Programming Assignment 3

R KAUSHIK — EE15B105

February 29, 2018

1 SMDP Q learning

SMDP Q-learning was implemented with value of $\epsilon = 0.1$ The optimal policy for options were hard-coded. More implementation details are present at the end of the section. The option space in the experiments consisted of both primitive actions and hallway options.

1.1 When initial state is in room 1

The start state for each episode of the experiment is set to the top left corner in room 1.

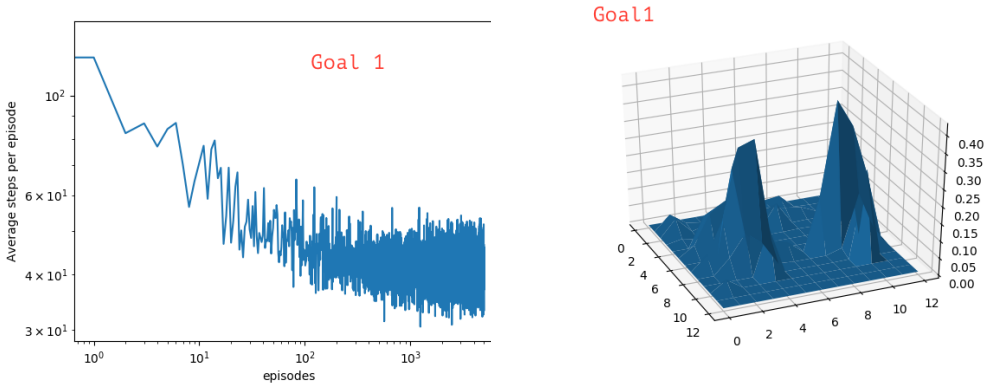

Figure 1: The plots here correspond to the environment where the agent starts from the top left corner and the goal being the hallway between rooms 2 and 3. Plot on the left shows the number of steps in an episode averaged over 10 independent experiments. Plot on the right shows the maximum of Q-values of each state learned by the SMDP algorithm

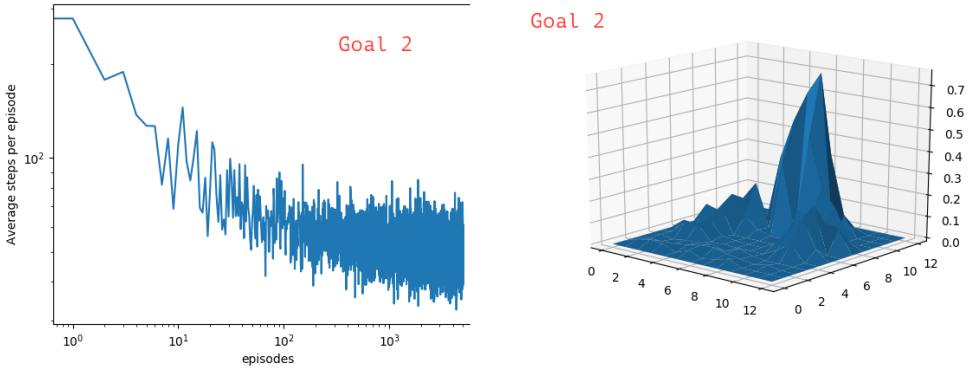

Figure 2: The plots here correspond to the environment where the agent starts from the top left corner of room 1 and the goal being in the center of room 3. Plot on the left shows the number of steps in an episode averaged over 10 independent experiments. Plot on the right shows the maximum Q-values of each state learned by the SMDP algorithm

