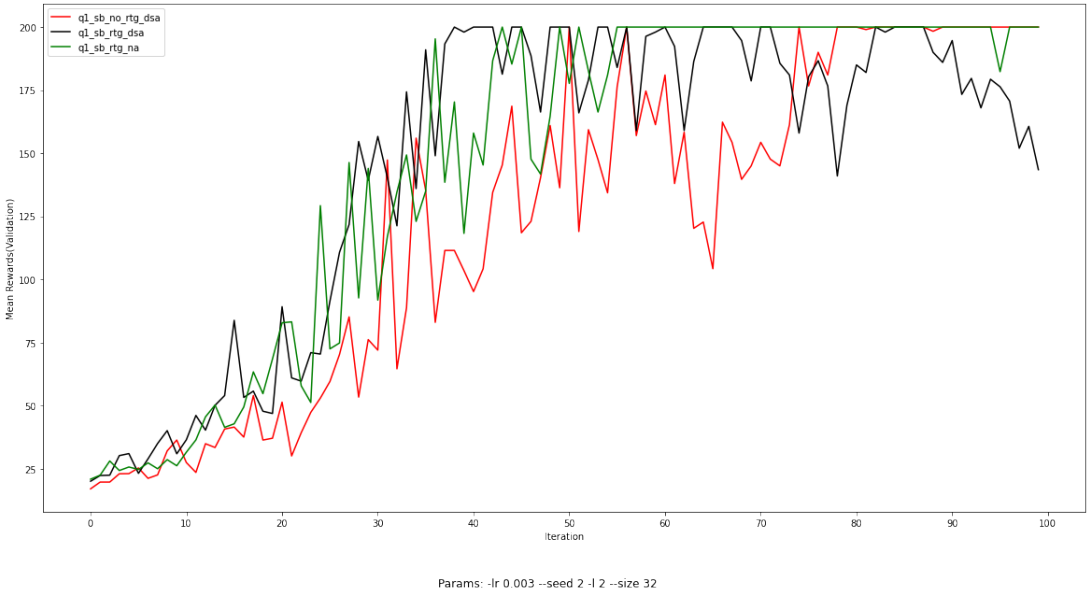
Assignment 2: Policy Gradients

huseyinabanox@gmail.com

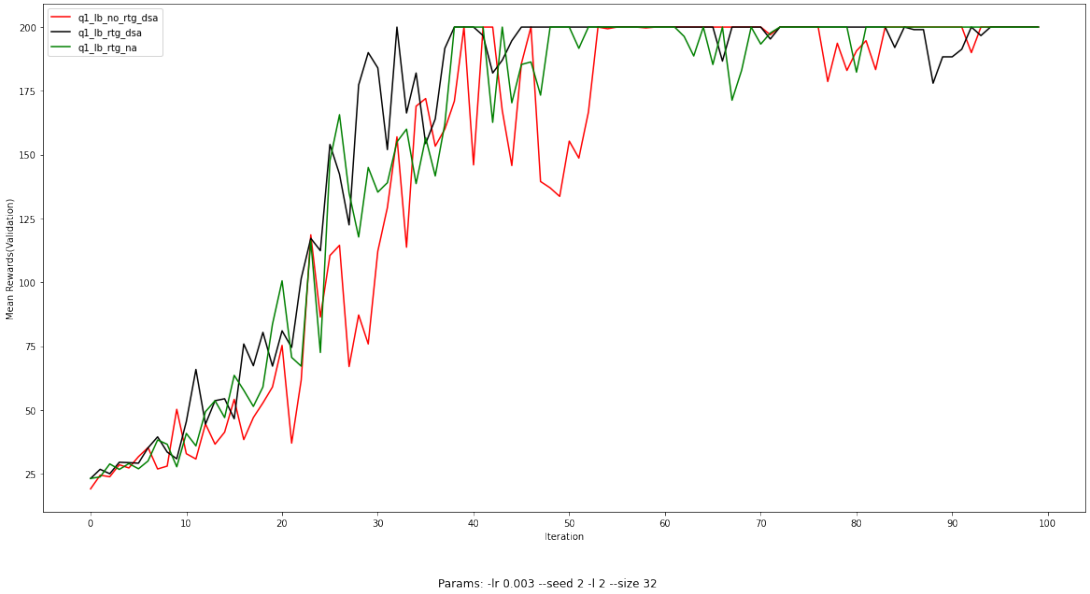
January 2023

1 Small-Scale Experiments

1.1 Experiment 1

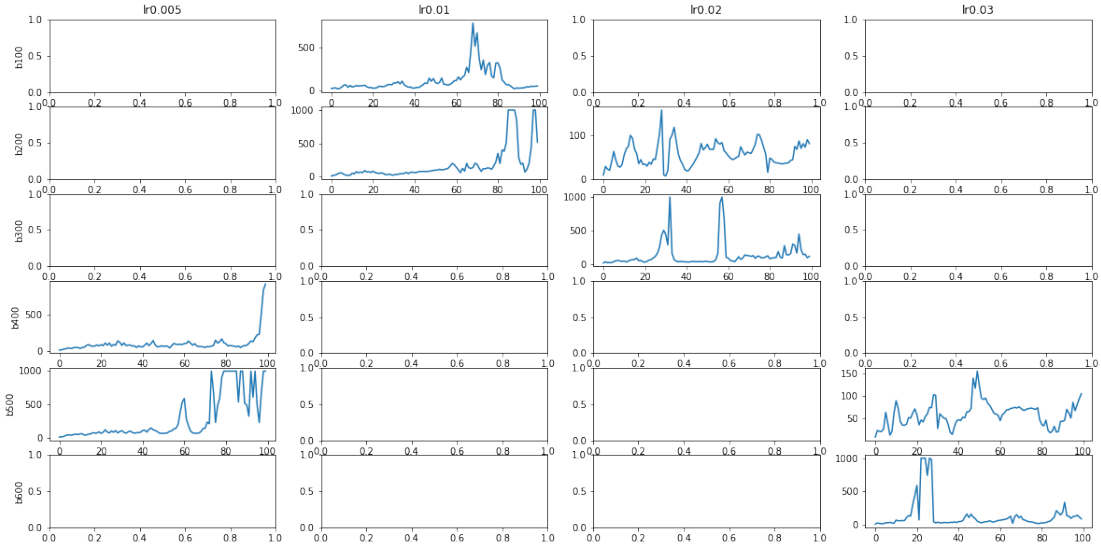


Reward-to-go estimator performs better. Advantage standardization difference is not statistically significant.



Reward-to-go estimator performs better. Advantage standardization difference is not statistically significant. Increasing the batch size helps the trajectory-centric algorithm converge with fewer iterations.

