Q1: A Variant of SARSA

The psuedocode of modified SARSA is given below. A few things to note:

1. The psuedocode gives a general version of modified SARSA for any value of n, i.e.

$$q(x_t, a_t) = r_t + \beta r_{t+1} + \beta^2 r_{t+2} + \cdots + \beta^n r_{t+n} + \beta^{n+1} q(x_{t+n+1}, a_{t+n+1})$$

   In minor 2, $n = 1$ and for the original SARSA that you know $n = 0$.

2. Lines 2, 8, 10, 11, and 13 are different for modified SARSA compared to the original SARSA.

3. Line 13 is important because while computing return, we can't consider reward across two different episodes.

Algorithm 1: Psuedocode of modified SARSA

```
1  For all $x \in \mathcal{S}$ and all $a \in \mathcal{A}(x)$ arbitrarily initialize $q(x, a)$ to any real value.
2  Initialize an empty buffer $G$ of size $n+1$ that will store the reward. The oldest reward (oldest with rest to time)
   is the first element of $G$ and the most recent reward is the last element of $G$. When this buffer is full, and a new
   reward gets appended to it, then the oldest reward is moved out of it. Let $G_i$, where $i = 0, 1, \ldots n$, denote the $i^{th}$
   element of buffer $G$.
3  for every episode do
4      Reset the environment to get the current state $x$.
5      Choose action, $a$, for current state, $x$, using an $\varepsilon$-greedy policy derived using the current Q-function, $q(x, \cdot)$.
6      while episode did not end do
7          Take action, $a$. Environment will return reward, $r$, and go to next state, $x'$.
8          Append reward, $r$, to buffer, $G$.
9          Choose action, $a'$, for the next state state, $x'$, using an $\varepsilon$-greedy policy derived using the current Q-function,
           $q(x', \cdot)$.
10         if length$(G) = n+1$ then
11             Update the Q-function only for the current state-action pair $(x, a)$ as follows:
                   $q(x,a) = q(x,a) + \alpha \left( \sum_{i=0}^{n} \beta^i G_i + \beta^{n+1} q(x', a') - q(x,a) \right)$
12             Set $x \leftarrow x'$, and $a \leftarrow a'$ for the next time slot.
13     Empty buffer $G$.
```

Q2: Bandit Setup

(a) The following are the two ways in which training a neural network for policy gradient algorithm is different from training a neural network for classification problems in supervised learning:

1. While training a neural network for classification problem in supervised learning we **minimize a loss function** (more specifically cross entropy loss). But, while training neural network for policy gradient, we **maximize the reward**.

2. A training sample for classification problem in supervised learning consist of a **feature** and a **label**. But, a training sample for policy gradient consist of a **context**, a **action taken**, and a **reward**. Context and the action taken is analogous to feature and label respectively. But, the reward is very specific to policy gradient which can be interpreted as a weight of the sample.s

(b) Let $r_{i,m}$ denote the m^{th} sample of the reward corresponding to action i. Then the sample average reward is,

$$Q(i) = \frac{1}{M} \sum_{m=1}^{M} r_{i,m}$$

Let $q_\star(i)$ and σ_i be the true mean and standard deviation of reward corresponding to action i. Now, $Q(i)$ is a random variable because it is a function of random variable $r_{i,m}$. The following are from college-level elementary probability course:

1. The mean of $Q(i)$ is $q_\star(i)$. This can be obtained using linearity of expectation.
2. The standard deviation of $Q(i)$ is $\frac{\sigma_i}{\sqrt{M}}$. This true for iif samples only which is a typical assumption.
3. Since $r_{i,m}$ are Gaussian random variables, $Q(i)$ too is a Gaussian random variable (sum of Gaussian is Gaussian). Since $Q(i)$ is Gaussian, it's mean and standard deviation is enough to characterize the probabability distribution, i.e.
$$Q(i) \sim \mathcal{N}\left(q_\star(i), \frac{\sigma_i}{\sqrt{M}}\right).$$
4. Since $Q(i)$ is Gaussian, therefore with a probability of 0.95, $Q(i)$ lies in the interval $q_\star(i) - 1.96\frac{\sigma_i}{\sqrt{M}}$ to $q_\star(i) + 1.96\frac{\sigma_i}{\sqrt{M}}$. This interval corresponds to a maximum percentage error of $\frac{1.96\sigma_i}{q_\star(i)\sqrt{M}} \cdot 100$.

We want,

$$\begin{aligned}\frac{1.96\sigma_i}{q_\star(i)\sqrt{M}} \cdot 100 &\leq 5 \\ M &\geq \left(\frac{1.96\sigma_i}{q_\star(i)} \cdot 20\right)^2 \qquad (1)\end{aligned}$$

Now we don't know $q_\star(i)$ and σ_i in a MAB setup. But as given in the question, $q_\star(i) \in [1, 5]$ and $\sigma_i \in [0.25, 4]$. To ensure that holds for any value of $q_\star(i)$ and σ_i, we substitute the lowest value of $q_\star(i)$ and the highest value of σ_i in (1). We get,

$$\begin{aligned}M &\geq \left(\frac{1.96 \cdot 4}{1} \cdot 20\right)^2 \\ M &\geq 24586.24\end{aligned}$$

Finally, the required value of M is 24587.

Q3: MDP for Cloud Computing

(a) The state is $(d, z_1, z_2, \ldots, z_{\tau-1})$ where d is the current demand and z_i, where $i = 1, 2, \ldots, \tau - 1$, is the number of demands not served i time slots back.

