CS6700 : Reinforcement Learning
Written Assignment #2

Deadline: 30-May-2020

- 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 : Rana Pragneshkumar Rajubhai
Roll Number : ME17S301

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: The REINFORCEMENT parameter update equation can be written as,

$$\Delta\theta_n = \alpha(R_n - b_n)\frac{\partial \ln \pi(a_n;\theta)}{\partial \theta_n} \tag{1}$$

The parameters for Gaussian policy is given as μ and σ and policy is given as,

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

Update of μ:

$$\frac{\partial \ln \pi(a_n;\mu_n,\sigma_n)}{\partial \mu_n} = \frac{\partial}{\partial \mu_n}\left\{\frac{-(a_n-\mu_n)^2}{2\sigma_n^2}\right\} = \frac{a_n-\mu_n}{\sigma_n^2} \tag{3}$$

Considering baseline performance to zero(b=0),

$$\textbf{update rule for mean :} \quad \mu_{n+1} = \mu_n + \alpha R_n\left(\frac{a_n-\mu_n}{\sigma_n^2}\right) \tag{4}$$

similarly, Update of variance(σ):

$$\begin{aligned}\frac{\partial \ln \pi(a_n;\mu_n,\sigma_n)}{\partial \sigma_n} &= \frac{\partial}{\partial \sigma_n}\left\{-\ln(\sqrt{2\pi\sigma_n})\right\} + \frac{\partial}{\partial \sigma_n}\left\{-\frac{-(a_n-\mu_n)^2}{2\sigma_n^2}\right\} \\ &= -\frac{1}{\sigma_n} + \frac{-(a_n-\mu_n)^2}{2\sigma_n^2} = \frac{1}{\sigma_n}\left\{\left(\frac{a_n-\mu_n}{\sigma_n}\right)^2 - 1\right\}\end{aligned} \tag{5}$$

$$\textbf{Update rule for variance :} \ \sigma_{n+1} = \sigma_n + \alpha R_n \frac{1}{\sigma_n}\left\{\left(\frac{a_n-\mu_n}{\sigma_n}\right)^2 - 1\right\} \tag{6}$$

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 appropriator 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: Value function based methods approximation the Q function and use to find optimal policy so, in case of non-existence of optimal policy, it may get stuck infinitely in loop or it might take all the upper defined limit of iteration. Whereas, the policy gradient has better convergence criteria as it directly optimize the policy space by finding the local optimal policy. The action is picked based on the state which maximized the reward which done using parameterθ.

   $$\pi(a|a, \theta = \Pr A_t = a, S_t, \theta_t = \theta) \tag{7}$$

   Eq.- shows that, the policy is denotes the probability of taking action based on state(s) and parameter(θ).

   Due to this characteristic, policy gradient method may converge to local optima or best action are more likely to get sampled and converges. Due to this, policy gradient approach is better than value function approximation.

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

   Solution: For the case of existence of optimal policy, value function based approximation technique is useful as policy gradient method may end up with sub optimal policy. Even value function method works better for high variance case. In case of high variance policy gradient requires other variance reduction treatment.

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

   Solution: Yes, in the case of deterministic policy value function works and the policy gradient may converge the best sub-optimal policy. If optima exist, then both method can find optimal learn optimal policy and policy gradient can do it efficiently.

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:

$$\Delta\theta = \alpha(R_{t+1} + \gamma V(S_{t+1}, \theta) - V(S_t, \theta))\nabla_\theta V(S_t, \theta) \tag{8}$$

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

Solution:

- Yes, the update is correct
- No term is missing.

$$\Delta\theta = \alpha(R_{t+1} + \gamma V(S_{t+1}, \theta) - V(S_t, \theta))\nabla_\theta V(S_t, \theta) \tag{9}$$

- For the given equation,
$(R_{t+1} + \gamma V(S_{t+1}, \theta)$ denotes the target and $V(S_t, \theta)$ term is required control function.
The update rule is given as,

$$\theta_{t+1} = \theta_t + \alpha\nabla_{\theta_t}[(R_{t+1} + \gamma\hat{V}(s_t, \theta_t) - \hat{V}(s_t, \theta_t))^2] \tag{10}$$

By differentiating w.r.t θ the target term will act as constant as it can be replace by return like G_t so, completer update rule will be,

$$\theta_{t+1} = \theta_t + \alpha(R_{t+1} + \gamma V(S_{t+1}, \theta) - V(S_t, \theta))\nabla_\theta V(S_t, \theta) \tag{11}$$

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

Solution: The two ways to update the parameters mentioned in the lecture is based on the sampling method. As the sampling method varies, the associated data and its up-gradation in parameters also varies.

- Online- off policy algorithm
- Memory based transition reply

Memory based transition reply is better

Out of which transition state reply is good as it solves the problem f repeated samples. In case of online algo. the obtained samples might correlated and may create problem in convergence which is not observed in case of transition reply.

