CS6700 : Reinforcement Learning
Written Assignment #3

Deadline: ??

- This is an individual assignment. Collaborations and discussions are strictly prohibited.
- Be precise with your explanations. Unnecessary verbosity will be penalized.
- Check the Moodle discussion forums regularly for updates regarding the assignment.
- **Please start early.**

AUTHOR : Name.
ROLL NUMBER :

1. (3 marks) Consider the problem of solving POMDPs using Deep Reinforcement Learning. Can you think of ways to modify the standard DQN architecture to ensure it can remember histories of states. Does the experience replay also need to be modified? Explain.

Solution: Ref: http://cs229.stanford.edu/proj2015/363_report.pdf
In deep Q learning, a neural network is used to approximate the POMDP Q-values. For POMDP, the Q values are parameterized by either the belief and the action Q(b,a) or an action-observation history h and Q(h,a). The modified Q-values can be learned by a neural network that is characterized by weights and biases combined denoted as θ. Q values will be $Q(b, a|\theta)$. The standard update of Q values may lead to divergence. Such issues is resolved by experience replay tuples (b,a,r,b) which are recorded in a reply memory. The aim of replay is to stabilize the learning by drawing samples uniformly.

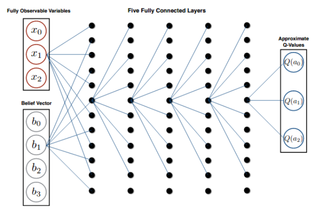


Figure 2: Five layer fully connected network that maps the concatenated fully observable variable and belief vectors to Q-values

Apart from this, a separate target network is used update state targets to the main network. An adaptive learning method can be used to regulate and adjustment the parameter rate of the network.

Changes in architecture:
The changed architecture of DQN is given in figure-1. In architecture, simulator is used to populate and experience reply dataset. The input layer consists the belief of the agent and fully observable state variables which generalize the representation of problems. The agent may have some knowledge of the system state known as mixed observability MDPs(MDOMPs). **A similar architecture was used for training the DQN on action observation histories which requires the manipulation of experience reply for efficient use of reply.** The fully observable state variables are used as inputs to the network. The current formulation uses a fully connected network that either takes the fully observable state variables and the belief or just the belief and outputs a value approximation.

2. (4 marks) Exploration is often ameliorated by the use of counts over the various states. For example, one could maintain a visitation count $N(s)$, for every state and use the same to generate an intrinsic reward $(r_i(s))$ for visiting that state.

$$r_i(s) = \tau \times \frac{1}{N(s)}$$

However, it is intractable to maintain these counts in high-dimensional spaces, since the count values will be zero for a large fraction of the states. Can you suggest a solution(s) to handle this scenario? How can you maintain an approximation of the counts for a large number of states and possibly generalize to unseen states?

Solution: Ref: `http://surl.tirl.info/proceedings/SURL-2017_paper_5.pdf`
The solution to this problem is count-based exploration algorithm. The proposed algorithm uses linear function approximation(LFA). The key idea behind the algorithm is to exploit the feature map that is uses of value function approximation, and construct a density model over the transformed feature space. The model assigns higher probability to state feature vectors that share features with visited states.

The algorithm is called ϕ-Exploration Bonus algorithm. A generalized count is a novelty measure that quantifies how dissimilar a state is from those already visited. $\phi_t = \phi(s_t)$ denote the state feature vector observed at time t. We denote the sequence of observed feature vectors after t timesteps by $\phi_{1:t} \in T_t$, and denote the set of all finite sequences of feature vectors by T^*. $\phi_{1:t}\phi$ denote the sequence where $\phi_{1:t}$ is followed by ϕ. $\rho_t(\phi)$ be the feature visit density after observation of $\phi_{1:t}$. $\rho'_t(\phi)$ density after observation of $\phi_{1:t}\phi$. The pseudocount N_t^{ϕ} is defined as,

$$N_t^{\phi} = \frac{\rho_t(\phi(s))(1 - \rho'_t(\phi(s)))}{\rho'_t(\phi(s)) - \rho_t(\phi(s))} \quad (1)$$

