CS294-112 Deep Reinforcement Learning HW2: Policy Gradients

Louis Tilloy

Problem 1

1) Let us show that :

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

$$\begin{aligned}
&\sum_{t=1}^{T} \mathbb{E}_{\tau\sim p_\theta(\tau)}[\nabla_\theta \log \pi_\theta(a_t|s_t)\, b(s_t)] \\
&= \sum_{t=1}^{T} \mathbb{E}_{\tau\sim p_\theta(\tau)}[\mathbb{E}_{s_t,a_t\sim p_\theta(s_t,a_t)}[\nabla_\theta \log \pi_\theta(a_t|s_t)\, b(s_t)|\tau/s_t,a_t]] \\
&= \sum_{t=1}^{T} \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\int_{s_t,a_t\in S_t,A_t} \nabla_\theta \log \pi_\theta(a_t|s_t)\, b(s_t) p_\theta(s_t,a_t|\tau/s_t,a_t)\, ds_t\, da_t\right] \\
&= \sum_{t=1}^{T} \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\int_{s_t,a_t\in S_t,A_t} \pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\, b(s_t) p_\theta(s_t|s_{t-1},a_{t-1})\, ds_t\, da_t\right] \\
&= \sum_{t=1}^{T} \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\int_{s_t\in S_t} \nabla_\theta(\int_{a_t\in A_t} \pi_\theta(a_t|s_t)\, da_t)\, b(s_t) p_\theta(s_t|s_{t-1},a_{t-1})\, ds_t\right] \\
&= \sum_{t=1}^{T} \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\int_{s_t\in S_t} \nabla_\theta(1)\, b(s_t) p_\theta(s_t|s_{t-1},a_{t-1})\, ds_t\right] \\
&= 0
\end{aligned}$$

2) a) Intuitively, conditioning over $(a_1, s_1, ..., a_{t^*-1}, s_{t^*})$ is the same as conditioning over s_{t^*} because of the markovian propriety of s_t that only depends on s_{t-1} and the fact that a_t only depends on s_t. (see FIGURE 1 taken from a slide used in class for illustration).

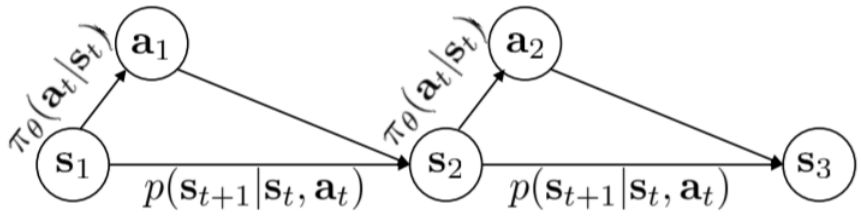

FIGURE 1 – Figure extracted from slide 13, lecture 4 of the fall 2018 UC Berkeley Deep Reinforcement Learning course (click here to see the slides)

We can also do a more mathematical proof :

$$\begin{aligned}
&\mathbb{E}_{\tau\sim p_\theta(\tau)}[\nabla_\theta \log \pi_\theta(a_{t^*}|s_{t^*}) b(s_{t^*})|(s_1,a_1,...,a_{t^*-1},s_{t^*})] \\
&= \int_{a_{t^*},s_{t^*+1},...,s_T,a_T} p(a_{t^*},s_{t^*+1},a_{t^*+1},...,s_T,a_T|s_1,a_1,..,a_{t^*-1},s_{t^*})\nabla_\theta \log \pi_\theta(a_{t^*}|s_{t^*}) b(s_1,...,s_T)\, da_{t^*}\, ...\, ds_T\, da_T \\
&= \int_{a_{t^*},s_{t^*+1},...,s_T,a_T} p(a_{t^*},s_{t^*+1},a_{t^*+1},...,s_T,a_T|s_{t^*})\nabla_\theta \log \pi_\theta(a_{t^*}|s_{t^*}) b(s_1,...,s_T)\, da_{t^*}\, ...\, ds_T\, da_T \\
&= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(a_{t^*}|s_{t^*}) b(s_{t^*})|s_{t^*}]
\end{aligned}$$

