Programming Assignment 3

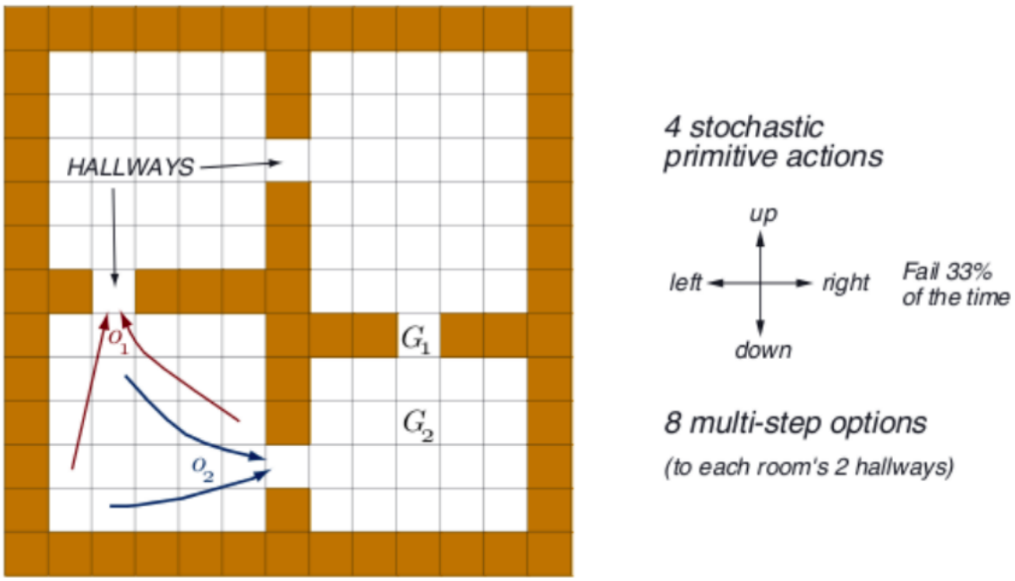
CS16B003

[a] *Indian Institute of Technology Madras*

Keyword: HRL, DQN, SMDP, Options, OpenAI Gym, Tensorflow

Abstract: This paper presents the solutions to the third programming assignment of Reinforcement Learning (CS6700) at *IIT Madras*. All notations used are as accordance with the textbook Reinforcement Learning by S. Sutton and G. Barto

PROBLEM 1



a) Starting state - Room 1 (Random pos)

