CS6700 : Reinforcement Learning
Written Assignment #2

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 : Dodiya Keval.
ROLL NUMBER : CS19M023

1. (3 points) Consider a bandit problem in which the policy parameters are mean μ and variance σ of normal distribution according to which actions are selected. Policy is defined as $\pi(a;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma}}e^{-\frac{(a-\mu)^2}{2\sigma^2}}$. Derive the parameter update conditions according to the REINFORCE procedure (assume baseline is zero).

Solution: We have given...

$\pi(a;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma}}e^{-\frac{(a-\mu)^2}{2\sigma^2}}$

$\ln \pi(a;\mu,\sigma^2) = \ln \dfrac{1}{\sqrt{2\pi\sigma}} - \dfrac{(a-\mu)^2}{2\sigma^2}$

By taking derivative with respect to parameters $\mu(avg)\ and\ \sigma$.

$$\frac{\partial \ln \pi(a;\mu,\sigma^2)}{\partial \mu} = \frac{(a-\mu)}{\sigma^2} \tag{1}$$

and

$$\frac{\partial \ln \pi(a;\mu,\sigma^2)}{\partial \sigma} = \frac{(a-\mu)^2}{\sigma^3} + \frac{(-1)}{\sigma} \tag{2}$$

so updation will be like

$$\mu_{t+1} = \mu_t + \alpha * r_t * \frac{(a_t - \mu_t)}{\sigma_t^2} \tag{3}$$

$$\sigma_{t+1} = \sigma_t + \alpha * \frac{r_t}{\sigma_t}\left(\frac{(a_t-\mu_t)^2}{\sigma_t^2} - 1\right) \tag{4}$$

2. (6 points) Let us consider the effect of approximation on policy search and value function based methods. Suppose that a policy gradient method uses a class of policies that do not contain the optimal policy; and a value function based method uses a function approximator that can represent the values of the policies of this class, but not that of the optimal policy.

   (a) (2 points) Why would you consider the policy gradient approach to be better than the value function based approach?

   Solution:

   - When action-space is very high dimensional or we can say when action-space is intractable to handle. then we can go for Policy Gradient Approach because it is not dealing with each action and state. so it cuts out computation compared to value function based method.
   - when action-space is large and continuous then we need to approximate action values that might not be optimal. further its representation is also an issue. and second thing mostly value-function based method gives policy are deterministic and mostly policies are stochastic in nature. since PG approach able to find stochastic policy its more appropriate to use in some cases.
   - sometimes both the methods are not able to find optimal policy. in that case PG apporach is computationally efficient.

   (b) (2 points) Under what circumstances would the value function based approach be better than the policy gradient approach?

   Solution:

   - Value function based methods are much faster in convergence compared to PG method. PG method might converged to local optima unlike Value Based methods those are converge to global optima.
   - so we can say if optimal policy is deterministic near-optimal policy then Value based method appropriate.

   (c) (2 points) Is there some circumstance under which either of the method can find the optimal policy?

Solution:

- if we have finite state-action space and function approximation is accurate then both will work well. and one more if optimal policy is deterministic and near-optimal then both will work fine e.x. gridworld with specific goal and removing stochasticity both will give optimal policies .

3. (4 points) Answer the following questions with respect to the DQN algorithm:

- (2 points) When using one-step TD backup, the TD target is $R_{t+1} + \gamma V(S_{t+1}, \theta)$ and the update to the neural network parameter is as follows:

$$\Delta\theta = \alpha(R_{t+1} + \gamma V(S_{t+1}, \theta) - V(S_t, \theta))\nabla_\theta V(S_t, \theta) \tag{5}$$

Is the update correct ? Is any term missing ? Justify your answer

Solution: The equation is incorrect. It needs to use a seperate target network. and this equation does not use experience replay for updation that can be missing term.

- (2 points) Describe the two ways discussed in class to update the parameters of target network. Which one is better and why?

Solution:

- **Hard Update :** we can say this is periodic update with period of some amounts of steps. so target network will got updated with some 'k' amount of steps.
- **Soft Update :** This update take place at every steps with some conditions. every steps training network moves towards target network and this is slow update . so updation can be take place as below.
  $\theta_{target} = \alpha * \theta_{target} + (1 - \alpha) * \theta_{train}$,here α is some probability.

