RL PA3

Ajay S Joshi (CS15B047)

May 5, 2018

1 Options

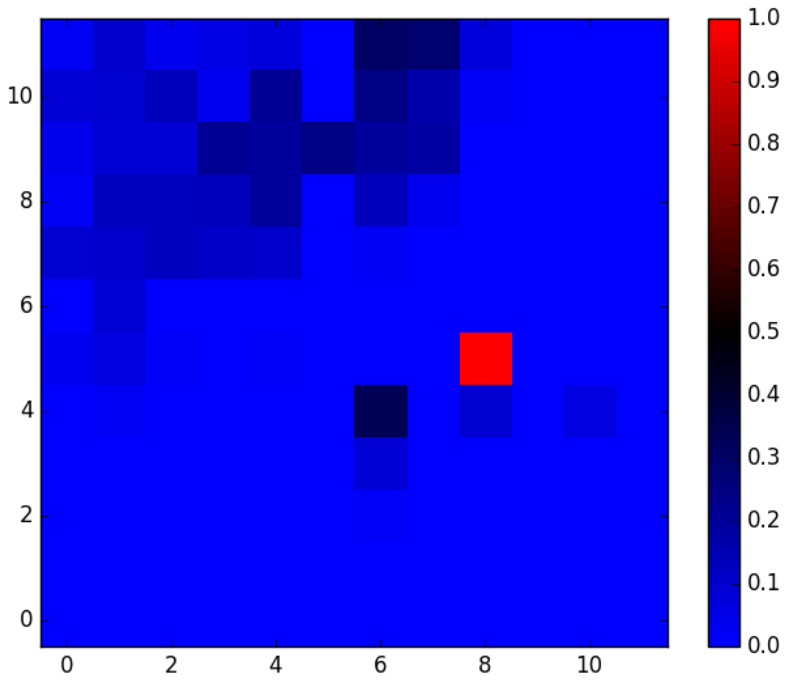
In the environment, all rewards are assumed to be 0 except for transitions involving goal states.
All Value function plots for this question plot the quantity $\max_o Q(s,o)$.

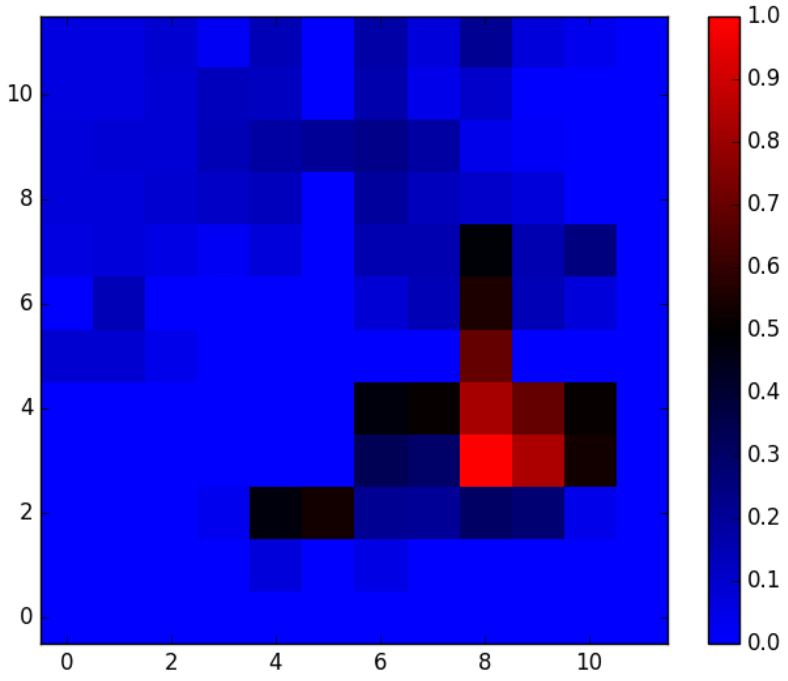
1.1 Visualization

(Plots shown for G1 and G2 respectively for both methods.)

- SMDP Q Learning:

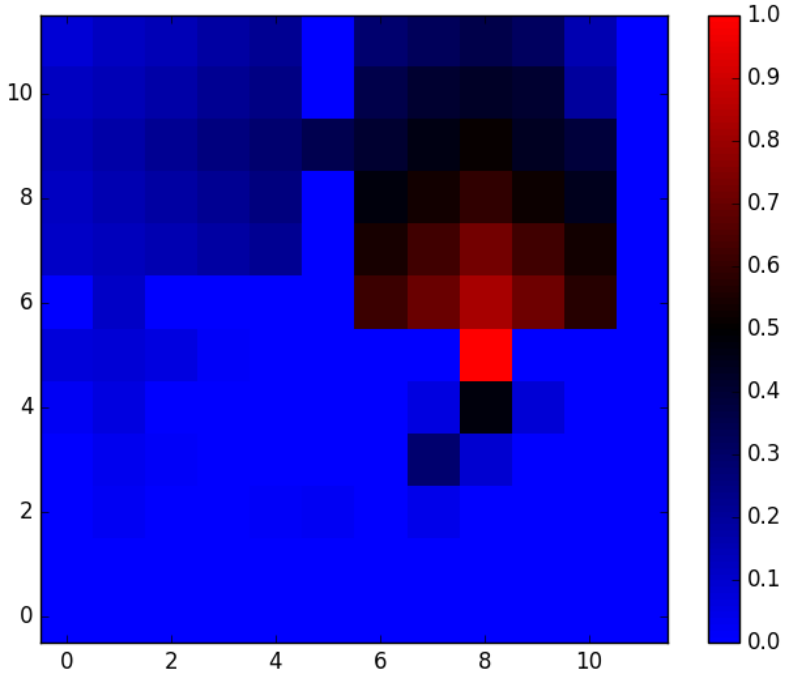
  1. As multi-step options are taken by agent, only values corresponding to initial states are updated, and thus only starting room values, hallways and some states near them are considerably updated.

