CS6700 : Reinforcement Learning
Programming Assignment-3 Report
HRL and DQN

Name: Pragnesh Rana **Roll number:** ME17S301

- Part-1 is on Hierarchical Reinforcement Learning
- Part-2 is on Deep Reinforcement Learning

Contents

Hierarchical Reinforcement Learning

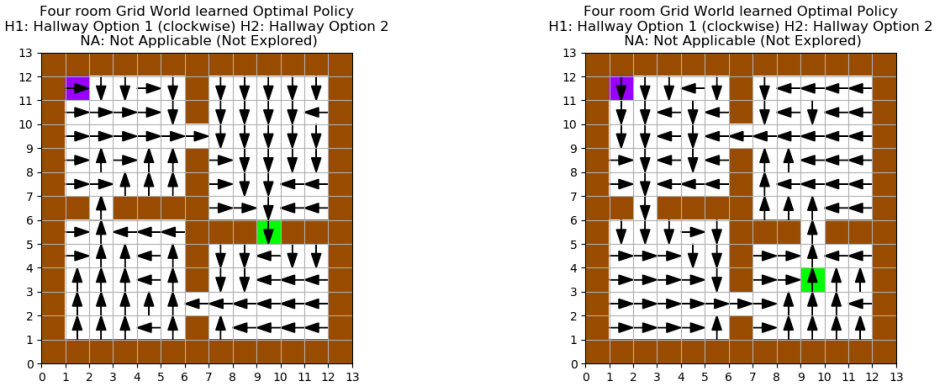
The SMDP and Intra option Q learning is used to solve the four room grid world problem. Let's start with brief introduction about the problem.

Answers-1: Grid World of Four Rooms and Visualization the learned Q values

The defined grid world is divided into four rooms. The upper left is room is define as Room-1. Numbering of room follows the clockwise notation. The agent is defined in upper left corner of room as given in fig-1 by blue colour cell. The brown colour indicates wall and green colour indicates terminal state as given in fig.-1. The study is conducted for two terminal state which is defined as G1 as in fig.-1(a) and G2 as in fig.-1(b). In the grid, each room has two hallways which can take agent from one room to another. The hall-way option follows policy π such that the agent get transferred to terminal state with shortest possible path and least possible obstacles.

For one move, agent is rewarded with 0. For terminal state reward is +1. With Pr $= \frac{2}{3}$, the agent take correct action and other actions are performed with Pr $= \frac{1}{9}$. The value used for discounted factor is $\gamma = 0.9$. Each hallway option has termination condition 0 for states lies within room

and 1 if outside. The initiation of state includes room as well as hallway. The initiation state is defined only inside the room which makes the world deterministic. To take agent from one room to another, there are two possible options. Option-1 follows the clockwise notation which take agent from room-1 to room-2. Option-2 follows anti-clockwise direction which can take agent from room-1 to room-4.

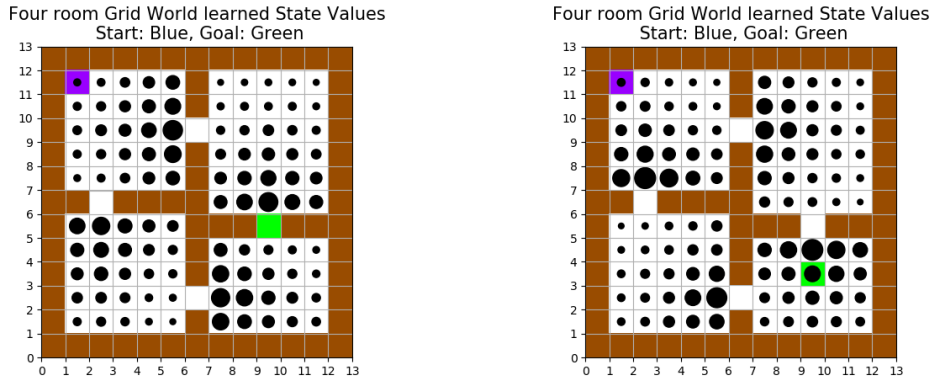


(a) Grid world with goal G1 (b) Grid world with goal G2

Figure 1: Grid world of four rooms. Blue:Agent, Green:Terminal The grid world-1(a) has terminal state G1 and grid world-1(b) has terminal state G2. Arrow indicates the optimal policy. The policy in fig-1(a) is obtained using option-1 where as same in fig.-1(b) by option-2

To solve the defined problem, SMDP and Intra-option Q learning is utilized. The optimal policy is the fundamental thing for used methods. To obtain the optimal policy using Q-learning, the initial states were randomly selected and goal is directed to hallway. likewise, eight policy is obtained. The state values are obtained using the maximum return.

$$V(s) = \underset{Q(s,a)}{\text{argmax}} \tag{1}$$



(a) State values with goal G1 by option-1 (b) State values with goal G2 by option-2

Figure 2: The grid world-2(a) has terminal state G1 and grid world-2(b) has terminal state G2. Circle size indicates the associated state value. The values in fig-2(a) is obtained using option-1 where as same in fig.-2(b) by option-2

The obtained optimal policy for goal-G1 and G2 is given in fig.-1. The visualization of state values are given in fig.-2. For goal both goals, higher state values are obtained near hallways and terminal state. High state values are indicated by bigger size of circle.

