

## Problem 4. CartPole



- The reward-to-go gradient estimator has better performance without advantage-centering.
- The advantage-centering does not help learn but helping reduce the variance.
- The batch size has a clear impact in learning performance, i.e. helping learn faster but not improving performance.

### **Command lines:**

```
python train_pg_f18.py CartPole-v0 -n 100 -b 1000 -e 3 -dna --exp_name sb_no_rtg_dna
python train_pg_f18.py CartPole-v0 -n 100 -b 1000 -e 3 -rtg -dna --exp_name sb_rtg_dna
python train_pg_f18.py CartPole-v0 -n 100 -b 1000 -e 3 -rtg --exp_name sb_rtg_na
python train_pg_f18.py CartPole-v0 -n 100 -b 5000 -e 3 -dna --exp_name lb_no_rtg_dna
python train_pg_f18.py CartPole-v0 -n 100 -b 5000 -e 3 -rtg -dna --exp_name lb_rtg_dna
python train_pg_f18.py CartPole-v0 -n 100 -b 5000 -e 3 -rtg --exp_name lb_rtg_na
```

```
python plot.py data/sb_no_rtg_dna_CartPole-v0_19-09-2018_21-26-20/ data/sb_rtg_dna_CartPole-
v0_19-09-2018_21-46-46/ data/sb_rtg_na_CartPole-v0_19-09-2018_21-49-55/
```

```
python plot.py data/lb_no_rtg_dna_CartPole-v0_19-09-2018_22-05-12/ data/lb_rtg_dna_CartPole-
v0_19-09-2018_22-19-22/ data/lb_rtg_na_CartPole-v0_19-09-2018_22-39-08/
```

## Problem 5. Inverted Pendulum

$b^* = 2000$ ,  $lr^* = 0.03$

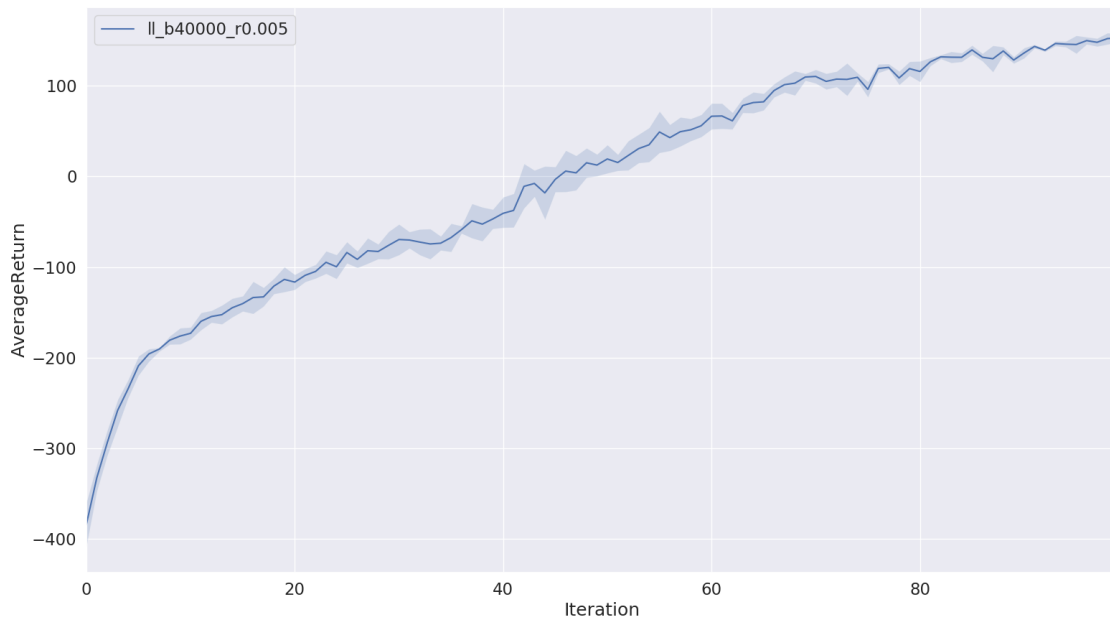


### Command lines:

```
python train_pg_f18.py InvertedPendulum-v2 -ep 1000 --discount 0.9 -n 100 -e 3 -l 2 -s 64 -b 2000 -lr 0.03 -rtg --exp_name hc_b2000_r0.03
```

```
python plot.py data/ip_b2000_r0.03_InvertedPendulum-v2_20-09-2018_00-21-21/
```

