Assignment 2: Policy Gradient

Andrew ID: `liweiy`
Collaborators: `azou`
NOTE: Please do **NOT** change the sizes of the answer blocks or plots.

5 Small-Scale Experiments

5.1 Experiment 1 (Cartpole) – [5 points total]

5.1.1 Configurations

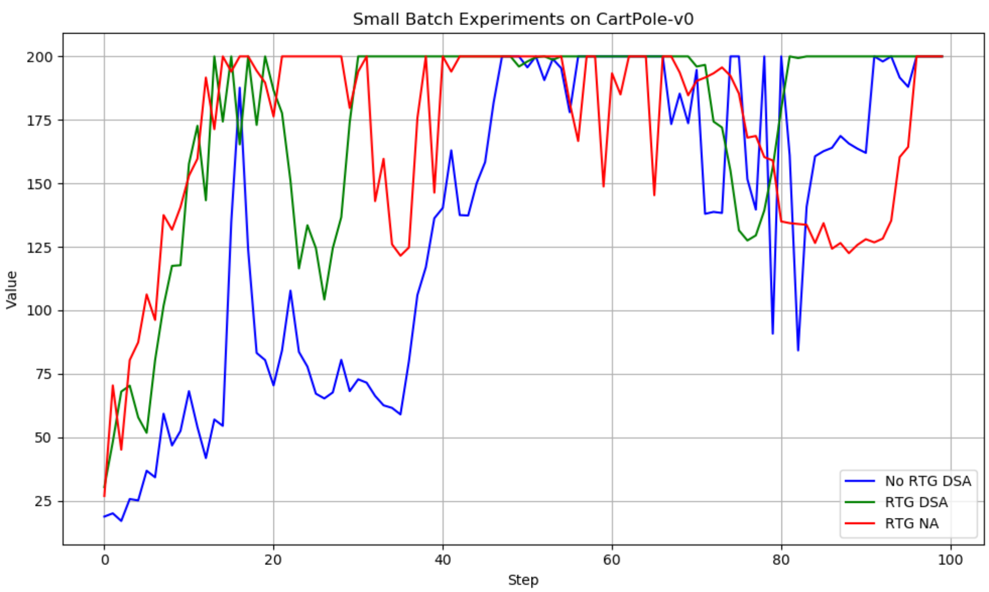
Q5.1.1

```
python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
    -dsa --exp_name q1_sb_no_rtg_dsa

python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
    -rtg -dsa --exp_name q1_sb_rtg_dsa

python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
    -rtg --exp_name q1_sb_rtg_na

python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 \
    -dsa --exp_name q1_lb_no_rtg_dsa

python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 \
    -rtg -dsa --exp_name q1_lb_rtg_dsa

python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 \
    -rtg --exp_name q1_lb_rtg_na
```

5.1.2 Plots

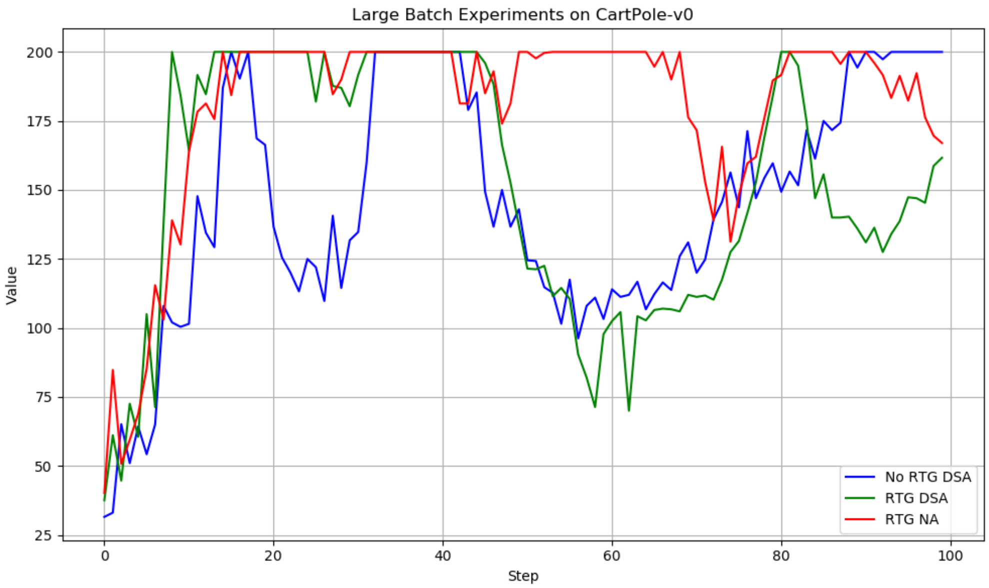
5.1.2.1 Small batch – [1 points]