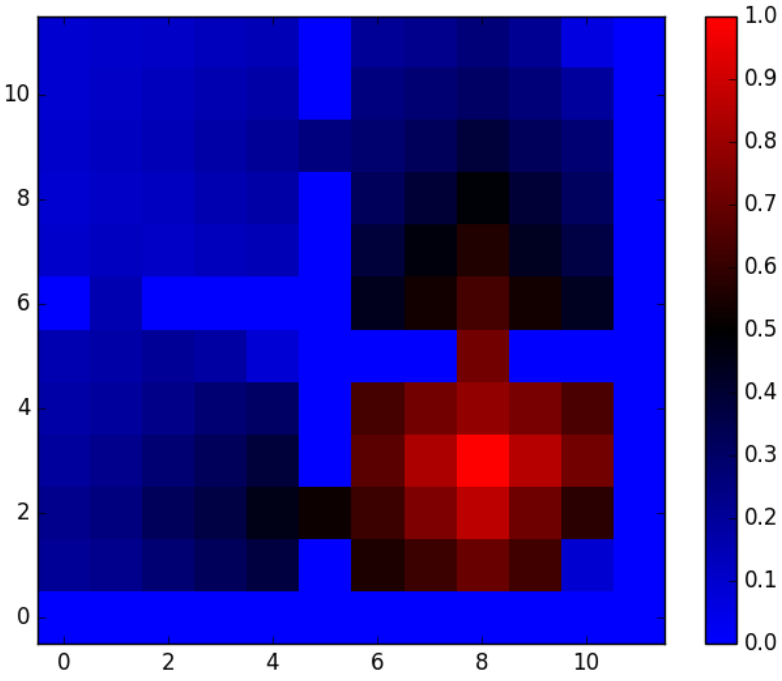




- Intra-Option Q-Learning:

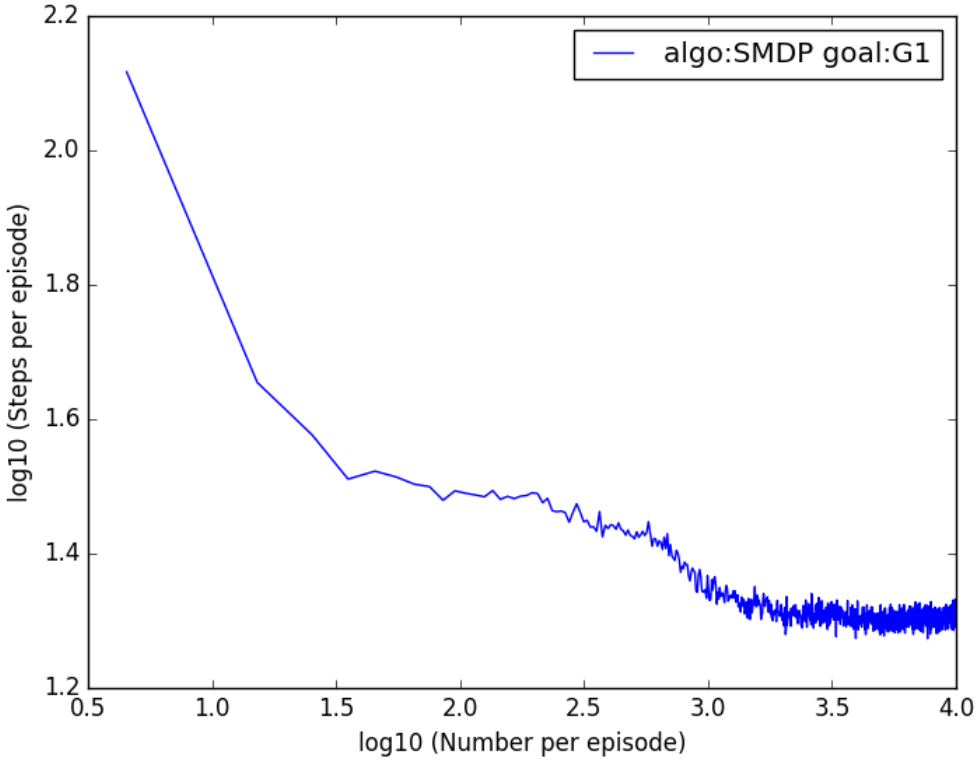
  1. All state values are updated properly, as agent explores the state space, and in between option execution also state values are updated.