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: The neural network can get learn samples from experience which might be consecutive but it might be highly correlated. Such correlation makes learning inefficient. To break such correlation experience reply memory us useful as it stores all the sample and randomly samples will be picked up. In short, **experience replay memory is useful to break the correlation between consecutive samples.**

   (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: The uniform random strategy may cause bias in certain state-action space. Due to repetition or any other reason, certain random sample might get sampled often. Such problem can be resolved by sampling from higher TD error transition. If samples from **certain state-action space** is not picked up from long duration then it may caused more error so, sampling based on TD error in next few iteration with higher frequency can stabilize the process. Eventually, **it will cover whole region and sampling transition will be smooth rather than abrupt.**

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: The reinforce can be written as,

$$\begin{aligned} \Delta\theta &= \alpha_n(r_n - b_n)\frac{\partial \ln \pi(a, \theta_n)}{\partial \theta} \\ &= \alpha_n(G_n - b_n)\frac{\partial \ln \pi(a, \theta_n)}{\partial \theta} \end{aligned} \tag{12}$$

r_n is related to return and can be replace by G_n.

The original version of reinforce contains G_n term in the formulation which causes high variance in the update as each time different sample might have different return. If the G_n term is replaced by the value function or some form of value function then variation in that term can be reduced which connects the reinforce to policy gradient. So, the original reinforce or actor critic in 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: The required five condition are mentioned below that permits "safe" introduction of state abstraction.

Condition 1: Max Node Irrelevance

Useful when a set of state variables is irrelevant to Max Node. It should satisfy the following properties:

– > The state transition probability distribution P^{π}(s',N—s,a) at node i can be factored into the product of two distribution.

$$P^{\pi}(x', y', N|x, y, a) = P^{\pi}(y'|y, a)P^{\pi}(x', N|x, a) \tag{13}$$

– > for any pair of the states $s_1 = (x, y_1)$ and $s_2 = (x, y_2)$ such that $\xi(s1) = \xi(s2) =$ x and for any child action a, $V^{\pi}(a, s_1) = V^{\pi}(a, s_1)$ and $R_i(s_1) = R_i(s_2)$

Condition 2: Leaf Irrelevance

This condition describes the situation under which it is possible to apply the state abstraction to leaf nodes of the MAXQ graphs.

$$V(a, s) = \sum_{s'} P(s'|s, a)R(s'|s, a)$$
$$\text{and} \tag{14}$$
$$\sum_{s'_1} P(s'_1|s, a)R(s'_1|s_1, a) = \sum_{s'_2} P(s'_2|s_2, a)R(s'_2|s_2, a)$$

Condition 3: Result Distribution Irrelevance

The mentioned condition is result of funnel condition.

– > A set of state variable Y_j is irrelevant for the result distribution of action j if, for all the abstract policies π executed by node j and its desccent in the MAXQ hirarchy, the following holds. for all the pair of state s_1 and s_2 that differ only in their values for the state variables in Y_j.

$$P_{\pi}(s', N|s_1, j) = P_{\pi}(s', N|s_2, j) \quad \forall s', N \tag{15}$$

Condition 4: Termination

This condition is also related 'funnel' property. It applies when sub-task is guaranteed to cause its parent task to terminate in a goal state. In a sense,

the subtask is funneling the environment into the set of states described by the goal predicate of the parent task.

Condition 5: Shielding

The shielding condition aries from the structure of MAX Q. graph. The Shielding condition can be verified by analyzing the structure of the MAXQ graph with termination condition.

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: In DQN algorithm, neural network function appropriators were used to estimate the action-value function. DQN can solve problems with high-dimensional observation space an low dimensional, discrete action space. Many real world application have continuous and high dimensional action space. **DQN cannot be directly apply to continuous domains as it depends on the finding the action that maximize the action-value function. In each step, the action value function in continuous valued case requires as iterative optimization process.**

The exact step in algorithm is,

$$\begin{aligned} \text{target} &= R(s,a,s') + \gamma \max_{a}^{'} Q_k(s',a') \\ Q_{k+1}(s,a) &\leftarrow (1-\alpha)Q_k(s,a) + \alpha[target] \end{aligned} \tag{16}$$

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

Solution: In DDPG algorithm. while selection of action, noise $\mathcal{N}_t$ is introduced which make sure of exploration.
step:
Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to current policy and exploration 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: Bottle necks are useful as it cut down the cost of **exploration**. Apart from that, one has to pass through such states which provides the **re-usability** of known states. In case of transfer of terminal state, bottle neck has some information about previous goal so it helps in **transfer learning**.

Heuristic failure experience : Take the example of two room, many sample path are obtained. Out which some sample generate successful trajectory whereas, some generate unsuccessful trajectory. In which, the goal is to focus on the trajectory which more successful and less likely to appear in unsuccessful trajectory.

Such heuristic method shows, certain states has to be covered to reach out the goal.While doing so, it may happen that in certain trajectory it model may take more time and on certain path. For example in room exploration case, agent takes more time one room and eventually it terminates the sample. If agent spends more time in one room then after certain period the sample will be discarded, which shows the failure of heuristic.