Assignment 2: Policy Gradients

Thanapat Trachu

January 15^{st} 2023

Abstract

The goal of this assignment is to experiment with policy gradient and its variants, including variance reduction tricks such as implementing reward-to-go and neural network baselines.

1 CartPole Experiment

Run multiple experiments with the PG algorithm on the discrete CartPole-v0 environment, using the following commands:

```
python cs285/scripts/run_hw2.py —env_name CartPole–v0 –n 100 –b 1000 \
    –dsa —exp_name q1_sb_no_rtg_dsa
python cs285/scripts/run_hw2.py —env_name CartPole–v0 –n 100 –b 1000 \
    –rtg –dsa —exp_name q1_sb_rtg_dsa
python cs285/scripts/run_hw2.py —env_name CartPole–v0 –n 100 –b 1000 \
    –rtg —exp_name q1_sb_rtg_na
python cs285/scripts/run_hw2.py —env_name CartPole–v0 –n 100 –b 5000 \
    –dsa —exp_name q1_lb_no_rtg_dsa
python cs285/scripts/run_hw2.py —env_name CartPole–v0 –n 100 –b 5000 \
    –rtg –dsa —exp_name q1_lb_rtg_dsa
python cs285/scripts/run_hw2.py —env_name CartPole–v0 –n 100 –b 5000 \
    –rtg —exp_name q1_lb_rtg_na
```

Question 1.1: Learning Curve Graph

Small batch learning curve

compare the learning curves (average return at each iteration) for the experiments prefixed with q1_sb_. (The small batch experiments.)

Large batch learning curve

compare the learning curves for the experiments prefixed with q1_lb_. (The large batch experiments.)

Question 1.2: Answer Question

Which value estimator has better performance without advantage-standardization: the trajectory-centric one, or the one using reward-to-go?

The model with reward to go is better than trajectory-centric, because it has lower variance. Even if it may have more bias, however it still surpasses the trajectory-centric. In my aspect, the reward-to-go balance the variance and bias better than the trajectory-centric.

Did advantage standardization help?

The advantage standardization does not help significantly. In addition, the model, which does not use advantage standardization, is more stable.

Did the batch size make an impact?

As increasing batch size, it requires less time-step to converge. Moreover, the large batch setting is more stable than the small batch setting. It can be view as the larger batch size setting make the model to see more data at the same time-step.

Question 1.3: Command Line

In order to replicate the experiment, please run the same command as the above (does not change any parameters from the provided command). The graph can be generated by running the following command `tensorboard --logdir data`. The learning curve graph is a Train_AverageReturn graph.

2 Inverted Pendulum Experiment

Run experiments on the InvertedPendulum-v4 continuous control environment as follows:

```
python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v4 \
    --ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b <b*> -lr <r*> -rtg \
    --exp_name q2_b<b*>_r<r*>
```

Question 2.1: Finding batch size and learning rate

The suitable batch size and learning rate for this experiment is 1000 and 0.005, respectively. The following graph is a learning of the given setting.

Question 2.2: Command Line

Please running the following command to replicate the experiment:

```
python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v4 \
    --ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 1000 -lr 0.005 -rtg \
    --exp_name q2_b1000_r005
```

3 Lunar Lander Experiment

You will now use your policy gradient implementation to learn a controller for LunarLanderContinuous-v2. The purpose of this problem is to test and help you debug your baseline implementation from Section 6. Run the following command:

```
python cs285/scripts/run_hw2.py \
    --env_name LunarLanderContinuous-v2 --ep_len 1000 \
    --discount 0.99 -n 100 -l 2 -s 64 -b 40000 -lr 0.005 \
    --reward_to_go --nn_baseline --exp_name q3_b40000_r0.005
```

Question 3.1: Learning Curve Graph

The following graph is a learning curve of this experiment.

4 Half Cheetah Experiment

Question 4.1: Searching Batch Size and Learning Rate

The effect of batch size is very similar to the first experiment. In addition, it will require more resource and a lot of time to run a larger batch. For the learning rate, the larger the learning rate is, the faster the model converges. Therefore, the best setting is learning rate = 0.02 and batch size = 50000. However for an convenient, we reduce the batch size to 30000 because the difference between this two batch size is not significant.

Question 4.2: Learning Curve Graph

The following graph is a learning curve of this experiment.

5 Hopper Experiment

You will now use your implementation of policy gradient with generalized advantage estimation to learn a controller for a version of Hopper-v4 with noisy actions. Search over $\lambda \in [0,0.95,0.98,0.99,1]$ to replace ¡λ¿ below. Note that with a correct implementation, $\lambda = 1$ is equivalent to the vanilla neural network baseline estimator. Do not change any of the other hyperparameters (e.g. batch size, learning rate).

```
python cs285/scripts/run_hw2.py \
    --env_name Hopper-v4 --ep_len 1000 \
    --discount 0.99 -n 300 -l 2 -s 32 -b 2000 -lr 0.001 \
    --reward_to_go --nn_baseline --action_noise_std 0.5 --gae_lambda <lambda> \
    --exp_name q5_b2000_r0.001_lambda<lambda>
```

Question 5.1: Learning Curve Graph

The GAE lambda is used to control bias and variance. As we increase the lambda, the variance of the model is also increase and the bias is reduced, vice and versa.

According the equation, if lambda is equal to 1, the model is equivalent to the vanilla neural network baseline estimator. In addition, if we set the lambda to 0, the advantage is equivalent to $r(s_t, a_t) + \gamma\hat{V}(s_{t+1}) - \hat{V}(s_t)$. Therefore, it have less variance and high bias (underfitting).