CD294-112 Deep Reinforcement Learning HW2: Policy Gradient

Ke Wang, EECS Ph.D. student, kewang@berkeley.com

2018/09/11

1 Problem 1. State-dependent baseline

1.1 policy gradient question 1

Note that by linearity of expectation the objective can be written as:

$$\begin{aligned} J(\theta) &= \mathbb{E}_{r\sim\pi_\theta(r)}[r(\tau)] \\ &= \mathbb{E}_{r\sim\pi_\theta(r)}[\sum_{t=1}^{T} r(s_t, a_t)] \\ &= \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p(s_t,a_t)}[r(s_t, a_t)] \end{aligned} \tag{1}$$

when we subtract the baseline $b(s_t)$, the objective becomes:

$$= \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p(s_t,a_t)}[r(s_t, a_t) - b(s_t)] \tag{2}$$

Please show that

$$\nabla_\theta \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p(s_t,a_t)}[b(s_t)] = 0 \tag{3}$$

According to the direct policy differentiation and law of iterated expectations.

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p(s_t,a_t)}[b(s_t)] \\ &= \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p(s_t,a_t)}[\nabla_\theta \log \pi_\theta(s_t, a_t) b(s_t)] \\ &= \sum_{t=1}^{T} \mathbb{E}_{s_t}[\mathbb{E}_{a_t}[\nabla_\theta \log \pi_\theta(a_t|s_s) b(s_t)]] \\ &= \sum_{t=1}^{T} \mathbb{E}_{s_t}[b(s_t) \cdot \mathbb{E}_{a_t}[\nabla_\theta \log \pi_\theta(a_t|s_t)]] \end{aligned} \tag{4}$$

Now, what we want to do is to show that $\mathbb{E}_{(a_t)}[\nabla_\theta \log \pi_\theta(a_t|s_t)] = 0$

$$\begin{aligned}
\mathbb{E}_{a_t}[\nabla_\theta \log \pi_\theta(a_t|s_t)] &= \int \pi_\theta(a_t) \nabla_\theta \log \pi_\theta(a_t|s_t) a a_t \\
&= \int \frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)} \pi_\theta(a_t|s_t) da_t \\
&= \int \frac{\pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)} \nabla_\theta \pi_\theta(a_t|s_t) da_t \\
&= \int \nabla_\theta \pi_\theta(a_t|s_t) da_t \\
&= \nabla_\theta \int \pi_\theta(a_t|s_t) da_t \\
&= \nabla_\theta 1 \\
&= 0
\end{aligned} \tag{5}$$

In this regard, we could prove the original equation

$$\begin{aligned}
\nabla_\theta J(\theta) &= \sum_{t=1}^{T} \mathbb{E}_{s_t}[b(s_t) \cdot \mathbb{E}_{a_t}[\nabla_\theta \log \pi_\theta(a_t|s_t)]] \\
&= \sum_{t=1}^{T} \mathbb{E}_{s_t}[b(s_t) \cdot \mathbb{E}_{a_t}[0]] \\
&= \sum_{t=1}^{T} \mathbb{E}_{s_t}[0] \\
&= 0
\end{aligned} \tag{6}$$

1.2 policy gradient question 2

1.2.1 problem a

Equation 12 could be split into two expectations, where the outer expectation is over $(s_1, a_1, ..., a_{t^*-1}, s_{t^*})$,and the inner expectation is over the rest of the trajectory. So the gradient could be rewritten as follow:

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[b(s_{t^*})] &= \nabla_\theta \mathbb{E}_{s_1, a_1, ..., a_{t^*-1}, s_{t^*}}[\mathbb{E}_{a_{t^*}, s_{t^*+1}, ..., s_T} b(s_{t^*})] \\
&= \nabla_\theta \mathbb{E}_{s_1, a_1, ..., a_{t^*-1}, s_{t^*}}[\int b(s_{t^*}) \pi_\theta(s_1, a_1, ..., s_T|s_1, a_1, ..., a_{t^*-1}, s_{t^*}) d\tau] \\
&= \nabla_\theta \mathbb{E}_{s_1, a_1, ..., a_{t^*-1}, s_{t^*}}[b(s_{t^*})[\int \pi_\theta(s_1, a_1, ..., s_T|s_1, a_1, ..., a_{t^*-1}, s_{t^*}) d\tau]] \\
&= \nabla_\theta \mathbb{E}_{s_1, a_1, ..., a_{t^*-1}, s_{t^*}}[b(s_{t^*}) \cdot 1] \\
&= \nabla_\theta \mathbb{E}_{s_1, a_1, ..., a_{t^*-1}, s_{t^*}}[b(s_{t^*})]
\end{aligned} \tag{7}$$

