CS294-112 HW2

Yujia Luo

September 2018

1 State-dependent baseline

Let $x = \nabla_\theta \log p_\theta(a_t|s_t)\,(b(s_t))$

1.1

$\mathrm{E}_{\tau\sim p_\theta(\tau)}[x]$
$= E_{(s_t,a_t)\sim p_\theta(s_t,a_t)} E_{\tau/(s_t,a_t)\ \ p_\theta(\tau/(s_t,a_t))}[x]$
$= E_{(s_t,a_t)\sim p_\theta(s_t,a_t)}[x]$
$= E_{s_t\sim p_\theta(s_t)} E_{a_t\sim p_\theta(a_t|s_t)}[x]$
$= E_{s_t\sim p_\theta(s_t)} E_{a_t\sim p_\theta(a_t|s_t)}[\nabla_\theta \log p_\theta(a_t|s_t)\,(b(s_t))]$
$= E_{s_t\sim p_\theta(s_t)} b(s_t) E_{a_t\sim p_\theta(a_t|s_t)}[\nabla_\theta \log p_\theta(a_t|s_t)]$
$\int_{s_t} p_\theta(s_t) b(s_t) \int_{a_t} p_\theta(a_t|s_t) \nabla_\theta \log p_\theta(a_t|s_t) da_t ds_t$
$\int_{s_t} p_\theta(s_t) b(s_t) \nabla_\theta \int_{a_t} p_\theta(a_t|s_t) da_t ds_t$
$\int_{s_t} p_\theta(s_t) b(s_t) \nabla_\theta 1 da_t ds_t$
$= 0$

1.2

According to Markov property, policy should only depend on current state:

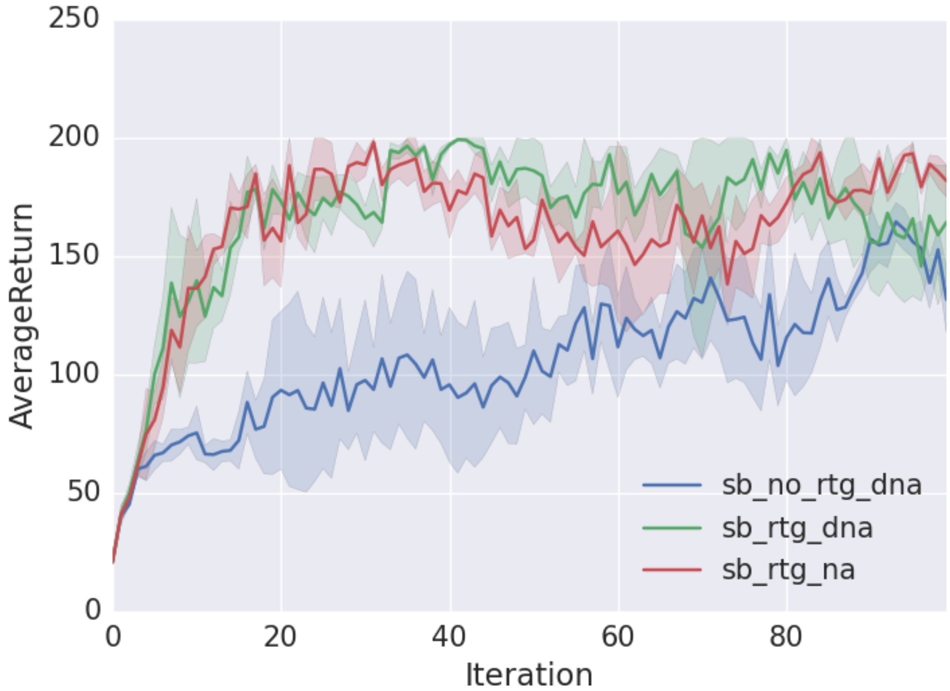
$$\pi_\theta(a_t|s_t, a_t, s_{t-1}, a_{t-1}...s_1, a_1) = \pi_\theta(a_t|s_t)$$