By setting the hyper parameter as β. The sample reward is defined as,

$$R_t(s,a) = \frac{\beta}{\sqrt{N_t^{\phi}(s)}} \quad (2)$$

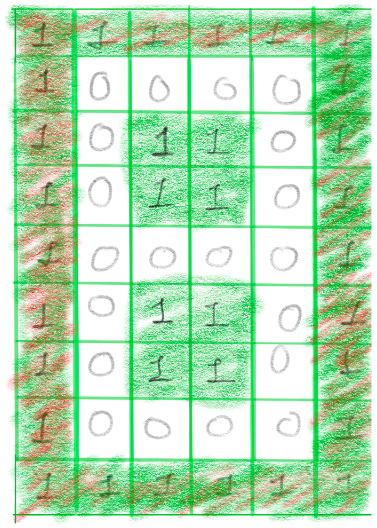
The bonus is added to reward r_t. The trained agent augmented reward r_t^+ can be obtained using any value based RL algo.

$$r_t^+ = r_t + R_t(s,a) \tag{3}$$

Due to functional transformation the cost of method is independent from size of the action space.

3. (5 marks) Suppose that the MDP defined on the observation space is k-th order Markov, i.e. remembering the last k observations is enough to predict the future. Consider using a belief state based approach for solving this problem. For any starting state and initial belief, the belief distribution will localize to the right state after k updates, i.e., the true state the agent is in will have a probability of 1 and the other states will have a probability of 0. Is this statement true or false? Explain your answer.

Solution: If the information obtained by all states is correct then for k^{th} order Markov chain last k observation is enough to predict the future. The example explained in the lecture have possible.



The Grid world is given in fig-3. Each cell with 0 indicates clear or approachable state and 1 indicates(green colour) are obstacles. Green and Red colour cell indicates wall. The location of state is give based on its neighbor cell. The state numbering starts from north and goes by following clockwise direction. States: 0001, 1010, 0101, 1100, 0110, 0011, 0100, 1001 have its count respectively 1, 6, 8, 1, 1, 1, 1, 1.

Let's take example of to reach 1001. There are many ways to reach at that spot. But if we know certain history then it attainable to reach desired state with Pr=1. Initial state have some initial belief p(s). As more observation is added in terms of history, the belief about the state space keeps getting more focused. After update $p(s) \to p'(s) \to p''(s)$ and $p(s) > p'(s) > p''(s)$ probability of getting correct belief goes higher. For k-th order Markov chain the belief gets updated k times in which there is high probability that all state in Markov chain gets correct belief state which in turn may give true state with probability 1.

4. (3 marks) Q-MDPs are a technique for solving the problem of behaving in POMDPs. The behavior produced by this approximation would not be optimal. In what sense is it not optimal? Are there circumstances under which it can be optimal?

Solution: In Q-MDP we have solved the problem assuming that we access to state at that point. But to solve the problem later, by heuristic approach to get the execution policy. The obtained execution policy obtained by heuristic way might not be optimal as heuristic might have uncertainty in states and there might be better action to pick in terms of total reward obtained. so, **execution policy obtained by heuristic might have uncertainty and this lead to sub-optimal approximation.**

If we solve the POMDP directly then there are chance to get optimal scenario given states are known. Known states and desired action gives next states and which makes policy deterministic. **Using the simplex belief states and relation between value function gives efficient way to solve POMDP directly to learn policy from history such deterministic scenario makes approximation optial.**

5. (3 marks) What are some advantages and disadvantages of A3C over DQN? What are some potential issues that can be caused by asynchronous updates in A3C?

Solution: In DQN, a single agent interacts with a single environment. whereas, A3C asynchronously launches many workers based on efficiency of CPU. All of them interact with their own environment. Every worker instance also has their own environment which gives more diverse data. This approach makes network robust and gives good results.
Advantages of A3C:

— By asynchronously launching more workers gives more training data, which makes the collection of the data faster.

— As every worker have different environment, the data obtained will be more diverse.

— The variation in data makes process more efficient and gives better result.

Disadvantages of A3C:

— More computational effort

— Some similar workers may generated repeated samples which may cause bias in the learning.

Potential issues that can be caused by asynchronous updates in A3C:

— some workers (copies of the Agent) will be playing with older version of the parameters because of the asynchronous nature and the aggregate update might not be optimal.

6. (6 marks) There are a variety of very efficient heuristics available for solving deterministic travelling salesman problems. We would like to take advantage of such heuristics in solving certain classes of large scale navigational problems in stochastic domains. These problems involve navigating from one well demarcated region to another. For e.g., consider the problem of delivering mail to the office rooms in a multi storey building.

   (a) (4 marks) Outline a method to achieve this, using concepts from hierarchical RL.

