CS 285 Set 2

Erich Liang

Due: 9/27/21

1 Experiment 1

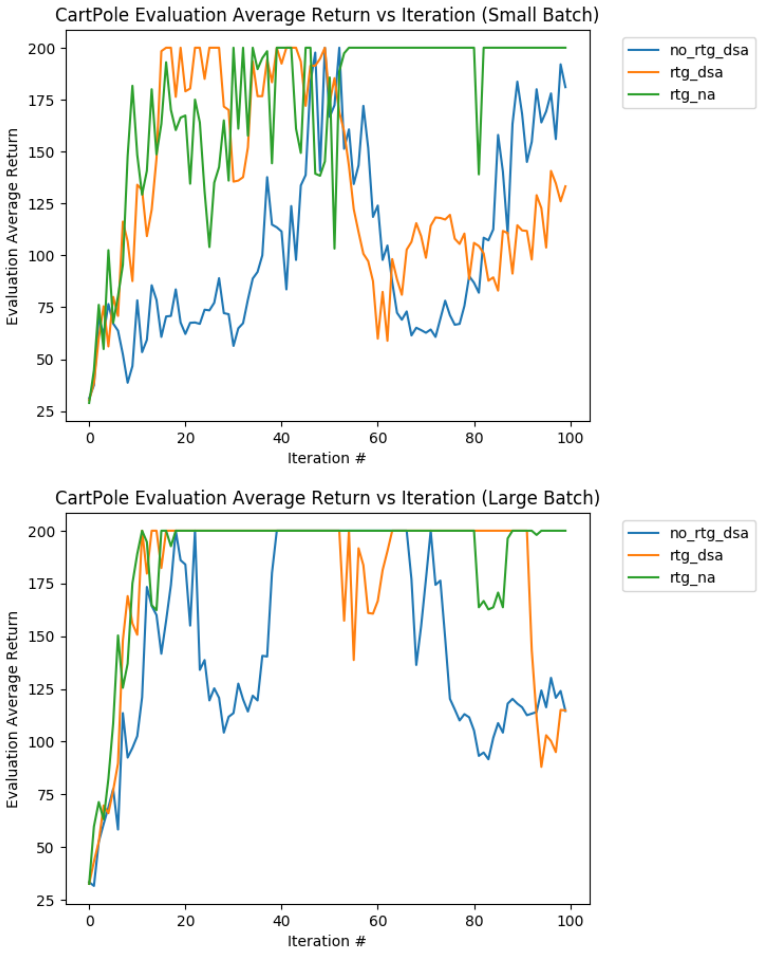

Figure 1: Experimental results for policy gradient ran on CartPole. no_rtg_dsa refers to trials ran without reward to go and unstandardized advantages. rtg_dsa refers to trials ran with reward to go and unstandardized advantages. rtg_na refers to trials ran with reward to go and normalized advantages. Small batch size was 1000, while large batch size was 5000.

Short analysis:

- Without advantage-standardization, it seems like using reward-to-go is better than trajectory-centric for value estimation. Reward-to-go did not outperform trajectory-centric for every single iteration, but overall it seems like using reward-to-go yields overall higher evaluation returns.
- Advantage standardization helped; when comparing between the run with rewards-to-go but no advantage-standardization and the run with rewards-to-go and advantage-standardization, the trial using advantage-standardization produced a higher evaluation return at almost all iterations.
- Changing the batch size from 1000 (small batch) to 5000 (large batch) seems to have the effect of "stabilizing" the results. Specifically, once an agent achieved the maximal reward of 200 at some iteration, the agent tended to stay at the reward of 200 for more iterations when using a large batch size. Larger batch size also seemed to marginally help the different agents reach the maximal reward of 200 in fewer iterations.

