CS294-112 HW2: Policy Gradients

Naijia Fan

September 19, 2018

Problem 1. State-dependent baseline

$$p_\theta(\tau) = p_\theta(s_t, a_t) p_\theta(\tau / s_t, a_t | s_t, a_t) \tag{1}$$
$$p_\theta(\tau) = p_\theta(s_{1:t}, a_{1:t-1}) p_\theta(s_{t+1:T}, a_{t:T} | s_{1:t}, a_{1:t-1}) \tag{2}$$

(a) Proof Our goal is to show that

$$\Sigma_{t=1}^{T} \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t)) \right] = 0 \tag{3}$$

Want to show that each term in this summation is equal to 0, that is:

$$\mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t)) \right] = 0 \tag{4}$$

Give the expression of $p_\theta(\tau)$ using the chain rule in Equation 1, we can rewrite our alternative target Equation 4 as below.

$$LHS = \mathbb{E}_{\tau \sim p_\theta(s_t, a_t) p_\theta(\tau / s_t, a_t | s_t, a_t)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t)) \right] \tag{5}$$
$$= \mathbb{E}_{\tau \sim p_\theta(\tau / s_t, a_t | s_t, a_t)} \mathbb{E}_{(s_t, a_t) \sim p_\theta(s_t, a_t)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t)) \right] \tag{6}$$

Using Markov Chain Rule, we have $p_\theta(s_t, a_t) = p_\theta(s_t) p_\theta(a_t|s_t)$. Then the inner expectation can be rewritten as

$$\mathbb{E}_{(s_t, a_t) \sim p_\theta(s_t, a_t)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t)) \right] \tag{7}$$
$$= \mathbb{E}_{(s_t, a_t) \sim p_\theta(s_t) p_\theta(a_t|s_t)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t)) \right] \tag{8}$$
$$= \mathbb{E}_{s_t \sim p_\theta(s_t)} \mathbb{E}_{a_t \sim p_\theta(a_t|s_t)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t)) \right] \tag{9}$$
$$= \int_{s_t} p_\theta(s_t) \int_{a_t} p_\theta(a_t|s_t) \left[\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t)) \right] da_t ds_t \tag{10}$$

Now it is time to use our favorite **convenient identity equation**(Equation 11).

$$\pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) = \nabla_\theta \pi_\theta(\tau) \tag{11}$$

Notice that

$$p_\theta(a_t|s_t) \equiv \pi_\theta(a_t|s_t).$$

Then, we can continuously rewrite the inner expectation in Equation 10.

$$\begin{aligned}
LHS = \cdots &= \int_{s_t} p_\theta(s_t) \int_{a_t} \nabla_\theta \pi_\theta(a_t|s_t) b(s_t) da_t ds_t && (12)\\
&= \int_{s_t} p_\theta(s_t) b(s_t) \left(\nabla_\theta \int_{a_t} \pi_\theta(a_t|s_t) da_t\right) ds_t && (13)\\
&= \int_{s_t} p_\theta(s_t) b(s_t) \left(\nabla_\theta 1\right) ds_t && (14)\\
&= \int_{s_t} p_\theta(s_t) b(s_t) \cdot 0 \cdot ds_t && (15)\\
&= 0 = RHS && (16)
\end{aligned}$$

□

(b.a) Proof We want to show the **property of causality**, that is

$$p_\theta(\cdot|s_1, a_1, ..., a_{t^*-1}, s_{t^*}) = p_\theta(\cdot|s_{t^*}) \tag{17}$$

Using Bayes Rule, we know that

$$LHS = \frac{p_\theta(\cdot, s_1, a_1, ..., a_{t^*-1}, s_{t^*})}{p_\theta(s_1, a_1, ..., a_{t^*-1}, s_{t^*})} \tag{18}$$

Then, using property of Markov Chain we have

$$p_\theta(\cdot, s_1, a_1, ..., a_{t^*-1}, s_{t^*}) = p_\theta(s_1) \prod_{t=1}^{T} p_\theta(a_t|s_t) \prod_{t=2}^{T} p_\theta(s_t|s_{t-1}, a_{t-1})) \tag{19}$$

$$p_\theta(s_1, a_1, ..., a_{t^*-1}, s_{t^*}) = p_\theta(s_1) \prod_{t=1}^{t^*} p_\theta(a_t|s_t) \prod_{t=2}^{t^*} p_\theta(s_t|s_{t-1}, a_{t-1}) \tag{20}$$

