CS 285 Homework 3

Jeffrey Cheng

October 18, 2022

1 Part 1: Q-Learning

Question 1: basic Q-learning performance (DQN)

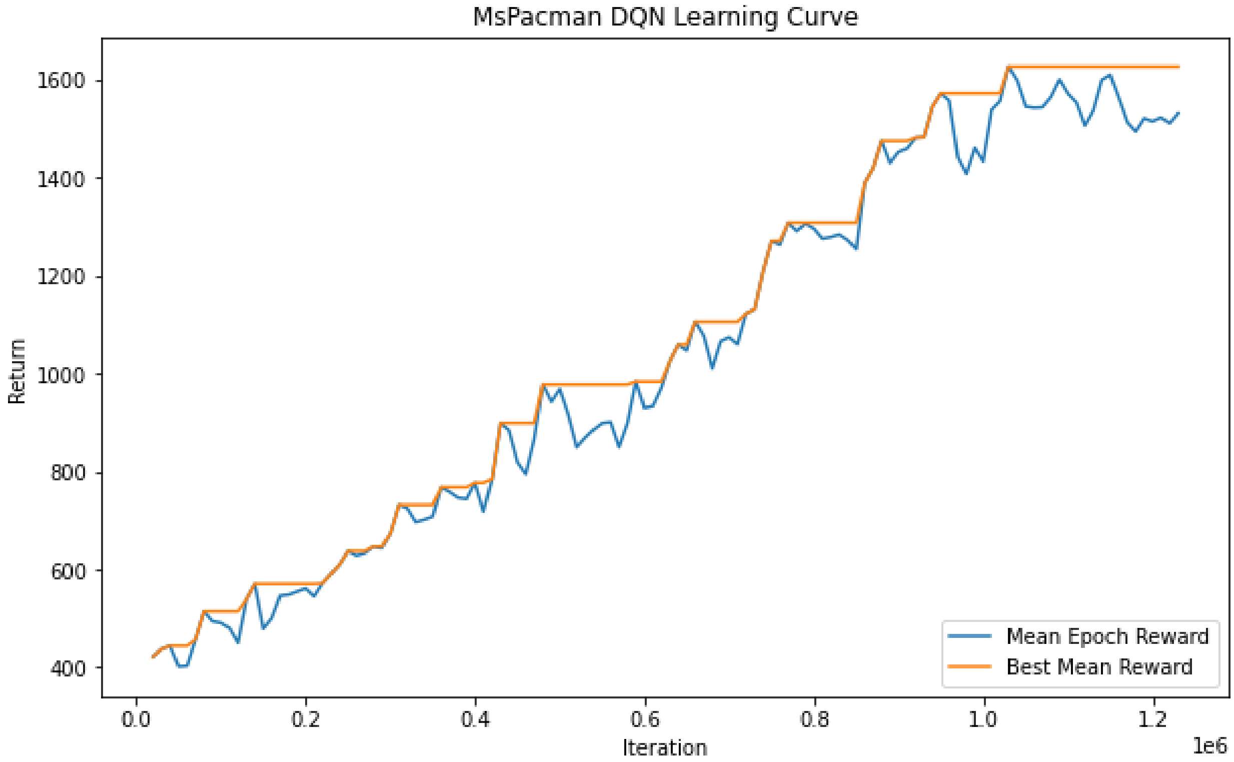

Figure 1: Learning curve for deep Q-learning in the MsPacman environment. Steady and quite stable learning.