I. Goal G1

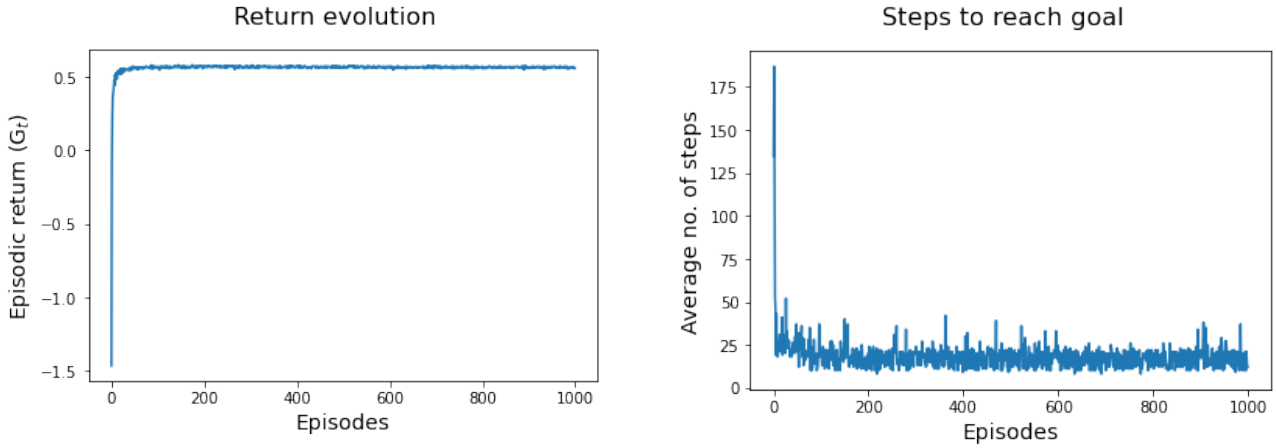


Fig 2: Performance of SMDPQ to achieve goal G1 from R1

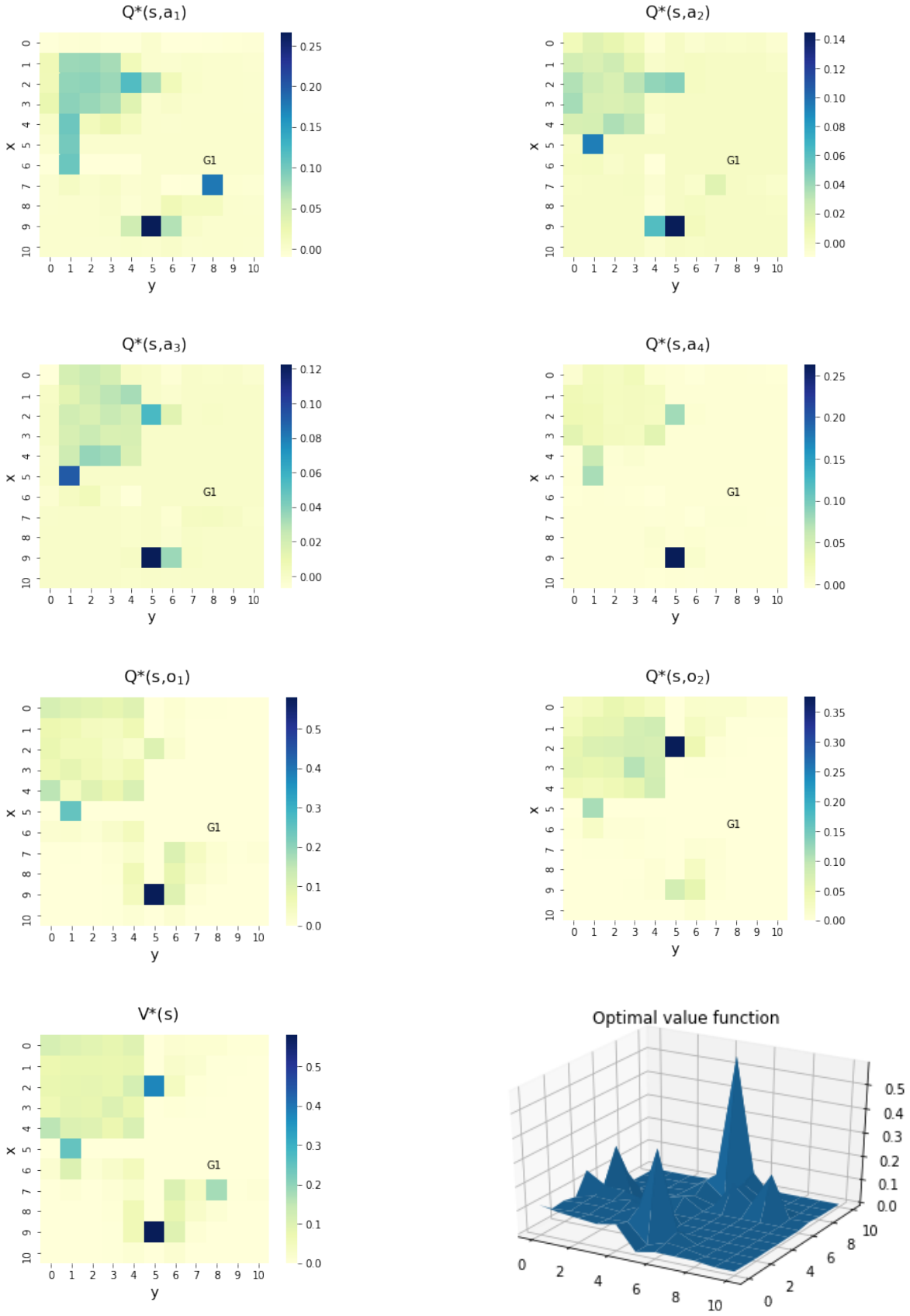


Fig 3: Optimal value functions (R1-G1)

II. Goal G2

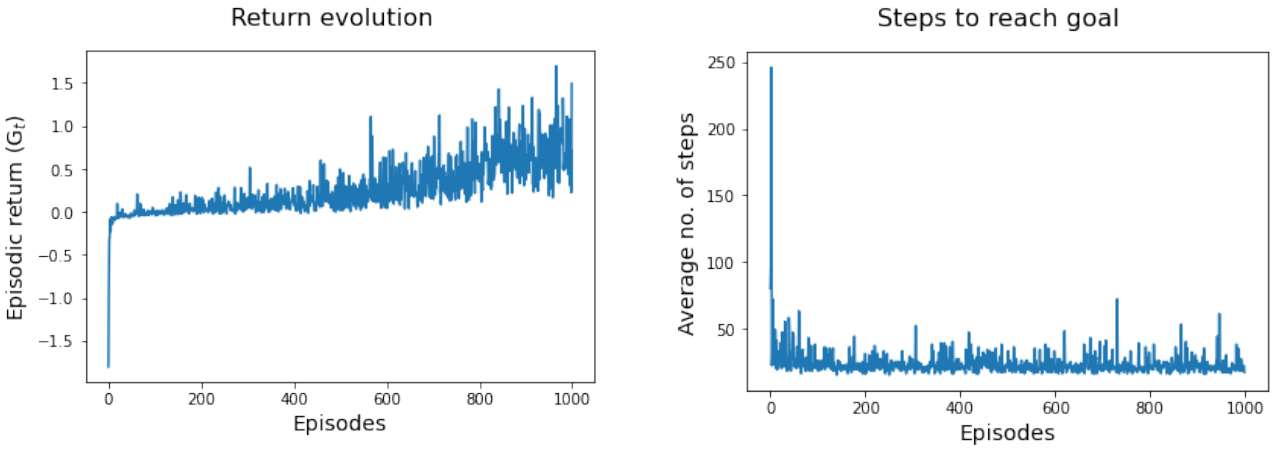
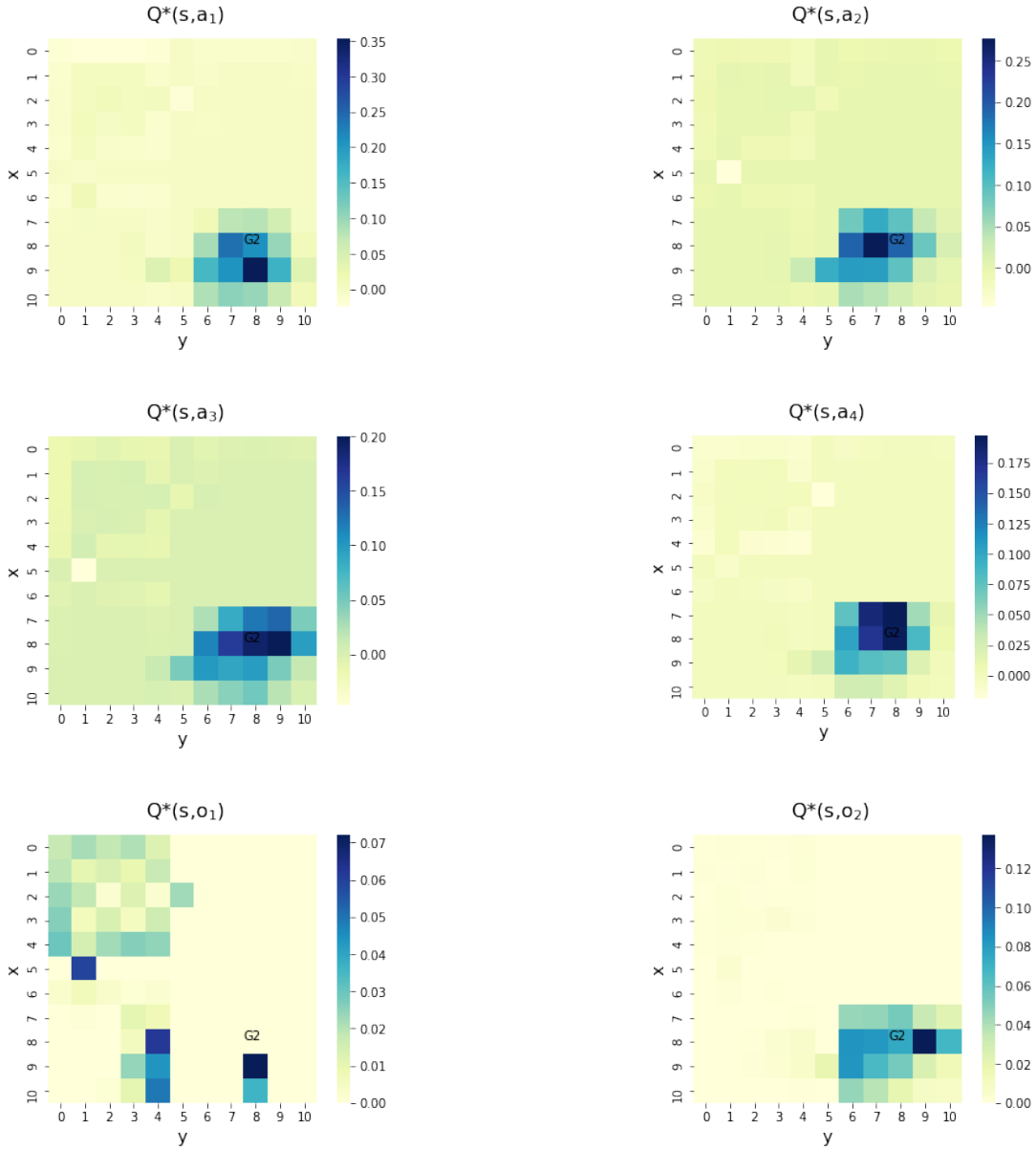