1.3

$\mathrm{E}_{\tau\sim p_\theta(\tau)}[x]$
$E_{s_{0:t},a_{0:t-1}}\left[E_{s_{t+1:T},a_{t:T-1}}[x]\right]$
$E_{s_{0:t},a_{0:t-1}}\left[E_{s_{t+1:T},a_{t:T-1}}[\nabla_\theta \log p_\theta(a_t|s_t)\,(b(s_t))]\right]$
$E_{s_{0:t},a_{0:t-1}}\left[b(s_t) E_{s_{t+1:T},a_{t:T-1}}[\nabla_\theta \log p_\theta(a_t|s_t)]\right]$
$E_{s_{0:t},a_{0:t-1}}[b(s_t) E_{a_t}[\nabla_\theta \log p_\theta(a_t|s_t)]]$
$E_{s_{0:t},a_{0:t-1}}[b(s_t)\cdot 0]]$
$= 0$

2 CartPole-v0

2.1 Small Batch



Commamds:

`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`

2.2 Big Batch

Commamds:

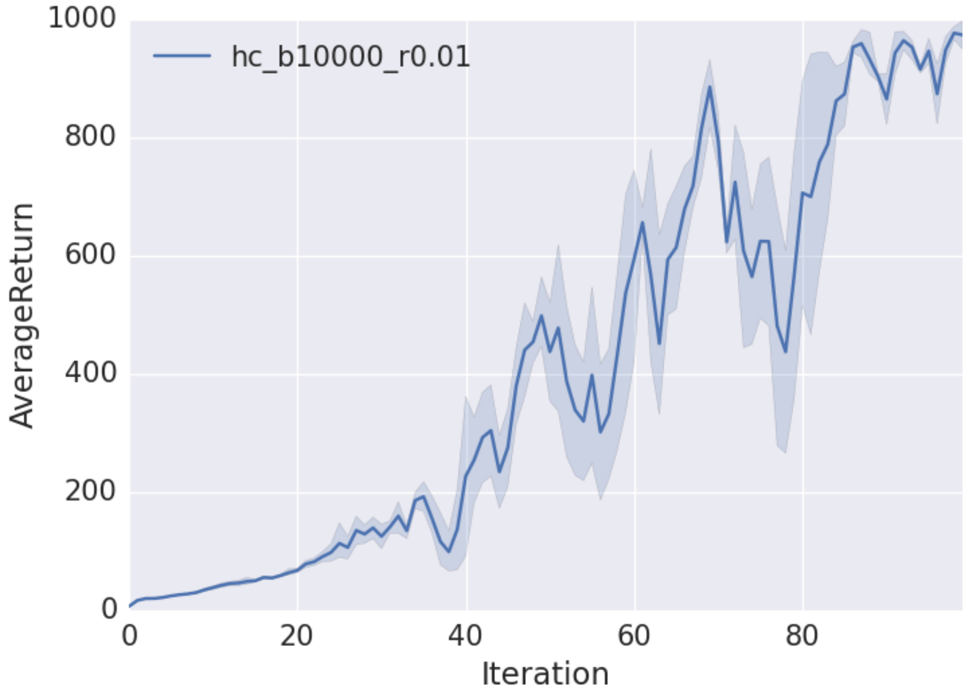
`python train_pg_f18.py CartPole-v0 -n 100 -b 5000 -e 3 -dna --exp_name lb_no_rtg_dna`

`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`

Comments:
Reward-to-go gradient estimator converges faster. Advantage centering does not have an obvious effect on the learning curve. In terms of batch size, if using reward-to-go, bigger batch size results in more stable learning