SMDP Q-learning :

Semi Markov Decision Processes are generalized MDPs, which allows policy maker to choose action according to change in state. It also provides the evolution of policy with continuous time while following arbitrary probability distribution. In short, SMDP follows the similar nature of MDP with options. Execution option starts with state $\mathscr{I}$ following policy π and jumps to terminating state s'. The SMDP Q-learning updates for option-value function is given by [1],

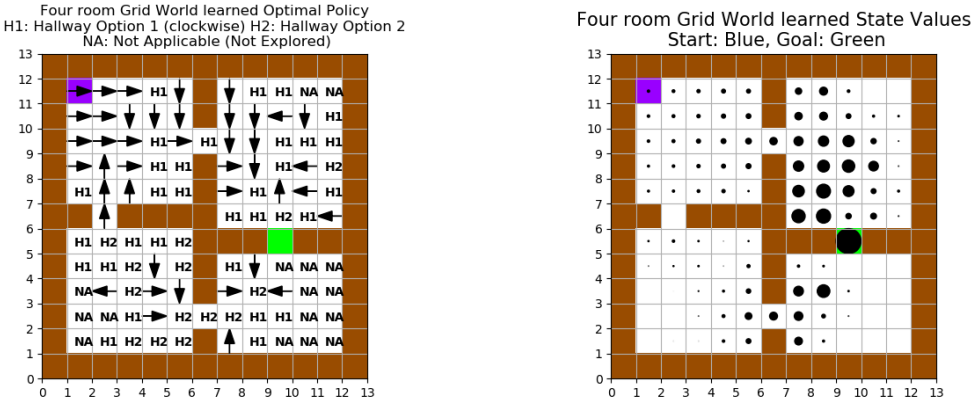
$$Q(s,o) = Q(s,o) + \alpha[r + \gamma^k \max_{o' \in s'} Q(s',o') - Q(s,o)] \tag{2}$$

where,
k - the number of time steps between s' and s
r - Cummulative discounted return over time

Observation for Goal-G1:

The learned optimal policy by SMDP-Q learning for goal-G1 is given in fig.-3(a). **The optimal policy is resultant of defined policy and primitive actions as well.** The primitive action are better than option away from goal and near the start state. Whereas, **near the terminal state Q-values are high and options play crucial role.**

Same can be observed from the state value diagram. **Near the goal state has higher valuer, which decrease as we move away from the goal.** Due to property of learning from experience and bound of wall near terminal state, there might be high chances of stumbling. Some of the states are not explored as it possible to reach toward goal by following any option from those states.



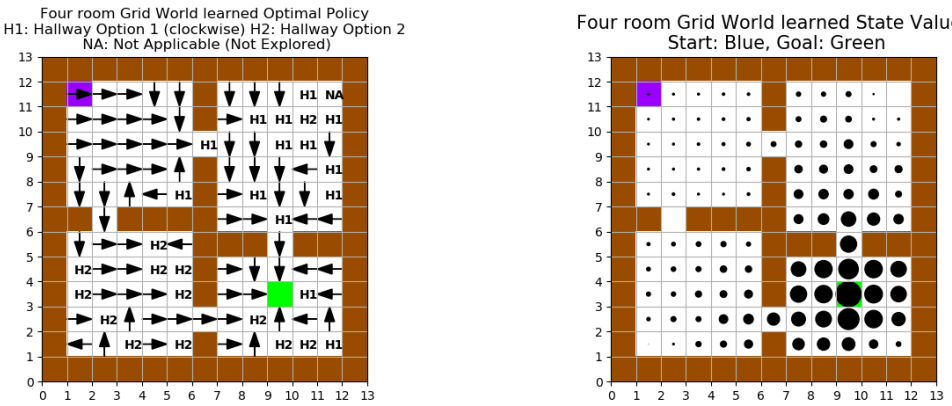
(a) Optimal policy by SMDP-Q

(b) State values for SMDP-Q

Figure 3: Grid world of four rooms. Blue:Agent, Green:Terminal. The optimal policy-3(a) and associated state values-3(b) for Goal-G1 after training of 10000 episodes.(By SMDP)

Observation for Goal-G2:

The optimal policy for goal-G2 is given in fig.4(a). For this case also, states near the goal selects primitive action over policy as it helps agent to move away from the hallways and direct it towards goal whereas, states away from the goal follows actions. From figure-4, it is clear that SMDP Q-learning has obtained optimal policy using **mix of primitive action as well as options.**



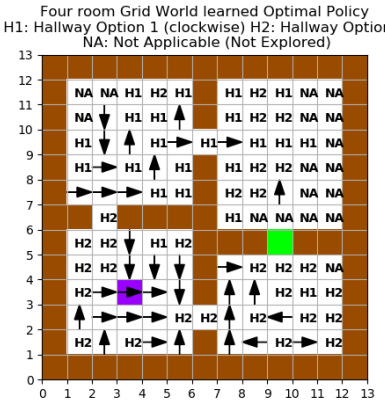
(a) Optimal policy by SMDP-Q

(b) State values for SMDP-Q

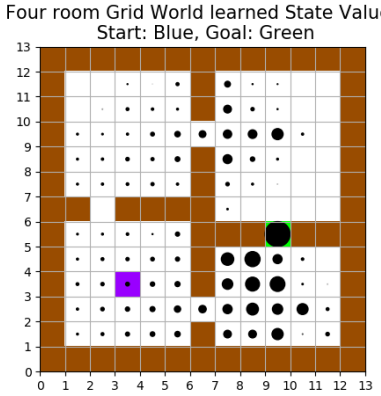
Figure 4: Grid world of four rooms. Blue:Agent, Green:Terminal. The optimal policy-4(a) and associated state values-4(b) for Goal-G2 after training of 10000 episodes.(By SMDP)

