Reinforcement Learning Algorithms

Chaitanya Patel

Problem 1 : Analysing the Performance of Baselines for Policy Gradient

Baselines help to reduce variance during training. As per assignment statement, I have tested following 4 cases with monte carlo learning:

1. Without baseline
2. State independent baseline (fixing a single number doesn't help in learning as rewards keep increasing as the agent learn. I used mean of rewards as state independent baseline.)
3. State value function as baseline
4. State value function as baseline with rewards scaled to have mean zero and std one

Implementation

- I have provided the source codes for both the environment and all 4 cases with proper comments.
- Case-1 and Case-2 are implemented in same file. Similarly, Case-3 and Case-4 are implemented in same file.
- Plots and some helper codes are also given.

Experiments

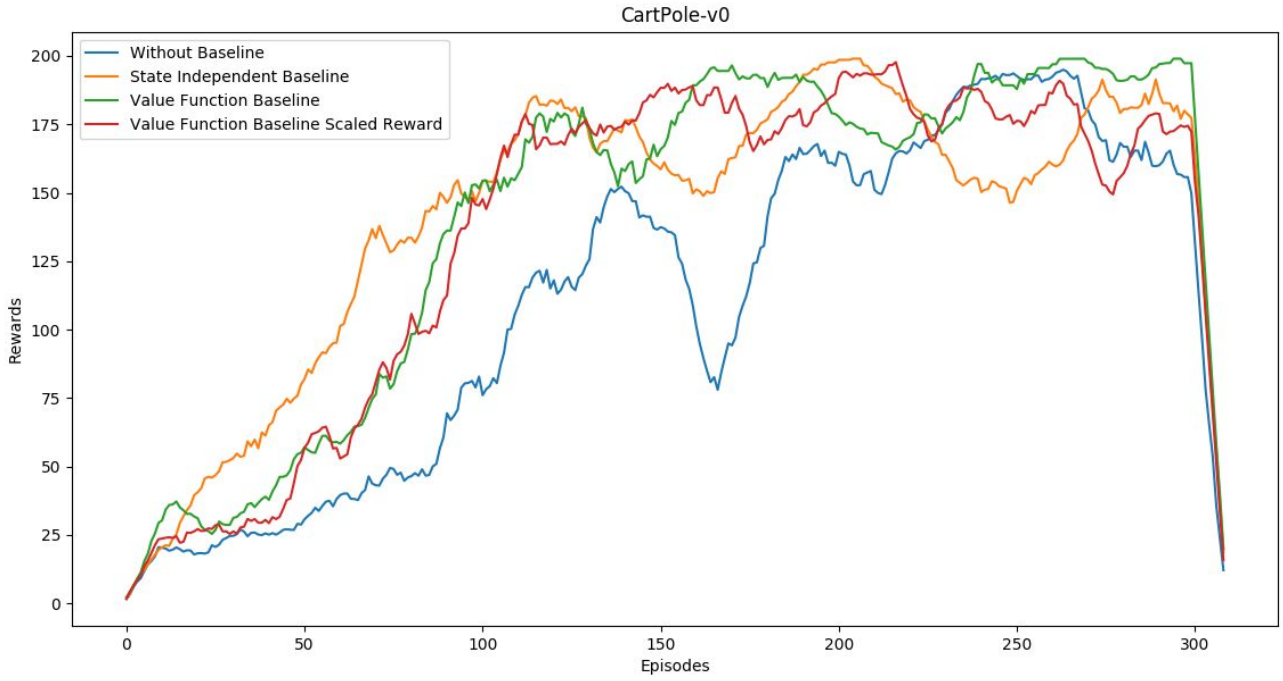
1. **Baselines helps to converge faster and better**

