Modelado de Incertidumbre en Inteligencia Artificial

Solución del problema Cart-Pole

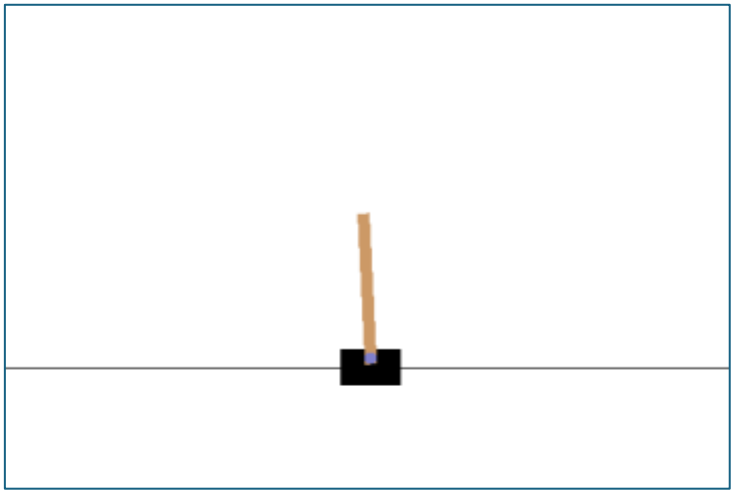
Alumnos:

Juan Herencia,
Jeyson Lino,
José Zúñiga

1. Introducción

Descripción del problema

Cartpole, también conocido como péndulo invertido, es un péndulo con un centro de gravedad por encima de su punto de pivote. Es inestable, pero se puede controlar moviendo el punto de pivote por debajo del centro de masa. El problema es la inestabilidad del poste cuando el carrito está en movimiento horizontal. La solución del problema es mover el carrito de manera que el poste se mantenga en posición vertical (o lo más cercano a la vertical posible) sin que se caiga.



Cart-Pole – Péndulo invertido

Objetivos del proyecto

Encontrar la solución al problema del Cart-Pole (péndulo invertido) usando algoritmos de aprendizaje por refuerzo que puedan controlar sistemas dinámicos.

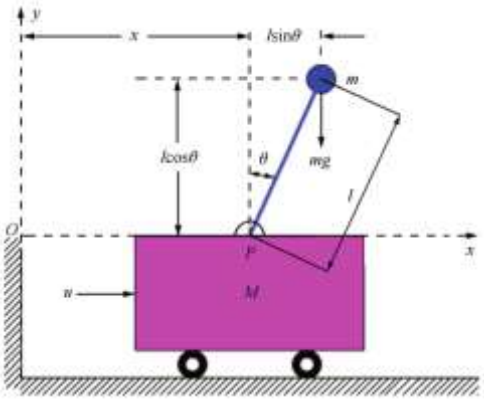
Criterio de Éxito

El problema se considera resuelto si el poste se mantiene en equilibrio por una cierta cantidad de pasos de tiempo (steps)

2. Marco Teórico

Dinámica del sistema Cart-Pole

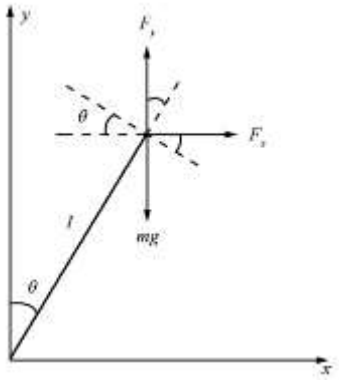
Explicación matemático-física de la dinámica del sistema Cart-Pole ó péndulo invertido, que es un clásico problema de control en física y matemáticas. Consiste en un carrito que se mueve en una pista horizontal con un poste (péndulo) unido por un pivote al carrito. El objetivo es aplicar fuerzas horizontales a la izquierda o derecha del carrito para mantener el poste en posición vertical, el cual es inherentemente inestable.



