REINFORCEMENT LEARNING CS5500

Assignment 3

Ritesh Yadav ma17btech11009@iith.ac.in
MA17BTECH11009

Question 1(a) : In online Q-learning same weight parameters are used for all Q(s,a) pairs. Hence, on updating Q, it will change Q(s,a) value of all (s,a) pairs and would not be able to retian old values.
Whereas , in (Watkin's) tabular Q-learning , their is seperate Q(s,a) values for all (s,a) pairs. So on updating value of one pair doesn't affects the value of another pair.

Question 1(b) : Since the replay buffer here is static. The stochastic batch update will make the approximator Q_{phi} to only learn Q-values corresponding to the replay buffer we have.
As Q_{phi} keeps updating ,we need to collect more replay samples using current Q-function to help the agent learn better and converge to the optimal state-value function.
Since, after each update, policy changes, one need to collect samples from updated policy not old one. Also, if target networks are not used, the network will further face convergence issues.

Question 2(a) :

- **For MountainCar-v0 :** Reward of -1 for each time step, until the goal position of 0.5 is reached. As with MountainCar-v0, there is no penalty for climbing the left hill, which upon reached acts as a wall. Random states with their respective choosen actions and corresponding rewards are recorded in the code. Their are two actions : move left and move right.
- **For Pong-v0 :** Similarly Pong-v0 have different rewards for different actions and different states reached which are recorded in the code.

Question 2(b) :