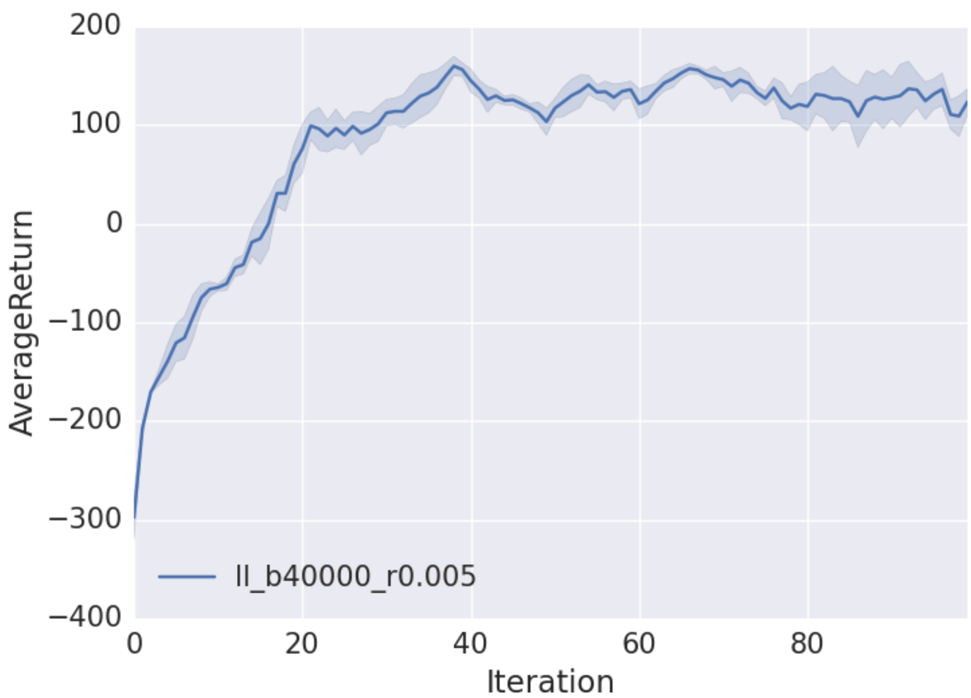
3 InvertedPendulum



The smallest batch size is 10000, and biggest learning rate is 0.01
Commamds:

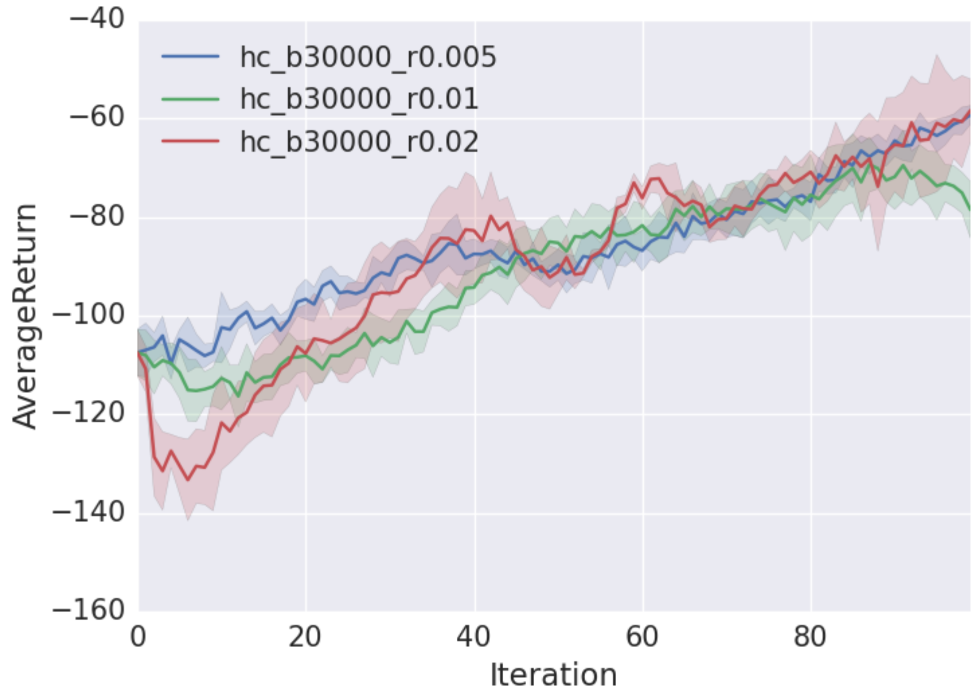
```
python train_pg_f18.py InvertedPendulum-v2 -ep 1000 --discount 0.9 -n 100 -e 3 -l 2 \
    -s 64 -b 1000 -lr 0.001 -rtg --exp_name hc_b1000_r0.001

python train_pg_f18.py InvertedPendulum-v2 -ep 1000 --discount 0.9 -n 100 -e 3 -l 2 \
    -s 64 -b 5000 -lr 0.001 -rtg --exp_name hc_b5000_r0.001

python train_pg_f18.py InvertedPendulum-v2 -ep 1000 --discount 0.9 -n 100 -e 3 -l 2 \
    -s 64 -b 10000 -lr 0.01 -rtg --exp_name hc_b10000_r0.01

python train_pg_f18.py InvertedPendulum-v2 -ep 1000 --discount 0.9 -n 100 -e 3 -l 2 \
    -s 64 -b 5000 -lr 0.01 -rtg --exp_name hc_b5000_r0.01

python train_pg_f18.py InvertedPendulum-v2 -ep 1000 --discount 0.9 -n 100 -e 3 -l 2 \
    -s 64 -b 5000 -lr 0.05 -rtg --exp_name hc_b5000_r0.05
```