Well both are best in its own way. hard update is best because it could be more stable baseline for updating the training network result in more stable learning. and if we tuned *alpha* perfectly it also can give stable learning but requires tuning otherwise it'll be very unstable learning!!

4. (4 points) Experience replay is vital for stable training of DQN.

(a) (2 points) What is the role of the experience replay in DQN?

Solution:

- one advantage of replay buffer is to remove correlation in experiences while training DQN. it is done by randomly sample experience result in more stable learning.
- another advantage is to reusing old experience if one experience appeared more frequently it has to be trained more frequently result in high updated value of corresponding action lead to more optimal policy.

(b) (2 points) Consequent works in literature sample transitions from the experience replay, in proportion to the TD-error. Hence, instead of sampling transitions using a uniform-random strategy, higher TD-error transitions are sampled at a higher frequency. Why would such a modification help?

Solution: if we have High TD-error that imply high difference in real value to predicted value. so sampling experienced with high Td-error gives DQN more training with those experiences. so it adapt more learning compared to uniform random sampling at equal chances of getting sampled for learning so in that case less significant sample has equal probability that has to be less.

5. (3 points) We discussed two different motivations for actor-critic algorithms: the original motivation was as an extension of reinforcement comparison, and the modern motivation is as a variance reduction mechanism for policy gradient algorithms. Why is the original version of actor-critic not a policy gradient method?

Solution: Original motivation : In this update rule is like ..

$$P(s_t, a_t) = P(s_t, a_t) + \beta(\delta_t) \tag{6}$$

here β is step size parameter and $\delta_t = \tau_{t+1} + \gamma V(s_{t+1}) - V(s_t)$

Modern motivation : In this update rule is like ..

$$P(s_t, a_t) = P(s_t, a_t) + \beta\delta_t(1 - \pi_t(a_t|s_t)) \tag{7}$$

here actions are generated from policy π which follows softmax function. hence $\pi_t(a|s) = \dfrac{e^{P(a|s)}}{\sum_b e^{P(b|s)}}$

so we can see here Original version does not follow policy by equation 6,7 or equation 6 does not depend on policy π. hence that motivation is not policy gradient

6. (4 points) This question requires you to do some additional reading. Dietterich specifies certain conditions for safe-state abstraction for the MaxQ framework. I had mentioned

in class that even if we do not use the MaxQ value function decomposition, the hierarchy provided is still useful. So, which of the safe-state abstraction conditions are still necessary when we do not use value function decomposition?

Solution:

- when we don't use value function decomposition then LeafNode Irrelevance and MaxNode Irrelevance conditions are have to be satisfied for safe state abstraction. because these conditions help to identify not relevant state variables in a task or sub-task and make an abstraction functions that use to map original state to reduced-relevant state. other conditions like Result Distribution Irrelevance, Shielding ,Termination are not required because they deal with completion cost values.

7. (3 points) Consider the problem of solving continuous control tasks using Deep Reinforcement Learning.

   (a) (2 points) Why can simple discretization of the action space not be used to solve the problem? In which exact step of the DQN training algorithm is there a problem and why?

   Solution:

   - one problem is high dimensional action space since we don't know how to discretize, what is the size of action space?, which is optimal for our world?so such a large action spaces are difficult to explore efficiently hence replay buffer does not have appropriate experiences, and thus that gives problem in successful training. so DQN-like networks in this context is likely intractable.
   - furthermore naive discretization of action spaces needlessly throws away information about the structure of the action domain, which may be essential for solving many problems.

   (b) (1 point) How is exploration ensured in the DDPG algorithm?

   Solution: By adding Noise to the actions while training, we can ensured exploration in DDPG algorithm.
   Noise can be Gaussian with some μ *and* σ , Ornstein-Uhlenbeck noise

8. (3 points) Option discovery has entailed using heuristics, the most popular of which is to identify bottlenecks. Justify why bottlenecks are useful sub-goals. Describe scenarios in which a such a heuristic could fail.

Solution:

- Bottleneck states are the states which are more likely to be appeared in path to the goal by agent. since bottlenecks are very useful it needs to be accessed before goal state so option can be created with bottleneck state is the goal state. and that option will be macro action for agent.
- Bottlenecks can also be defined as junction e.x. doorways in Gridworld . so it is transferable and reusable so it cuts down exploration. but for that we need information regarding environment
- heuristic would fail if the goal cannot be reached with only primitive actions. and if we have high dimensional action space then it required more exploration that also cumbersome .