1.2 When initial state is in room 4

1.2.1 Results

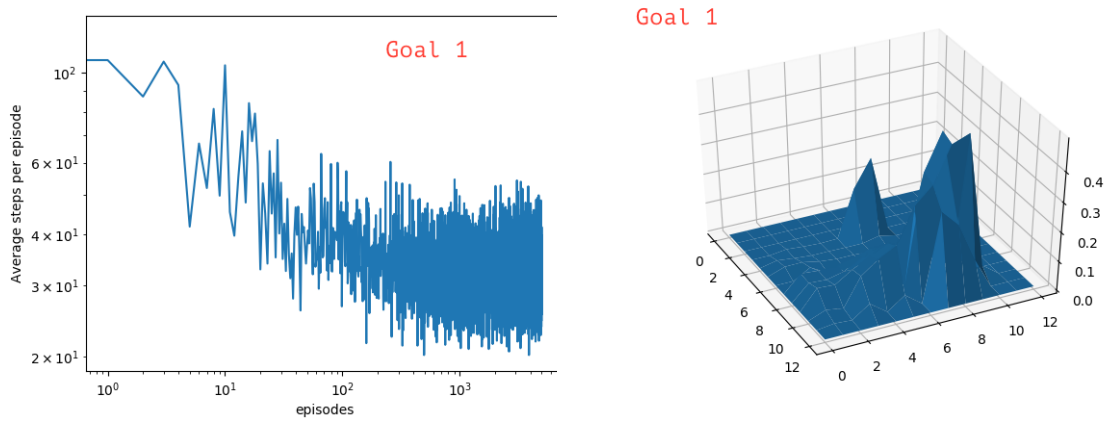

Figure 3: The plots here correspond to the environment where the agent starts from the center of room 4 and the goal being the hallway between rooms 2 and 3. Plot on the left shows the number of steps in an episode averaged over 10 independent experiments. Plot on the right shows the maximum of Q-values of each state learned by the SMDP algorithm

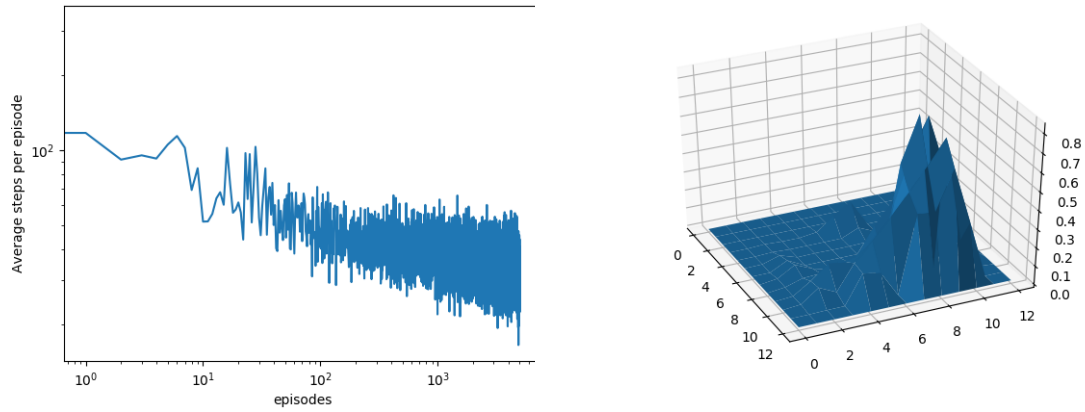

Figure 4: The plots here correspond to the environment where the agent starts from the center of room 4 and the goal being in the center of room 3. Plot on the left shows the number of steps in an episode averaged over 10 independent experiments. Plot on the right shows the maximum Q-values of each state learned by the SMDP algorithm

Observations

- We can see from the visualized Q-values that they are not learned properly for all the states. It is because from some states, hallway options were found to be the best among all options. Since the SMDP Q-learning updates only the Q-values of the starting state of an option, the intermediary states are ignored
- We can see a lot of jumps in the Q-value visualization plot form *Goal 1*. But there are not much jumps for *Goal 2*. This means the optimal options for most number of states were primitive actions for *Goal 2*. This shows that options are quite useless when when the terminal state is at such a location as that of *Goal 2*.

