CS294-112 Deep Reinforcement Learning HW2: Policy Gradients

Ninh DO - SID#25949105

Problem 1. State-dependent baseline:

$$\sum_{t=1}^{T} \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \left(b(s_t) \right) \right] = 0. \tag{1}$$

(a) Please show equation 1 by using the law of iterated expectations, breaking $\mathbb{E}_{\tau \sim p_\theta(\tau)}$ by decoupling the state-action marginal from the rest of the trajectory.

Given $p_\theta(\tau) = p_\theta(s_t, a_t) p_\theta(\tau/s_t, a_t | s_t, a_t)$, we write:

$$\sum_{t=1}^{T} \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \left(b(s_t) \right) \right]$$

$$= \sum_{t=1}^{T} \mathbb{E}_{p_\theta(s_t, a_t)} \left[\mathbb{E}_{p_\theta(\tau/s_t, a_t|s_t, a_t)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \left(b(s_t) \right) \right] \right]$$

$$= \sum_{t=1}^{T} \int_{s_t} p_\theta(s_t, a_t) \int_{a_t} p_\theta(a_t|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \left(b(s_t) \right) da_t ds_t$$

(since $p_\theta(\tau/s_t, a_t|s_t, a_t) = p_\theta(a_t|s_t)$)

$$= \sum_{t=1}^{T} \int_{s_t} p_\theta(s_t, a_t) \int_{a_t} \nabla_\theta \pi_\theta(a_t|s_t) \left(b(s_t) \right) da_t ds_t$$

(since $p_\theta(a_t|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) = \nabla_\theta \pi_\theta(a_t|s_t)$, p_θ and π_θ are the same)

$$= \sum_{t=1}^{T} \int_{s_t} p_\theta(s_t, a_t) b(s_t) \nabla_\theta \int_{a_t} \pi_\theta(a_t|s_t) da_t ds_t$$

$$= \sum_{t=1}^{T} \int_{s_t} p_\theta(s_t, a_t) b(s_t) \nabla_\theta 1 da_t ds_t = 0$$

(since $\int_{a_t} \pi_\theta(a_t|s_t) da_t = 1, \nabla_\theta 1 = 0$)

(b) Alternatively, we can consider the structure of the MDP and express $p_\theta(\tau)$ as a product of the trajectory distribution up to s_t (which we denote as $(s_{1:t}, a_{1:t-1})$) and the trajectory distribution after s_t conditioned on the first part (which we denote as $(s_{t+1:T}, a_{t:T}|s_{1:t}, a_{1:t-1})$):

(a) Explain why, for the inner expectation, conditioning on $(s_1, a_1, ..., a_{t^*-1}, s_{t^*})$ is equivalent to conditioning only on s_{t^*}.

Since the Markov chain is memoryless, the current state/action only depends on its most recent action/state.

(b) Please show equation 1 by using the law of iterated expectations, breaking $\mathbb{E}_{\tau\sim p_\theta(\tau)}$ by decoupling trajectory up to s_t from the trajectory after s_t.

Given

$$\begin{aligned} p_\theta(\tau) &= p_\theta(s_{1:t}, a_{1:t-1})p_\theta(s_{t+1:T}, a_{t:T}|s_{1:t}, a_{1:t-1}) \\ &= p_\theta(s_{1:t}, a_{1:t-1})p_\theta(s_{t+1:T}, a_{t:T}|s_t) \end{aligned}$$

We write:

$$\begin{aligned}
&\sum_{t=1}^{T} \mathbb{E}_{\tau\sim p_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t)\,(b(s_t))\right] \\
&= \mathbb{E}_{p_\theta(s_{1:t^*}, a_{1:t^*-1})}\left[\mathbb{E}_{p_\theta(s_{t^*+1:T}, a_{t^*:T}|s_{t^*})}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\,(b(s_t))\right]\right] \\
&= \mathbb{E}_{p_\theta(s_{1:t^*}, a_{1:t^*-1})}\left[\mathbb{E}_{p_\theta(s_{t^*+1:T}, a_{t^*:T}|s_{t^*})}\left[\sum_{t=t^*}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\,(b(s_t))\right]\right] \\
&\qquad \text{(truncating the head of sum because probabilily distribution is } p_\theta(s_{t^*+1:T}, a_{t^*:T}|s_{t^*})) \\
&= \mathbb{E}_{p_\theta(s_{1:t^*}, a_{1:t^*-1})}\left[\mathbb{E}_{p_\theta(s_{t^*+1:T}, a_{t^*:T}|s_{t^*})}\left[\nabla_\theta \log \prod_{t=t^*}^{T} \pi_\theta(a_t|s_t)\,(b(s_t))\right]\right] \\
&= \mathbb{E}_{p_\theta(s_{1:t^*}, a_{1:t^*-1})}\left[\mathbb{E}_{p_\theta(s_{t^*+1:T}, a_{t^*:T}|s_{t^*})}\left[\nabla_\theta \log \pi_\theta(s_{t^*+1:T}, a_{t^*:T}|s_t)\,(b(s_{t^*}))\right]\right] \\
&= \int_{s_t} p_\theta(s_{1:t^*}, a_{1:t^*-1}) \int_{a_t} p_\theta(s_{t^*+1:T}, a_{t^*:T}|s_{t^*}) \nabla_\theta \log \pi_\theta(s_{t^*+1:T}, a_{t^*:T}|s_t)\,(b(s_t))\, da_t ds_t \\
&= \int_{s_t} p_\theta(s_{1:t^*}, a_{1:t^*-1}) \int_{a_t} \nabla_\theta \pi_\theta(s_{t^*+1:T}, a_{t^*:T}|s_t)\,(b(s_{t^*}))\, da_t ds_t \\
&= \int_{s_t} p_\theta(s_{1:t^*}, a_{1:t^*-1}) b(s_{t^*}) \nabla_\theta \int_{a_t} \pi_\theta(s_{t^*+1:T}, a_{t^*:T}|s_t) da_t ds_t \\
&= \int_{s_t} p_\theta(s_{1:t^*}, a_{1:t^*-1}) b(s_{t^*}) \nabla_\theta 1 da_t ds_t = 0 \\
&\qquad \left(\text{since } \int_{a_t} \pi_\theta(s_{t^*+1:T}, a_{t^*:T}|s_t) da_t = 1, \nabla_\theta 1 = 0\right)
\end{aligned}$$

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

<u>Command lines:</u>

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

<u>Command line:</u>

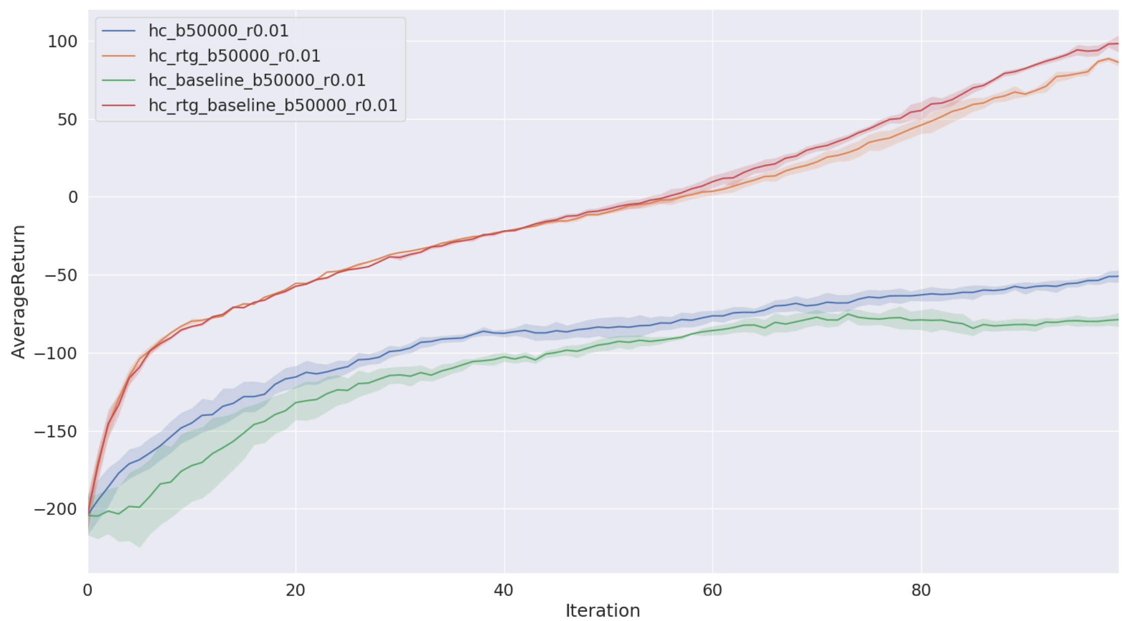
python train_pg_f18.py LunarLanderContinuous-v2 -ep 1000 --discount 0.99 -n 100 -e 3 -l 2 -s 64 -b 40000 -lr 0.005 -rtg --nn_baseline --exp_name ll_b40000_r0.005

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.



<u>Command lines:</u>

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/