CS285 Homework 2

Fall 2021

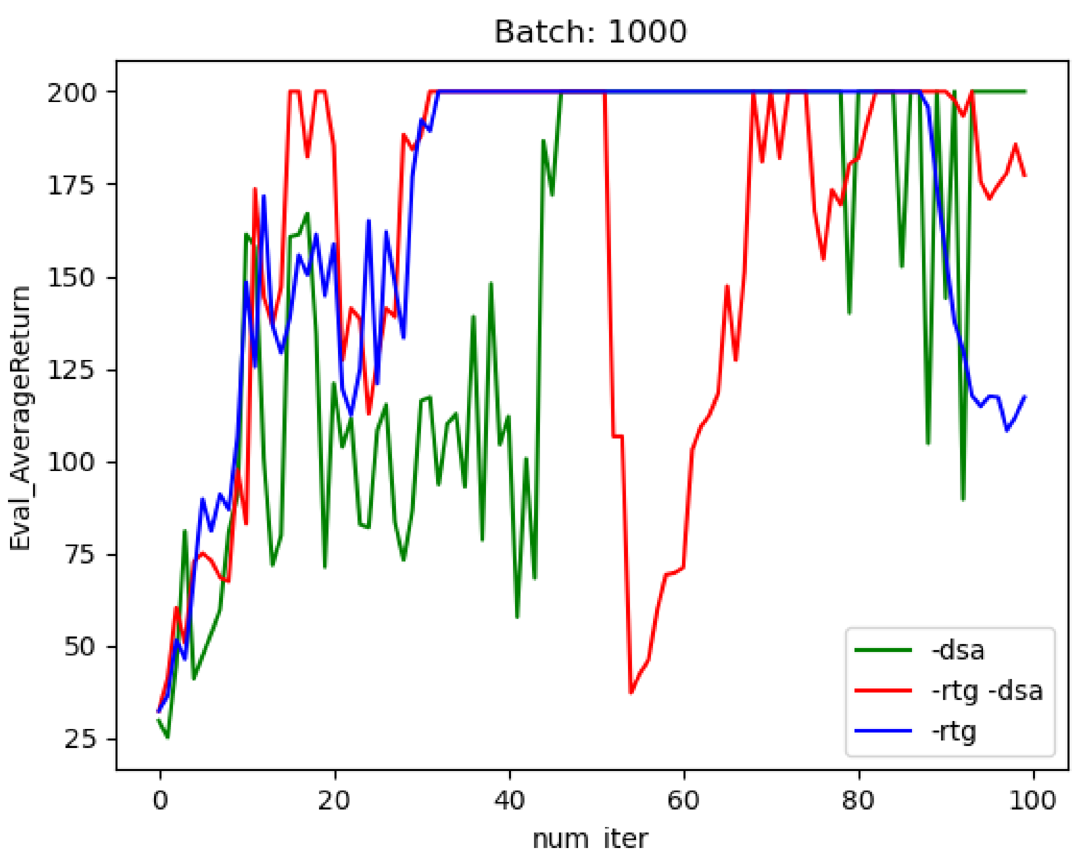
Cuiqianhe Du

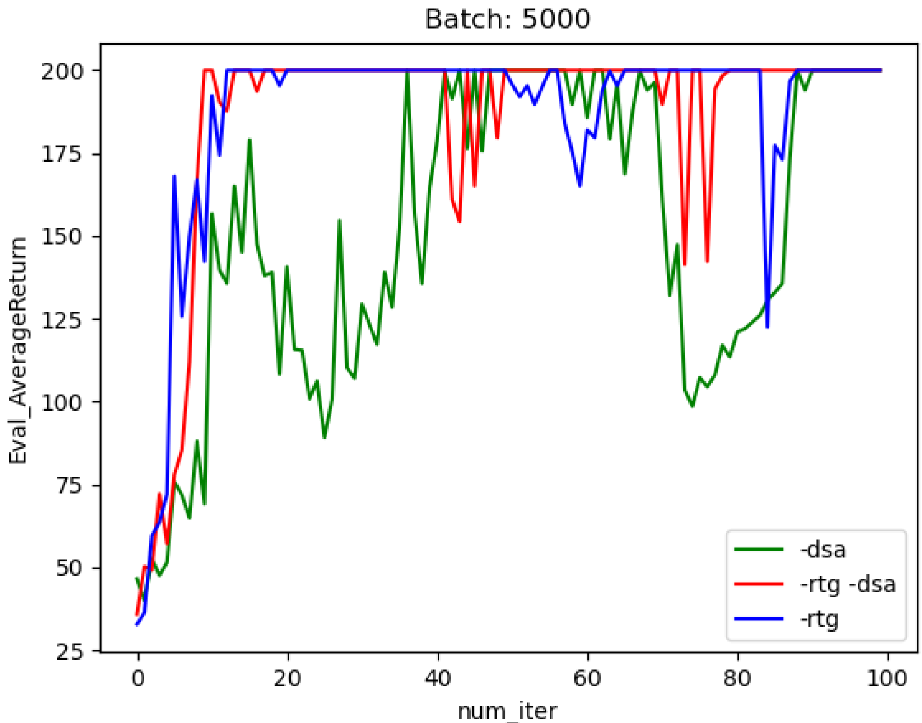
Experiment 1: CartPole

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

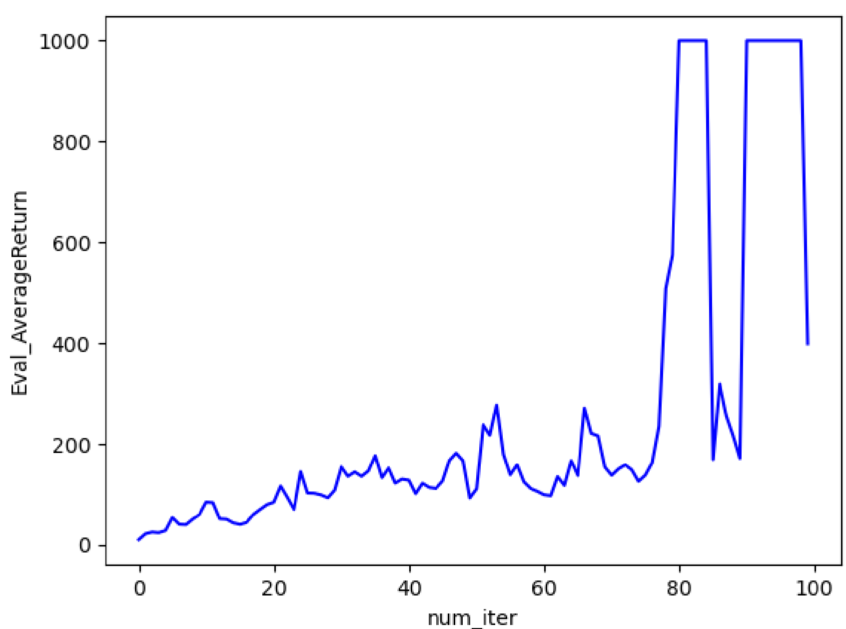
By comparing the results, I find out that the ones with Reward-to-go and advantage standardization performs better. Furthermore, larger batch size will learn faster when comparing at the same number of iterations.