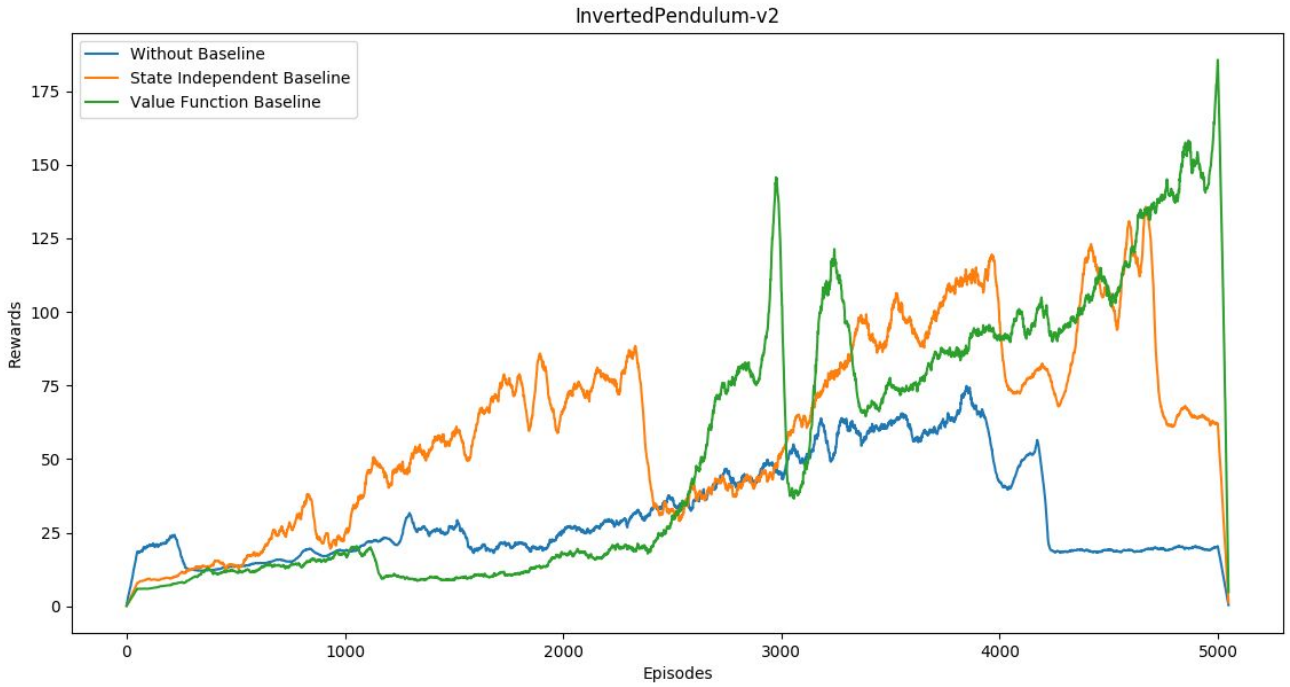
- Case-1 to Case-2 : From the plots below, we can see that adding baseline helps to converge faster.

- Case-2 to Case-3 : State value generally increases while training. Hence it is more sensible to keep baseline according to state value. Thus state dependent baseline converges better than state independent baseline.
- Case-3 to Case-4 : Scaling rewards to have mean zero and std one helps to normalize the behaviour across the episodes. Thus it converges better.

Plots given below shows these relative convergence. The results are averaged over 3 runs of training.

CartPole-v0 is relatively simple problem, thus the differences are not distinguishable. But InvertedPendulum-v2 is harder because of continuous action space. Thus the better convergence is observed with state independent baseline and even better with value function baseline.

