CS6700 - Reinforcement Learning - Written Assignment 3

Atul Balaji

April 3, 2019

1 Question 1

1.1 (a)

Policy gradient methods can learn stochastic policies while only deterministic policies can be learned using value function based methods which means that they can represent only a smaller variety of functions than policy gradient methods. According to the question, the optimal policy cannot be represented by either of the methods. Therefore, both the algorithms try to approximate the optimal policy. But since policy gradient methods can represent a larger span of policies compared to value function based methods, the policy gradient methods are better suited to approximate the optimal policy.

1.2 (b)

In some (s,a) to take optimal actions, it is enough for the value of $q(s, a)$ to be greatest. If the estimated policy is such that it has maximum value for (s,a) pairs which lead to optimality, then it can act as a proxy for the optimal policy. However, the policy obtained using the policy gradient method cannot behave similar to the optimal policy. Therefore, in this circumstance, the value function based approach outperforms the policy gradient method.

1.3 (c)

The value function method can behave like optimal policy for a set of $q(s, a)$ values given the circumstance in (b).
Unlike the value function based method, the policy gradient method can neither converge to the optimal policy nor behave like it. Only an approximation of the optimal policy can be done in the policy gradient method unlike the value function method. Therefore, only in the circumstance of (b) it is possible to find optimal policy (where value function method is used).

2 Question 2

2.1 (a)

We have $\phi(s) = \begin{bmatrix}\phi_1(s)\\ \phi_2(s)\\ \phi_3(s)\end{bmatrix}$ for states s_1 and s_2. We can write the value function vector(V) as

$$V = \phi\theta \tag{1}$$

where $V = \begin{bmatrix}V(s_1)\\ V(s_2)\end{bmatrix}$, $\theta = \begin{bmatrix}\theta_1\\ \theta_2\\ \theta_3\end{bmatrix}$ is the parameter vector, and $\phi = \begin{bmatrix}1 & -1 & -1\\ -1 & -1 & 1\end{bmatrix}$.
Now, $V = \begin{bmatrix}\theta_1 - \theta_2 - \theta_3\\ -\theta_1 - \theta_2 + \theta_3\end{bmatrix}$
V is a point in the 2-D space and can therefore be represented by suitable values of θ_i's. Let us take $V^*(s_1) = x$ and $V^*(s_2) = y$. Then, $x - y = 2\theta_1 - 2\theta_3$ and $x + y = -2\theta_2$. Taking $\theta_3 = z$, we have: $\theta_1 = \frac{x-y}{2} + z$ and $\theta_2 = \frac{-x-y}{2}$.
Given x and y, we have three variables θ_1, θ_2 and θ_3. This means that for any point in the 2-D space, values of parameters θ_i can be found and therefore, the whole space can be represented using these features. Hence any value function can be learned.

2.2 (b)

Linear gradient descent TD(0) update rule is given below.

$$\theta_{t+1} \leftarrow \theta_t + \alpha[R_{t+1} + \gamma V_t(s_{t+1}) - V_t(S_t)]\nabla_{\theta_t} V_t(s_t) \tag{2}$$

For the given experience $s_2, a_2, (-5), s_1, a_1$, the $TD(0)$ backup equation is:

$$\theta_{t+1} \leftarrow \theta_t + \alpha[-5 + \gamma V(s_1] - V(s_2)] \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}, \tag{3}$$

where $\begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} = \phi(s_2)$

This implies that: $\begin{bmatrix} \theta_{t+1}^{(1)} \\ \theta_{t+1}^{(2)} \\ \theta_{t+1}^{(3)} \end{bmatrix} = \begin{bmatrix} \theta_t^{(1)} - h \\ \theta_t^{(2)} - h \\ \theta_t^{(3)} + h \end{bmatrix}$ where $h = \alpha(-5 + \gamma V(s_1) - V(s_2))$.

3 Question 3

3.1 (a)

Linear function approximator is being used for TD(0). The feature vector for a particular state 's' is denoted by $\phi(s)$. For the linear function approximator,

$$V(s; \theta) = \theta^T \phi(s) \tag{4}$$

Using eligibility traces we can change the update rules by also including past-gradient information. This gives rise to $TD(\lambda)$. Let e_t denote the eligibility vector at time instant t.

$$e_t = \gamma \lambda e_{t-1} + \nabla_{\theta_t} V(s_t; \theta_t) \tag{5}$$

$$e_t = \gamma \lambda e_{t-1} + \phi(s_t) \tag{6}$$

Using e_t, the update rule for θ becomes:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \delta_t e_t \tag{7}$$

where $\delta_t = R_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ is the TD error at time step t.