4 LunarLander



5 HalfCheetah

5.1 Effect of learning rate

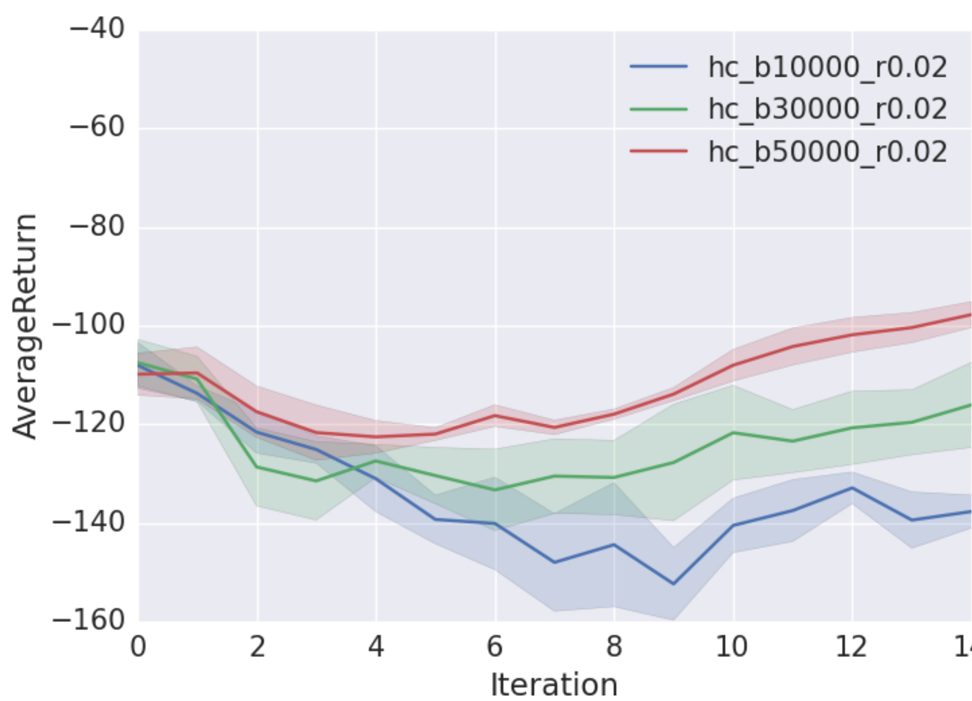


Commamds:

```
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 \
    -s 32 -b 30000 -lr 0.005 --exp_name hc_b30000_r0.005
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 \
    -s 32 -b 30000 -lr 0.01 --exp_name hc_b30000_r0.01

python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 \
    -s 32 -b 30000 -lr 0.02 --exp_name hc_b30000_r0.02
```

Comments:
Without using reward-to-go or baseline, increasing the learning rate does not really make a difference.