Command line configurations for this experiment:

```
python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
-dsa --exp_name q1_sb_no_rtg_dsa

python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
-rtg -dsa --exp_name q1_sb_rtg_dsa

python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
-rtg --exp_name q1_sb_rtg_na

python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 \
-dsa --exp_name q1_lb_no_rtg_dsa

python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 \
-rtg -dsa --exp_name q1_lb_rtg_dsa

python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 \
-rtg --exp_name q1_lb_rtg_na
```

2 Experiment 2

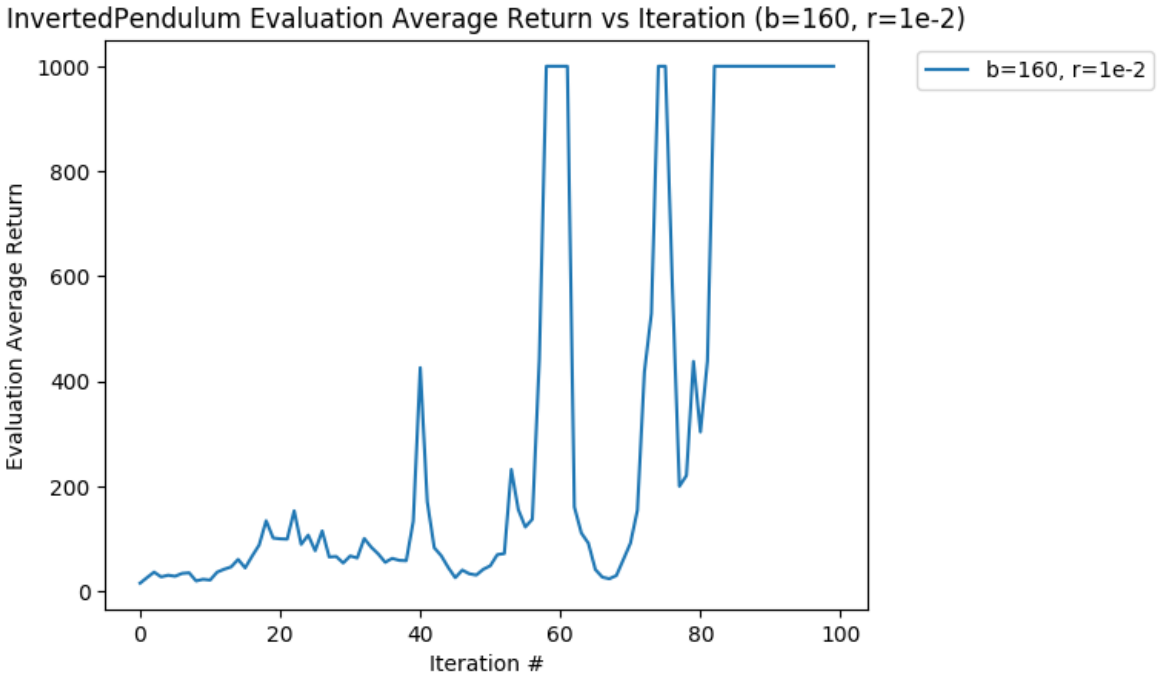

Figure 2: Experimental results for policy gradient ran on InvertedPendulum. Here, a batch size of 160 and a learning rate of 1e-2 was used. Within 100 iterations, the agent is able to achieve the best reward of 1000.

Command used to achieve this result:

```
python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v2 \
--ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 160 -lr 1e-2 -rtg \
--exp_name q2_b160_r1e-2
```

Many other experiments were performed on the way to find these parameters. The experiments performed used the following commands:

```
python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v2 \
--ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 1000 -lr 5e-3 -rtg \
--exp_name q2_b1000_r5e-3

python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v2 \
--ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 500 -lr 5e-3 -rtg \
--exp_name q2_b500_r5e-3

python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v2 \
--ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 500 -lr 1e-2 -rtg \
--exp_name q2_b500_r1e-2
```

```
python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v2 \
--ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 250 -lr 1e-2 -rtg \
--exp_name q2_b250_r1e-2

python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v2 \
--ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 250 -lr 7e-3 -rtg \
--exp_name q2_b250_r7e-3

python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v2 \
--ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 125 -lr 7e-3 -rtg \
--exp_name q2_b125_r7e-3

python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v2 \
--ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 150 -lr 8e-3 -rtg \
--exp_name q2_b150_r8e-3

python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v2 \
--ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 200 -lr 7e-3 -rtg \
--exp_name q2_b200_r7e-3

python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v2 \
--ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 200 -lr 8e-3 -rtg \
--exp_name q2_b200_r8e-3

python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v2 \
--ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 190 -lr 9e-3 -rtg \
--exp_name q2_b190_r9e-3

python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v2 \
--ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 175 -lr 1e-2 -rtg \
--exp_name q2_b175_r1e-2

python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v2 \
--ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 160 -lr 1e-2 -rtg \
--exp_name q2_b160_r1e-2

python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v2 \
--ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 130 -lr 1e-2 -rtg \
--exp_name q2_b130_r1e-2

python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v2 \
--ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 150 -lr 1e-2 -rtg \
--exp_name q2_b150_r1e-2
```