b) Let us show that :

$$\sum_{t=1}^{T} \mathbb{E}_{\tau\sim p_\theta(\tau)}[\nabla_\theta \log \pi_\theta(a_t|s_t)\, b(s_t)] = 0 \tag{2}$$

$$
\begin{aligned}
&\sum_{t=1}^{T} \mathbb{E}_{\tau \sim p_\theta(\tau)}[\nabla_\theta \log \pi_\theta(a_t|s_t)\, b(s_t)] \\
&= \sum_{t=1}^{T} \mathbb{E}_{\tau \sim p_\theta(\tau)}[\mathbb{E}_{s_t, a_t, \ldots \sim p_\theta(s_t, a_t, \ldots)}[\nabla_\theta \log \pi_\theta(a_t|s_t)\, b(s_t)|s_1, a_1, ..., a_{t-1}, s_t]] \\
&= \sum_{t=1}^{T} \mathbb{E}_{\tau \sim p_\theta(\tau)}[\mathbb{E}_{s_t, a_t, \ldots \sim p_\theta(s_t, a_t, \ldots)}[\nabla_\theta \log \pi_\theta(a_t|s_t)\, b(s_t)|s_t]] \\
&= \sum_{t=1}^{T} \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\int_{a_t, s_{t+1}, \ldots \in A_t \times S_{t+1}, \times \ldots} \nabla_\theta \log \pi_\theta(a_t|s_t)\, b(s_t) p_\theta(a_t, s_{t+1}, a_{t+1}, ...|s_t)\, da_t\, ds_{t+1} \ldots \right] \\
&= \sum_{t=1}^{T} \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\int_{a_t \in A_t} \pi_\theta(a_t|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t)\, b(s_t) \left(\int_{s_{t+1}, a_{t+1}, \ldots} p_\theta(s_{t+1}, a_{t+1}, ...|s_t, a_t) ds_{t+1}\, da_{t+1} \ldots \right) da_t \right] \\
&= \sum_{t=1}^{T} \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \left(\int_{a_t \in A_t} \pi_\theta(a_t|s_t)\, da_t \right) b(s_t) \right] \\
&= \sum_{t=1}^{T} \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta (1) b(s_t) \right] \\
&= 0
\end{aligned}
$$

Problem 4

FIGURE 2 – Learning curves for the **small batches** experiments ("no rtg dna" : experiment without reward-to-go and without normalizing advantages, "rtg dna" : experiment with reward-to-go and without normalizing advantages, "rtg na" : experiment with reward-to-go and advantages normalization). We can see that the reward-to-go largely improves the learning of the network. The gain from advantages normalization is more subtle, but we can see that it indeed reduces variance : the experiment with normalization is a lot more stable at the end of the training.

