**CS285 Homework 2**

Fall 2021

Cuiqianhe Du

**Experiment 1: CartPole**

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

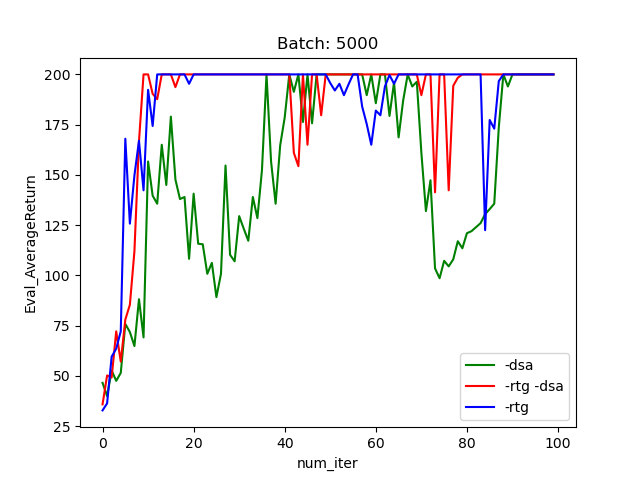
python cs285/scripts/run\_hw2.py --env\_name CartPole-v0 -n 100 -b 5000 -rtg -dsa --exp\_name q1\_lb\_rtg\_dsa

python cs285/scripts/run\_hw2.py --env\_name CartPole-v0 -n 100 -b 5000 -rtg --exp\_name q1\_lb\_rtg\_na

By comparing the results, I find out that the ones with Reward-to-go and advantage standardization performs better. Furthermore, larger batch size will learn faster when comparing at the same number of iterations.

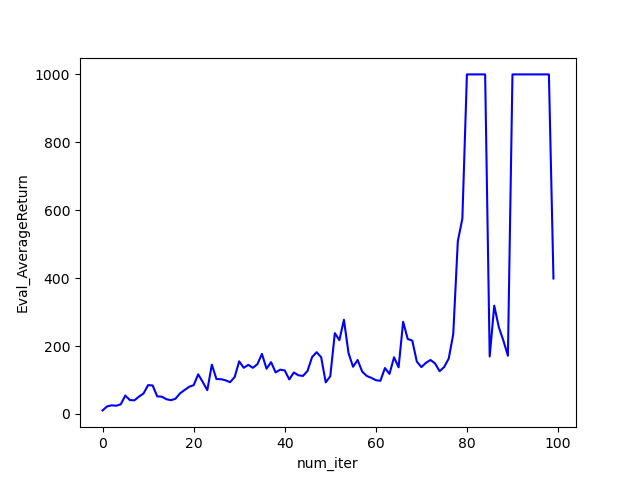
Chart, line chart, histogram

Description automatically generated



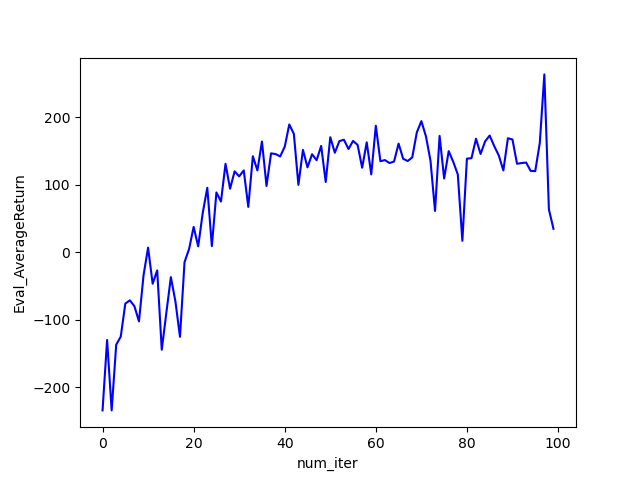
**Experiment 2: finding smallest batch size b\* and largest learning rate r\***

python cs285/scripts/run\_hw2.py --env\_name InvertedPendulum-v2 --ep\_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 300 -lr 1e-2 -rtg --exp\_name q2\_b300\_r1e-2



**Experiment 3: LunarLander**

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: HalfCheetah**

python cs285/scripts/run\_hw2.py --env\_name HalfCheetah-v2 --ep\_len 150 --discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.005 -rtg --nn\_baseline --exp\_name q4\_search\_b10000\_lr0.005\_rtg\_nnbaseline

python cs285/scripts/run\_hw2.py --env\_name HalfCheetah-v2 --ep\_len 150 --discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.005 -rtg --nn\_baseline --exp\_name q4\_search\_b30000\_lr0.005\_rtg\_nnbaseline

python cs285/scripts/run\_hw2.py --env\_name HalfCheetah-v2 --ep\_len 150 --discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.005 -rtg --nn\_baseline --exp\_name q4\_search\_b50000\_lr0.005\_rtg\_nnbaseline

python cs285/scripts/run\_hw2.py --env\_name HalfCheetah-v2 --ep\_len 150 --discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.01 -rtg --nn\_baseline --exp\_name q4\_search\_b10000\_lr0.01\_rtg\_nnbaseline

python cs285/scripts/run\_hw2.py --env\_name HalfCheetah-v2 --ep\_len 150 --discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.01 -rtg --nn\_baseline --exp\_name q4\_search\_b30000\_lr0.01\_rtg\_nnbaseline

python cs285/scripts/run\_hw2.py --env\_name HalfCheetah-v2 --ep\_len 150 --discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.01 -rtg --nn\_baseline --exp\_name q4\_search\_b50000\_lr0.01\_rtg\_nnbaseline

python cs285/scripts/run\_hw2.py --env\_name HalfCheetah-v2 --ep\_len 150 --discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.02 -rtg --nn\_baseline --exp\_name q4\_search\_b10000\_lr0.02\_rtg\_nnbaseline

python cs285/scripts/run\_hw2.py --env\_name HalfCheetah-v2 --ep\_len 150 --discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.02 -rtg --nn\_baseline --exp\_name q4\_search\_b30000\_lr0.02\_rtg\_nnbaseline

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

Chart

Description automatically generated

As we can defer from the plot, normally, larger batch size and larger learning rate will let the model learn faster when comparing between same iterations.

The optimal batch size and learning rate is 30000 and 0.02 respectively

python cs285/scripts/run\_hw2.py --env\_name HalfCheetah-v2 --ep\_len 150 --discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.02 --exp\_name q4\_b30000\_r0.02

python cs285/scripts/run\_hw2.py --env\_name HalfCheetah-v2 --ep\_len 150 --discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.02 -rtg --exp\_name q4\_b30000\_r0.02\_rtg

python cs285/scripts/run\_hw2.py --env\_name HalfCheetah-v2 --ep\_len 150 --discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.02 --nn\_baseline --exp\_name q4\_b30000\_r0.02\_nnbaseline

python cs285/scripts/run\_hw2.py --env\_name HalfCheetah-v2 --ep\_len 150 --discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.02 -rtg --nn\_baseline --exp\_name q4\_b30000\_r0.02\_rtg\_nnbaseline

Chart, line chart, histogram

Description automatically generated

**Experiment 5: HopperV2**

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

As we can see from the graph, best performances are achieved when lambda is very close to 1