Fig 4: Performance of SMDPQ to achieve goal G2 from R1



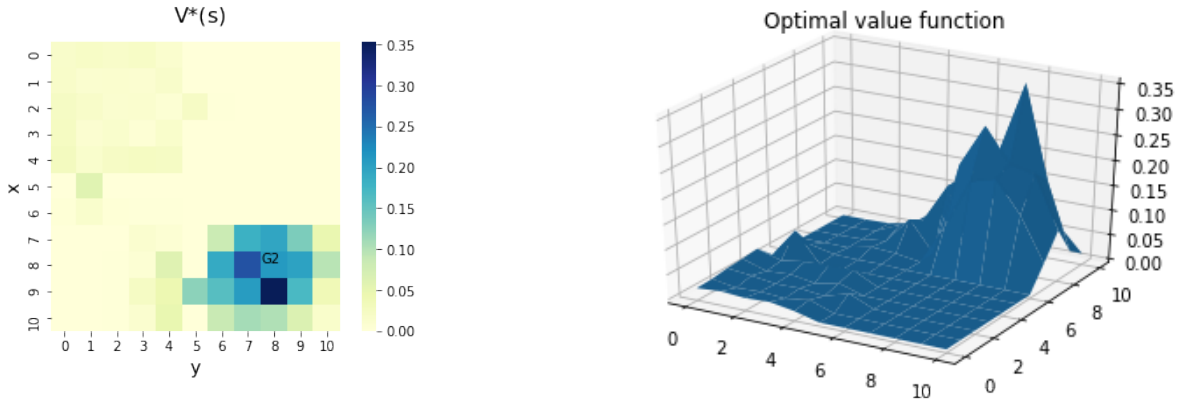


Fig 5: Optimal value functions (R1-G2)

b) <u>Starting state - Room 4</u> (fixed)

I. Goal G1

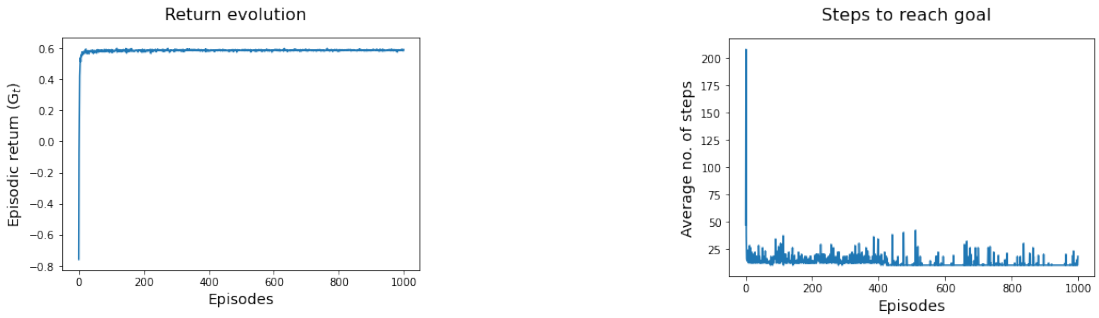
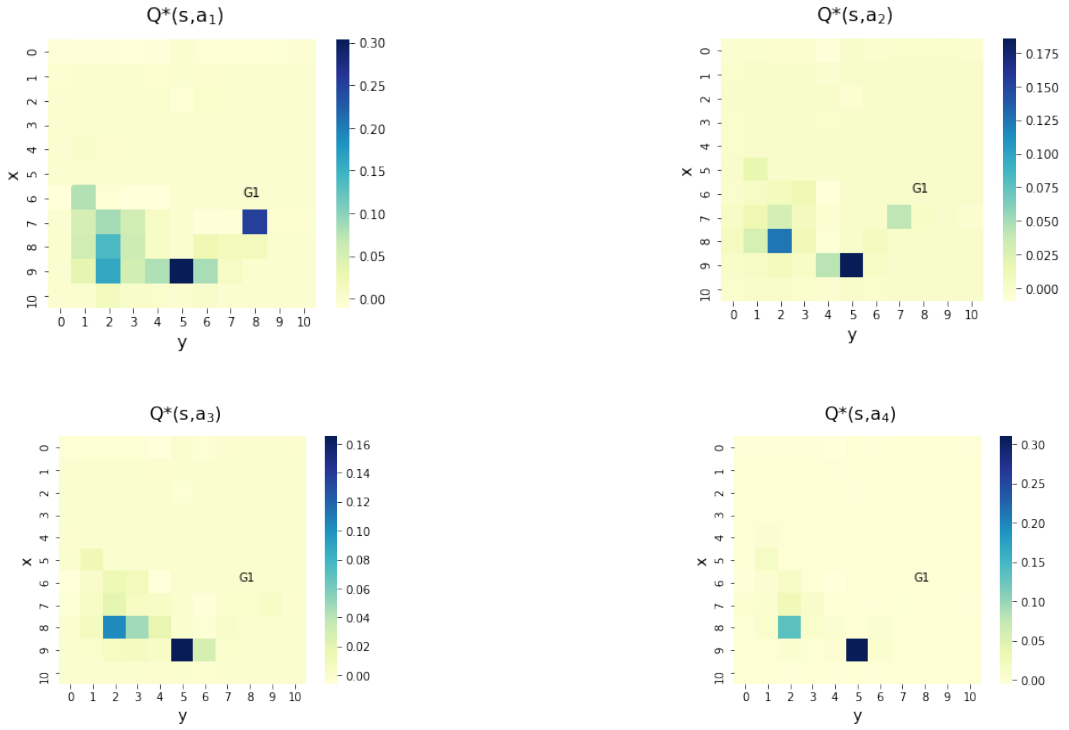