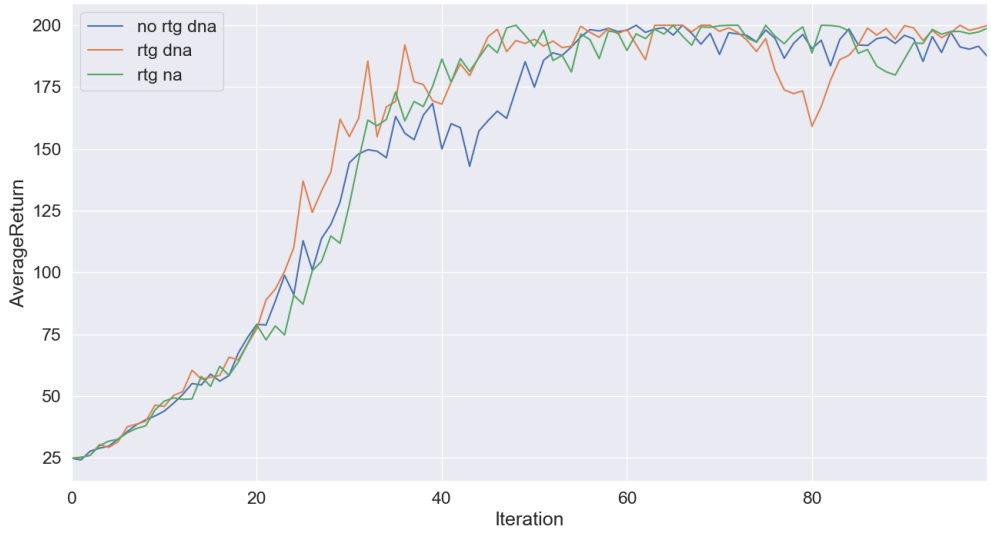

FIGURE 3 – Learning curves for the **large batches** experiments ("no rtg dna" : experiment without reward-to-go and without normalizing advantages, "rtg dna" : experiment with reward-to-go and without normalizing advantages, "rtg na" : experiment with reward-to-go and advantages normalization). We can see that training with large batches reduces the difference between the 3 experiments. Still, reward-to-go seems to provide better results and advantages normalization seems to produce a more stable training.

Questions

☐ Which gradient estimator has better performance without advantage-centering, the trajectory-centric one, or the one using reward-to-go ?

The gradient estimator with reward-to-go has better performance without advantage-centering than the trajectory-centric one.

☐ Did advantage centering help ?

Yes, it reduced the variance of the return.

☐ Did the batch size make an impact ?

Yes, the larger batches had better results because they largely reduced the variance of the returns.

All 6 experiments were made with the same command as the ones provided in the pdf :

```
python train_pg_f18.py CartPole-v0 -n 100 -b 1000 -e 1 -dna --exp_name
    sb_no_rtg_dna
python train_pg_f18.py CartPole-v0 -n 100 -b 1000 -e 1 -rtg -dna --exp_name
    sb_rtg_dna
python train_pg_f18.py CartPole-v0 -n 100 -b 1000 -e 1 -rtg --exp_name
    sb_rtg_na
python train_pg_f18.py CartPole-v0 -n 100 -b 5000 -e 1 -dna --exp_name
    lb_no_rtg_dna
python train_pg_f18.py CartPole-v0 -n 100 -b 5000 -e 1 -rtg -dna --exp_name
    lb_rtg_dna
python train_pg_f18.py CartPole-v0 -n 100 -b 5000 -e 1 -rtg --exp_name
    lb_rtg_na
```

Problem 5

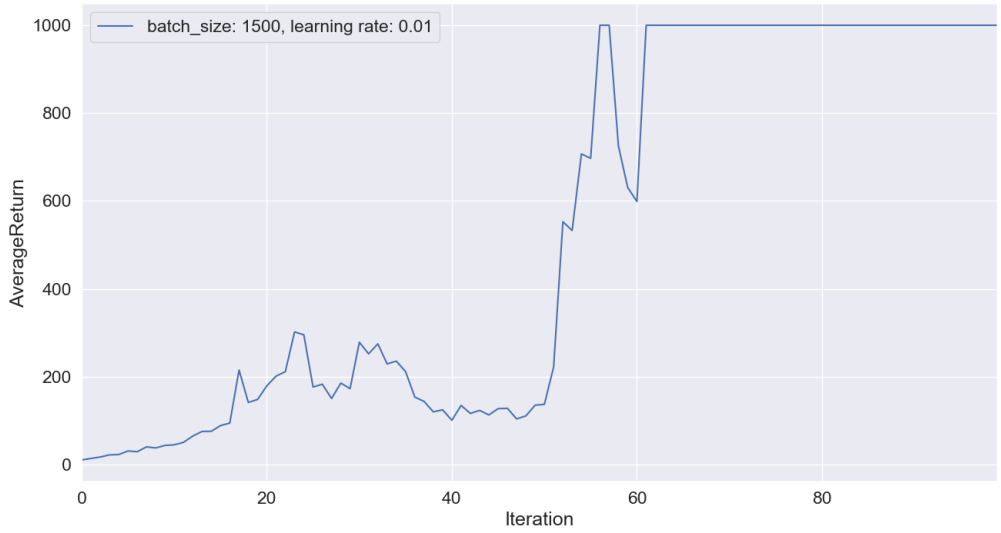

FIGURE 4 – Learning curve of the neural network for the Inverted Pendulum with a batch size of 1500 and a rate of 0.01 . Smaller batches did not go to the score of 1000 as fast and bigger learning rates created high instability at the end of the training, leading to random spikes of scores of about 300.