## Problem 7. Lunar Lander



### **Command line:**

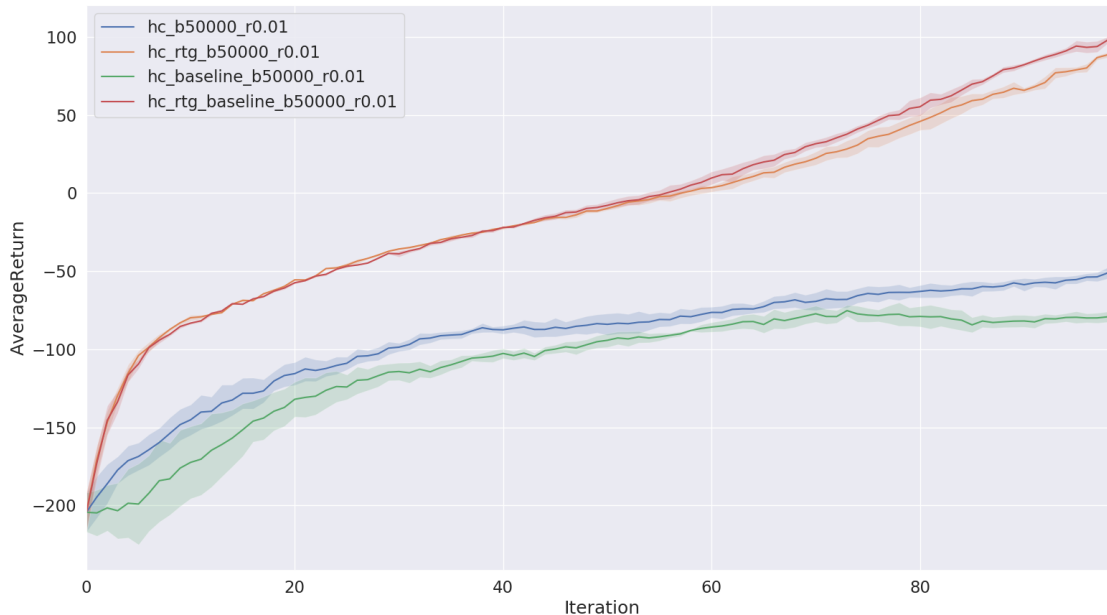
```
python train_pg_f18.py LunarLanderContinuous-v2 -ep 1000 --discount 0.99 -n 100 -e 3 -l 2 -s 64 -b 40000 -lr 0.005 -rtg --nn_baseline --exp_name ll_b40000_r0.005
```

```
python plot.py data/ll_b40000_r0.005_LunarLanderContinuous-v2_20-09-2018_01-32-48/
```

## Problem 8. Half Cheetah

- The batch size improves the learning performance, i.e. helping learn better, but increasing the learning time.
- The learning rate help learn faster, thus helping learn more given the same time/number of iteration, but if the large learning rate harms the learning performance.

$b^* = 50000$ ,  $lr^* = 0.01$ . Old-version homework#2 → average reward is around 100.



### Command lines:

```
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.01  
--exp_name hc_b50000_r0.01
```

```
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.01  
-rtg --exp_name hc_rtg_b50000_r0.01
```

```
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.01  
--nn_baseline --exp_name hc_baseline_b50000_r0.01
```

```
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.01  
-rtg --nn_baseline --exp_name hc_rtg_baseline_b50000_r0.01
```

```
python plot.py data/hc_b50000_r0.01_HalfCheetah-v2_20-09-2018_22-24-  
53/data/hc_rtg_b50000_r0.01_HalfCheetah-v2_20-09-2018_21-12-  
18/data/hc_baseline_b50000_r0.01_HalfCheetah-v2_20-09-2018_20-01-  
17/data/hc_rtg_baseline_b50000_r0.01_HalfCheetah-v2_20-09-2018_18-51-17/
```