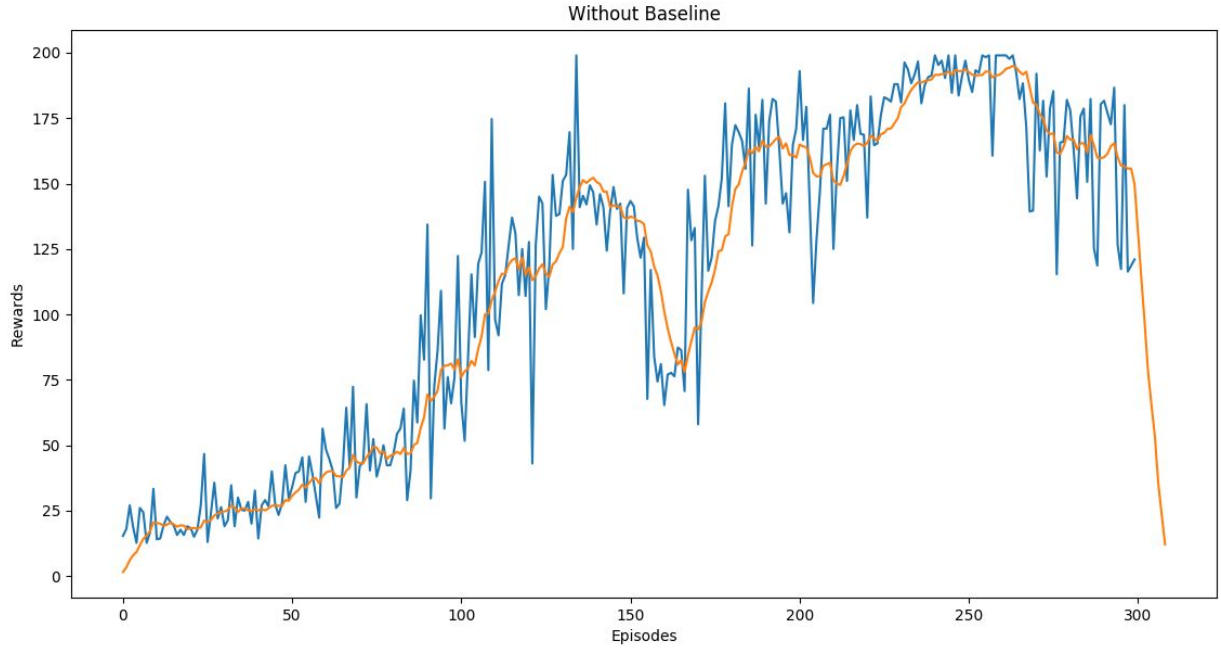


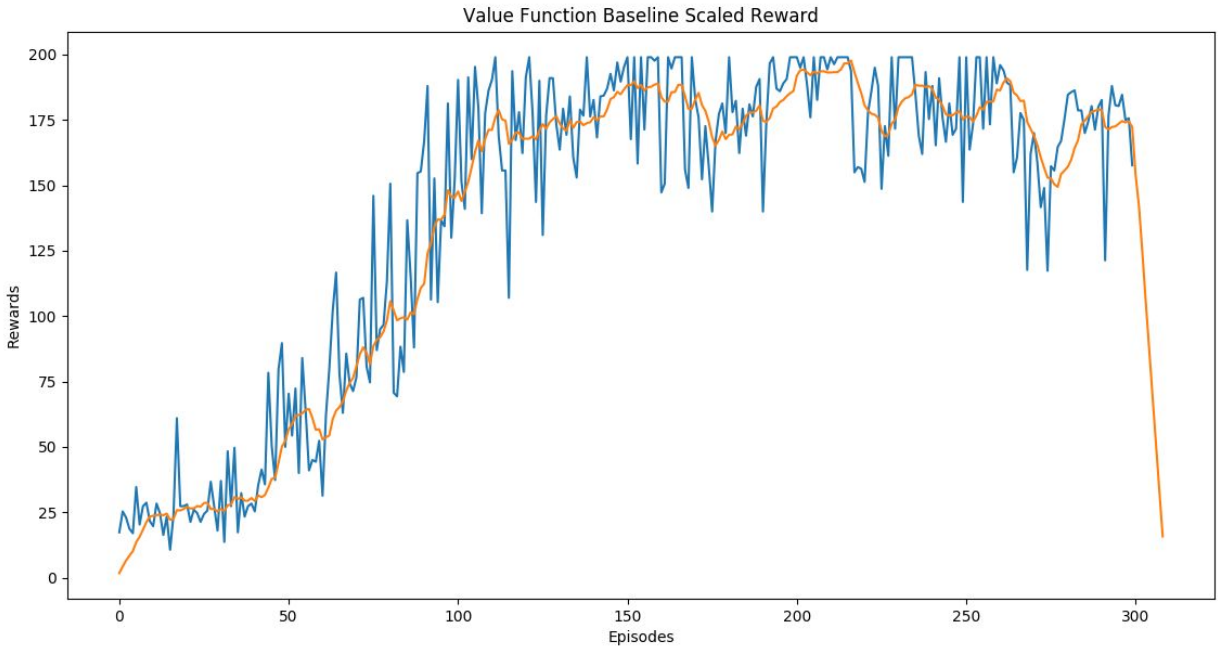


2. Baselines reduces the variance

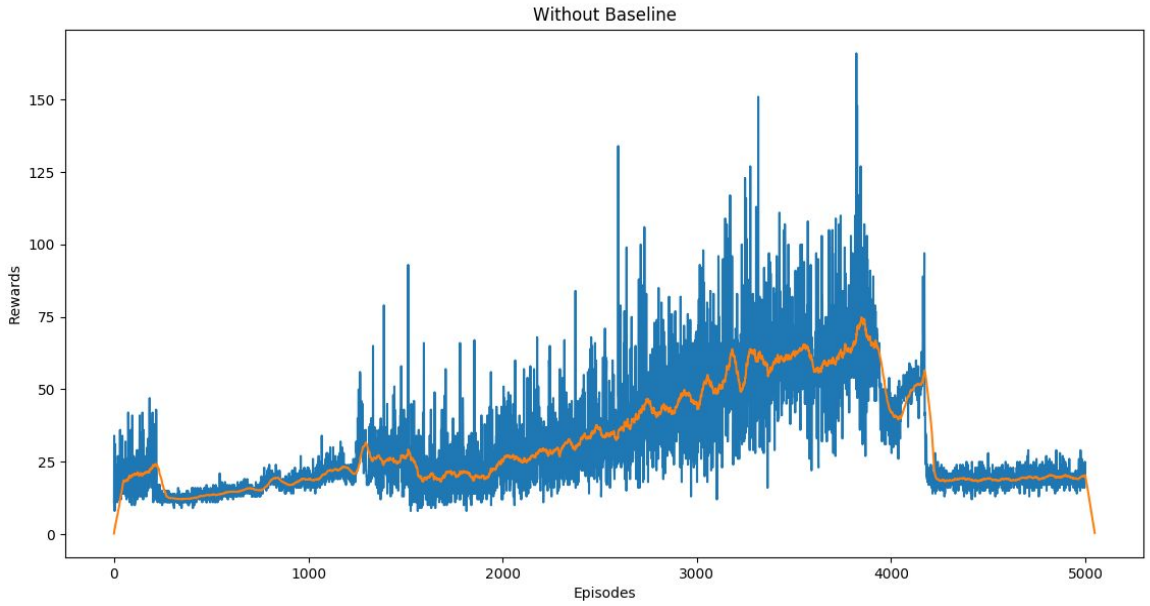
- Below, the plots for individual cases are given. Plots have rewards for each episode and running window mean for those rewards.
- We can observe that variance reduces from Case-1 to Case-4.
- Here I am showing plots for Case-1 and Case-4 only. We can see the reduction in variance by observing the less width of the fluctuations. Also with baseline, the learning is more stable than without baseline.
- ***Plots of all 4 cases for both environments are submitted in plots directory.***