For goal-2 compared to goal-1, the major states follow primitive actions where as options are dominant in case of goal-G1. Mainly for room-1 primitive actions plays major role, whereas for room 2 and 4 it requires both. **Near terminal state, dominance of primitive actions make algorithm faster by increasing the learning rate.**

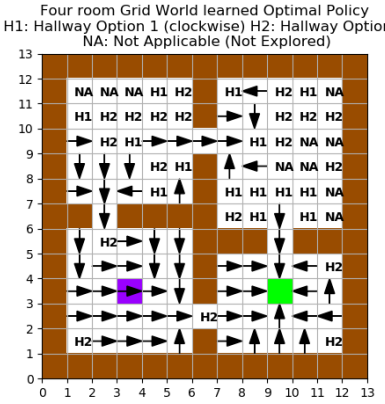
Answers-2 : Changed initial state to the centre of room 4



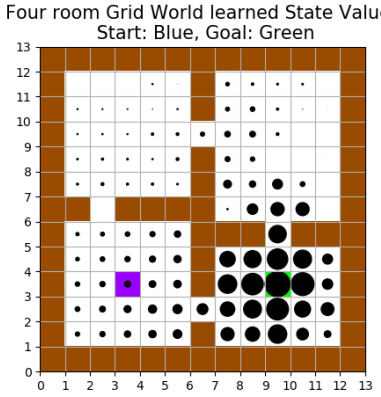
(a) Optimal policy for goal G1



(b) State values for goal G1



(c) Optimal policy for goal G2



(d) State values for goal G2

Figure 5: Grid world of four rooms. Blue:Agent, Green:Terminal. The optimal policy-5(a) and associated state values-5(b) for Goal-G1 obtained after training of 10000 episodes. Same way, the optimal policy-5(c) and associated state values-5(d) for Goal-G2 (By SMDP)

Observations:

For the same terminal state, initial state is directed to center of room-4. **The change in state also causes the change in value function and optimal policy.** The optimal policy obtained in these cases are also mixtures of options as well as primitive actions.

Change of state does not affect the role of primitive action and policy near the terminal states as well as far away from it. The major difference in both scenario (policy for same goal and change of initial state) is change in state value function. By comparing the fig.-3(a) and fig.-5(a), it is clear that the **state value of adjutant room of terminal and start state has higher state values. Change of the initial state directly affect the optimal policy, which varies in both scenario and finds the best possible route using primitive action and options.**

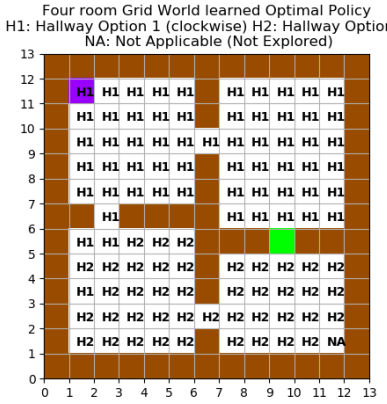
The value obtained in case of goal G2 is higher and shows nature of gradient. As goal-G1 is constrained by two wall same varies lot in this case. **Options play vital role near terminal state of goal-G1, whereas, primitive actions play important role in case of goal-G2.**

Bonus Answers-3: Intra-option Q learning

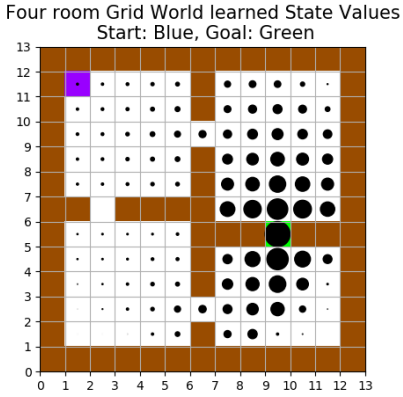
Intra option Q learning learns from the execution of one policy while following another behavioral policy. It is an off policy algorithm. The downside about SMDP is that it has to execute a termination condition and due to that, only single option is applied at time. This limitation is removed in Intra-option Q learning. It can learn different options at time without prior execution of option from same experience. The update rule for intra-option Q learning is given in as below [2],

$$\begin{aligned} Q(s,0) &= Q(s,o) + \alpha(s,0)[r + \gamma\hat{Q}(s',o) - Q(s,o)] \\ \hat{Q}(s,0) &= (1-\beta(s'))Q(s',o) + \beta(s')\max_{o'\in s'} Q(s',o') \end{aligned} \tag{3}$$

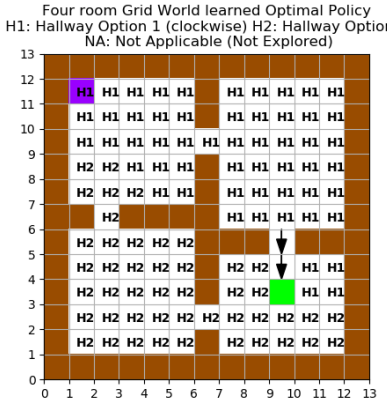
For Intial State in Upper Left Corner:



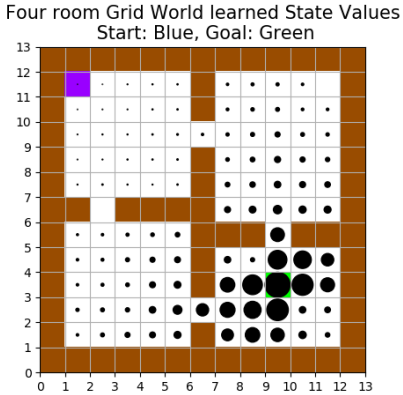
(a) Optimal policy for goal G1



(b) State values for goal G1



(c) Optimal policy for goal G2



(d) State values for goal G2

Figure 6: Grid world of four rooms. Blue:Agent, Green:Terminal. The optimal policy-6(a) and associated state values-6(b) for Goal-G1 after training of 10000 episodes. Same way, the optimal policy-6(c) and associated state values-6(d) for Goal-G2 (By Intra option Q learning)

Observation and Improvement:

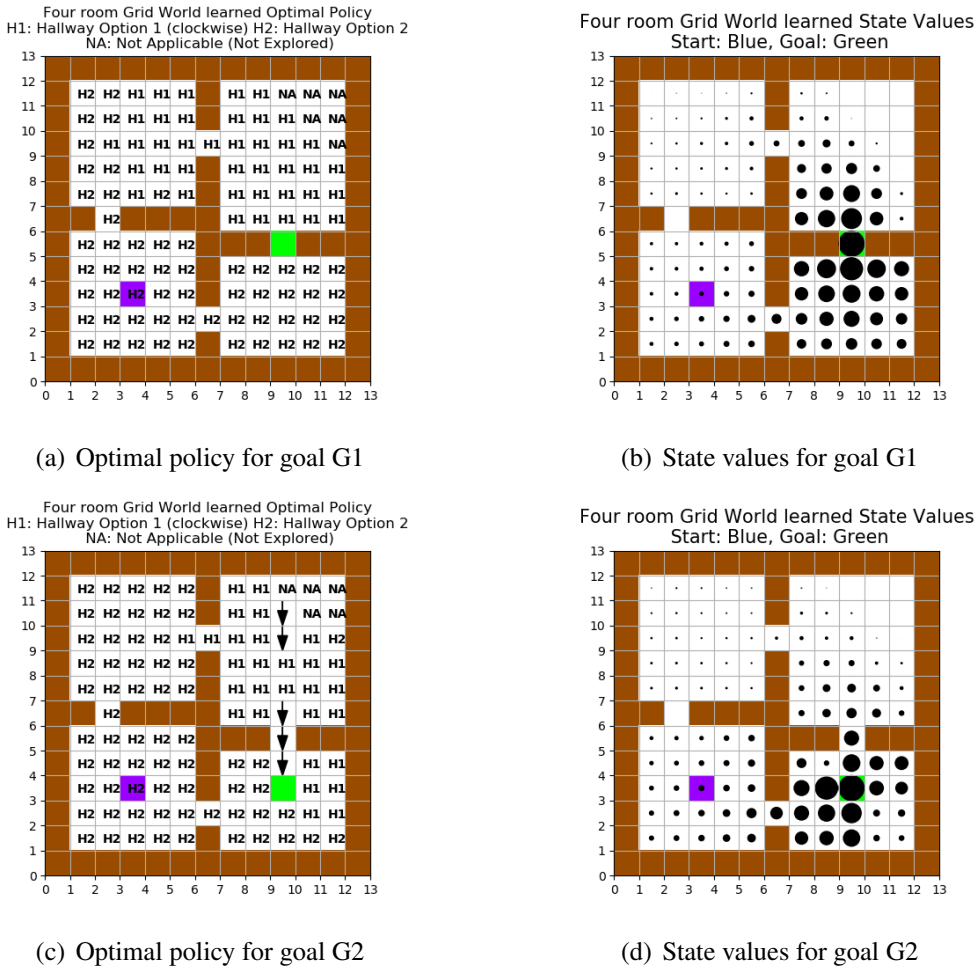
The intra-option Q learning, learns most possible correct options. It can be observed that, it **only uses correct options to reach the goal.** The optimal policy does not change by learning from

more episodes. By comparing result of SMDP and intra-option, it is clear that for 10000 episodes intra-option Q learning outperforms the SMDP. Even around **1000 episodes are enough to give similar result in case of intra-opion Q learning.**

From figure-6, it can be seen that for **goal-G1 optimal policy is learned only using correct options rather than primitive actions.** For goal-G2, **it is not using any primitive action till old state of goal-G1 as the agent has learned to reach the goal previously. After that, it just take primitive action from near goal location.**

For Initial State in the center of room-4 :

Similar observation has been made by transferring the initial state to center of the room-4 as given in fig.-7. **For goal-G1 it has only obtained the optimal policy using options. Once it has learned to reach that state then change of goal is attained by primitive actions.**