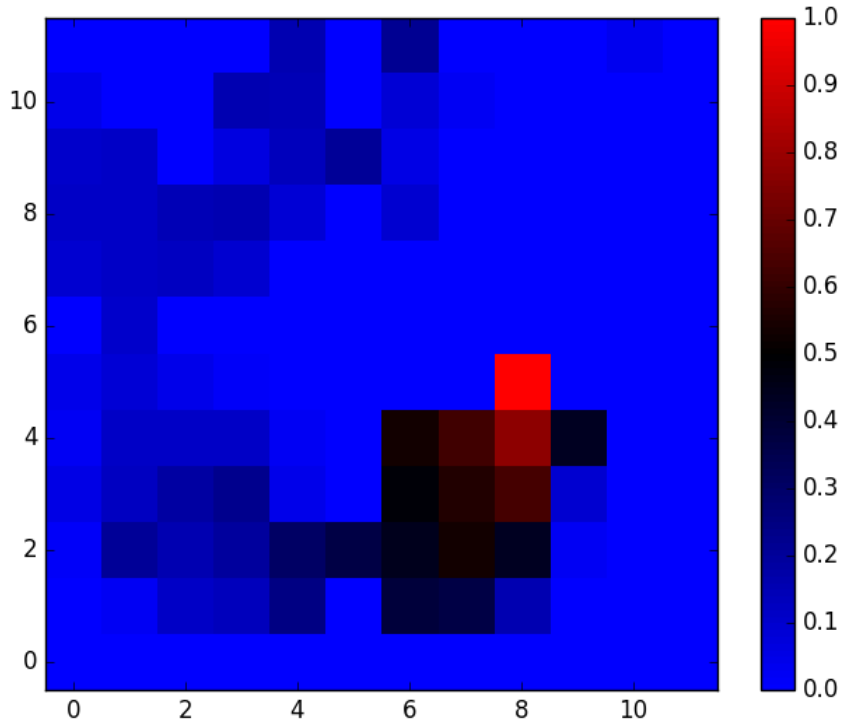


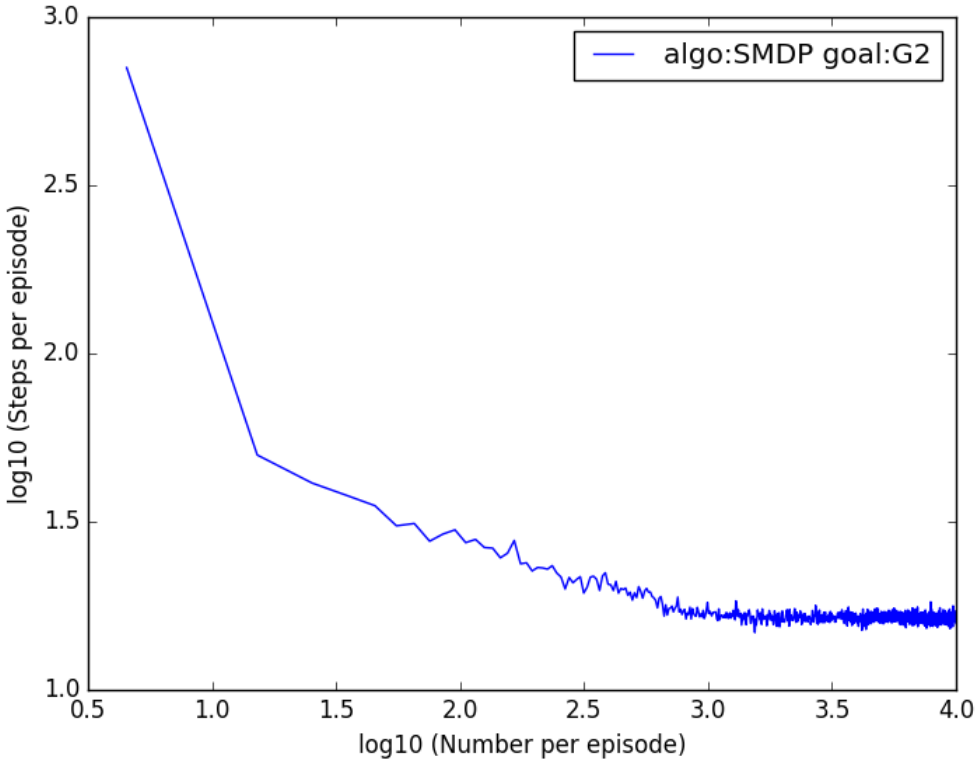


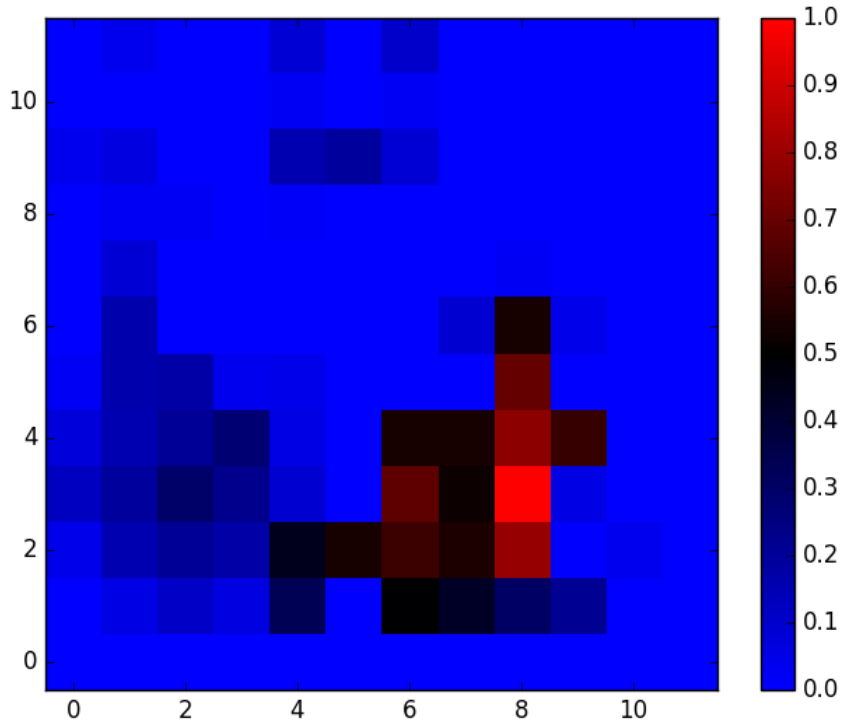
1.2 Changing start state

1. Below plots show Value functions and steps per episode v/s number of episodes for start state fixed to center of bottom left room.
2. Path through lower hallway is taken by agent, as can be observed from lesser updated values of top right room.