1.3 Intra-option Q-learning

1.3.1 Results

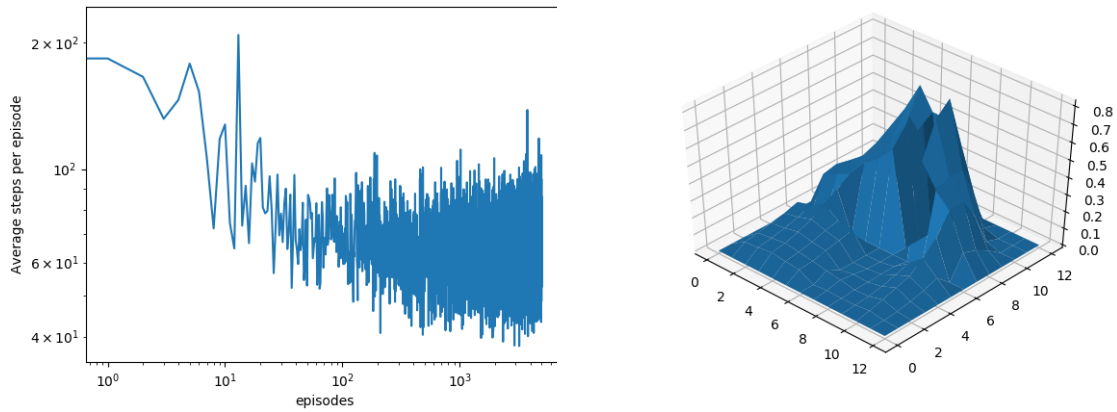

Figure 5: The plots here correspond to the environment where the agent starts from the top left corner of room 4 and the goal being in the hallway between room 2 and 3. Plot on the left shows the number of steps in an episode averaged over 10 independent experiments. Plot on the right shows the maximum Q-values of each state learned by the intra-option Q-learning algorithm

1.3.2 Observation

We can see a more smooth variation in Q values. We get more accurate Q values of more states with same number of iterations. This method will be more helpful when the starting state is not fixed as it predicts the Q-values of more number of states. If episodes were trained with random starting states, it could help in faster convergence of Q-values of states

1.4 Implementation details

The following points can be used for reference when reading the code.

- The option policies were hard coded such that every step in an option will try to minimize the Manhattan distance between the current state and the target.
- Each state can have only 2 of the 4 available options available to them. Hence the size of action space of each state can only be 6 (including 4 primitive actions).
- The actions in the `Q[state]` array is ordered as follows. The first 4 indices correspond to 4 primitive actions. The action in the 5^{th} and 6^{th} indices correspond to hallway options. The 5^{th} index of Q value of a state in room number n corresponds to the hallway that lies in the between room n and $n-1$, and 6^{th} index corresponds to the one between n and $n+1$ in the implementation.
- The difference between the intra-option Q-learning and normal SMDP Q-learning is in the function `execute option`. For intra-option Q-learning, the Q values are updated for during every step of an option execution unlike normal SMDP Q-learning.

2 Deep RL

2.1 Results

In all the plots below the value for episode i is the average taken over past $min(100, i)$ steps. The parameters for best performance is not a particular value but a range of values. The performance kept varying after every try even with the same set of hyper parameters. So in this report, the performance reported is that of **best performance for each hyperparameter set tried**.

1. Replay memory size = 10000
2. Exploration Parameter

- $\epsilon = 0.5$
- ϵ **decay = 0.99**
- **Minimum value to which ϵ can be decayed = 0.05**. The need for introducing this parameter is because it is found that after around 500 steps, the value of exploration parameter becomes very less that the agent literally stops exploring even though the agent had not learned the policy by then. This parameter ensures that if the agent lands in any sub-optimal policy, it could always come out by exploring and learn the optimal policy eventually. This parameter didn't affect the convergence much as the problem seemed to be solved before 500 steps when the value of ϵ is not too small.
- **Number of steps for which value of ϵ should be maintained constant without decaying them = 20 (Just to encourage more exploration)**. This parameter was introduced to encourage more exploration. When this parameter was made equal to 1, it took a longer time for the agent to solve the problem. The following plots show the difference between having this parameter active and inactive