3 Experiment 3

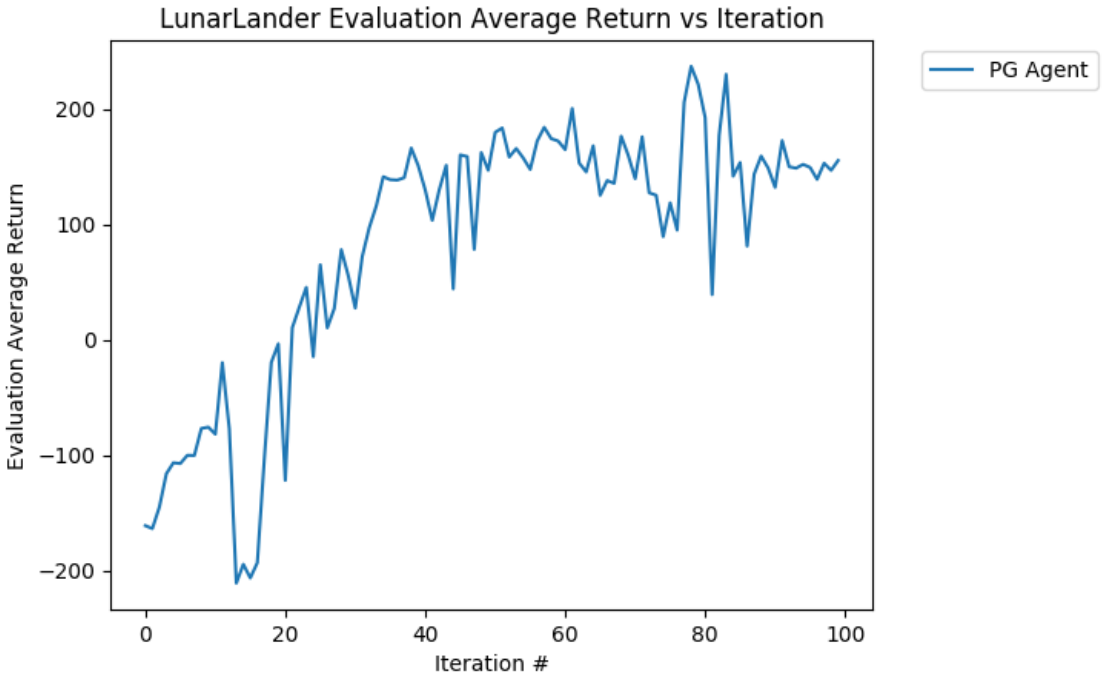

Figure 3: Experimental results for policy gradient ran on LunarLander. Unlike previous experiments, a neural net baseline was used as well.

Command used to achieve this result:

```
python cs285/scripts/run_hw2.py \
--env_name LunarLanderContinuous-v2 --ep_len 1000 \
--discount 0.99 -n 100 -l 2 -s 64 -b 40000 -lr 0.005 \
--reward_to_go --nn_baseline --exp_name q3_b40000_r0.005
```

