# Assignment 2: Policy Gradient

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

## 5 Small-Scale Experiments

### 5.1 Experiment 1 (Cartpole) – [25 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 – [5 points]

Q5.1.2.1

car_sb.png

#### 5.1.2.2 Large batch – [5 points]

Q5.1.2.2


### 5.1.3 Analysis

#### 5.1.3.1 Value estimator – [5 points]

Q5.1.3.1

The reward-to-go performs better.

#### 5.1.3.2 Advantage standardization – [5 points]

Q5.1.3.2

The advantage standardization helps the task to faster converge to optimal.

#### 5.1.3.3 Batch size – [5 points]

Q5.1.3.3

Small batch size helps tasks to faster converge to optimal.

## 5.2 Experiment 2 (InvertedPendulum) – [15 points total]

### 5.2.1 Configurations – [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 1000 -lr 5e-2 -rtg \
    --exp_name q2_b1000_r5e_2
```

### 5.2.2 smallest b* and largest r* (same run) – [5 points]

Q5.2.2

I found to get to optimum in less than 100 iterations, my smallest batch size is 1000 while my largest learning rate is 5e-2.

#### 5.2.3 Plot – [5 points]

Q5.2.3


# 7 More Complex Experiments

## 7.1 Experiment 3 (LunarLander) – [10 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 – [10 points]

Q7.1.2

land.png

### 7.2 Experiment 4 (HalfCheetah) – [30 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 – [10 points]

Q7.2.2

cheetah_basic.png

#### 7.2.3 (Optional) Optimal b* and r* – [3 points]

Q7.2.3

Optimal b*: 50000
Optimal r*: 0.02

### 7.2.4 (Optional) Plot – [10 points]

Q7.2.4

cheetah_exp4.png

### 7.2.5 (Optional) Describe how b* and r* affect task performance – [7 points]

Q7.2.5

Generally, as the batch size b* increases, the train cost time and final average return also increase. When the learning rate r* decreases, the task will take longer to converge or fail to achieve converge during 100 iterations.

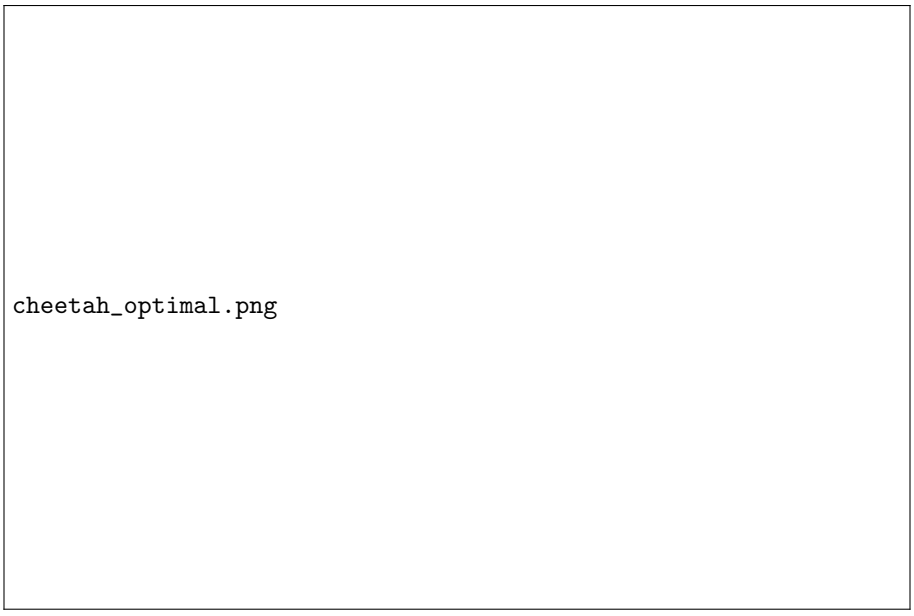
### 7.2.6 (Optional) Configurations with optimal b* and r* – [3 points]

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 50000 -lr 0.02 \
    --exp_name q4_b50000_re_2

python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 -rtg \
    --exp_name q4_b50000_re_2_rtg

python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 --nn_baseline \
    --exp_name q4_b50000_re_2_nnbaseline

python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 -rtg --nn_baseline \
    --exp_name q4_b50000_re_2_rtg_nnbaseline
```

### 7.2.7 (Optional) Plot for four runs with optimal b* and r* – [7 points]

Q7.2.7




# 8 Implementing Generalized Advantage Estimation

## 8.1 Experiment 5 (Hopper) – [20 points]

### 8.1.1 Configurations

Q8.1.1

```
# [0,0.95,0.99,1]\lambda \in [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 – [13 points]

Q8.1.2

hopper.png

### 8.1.3 Describe how $\lambda$ affects task performance – [7 points]

Q8.1.3

The task performs worst when lambda = 0. From lambda = 0.95 - 1, the task performs best when lambda = 0.95 and get worse when lambda increases.

# 9 Bonus! (optional)

## 9.1 Parallelization – [15 points]

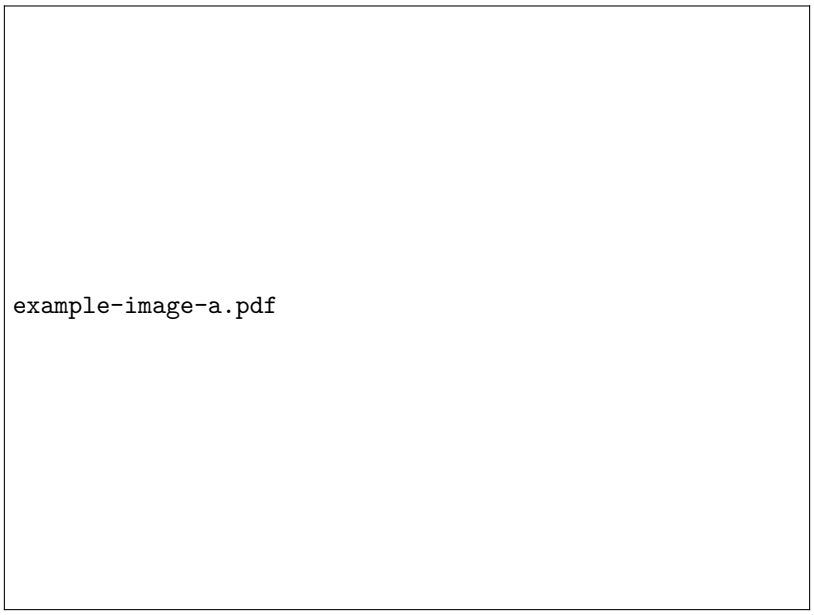
Q9.1

Difference in training time:

```
python rob831/scripts/run_hw2.py \
```

## 9.2 Multiple gradient steps – [5 points]

Q9.1




```
python rob831/scripts/run_hw2.py \
```