Now, the demand is either 0 or 1. Hence, $d \in \{0, 1\}$. Since the demand is at most 1, the maximum number of demands not served in a time slot can be at most 1. Hence, $z_i \in \{0, 1\}$, $\forall i = 1, 2, \ldots, \tau - 1$. Hence, the state space is

$$\{0,1\} \times \underbrace{\{0,1\} \times \{0,1\} \times \cdots \times \{0,1\}}_{\tau-1 \text{ times}} = \{0,1\}^\tau$$

(b) The action is (u, θ) where u is the type of instance and θ is the bid.

$u \in \{1, 2\}$ where $u = 1$ and $u = 2$ implies on-demand instance and resrved instance respectively. As mentioned in the question, bid $\theta \in \{\delta, 2\delta, \ldots, N\delta\}$. When $u = 1$, the only value θ can assume is 0 because if the agent decides to use on-demand instance in the current time slot, it does not bid. Accordingly, the action space is,

$$\mathcal{A}(d, z_1, z_2, \ldots, z_{\tau-1}) = \begin{cases}\{(1,0)\} & , z_1 + z_2 + \ldots + z_{\tau-1} = \eta \\ \{(1,0), \{2,\delta\}, \{2, 2\delta\}, \ldots, \{2, N\delta\}\} & , z_1 + z_2 + \ldots + z_{\tau-1} < \eta\end{cases}$$

Explanation: Remember that according to the question, demand not processed in the last τ time slots can't be greater than η. This leads to two cases:

1. $z_1 + z_2 + \ldots + z_{\tau-1} = \eta$: This implies that demand not processed in the last $\tau - 1$ time slots has reached the threshold η. Hence, the agent MUST serve the demand in the current time slot. This is only possible is the agent buys an on-demand instance because with spot instance there is always a probability of loosing.

2. $z_1 + z_2 + \ldots + z_{\tau-1} < \eta$: In this case, the agent can use both on-demand and spot instances.

(c) The average reward corresponding to a state-action (x, a) pair is

$$r\left(\underbrace{d, z_1, z_2, \ldots, z_{\tau-1}}_{x}, \underbrace{u, \theta}_{a}\right) = \begin{cases} -\alpha & , u = 1 \\ -\rho_\theta \cdot \theta & , u = 2 \end{cases}$$

Explanation: When $u = 2$ the agent bids for spot instance. If bid is θ then the agent wins the auction with probability ρ_θ and pays its bid θ and looses the auction with probability $1 - \rho_\theta$ and pays 0. Hence, the average reward is $\rho_\theta \cdot -\theta + (1 - \rho_\theta) \cdot 0 = -\rho_\theta \cdot \theta$.

(d) Let $\left(d^{'}, z_1^{'}, z_2^{'}, \ldots, z_{\tau-1}^{'}\right)$ be the next state. We first discuss how to get the next state $z_i^{'}$ using the current state $(d, z_1, z_2, \ldots, z_{\tau-1})$ and the current action (u, θ). Note that, the demand not served in the current time slot i time slots back is equal to the demand not served in the next time slot $i + 1$ time slots slots back. Therefore,

$$z_2^{'} = z_1 \quad (2)$$
$$z_3^{'} = z_2 \quad (3)$$
$$\vdots \quad \vdots \quad \vdots$$
$$z_{\tau-1}^{'} = z_{\tau-2} \quad (4)$$

But, what about $z_1^{'}$? If the demand d in the current time slot is served then $z_1^{'} = 0$. If the demand d in the current time slot is NOT served then $z_1^{'} = d$.

The Bellman optimality equation is,

$$V^\star(d, z_1, z_2, \ldots, z_{\tau-1}) = \max_{(u,\theta) \in \mathcal{A}(d, z_1, z_2, \ldots, z_{\tau-1})} q^\star(d, z_1, z_2, \ldots, z_{\tau-1}, u, \theta)$$

where,

$$q^\star(d, z_1, z_2, \ldots, z_{\tau-1}, 1, 0) = -\alpha + \beta \sum_{d^{'} \in \{0,1\}} p_{d,d^{'}} V^\star\left(d^{'}, 0, z_1, \ldots, z_{\tau-2}\right) \quad (5)$$

$$q^\star(d, z_1, z_2, \ldots, z_{\tau-1}, 2, \theta) = -\rho_\theta \cdot \theta + \beta \sum_{d^{'} \in \{0,1\}} p_{d,d^{'}} \left(\rho_\theta V^\star\left(d^{'}, 0, z_1, \ldots, z_{\tau-2}\right) + (1 - \rho_\theta) V^\star\left(d^{'}, d, z_1, \ldots, z_{\tau-2}\right)\right), \forall \theta \quad (6)$$

Explanation:

1. $z_1, \ldots, z_{\tau-2}$ in the RHS of the equations (5) and (6) can be explained using equations (2) to (4).
2. Equation (5) is when the action is to use on-demand instance. In this case, the current demand d definitely gets served and hence is the 0 in $V^\star\left(d^{'}, 0, z_1, \ldots, z_{\tau-2}\right)$.
3. Equation (6) is when the action is to use on-demand instance. In this case, the probability of winning is ρ_θ. If the agent wins a spot instance, the current demand d gets served. This explains the term $\rho_\theta V^\star\left(d^{'}, 0, z_1, \ldots, z_{\tau-2}\right)$. If the agent does not win a spot instance, which happens with probability $(1 - \rho_\theta)$, the current demand d does NOT get served. This explains the term $(1 - \rho_\theta) V^\star\left(d^{'}, d, z_1, \ldots, z_{\tau-2}\right)$.