Berekeley CS 294-112: Deep Reinforcement Learning Homework 2

John Dang

1 State-Dependent Baseline is Unbiased

1.1 Part A

We wish to show that subtracting a state-dependent baseline $b(s_t)$ results in the same policy gradient in expectation:

$$E_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(\left(\sum_{t'=1}^{T} r(s_{t'}, a_{t'}) \right) - b(s_t) \right) \right]$$

$$= E_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(\sum_{t'=1}^{T} r(s_{t'}, a_{t'}) \right) \right]$$

By linearity of expectation, this equality is true only if

$$\sum_{t=1}^{T} E_{(s_t,a_t) \sim p_\theta(s_t,a_t)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t) \right] = 0$$

We can equivalently show this by showing that the expectation term is zero for all timesteps using the law of iterated expectations

$$E_{(s_t,a_t) \sim p_\theta(s_t,a_t)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t) \right]$$

$$E_{s_t \sim p_(s_t)} \left[E_{a_t \sim \pi_\theta(a_t|s_t)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t) | \, s_t \right] \right]$$

$$\int_{s_t} p(s_t) b(s_t) \int_{a_t} \pi_\theta(a_t|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) da_t ds_t$$

$$\int_{s_t} p(s_t) b(s_t) \int_{a_t} \pi_\theta(a_t|s_t) \frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)} da_t ds_t$$

$$\int_{s_t} p(s_t) b(s_t) \nabla_\theta \int_{a_t} \pi_\theta(a_t|s_t) da_t ds_t$$

$$\int_{s_t} p(s_t) b(s_t) \nabla_\theta 1 ds_t = 0$$

Thus, subtracting a state-dependent baseline from the reward results in the same policy gradient in expectations and is unbiased.

1.2 Part B

1.2.1 Part a

The reinforcement learning problem can be formulated as a markov chain of state-action tuples (s_t, a_t), which satisfy the Markov Property:

$$p(s_{t+1}, a_{t+1}|s_t, a_t) = p(s_{t+1}, a_{t+1}|s_t, a_t, \ldots, s_0, a_0)$$

Thus, the trajectory after timestep t is dependent only on (s_t, a_t) (independent of what happened before timestep t).

1.2.2 Part b

The trajectory distribution $p_\theta(\tau)$ can be expressed in terms of the trajectory before and after time step t.

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

We can show

$$E_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t)\right] = 0$$

by breaking up the expectation into an expectation under the trajectory before and after time step t. Let $\tau_{t:t'} = (s_t, a_t, ..., s_{t'}, a_{t'})$ be the trajectory between time steps t and t'.

$$E_{\tau_{1:t} \sim p_\theta(\tau_{1:t})} \left[E_{\tau_{t:T} \sim p_\theta(\tau_{t:T}|\tau_{1:t})} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t)|\, \tau_{1:t}\right]\right]$$

The inner expectation evaluates to

$$\int_{s_t} b(s_t) p(s_t|s_{t-1}, a_{t-1}) \int_{a_t} \pi_\theta(a_t|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \int_{s_{t+1}} p(s_{t+1}|s_t, a_t) \int_{a_{t+1}} \pi_\theta(a_{t+1}|s_{t+1}) \ldots \int_{s_T} p(s_T|s_{t-1}, a_{t-1}) ds_T ds_{t+1} da_{t+1} da_t ds_t$$

$$\int_{s_t} b(s_t) p(s_t|s_{t-1}, a_{t-1}) \int_{a_t} \pi_\theta(a_t|s_t) \frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)} \cdot 1 da_t ds_t$$