4 Experiment 4

4.1 Hyperparameter Tuning

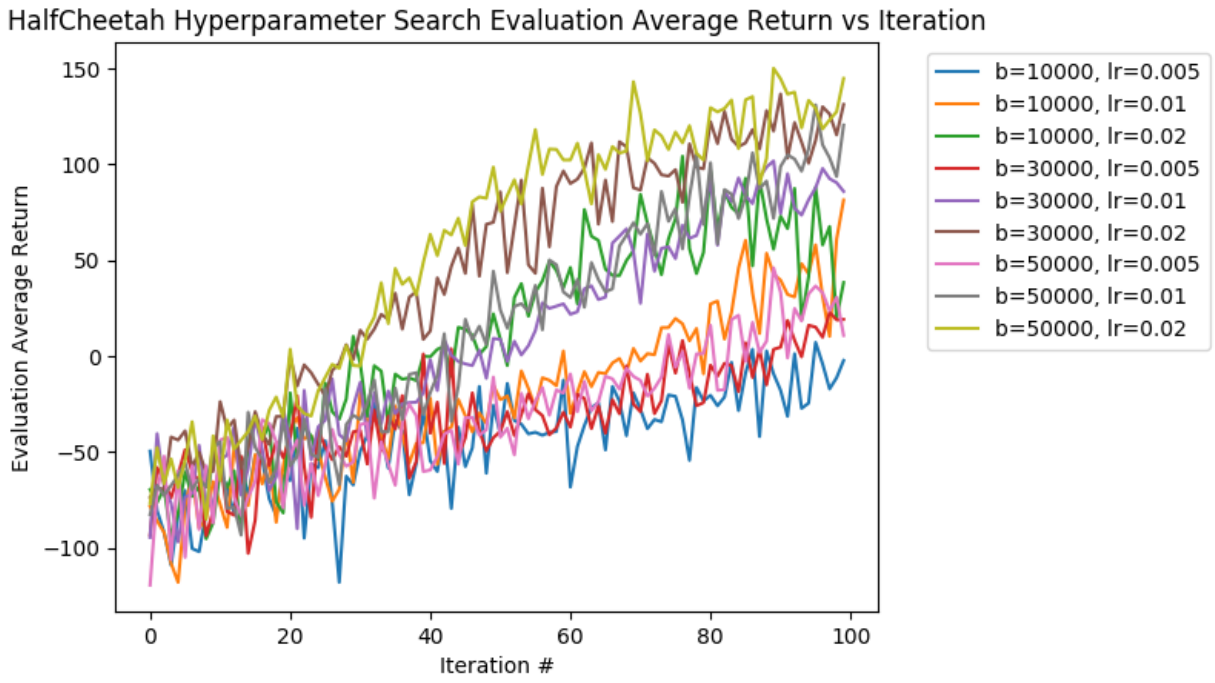

Figure 4: Experimental results for policy gradient with neural net baseline ran on HalfCheetah. These trials were ran for hyperparameter tuning, and the two hyperparameters tuned were the batch size and the learning rate. From this plot, we can see that as batch size increases, the evaluation average return increases as well. Also, we can see that as learning rate increases, the evaluation average return also increases as well.

Commands used to achieve this result:

```
python cs285/scripts/run_hw2.py --env_name HalfCheetah-v2 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.005 -rtg --nn_baseline \
--exp_name q4_search_b10000_lr0.005_rtg_nnbaseline

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v2 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.01 -rtg --nn_baseline \
--exp_name q4_search_b10000_lr0.01_rtg_nnbaseline

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v2 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.02 -rtg --nn_baseline \
--exp_name q4_search_b10000_lr0.02_rtg_nnbaseline

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v2 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.005 -rtg --nn_baseline \
```

```
--exp_name q4_search_b30000_lr0.005_rtg_nnbaseline

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v2 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.01 -rtg --nn_baseline \
--exp_name q4_search_b30000_lr0.01_rtg_nnbaseline

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v2 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.02 -rtg --nn_baseline \
--exp_name q4_search_b30000_lr0.02_rtg_nnbaseline

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v2 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.005 -rtg --nn_baseline \
--exp_name q4_search_b50000_lr0.005_rtg_nnbaseline

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v2 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.01 -rtg --nn_baseline \
--exp_name q4_search_b50000_lr0.01_rtg_nnbaseline

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v2 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 -rtg --nn_baseline \
--exp_name q4_search_b50000_lr0.02_rtg_nnbaseline
```