(a) Optimal policy for goal G1

(b) State values for goal G1

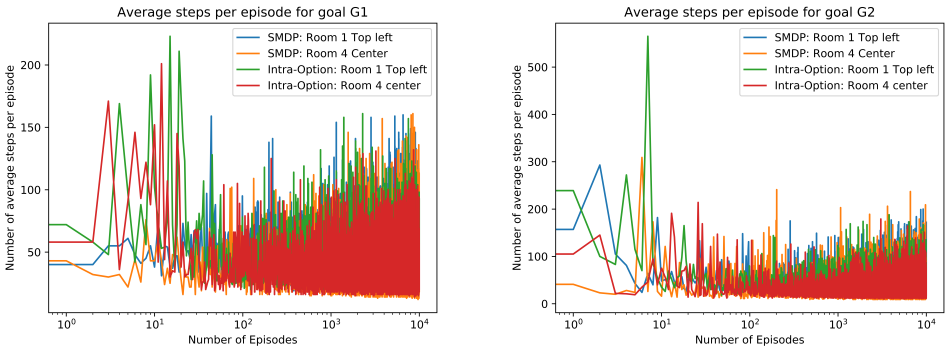
(c) Optimal policy for goal G2

(d) State values for goal G2

Figure 7: Grid world of four rooms. Blue:Agent, Green:Terminal. The optimal policy-7(a) and associated state values-7(b) for Goal-G1 after training of 10000 episodes. Same way, the optimal policy-7(c) and associated state values-7(d) for Goal-G2 (By Intra option Q learning)

Learning Curves of Average Return and Steps :

For all the possible cases comparative graphs of average return and average steps is mentioned here.



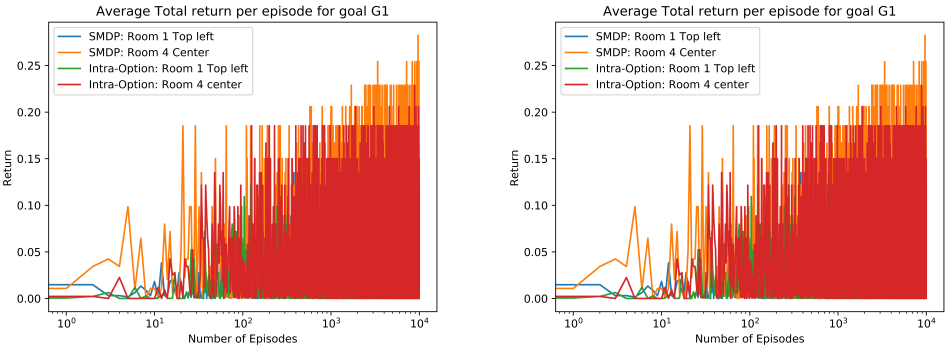
(a) Number of average steps for goal G1 (b) Number of average steps for goal G2

Figure 8: Total number of average steps in case of initiation of agent at different states for SMDP and Intra-Option Q learning for goal-G1 8(a) and for goal-G2-8(b)

As given in fig.-8(a) in case for goal G1 for SMDP, initially as it does the exploration and takes more steps to reach the goal. SMDP takes on average 50 steps per episode which goes to 75 steps and almost remains constant. Whereas, **when initial state is shifted to center of the room-4, the number of average steps taken are slightly lesser in case of SMDP.**

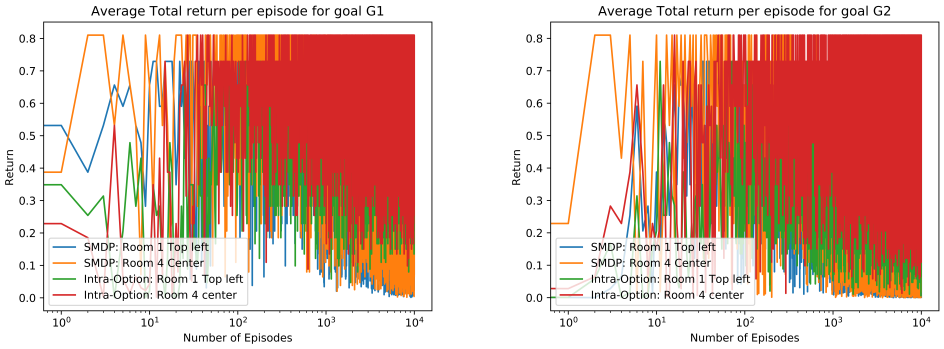
Intra option Q learning has tendency to learn from applicable options of past experience so **initially due to exploration, intra option takes slightly more steps then SMDP.** But once it has learned the policy, **later the average steps comes down significantly. Further it only requires around half amount of average steps in learning the policy.**

Similar observation can be made for goal-G2. **Initially average steps taken by intra-option method is almost double but with increase in number of episodes intra option learns faster and outperforms the SMDP.**



(a) Total return vs epsiode for goal G1 (b) Total return vs epsiode for goal G2

Figure 9: Average total return by SMDP and Intra-option Q learning for goal-G1 8(a) and for goal-G2-8(b). Each primitive action denotes the single step. The average is taken over 10 episodes and x-axis follows log 10 scale.



(a) Total return vs epsiode for goal G1 (b) Total return vs epsiode for goal G2

Figure 10: Average total return by SMDP and Intra-option Q learning for goal-G1 8(a) and for goal-G2-8(b). Each option denotes the single step. The average is taken over 10 episodes and x-axis follows log 10 scale.

From fig.-9 and fig.-10, it can be observed that **the average reward is higher for the case of Intra-option Q learning compared to SMDP. Variations in average reward is also slightly lower in case of intra-option Q leaning for the case of goal G1 and G2.**

Deep Reinforcement Learning:

The downside of the reinforcement learning is instability for non-linear function. In some cases, due to correlation in the observations, small change in Q value may adversely affect the target values and policy. Use of neural network with reinforcement learning to represent Q values can also diverge the learning.

To improve the stability of reinforcement learning with neural network, **deep neural network(CNN)** can be used **with experience reply.**[3] [4] [5] The role of experience reply is to reduce the correlation in the training set, by which learning can be smooth and variance can be reduced. Along with experience reply **target network** also play major role in stabilization of learning. Rather than computing the gradient of value function, target network can be used for generating $\hat{Q}$ from cloning of Q which can be used for learning targets y_j. This method can reduce the correlation.

