are the details:

```
Episode 10  Average Score: 0.94     Latest Score: 1.07
Episode 20  Average Score: 2.57     Latest Score: 7.89
Episode 30  Average Score: 5.24     Latest Score: 13.67
Episode 40  Average Score: 7.99     Latest Score: 20.14
Episode 50  Average Score: 11.97    Latest Score: 34.26
Episode 60  Average Score: 16.07    Latest Score: 35.68
Episode 70  Average Score: 19.02    Latest Score: 36.87
Episode 80  Average Score: 21.16    Latest Score: 36.43
Episode 90  Average Score: 22.88    Latest Score: 36.08
Episode 100       Average Score: 24.03    Latest Score: 35.18
Episode 110       Average Score: 27.38    Latest Score: 30.99
Episode 119       Average Score: 30.12    Latest Score: 35.37
 * Environment first solved in 119 episodes! Average Score: 
30.12. Continue training...
Episode 120       Average Score: 30.41    Latest Score: 36.85
Episode 130       Average Score: 33.06    Latest Score: 37.17
Episode 140       Average Score: 35.11    Latest Score: 35.58
Episode 150       Average Score: 35.97    Latest Score: 36.64
Episode 160       Average Score: 35.95    Latest Score: 35.66
Episode 170       Average Score: 35.90    Latest Score: 36.85
Episode 180       Average Score: 35.92    Latest Score: 35.85
Episode 190       Average Score: 35.90    Latest Score: 35.82
Episode 200       Average Score: 36.09    Latest Score: 37.35
```

Total Reward/Score per Episode



Ideas for Future Work

For this work, we can make more improvements by:

- N-step returns:
  - Instead of using just 1-step ahead, we can use N-steps to gain more information about the environment.
- Batch normalization:
  - This could help us find a better or faster solution by scaling the inputs.
- Prioritized experience replay:
  - Same as DQN, when sample from the replay buffer, we may want to give different weights to different experiences. This should be able to help us use the past experiences more efficiently
- Search for better architectures:
  - I tried two sets of different architectures for the Actor and Critic networks. The one shown above is significantly better than the other one in terms of converging speed. There might be better architectures that can help improve the performance further.