When the inner expectation is conditioning only on s_{t^*}

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[b(s_{t^*})_{inner}] &= \nabla_\theta \mathbb{E}_{s_1, a_1, ..., a_{t^*-1}, s_{t^*}}[\int b(s_{t^*}) \pi_\theta(s_1, a_1, ..., s_T|s_{t^*}) d\tau] \\
&= \nabla_\theta \mathbb{E}_{s_1, a_1, ..., a_{t^*-1}, s_{t^*}}[b(s_{t^*})[\int \pi_\theta(s_1, a_1, ..., s_T|s_{t^*}) d\tau]] \\
&= \nabla_\theta \mathbb{E}_{s_1, a_1, ..., a_{t^*-1}, s_{t^*}}[b(s_{t^*}) \cdot 1] \\
&= \nabla_\theta \mathbb{E}_{s_1, a_1, ..., a_{t^*-1}, s_{t^*}}[b(s_{t^*})]
\end{aligned} \tag{8}$$

Based on the previous analysis, the inner expectation, conditioning on $(s_1, a_1, ..., a_{t^*-1}, s_{t^*})$ is equivalent to conditioning only on s_{t^*}.

1.2.2 problem b

Using the iterated expectation described above and the results in problem a, we could get

$$\begin{aligned}\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[b(s_{t^*})] &= \nabla_\theta \mathbb{E}_{s_1,a_1,\ldots,a_{t^*-1},s_{t^*}}[b(s_{t^*})] \\ &= \nabla_\theta \mathbb{E}_{s_1,a_1,\ldots,a_{t^*-1},s_{t^*},a_{t^*}}[b(s_{t^*})]\end{aligned} \tag{9}$$

Set $(s_1, a_1, s_2, a_2, ..., s_{t^*}) = k_{t^*}$

$$\begin{aligned}\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[b(s_{t^*})] &= \nabla_\theta \mathbb{E}_{k_{t^*},a_{t^*}}[b(k_{t^*})] \\ &= \mathbb{E}_{(k_{t^*},a_{t^*})\sim p(k_{t^*},a_{t^*})}[\nabla_\theta \log \pi_\theta(k_{t^*}, a_{t^*}) b(k_{t^*})] \\ &= \mathbb{E}_{k_t}[\mathbb{E}_{a_t}[\nabla_\theta \log \pi_\theta(a_t|k_s) b(k_t)]] \\ &= \mathbb{E}_{k_t}[b(k_t)\mathbb{E}_{a_t}[\nabla_\theta \log \pi_\theta(a_t|k_s)]] \\ &= \mathbb{E}_{k_{t^*}}[b(k_t) \cdot \mathbb{E}_{a_t}[0]] \\ &= 0\end{aligned} \tag{10}$$

2 Problems 4,5: Implement Policy Gradient

2.1 command line for implementing policy gradient and plot figures

Implement Policy Gradient:

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 -rtg –exp_name lb_no_rtg_na

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

plot figures:

python plot.py data/file1 data/file2 data/file3 ...

