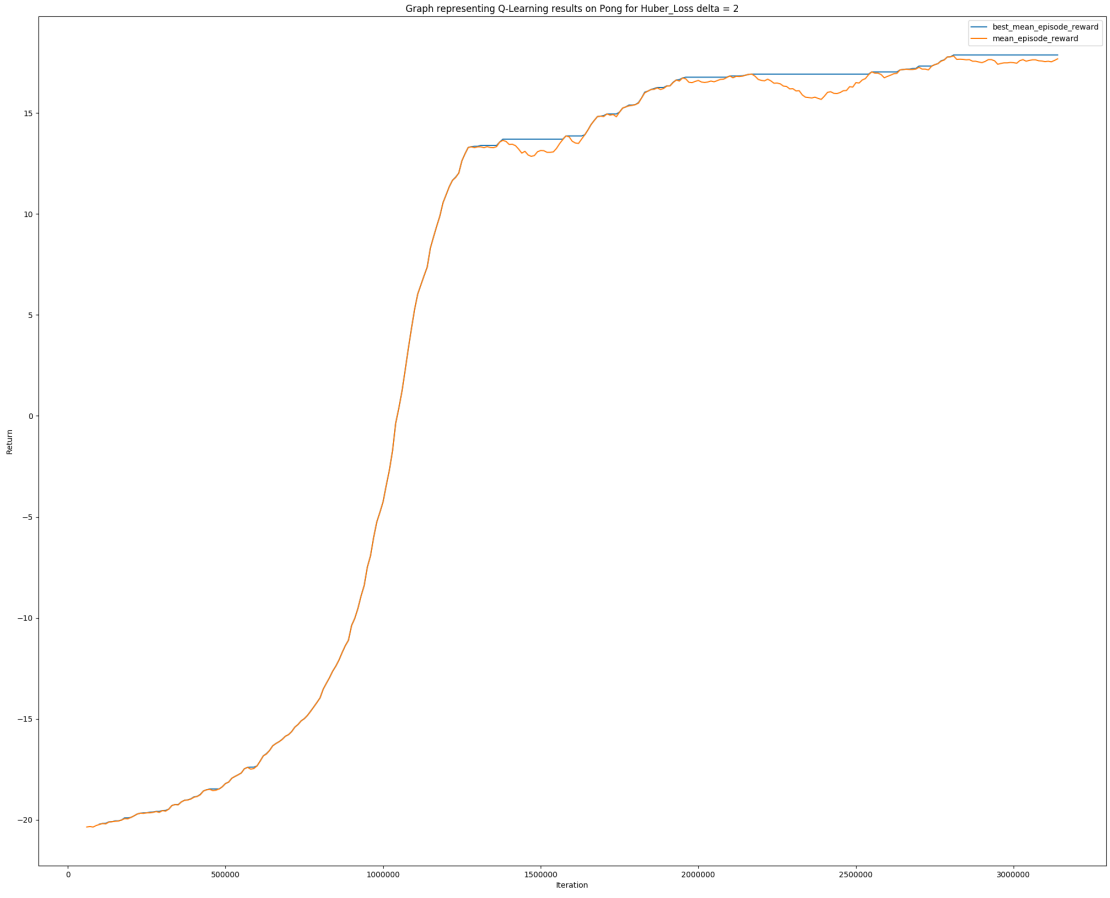
Ezbakhe Hatim

CS 294 – Deep Reinforcement Learning
Homework 3

Part 1

Question 1

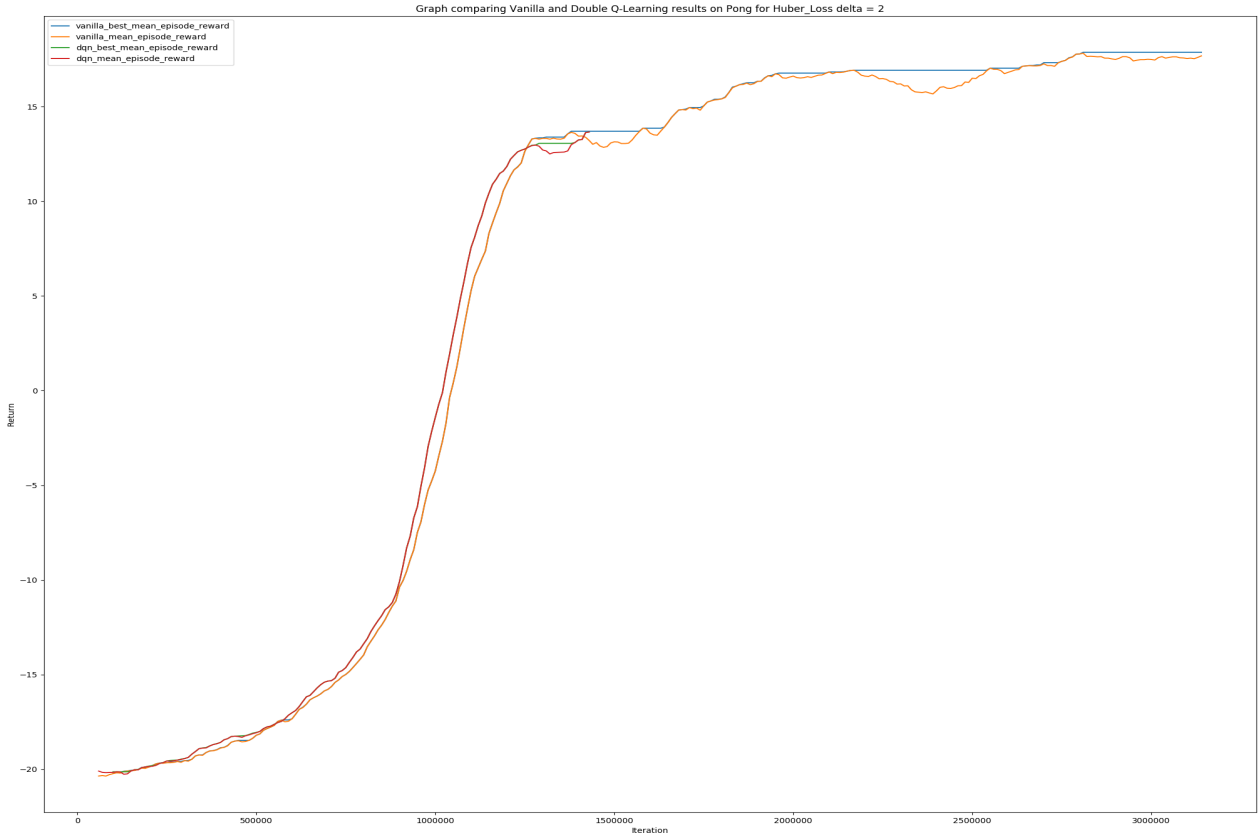
I changed the parameter delta of the Huber loss to 2. With simple Q Learning, I unfortunately don't know if this parameter is the setting that gives the best result as I didn't have the time to compile other significant tests on my CPU. I ran it the first time with this parameter and took ~60 hours to compile.



Ezbakhe Hatim

Question 2

Let's plot the results for Double Q Learning with the exact same set of hyper parameters (Huber loss's delta = 2).
As we can see, DQN has slightly better results as it increases faster, and reaches the plateau first before increasing again.