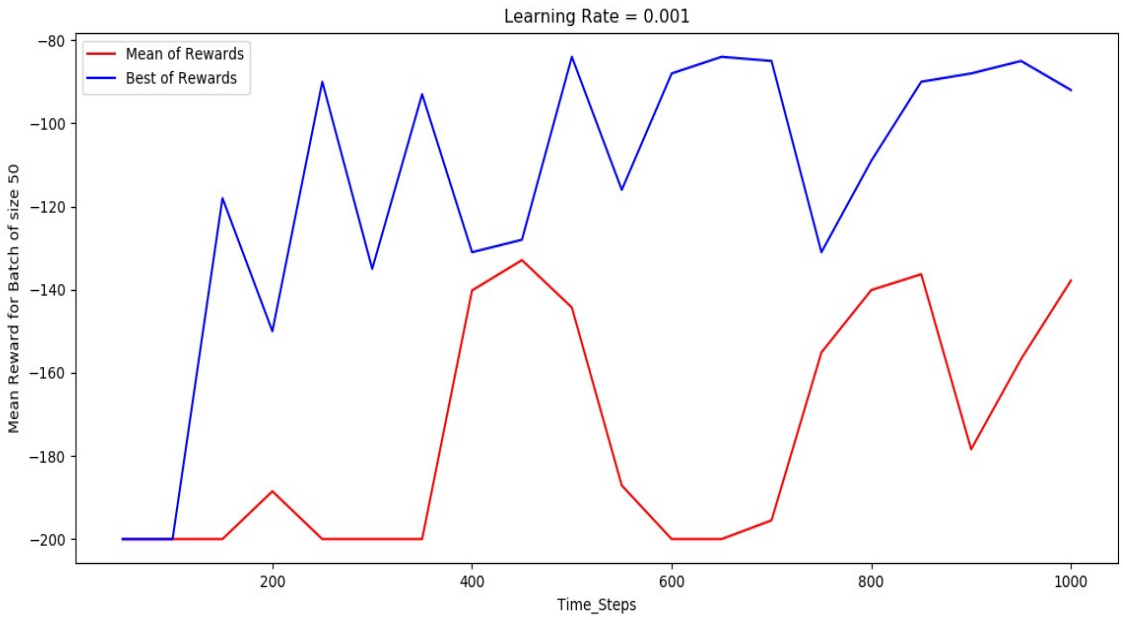

Figure 1 : MountainCar-v0

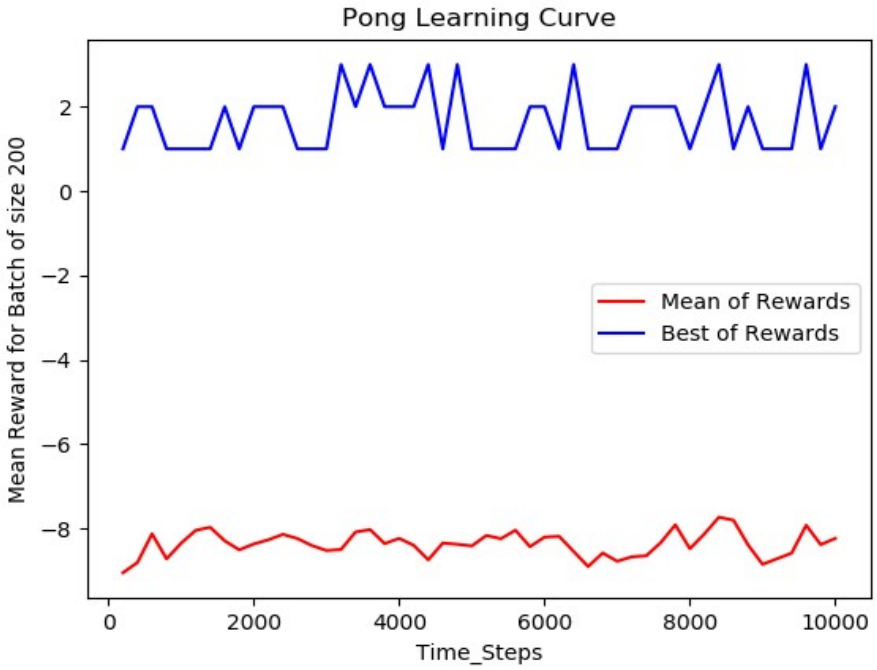

Figure 2 : Pong-v0

Pong wasn't able to completely train. Due to lack of computational resources it was able to reach to 10,000 epsiodes only. The code took huge amount of time in preprocessing and training .

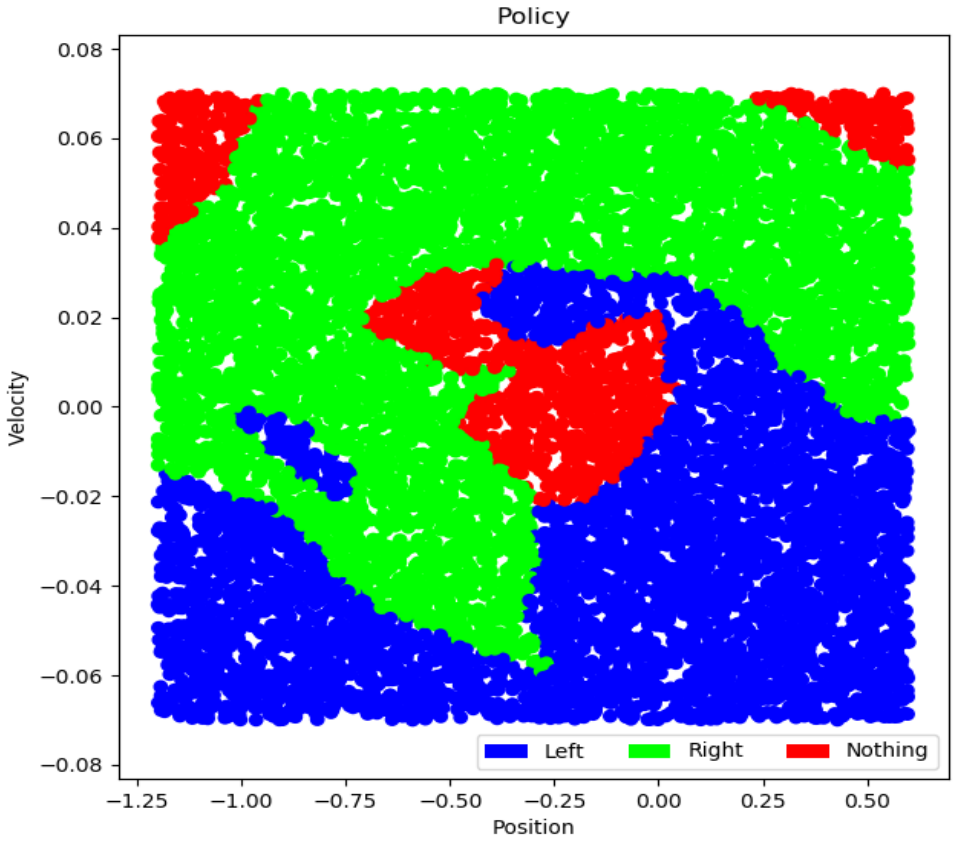

Figure 3 : Action Choices of trained MountainCar-v0

Question 2(c) :