Fig 6: Performance of SMDPQ to achieve goal G1 from R4



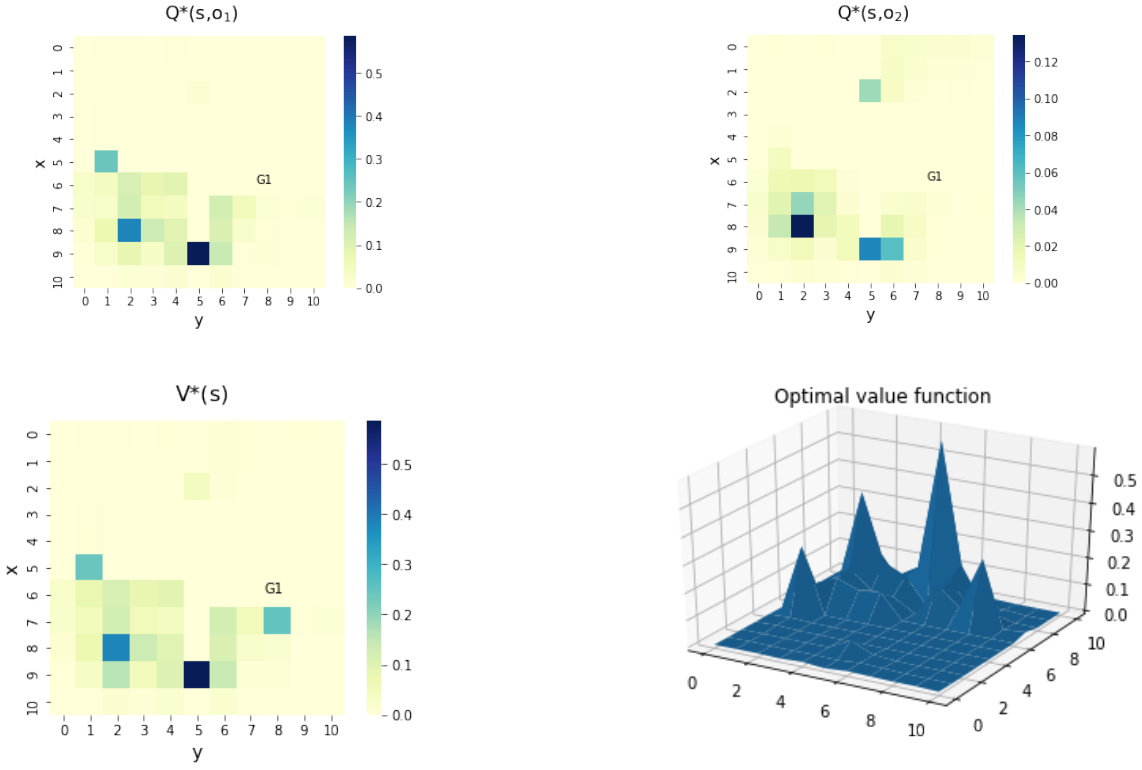


Fig 7: Optimal value functions (R4-G1)

II. Goal G2

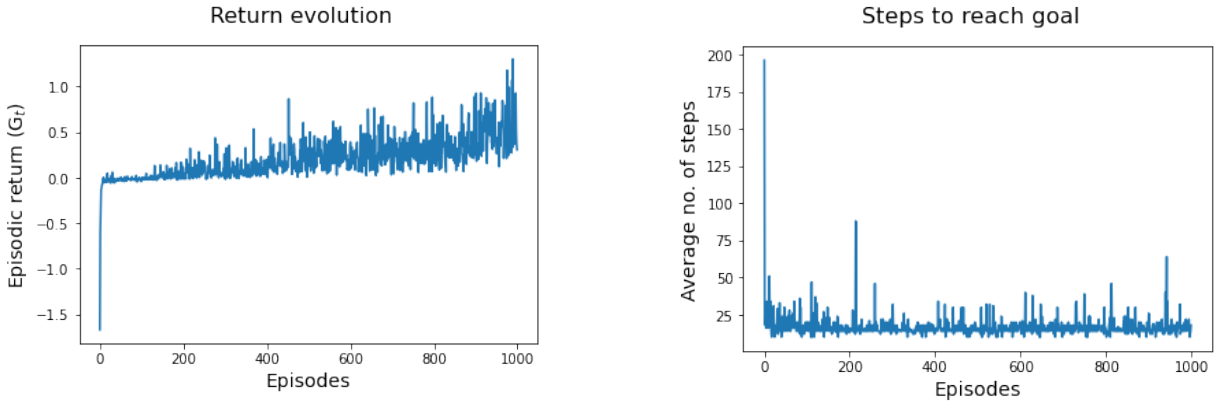
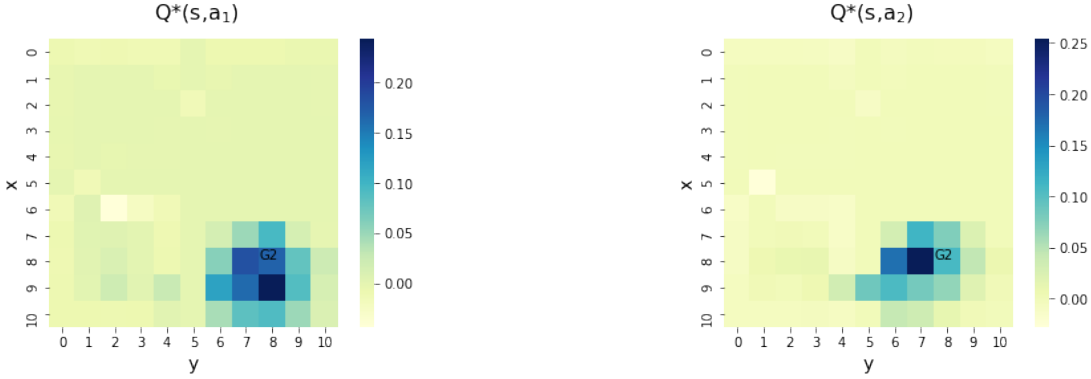