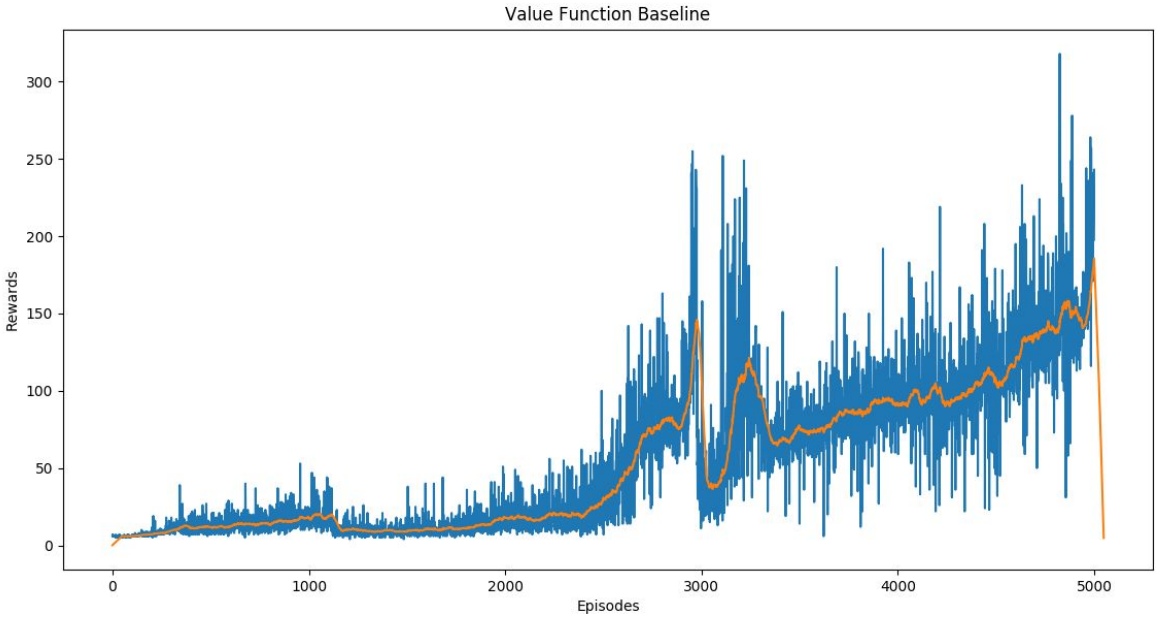
Plots for CartPole-v0





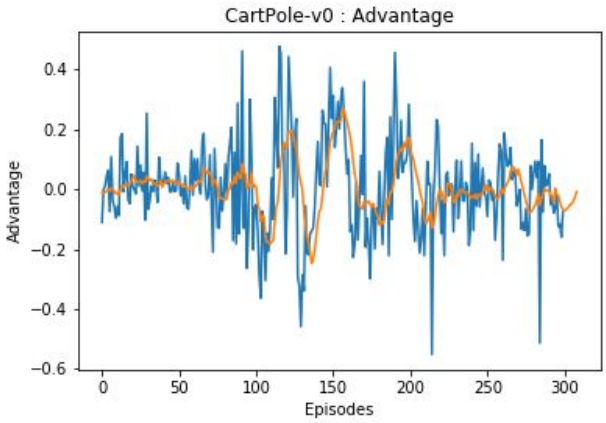
Plots for InvertedPendulum-v2

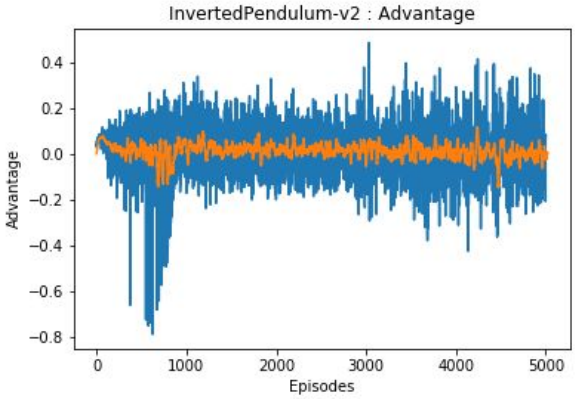




3. Advantage

For Case-4, scaling the targets of baseline network to have mean zero and std one, the plot for advantage i.e. rewards subtracted by this baseline is given below. It is converging to zero because value network is learning to estimate correct value.