Then, move on with LHS, we get

$$LHS = \cdots = \prod_{t=t^*+1}^{T} p_\theta(a_t|s_t) \prod_{t=t^*+1}^{T} p_\theta(s_t|s_{t-1}, a_{t-1}) \tag{21}$$

Similarly, we can rewrite $p_\theta(\cdot|s_{t^*})$ as

$$\begin{aligned}
RHS = \frac{p_\theta(\cdot, s_{t^*})}{p_\theta(s_{t^*})} &= \frac{p_\theta(s_{t^*}) \prod_{t=t^*+1}^{T} p_\theta(a_t|s_t) \prod_{t=t^*+1}^{T} p_\theta(s_t|s_{t-1}, a_{t-1})}{p_\theta(s_{t^*})} && (22)\\
&= \prod_{t=t^*+1}^{T} p_\theta(a_t|s_t) \prod_{t=t^*+1}^{T} p_\theta(s_t|s_{t-1}, a_{t-1}) && (23)
\end{aligned}$$

Thus, clearly, we have

$$LHS = RHS$$

$\square$

(b.b) Proof We want to proof Equation 3 again given **breaking down rule** in Equation 2 now. At this time, applying the conclusion we have in the previous question, the Equation 6 will be like

$$\mathbb{E}_{(s_{1:t},a_{1:t-1})\sim p_\theta(s_{1:t},a_{1:t-1})}\mathbb{E}_{\tau\sim p_\theta(s_{t+1:T},a_{t:T}|s_{1:t},a_{1:t-1})}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t))\right] \tag{24}$$
$$= \mathbb{E}_{s_t\sim p_\theta(s_{1:t},a_{1:t-1})}\mathbb{E}_{\tau\sim p_\theta(s_{t+1:T},a_{t:T}|s_t)}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t))\right] \tag{25}$$
$$= \mathbb{E}_{\tau\sim p_\theta(s_{t+1:T},a_{t:T}|s_t)}\mathbb{E}_{s_t\sim p_\theta(s_{1:t},a_{1:t-1})}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t))\right] \tag{26}$$

Given the property of Markov Chian, the inner expectation of Equation 26 can be rewritten as

$$\mathbb{E}_{s_t\sim p_\theta(s_{1:t},a_{1:t-1})}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t))\right] \tag{27}$$
$$= \mathbb{E}_{s_t\sim p_\theta(s_1)\prod_{t'=1}^{t} p_\theta(a_t'|s_t')\prod_{t'=2}^{t} p_\theta(s_{t'}|s_{t'-1},a_{t'-1})}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t))\right] \tag{28}$$
$$= \prod_{t'=2}^{t} \mathbb{E}_{s_t'\sim p_\theta(s_{t'}|s_{t'-1},a_{t'-1})} \prod_{t'=1}^{t} \mathbb{E}_{a_t'\sim p_\theta(a_t'|s_t')} \nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t)) \tag{29}$$
$$= [\mathbb{E}\cdots\mathbb{E}]\left[\mathbb{E}_{a_t\sim p_\theta(a_t|s_t)} \nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t))\right] \tag{30}$$
$$= [\mathbb{E}\cdots\mathbb{E}]\underbrace{\int_{a_t} p_\theta(a_t|s_t)\left[\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t))\right] da_t}_{0} \tag{31}$$
$$= 0 \tag{32}$$

Notice that under-braced part in Equation 30 is implied to be 0 by the Equation 13, 14 and 15.

Problem 2. Neural networks

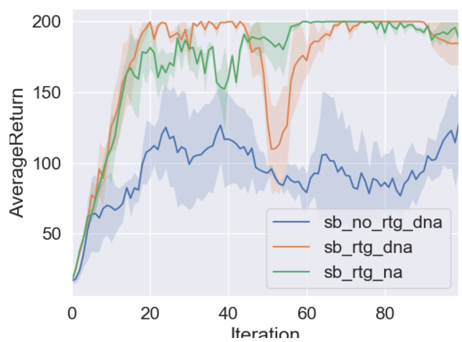
See the codes in `codes.zip`.