2.2 graphs of learning curves

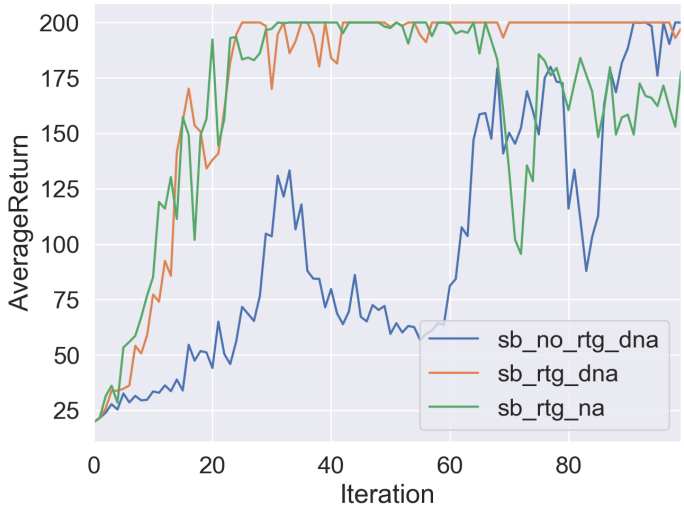

Figure 1: Comparation between different learning curves under small batch sizes

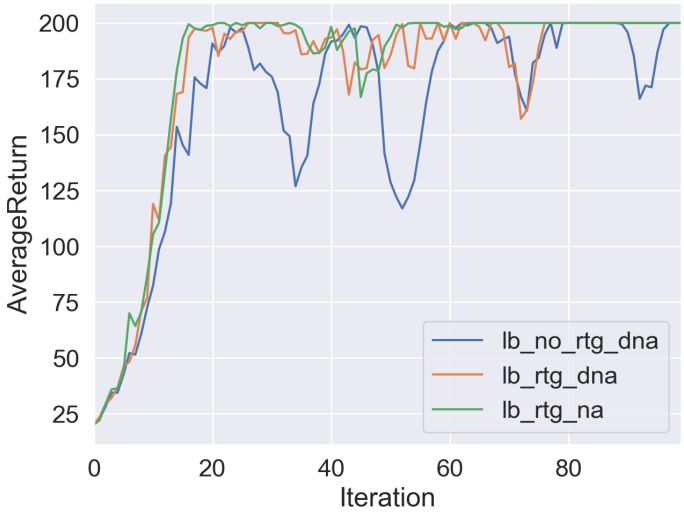

Figure 2: Comparation between different learning curves under large batch sizes

2.3 answer the following questions:

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

A:From Figure 1, we could clearly find out that the policy using reward to go with out advantage-centering has a better performance compared with the trajectory-centric one under small batch sizes. But the situation is opposite when the batch sizes is large.

Q:Did advantage centering help? A:It depends, when the batch sizes is large, it does help. However, when the batch sizes is small, it has negative impacts.

Q:Did the batch size make an impact? A:From the Figure 1 and Figure 2, when using reward to go policy, the increase of the batch size would apparently increase the average return, while the policy with out return to go can results in worse average return.

3 InvertedPendulum

In order to determine the smallest batch size b* and largest learning rate r*, I tried different parameters:

b = 50,100,500,1000,2000,5000,10000,20000

rl = 1e-3, 2e-3, 5e-3, 1e-2, 5e-3, 1e-2, 5e-2, 1e-1

After the experiments, I determined the r* = 5e-2, b* = 1000, the following figure shows the reason.

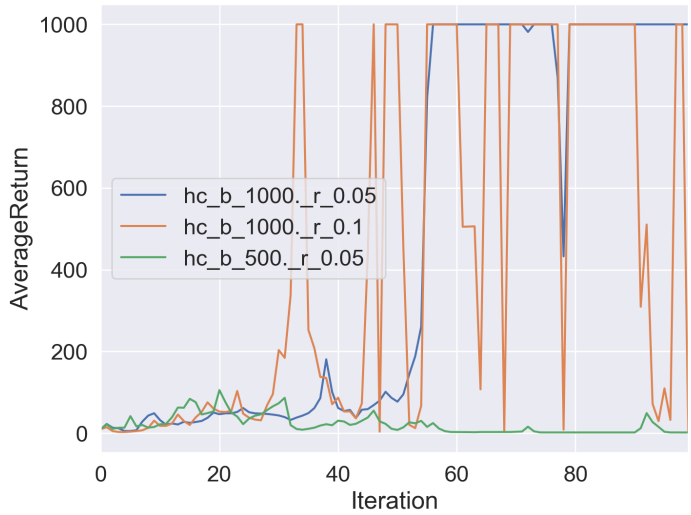

Figure 3: Comparations between different batch sizes and learning rates to determine the minimum batch size and the maximum learning rate

Besides, I also tried to determine which parameters would converge in the less iterations. I tried different parameters and got the results as follow which shows the best parameter is 20000 batch size with learning rate equals to 0.01:

