Udacity

Christian Leininger

Supervisor:
Advisor:

1 Introduction

This work is about solving the second project of the deep reinforcement learning course. In this environment the goal is to control a double-jointed arm. The agent receives an +0.1 reward if the arm is in the goal position, The agent observes an state vector with an 33 dimension corresponding to position, rotation, velocity, and angular velocities of the arm. The action has an 4 dimensional action space every entry is continuous between -1 and 1.

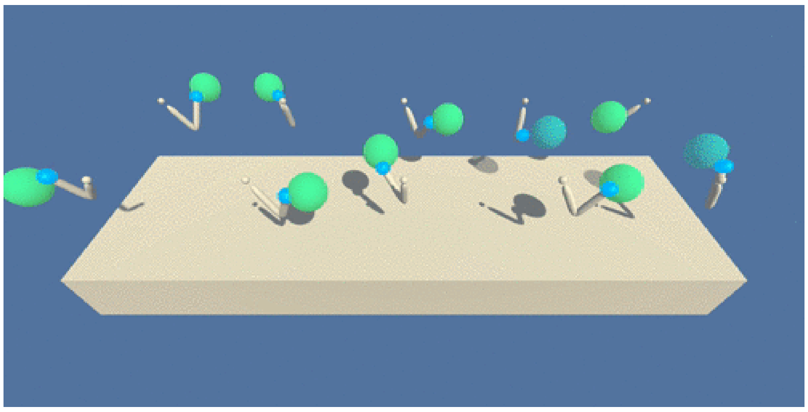

Figure 1: Continuous enviroment

2 Learning algorithm

2.1 hyper parameter

Actor:
nodes of hidden layer 1: 256
nodes of hidden layer 2: 128
Critic:
nodes of hidden layer 1: 256
nodes of hidden layer 2: 256
nodes of hidden layer 3: 128
memory-capacity = 1000000
batch-size = 1024
train-every-step = 20
learning-updates = 4
discount = 0.99
learning-rate = 0.002
leak = 0.01
epsilon = 1.0
epsilon min = 0.05
epsilon decay = 0.990

3 Architecture of the Neural Networks

Actor (policy) Network maps a given state to an action. Input layer shape of the states that uses batchnormal1d (input state has 1 dimension) to map all values between 0 and 1 this helps to avoid large or vanishing gradients during the back propagation. first hidden layer linear fully connected 256 nodes, second layer has 128 nodes As activation function the leaky relu was used to overcome the issue that the gradient is negative

$$f(x) = \begin{cases} x & , if\ x > 0 \\ ax & , else \end{cases} \qquad f'(x) = \begin{cases} 1, if\ x > 0 \\ a, else \end{cases}$$

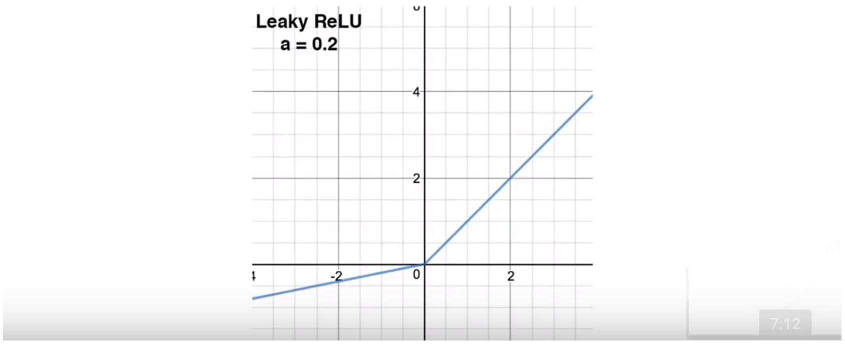

Figure 2: the leaky relu activation function

Critic (value function) Network maps a given state and action to the Q-value Input layer shape of the states and actions, first hidden layer linear fully connected 256 nodes, second layer has 128 nodes

4 DDPG Algorithm

The following algorithm was proposed in the paper CONTINUOUS CONTROL WITH DEEP REINFORCEMENT LEARNING [1]

Algorithm 1 DDPG algorithm

```
Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $R$
for episode = 1, M do
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state $s_1$
  for t = 1, T do
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$
    Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
    Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:
        $$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$
    Update the target networks:
        $$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
        $$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$
  end for
end for=0
```

What is the algorithm doing first the Actor and Critic online network the weights are initialized with random variables from a normal distribution and transferred to the target network weights. Also the Replay Buffer is created. Then loop over the given amounts of episodes until the goal is reached or run out of episodes. At the beginning of every episode the Ornstein-Uhlenbeck process is reset to the given mean and the environment is reset. Then select the action according to the policy actor network. The environment returns next state, reward and if its done this will be saved in the Replay Buffer $s_t, a_t, r_t, s_{t+1},$.
Sample a random mini batch calculate the bellman equation. Change the network weights from the critic to minimize the loss.Use policy gradient to change the actor network weights

according the equation. Then soft update the target network with the online network.

5 Results

One Agent with one online and target network for each Actor and Critic was used to solve the second version with the 20 different agents. The Following plot shows how the average reward is changing over the episodes until the it reaches the goal of 30 points averaged over the last 100 episodes, in the 140 episode.

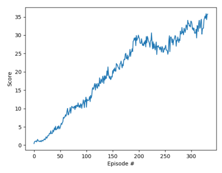

Figure 3: Average scores of the different agents

6 Future ideas to improve the performance

Search for even better hyper-parameters. Use an prioritized experience replay buffer. Use a different algorithm D4PG or one that uses planning.

References

[1] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.