Solution: Traveling salesman problem or post delivery kind of problems have repetition of certain action. Such large navigation problem in stochastic domain may follow certain in structure in state space. Large navigation problem can be divided into smaller, repetitive, reusable, sub problems which follows repetition of certain actions.

Solving sub-problems can take varying amount of time and space which is expressed as **Temporal abstraction.**

Smaller sub problems can be optimized and for new problem, some part of optimized problem can be reused. Such transfer of problem provides **reuseability.**

Stpes: Divide the the lager problem into smaller, repetitive sub problems. Each smaller problems can be optimized and each smaller object can be called step by step. One of the approach to solve such kind of problem is **Hierarchical Abstract Machine(HAM).**

In this each machine contains four steps:

Action state : At this state it performs the action. Inputs will be state of the problem. The control will be transfer to call state.

Call state : This state machine will check different possible condition and call most relevant machine.

Choice state : Based on choice this machine will perform the relevant choice of action.

Stop state : If the goal is reached this machine will stop all machine and give final output in terms of primitive states.

−− >To explain this in terms of traveling sales man problem, Based on initial state, salesman will perform action means he/she can go to can take cab or travel to new nearest city. Out of all available choice, the most relevant choice will be called means salesman will look at the available choice and optimize the state action space and perform the required action like doing deals or sales. After performing all the action salesman, he can take the most optimized route for next deal. If goal is reached then the stop machine will check the end condition and can give final output. All the different smaller action can be performed parallely and combined result can be obtained.

(b) (2 marks) What problems would such an approach encounter?

Solution:
– Hierarchical reinforcement requires proper connection of smaller problem.
– Wrong hierarchy may generate the unexpected result.
– Connection between action and optimization of state space can increase the computational cost.
– The hierarchy may go indefinitely deep, so about hierarchy is crucial.

7. (6 marks) This question may require you to refer to https://link.springer.com/content/pdf/10.1007/BF paper on average reward RL. Consider the 3 state MDP shown in Figure 1. Mention the recurrent class for each such policies. In the average reward setting, what are the corresponding ρ^{π} for each such policy ? Furthermore, which of these policies are gain optimal ?

(a) (3 marks) What are the different deterministic uni-chain policies present ?

Solution: An MDP is termed unichain if the transition matrix corresponding to every policy contains a single recurrent class, and a (possibly empty) set of transient states.

Class of MDP: The problem contains two recurrent unichain MDPs.

Class-1 policy : {State-A, State-B} From state-A if action a1 is performed then it goes to state-B and again come back to state-A so with period of 2. The formed policy is {A;a1(2,1), B;a2(0,1)}

Class-2 policy: {State-A;action-a2, State-C} From state A, action a2 is performed then control will go to state-c it will come again to state-A. so given states also form policy by performing action policy :{A;a2(0,1) , C;a2(1,1)}.

Class-3 policy: {State-A; action-a3, State-C} If action a3 is performed then from state-c will come to state-A. so given states also form policy by performing action:{A;a3(-1,1), C;a2(1,1)}.

(b) (3 marks) In the average reward setting, what are the corresponding ρ^π for each such policy ? Furthermore, which of these policies are gain optimal ?

Solution: The average reward is defined as,

$$\rho^\pi = \lim_{N\to\infty} \frac{E(\sum_{t=0}^{N-1} R_t^\pi(x))}{N} \tag{4}$$

For,
class-1 average reward : 2
class-2 average reward : 1
class-3 average reward : 0

Out of these policy, the policy which gives maximum expected reward is defined as gain optimal ρ^*. class-1 policy, by performing action:{a1(0,1),a1(2,1)} gives maximum reward-2 , is optimal gain policy.

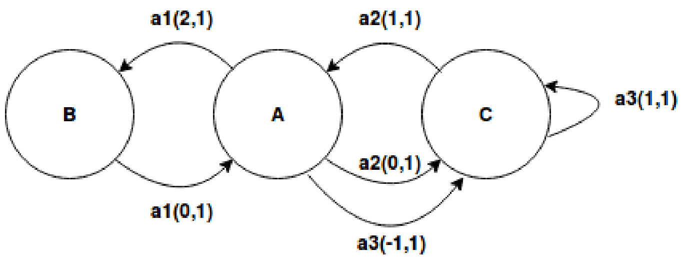


Figure 1: Notation : action(reward, transition probability). Example : a1(3, 1) refers to action a1 which results in a transition with reward +3 and probability 1