5.2 Effect of batch size



Commamds:

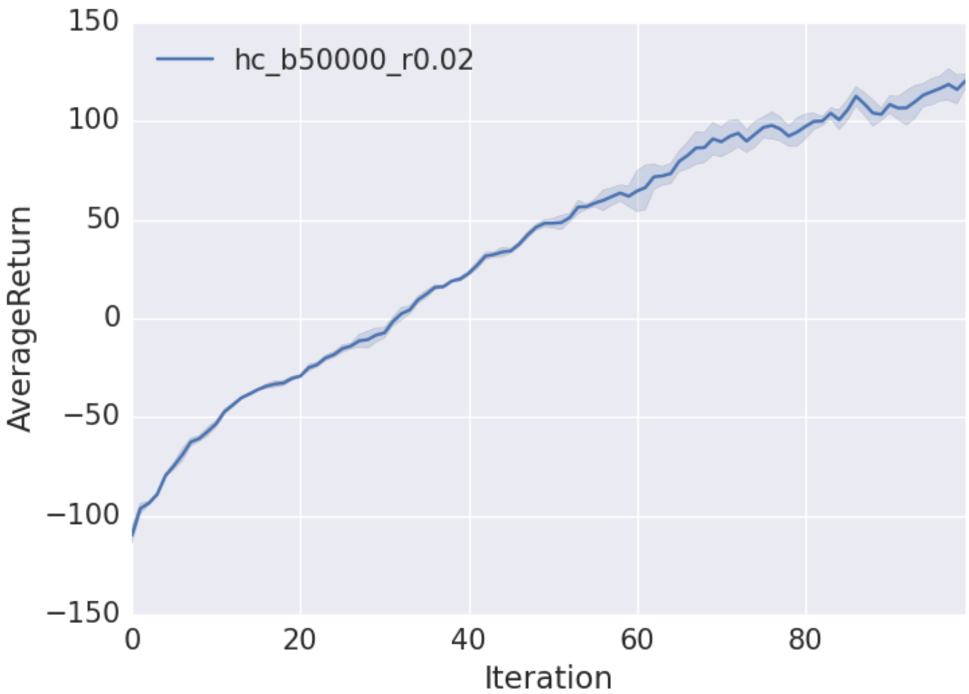
```
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 \
    -b 30000 -lr 0.02 --exp_name hc_b30000_r0.02

python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 \
    -b 10000 -lr 0.02 --exp_name hc_b10000_r0.02
```

```
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 \
    -b 10000 -lr 0.02 --exp_name hc_b50000_r0.02
```

Comments:
Without using reward-to-go or baseline, increasing the batch size makes the learning more stable, but does not make it converge.

5.3 Effect of reward-to-go and baseline

I plan to use 50000 as batch size and 0.02 as learning rate.



Commamds:

```
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 \
    -b 50000 -lr 0.02 -rtg --exp_name hc_b50000_r0.02

python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 \
    -b 50000 -lr 0.02 --nn_baseline --exp_name hc_b50000_r0.02

python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 \
    -b 50000 -lr 0.02 -rtg --nn_baseline --exp_name hc_b50000_r0.02
```

Comments:
Reward-to-go significantly improve the learning. However, I did not plot the two commands with the baseline because my implementation of baseline doesn't seem to have any effect - the learning curve is the same as the case without baseline.