The following command was used in this experiment:

```
python python cs285/scripts/run_hw3_dqn.py --env_name MsPacman-v0 --exp_name q1
```

Question 2: double Q-learning (DDQN)

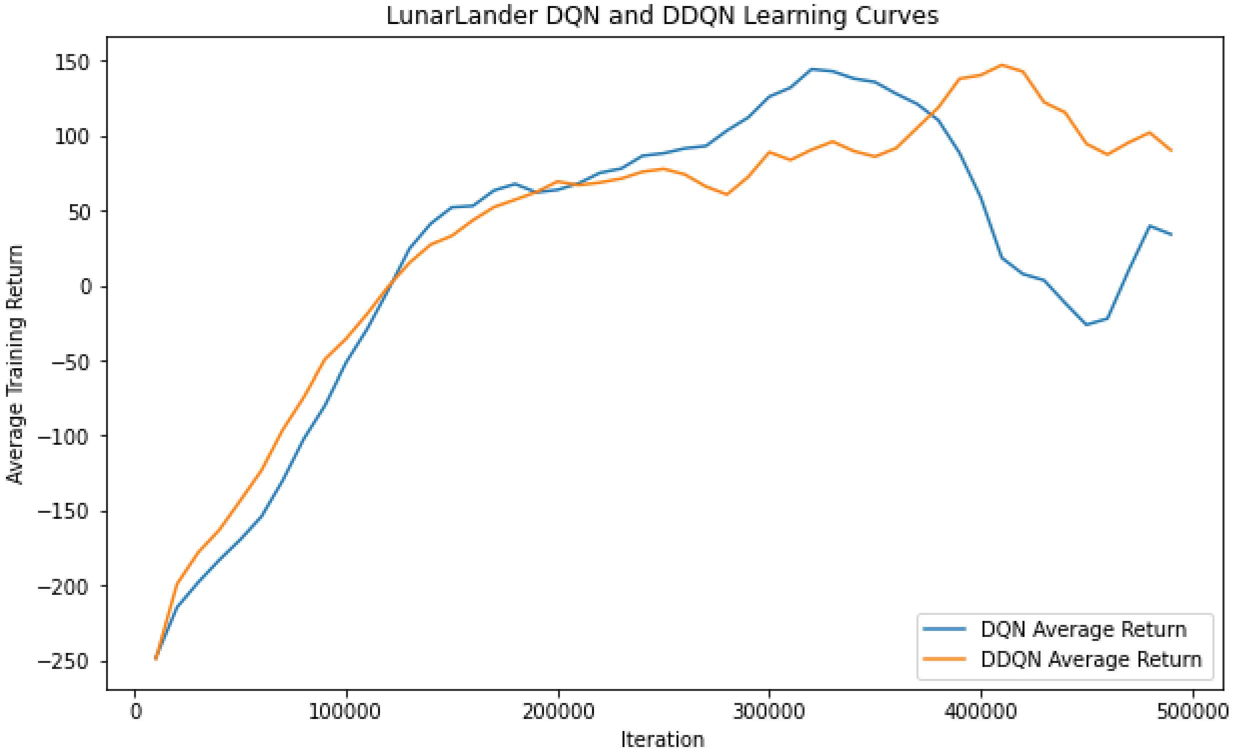

Figure 2: Learning curves for deep double Q-learning in the LunarLander environment, with each curve averaged over three random seeds. The averaged DQN curve exhibits drop-off at the end of training, whereas the averaged DDQN curve does not.

The following commands were used in this experiment:

```
python cs285/scripts/run_hw3_dqn.py --env_name LunarLander-v2 \
    --exp_name q2_dqn_1 --seed 1
python cs285/scripts/run_hw3_dqn.py --env_name LunarLander-v2 \
    --exp_name q2_dqn_2 --seed 2
python cs285/scripts/run_hw3_dqn.py --env_name LunarLander-v2 \
    --exp_name q2_dqn_3 --seed 3

python cs285/scripts/run_hw3_dqn.py --env_name LunarLander-v2 \
    --exp_name q2_doubledqn_1 --double_q --seed 1
python cs285/scripts/run_hw3_dqn.py --env_name LunarLander-v2 \
    --exp_name q2_doubledqn_2 --double_q --seed 2
python cs285/scripts/run_hw3_dqn.py --env_name LunarLander-v2 \
    --exp_name q2_doubledqn_3 --double_q --seed 3
```