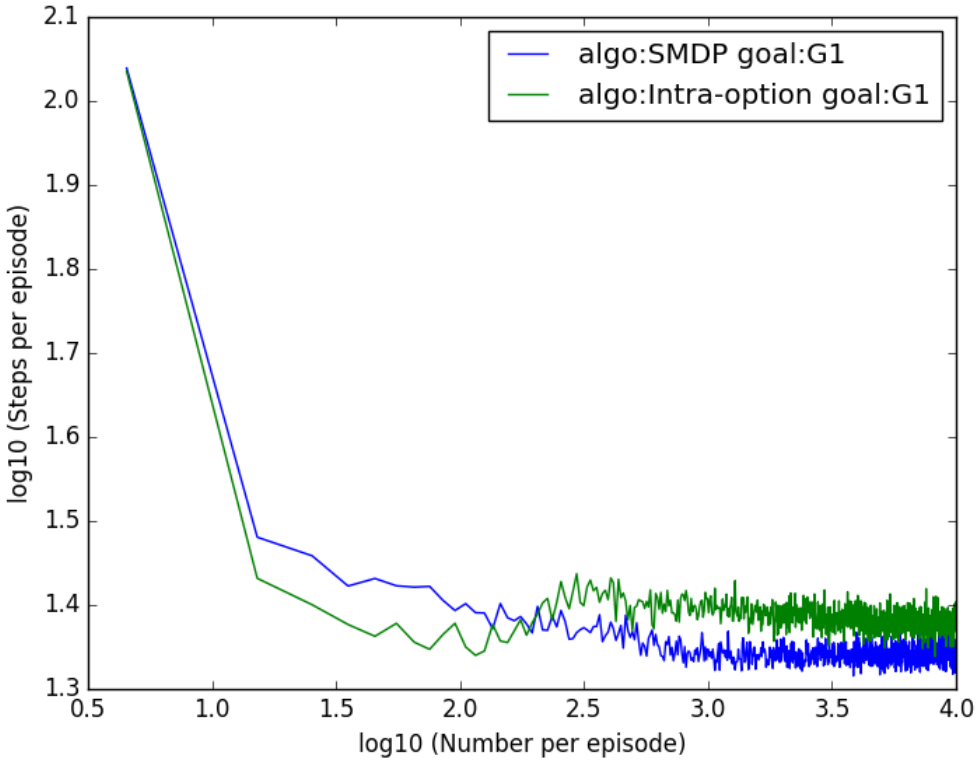

0
2
4
6
8
10
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

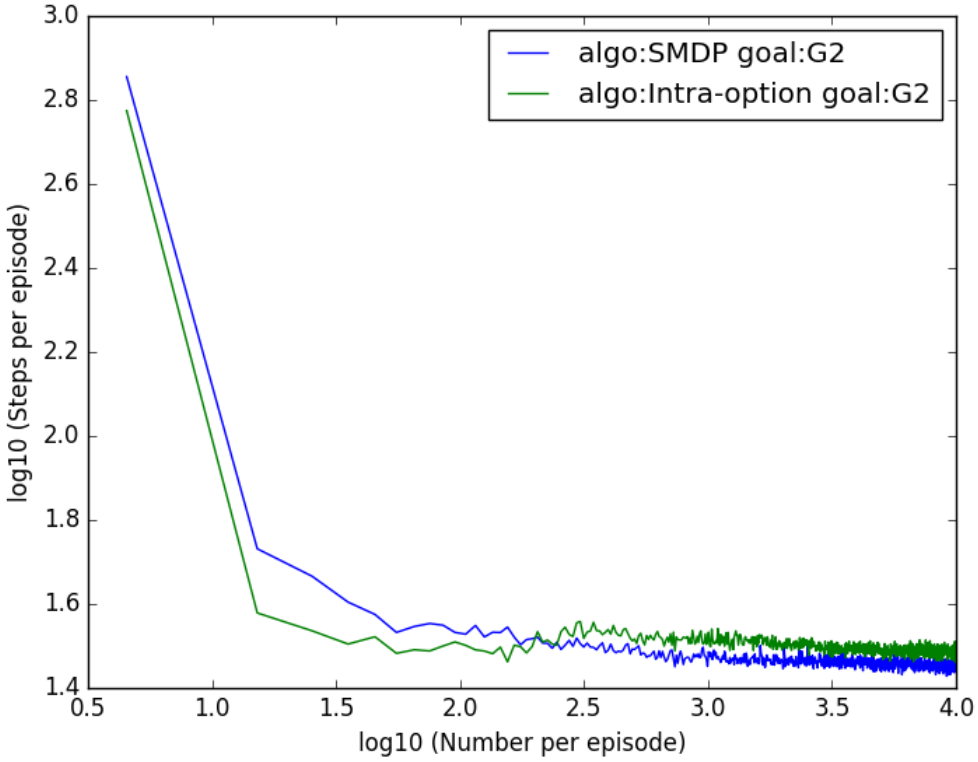

algo:SMDP goal:G2
log10 (Steps per episode)
log10 (Number per episode)
3.0
2.5
2.0
1.5
1.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0



1.3 Intra-option Q Learning

1. Using intra-option gives some amount of speedup at the starting of learning due to higher frequency of updates leading to faster learning initially.
2. Also, similar performance is obtained at convergence for both methods.