Problem 3. Policy Gradient

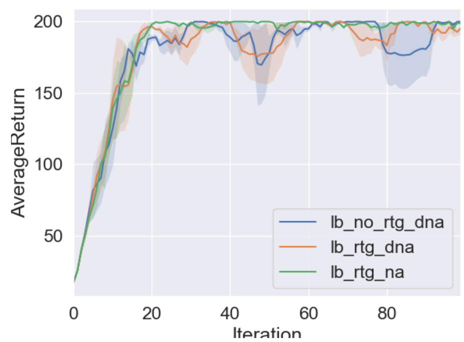
See the codes in `codes.zip`.

Problem 4. CartPole

Graphs



(a) Compare the learning curves with small batch size.



(b) Compare the learning curves with large batch size.

Figure 1: The same cup of coffee. Two times.

Answers

- Which gradient estimator has better performance without advantage-centering—the trajectory-centric one, or the one using reward-to-go?
  The one using reward-to-go.
- Did advantage centering help?
  Yes. It helps reduce the variance, which is more obvious when using small batch.
- Did the batch size make an impact?
  Yes. Larger batch size results in better performance.

Command Line Configurations

Configurations can be refered in the bash script. More details in the `cartpole.sh` in `codes.zip`.

```
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
```

Listing 1: run CartPole

Problem 5. InvertedPendulum

Graph

We determine `b⋆=900, r⋆=0.03` as the smallest batch size and largest learning rate to get the optimum (~ 1000) in less than 100 iterations. And its learning curve is shown in Figure 2.

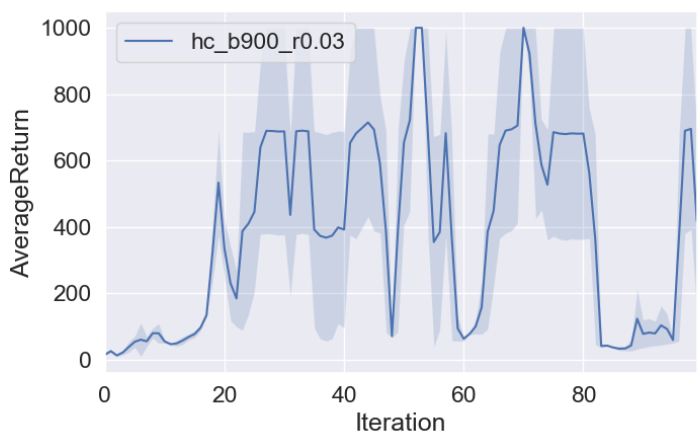


Figure 2: Learning curve where the policy get the optimum within 100 iterations in a fluctuate way.

Command Line Configurations

The fixed configuration is `-ep 1000 --discount 0.9 -n 100 -e 3 -l 2 -s 64 -rtg`. And we search for the boundary `b` and `r` by several rounds.

1. Coarse search for `r`: Let `b=5000` and `r` $\in [0.0001, 0.001, 0.01, 0.1]$; Picked `r=0.01`.
2. Coarse search for `b`: Let `r=0.01` and `b` $\in [50, 500, 5000, 50000]$; Picked `b=5000`.
3. Fine search for smaller `b`: Let `r=0.01` and `b` $\in [800, 1000, 2000, 5000]$; Picked `b=800-1000`.
4. Fine search for larger `r`: Let `b=1000` and `r` $\in [0.005, 0.01, 0.02, 0.04]$; Picked `r=0.02-0.04`.
5. Final search for largest `r`: Let `b=1000` and `r` $\in [0.02, 0.03, 0.04]$; Picked `r=0.03`.

6. Final search for smallest b: Let r=0.03 and b $\in [800, 900, 1000]$; Picked b=900.

Part of the bash script is as follows. More details in the inverted_pendulum.sh in codes.zip.

```
for b in 5000
do
    for r in 0.0001 0.001 0.01 0.1
    do
        tailtag="b$b""_r$r"
        echo "b = $b, r = $r, tail_tag = $tailtag"
        cmd="python train_pg_f18.py InvertedPendulum-v2 -ep 1000 --discount
   0.9 -n 100 -e 3 -l 2 \
            -s 64 -b $b -lr $r -rtg --exp_name hc_$tailtag"
        echo $cmd
        eval $cmd
    done
done

......
```

Listing 2: run InvertedPendulum

Problem 6. Neural network baseline

See the codes in codes.zip.