Question 3: experimenting with hyperparameters

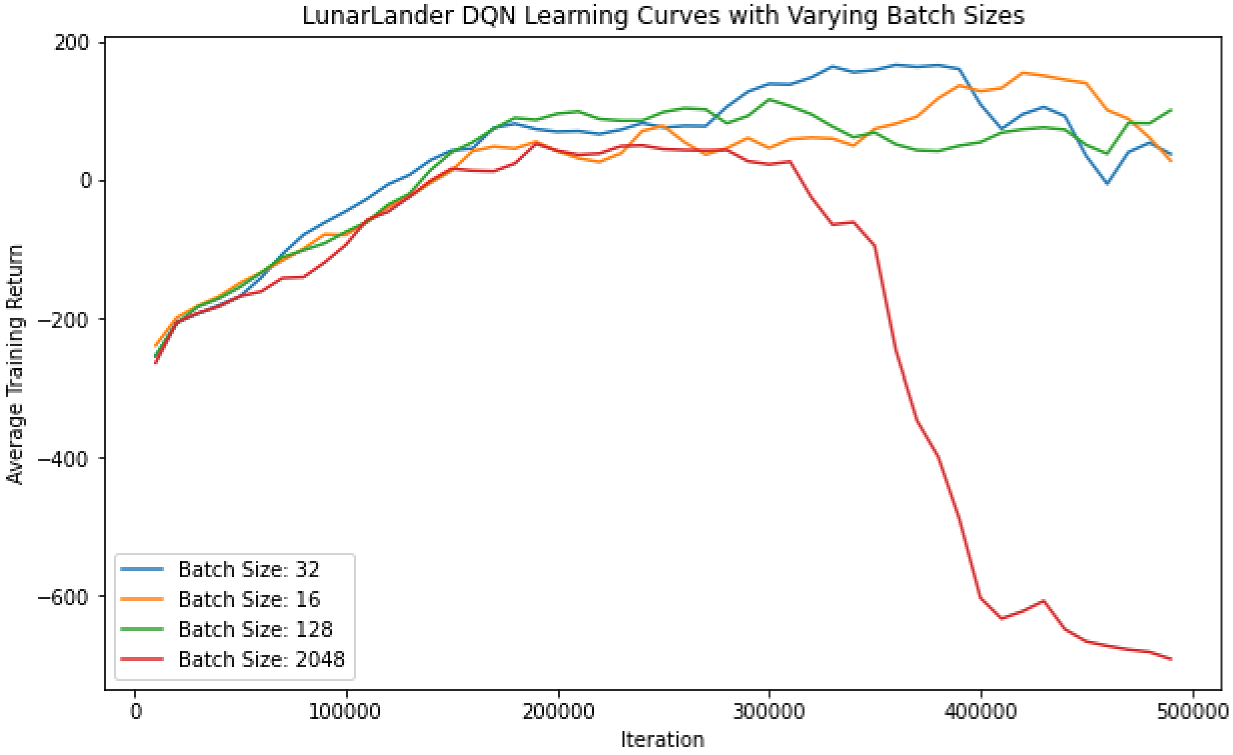

Figure 3: Learning curves for deep Q-learning in the LunarLander environment, using varying training batch sizes. Interestingly, the training with the largest batch size performs the worst after a while, getting terrible returns towards the end of training. A batch size of 32 is default.

The following commands were used in this experiment:

```
python cs285/scripts/run_hw3_dqn.py --env_name LunarLander-v2 \
    --exp_name q2_dqn_1 --seed 1

python cs285/scripts/run_hw3_dqn.py --env_name LunarLander-v3 \
    --batch_size 16 --exp_name q3_hparam1
python cs285/scripts/run_hw3_dqn.py --env_name LunarLander-v3 \
    --batch_size 128 --exp_name q3_hparam2
python cs285/scripts/run_hw3_dqn.py --env_name LunarLander-v3 \
    --batch_size 2048 --exp_name q3_hparam3
```

An experiment from question 2 (first command) was reused to produce the default batch size learning curve.