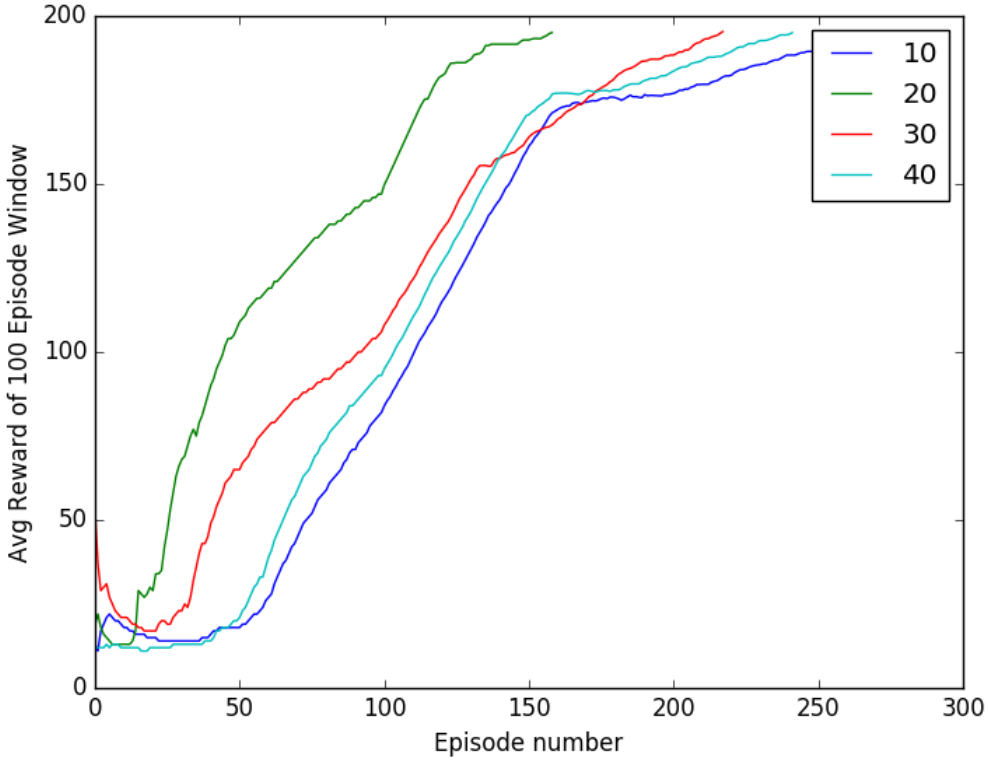

algo:SMDP goal:G2
algo:Intra-option goal:G2
log10 (Steps per episode)
log10 (Number per episode)
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