The loss function for DQn can be define as,

$$L_i(\theta_i) = Es, a, r, s'[(r + \gamma \max_{a'} Q(s', a', \theta_i) - Q(ss, a, \theta_i)^2] \tag{4}$$

Answers-1: Hyper-parameters and Learning Curve

[6]

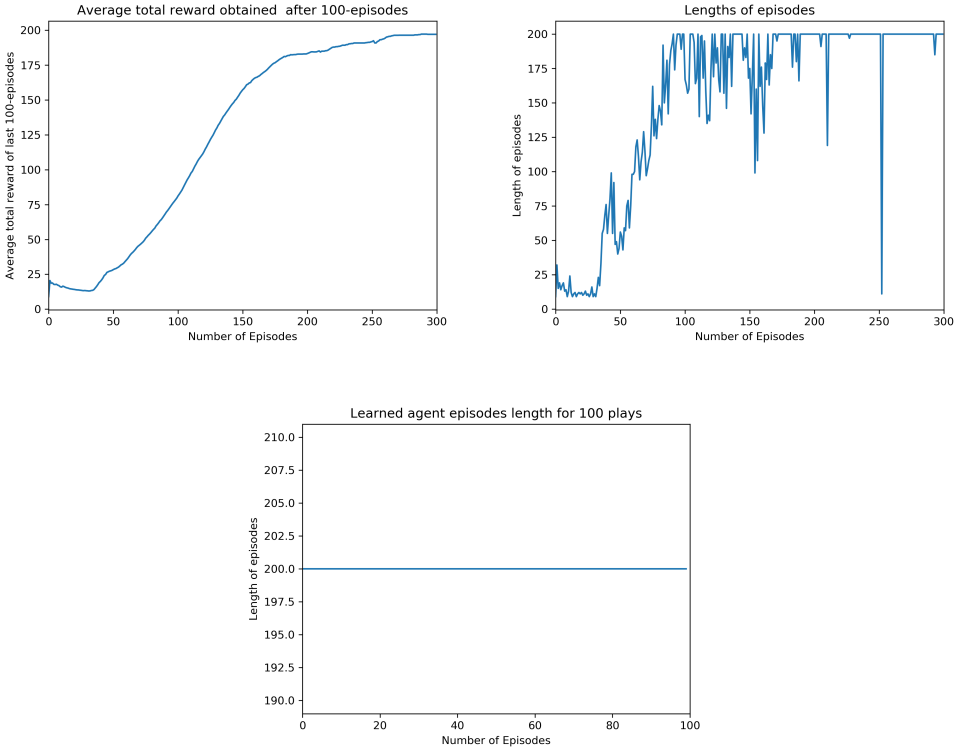


Figure 11: Plots of Average total reward of last 100 episode 11(a), episode length 11(b), learned agent episode length 11(c)

The goal is to train the DQN such that it the **average reward for cart-pole should be more**

than 200 which is obtain as given in fig.-11(a). Figure-11(a) indicates that performance agent is poor in initial stage due to set of ε=1 and slow decay rate 0.995. After certain period of exploration, agent learned the optimal policy which can be concluded from figure-11(a). **Till 180 episodes maximum reward obtained is 175, whereas after around 200 episodes agent has learned the optimal policy and has obtained reward around 190 to 200 for next 100 episodes.** Similar observation can be concluded from the length of episode plot.

Around 50 episodes there is lot of fluctuation in episode length which reduces till 200. **After 200 episode there is almost no fluctuation in result which is indication of learned optimal policy.** Figure-11(c) indicates the episode length after 200 episodes. **The constant value of episode length is indication of learned optimal policy.**

Answers-2: Report of hyper-parameters tuning

The set of hyper parameters which worked best for cart-pole problem are given in table-1.

Hyperparameters	**Value**	**Description**
Initial Epsilon	1	Start value of ε for exploration
Epsilon Decay Rate	0.995	Decay Multiplier of ε
Discount Factor	0.99	Discount factor for Q learning
Max Steps	200	Maximum Number of steps for each episodes
Episode Num	300	Count of Number of episodes for traning
Target Update Freq	200	Number of steps after which target network has to be upated
Minibatch Size	32	Size of batch samples taken from experience reply
Learning rate	0.001	Learning rate Adam Solver
Reply Memory Size	10000	Size of reply memory
Regularization Factor	0.0001	Regularization factor for overfitting
Min Epsilon	0.01	Minimum value ε can achieve after decay
Hidden1 Size	20	Size of neural network for Hidden layer - 1
Hidden2 Size	20	Size of neural network for Hidden layer - 2
Hidden3 Size	20	Size of neural network for Hidden layer - 3

Table 1: Set of Hyperparameters

Answers-3: Report of the variation of hyper-parameters like hidden layer sizes, epsilon, mini-batch size, target frequency.

