CS294-112 Deep Reinforcement Learning HW3

Heri Zhao

Fall 2018

Part1 Q-learning

Question 1: basic Q-learning performance

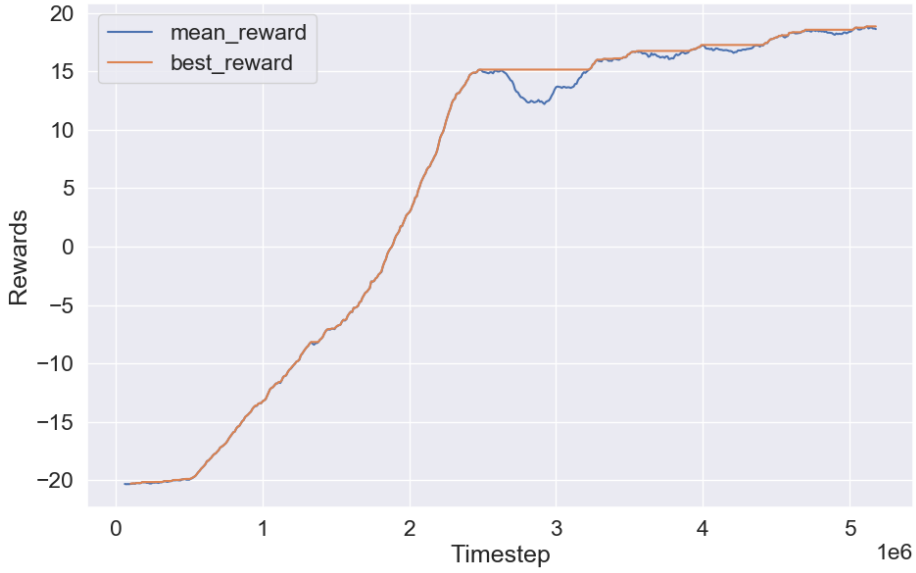

Figure 1: Learning curves for basic Q-learning on game Pong

```
python run_dqn_atari.py
```

Question 2: double Q-learning

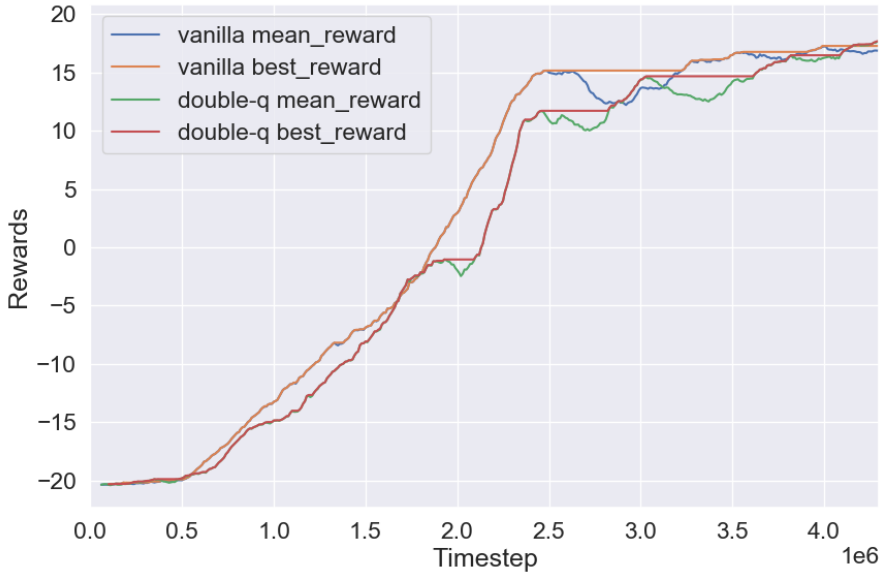

Figure 2: Learning curves for vanilla vs double Q-learning on game Pong

```
# manually change line 87 in run_dqn_atari.py to pass double_q=True
python run_dqn_atari.py
```

We can see that both vanilla and double Q-learning reach to rewards 15+ after 4M timesteps. The double Q-learning is slightly better than vanilla at 4M timesteps. This matches the Figure 4 of this paper. Also noticed that the learning curve is little slow for double Q-learning, however this probably because of the random seed.