Q5.1.2.1



5.1.2.2 Large batch – [1 points]

Q5.1.2.2



5.1.3 Analysis

5.1.3.1 Value estimator – [1 points]

Q5.1.3.1

The reward-to-go value estimator is better. After roughly 30 iterations, the RTG DSA can constantly reach 200 rewards in both small and large batches.

5.1.3.2 Advantage standardization – [1 points]

Q5.1.3.2

Standardized advantage can help the model learn faster and more stable when the batch size is large. We can see it from the RTG DSA and RTG NA in large batch experiments.

5.1.3.3 Batch size – [1 points]

Q5.1.3.3

Big batch has a noisy learning curve but learns faster. We can see that most curves hit 200 rewards multiple times before 20 steps, while with small batches, the curves only hit 200 a couple of times.

5.2 Experiment 2 (InvertedPendulum) – [4 points total]

5.2.1 Configurations – [1.5 points]

Q5.2.1

```
python rob831/scripts/run_hw2.py --env_name InvertedPendulum-v4 --ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 5000
↪  -lr 5e-3 -rtg --exp_name q2_b1000_r5e-3
```

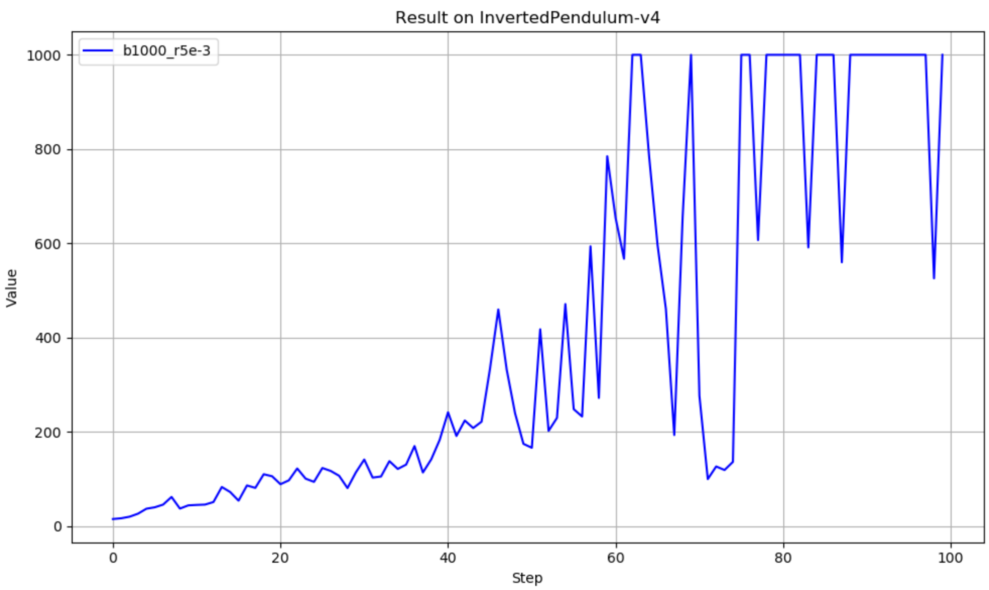
5.2.2 smallest b* and largest r* (same run) – [1.5 points]

Q5.2.2

batch size: 1000, learning rate: 5e-3.