To get the data for this figure, the same command as in the pdf was used :

```
python train_pg_f18.py InvertedPendulum-v2 -ep 1000 --discount 0.9 -n 100
    -e 1 -l 2 -s 64 -b 1500 -lr 0.01 -rtg --exp_name hc_b1500_r0.01
```

Problem 7

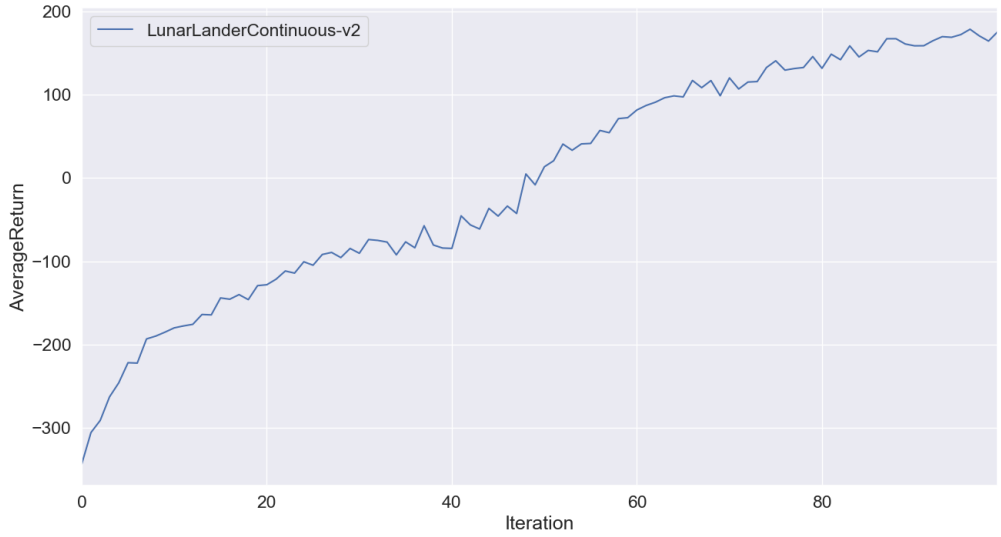

FIGURE 5 – Lurning curve from the Lunar Lander experiment with baseline computed by a side neural network.

The data to plot figure was obtained by running the same command as in the pdf :

```
python train_pg_f18.py LunarLanderContinuous-v2 -ep 1000 --discount 0.99 -n
100 -e 1 -l 2 -s 64 -b 40000 -lr 0.005 -rtg --nn_baseline --exp_name
ll_b40000_r0.005
```