Símbolo	Descripción
M	Masa del carro
m	Masa del centro de gravedad
ℓ	Longitud de la barra
I	Inercia del péndulo
u	Fuerza aplicada al carro
x	Posición del carro
θ	Ángulo del péndulo desde la vertical superior
g	Constante gravitacional
P	Pivote

Sistema del Péndulo invertido y sus variables

Todo el sistema se reduce a:



Sistema principal del Cart-Pole

Las ecuaciones para dicho sistema son:

$$x_{cg} = x + l.sen(\theta) \quad \text{..... (1)}$$

$$y_{cg} = l.\cos(\theta) \quad \text{..... (2)}$$

$$F_x = M.\ddot{x} + m.\ddot{x}_{cg} = u(t) \quad \text{..... (3)}$$

$$F_y = m.\ddot{y}_{cg} + m.g \quad \text{(el peso } M.g \text{ del carro no influye en este sistema)} \quad \text{..... (4)}$$

Reemplazando (1) y (2) en (3) y (4) respectivamente se obtiene:

$$\ddot{x} = -\frac{m}{M}.g.\theta + \frac{1}{M}\,u(t) \quad \text{..... (5)}$$

$$\ddot{\theta} = \frac{(M-m)}{M.l}.g.\theta - \frac{1}{M.l}\,u(t) \quad \text{.....(6)}$$

Haciendo cambio de variables para las ecuaciones (5) y (6):

$$x_1 = \theta,\ x_2 = \dot{\theta},\ x_3 = x,\ x_4 = \dot{x}$$

Se lleva a la forma matricial:

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \dot{x}_4\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ g\frac{(M+m)}{M\ell} & 0 & 0 & 0\\ 0 & 0 & 0 & 1\\ -g\frac{m}{M} & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix} + \begin{bmatrix}0\\ \frac{-1}{M\ell}\\ 0\\ \frac{1}{M}\end{bmatrix}u(t) \quad \text{..... (7)}$$

$$y = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix} \quad \text{.....(8)}$$

Se resuelve (7) por métodos iterativos para encontrar los valores de los estados x_1, x_2, x_3 y x_4 donde vienen a ser:

Estados de Cart – Pole	
x_3	Posición horizontal del carro
x_4	Velocidad del carro
x_1	Angulo del poste

x_2	Velocidad angular del poste

Opcionalmente los resultados de los estados se reemplazan en (8) para conocer la posición vertical del poste.

Aprendizaje por refuerzo – Reinforcement Learning (RL)

Consta de un agente que aprende a tomar decisiones secuenciales para maximizar una recompensa acumulada a lo largo del tiempo. A diferencia del aprendizaje supervisado, donde el modelo aprende a partir de un conjunto de datos etiquetados, en RL el agente aprende interactuando con su entorno.

Adaptación de la solución de Cart-Pole con Aprendizaje por refuerzo

Objetivo

El objetivo es mover el carrito de manera que el poste se mantenga en posición vertical (o lo más cercano a la vertical posible) sin que se caiga.

Agente y Entorno

- **Agente**: La entidad que toma acciones para interactuar con el entorno, en este caso nosotros a través de un sistema que permite el ingreso de valores.
- **Entorno**: Todo lo que rodea al agente y con lo que el agente interactúa. El entorno responde a las acciones del agente con nuevos estados y recompensas. En este caso es el carrito que se puede mover horizontalmente apoyando a un poste que debe mantenerse verticalmente.

Estado (s)

- Representa la situación actual del entorno. En el problema de Cart-Pole:
  - La posición del carrito.
  - La velocidad del carrito.
  - El ángulo del poste con respecto a la vertical.
  - La velocidad angular del poste

Acción (a)

- Una decisión que toma el agente. En Cart-Pole, las acciones pueden ser mover el carrito hacia la izquierda o hacia la derecha aplicando fuerzas.

Recompensa (r)

- Una señal de retroalimentación que indica qué tan buena fue la acción tomada por el agente. En Cart-Pole, la recompensa puede ser positiva por mantener el poste equilibrado y negativa si el poste se cae.

Política (π)

- Una estrategia que define qué acción tomar en cada estado. La política puede ser determinística (siempre toma la misma acción para un estado dado) o estocástica (toma acciones basadas en una distribución de probabilidad).