The learning rate hyperparameter controls the rate or speed at which the model learns.
In the figure given below we can see that the curves with large learning (e.g 0.1 and 0.01) rate allows the model to learn faster, at the cost of arriving on a sub-optimal final set of weights.Whereas smaller learning rates (e.g 0.001 and 0.0001) may allow the model to learn slower but attains optimal or even globally optimal set of weights and may take significantly longer to train.If a learning rate that is too large will result in weight updates that will be too large and the performance of the model (such as its loss on the training dataset) will oscillate over training epochs. Oscillating performance is said to be caused by weights that diverge. A learning rate that is too small may never converge or may get stuck on a suboptimal solution.In the worst case, weight updates that are too large may cause the weights to explode and reults in to high variance in the curve like of 0.1.Therefore, we should not use a learning rate that is too large or too small.

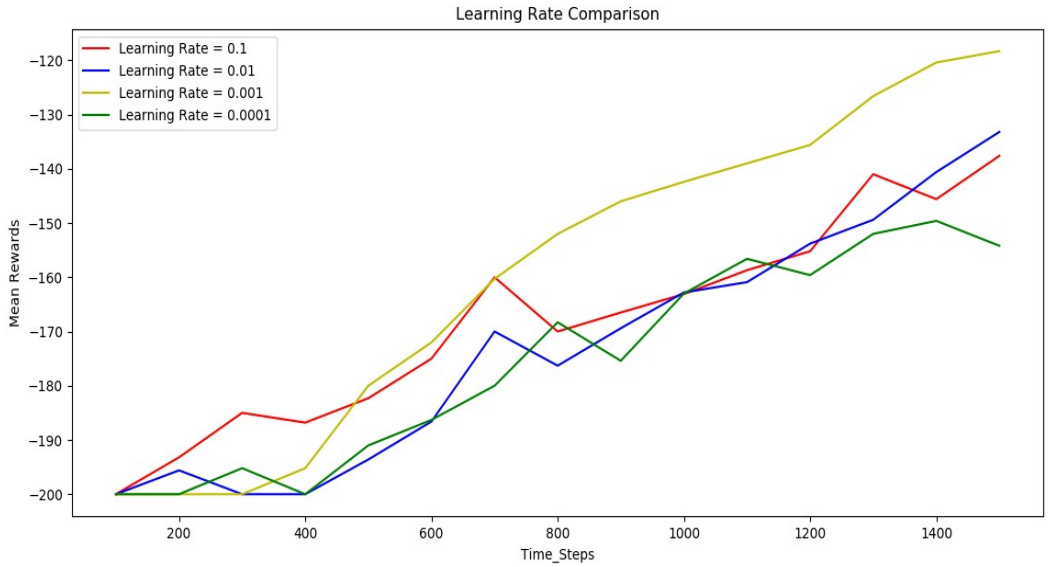

Figure 4 : Comparison of Learning Rate Hyper-parameter

Question 3(a) :

- **For CartPole-v0 :** Their are 2 actions named : Push cart to left and Push cart to right.Reward is 1 for every step taken, including the termination step. Observations are recorded in the code.
- **For LunarLander-v2 :** Reward for moving from the top of the screen to landing pad and zero speed is about 100..140 points. If lander moves away from landing pad it loses reward back. Episode finishes if the lander crashes or comes to rest, receiving additional -100 or +100 points. Each leg ground contact is +10. Firing main engine is -0.3 points each frame. Solved is 200 points . Random observations are recorded in the code snippet. Their are three actions : fire main engine, fire left engine and fire right engine.

Question 3(b) :

On comparison we can see that advantage normalisation technique of finding the rewards leads to low variance in the estimates then other two with output to be stable in nature.Also advantage normalisation leads to achieving the optimal reward and training in less number of episodes. Rewards-to-go technique leads to stable estimate then simple rewards generating method.And have more rising and low variance learning curve in comparison to later one. Number of iterations required to train till the optimal reward is attained is also less then the simple rewards generating method.