Fig 8: Performance of SMDPQ to achieve goal G2 from R4



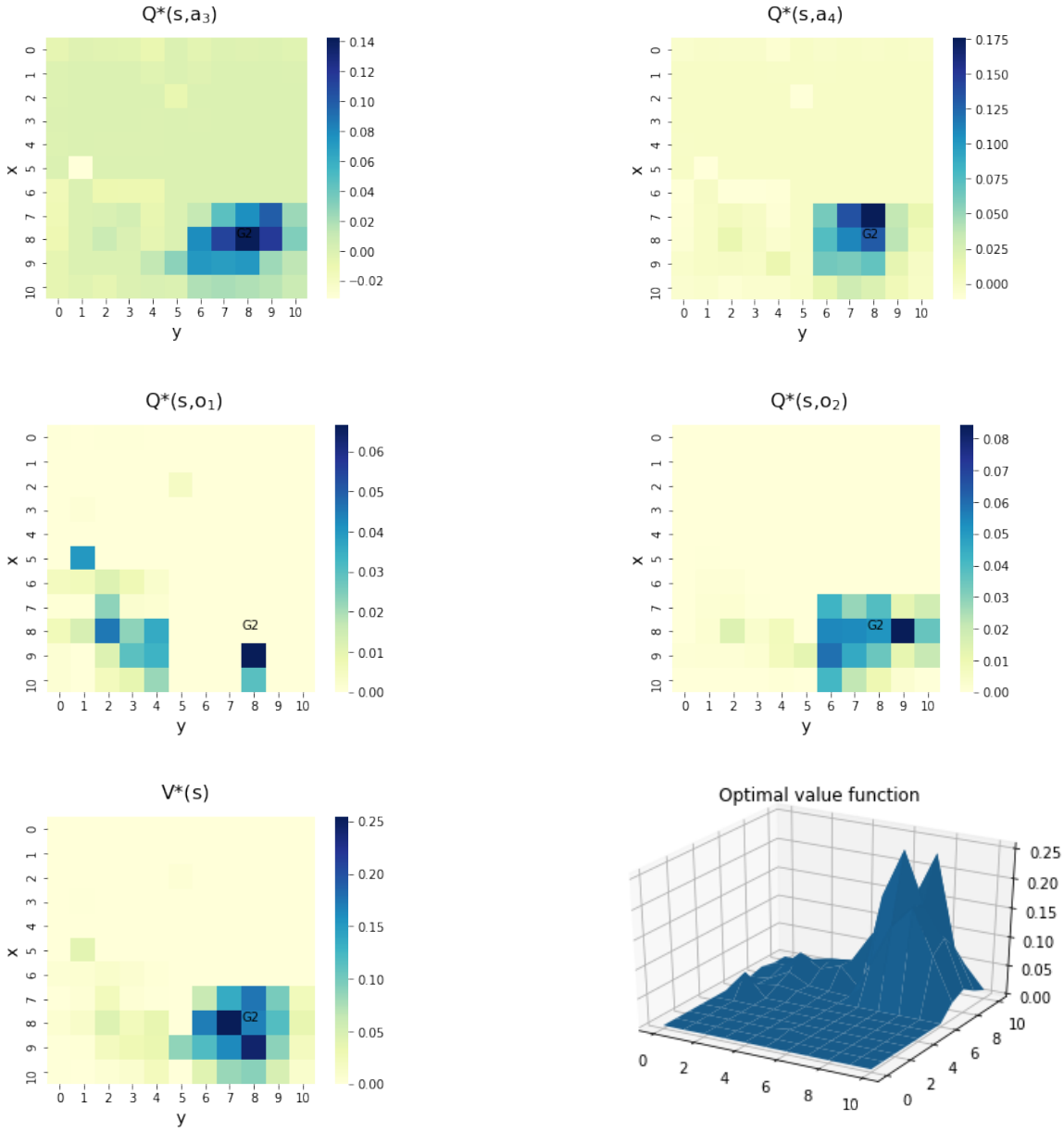


Fig 9: Optimal value functions (R4-G2)

Discussion

- Options are useful to quickly navigate between hallway states. When the goal is at a hallway state such as G1, the hallway options dominate as they quickly lead the agent to goal state. However, when the goal is at a non-hallway state such as G2, the optimal option for most of the states are primitive actions. Therefore, options are not always better than primitive actions. It depends on the structure of the environment, initiation set and the target state.
- When the starting position is **fixed** at the center of room 4, learning is slighly faster as initial state is closer to the goal (based on avg. return and avg step plots), but the Q values are not learnt properly. This is attributed to the fact that since the initial state is fixed, and as options update only the Q values of the starting state, there are jumps in the Q function. To speed up learning as well as arrive at convergence to the appropriate Q values, we attempt intra-option QLearning in the subsequent section.
- *Implementation note:* Stochasticity is enabled only for primitive actions. Options are comprised of only deterministic actions as the purpose of an option is to reach the target hallway as quickly as possible.

c) Intra-Option QLearning (R4, goal G1)