1.2 Experiment 2

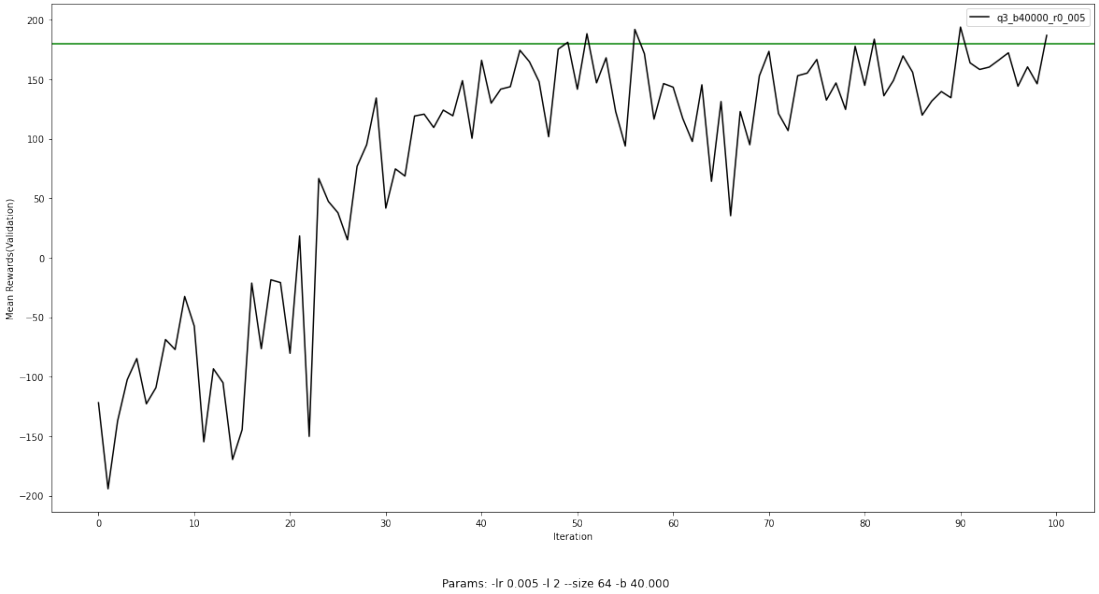


A grid search with default parameters shows that it is possible to reach the target reward of 1000 with learning rate 0.01 and batch size 200 or with

learning rate 0.02 and batch size 300. The first options seems to be more stable.

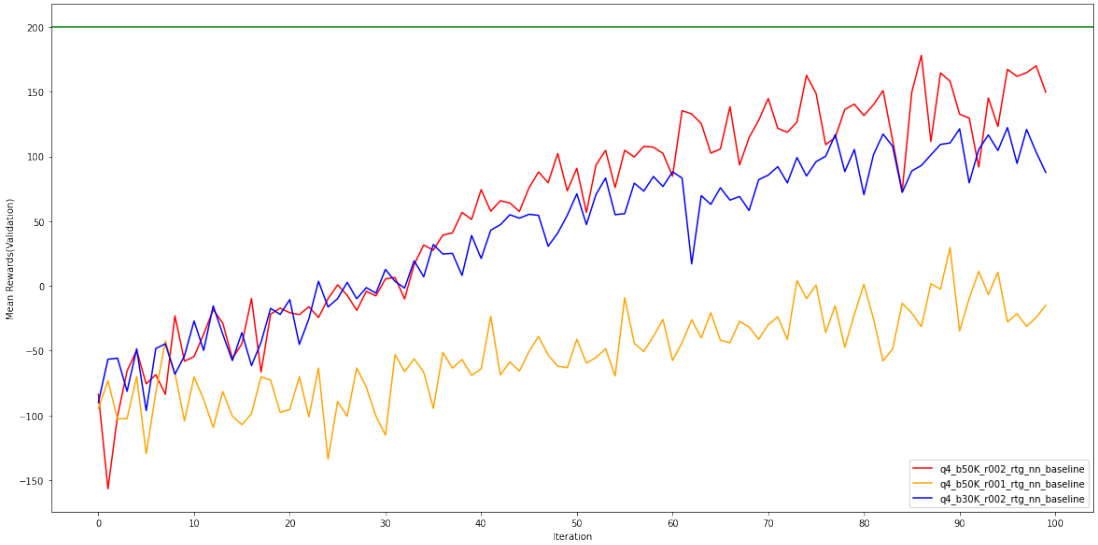
Please note that the figure tries to show the boundary in hyperparameter space. In each column the first chart indicates the largest learning rate and smallest batch size that fails to reach 1000. In each column the second chart indicates the largest learning rate and smallest batch size that succeeds in reaching 1000.

1.3 Experiment 3



1.4 Experiment 4

A trial and error search is run on hyperparameter space. Learning rate 0.02 and batch size 50K are found to be best performing.



