# CS6700 : Reinforcement Learning
# Written Assignment #4

Sai Vinay G
CE17B019
Indian Institute of Technology, Madras

May 5, 2019

## Problem 1

As we are getting a reward of -1 for each step we take, initially when we perform a primitive action near start states we don't move to a goal state (assuming we don't start in the room which contains goal) and get a reward of -1. In case if we perform an option, which leads to a doorway state (which will not be a goal, as the goal is in the corner of one of the rooms) which gives us a large negative reward (the number of steps taken to reach the doorway). So, initially all the primitive actions have better action values, as compared to the actions values of options available. So, the agent tends to pick more primitive actions than multistep options. So, due to this we don't get the expected faster learning behaviour even in the presence of options.

---

## Problem 2

---

## Problem 3

In A3C we by running asynchronously more workers we are actually collecting more training data, which make the collection of data faster. As every single work has its own instance of environment, so we can get very different data to improve the process of learning to get better results. and experience replay has several drawbacks like: it uses more memory and computation per real interaction and it requires off-policy learning algorithms that can update from data generated by an older policy.

In A3C each agent interacts with global parameters independently, so sometimes some agents will be trying updating the weights of previous version, but by then the weights might have already changed by other agents therefore update may not be optimal. The Advantage of DQN over A3C is that A3C has variance due to the sampling of gradient for updating.

The problem of asynchronous updates is tackled in A2C, where a coordinator waits for all the parallel actors to finish their work before updating the global parameters and then in the next iteration parallel actors starts from the same policy which leads to better convergence

## Problem 4

Bottlenecks are the states that the agent visits frequently on its successful paths to goals but not on unsuccessful paths. If the agent discovers these bottlenecks and learn policies to navigate to them, during initial stages, we can quickly explore the world to learn fast. This sub-goals (bottlenecks) can be used to learn similar sub problems.

If we take for example the problem of room navigation, our goal will be in one of the rooms, leaving our starting room. In this case in order to move to another rooms we need to pass through the doors (bottleneck states). If we don't learn policies to navigate to bottlenecks, it will take the agent long amount of time to change rooms as it will be exploring the present room most of the time. But, if we had the agent had learned the policies for navigating to the bottleneck states(doorways) as a sub-goal then it can quickly move to other room and explore in all rooms to find the goal states and learn a better overall policy.

## Problem 5

By having the model of the environment, we have a function which predicts state transitions and rewards. Using the model we can do planning to learn a optimal policy. They are very sample efficient as compared to methods which don't use sampling i.e, are samples produced are produced are more important in the process of learning than the once which are normally produced which increases speed of learning.

Also, as the agent needn't wait for the environment to respond to the action and no need to wait for the environment to reset, to continue the learning. In some case, like training a car to navigate, is much difficult to perform in real world as its very expensive, time consuming and dangerous, But if we had a model similar to the real world conditions we can quickly train our agent on that model to overcome the above difficulties.

## Problem 6

In Q-MDP we solve the MDP component of the POMDP and use the Q(s,a) and beliefs to compute the Q(b(s),a) value function over belief states and actions. Here we are solving the MDP assuming that we have access to know all the states without uncertainty and using a heuristic to solve the POMDP.

But there can exist an optimal policy if we directly solve the POMDP, considering the uncertainty in the state that we are present in, there may be a better action to take than the action we get using Q-MDP.

So if we consider a set of states which result in the same belief(all states in the set seems to be same to the agent whenever encountered) and we perform an action which gets us different results (maybe some gives use positive rewards and others negative rewards) but as the beliefs and actions are same so we are updating the same Q-value based on outcomes from different states.

This does not give us an optimal behaviour as the actual states which are near the goal are getting the updates of some other states which maybe near negative rewarding states.

For the Q-MDP behaving optimally, we can consider the case in which we get similar beliefs for the states which give us similar rewards then updating in all states would not cause a problem (which is like coming to the same state and updating based on the reward obtained).

---

## Problem 7

---

## References

[1] J. Achiam. Rl algorithms. [Online]. Available: https://spinningup.openai.com/en/latest/spinningup/rl$_{i}ntro2.html$

[2] jonathan$_{h}ui.Rldqndeepq - network.[Online].Available : https : //medium.com/@jonathan_{h}ui/rl - dqn - deep - q - network - e207751f7ae4$

[3] lilianweng. Policy gradient methods. [Online]. Available: https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.htmlpolicy-gradient

[4] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, “Learning policies for partially observable environments: Scaling up,” in *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 362–370.