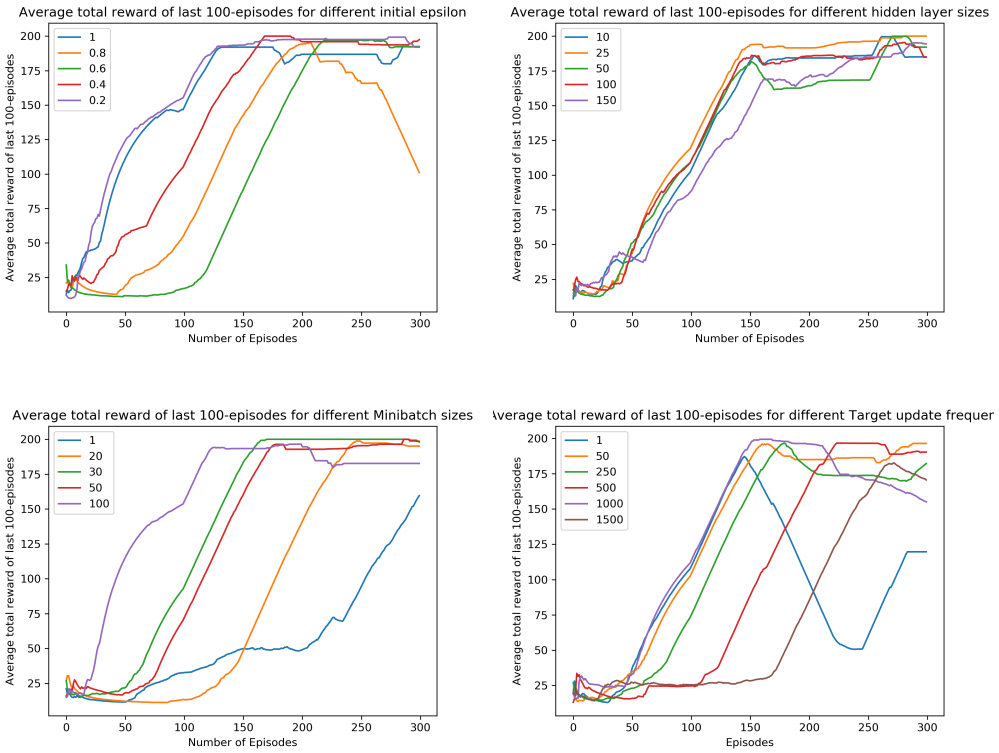


Figure 12: Comparative plots of learning curves of different hyper parameters : Initial value of epsilon-12(a), Hidden layers-12(b), size of minibatch-12(c), update frequency of target networks-12(d)

The comparative plots of different hyper-parameters and its different values are given in fig.-12. The initial value of ε is selected as **{1, 0.8, 0.6, 0.4, 0.2}** and its result is given in fig.-12(a). For very high value(1) and small value (0.2), the agent learns policy faster and its gains the average reward of 200. **For high value of ε, agent does much exploration which helps to learn optimal policy faster. Whereas, for very small value of ε, the greedy behavior also helps to learn optimal policy faster.**

But in **both cases transition has fluctuation.** For value of 0.8, due to exploration there is slight decline is average reward. For case of 0.4 result has slight fluctuation. But, **in case of $\varepsilon = 0.6$ the average reward transition is intially slow due to exploration but slowly it stabilize and gives maximum average reward which gives overall smooth transition in the result.**

The comparison plot of average total reward and its different values of hidden layer is given in fig-12(b). The different values used for hidden layers are $\{10, 20, 50, 100, 150\}$. **For smaller values of hidden layers < 50, agent can learn policy faster as required less data.** But as hidden layer increases to ≥ 100 it also learns policy at almost same rate but with less fluctuations. In general if policy is s**imple, then fewer hidden layers** are advisable to use it helps to learn faster as well as it also reduces over-fitting. For **complex problem, more hidden layers** are advisable.

The result of minibatch size is given in fig.-12(c). Sizes of used batches are {1,20,30,50,100}. The result shows that for very small value of minibatch average reward is very small. For very high

value of mini batch, average reward per episode increases suddenly with fluctuation. Whereas, for value around 20 transition is very slow. **For value around 30 transition is moderate and smooth, which again become fluctuative with increment in value. So intermediate minibatch size value of 30 is ideal.**

For target updates, {1,50,250,500,1000,1500} are the values used for network frequency. The comparative result is given in fig.-12(d). For very small value of network frequency, after certain episodes there is decline in average reward. For very high value of frequency >500, there is very less variation in learning curve due to correlation. In these case, it take much time to learn optimal policy. **For intermediate values of** $50 \leq x \leq 250$**, the average reward per episode is quite high and variation in learning curve is also less. In short , intermediate value of frequency is more suitable.**

The variation in episode length also supports the parameter discussed earlier. The ideal parameter mentioned above in all cases have very less fluctuation in average length of episodes as mentioned in fig.-13

