CS 285 Homework 2

Neo Yu Yao Terence

# Experiment 1

## Learning curves for small batch experiments q1\_sb\_

Graphical user interface, chart

Description automatically generated

## Learning curves for large batch experiments q1\_lb\_

A picture containing chart

Description automatically generated

## Questions

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

Without advantage-standardization (-dsa flag present), the reward-to-go value estimator (red and grey) has better performance as compared to (blue and green) for both small batch (red) and large batch (grey) experiments.

**Did advantage standardization help?**

Yes, with the same batch size and value estimator, the experiments with advantage standardization (orange) converges faster than the experiments without advantage standardization (grey and red).

**Did the batch size make an impact?**

Yes. Larger batch size helps the model converge faster and with higher performance.

## Commands Used

echo running q1\_sb\_no\_rtg\_dsa...

python cs285/scripts/run\_hw2.py --env\_name CartPole-v0 -n 100 -b 1000 \

-dsa --exp\_name q1\_sb\_no\_rtg\_dsa

echo running q1\_sb\_rtg\_dsa...

python cs285/scripts/run\_hw2.py --env\_name CartPole-v0 -n 100 -b 1000 \

-rtg -dsa --exp\_name q1\_sb\_rtg\_dsa

echo running q1\_sb\_rtg\_na...

python cs285/scripts/run\_hw2.py --env\_name CartPole-v0 -n 100 -b 1000 \

-rtg --exp\_name q1\_sb\_rtg\_na

echo running q1\_lb\_no\_rtg\_dsa...

python cs285/scripts/run\_hw2.py --env\_name CartPole-v0 -n 100 -b 5000 \

-dsa --exp\_name q1\_lb\_no\_rtg\_dsa

echo running q1\_lb\_rtg\_dsa...

python cs285/scripts/run\_hw2.py --env\_name CartPole-v0 -n 100 -b 5000 \

-rtg -dsa --exp\_name q1\_lb\_rtg\_dsa

echo running q1\_lb\_rtg\_na...

python cs285/scripts/run\_hw2.py --env\_name CartPole-v0 -n 100 -b 5000 \

-rtg --exp\_name q1\_lb\_rtg\_na

# Experiment 2

## Values found

From trying many different variations, the smallest batch size and largest learning rate found:

|  |  |
| --- | --- |
| Hyperparameter | Value |
| Batch size b\* | 149 |
| Learning rate r\* | 0.05 |

As shown in the chart below, we are able to hit 1,000 AverageReturn before 100 iterations.

Chart, line chart

Description automatically generated

## Commands Used

python cs285/scripts/run\_hw2.py --env\_name InvertedPendulum-v2 \

--ep\_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 149 -lr 0.05 -rtg \

--exp\_name q2\_b149\_r0.05

# Experiment 3

## Learning Curve

Chart, line chart

Description automatically generated

## Commands Used

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

# Experiment 4

## Learning curve for searching, plot optimal values of b\* and r

Chart, line chart

Description automatically generated

Increasing batch size improves performance slightly, and increasing the learning rate improves performance significantly.

## Commands Used for optimal values

python cs285/scripts/run\_hw2.py --env\_name HalfCheetah-v4 --ep\_len 150 \

--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 -rtg --nn\_baseline \

--exp\_name q4\_search\_b50000\_lr0.02\_rtg\_nnbaseline

## Learning curve with optimal values of b\* and r\*

Chart, line chart

Description automatically generated

## Commands Used

echo running experiment 4 HalfCheetah with optimal values -b 50000 -lr 0.02...

python cs285/scripts/run\_hw2.py --env\_name HalfCheetah-v4 --ep\_len 150 \

--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 \

--exp\_name q4\_b50000\_r0.02

python cs285/scripts/run\_hw2.py --env\_name HalfCheetah-v4 --ep\_len 150 \

--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 -rtg \

--exp\_name q4\_b50000\_r0.02\_rtg

python cs285/scripts/run\_hw2.py --env\_name HalfCheetah-v4 --ep\_len 150 \

--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 --nn\_baseline \

--exp\_name q4\_b50000\_r0.02\_nnbaseline

python cs285/scripts/run\_hw2.py --env\_name HalfCheetah-v4 --ep\_len 150 \

--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 -rtg --nn\_baseline \

--exp\_name q4\_b50000\_r0.02\_rtg\_nnbaseline

# Experiment 5

## Learning curves for Hopper-v2 experiments

Graphical user interface, chart

Description automatically generated

Performance increased as increases from 0.95 to 1. However, at the model starts off with very low performance initially, but shoots up after more iterations as shown by the blue graph above.

## Commands Used

echo running experiment 5...

python cs285/scripts/run\_hw2.py \

--env\_name Hopper-v2 --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 0 \

--exp\_name q5\_b2000\_r0.001\_lambda0

python cs285/scripts/run\_hw2.py \

--env\_name Hopper-v2 --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 0.95 \

--exp\_name q5\_b2000\_r0.001\_lambda0.95

python cs285/scripts/run\_hw2.py \

--env\_name Hopper-v2 --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 0.99 \

--exp\_name q5\_b2000\_r0.001\_lambda0.99

python cs285/scripts/run\_hw2.py \

--env\_name Hopper-v2 --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 1 \

--exp\_name q5\_b2000\_r0.001\_lambda1