2 Part 2: Actor-Critic

Question 4: Sanity check with Cartpole

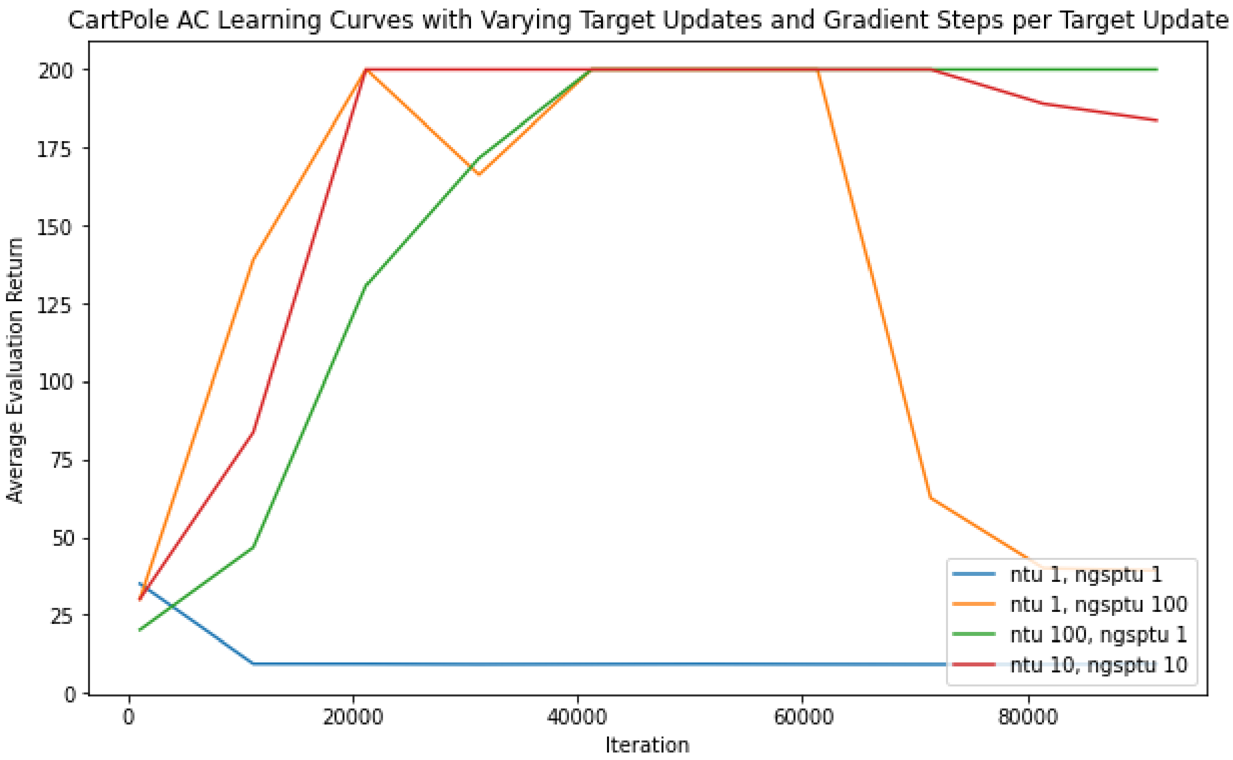

Figure 4: Learning curves for actor critic reinforcement learning in the Cartpole environment with varying target updates and gradient steps per target update.

We see that 1 target update and 1 gradient update is not enough updates to lead to good learning. Having 100 gradient steps per target update learns quickly, but seems to be relatively unstable. Having 10 target updates and 10 gradient steps per target update seems to work the best, learning quickly and having relatively stable returns. 100 target updates with 1 gradient steps per target update learns slightly slower, but has stable returns. As a result, 10 target updates and 10 gradient steps per target update were used for question 5.