Valor (V) y Función Q (Q)

- **Valor de un estado (V(s))**: La recompensa esperada a largo plazo desde un estado dado, siguiendo una política particular.
- **Función Q (Q(s, a))**: La recompensa esperada a largo plazo de tomar una acción particular en un estado dado y luego seguir una política particular.

3. Metodología

Selección de algoritmos

Para estas pruebas se usarán los algoritmos de Q-Learning y Sarsa.

Algoritmo Q-Learning

Q-Learning es un algoritmo de aprendizaje fuera de la política que busca aprender la función de acción-valor $\boldsymbol{Q(s,a)}$. Esta función evalúa la calidad de una acción $\boldsymbol{a}$ en un estado $\boldsymbol{s}$ específico, en términos de la recompensa acumulada esperada al seguir la política óptima a partir de ese estado.

La fórmula de actualización de Q-Learning es:

$$Q(s,a) \leftarrow Q(s,a) + \alpha.[r + \gamma.max_{a'}.Q(s',a') - Q(s,a)]$$

- α: tasa de aprendizaje
- γ: factor de descuento
- r: recompensa ontenida al tomar la acción a en el estado s
- s': nuevo estado después de tomar la acción a

El método Q-Learning es un algoritmo de aprendizaje por refuerzo basado en el valor que puede ser utilizado para resolver el problema del equilibrio del péndulo invertido (Cart-Pole).

Formulación del Problema Cart-Pole con Q-Learning

En el problema Cart-Pole, el agente (el carrito) debe aprender a mantener el poste en equilibrio aplicando fuerzas hacia la izquierda o la derecha. Las acciones disponibles son discretas: aplicar una fuerza hacia la izquierda o hacia la derecha.

Variables de estado

- Posición del carrito.
- Velocidad del carrito.
- Ángulo del poste.
- Velocidad angular del poste.

Pasos para Adaptar Q-Learning al Problema Cart-Pole

i. **Discretización del Espacio de Estados**:
Dado que Q-Learning es más manejable en espacios de estado discretos, el espacio de estado continuo del problema Cart-Pole debe ser discretizado. Esto se puede hacer dividiendo cada variable de estado en intervalos discretos.

ii. **Inicialización de la Tabla Q**:
Crear una tabla Q con todas las combinaciones posibles de estados discretos y acciones, inicializándola a cero o a valores pequeños.

iii. **Interacción con el Entorno**:
En cada paso de tiempo, el agente observa el estado actual s, selecciona una acción a basada en una política ϵ-greedy (para balancear exploración y explotación), y recibe una recompensa r y un nuevo estado s'.

iv. **Actualización de la Tabla Q**:
Utilizar la fórmula de actualización de Q-Learning para actualizar el valor Q para el par (estado, acción) actual.

v. **Repetición del Proceso**:
Repetir el proceso durante múltiples episodios hasta que la política del agente converja a una política óptima.

Algoritmo SARSA

El método **SARSA (State-Action-Reward-State-Action)** es un algoritmo de aprendizaje por refuerzo que aprende la función de acción-valor $Q(s,a)$ en base a la experiencia obtenida de las transiciones del entorno. A diferencia de Q-learning, que es un algoritmo fuera de la política, SARSA es un algoritmo en la política porque aprende de la política que actualmente está siguiendo.

Pasos para Adaptar SARSA al Problema Cart-Pole

i. **Discretización del Espacio de Estados**:
Dado que SARSA trabaja mejor en espacios de estado discretos, discretizaremos las variables continuas del estado del Cart-Pole.

ii. **Inicialización de la Tabla Q**:
Crearemos una tabla Q para almacenar los valores $Q(s,a)$ para cada par de estado-acción.

iii. **Selección de la Política**:
Usaremos una política ϵ-greedy para balancear la exploración y la explotación.

iv. **Actualización de la Tabla Q**:
Actualizaremos la tabla Q usando las transiciones observadas del entorno.