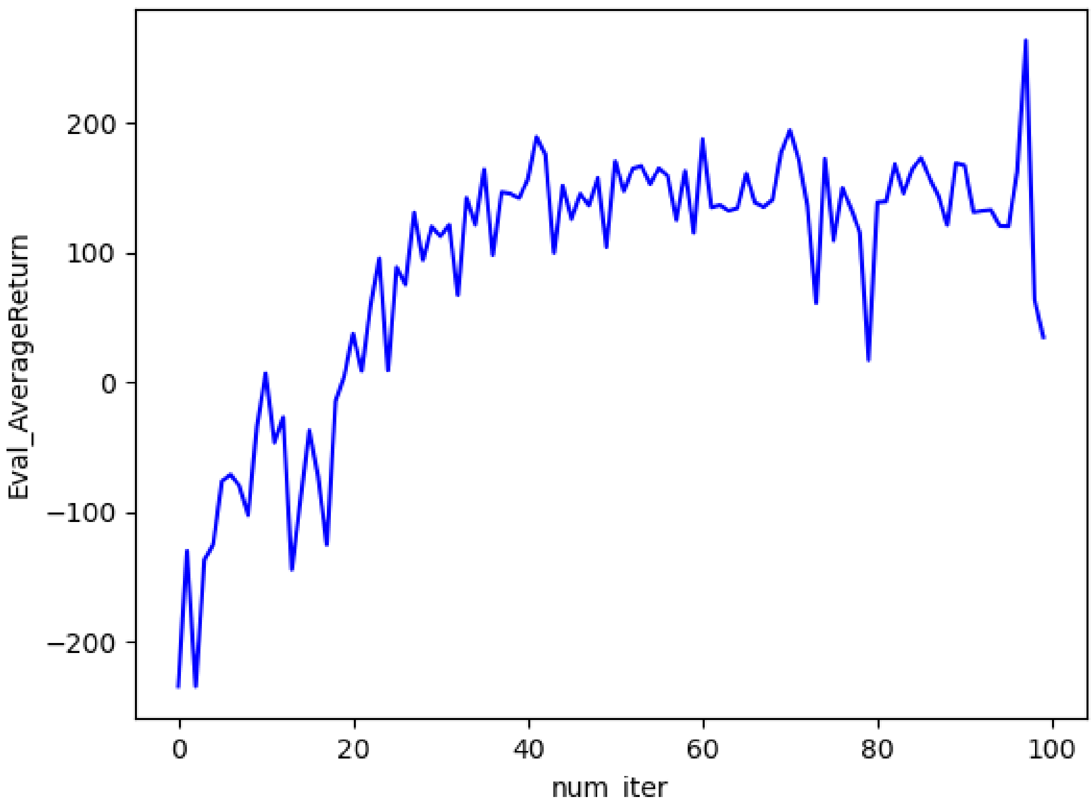
Experiment 2: finding smallest batch size b* and largest learning rate r*

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

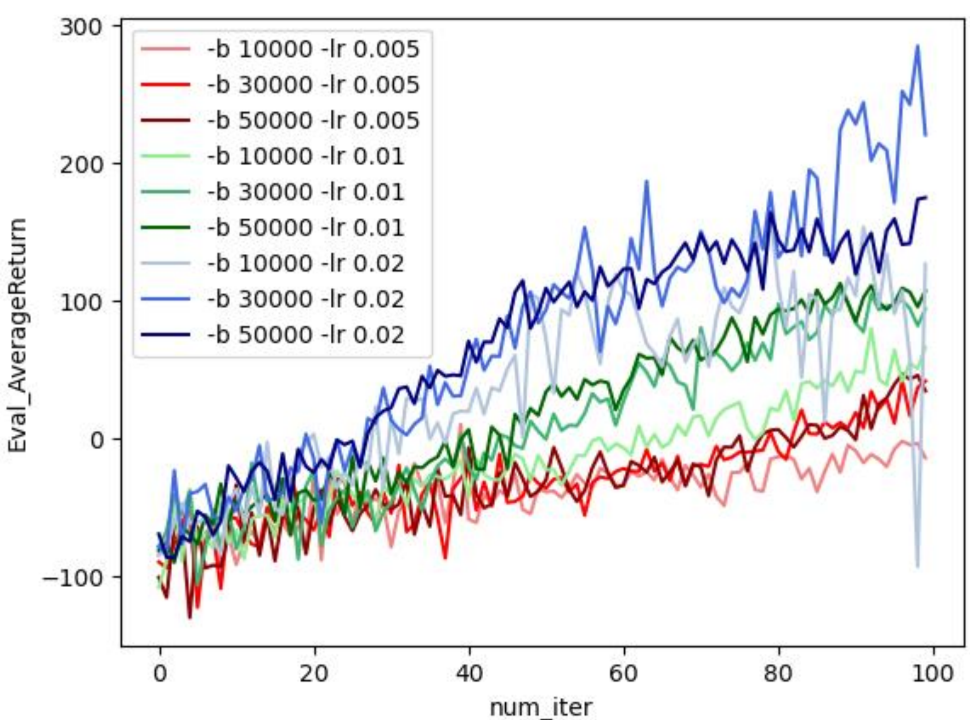
python cs285/scripts/run_hw2.py --env_name HalfCheetah-v2 --ep_len 150 --discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.01 -rtg --nn_baseline --exp_name q4_search_b30000_lr0.01_rtg_nnbaseline