4.2 Optimal Parameter Experiments

Commands used to achieve this result:

```
python cs285/scripts/run_hw2.py --env_name HalfCheetah-v2 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 \
--exp_name q4_b50000_r0.02

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v2 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 -rtg \
--exp_name q4_b50000_r0.02_rtg

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v2 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 --nn_baseline \
--exp_name q4_b50000_r0.02_nnbaseline

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v2 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 -rtg --nn_baseline \
--exp_name q4_b50000_r0.02_rtg_nnbaseline
```

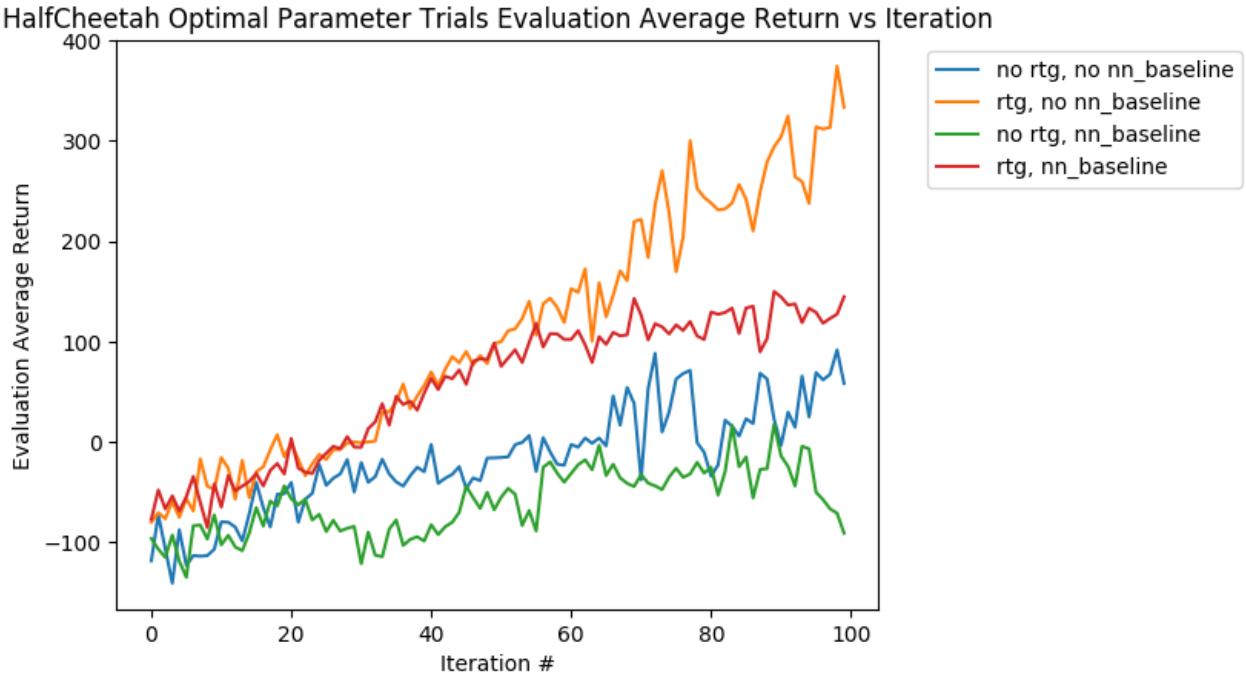

Figure 5: Experimental results for policy gradient with neural net baseline ran on HalfCheetah. These trials were ran with batch size of 50000 and learning rate of 0.02.