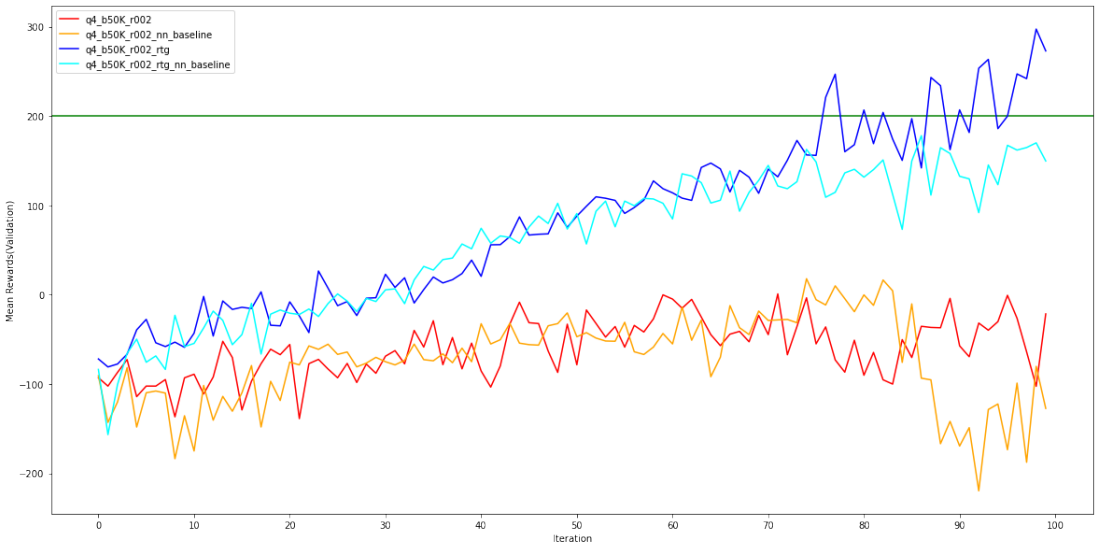
Params: --ep_len 150 --discount 0.95 -n 100 -l 2 -s 32 -rtg --nn_baseline

The chart above shows evaluation results by iteration for different parameter sets.

Bigger batch size is better. It reduces variance and speeds up learning.

Bigger learning rate is better. It speeds up learning. If learning rate is too large, it may cause converging to a local minimum or divergence.

While doing trial and error, largest possible values are selected at first. Then a smaller learning rate configuration is tested. Then a smaller batch size configuration is tested. Both configurations are found to perform worse than the first test.