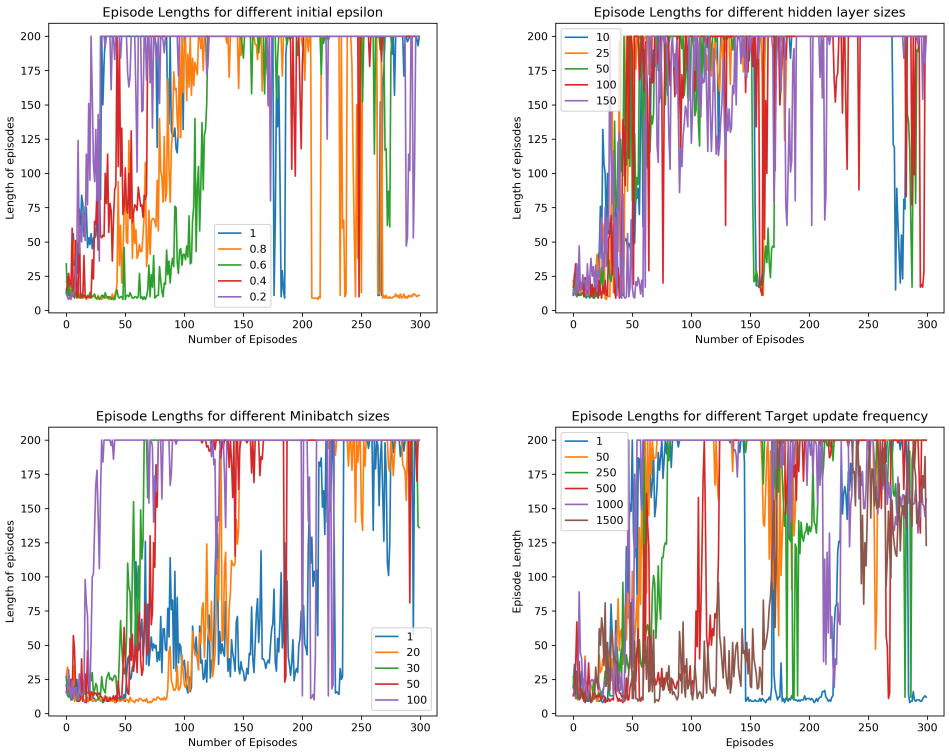


Figure 13: Comparative Plots of episode length for different hyper parameters : Initial value of epsilon-12(a), Hidden layers-12(b), size of minibatch-12(c), update frequency of target networks-12(d)

Bonus Answers-4: Observations and inferences of removal of the experience replay and/or the target network

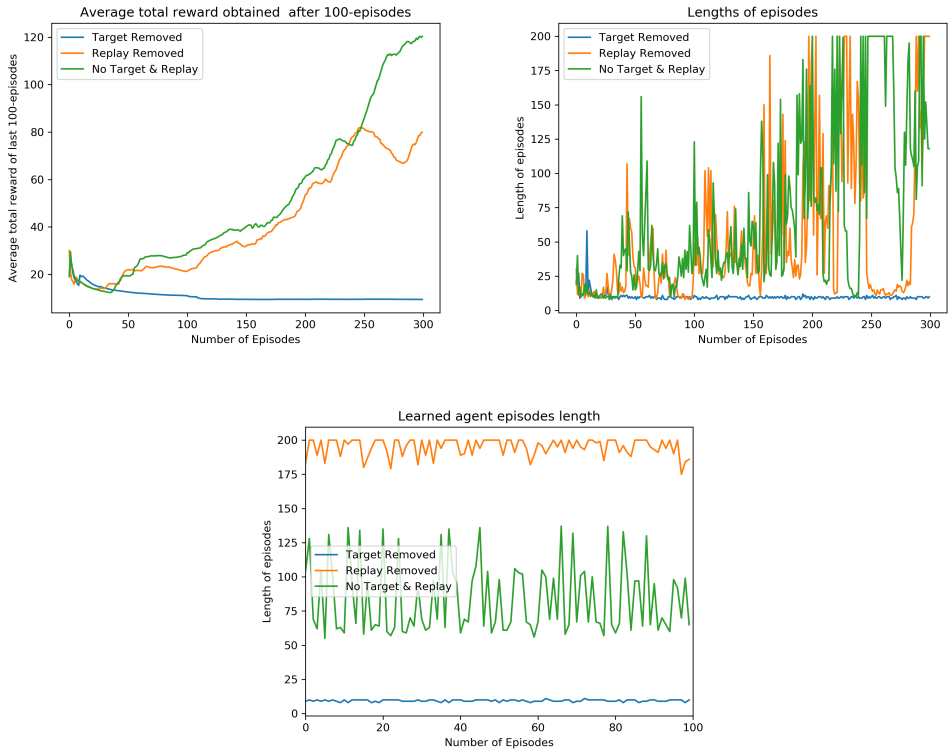


Figure 14: Plots of average reward-14(a), length of episodes-14(c), episode length for leaned agent-14(b) after removing the target network, experience reply and both together.

The **removal of target network** by keeping the reply memory computes the target from samples received from reply. It **may cause the high correlation** and can **make network unstable**. Updation of Q-values by **removal of only reply memory force to use current observation** which is higly correlated and makes **learning slower**. The stable average reward is around 80 in case of removal of reply memory whereas, removal of only targets gives comparatively better results with average reward of 120. **Removal of both targets as well as reply memory makes whole process very slow** and average reward is also very poor.

Similar observation can be made for episode length as given in fig.-14(c). **Removal of both generates very small episode length, which ultimately helps in fast learning of optimal policy.** Removal of reply memory causes high variance in the episode length due to correlated observations.

Use of reply memory and target network **both has high value of episode length for trained agent which is around 200.** From fig-14(b), it can be observed that **removal of reply does not affect much** whereas, removal of target reduce the episode length by half and **removal of both generates ideal result.**

References:

[1] R. S. Sutton, "Between mdps and semi-mdps: Learning, planning, and representing knowledge at multiple temporal scales," 1998.

[2] R. S. Sutton, D. Precup, and S. P. Singh, "Intra-option learning about temporally abstract actions.," in *ICML*, vol. 98, pp. 556–564, 1998.

[3] "Rl course by david silver - youtube,"

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[5] *Reinforcement learning: An introduction*.

[6] "Gridworldenvs - https://github.com/opocaj92/gridworldenvs,"