- **For CartPole-v0 Enviorment :**

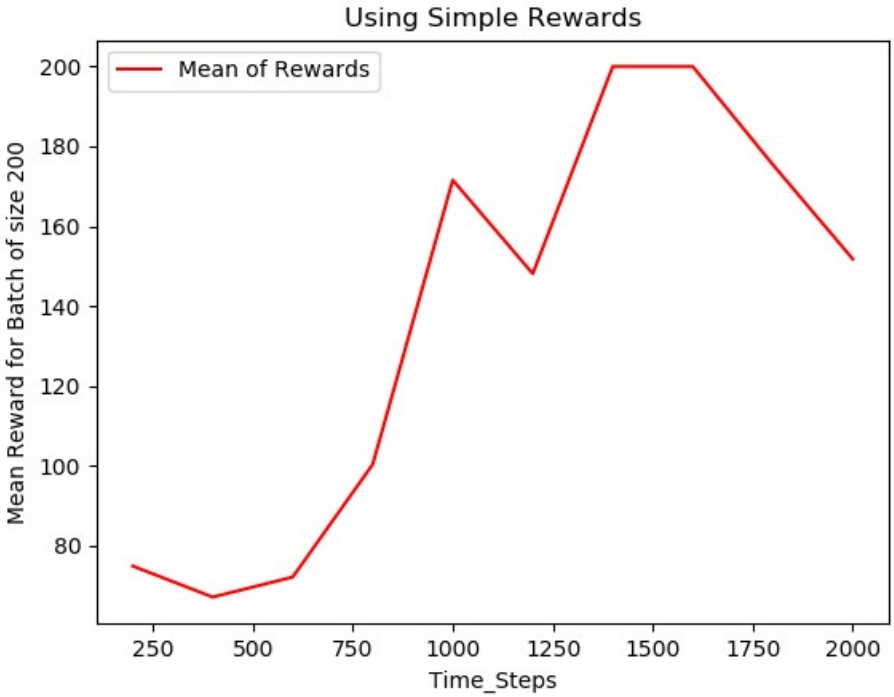

Figure 5

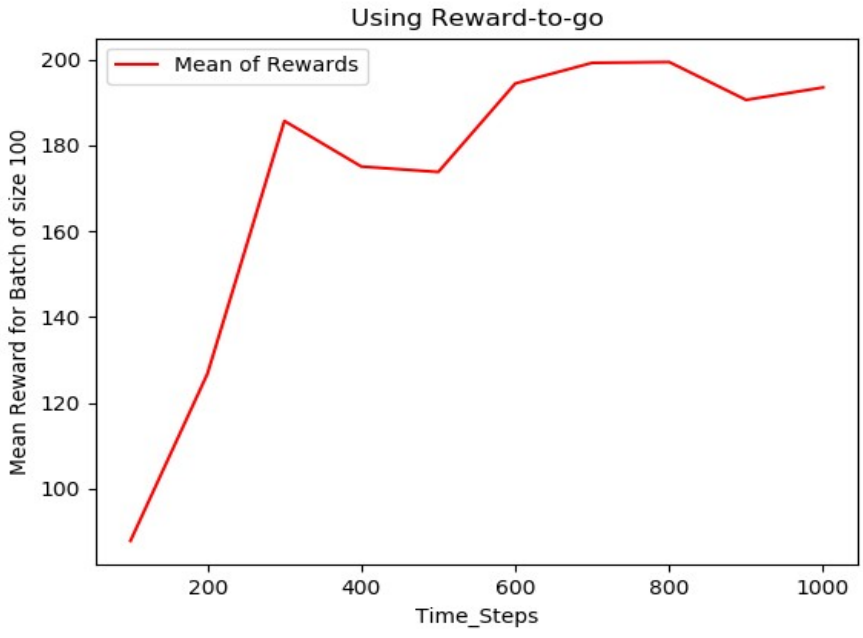

Figure 6

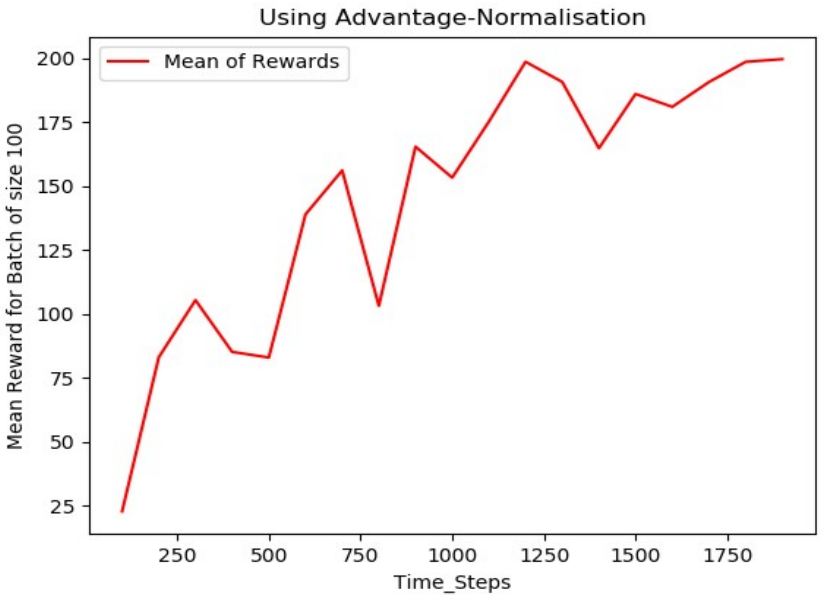

Figure 7

- **For LunarLander-v2 Enviorment :**

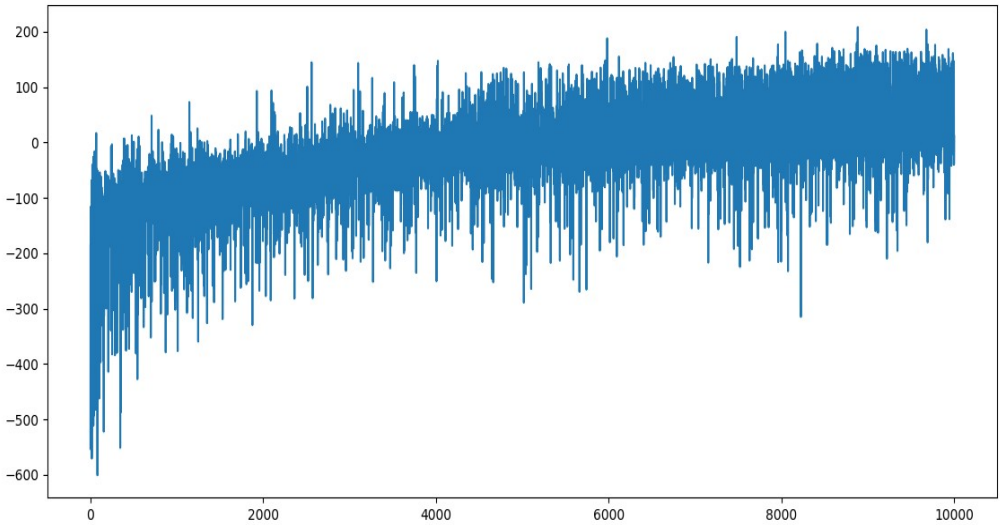

Figure 8 : Using Simple Rewards

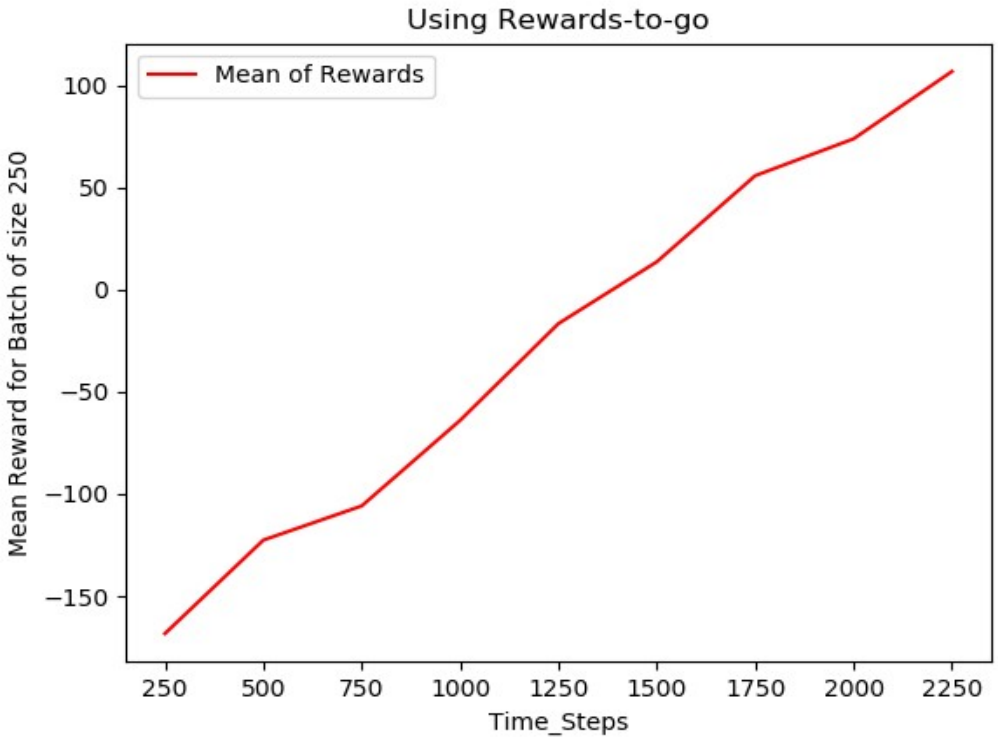

Figure 9

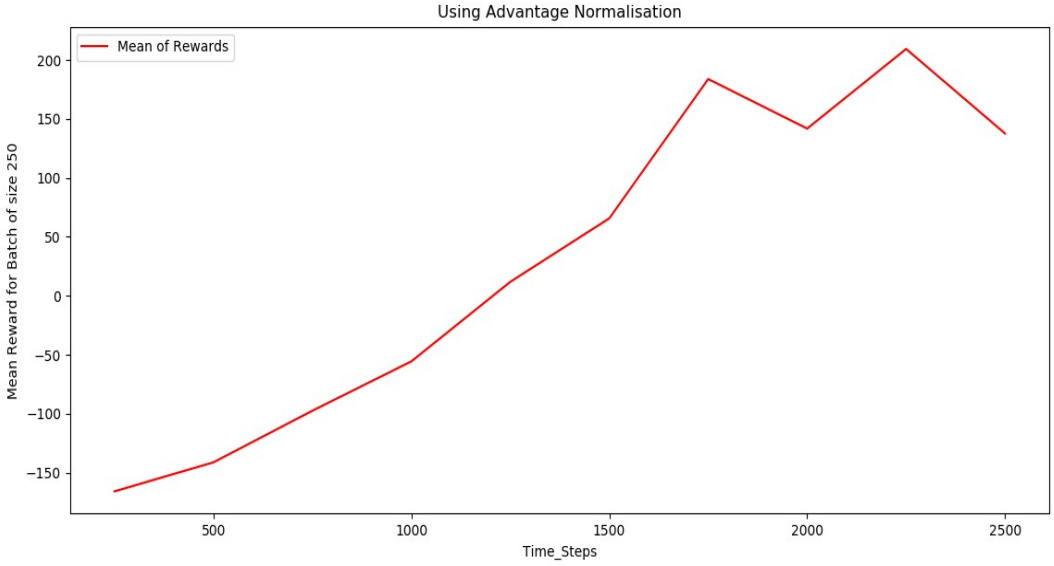

Figure 10

Question 3(c) :

Batch size impacts learning significantly. If your batch size is big enough, this will provide a stable enough estimate of what the gradient of the full dataset would be. By taking samples from the dataset, we estimate the gradient while reducing computational cost significantly. The lower we go, the less accurate our esttimate will be, however in some cases these noisy gradients can actually help escape local minima. Another advantage of batching is for GPU computation, GPUs

are very good at parallelizing the calculations that happen in neural networks if part of the computation is the same .
In here for comparison we have taken three different batch sizes i.e 4, 16 and 64 experimented on CartPole-v0 enviorment. We can see the learning curve for the batch size = 4 attains optimal reward after the policy gradient training of around 1500 iterations. Whereas in the learning curve of batch size = 16 it is attained at around 1000 iteration. And if we increase the batch size to 64 the number of iterations required to attain the optimal rewards decreses to around 800.
Hence ,we can conclude that as the batch size increases to a particular the time taken to achieve the optimal rewards decreases and our estimates are more stable enough. But this increase in batch size must be to a particular extent otherwise it will leads to a large amount of variance within the batch