Problem 8

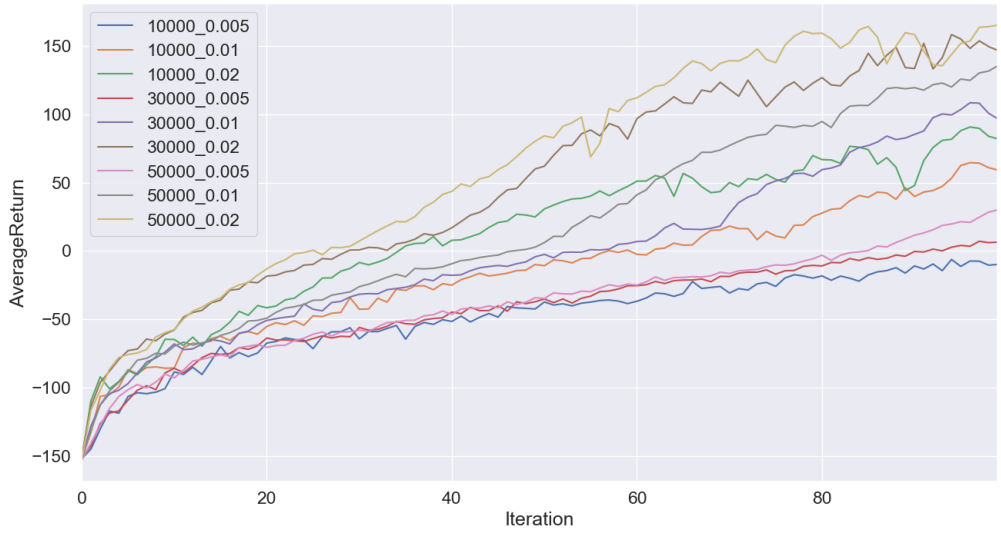

FIGURE 6 – This figure has been made only to select which batch size and learning rate are the best. It represents learning curves of the neural network with different batch size and learning rate. The reward to go and the baseline was used, with a discount factor of 0.9 . It seems like it is **batch size of 50000** and **learning rate of 0.02** that have the higher curve overall.

□ How did the batch size and learning rate affect the performance ?

Increasing the batch size tends to increase performance with no other downside than the higher time it takes to train the network with a bigger batch. Increasing the learning rate is not as straightforward, it increases performance while adding some noise that grows the higher the rate is.

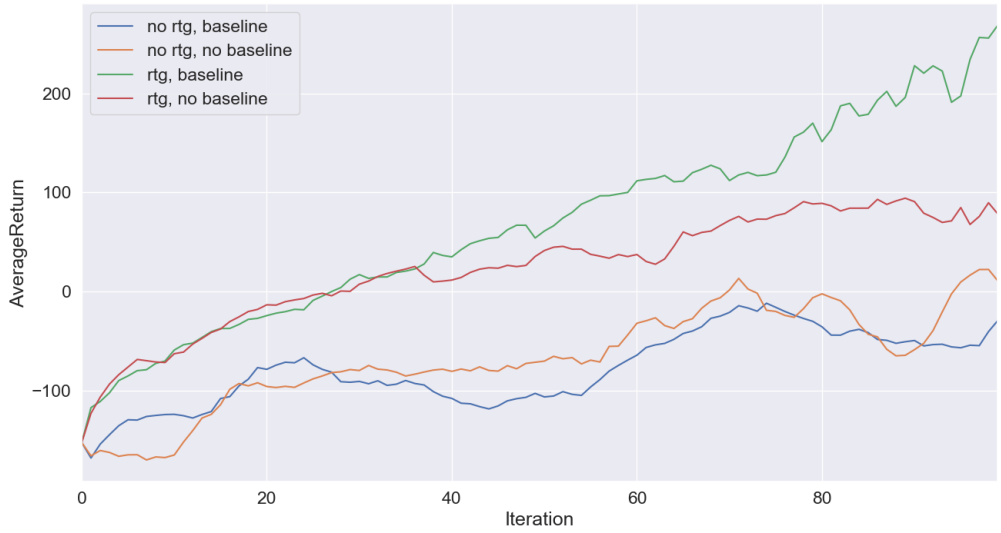

FIGURE 7 – Learning curves of the neural network with batch of 50000, learning rate of 0.02 and different parameters ("rtg"/"no rtg" : whether or not the reward-to-go was used to compute the gradient of the main network, "baseline"/"no baseline" : whether or not a baseline predicted by another neural network was used).

To get the data for this figure, the same command as in the pdf was used :

```
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 1
    -l 2 -s 32 -b 50000 -lr 0.02 --exp_name hc_b50000_r0.02_batch_5
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 1
    -l 2 -s 32 -b 50000 -lr 0.02 -rtg --exp_name hc_b50000_r0.02_rtg_batch_5
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 1
    -l 2 -s 32 -b 50000 -lr 0.02 --nn_baseline
    --exp_name hc_b50000_r0.02_b_batch_5
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 1
    -l 2 -s 32 -b 50000 -lr 0.02 -rtg --nn_baseline
    --exp_name hc_b50000_r0.02_rtg_b_batch_5
```