4. To accelerate learning

To accelerate learning, we can store each transitions in a buffer and we can do reply of random transitions. We need to store the log probabilities and other needed values. Thus we can do multiple gradient descent using same batch of data.

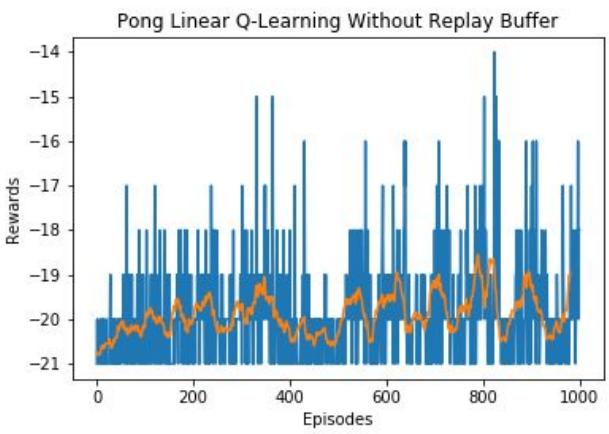
Problem 2 : Q-Learning, Double Q-Learning and Policy Gradient with Linear and Nonlinear Approximation

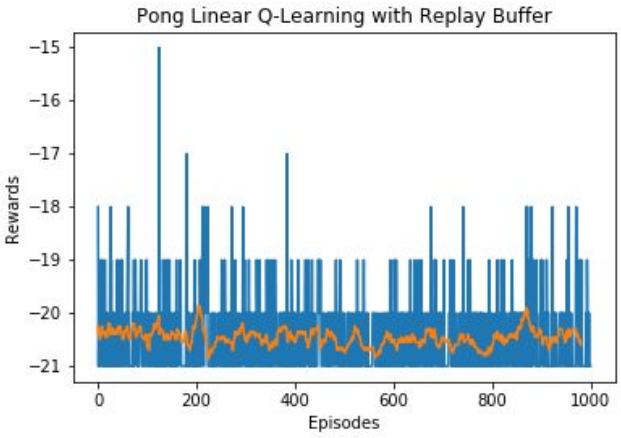
- I implemented Q Learning, Double Q Learning and Policy Gradient (with and without baseline) with linear approximation and nonlinear approximation.
- I also extended nonlinear approximation with commonly used techniques like using CNNs, replay buffer, frameskip, etc. to stabilize learning.
- Source codes are submitted with proper commenting.
- Each episode of pong ends when one of the players get 21 score. Thus each episode is very long. So monte carlo technique in Policy Gradient can be very slow to learn. To overcome this problem, I considered the end of life as end of episode and thus the network can be updated frequently. This update doesn't affect the game learning dynamics at all.

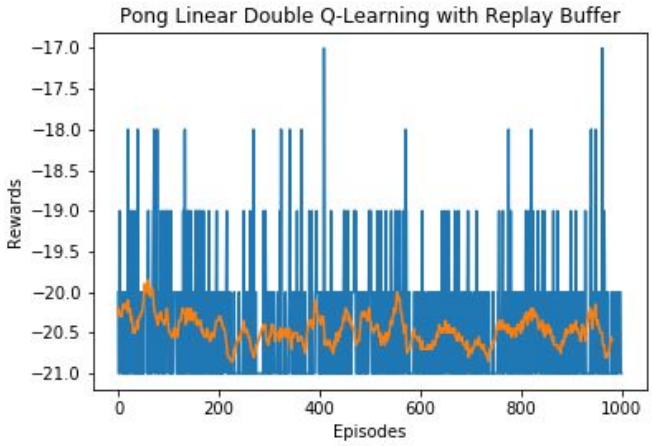
Linear Approximation

- Each state is represented as linearized vector of 80*80 grayscale image.
- A linear function approximation here is a matrix of size (6400, num_actions).

