Problem 1 Stare-dependent baseline

Proof:

(a)

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

(b) By the Markovian assumption that the future trajectory is influenced only by the present state. Therefore, for the inner expectation, conditioning on $(s_1, a_1, \cdots, a_{t^*-1}, s_{t^*})$ is equivalent to conditioning only on s_{t^*}.

$$
\begin{aligned}
&\mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)\right]\\
=&\quad\sum_{t=1}^{T}\mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)\right]\\
=&\quad\sum_{t=1}^{T}\mathbb{E}_{\tau\sim p_\theta(s_{1:t},a_{1:t-1})p_\theta(s_{t+1:T},a_{t:T}|s_{1:t},a_{1:t-1})}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)\right]\\
=&\quad\sum_{t=1}^{T}\mathbb{E}_{(s_{1:t},a_{1:t-1})\sim p_\theta(s_{1:t},a_{1:t-1})}\left[b(s_t)\mathbb{E}_{(s_{t+1:T},a_{t:T})\sim p_\theta(s_{t+1:T},a_{t:T}|s_{1:t},a_{1:t-1})}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\right]\right]
\end{aligned}
$$

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

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

$$= \sum_{t=1}^{T} \mathbb{E}_{(s_{1:t}, a_{1:t-1}) \theta(s_{1:t}, a_{1:t-1})} \left[b(s_t) \mathbb{E}_{(s_{t+1:T}, a_{t+1:T}) \sim p_\theta(s_{t+1:T}, a_{t:T} | s_t, a_t)} \left[0 \right] \right] = 0$$

Problem 2: Neural networks
See the code file “train_py_f18.py”.

Problem 3: Policy Gradient
See the code file “train_pg_f18.py”.

Problem 4: CartPole

Figure 1. Learning curves for CartPole with batch size 1000.

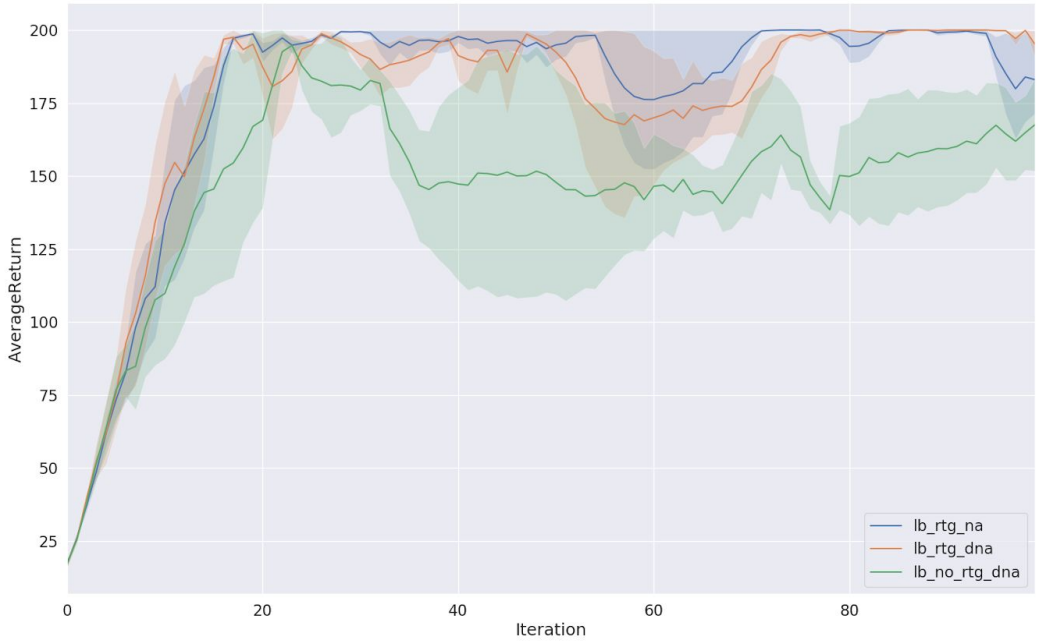

Figure 2. Learning curves for Cartpole with batch size 5000.

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
```

1. The gradient estimator using reward-to-go has better performance without advantage-centering.
2. Advantage centering helps reduce the variance of policy gradient approximation.
3. The experiments using large batch size have better performance with in same iterations.

Problem 5: Inverted Pendulum

The optimal batch size is b* = 5000. The optimal learning rate lr* = 0.005.

Figure 3. Learning curve for Inverted Pendulum.

Command line

```
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 the code file “train_pg_f18.py”.