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



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.

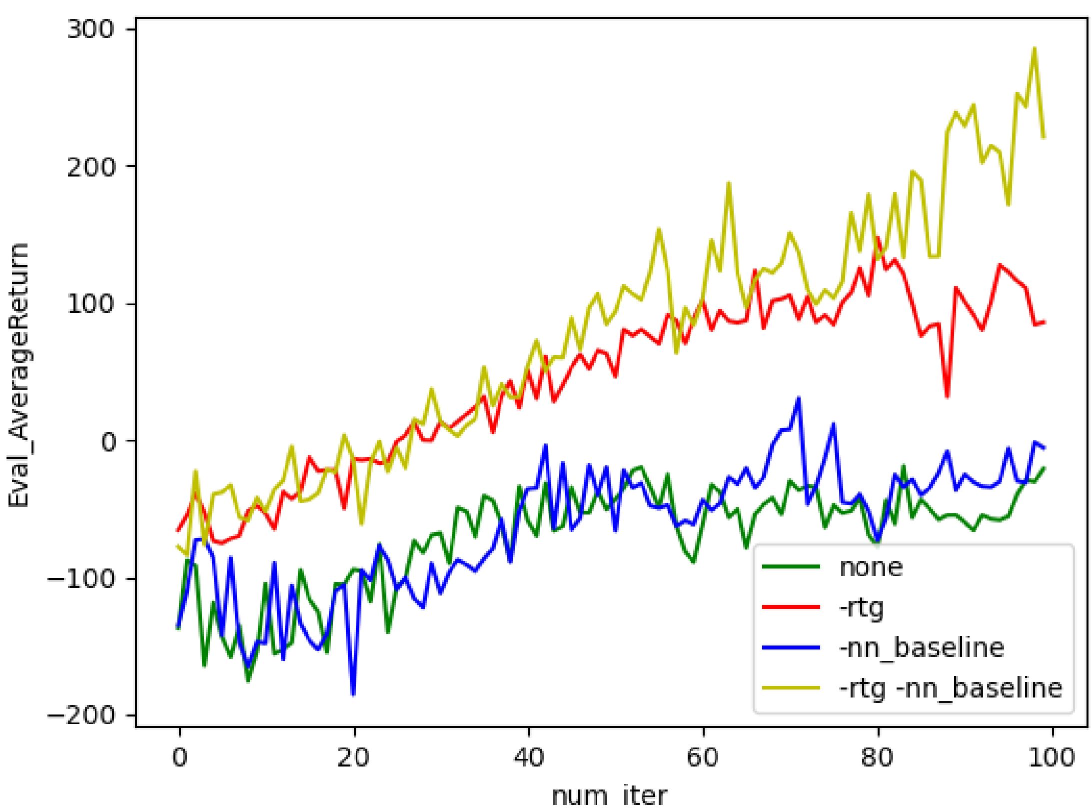
The optimal batch size and learning rate is 30000 and 0.02 respectively

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



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