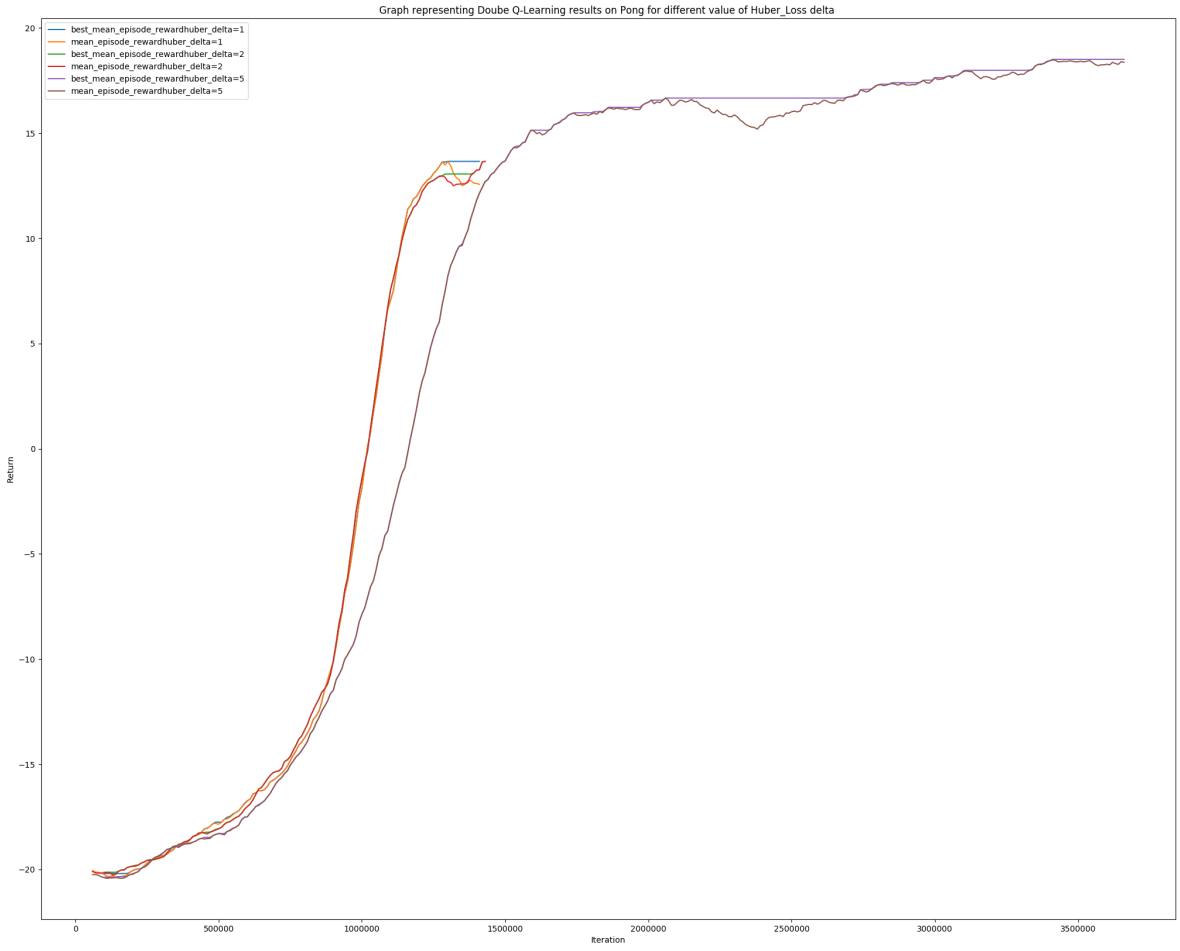


Question 3

When I started working on the code, the first hyper parameter that struck me was the Huber Loss's delta. I did not know which one to choose and looked on the internet for quite some time trying to learn more of the effect of that parameter. I did not know how much resilient to outliers the model should be. This is why I decided to analyze the sensitivity of Q-Learning to this parameter, to have a practical result to my questions.

As I did not start the project early and because of some computer issues (computer shut down twice after ~20 hours of computing), I unfortunately could not have all the results I wanted. The results would have been more interesting if I could have compared them all for 4M of steps.
As we can see, increasing the Huber loss's delta (which increases the losses values as we compute the mean squared error for losses inferior to delta) decreases the rate at which the mean return

increases but seems to make it more consistent. We can't really say much more as the graphs for delta = 1 and 2 are similar and seem to reach a plateau before increasing again.