5 Experiment 5

Command used to achieve this result:

```
python cs285/scripts/run_hw2.py \
--env_name Hopper-v2 --ep_len 1000 \
--discount 0.99 -n 300 -l 2 -s 32 -b 2000 -lr 0.001 \
--reward_to_go --nn_baseline --action_noise_std 0.5 --gae_lambda 0 \
--exp_name q5_b2000_r0.001_lambda0

python cs285/scripts/run_hw2.py \
--env_name Hopper-v2 --ep_len 1000 \
--discount 0.99 -n 300 -l 2 -s 32 -b 2000 -lr 0.001 \
--reward_to_go --nn_baseline --action_noise_std 0.5 --gae_lambda 0.95 \
--exp_name q5_b2000_r0.001_lambda0.95

python cs285/scripts/run_hw2.py \
--env_name Hopper-v2 --ep_len 1000 \
--discount 0.99 -n 300 -l 2 -s 32 -b 2000 -lr 0.001 \
--reward_to_go --nn_baseline --action_noise_std 0.5 --gae_lambda 0.99 \
--exp_name q5_b2000_r0.001_lambda0.99

python cs285/scripts/run_hw2.py \
```

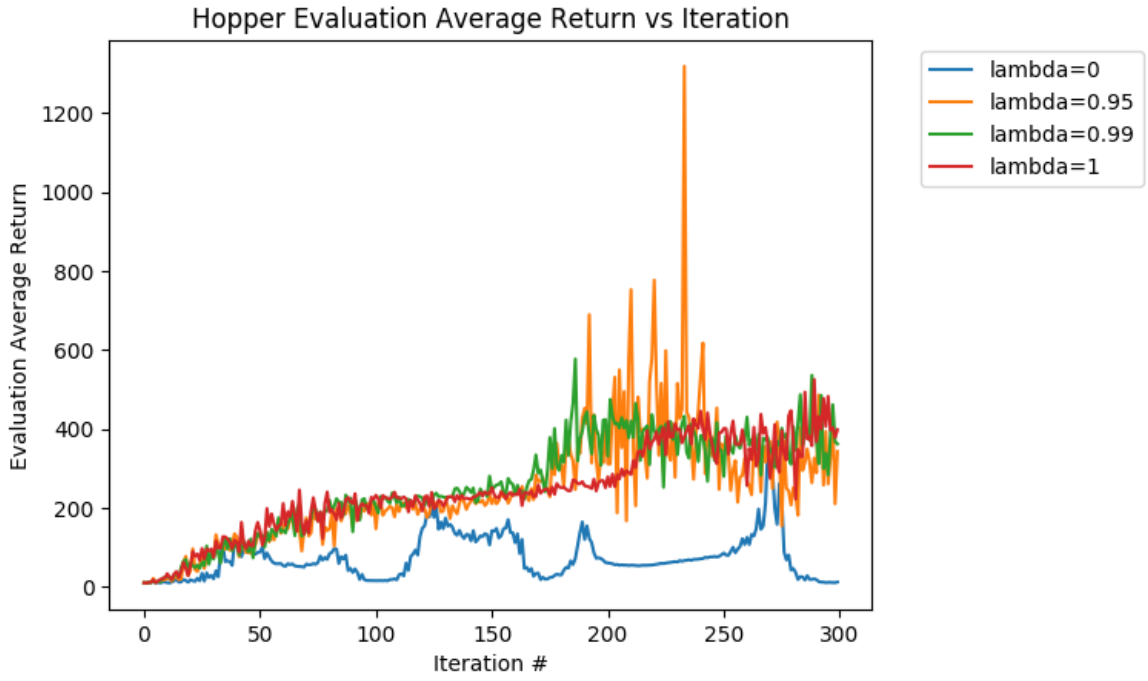

Figure 6: Experimental results for policy gradient and GAE-λ ran on Hopper. During these different trials, the value of λ was changed. The most optimal value of λ among the values tried was 0.95. In GAE-λ, the λ term functions as a control for the bias-variance trade off. When λ's value is too low, the model drastically reduces its variance, at the cost of high bias. On the other hand, when λ's value is too high, the model drastically reduces its bias, at the cost of high variance. As a result, the evaluation average return decreases when λ is too low and too high, and achieves a maximal value at some optimal value of λ in the middle.

```
--env_name Hopper-v2 --ep_len 1000 \
--discount 0.99 -n 300 -l 2 -s 32 -b 2000 -lr 0.001 \
--reward_to_go --nn_baseline --action_noise_std 0.5 --gae_lambda 1 \
--exp_name q5_b2000_r0.001_lambda1
```