Q Learning and Double Q Learning

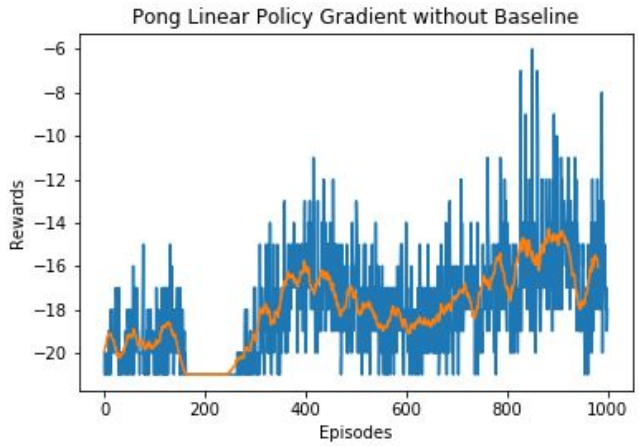


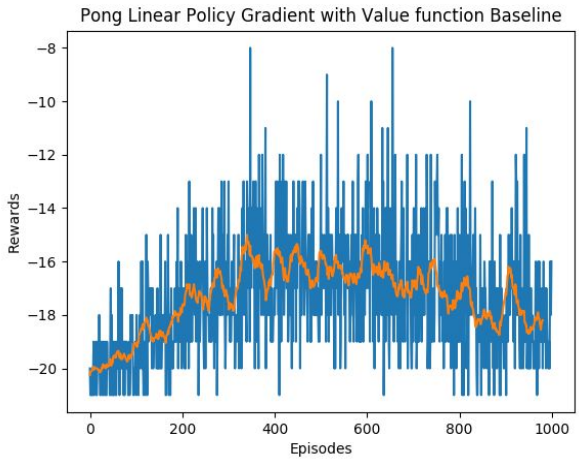




- Here Q Learning and Double Q Learning - both are not able to learn properly because linear approximation is not able to capture the complexity of game.
- For each case, average score is around -20 to -20.5 for 1000 episodes.
- Without replay buffer, the variance in learning is high. Uniformly sampling from replay buffer helps in reducing variance because the agent has diverse set of samples in each batch update.

Policy Gradient without baseline and with baseline





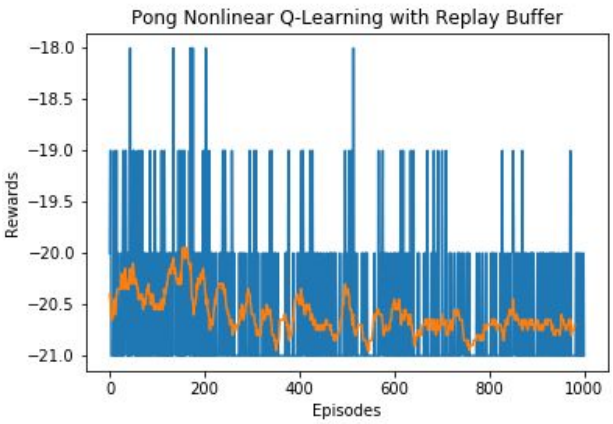
- In policy gradient, linear approximation is able to learn slightly better than Q Learning and Double Q Learning. It is because we are directly learning policy unlike Q or Double Q Learning.

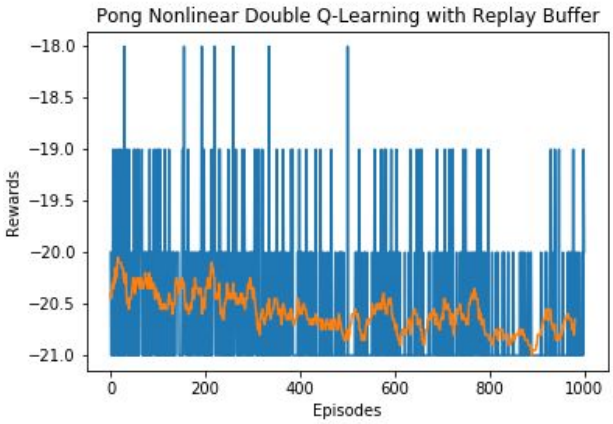
- Score went on to -16 at convergence point for Policy Gradient without baseline trained for 1000 episodes. Whereas with value function as baseline, the score reached -15 at the convergence point. Also with baseline, the learning was more stable than without baseline.
- We can see that using baseline is helpful to stabilize the training and achieve better convergence.

Nonlinear Approximation

- Again, each state is represented as linearized vector of 80*80 grayscale image.
- Now nonlinear function approximation here is a small neural network with 4 layers. Input layer has 6400 neurons, two hidden layers has 96 and 32 neurons respectively. Output layer has 6 neurons.
- This nonlinear approximation doesn't increases the performance much. Main reasons are :
    - We are going 6400 to 96 in the first layer itself (it is a fully connected network. So having 1024 neurons in second layer yields around 6 million parameters, which are very hard to train using limited computing resources).
    - Input representation is linearized image which is very inefficient way to parameterize input image. We will see that using CNNs, we get better results.