4. Herramientas utilizadas

Librería de aprendizaje con refuerzo Gymnasium

Entorno CartPole-v0

- Se puede considerar como una versión más simple del problema, adecuada para comenzar a experimentar con algoritmos de aprendizaje por refuerzo.
- Útil para probar rápidamente ideas nuevas o para fines educativos debido a su límite de episodios más corto (200 pasos).
- Condiciones de Terminación:
  - El episodio termina si el poste se inclina más de 15 grados desde la vertical.
  - El episodio termina si el carro se desplaza más de 2.4 unidades desde el centro.
  - El episodio termina si el poste se mantiene en posición durante 200 pasos.
- Recompensas:
  - La recompensa es +1 por cada paso de tiempo en el que el poste se mantiene en posición.

Entorno CartPole-v1

- Ofrece un desafío más difícil debido a los límites más estrictos en el ángulo del poste y la duración más larga del episodio (500 pasos).
- Es más adecuada para probar la robustez de algoritmos de aprendizaje por refuerzo y para una evaluación más exhaustiva del rendimiento del agente.
- Ideal para aplicaciones más avanzadas y para agentes que han superado consistentemente CartPole-v0.
- Condiciones de Terminación:
  - El episodio termina si el poste se inclina más de 12 grados desde la vertical (más estricto que en v0).
  - El episodio termina si el carro se desplaza más de 2.4 unidades desde el centro (igual que en v0).

- El episodio termina si el poste se mantiene en posición durante 500 pasos (más largo que en v0).

- Recompensas:
  - La recompensa es +1 por cada paso de tiempo en el que el poste se mantiene en posición.

Estado inicial

En todas las implementaciones se han considerado los mismos estados iniciales:

Estado inicial de Cart – Pole	
Posición horizontal del carro	0 m
Velocidad del carro	0 m/s
Angulo del poste	45° sexag. (π/4 rad)
Velocidad angular del poste	0 rad/s

Implementación

i. Implementación de la solución del sistema de Cart-Pole mediante la física clásica, usándose las ecuaciones matemáticas

Además del estado inicial, se ha considerado:

Masa del carro = 1.0 Kg
Masa del poste = 0.3 Kg
Longitud del poste = 0.5 m
Gravedad = 9.81 m/s2

Planteando las ecuaciones del sistema Cart-Pole

Solución iterativa de las ecuaciones

```
# Solucionar las ecuaciones diferenciales
states = [initial_state]
F = F_initial

for i in range(len(t) - 1):
    current_state = states[-1]
    next_state = odeint(cart_pole_dynamics, current_state, [t[i], t[i+1]], args=(F,))[1]
    states.append(next_state)

    # Cambio de fuerza según el ángulo
    if abs(next_state[2]) > abs(current_state[2]):
        F = -F
```

ii. Implementación en Gymnasium de Q-Learning

```python
# Q-Learning
for episode in range(episodes):
    state = discretize_state(env.reset()[0])
    total_reward = 0
    consecutive_steps = 0

    for step in range(max_steps):
        if random.uniform(0, 1) < epsilon:
            action = env.action_space.sample()
        else:
            action = np.argmax(q_table[state])

        next_state, reward, done, _, _ = env.step(action)
        next_state = discretize_state(next_state)

        # Actualización de Q-Table
        q_table[state][action] = q_table[state][action] + learning_rate * (
            reward + discount_rate * np.max(q_table[next_state]) - q_table[state][action])

        state = next_state
        total_reward += reward

        # Verificar si el poste se ha mantenido en posición por más de 500 pasos consecutivos
        if abs(state[2]) < np.pi / 20:
            consecutive_steps += 1
        else:
            consecutive_steps = 0

        if consecutive_steps >= 500:
            done = True

        if done:
            break

    print(f'Episodio {episode}, Estado final: {state}, Recompensa total: {total_reward}')

    # Reducción del valor de epsilon
    if epsilon > epsilon_min:
        epsilon *= epsilon_decay

    rewards_per_episode.append(total_reward)

    if episode % 100 == 0:
        print(f'Episodio: {episode}, Recompensa total: {total_reward}, Epsilon: {epsilon}')
```