The following commands were used in this experiment:

```
python cs285/scripts/run_hw3_actor_critic.py --env_name CartPole-v0 \
    -n 100 -b 1000 --exp_name q4_ac_1_1 -ntu 1 -ngsptu 1
python cs285/scripts/run_hw3_actor_critic.py --env_name CartPole-v0 \
    -n 100 -b 1000 --exp_name q4_ac_1_100 -ntu 1 -ngsptu 100
python cs285/scripts/run_hw3_actor_critic.py --env_name CartPole-v0 \
    -n 100 -b 1000 --exp_name q4_ac_100_1 -ntu 100 -ngsptu 1
python cs285/scripts/run_hw3_actor_critic.py --env_name CartPole-v0 \
    -n 100 -b 1000 --exp_name q4_ac_10_10 -ntu 10 -ngsptu 10
```

Question 5: Run actor-critic with more difficult tasks

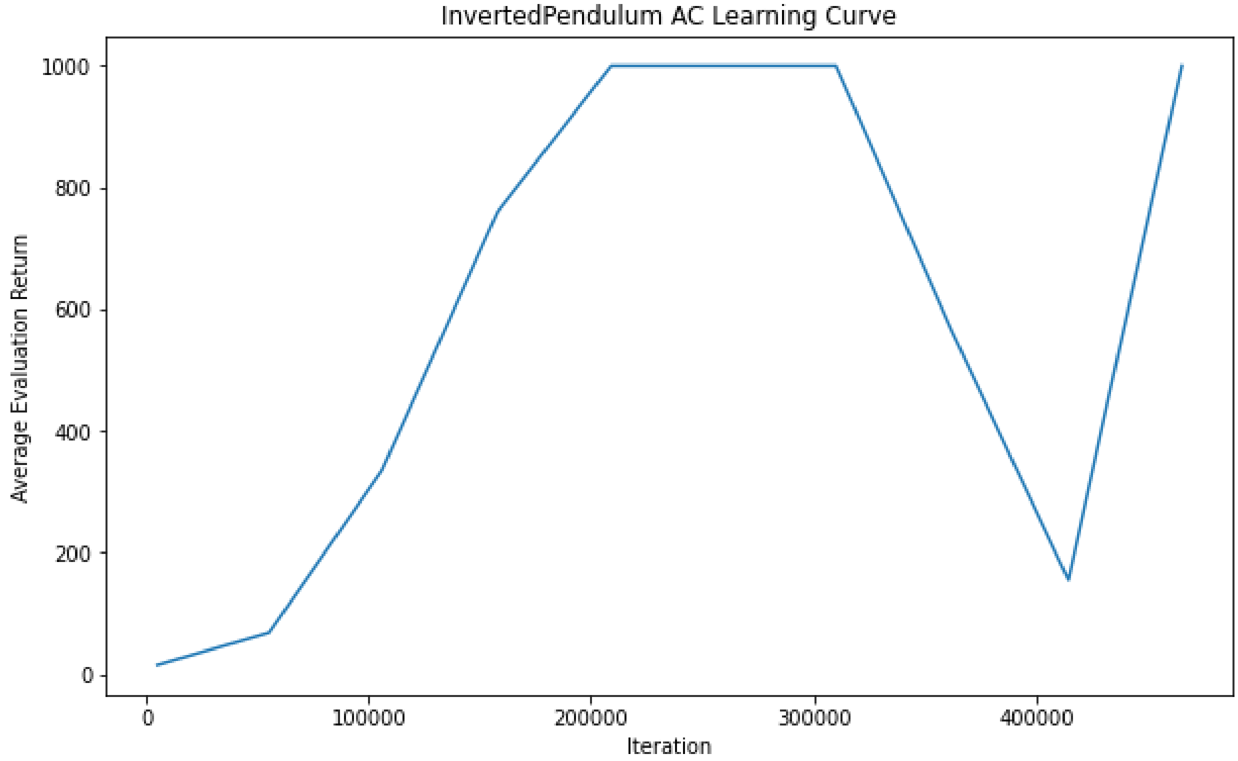

Figure 5: Learning curve for Actor-Critic learning in the InvertedPendulum environment, using 10 target updates and 10 gradient steps per target update. Exhibits unstable behavior towards end of learning.