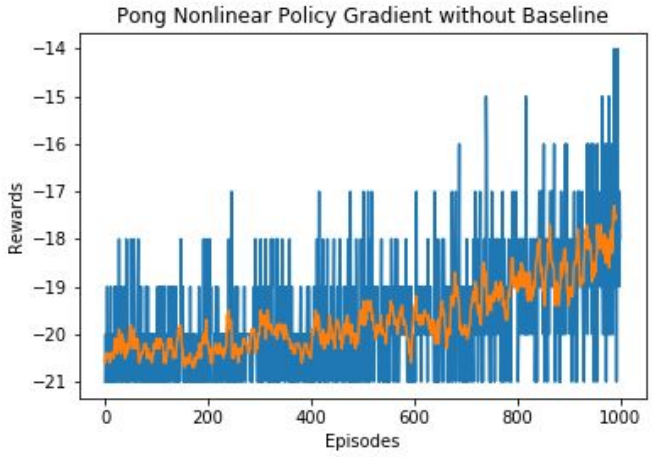
Q Learning and Double Q Learning

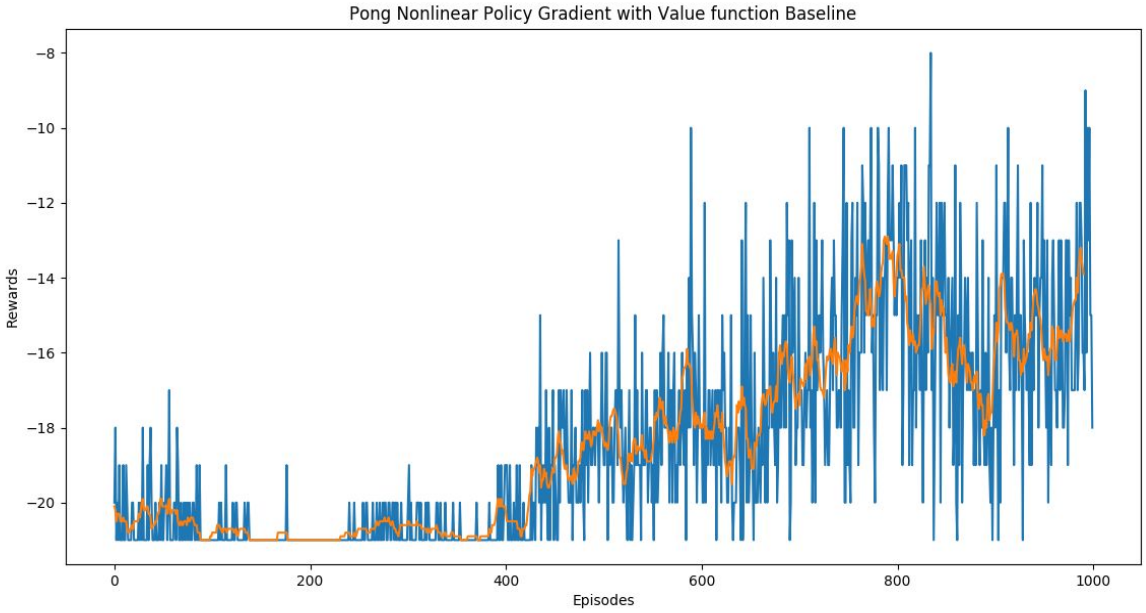




Policy Gradient without baseline and with baseline

- With nonlinear approximation, policy gradient performs better. See the plot below. Although, the average reward at 1000th episode is -16, it is slowly but continuously learning the nonlinear network. With more training, it can surpass the linear approximation.
- Also with baseline, the average reward at 1000th episode is around -14 which better than without baseline.