Question 3: experimenting with hyperparameters

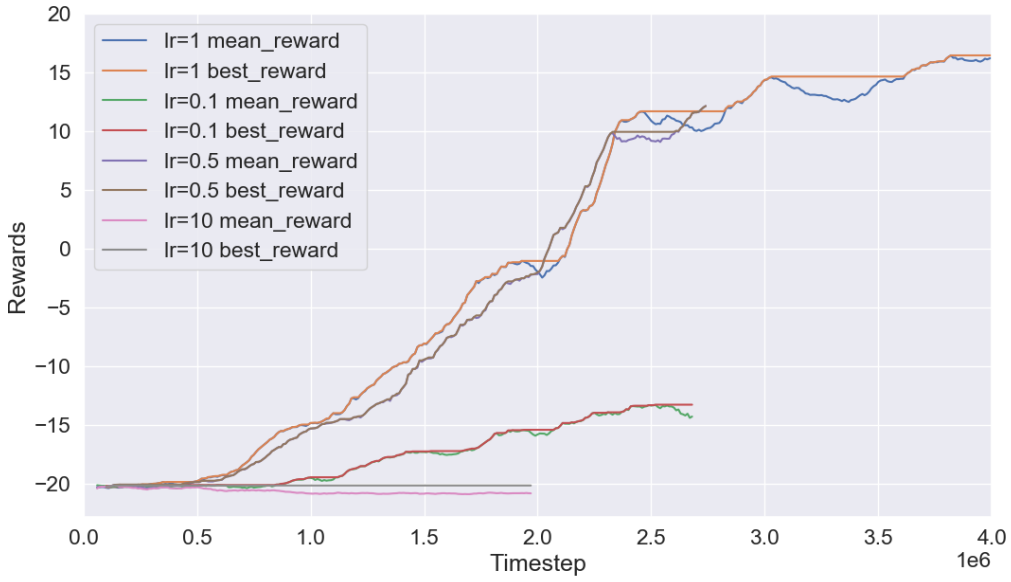

Figure 3: Learning curves of *learning rate multiplier=0.1, 0.5, 1, 10* on game Pong with double Q-learning

```
# manually change line 87 in run_dqn_atari.py to pass double_q=True
python3 run_dqn_atari.py 0.1
python3 run_dqn_atari.py 0.5
python3 run_dqn_atari.py 10
```

Here we choose the *learning rate multiplier* to experiment with the hyper-parameters. We can see that with learning rate very large the Q-learning basically can not learn anything, the rewards of lr=10 did not increase after 2M timesteps. The curve of lr=0.1 is actually increasing, but slowly, it probably will get the high rewards, but will take more time. The performance of lr=0.5 is very much similar to lr=1.

Part2 Actor-Critic

Question 1: Sanity check with Cartpole

Figure 4: Learning curves of different number of target updates and number of gradient combinations on CartPole with Actor-Critic. We can see that (ntu=10, ngsptu=10) is best parameter combination, as it converges to the optimal rewards fast and more stable than (1, 100).

Explanation: Since we are using neural network to fit value function, (ntu=1, ngsptu=100) means we only update target once, while taking many gradient descent steps. This will probably bring network too close to the current target, instead of fitting to get a more generalized value function. The bias will be larger compared with (ntu=10, ngsptu=10). For (ntu=100, ngsptu=1), we only take one gradient step, so it is likely that we will not have chance to fit value function, and finally get larger bias.

Question 2: Run actor-critic with more difficult tasks

We can see that the performance is similar to what we got in HW2, but slightly worse.