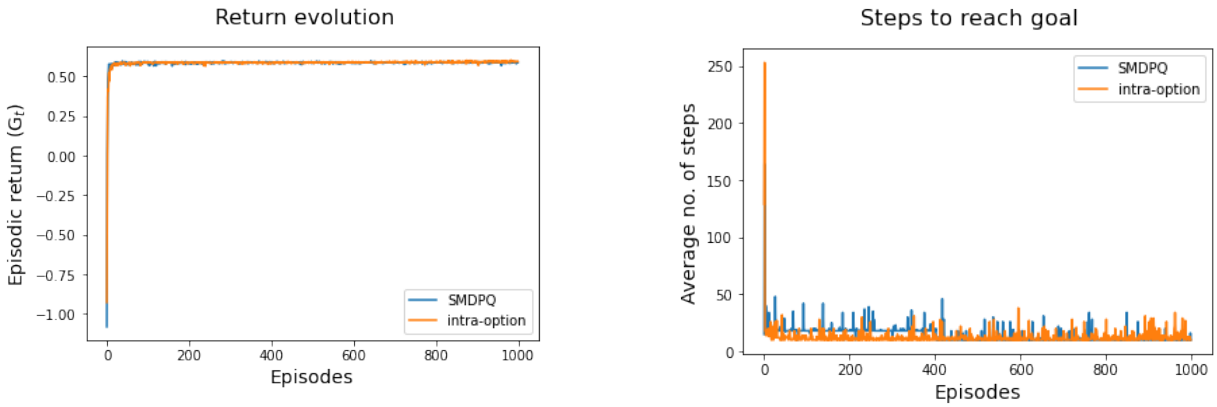
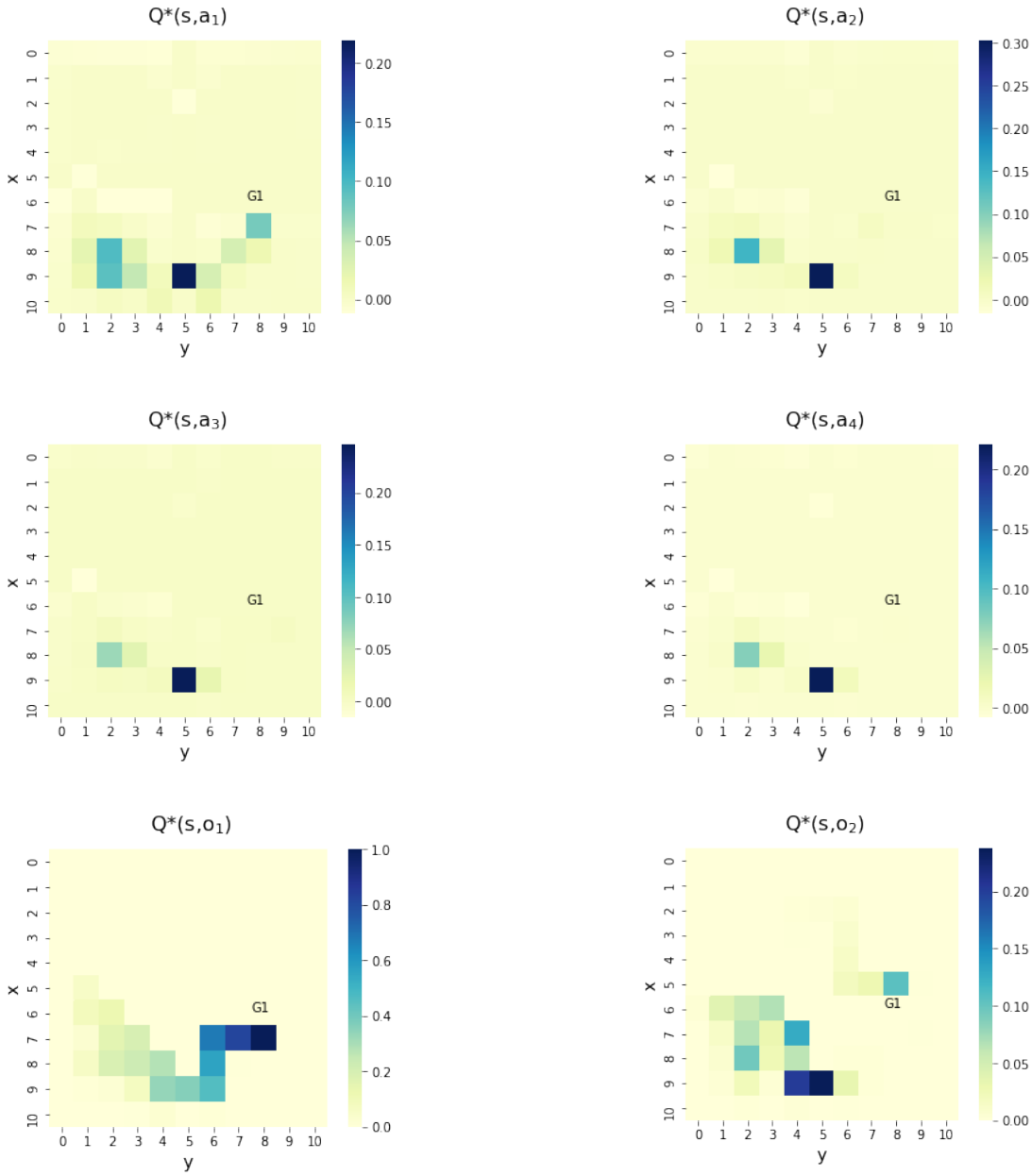


Fig 10: Performance of intra-option QLearning to achieve goal G1 from R4



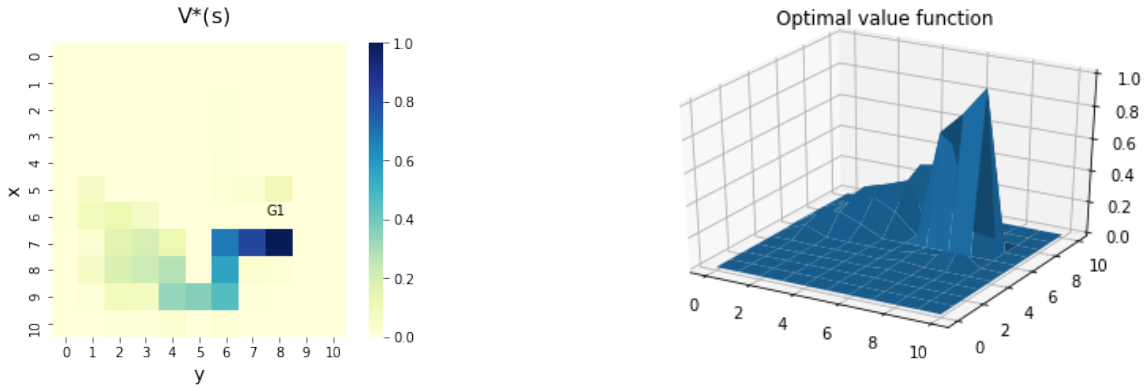


Fig 11: Intra-option optimal value functions (R4-G1)