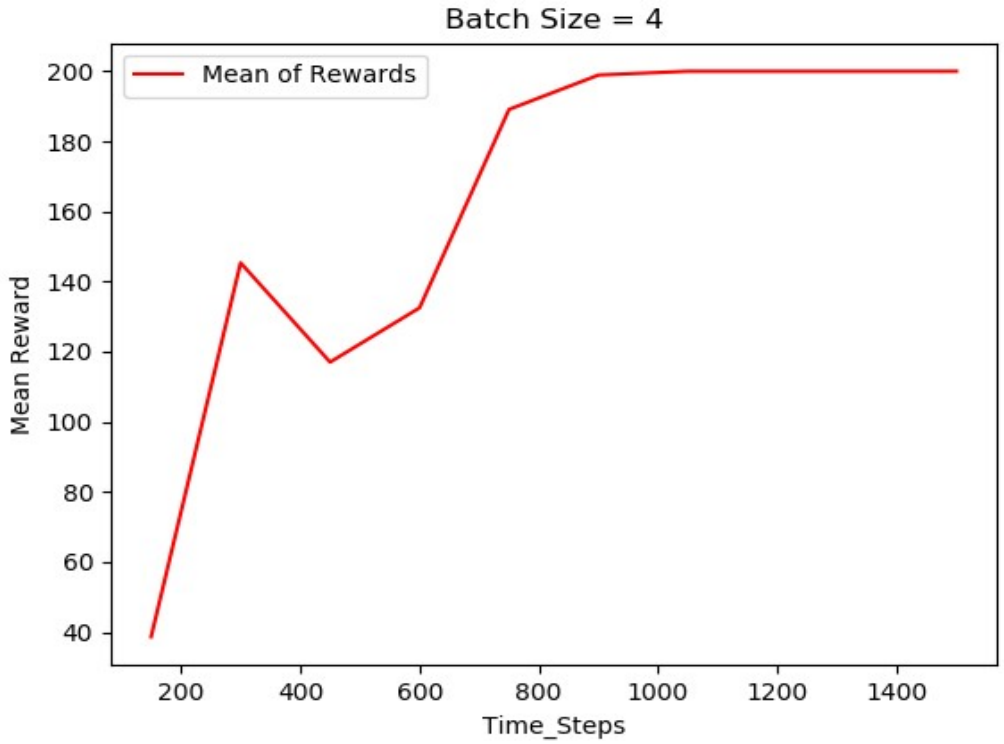

Figure 11

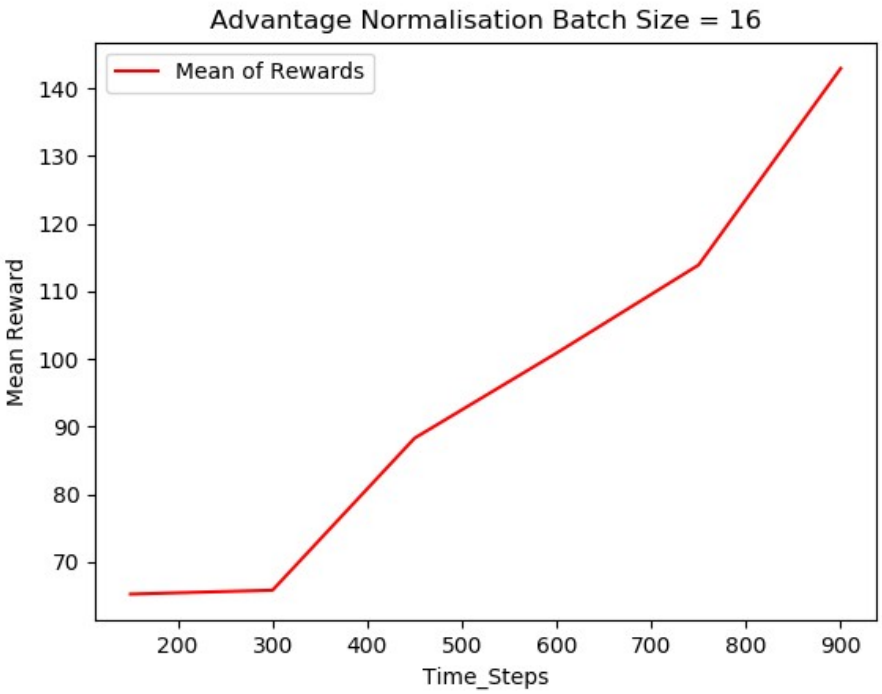

Figure 12

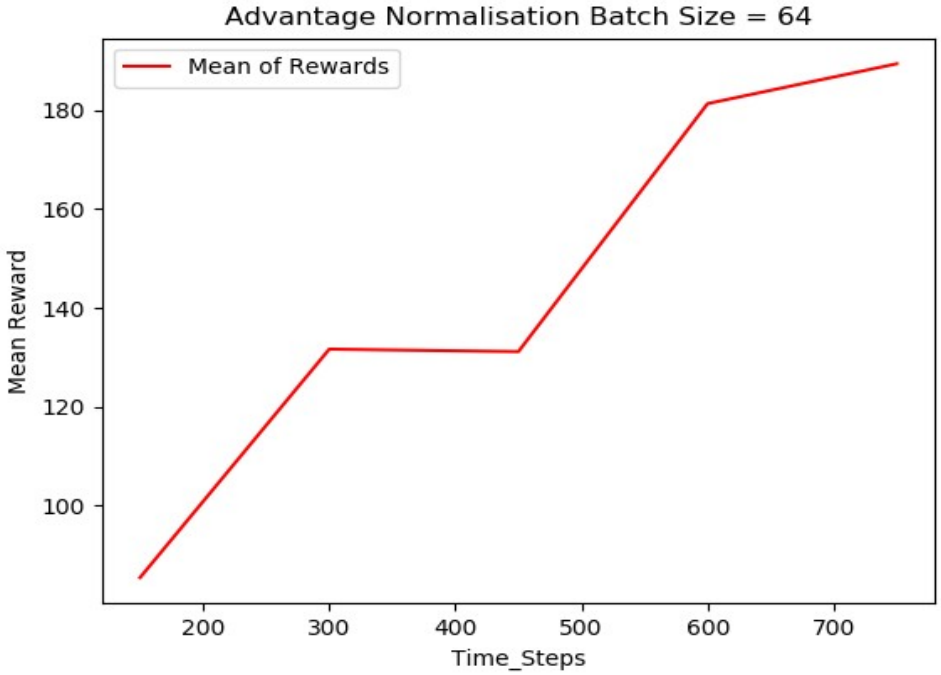

Figure 13

Question 3(d) :

3.(d) We know:

$$\nabla_\theta(J(\theta)) = E_{\pi_\theta}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t/s_t)\psi_t\right]$$

where,

$$\psi_t = \sum_{t'=t}^{\infty} \gamma^{t'} r_{t'+1} - b(s_{t'}) = q_{t:\infty} - b(s_t).$$

Now,

$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau)\,(r(\tau) - b)\right]$$

Also, we know that:

$$Var(x) = E(x^2) - E(x)^2$$

$\therefore \Rightarrow$ Var $= E_{\tau \sim \pi_{\theta(\tau)}}\left[\left(\nabla_\theta \log \pi_\theta(\tau)\,(r(\tau) - b)\right)^2\right]$

$$- E_{\tau \sim \pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau)\,(r(\tau) - b)\right]^2$$

We know that baselines are unbiased in expectation:

$$E_{\tau \sim \pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau)\,(r(\tau) - b)\right]^2$$