Ezbakhe Hatim

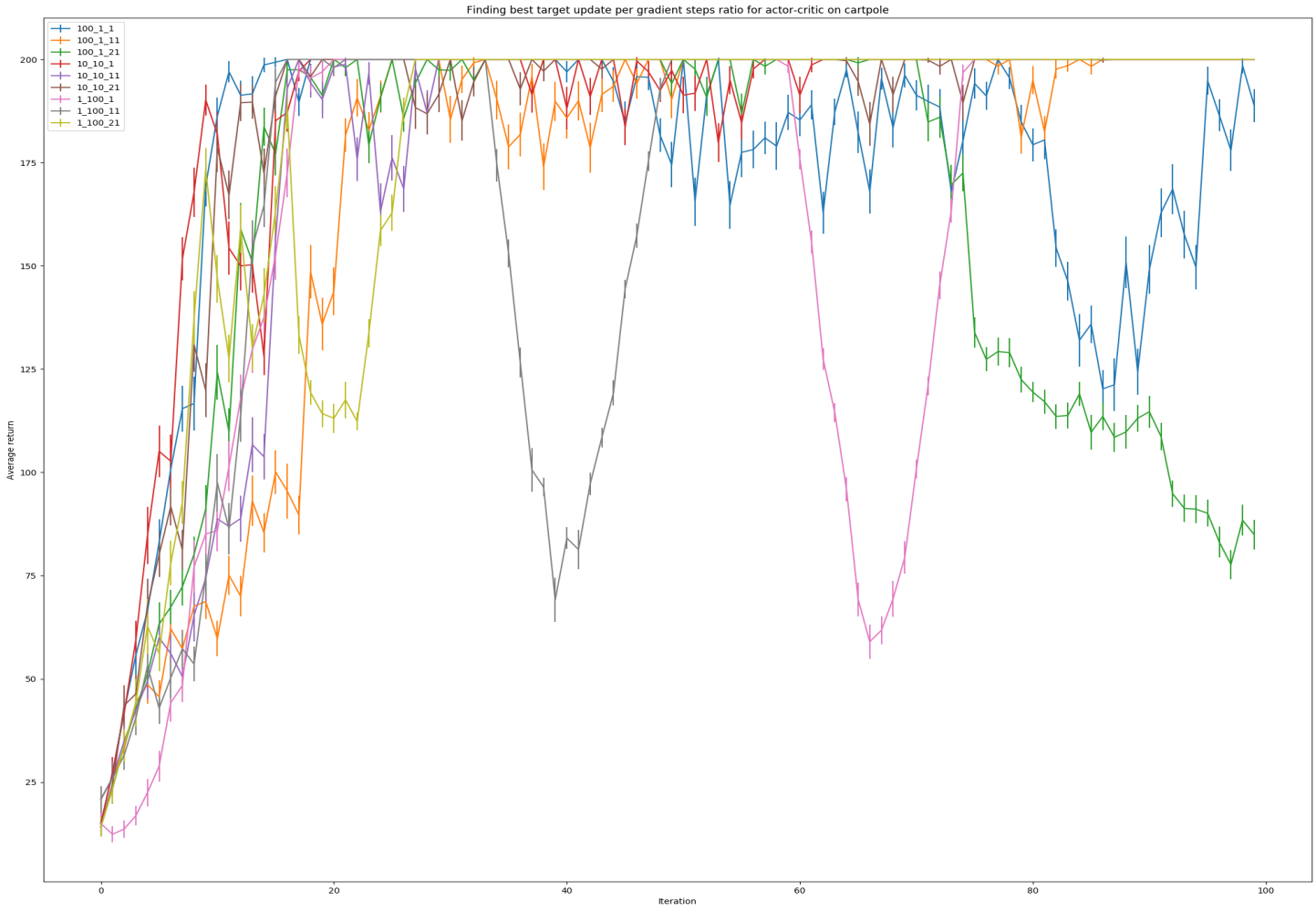
Part 2

Question 1

In the graph below, I represented the average return function of the iteration number (with error lines representing the squared standard deviation) for the cartpole environment. Each graph corresponds to an experiment for a given pair of settings represented on the label (“1_100” represents an experiment ran with 1 target update every 100 gradients steps).

Ezbakhe Hatim

The best results (reaching highest average quickly and consistently) is clearly obtained for "10_10", as the others are fairly inconsistent. This means that for every target value obtained, we do 10 gradient update steps of the critic before updating the target. This operation is repeated 10 times. This is completely different than the results obtained for "1_1" because here we update the critic 100 times more than we update the actor (which makes sense, as operating one gradient step on the critic for one gradient step for the actor wouldn't let the opportunity to the critic to learn).