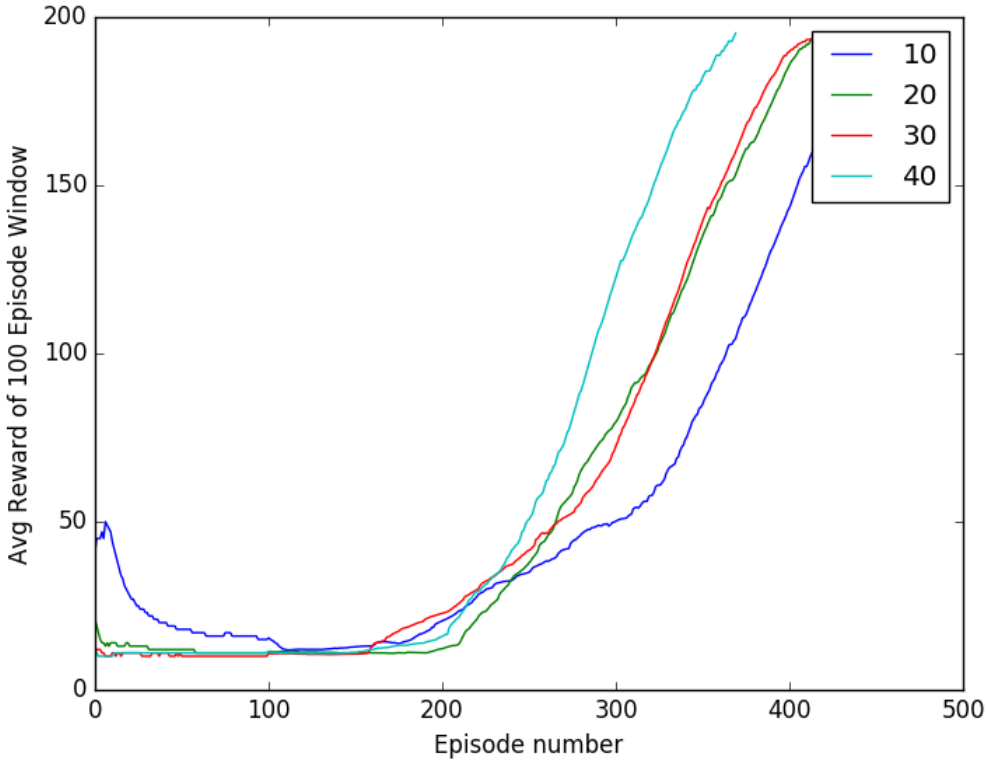
2 DQN

2.1 Variation of Parameters

2.1.1 Batch size

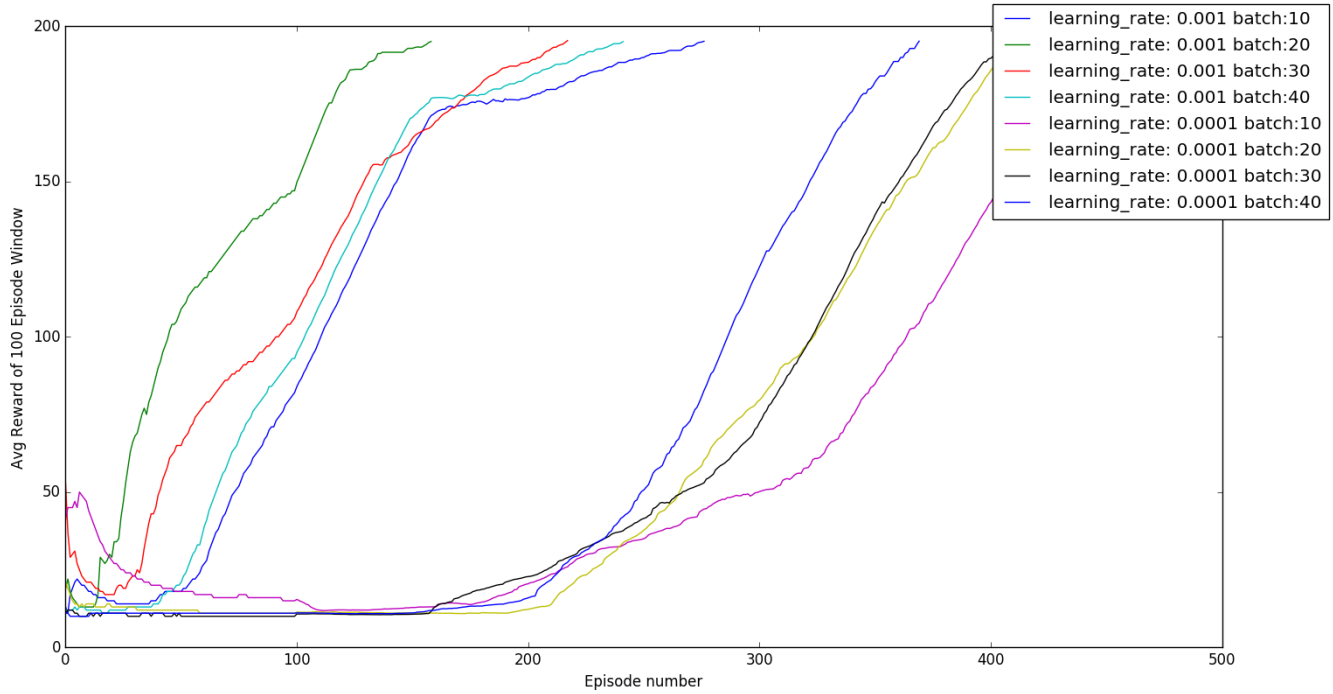
1. Below graphs are for different batch sizes for learning rates of 0.001 and 0.0001 respectively.
2. Some amount of improvement is obtained by increasing batch size from 10.
3. Also, sometimes performance seems to worsen on increasing batch size a lot.
4. This maybe explained by bias-variance, i.e. for very lower batch sizes, updates made are subject to higher variance, and for higher batch sizes updates maybe biased. So, medium batch size works better here.





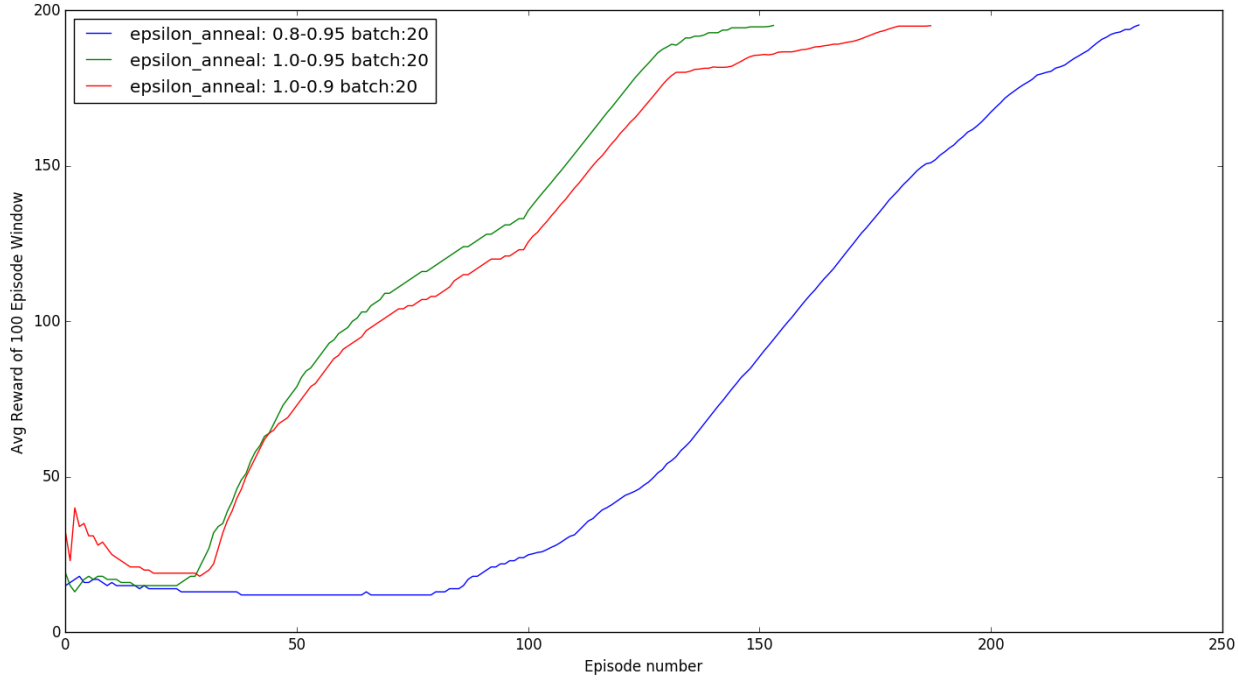
2.1.2 Learning Rate

1. Clearly, increasing learning rate increases performance of agent.
2. However, increasing it further from 0.001 to 0.01 leads to almost no convergence at all (so graphs couldn't be included), indicating that a learning rate around 0.001 is good for this problem.