$$= E_{\tau \sim \pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau)\, r(\tau)\right]^2 = 0$$

$$\Rightarrow \text{Var} = E_{\tau \sim \pi_\theta(\tau)}\left[\left(\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)\right)^2\right]$$

Now, for minimum variance:

$$\frac{d(Var)}{db} = 0$$

$$\Rightarrow \frac{d(Var)}{db} = \frac{d}{db}\left(E[\nabla_\theta \log \pi_\theta(\tau)^2 (r(\tau) - b)^2] \right)$$

~~$$\Rightarrow 0 = \frac{d}{db}\left(E[\nabla_\theta \log \pi_\theta(\tau)^2 (r(\tau))^2] - 2bE[(\nabla_\theta \log \pi_\theta(\tau))^2] \right)$$~~

$$\Rightarrow 0 = -2E[(\nabla_\theta \log \pi_\theta(\tau)^2 r(\tau)] + 2bE[(\nabla_\theta \log \pi_\theta(\tau)^2)]$$

$$\Rightarrow b = \frac{E_{\pi_\theta}(\nabla_\theta \log \pi(a_t/s_t)^2 G_{t:\infty}(\tau)}{E_{\pi_\theta}(\nabla_\theta \log \pi(a_t/s_t)^2}$$

Hence, Proved !!!