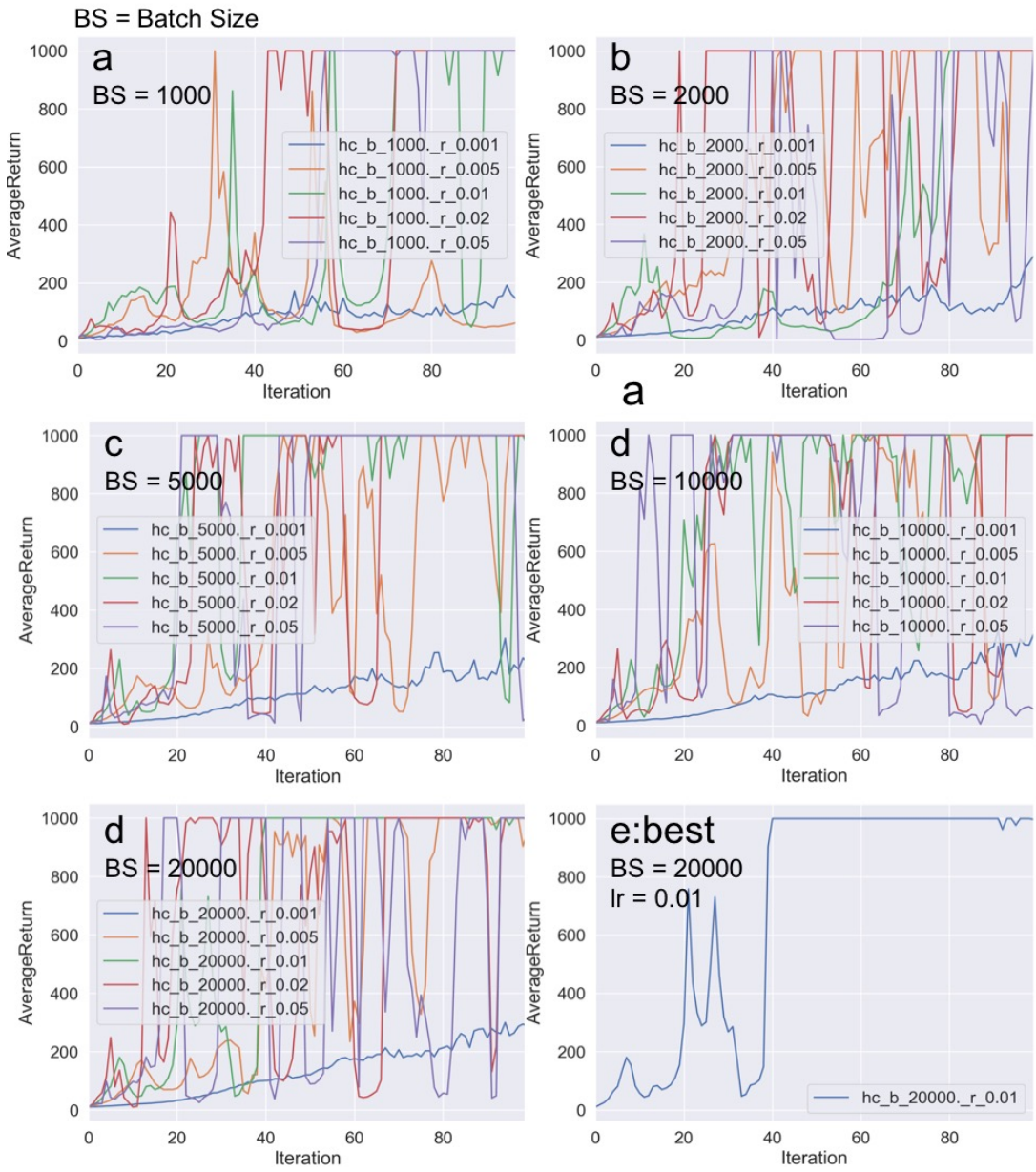

Figure 4: Comparations between different batch sizes and learning rates

4 Problem 7: LunarLander

The experiments could achieve a average return close to 180:

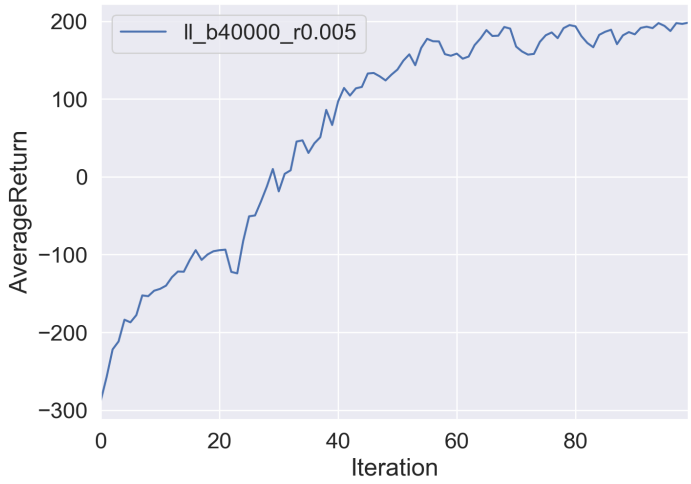

Figure 5: Learning curve of the Lunar Lander task

5 Problem 8: HalfCheetah

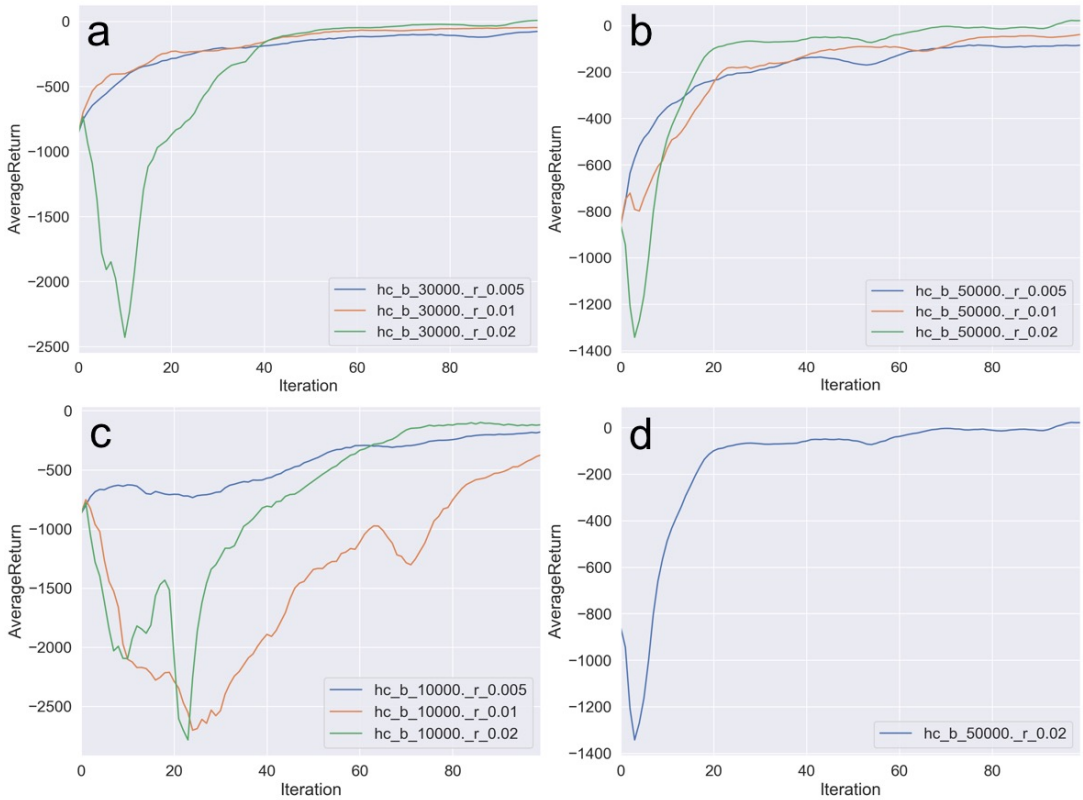

Figure 6: Learning curve under different parameters

From the results, we could clearly see that batch sizes and learning rate do indeed affect the performance. The bigger batch sizes would get better results and become more smooth. As for the learning rate, the larger learning rater would have a negative affect at beginning but result in higer reward after a certain number of iterations. The best parameters I chose is batch size = 50000, while learning rate = 0.02

Using these parameters, we could get the results:

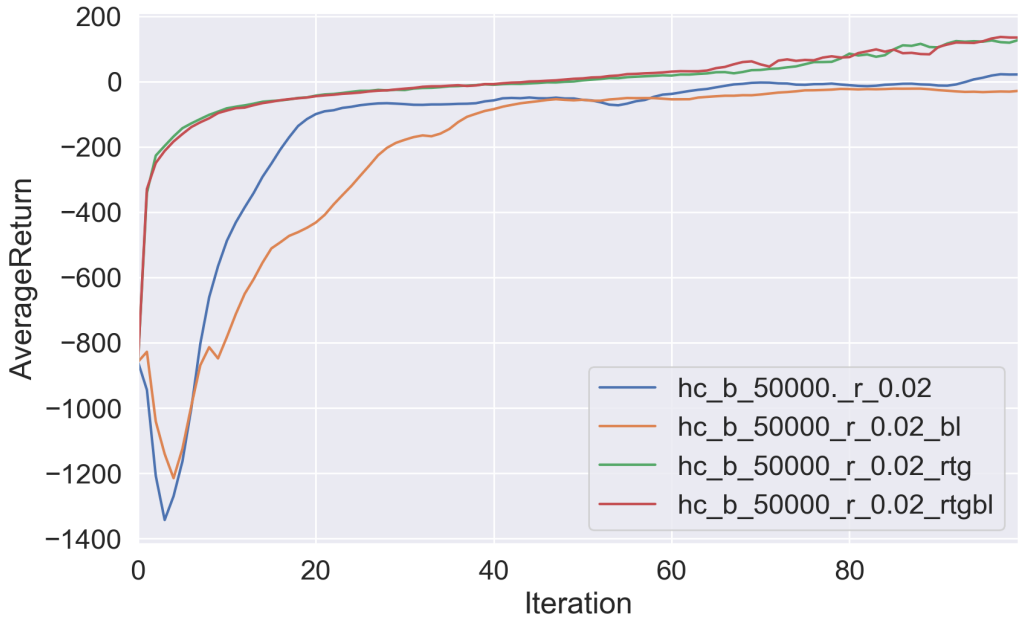

Figure 7: Learning curve under different conditions

When applying "return to go" method in our algorithm, we could get the better results close to 200. (about 140)

6 Bonus:Multi-gradient decent

Here, we try to use Multi-gradient decent to perform the policy gradient decent. for the same batch of data, we take two gradient decent step. Besides, in order to accelerate the computation process, we use multi-threads to run the tensorflow session. The results are as follow:

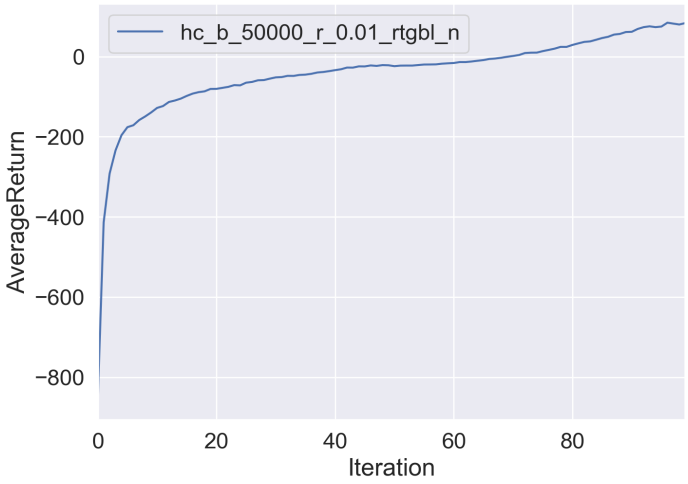

Figure 8: Reward curve under single gradient decent

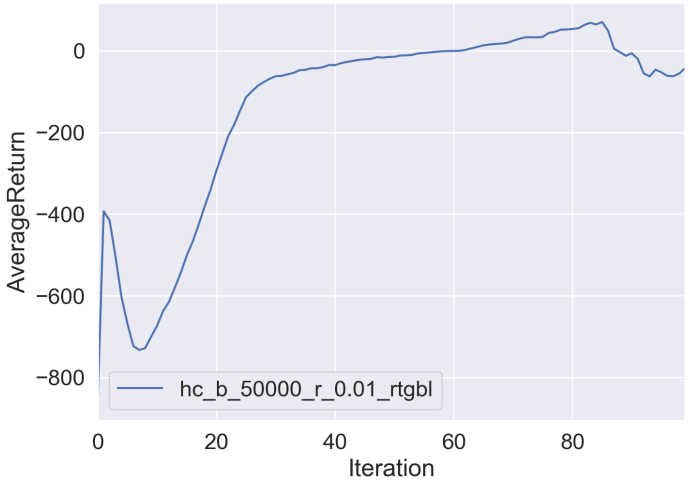

Figure 9: Reward curve under multiple gradient decent

From the results, in these parameters, multiple gradient decent method would not get a better result compared to single gradient decent. As for the processing time, the original time is 2.02e3 seconds, which has been shorten to 1.60e3 seconds.