Problem 7: Lunar Lander

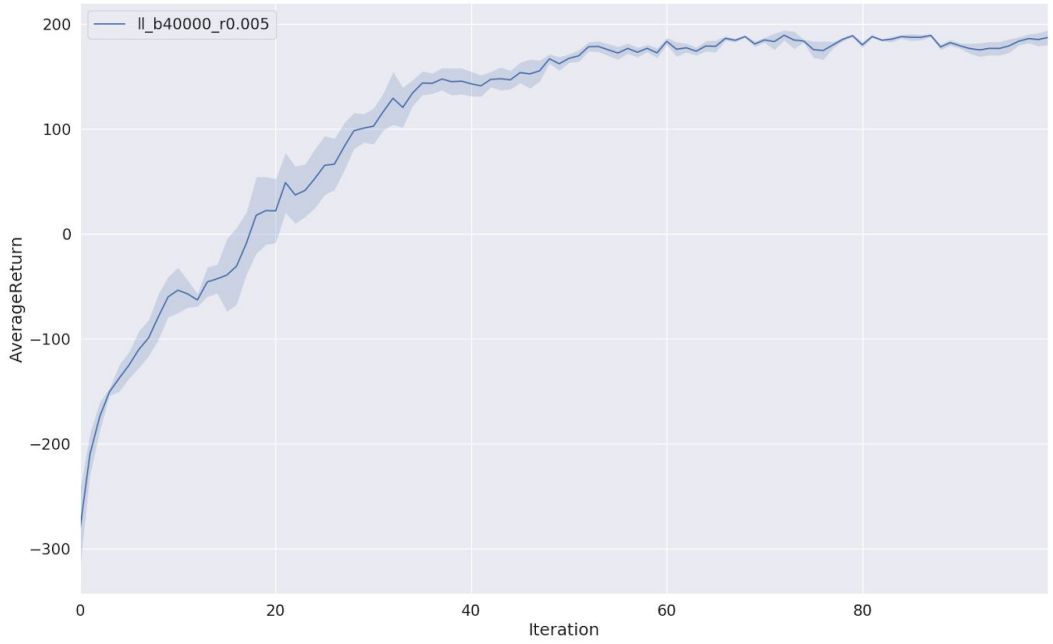

Figure 4. Learning curve for Lunar Lander.

Command line

```
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: Half Cheetah

Batch size = 50000. Learning rate = 0.01.

As the batch size increases and the learning rate decreases, the training process becomes more stable. The variance of the performance with different random seeds also decreases, but the computation cost increases.

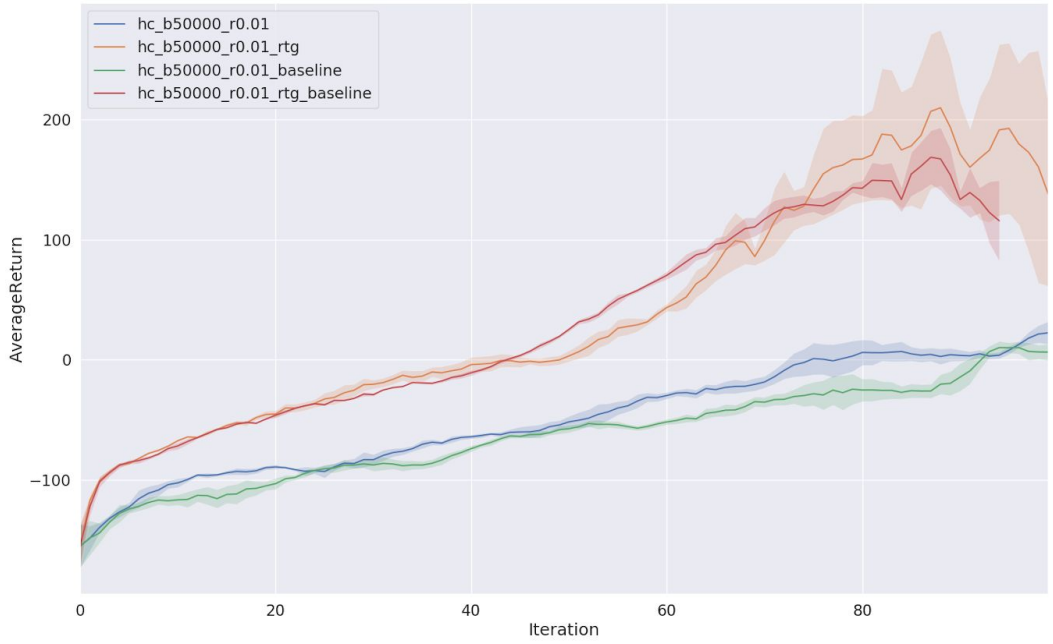

Figure 5. Learning curves for Half Cheetah.

Command lines

```
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.95 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.01 --exp_name ll_b40000_r0.005
```

```
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.95 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.01 -rtg --exp_name ll_b40000_r0.005
```

```
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.95 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.01 --nn_baseline --exp_name ll_b40000_r0.005
```

```
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.95 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.01 -rtg --nn_baseline --exp_name ll_b40000_r0.005
```