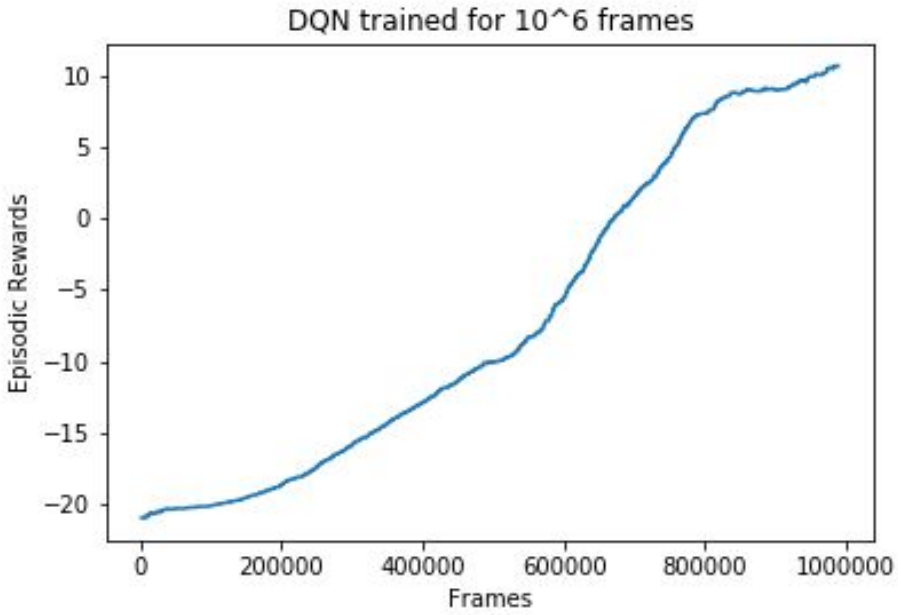




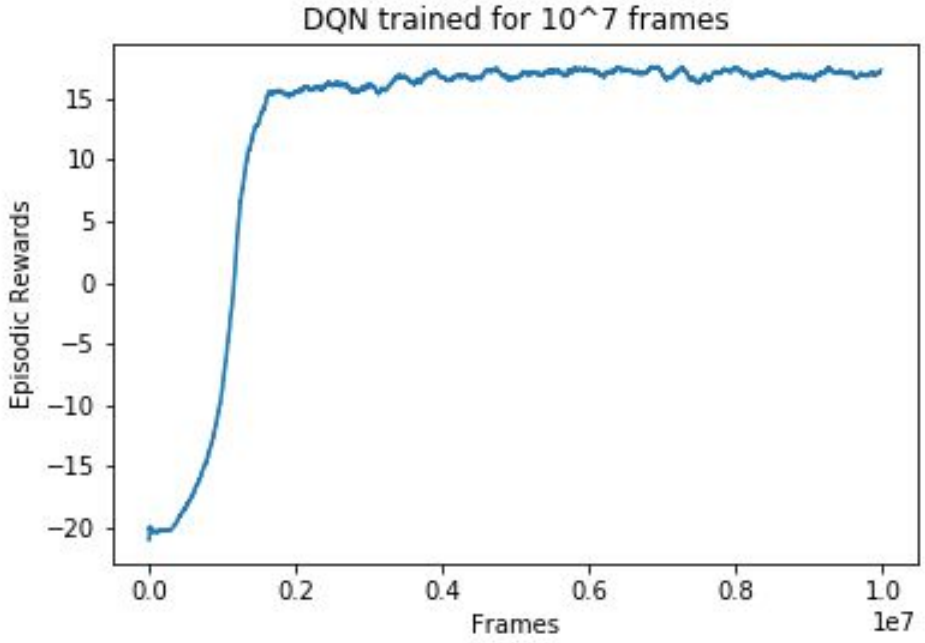
Nonlinear Approximation with some tricks

- Reason for low performance even in nonlinear case was the inefficient representation of state. Here I incorporated some commonly used tricks for better and stable learning.
- **Using CNN :** Instead of linearized image and fully connected layer, a small CNN with 3 convolutional layer and 2 fully connected layers is used. Note that CNN has much less number of parameters than our previous fully connected network. Even with less number of parameters, it can featurize the image better.
- **Consecutive frames as input :** Single frame is not sufficient to know the state of the game fully because the motion information is not there. So last 4 frames is given as input to the network.
- **Frame Skip :** Framerate of the game is high. So we process more frames with less information. Here I used each 4th frame.

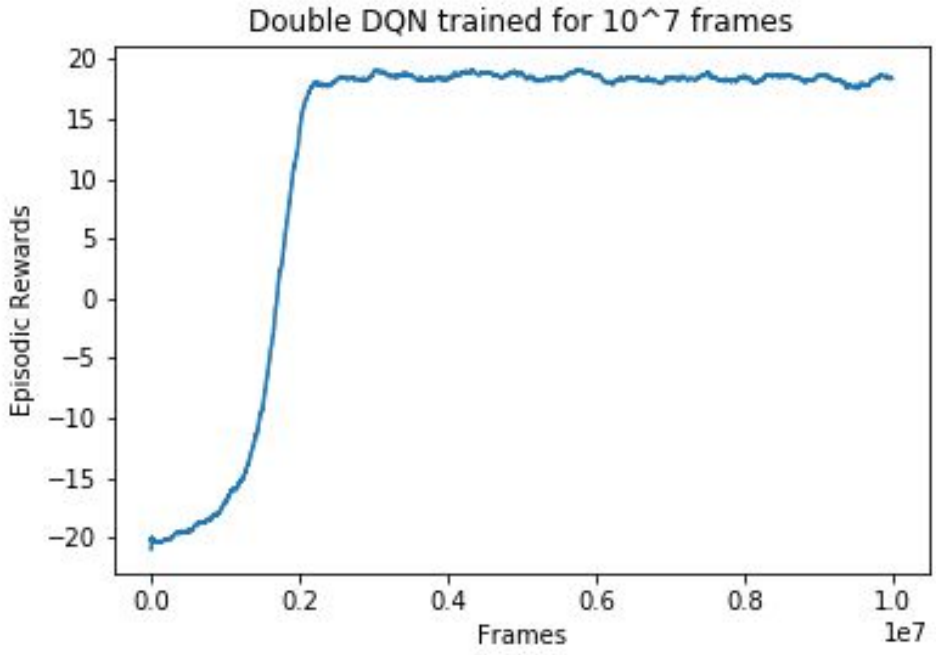
These ideas were proposed for DQN. I trained DQN for 10^6 frames which is roughly equivalent to 800 episodes. The average reward after training was 10 and it was still increasing. See the following plot :



I continued the training till 10^7 frames (training 1.5 day) and it converged to the score around 17. See the plot below :



I also implemented update equation for double learning variant of DQN i.e. DoubleDQN and trained for 10^7 frames. As expected, it learned better than DQN. At convergence, it achieved the reward around 19.



Videos of one episode of DQN and DDQN after training, are provided with submission. Agent is playing like pro.