2.1.3 Epsilon Annealing

1. Higher initial exploration leads to better performance, as lower exploration may lead to picking of suboptimal actions that prohibit/delay convergence.
2. Also, slower decay works well for this problem, for the same reason above.



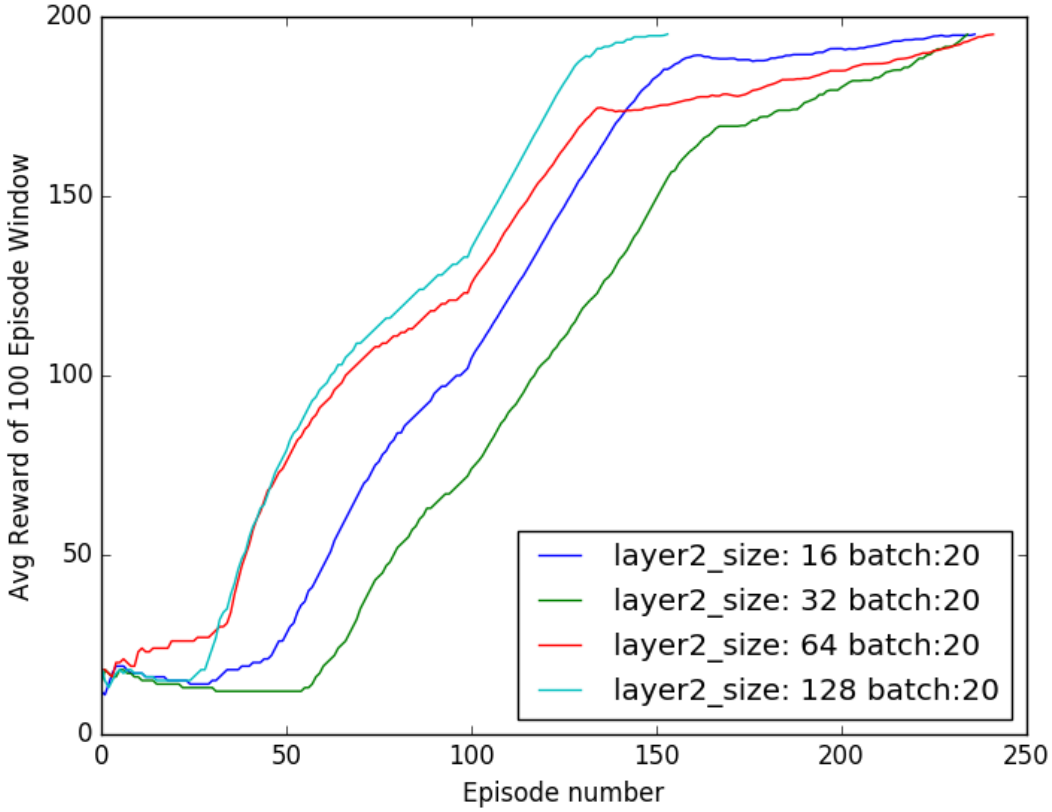
2.1.4 Hidden Layers

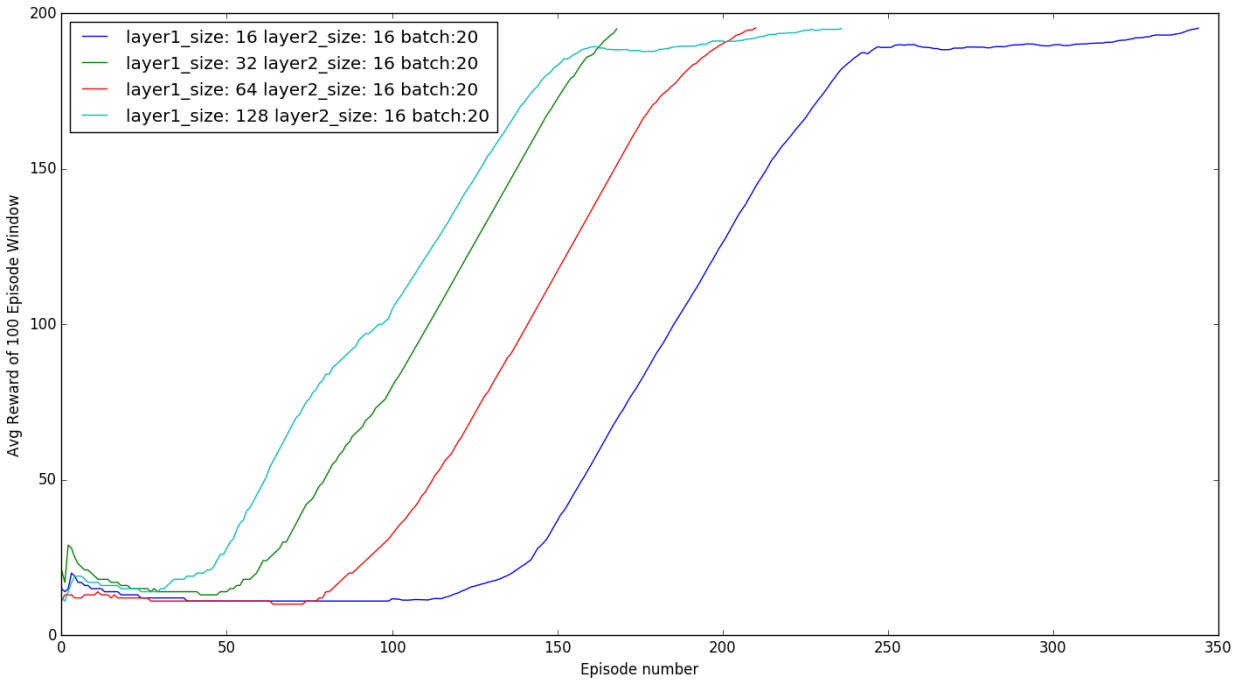
1. Variation in units in both layers leads to similar changes.