$$\int_{s_t} b(s_t) p(s_t|s_{t-1}, a_{t-1}) \nabla_\theta 1 ds_t = 0$$

Thus, have the policy gradient element at time step t is

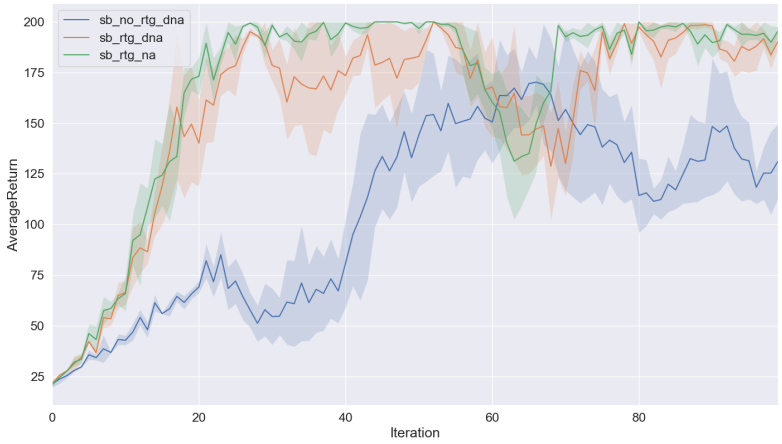
$$E_{\tau_{1:t} \sim p_\theta(\tau_{1:t})} [0] = 0$$

Again, we have shown that subtracting a state-dependent baseline is unbiased in expectation.

2 Experiments

2.1 Vanilla Policy Gradient

2.1.1 Small Batch Performance (CartPole-v0)



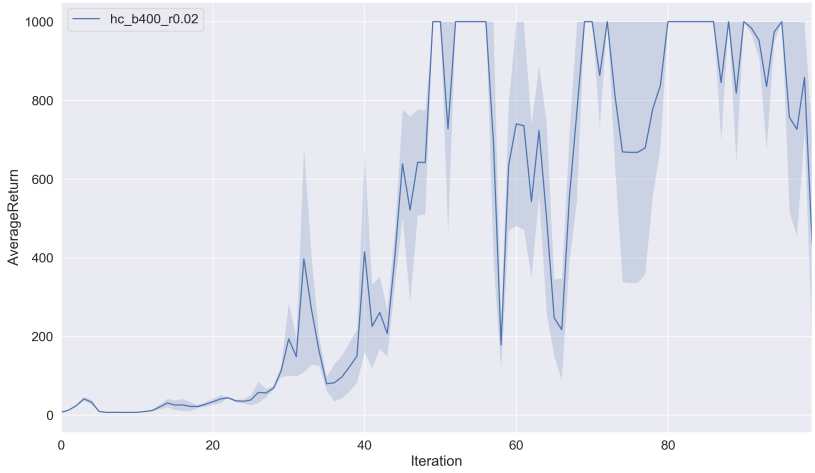
2.1.2 Large Batch Performance (CartPole-v0)

2.1.3 Discussion

Using reward to go inmproved performance and reduced variance for both small and large batch experiments. Advantage-centering does improve performance. Using a larger batch size reduces varaince and speeds up learning for the first few iterations. Both batch sizes were able to converge to near optimal performance, with the exception of the experiment with no variance reduction techniques. In this case, the larger batch size improved performance and reduced variance significantly.

2.2 Hyperparameter Tuning (InvertedPendulum-v2)

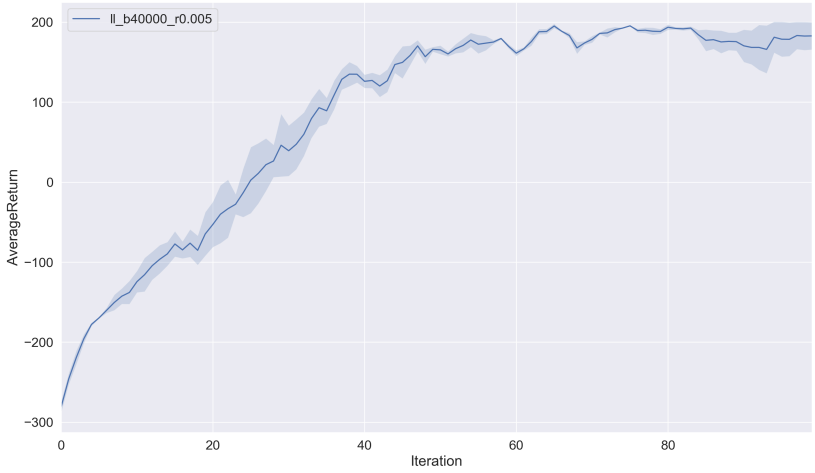
Using a batch size of 400 and learning rate of 0.02, resulted in reaching optimal performance, while minimizing batch size and maximizing learning rate. These hyperparameter values were found by grid search over batch sizes $[300, 400, 500]$ and learning rates $[0.03, 0.02, 0.01]$.