Discussion

- The value function varies smoothly as compared to traditional SMDP QLearning. This is because the Q function is continuously updated all along without waiting for the option to complete execution.
- For the case of goal at G1 where options are useful, intra-option learning is better in terms of accuracy and convergence. This is due to the aforementioned reason that the Q values are dynamically updated without even needing to complete the execution of an option. Therefore, the intra-option method is able to assign the correct Q values to a larger section of the state space as compared to SMPDQLearning for the same duration of learning. However, there is still scope for improvement. If the initial state is randomized as opposed to being fixed like in the current example, Q values for far more number of states can be learnt leading to even more accurate Q values and thus improved learning. (Refer run.ipynb)

PROBLEM 2

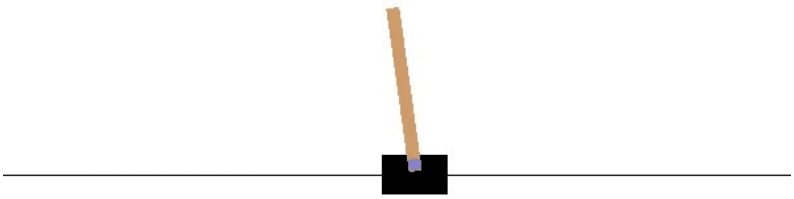


Table 1
Hyperparameter setting for the Cartpole *DQN*

Parameter	Description	Value
REPLAY_MEMORY_SIZE	Size of replay memory buffer	10^6
EPSILON	Exploratory parameter	0.5
EPSILON_DECAY	Exponential decay multiplier	0.999
EPS_STOP	Stopping criterion for epsilon-decay	0.05
HIDDEN1_SIZE	Size of first hidden layer	85
HIDDEN2_SIZE	Size of second hidden layer	85
EPISODES_NUM	Number of episodes to train on	2000
MAX_STEPS	Maximum number of steps in an episode	500
LEARNING_RATE	Learning rate for *AdamOptimizer*	10^{-4}
MINIBATCH_SIZE	Size of mini-batch sampled from the experience replay	50
DISCOUNT_FACTOR	MDP's gamma	0.99
TARGET_UPDATE_FREQ	Number of steps after which to update the target network	1000
LAMBDA	Regularization parameter	0.001

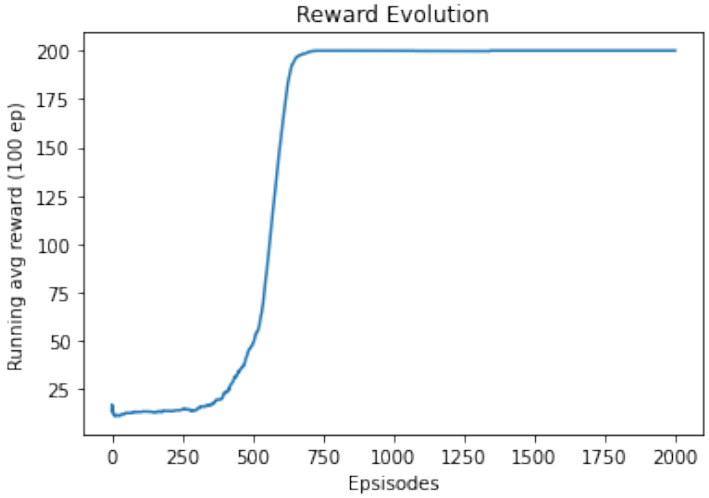


Fig 13: Performance of *DQN* to solve the CartPole task

Discussion

- The CartPole task is considered to be solved if the agent maintains an average reward of atleast 195 over a window of 100 episodes. The agent achieves this at the end of about 700 episodes. After achieving this, it is also important to maintain it over a significant period of time. It can be inferred from the above figure that the DQN has converged to the appropriate reward value and is stable.

Hyper-parameter Tuning

a) Hidden layer size: The agent fails to solve the task if the hidden layer size is too small. A deeper network solves the problem, but might require more episodes for convergence. The network size should be chosen in such a way that the problem is solved accurately enough with less trajectory sampling.

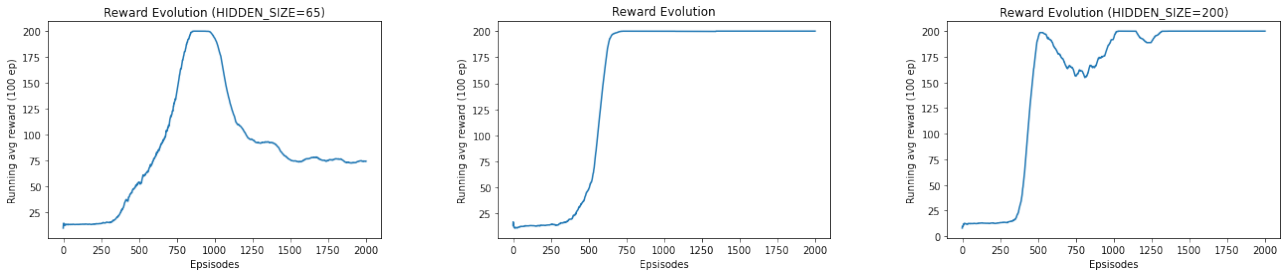


Fig 14: Effect of hidden layer size (a) 65 (b) 85 (c) 200 on training

b) Epsilon: Epsilon is chosen to balance explore-exploit as in traditional RL problems.

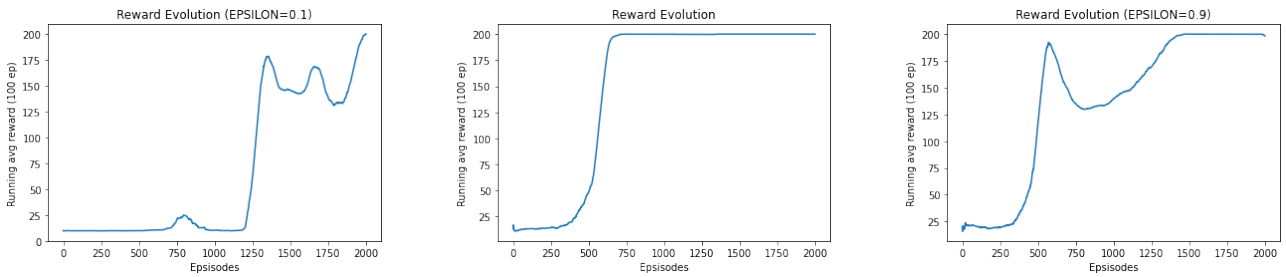


Fig 15: Effect of epsilon (a) 0.1 (b) 0.5 (c) 0.9 on training

c) Mini-Batch size: Small mini-batch size reflects increased variance in the reward values and therefore requires more number of episodes for the algorithm to achieve convergence. But at the same time, the downside of having too high of a mini-batch size is that it doesn't generalize well.