2. Generally, more units in hidden layer gives a better performance, which can be seen from graph for 128 units(in layer 2), and not much difference for other lower sizes.

3. Also, a significant difference observed due to number of units is the quality of policy learnt by agent. For eg: For 128 units case, agent manages to balance pole without sliding in any direction, while with lower number of units, agent learns to slide slowly to one side, which is easier but not a better solution than former, as this cannot be extended, and is just a getaway for this problem.

4. Also, with larger units, solution is more stable and more repeatable than solutions for lower batch sizes.

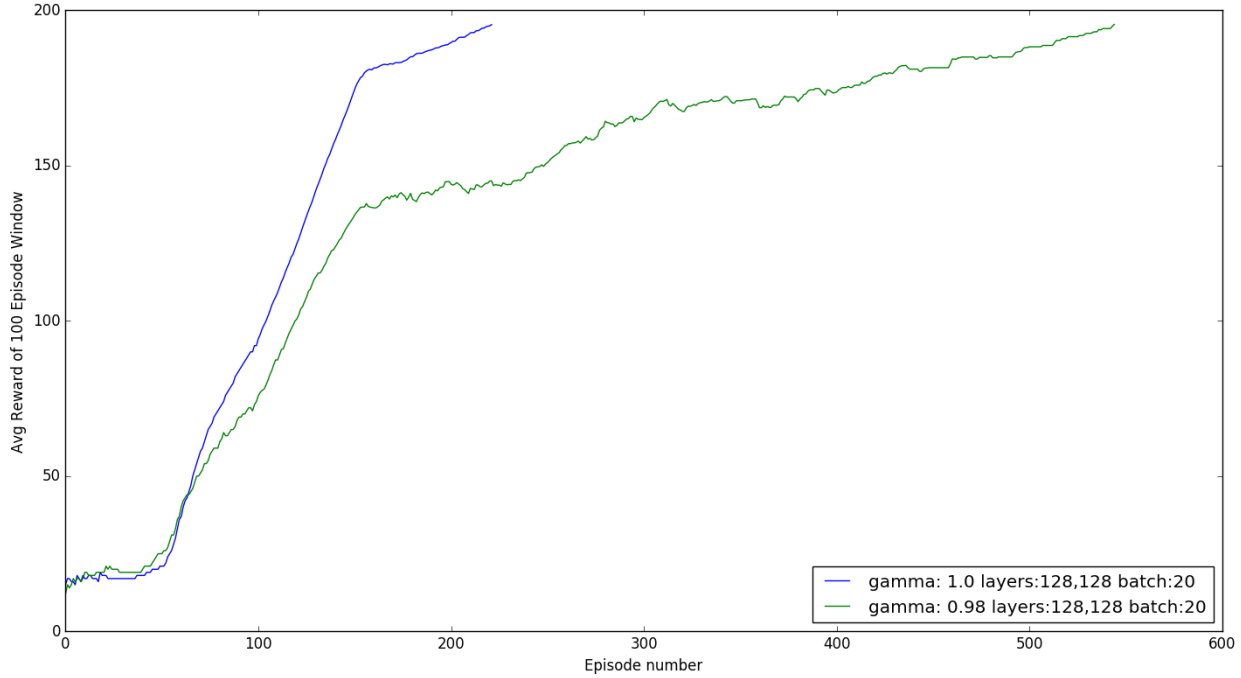




2.1.5 Discount Factor

1. Variation in γ makes a huge impact on the solution, due to ability to distinguish between rewards, when number of steps are high.

2. When γ is lower, agent gets very weak returns for higher number of steps per episode, and thus learning suffers.
3. So, best performance is obtained for $\gamma = 1$, and then decreases with decreasing γ.



2.2 Hyper-parameters

From the above variation in parameters, the following choice of hyper-parameters can be made :

1. $\gamma = 1.0$ (not a hyper-parameter, but can be varied to solve the same objective)
2. Batch size of approximately 20-30
3. ϵ - annealing : $\epsilon = 1.0$ at start , and decay with 0.95 or a slower rate.
4. Learning rate of 0.001 or higher, but lower than 0.01, as instability observed at 0.01 learning rate.
5. Hidden layer size depends on whether quality of solution matters or not.
    - If solution in which agent has to move slowly towards one side, as it cannot balance the pole at one place, is acceptable, then 128 and 16 neurons in layer 1 and 2 is a good candidate solution.
    - If unstable solutions are acceptable, then size of network can further be reduced to about (64 or 32) and 16 neurons in layer 1 and 2 works.
    - But, the best stable solution, with agent balancing pole at a place requires 128 , (64-128) neurons in layer 1 and 2.