5.2.3 Plot – [1 points]

Q5.2.3



7 More Complex Experiments

7.1 Experiment 3 (LunarLander) – [1 points total]

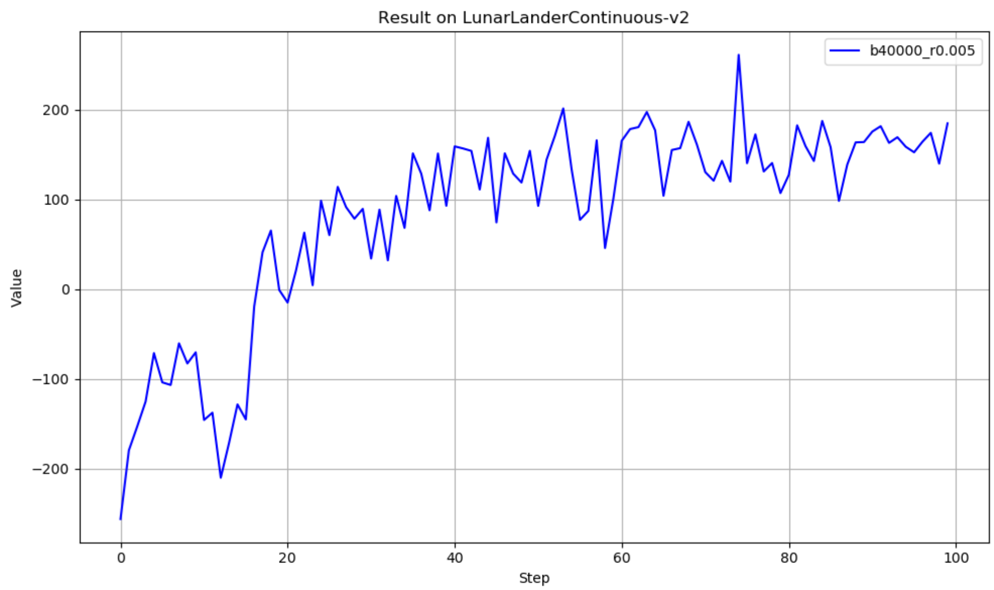
7.1.1 Configurations

Q7.1.1

```
python rob831/scripts/run_hw2.py \
    --env_name LunarLanderContinuous-v4 --ep_len 1000
    --discount 0.99 -n 100 -l 2 -s 64 -b 10000 -lr 0.005 \
    --reward_to_go --nn_baseline --exp_name q3_b10000_r0.005
```

7.1.2 Plot – [1 points]

Q7.1.2



7.2 Experiment 4 (HalfCheetah) – [1 points]

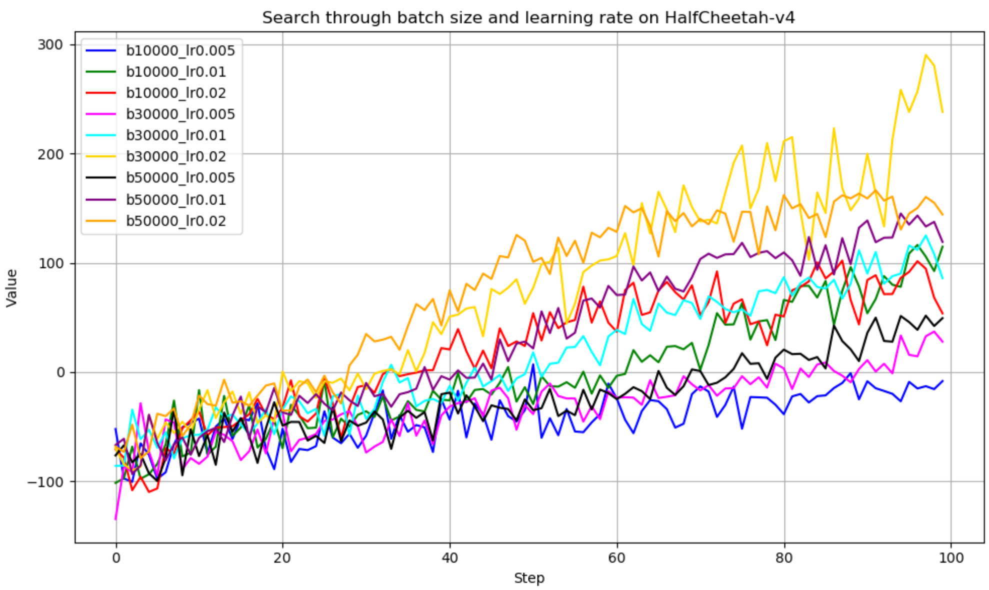
7.2.1 Configurations

Q7.2.1

```
python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.02 \
    --exp_name q4_search_b10000_lr0.02
python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.02 -rtg \
    --exp_name q4_search_b10000_lr0.02_rtg
python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.02 --nn_baseline \
    --exp_name q4_search_b10000_lr0.02_nnbaseline
python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --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
```