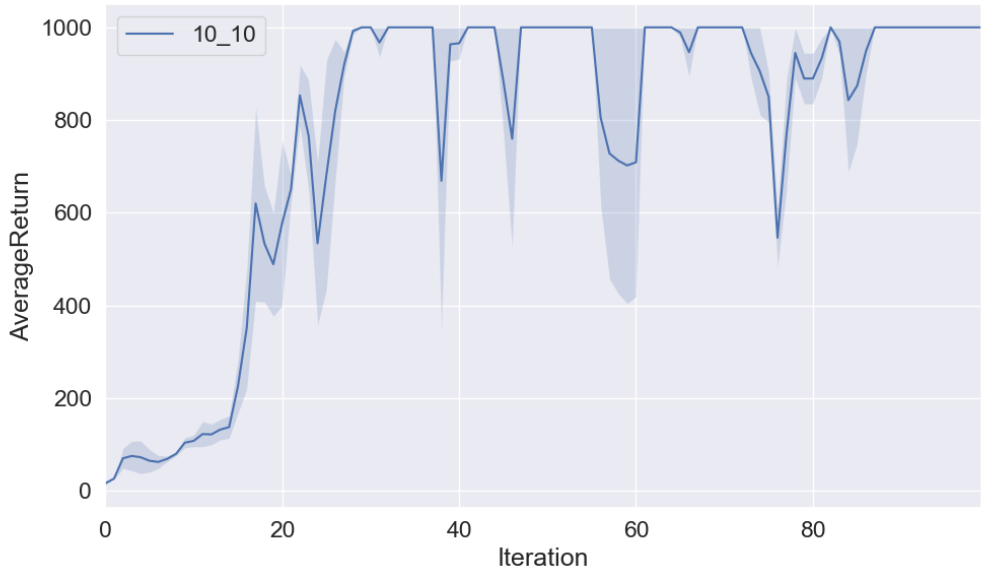

Figure 5: Learning curves of InvertedPendulum with ntu=10 and ngsptu=10

```
python train_ac_f18.py InvertedPendulum-v2 -ep 1000 --discount 0.95 -n 100 -e
    3 -l 2 -s 64 -b 5000 -lr 0.01 -clr 0.01 --exp_name 10_10 -ntu 10 -ngsptu
    10
```

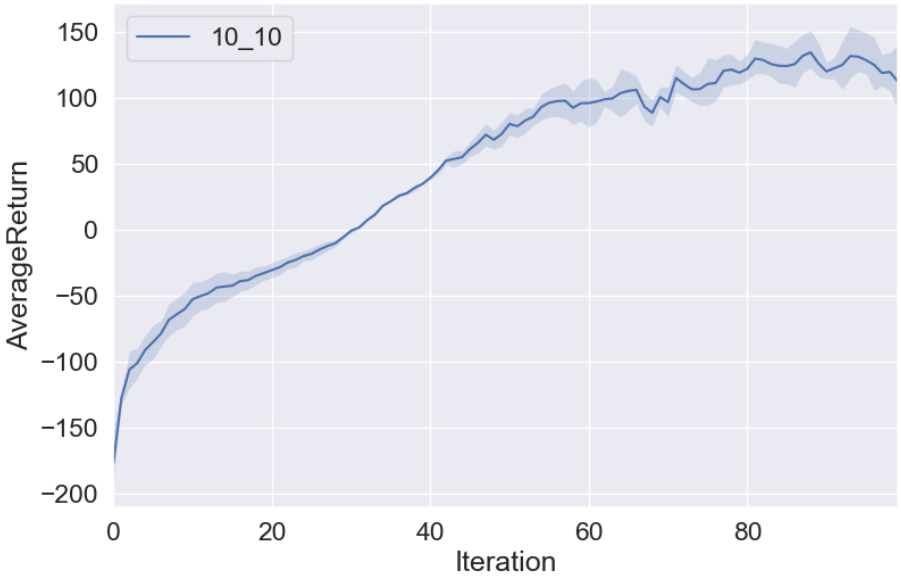

Figure 6: Learning curves of HalfCheetah with ntu=10 and ngsptu=10

```
python train_ac_f18.py HalfCheetah-v2 -ep 150 --discount 0.90 -n 100 -e 3 -l 2
    -s 32 -b 30000 -lr 0.02 -clr 0.02 --exp_name 10_10 -ntu 10 -ngsptu 10
```