The following command was used in this experiment:

```
python cs285/scripts/run_hw3_actor_critic.py --env_name InvertedPendulum-v4 \
    --ep_len 1000 --discount 0.95 -n 100 -l 2 -s 64 -b 5000 -lr 0.01 \
    --exp_name q5_10_10 -ntu 10 -ngsptu 10
```

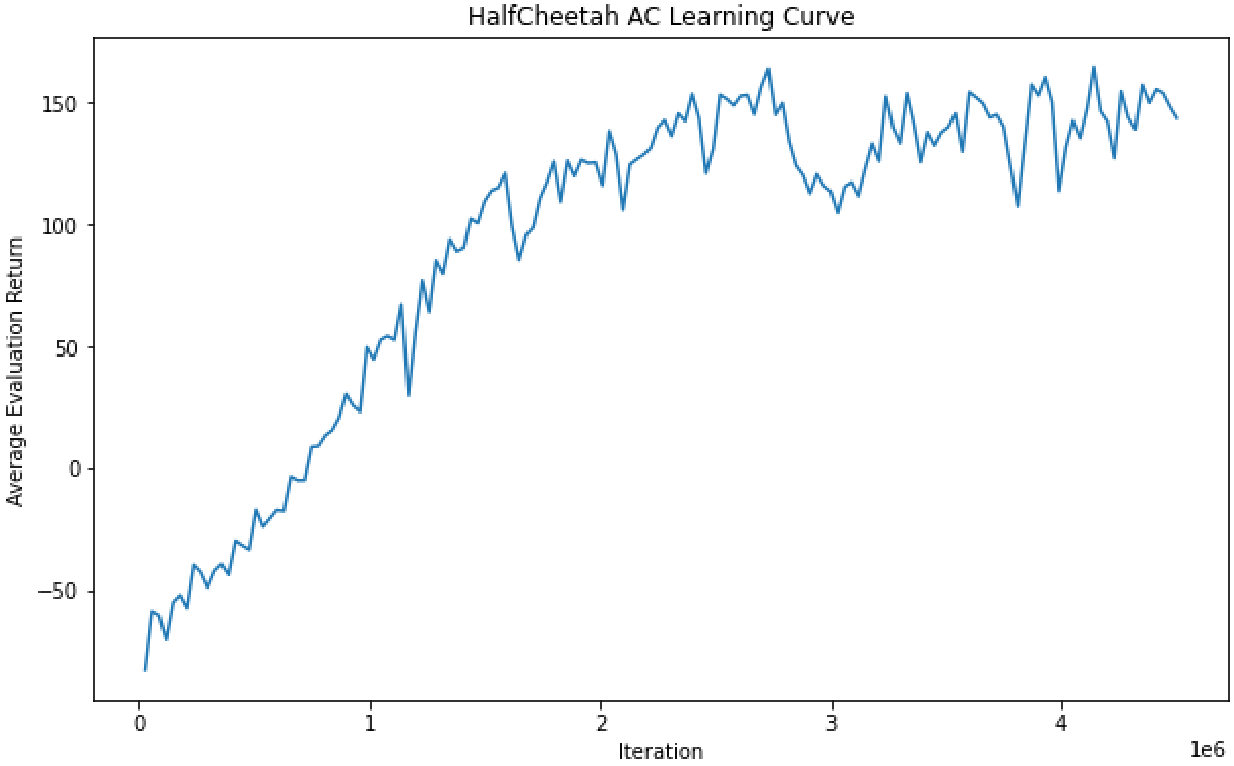

Figure 6: Learning curve for Actor-Critic learning in the HalfCheetah environment, using 10 target updates and 10 gradient steps per target update. Exhibits fairly stable learning. Performs slightly worse than the best policy gradient algorithm from the last homework.

