CS294: Deep Reinforcement Learning

Homework 2 - Policy Gradients

Siyao Jia, SID: 3032531126

Problem 1. State-dependent baseline

(a)

$$
\begin{aligned}
&\sum_{t=1}^{T} E_{\tau\sim p_\theta(\tau)}[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)] \\
&= \sum_{t=1}^{T} E_{s_t,a_t\sim p_\theta(s_t,a_t)}[E_{p_\theta(\tau/s_t,a_t|s_t,a_t)}[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)]] \\
&= \sum_{t=1}^{T} E_{s_t,a_t\sim p_\theta(s_t,a_t)}[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)] \\
&= \sum_{t=1}^{T} b(s_t)E_{s_t,a_t\sim p_\theta(s_t,a_t)}[\nabla_\theta \log \pi_\theta(a_t|s_t)] \\
&= \sum_{t=1}^{T} b(s_t)E_{s_t}[E_{a_t\sim\pi_\theta(a_t|s_t)}[\nabla_\theta \log \pi_\theta(a_t|s_t)]] \\
&= \sum_{t=1}^{T} b(s_t)E_{s_t}[\int \pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t)da_t] \\
&= \sum_{t=1}^{T} b(s_t)E_{s_t}[\int \pi_\theta(a_t|s_t)\frac{\nabla_\theta\pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)}da_t] \\
&= \sum_{t=1}^{T} b(s_t)E_{s_t}[\nabla_\theta \int \pi_\theta(a_t|s_t)da_t] \\
&= 0
\end{aligned}
\tag{1}
$$

(b)

(a) Because in Markov process, the current state s_{t+1} only depends on previous s_t and a_t, and is independent of s_{t-1} or a_{t-1}. Therefore conditioning on $(s_1, a_1, ..., a_{t^*-1}, s_{t^*})$ is equivalent to conditioning only on s_{t^*}.

(b)

$$
\begin{aligned}
&\sum_{t=1}^{T} E_{\tau \sim p_\theta(\tau)}[\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t)] \\
&= \sum_{t=1}^{T} E_{p_\theta(s_{1:t}, a_{1:t-1})}[E_{p_\theta(s_{t+1:T}, a_{t:T}|s_{1:t}, a_{1:t-1})}[\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t)]] \\
&= \sum_{t=1}^{T} E_{p_\theta(s_{1:t}, a_{1:t-1})}[E_{p_\theta(s_{t+1:T}, a_{t:T}|s_t)}[\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t)]] \\
&= \sum_{t=1}^{T} E_{p_\theta(s_{1:t}, a_{1:t-1})}[b(s_t) E_{p_\theta(s_{t+1:T}, a_{t:T}|s_t)}[\nabla_\theta \log \pi_\theta(a_t|s_t)]] \\
&= \sum_{t=1}^{T} E_{p_\theta(s_{1:t}, a_{1:t-1})}[b(s_t) E_{\pi_\theta(a_t|s_t)}[\nabla_\theta \log \pi_\theta(a_t|s_t)]] \\
&= 0
\end{aligned}
\tag{2}
$$

Problem 2. Neural networks
See details in code.

Problem 3. Policy Gradient
See details in code.

Problem 4. CartPole

1. Figure 1 compares the learning curves for the experiments prefixed with sb_ (The small batch experiments). Figure 2 compare the learning curves for the experiments prefixed with lb_. (The large batch experiments.)

2. - The gradient estimator using reward-to-go has better performance without advantange-centering.
   - Yes, advantage centering helps slightly.
   - Yes, batch size makes a huge impact by stabilizing the return value.