Bonus

Here I am trying to tune actor-critic such that it could perform better.

1. Add 2 more hidden layers for critic network.
2. Use separate smaller learning rate to train critic network (0.01, 0.001).
3. Increase the number of target updates to 20.

From the Figure 7, we can see that after tunning, the performance is getting better, and is very similar to what we got in HW2. The best one has critic_learning_rate=0.01, ntu=20, ngsptu=10, and this also has lower variance.

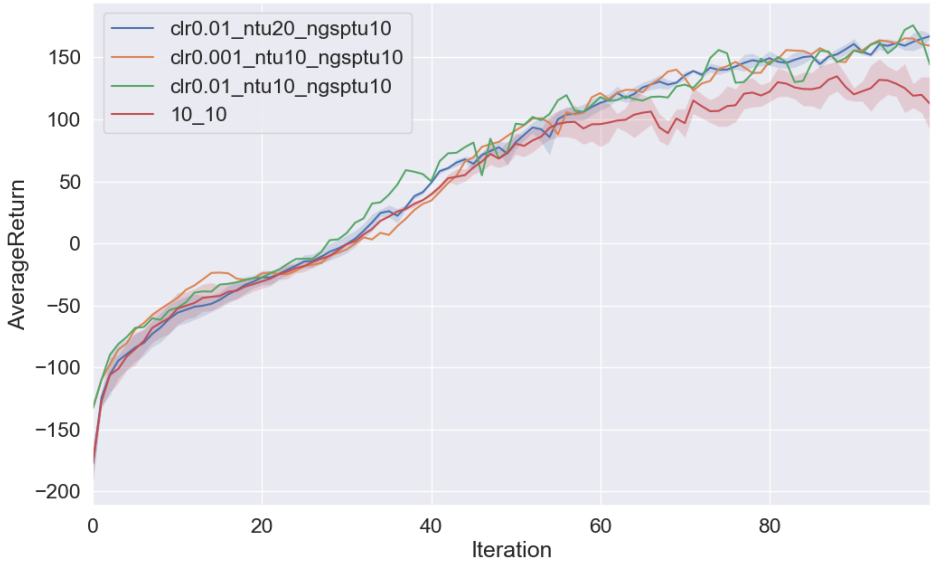

Figure 7: Learning curves of HalfCheetah. *clr* is representing critic-learning-rate.

```
# please change line 263 in train_ac_f18.py to use "n_layers=self.n_layers +
    2" before running
python train_ac_f18.py HalfCheetah-v2 -ep 150 --discount 0.90 -n 100 -e 3 -l 2
     -s 32 -b 30000 -lr 0.02 -clr 0.01 --exp_name clr0.01_ntu10_ngsptu10 -ntu
    10 -ngsptu 10
python train_ac_f18.py HalfCheetah-v2 -ep 150 --discount 0.90 -n 100 -e 3 -l 2
     -s 32 -b 30000 -lr 0.02 -clr 0.001 --exp_name clr0.001_ntu10_ngsptu10 -
    ntu 10 -ngsptu 10
python train_ac_f18.py HalfCheetah-v2 -ep 150 --discount 0.90 -n 100 -e 3 -l 2
     -s 32 -b 30000 -lr 0.02 -clr 0.01 --exp_name clr0.01_ntu20_ngsptu10 -ntu
    20 -ngsptu 10
```