The following command was used in this experiment:

```
python cs285/scripts/run_hw3_actor_critic.py --env_name HalfCheetah-v4 \
    --ep_len 150 --discount 0.90 --scalar_log_freq 1 -n 150 -l 2 -s 32 \
    -b 30000 -eb 1500 -lr 0.02 --exp_name q5_10_10 -ntu 10 -ngsptu 10
```

3 Part 3: Soft Actor-Critic

Question 6: Run soft actor-critic with more difficult tasks

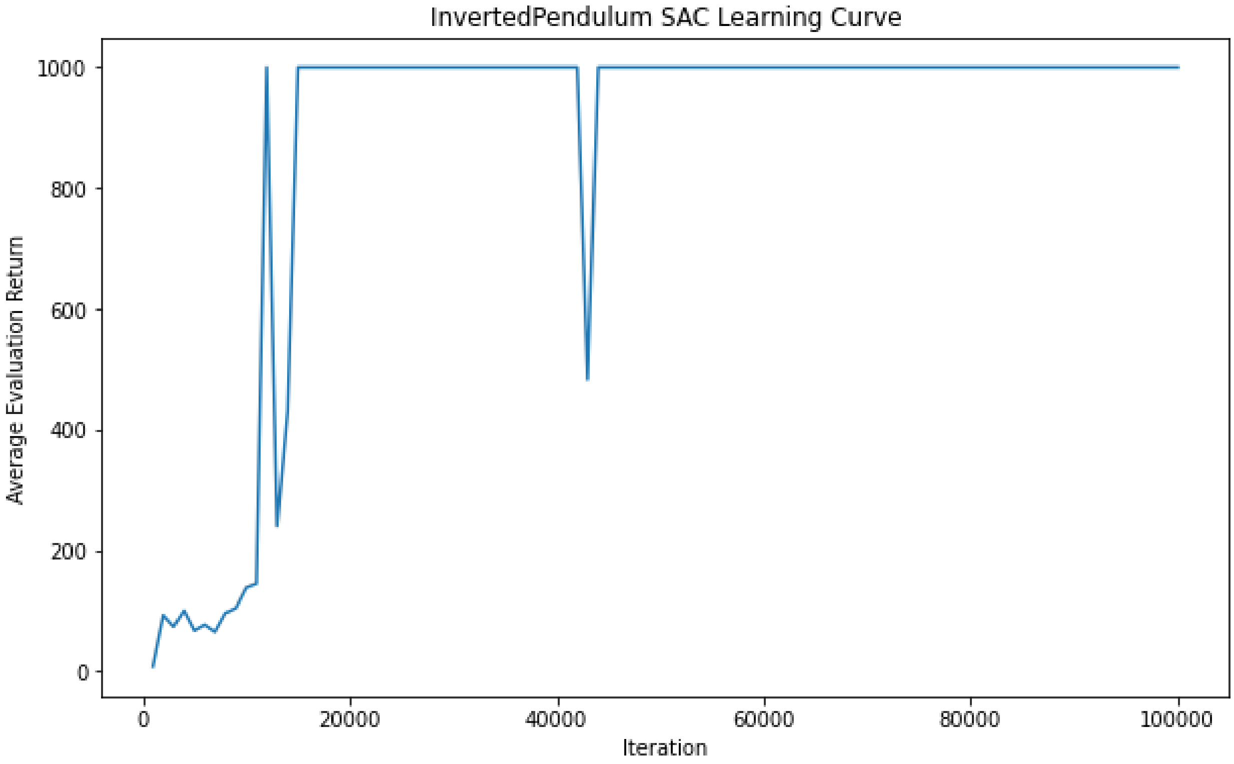

Figure 7: Learning curve for Soft Actor-Critic learning in the InvertedPendulum environment. Reaches maximum reward quickly and is more stable than regular actor critic learning.