2.3 Policy Gradient with Baselines

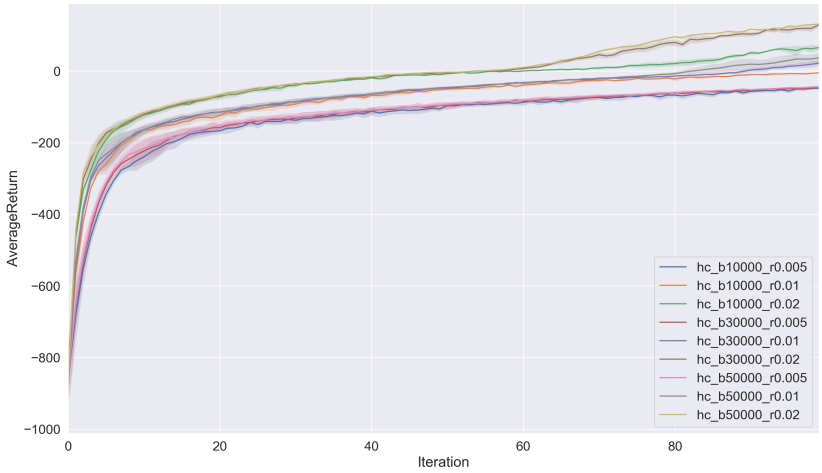
2.3.1 Performance (LunarLander-v2)

As expected, using policy gradients with a neural network baseline was able to achieve desired performance.



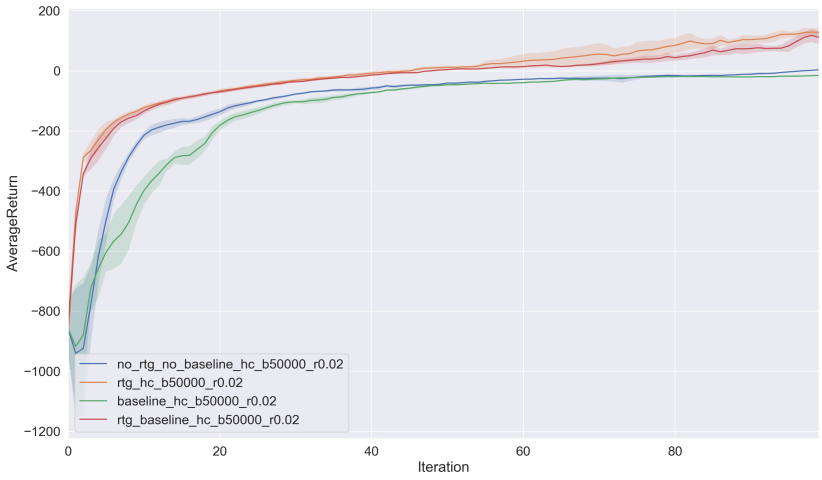
2.3.2 Hyperparameter Tuning (HalfCheetah-v2)

In this experiment, using a higher learning rate results in better performance. Increasing batch size also generally improved performance or at least maintained performance. Based on this experiment we run the remaining experiments with batch size 50000 and learning rate 0.02.



2.3.3 Performance (HalfCheetah-v2)

As shown before using reward to go achieves better performance and lower variance. Using a baseline with reward to go also achieves better performance and significantlt reduces varaince. Using a baseline alone without reward to go performs poorly.



3 Scripts Used

To reproduce these results run the following commands from the hw2 directory.

```
./p4.sh
./p5.sh
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
./p8HyperParamSearch.sh
./p8.sh
```