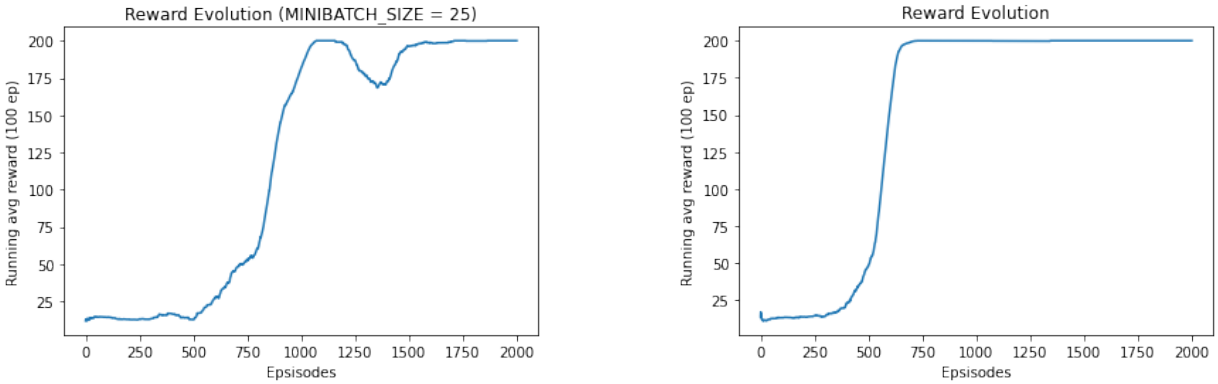


Fig 16: Effect of mini-batch size (a) 25 (b) 50 on training

d) Learning rate: When the learning rate is high, non-monotonic behavior is observed. The average reward grows more quickly, but after some point oscillations are introduced which makes learning unstable. In effect, a high learning rate makes the agent achieve the average reward of 195 sooner but induces instability.

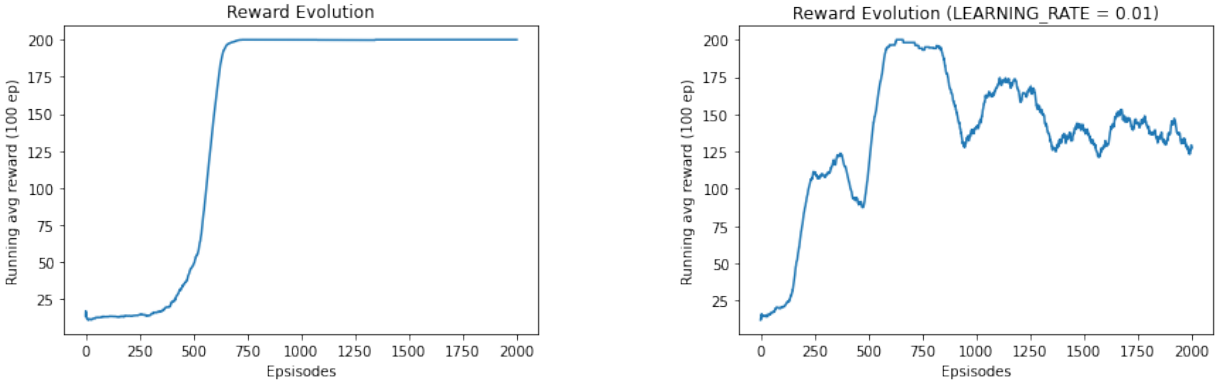


Fig 17: Effect of learning-rate (a) 10^{-4} (b) 10^{-2} on training

Experience replay and target network

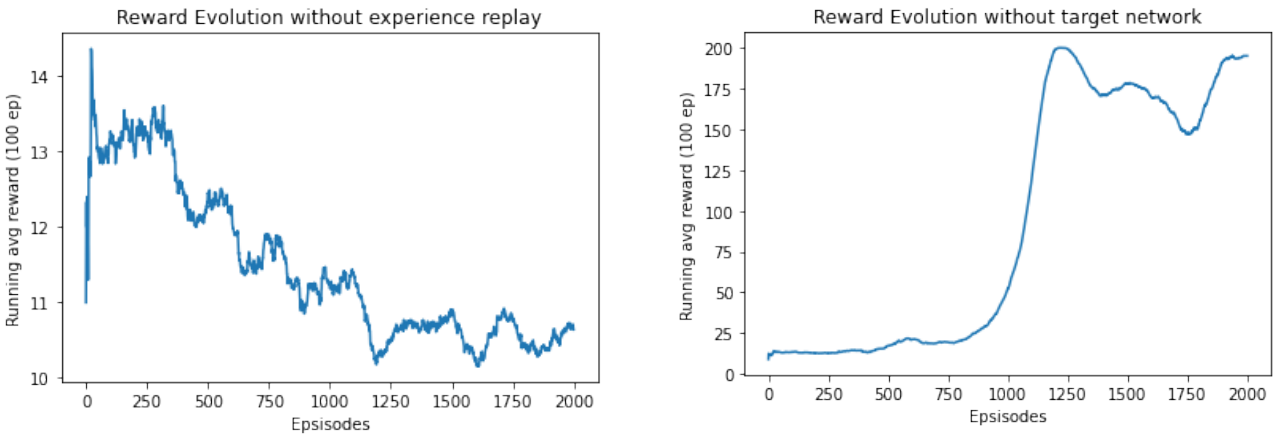


Fig 18: Learning curves without experience replay and target network

Discussion

- Removing experience replay has severe effects on the learning process. Every time the agent tries to learn using the current experience (small upward spikes), it fails. Since the training examples are generated sequentially, it doesn't have enough data of different experiences to learn from effectively.
- On the other hand, removing target network seems to not affect the learning so much. One observation is that the learning process is faster as removing target network reduces computations. Another observation is that the reward varies slowly and therefore more number of episodes are required to solve the task. Also, oscillations at the end reflect lack of stability which the target network would have brought in otherwise.