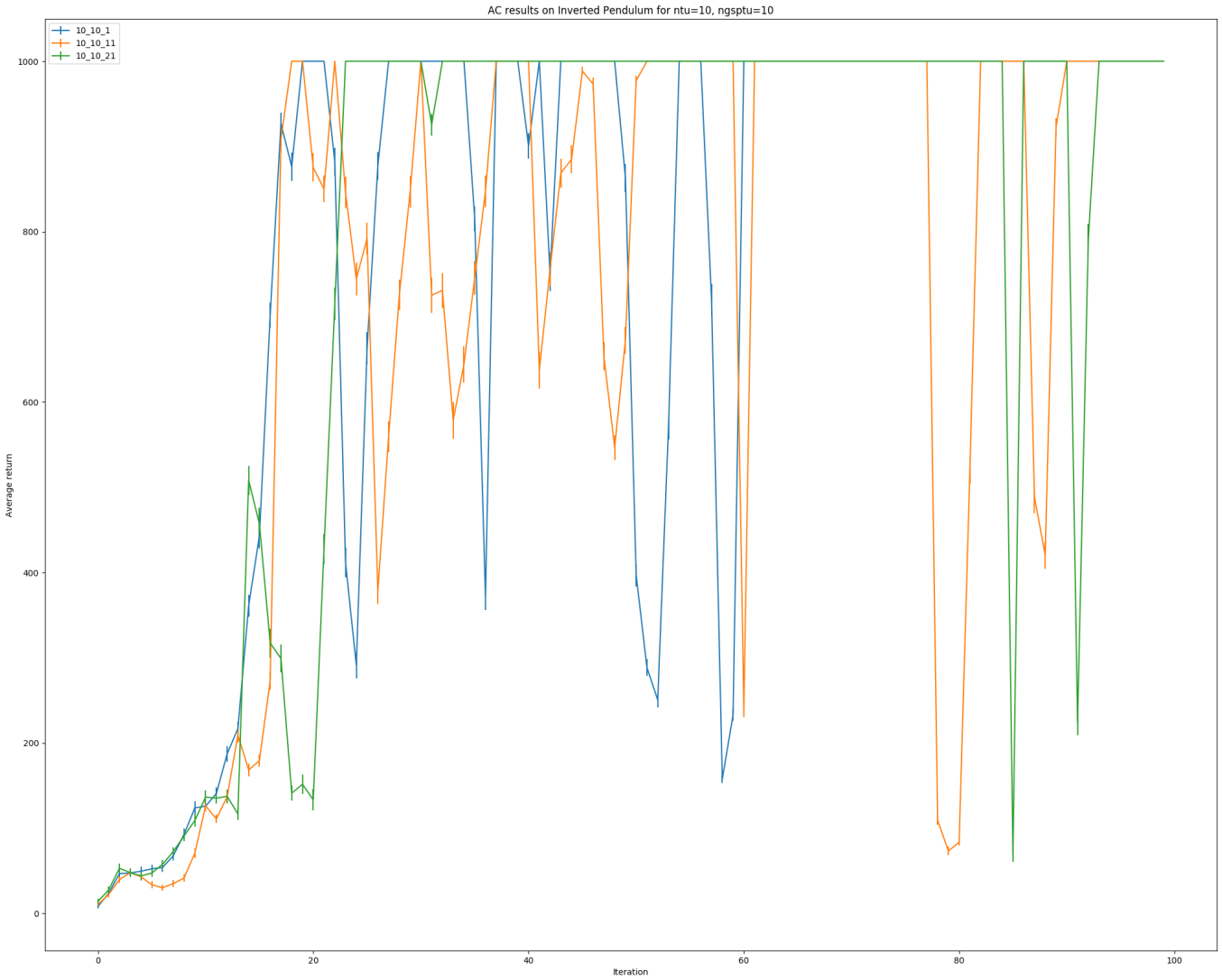


Question 2

Using this best setting, we can see that the results are almost as good as those of policy gradient:

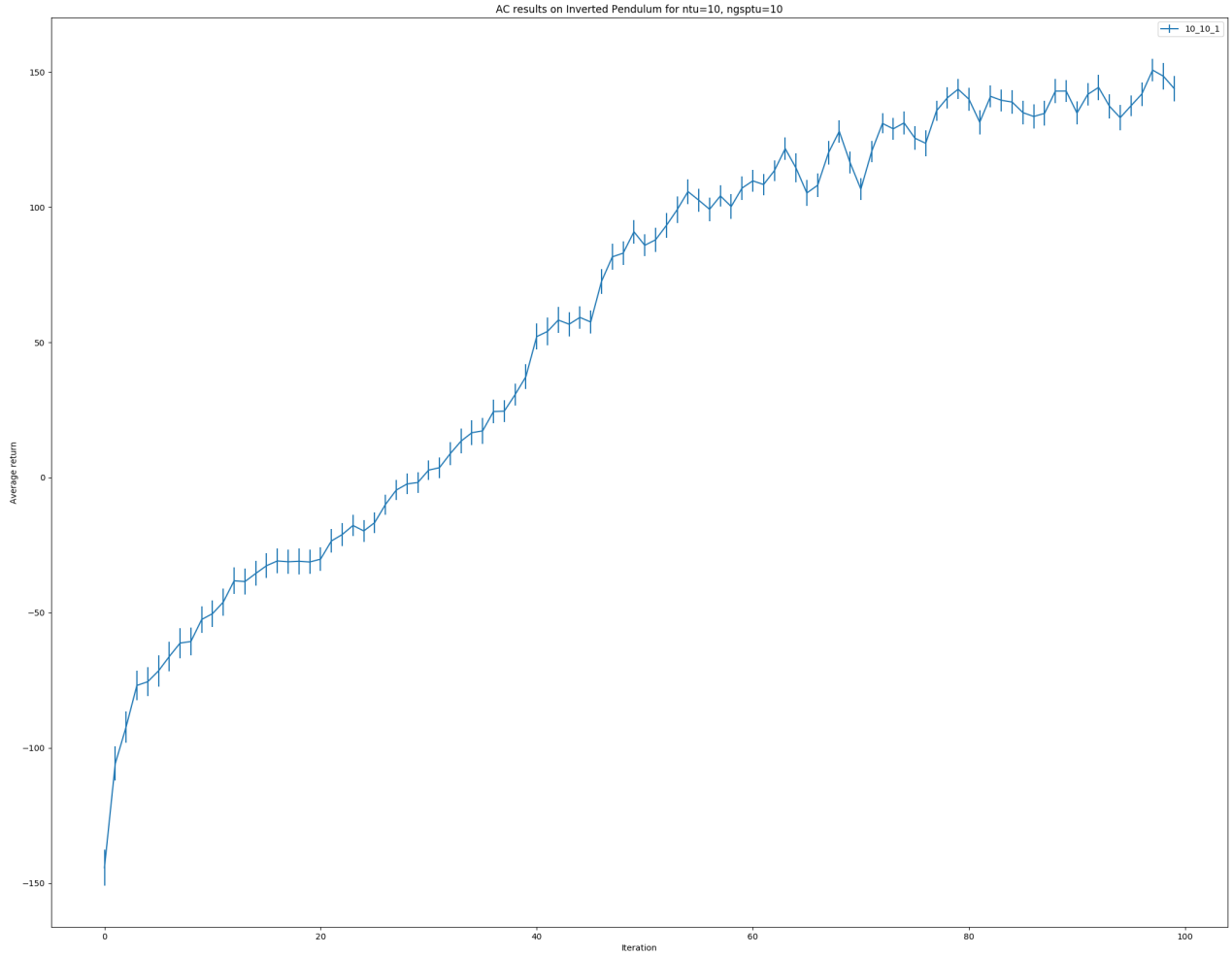
- Inverted Pendulum

Ezbakhe Hatim



- Half Cheetah

Ezbakhe Hatim



Ezbakhe Hatim