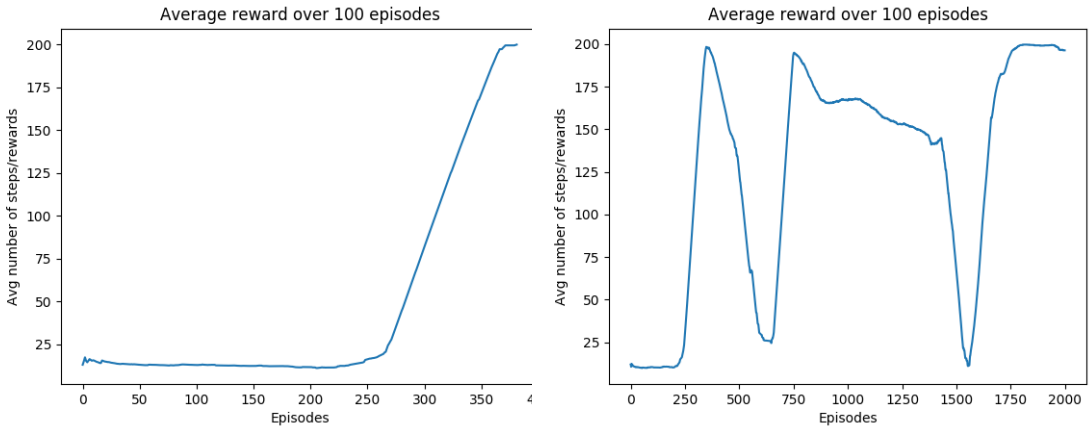

Figure 6: All the plots are plotted untill the problem gets solved. Plot in the left corresponds to the value of the constant step parameter = 20, implying that the problem got solved before 400 episodes. While plot in the right corresponds to the value of constant step parameter = 1, where the problem did not get solved even after 2000 episodes

3. **Neural Network:** The performance of the agen kept varying for every try of the same set of hyperparameters. The hidden layer of size 100 once solved the problem but couldn't solve the other time. Hence the report here presents the results considering the best performance of each set of hyperparameters.
   - **Hidden layer 1 = 100 (or 80)**
   - **Hidden layer 2 = 100 (or 80)**
     The size of hidden layers played a crucial role in solving the problem. In the all trials, the size of 2 hidden layers were kept equal. When the size of hidden layer was ¿= 120, the problem didn't get solved at all (this was a crude observation. The problem sometimes didn't get solved for values ¡ 120 too). The plots below show the performance of the agent for different hidden layer sizes

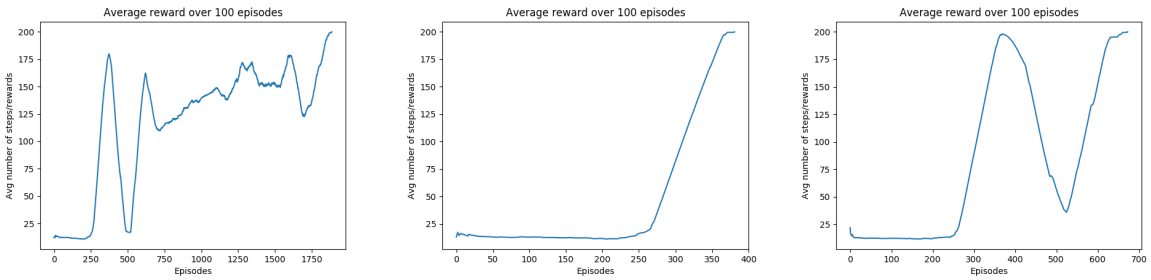

Figure 7: All the plots are plotted uptill the problem gets solved. The leftmost plot corresponds hidden layer size 50, where that the problem got solved after 1900 episodes. While plot in the middle corresponds to the size of hidden layers = 80, where the problem got solved before 400 episodes! The rightmost plot corresponds to hidden layer size = 100, where the problem got solved before 700 episodes

- Learning rate = 0.00025

4. Mini batch size = 30
5. Target Network update frequency = 300 steps