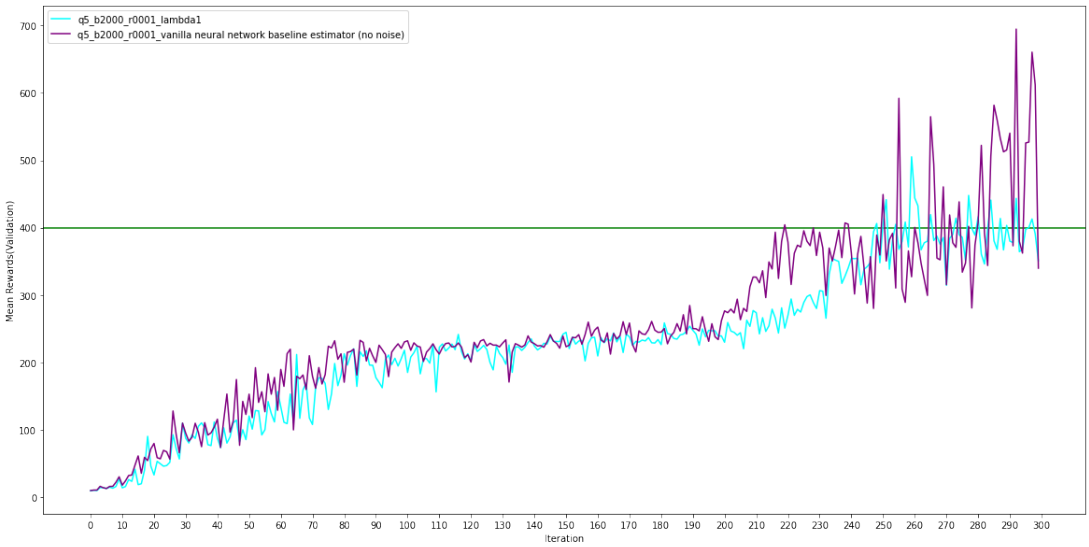


Params: --ep_len 150 --discount 0.95 -n 100 -l 2 -s 32

The chart above shows performance for different reward-to-go and baseline parameter configurations. Reward-to-go always performs better. However, the one with baseline was expected to perform better with less variance which seems not to be the case. It is not clear whether this is bacause of random initialization or implementation issues.

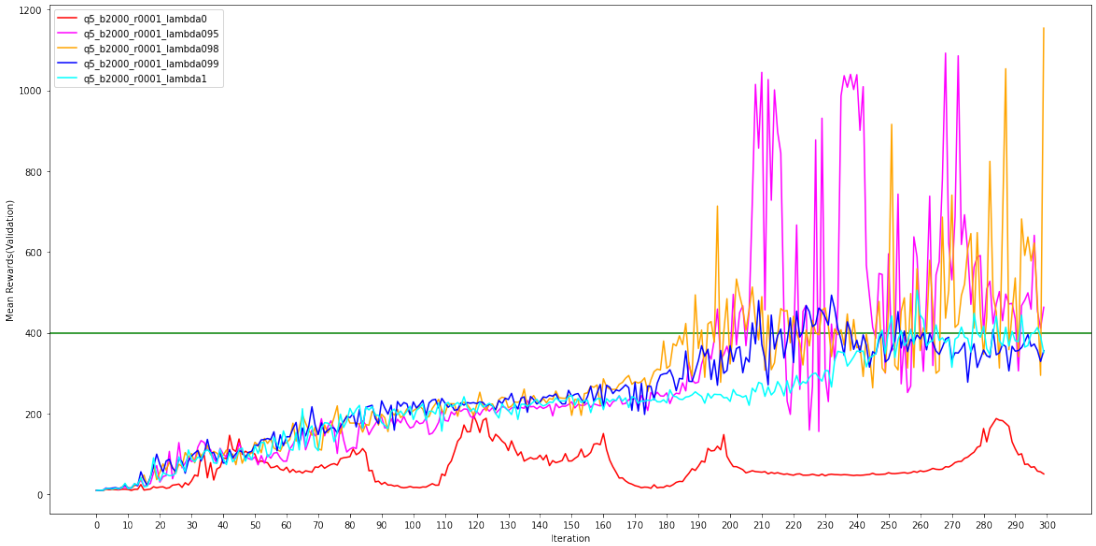
1.5 Experiment 5

First chart shows lambda 1.0 and vanilla neural network baseline estimator. Noise parameter is not added to the vanilla estimator. They are somewhat similar which is expected.



Params: --ep_len 1000 --discount 0.99 -n 300 -l 2 -s 32 -b 2000 -lr 0.001 --action_noise_std 0.5 --reward_to_go --nn_baseline

The following chart shows learning curves for different lambda values.



Params: --ep_len 1000 --discount 0.99 -n 300 -l 2 -s 32 -b 2000 -lr 0.001 --action_noise_std 0.5 --reward_to_go --nn_baseline

As lambda moves away from 1, variance grows. The best performance is obtained when the lambda is set to 1.0. The second best is lambda 0.99. lambda value 0.99 learns faster, it reaches the target mean reward more quickly. However, it fluctuates a lot when compared to lambda value 1.0.

The lower the lambda parameter, the algorithm cares less about the

future and this behaviour adds to variance. When lambda is zero, the algorithm onl cares about the current step value. In this case the performance is the worst. When lambda is set to 1, the algorithm cares about future rewards as much as the current reward and the performance is the best.