Problem 7. LunarLander

The learning curve of using our policy gradient implementation with an episode length of 1000 to solve LunarLanderContinuous-v2 is shown in Figure 3.

Problem 8. HalfCheetah

Answer

Within those choices, where b $\in [10000, 30000, 50000]$ and r $\in [0.005, 0.01, 0.02]$, the larger batch size and larger learning rate result in better performance. Figure 4 depicts this conclusion.

Graph

b⋆=50000, r=0.02 are picked as the most suitable. The single plot plotting the learning curves for all four runs using this suitable configuration is shown in Figure 5.

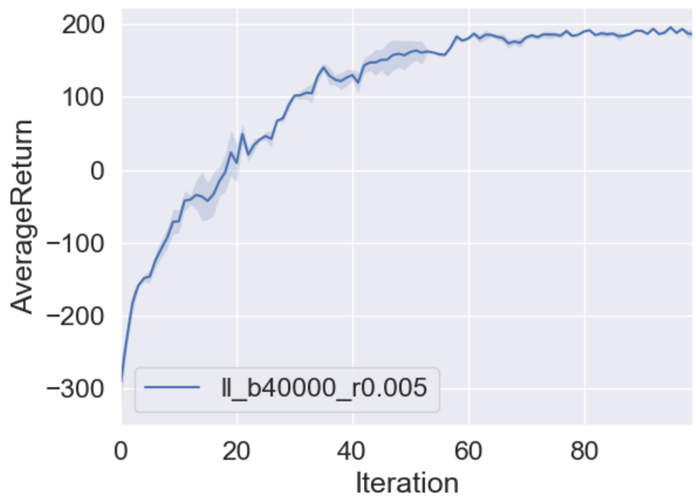


Figure 3: Learning curve of LunarLander with reward-to-go and baseline strategy, whose `b=40000` and `r=0.005`.

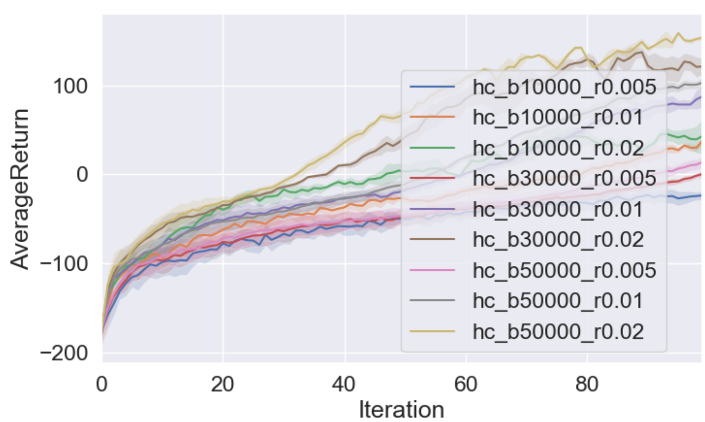


Figure 4: Search over batch sizes b [10000, 30000, 50000] and learning rates r [0.005, 0.01, 0.02].

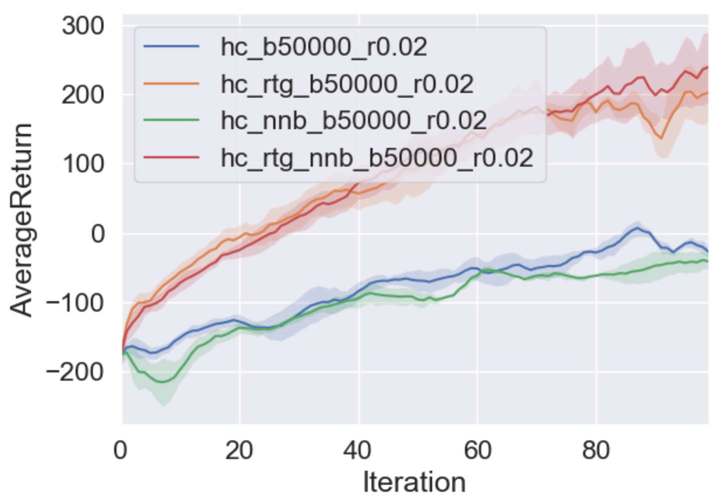


Figure 5: Use suitable `b*` and `r*` to compare different strategies with/without reward-to-go or baseline.