The following command was used in this experiment:

```
python cs285/scripts/run_hw3_sac.py --env_name InvertedPendulum-v4 \
    --ep_len 1000 --discount 0.99 --scalar_log_freq 1000 -n 100000 \
    -l 2 -s 256 -b 1000 -eb 2000 -lr 0.0003 --init_temperature 0.1 \
    --exp_name q6a_sac_InvertedPendulum --seed 1
```

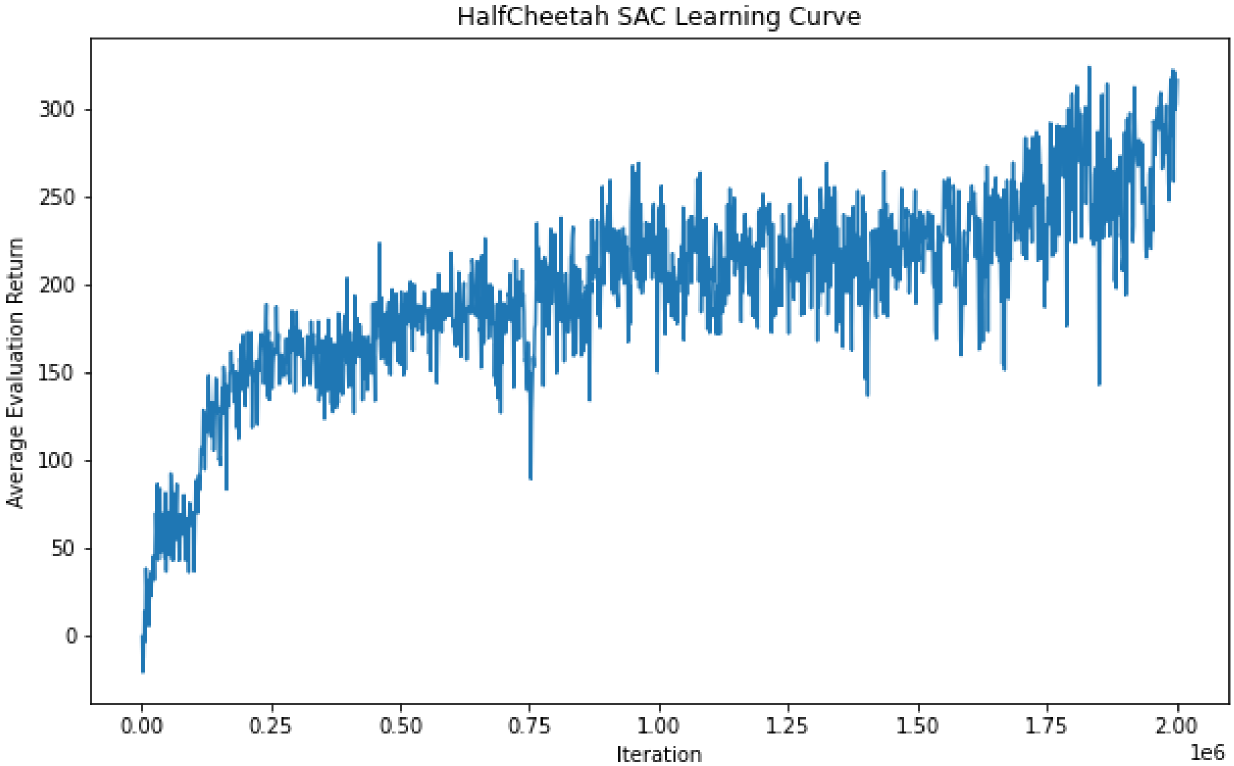

Figure 8: Learning curve for Soft Actor-Critic learning in the InvertedPendulum environment. Reaches higher returns compared to standard actor-critic learning.

The following command was used in this experiment:

```
python cs285/scripts/run_hw3_sac.py --env_name HalfCheetah-v4 \
    --ep_len 150 --discount 0.99 --scalar_log_freq 1500 \
    -n 2000000 -l 2 -s 256 -b 1500 -eb 1500 -lr 0.0003 \
    --init_temperature 0.1 --exp_name q6b_sac_HalfCheetah --seed 1
```