3. The command line configurations I used:
   "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_no_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 sb_rtg_dna"
   "python train_pg_f18.py CartPole-v0 -n 100 -b 5000 -e 3 -rtg –exp_name sb_no_rtg_na"

Figure 1: Learning Curves for Small Batch Experiments

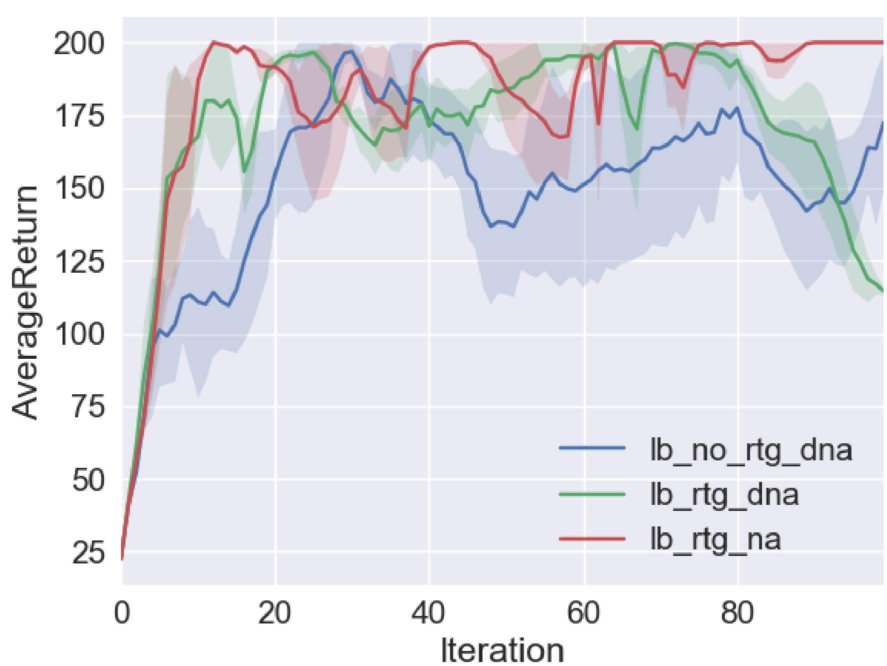

Figure 2: Learning Curves for Large Batch Experiments

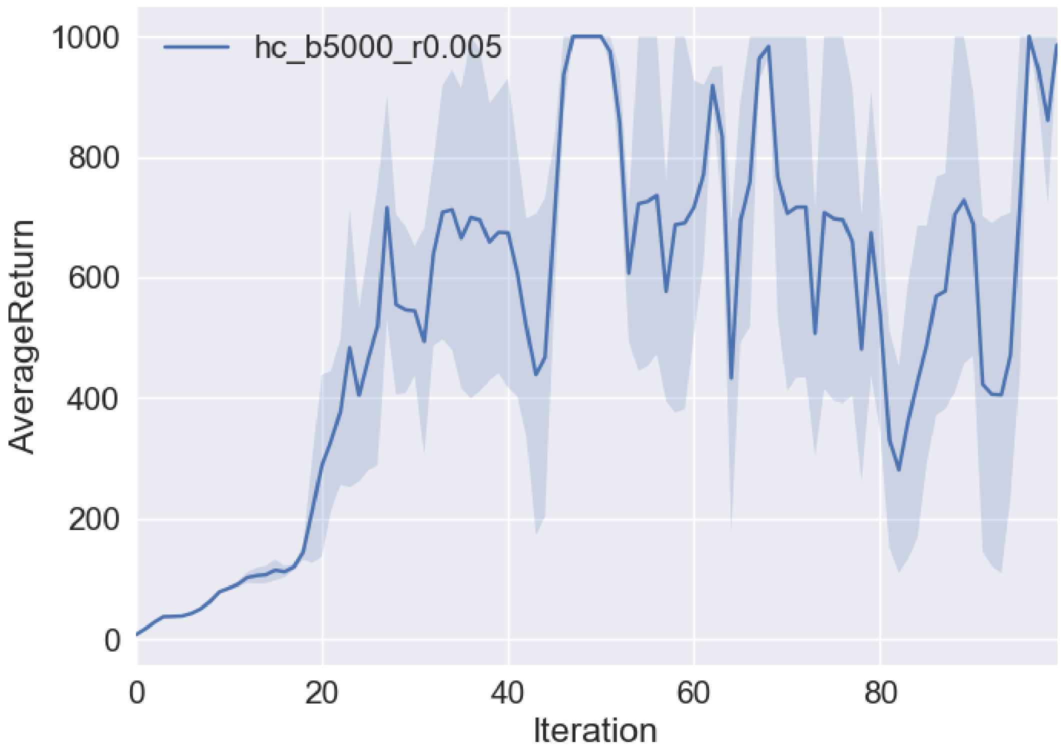

Figure 3: Learning Curve for InvertedPendulum

Problem 5. IvertedPendulum

1. I choose batch size b of 5000 and learning rate lr of 0.005. The learning curve is showed in Figure 3.
2. The command line configurations I used:
   "python train_pg_f18.py InvertedPendulum-v2 -ep 1000 –discount 0.9 -n 100 -e 3 -l 2 -s 64 -b 5000 -lr 0.005 -rtg –exp_name hc_b5000_r0.005"

Problem 6. Neural network baseline
See details in code.

Problem 7. LunarLander
The learning curve from the following command is plotted in Figure 4
"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"

Problem 8. HalfCheetah

