Sistemas Inteligentes PRÁCTICA 1 (25% Nota final)

Utilizando el entorno `'Taxi-v3'` de la librería **Gymnasium**:

Ejercicio 1 (1 punto)

Crea el entorno y descríbelo, indicando, al menos:

- Tipo de espacio de observación y de acción.
- Número de estados y de acciones posibles.
- Sistema de recompensas (cuándo recibe recompensas positivas, negativas o nulas).
- Acciones posibles que el agente puede ejecutar.
- Condiciones de terminación del episodio.

Ejercicio 2 (2 puntos)

Implementa el algoritmo ***Monte Carlo on-policy first-visit***. Define los hiperparámetros utilizados y ejecuta el algoritmo para obtener la tabla Q de valores óptimos y derivar la política óptima.

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \text{argmax}_a\, Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

Ejercicio 3 (3 puntos)

Implementa el algoritmo ***Double Q-learning***. Define los hiperparámetros de entrenamiento utilizados y obtén la tabla *Q* de valores óptimos, de la cual se derivará la política óptima.

Double Q-learning, for estimating $Q_1 \approx Q_2 \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, such that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using the policy ε-greedy in $Q_1 + Q_2$
    Take action A, observe R, S'
    With 0.5 probabilility:
      $Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \arg\max_a Q_1(S', a)\big) - Q_1(S, A)\Big)$
    else:
      $Q_2(S, A) \leftarrow Q_2(S, A) + \alpha\Big(R + \gamma Q_1\big(S', \arg\max_a Q_2(S', a)\big) - Q_2(S, A)\Big)$
    $S \leftarrow S'$
  until S is terminal

En *Q-learning*, la actualización de los valores *Q* depende de elegir la acción que maximiza el valor futuro usando la misma tabla *Q* que se actualiza. Esto introduce un sesgo positivo hacia la sobreestimación.

Double Q-learning soluciona este problema manteniendo dos tablas *Q*: Q_1 y Q_2. La política ε-greedy selecciona la acción en base a la suma de ambas tablas $Q_1 + Q_2$ y actualiza con probabilidad del 50% cada una de las tablas. De este modo, una tabla elige la acción y la otra estima su valor, evitando que ambas operaciones se realicen con la misma *Q*.

Ejercicio 4 (1 punto)

Ejecuta un episodio siguiendo la política aprendida con ***Monte Carlo*** o ***Double Q-learning*** y genera una animación a partir de los *frames* capturados durante la ejecución del episodio que ilustre el comportamiento del agente bajo la política obtenida.

Ejercicio 5 (3 puntos)

Implementa el algoritmo n-step SARSA. Define los hiperparámetros necesarios para su entrenamiento. Manteniendo estos fijos, obtén la tabla *Q* de valores óptimos para los distintos valores de $n = \{1, 2, 4, 6, 8\}$.

Registra la recompensa total por episodio y grafica su evolución para los distintos valores de *n*.

Considera el error TD definido como $\delta_\tau = G - Q(S_\tau, A_\tau)$. Registra el valor medio de $|\delta|$ por episodio y grafica su evolución para los distintos valores de *n*.

Para obtener curvas más suaves, utiliza la función `moving_average` proporcionada para aplicar una media móvil (SMA) al registro de recompensas y de error TD antes de graficar.

```
def moving_average(x, w=100):
    out = np.empty_like(x, dtype=float)
    s = 0.0; k = 0
    for i, v in enumerate(x):
        s += v; k += 1
        if k > w: s -= x[i - w]; k -= 1
        out[i] = s / k
    return out
```

n-step Sarsa for estimating $Q \approx q_*$ or q_π

Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be ε-greedy with respect to Q, or to a fixed given policy
Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n
All store and access operations (for S_t, A_t, and R_t) can take their index mod $n+1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$:
  | If $t < T$, then:
  | Take action A_t
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | If S_{t+1} is terminal, then:
  | $T \leftarrow t+1$
  | else:
  | Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$
  | $\tau \leftarrow t-n+1$ (τ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau+n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\left[G - Q(S_\tau, A_\tau)\right]$
  | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
  Until $\tau = T-1$