7.2.2 Plot – [1 points]

Q7.2.2



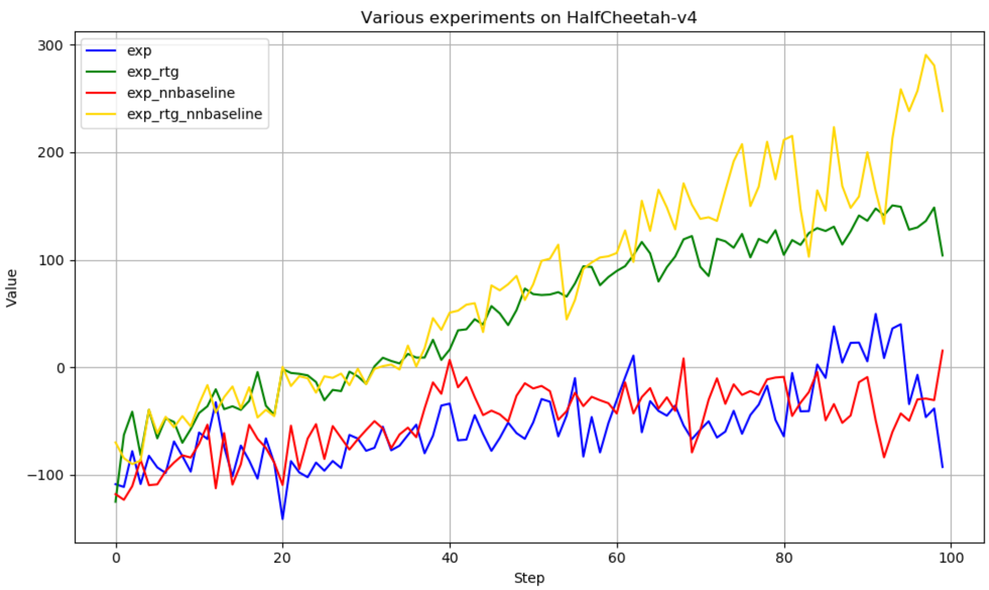
7.2.3 (bonus) Optimal b* and r* – [0.5 points]

Q7.2.3

Optimal b* = 30000, optimal r* = 0.02

7.2.4 (bonus) Plot – [0.5 points]

Q7.2.4



7.2.5 (bonus) Describe how b* and r* affect task performance – [0.5 points]

Q7.2.5

If the batch size is too small, the model would have a hard time learning. We can see generally experiments with b10000 do not pass 100 too much. The large batch size can make the learning more efficient, but it will also make the learning noisy, sometimes it makes us miss the local maxima, as we can see from the result from b50000. It is better to use a larger learning rate with a larger batch size to learn faster and update the model promptly. The b* is 30000 with a learning rate of 0.02.

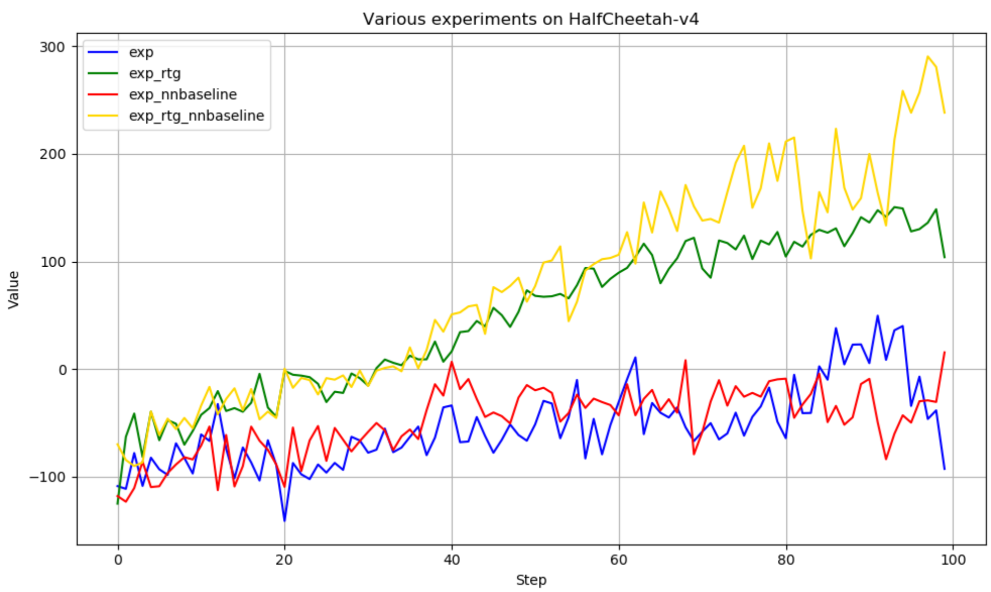
7.2.6 (bonus) Configurations with optimal b* and r* – [0.5 points]