1. Larger batch size reduces variance and stabilizes the performance. Lower learning rate takes longer time, but are more likely to converge. Vice versa for high learning rate.
2. I choose batch size b of 50000 and learning rate lr of 0.02. The learning curves of the

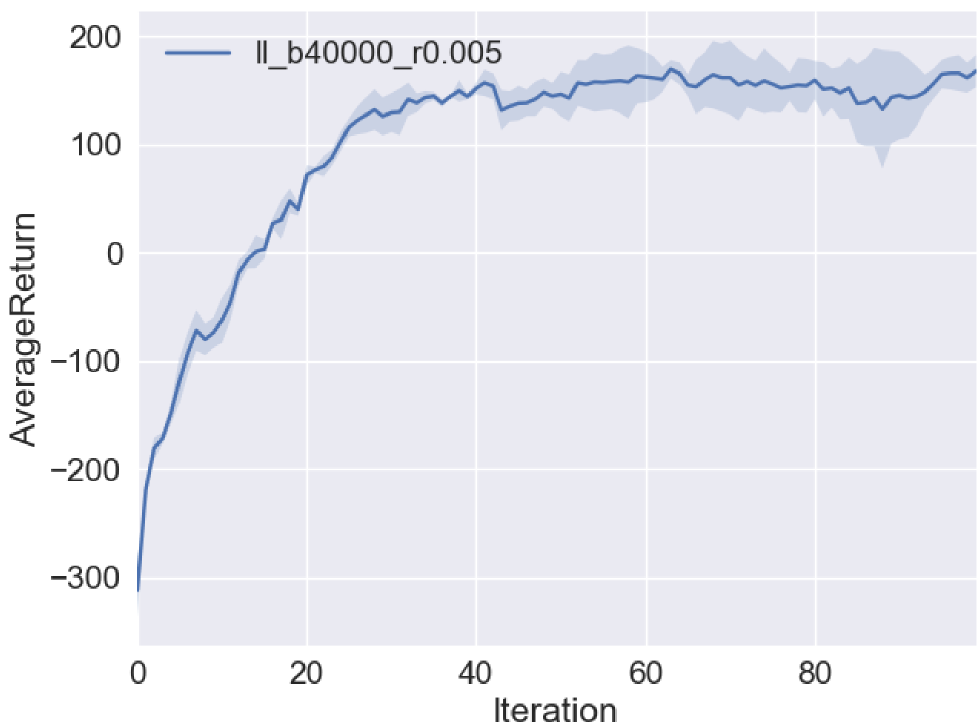

Figure 4: Learning Curve for LunarLander

Figure 5: Learning Curves for HalfCheetah

following commands is plotted in Figure 5
"python train_pg_f18.py HalfCheetah-v2 -ep 150 –discount 0.95 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.02 –exp_name hc_b50000_r0.02"
"python train_pg_f18.py HalfCheetah-v2 -ep 150 –discount 0.95 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.02 -rtg –exp_name hc_rtg_b50000_r0.02"
"python train_pg_f18.py HalfCheetah-v2 -ep 150 –discount 0.95 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.02 –nn_baseline –exp_name hc_baseline_b50000_r0.02"
"python train_pg_f18.py HalfCheetah-v2 -ep 150 –discount 0.95 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.02 -rtg –nn_baseline –exp_rtg_baseline_name hc_b50000_r0.02"