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 : Arabhi Subhash
ROLL NUMBER : cs17b005

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:

Bandit REINFORCE parameter update

$$\Delta\theta_n = \alpha_n (R_n - b_n) \left\{ \frac{\partial \ln \pi(a_n;\theta)}{\partial \theta} \right\}$$

Here $b_n = 0$, $\pi(a,\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(a-\mu)^2}{2\sigma^2}}$

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

$$\frac{\partial \ln \pi}{\partial \sigma} = -\frac{(a-\mu)^2}{2} \cdot -2 \cdot \sigma^{-3} - \frac{1}{\sigma}$$

$$= \left[\frac{(a-\mu)^2}{\sigma^3} - \frac{1}{\sigma} \right] = \frac{(a-\mu)^2 - \sigma^2}{\sigma^3}$$

The parameter update conditions -

$$\Delta\mu = \frac{\alpha R (a-\mu)}{\sigma^2} \simeq \alpha R (a-\mu)$$

$$\Delta\sigma = \frac{\alpha R ((a-\mu)^2 - \sigma^2)}{\sigma^3} \simeq (\alpha/\sigma)((a-\mu)^2 - \sigma^2)$$

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:** Policy gradient approach is better because it has better convergence properties, it is effective in continuous action spaces and it can learn stochastic policies.

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

   > **Solution:** PG works on gradient descent, so it typically converge at local rather than the global minima and have high variance. In these circumstances value function based approach is better.

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

   > **Solution:** Environments with less variance in their returns for similar trajectories typically yield optimal policy with either of the methods.

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:

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

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

   > **Solution:** The equation as such is correct but we can make it more appropriate for practical usage by separating target network and update network
   >
   > $$\Delta\theta = \alpha(R_{t+1} + \gamma V(S_{t+1}, \theta^*) - V(S_t, \theta))\nabla_\theta V(S_t, \theta) \tag{2}$$
   >
   > Using same network, the algorithm will not converge as expected

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

Solution: The ways are online mode and experience replay mode. In the online mode as we know we sample, use and throw it away. That itself is the disadvantage of this method. The better one is replay memory in which we sample, store it in the replay buffer and once in a while we replay from the memory without interacting with the system. We should be keen in using an off policy algorithm like Q-learning in later way.

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: In experience replay we store large number of most recent samples. The Q network is updated from the batches sampled uniform-random from this buffer. This way we use same sample many times increasing data efficiency and randomizing the samples reduce variance as it breaks the correlations between adjacent 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 equal importance to all samples but that doesn't guarantee that equal stability or variance among different transitions. Hence, by picking transitions of higher TD-error more frequently we would reduce overall TD-error which is our main motto. In literature this is called Prioritized Experience Replay and it is seen to have better performance over general one in many Atari games.

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: Even before the introduction of policy gradient methods there are several actor only, critic only and actor-critic methods. Here the terminology actor and critic is similar to that of policy and value function. Actor methods work on policies on which different optimisations can be applied. They suffer from high variance.

Critic only methods select greedy actions thus have lower variance but computationally expensive and not applicable to continuous action spaces. The original AC is combining these two ideas, the critic's estimate of the expected return allows for the actor to update with gradients that have lower variance, speeding up the learning process.

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: Out of 5 conditions cited in the paper, Max Node Irrelevance, Leaf Irrelevance are necessary even we do not use value function decomposition.

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: Discretization undermines the ability of using continuous actions and will not yield true optimum. If we descrtize into less number of actions the error we calculate using $J(w) = (1/N) * \sum_{i=1}^{n} (y_j - q(sj, aj, w))^2$ will vary largely from original error and there will also be large variance in Q(s,a). If we descrtize into more number of actions then we will be having big last or output layer and is difficult to train.

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

   Solution: DDPG is an off-policy algorithm hence so we can take the exploration and learning problems independently. Here, we add some noise to our actor policy to ensure the exploration. $\mu'(s_t) = \mu(s_t|\theta_t^{\mu}) + \mathcal{N}$

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 such a heuristic could fail.

Solution: Bottle necks are states or sets of states which are present in every trajectory. As every trajectory should pass through them the optimal policy from start to bottleneck combined with policy from bottleneck to terminal state will be optimal for whole journey. Hence bottlenecks are useful sub-goals. For identification of bottlenecks we use heuristics like betweeness, graph partition, diverse density etc. As we are using trails to determine them, we cannot be sure of a bottleneck even after many tries. If the bottle neck is a set of states, the optimal for sub environment and the true optimum might not pass through same state in this set.