Q7.2.6

```
python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.02 \
    --exp_name q4_b30000_r0.02

python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.02 -rtg \
    --exp_name q4_b30000_r0.02_rtg

python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.02 --nn_baseline \
    --exp_name q4_b30000_r0.02_nnbaseline

python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.02 -rtg --nn_baseline \
    --exp_name q4_b30000_r0.02_rtg_nnbaseline
```

7.2.7 (bonus) Plot for four runs with optimal b* and r* – [0.5 points]

Q7.2.7



8 Implementing Generalized Advantage Estimation

8.1 Experiment 5 (Hopper) – [4 points]

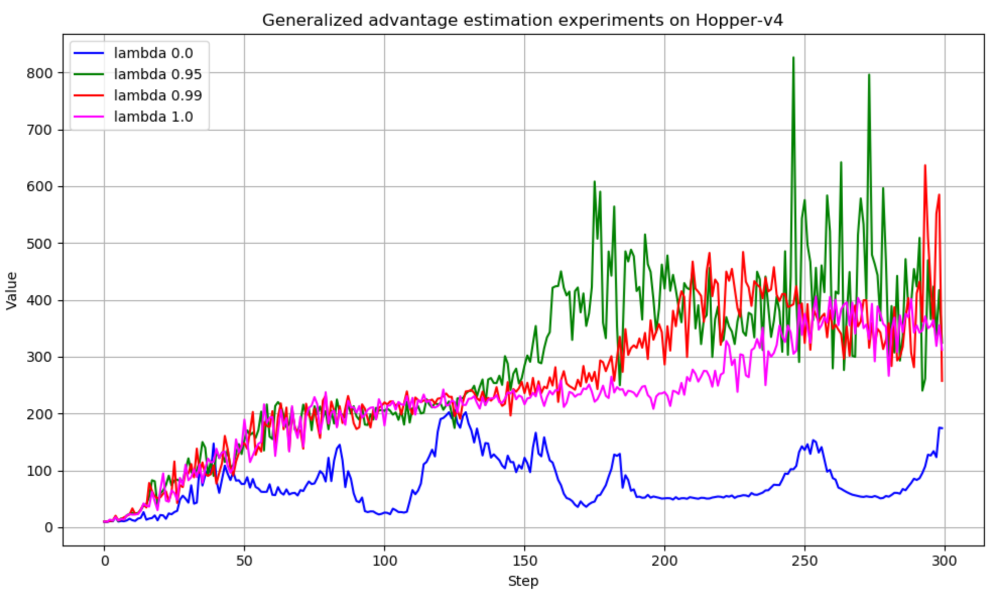
8.1.1 Configurations

Q8.1.1

```
# λ ∈ [0, 0.95, 0.99, 1]
python rob831/scripts/run_hw2.py \
    --env_name Hopper-v4 --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 <λ> \
    --exp_name q5_b2000_r0.001_lambda<λ>
```

8.1.2 Plot – [2 points]

Q8.1.2



8.1.3 Describe how λ affects task performance – [2 points]

Q8.1.3

If the lambda is too low, the model would biased more toward immediate advantage, thus causing the model to have a hard time learning. If the lambda is large, it will calculate the advantage more globally to the entire trajectory. However, this might introduce more variance. Thus it is better to set a large value but not too large.

9 More Bonus!

9.1 Parallelization – [1.5 points]

Q9.1

With Parallelization:

```
python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 -rtg --exp_name q1_parallel_sb_rtg_na
```

Training time: 66.87 seconds

Without Parallelization:

```
python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 -rtg --exp_name q1_no_parallel_sb_rtg_na
```

Training time: 70.10 seconds

Difference in training time: 3.23, Percentage difference: 4.6%

9.2 Multiple gradient steps – [1 points]

Q9.1