3.1.1 Algorithm

Input: policy π
Initialize weights θ for value function randomly
Repeat (for each episode):
    State = s_t
    Obtain feature vector $\phi(s_t)$
    $e_t \leftarrow 0$
    $V_o(s_t) \leftarrow 0$
    Repeat (for each step in an episode):
        Choose a $\sim \pi$
        Take action a, obtain reward R, ϕ' (feature vector of next state)
        $V \leftarrow \theta^T \phi$
        $V' \leftarrow \theta^T \phi'$
        $e \leftarrow \gamma \lambda e + (1 - \alpha \gamma \lambda e^T \phi) \phi$
        $\delta \leftarrow R + \gamma V' - V$
        $\theta \leftarrow \theta + \alpha(\delta + V - V_{old}) e - \alpha(V - V_{old}) \phi$
        $V_o \leftarrow V'$
        $\phi \leftarrow \phi'$
    exit when a terminal state is reached ($\phi' = 0$)

3.2 (b)

For replacing traces (with feature vector ϕ):

$$e_t(s, a) = \begin{cases} \gamma \lambda e_{t-1}(s, a) & s \neq s_t \\ 1 & s = s_t, a = a_t \\ 0 & s = s_t, a \neq a_t \end{cases}$$

For accumulating traces,

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & s = s_t, a = a_t \\ \gamma\lambda e_{t-1}(s,a) & otherwise \end{cases}$$

Replacing eligibility traces are more appropriate in this case as compared to accumulating traces. This is because the accumulating traces update the parameters more since the value of e_t goes above 1 which does not occur in the replacing case. This leads to the updates to the parameters θ becoming high. So, unless we have a sufficiently low learning rate (with decay), the updates keep overshooting optimality making the learning unstable.

4 Question 4

4.1 (a)

The equation given is incorrect since it does not make use of experience replay while performing updates on the parameter θ and also does not use a separate target network. The transition (S_t, a_t, r_t, S_{t+1}) is stored in the replay buffer D as an experience and is then sampled randomly at training time.
The corrected equation is:

$$\Delta\theta = \alpha E_{(s,a,r,s')\sim D}(r + \gamma V(s', \theta) - V(s, \theta^-))\nabla_\theta V(s, \theta) \tag{8}$$

where θ^- are the parameters of the target network.
The parameters θ^- are updated from time to time.

4.2 (b)

In DQN, while training the Q-network, the target depends on the same parameters we are trying to train (θ). This leads to the learning being unstable. The solution is to use a second network with parameter ϕ^- which comes close to θ, but with a time delay.
The parameters of the target network can be updated in the following ways:

- The target network $\hat{Q}$ is maintained and after every K steps of updates, the parameters of Q-network are cloned to give the new target network parameters $\hat{Q}$: $\theta^- = \theta_{i-1}$.
- The target network parameters are updated once per main network update: $\theta^- = \rho\theta^- + (1-\rho)\theta$, where $0 < \rho < 1$ is a hyperparameter.

The second method is better than the first. This is because in this case the target values are constrained to change slowly as it updates to the new value of θ^- gradually in each step, which improves the stability of learning. The parameters of the target network are thus updated by having them slowly track the Q-Network. However, in the first update the target network parameters are constant for C steps and then updated all at once in the K^{th} step.

5 Question 5

The episodes of experience obtained by the interaction with the environment are stored in a replay buffer.
If training is done using only most recent experience, there is a considerable temporal correlation with previous experience will increase and this leads to an increase in variance and overfitting. To avoid this, we sample a random experience from the replay buffer (experience replay) for learning in order to remove these correlations. This is the role played by experience replay in DQN.
If transitions are sampled more frequently in proportion to the TD-error, rather than uniform random sampling, it is a form of **Prioritized experience replay**. The TD error is a measure of how far the value is from its next bootstrapped estimate, which means that higher the TD error, the more we can learn from that transition. So, less number of updates are needed to achieve the same level of accuracy. Therefore, sampling based on magnitude of TD error will make the learning faster and more effective than random sampling.

6 Question 6

Policy gradient method (Modern motivation):
Policy gradient methods use a parameterization of the policy π_θ with parameter θ. The gradient of an objective function $J(\theta)$ is used for learning. Actions are generated according to a policy π which is the output of a softmax function:

$$\pi_t(s,a) = \frac{e^{p(s,a)}}{\sum_b e^{p(s,b)}}$$

where $p(s, a)$ are the values of the policy parameters of the actor.
For this case, we have:

$$\nabla(log\ \pi_t(a_t|s_t)) = 1 - \pi_t(a_t|s_t)$$

The policy gradient theorem states that:

$$\nabla J(\theta) \propto E_\pi[\nabla(log\ \pi(a|s))(q_\pi(s, a) - b_\pi(s))] \tag{9}$$

The update rule based on policy gradient theorem is:

$$p(s_t, a_t) \leftarrow p(s_t, a_t) + \beta\delta_t(1 - \pi_t(a_t|s_t))$$

Original method:

The update equation is:

$$p(s_t, a_t) \leftarrow p(s_t, a_t) + \beta\delta_t$$

where $\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ and β is a step size parameter.

This update rule does not contain the term $(1 - \pi_t(a_t|s_t))$. Therefore, it is not a policy gradient method.