iii. Implementación en Gymnasium de SARSA

```python
# SARSA
for episode in range(episodes):
    state = discretize_state(env.reset()[0])
    total_reward = 0
    consecutive_steps = 0

    # Selección de la acción inicial
    if random.uniform(0, 1) < epsilon:
        action = env.action_space.sample()
    else:
        action = np.argmax(q_table[state])

    for step in range(max_steps):
        next_state, reward, done, _, _ = env.step(action)
        next_state = discretize_state(next_state)

        # Selección de la siguiente acción
        if random.uniform(0, 1) < epsilon:
            next_action = env.action_space.sample()
        else:
            next_action = np.argmax(q_table[next_state])

        # Actualización de Q-Table
        q_table[state][action] = q_table[state][action] + learning_rate * (
            reward + discount_rate * q_table[next_state][next_action] - q_table[state][action])

        state = next_state
        action = next_action
        total_reward += reward

        # Verificar si el poste se ha mantenido en posición por más de 500 pasos consecutivos
        if abs(state[2]) < np.pi / 20:
            consecutive_steps += 1
        else:
            consecutive_steps = 0

        if consecutive_steps >= 500:
            done = True

        if done:
            break

    print(f'Episodio {episode}, Estado final: {state}, Recompensa total: {total_reward}')

    # Reducción del valor de epsilon
    if epsilon > epsilon_min:
        epsilon *= epsilon_decay

    rewards_per_episode.append(total_reward)

    if episode % 100 == 0:
        print(f'Episodio: {episode}, Recompensa total: {total_reward}, Epsilon: {epsilon}')
```

iv. Comparación de los algoritmos Q-Learning y Sarsa

Creacion de diferentes espacios para cada método, los algoritmos se mantienen como los indicados anteriormente

```python
# Función para discretizar el espacio de estados
def discretize_state(state):
    indices = []
    for i in range(state_size):
        indices.append(np.digitize(state[i], bins[i]) - 1)
    return tuple(indices)

# Función epsilon-greedy para seleccionar una acción
def epsilon_greedy_action(q_table, state, epsilon):
    if random.uniform(0, 1) < epsilon:
        return env.action_space.sample()
    else:
        return np.argmax(q_table[state])
```

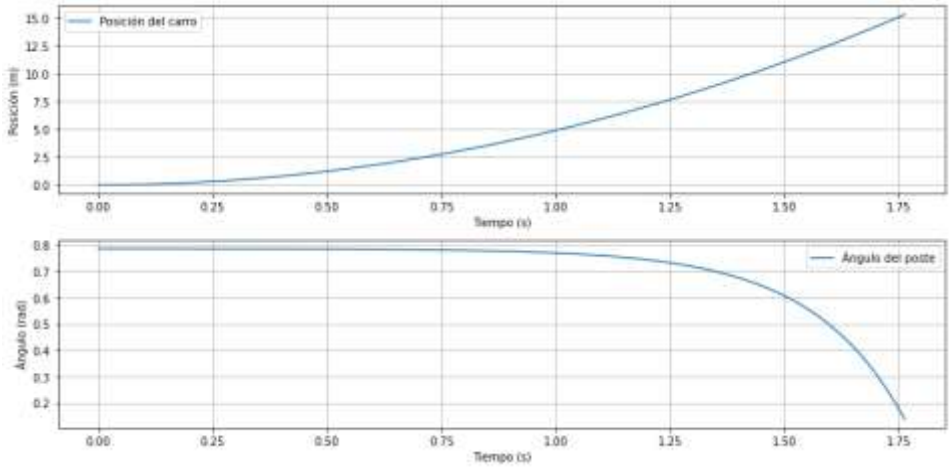
5. Experimentos y Resultados

i. Implementación de la solución del sistema de Cart-Pole mediante la física clásica, usándose las ecuaciones matemáticas

Resultado de 1000 iteraciones resolviendo por Euler las ecuaciones diferenciales:

Caso converge a la solución con Fuerza inicial = 12.76N

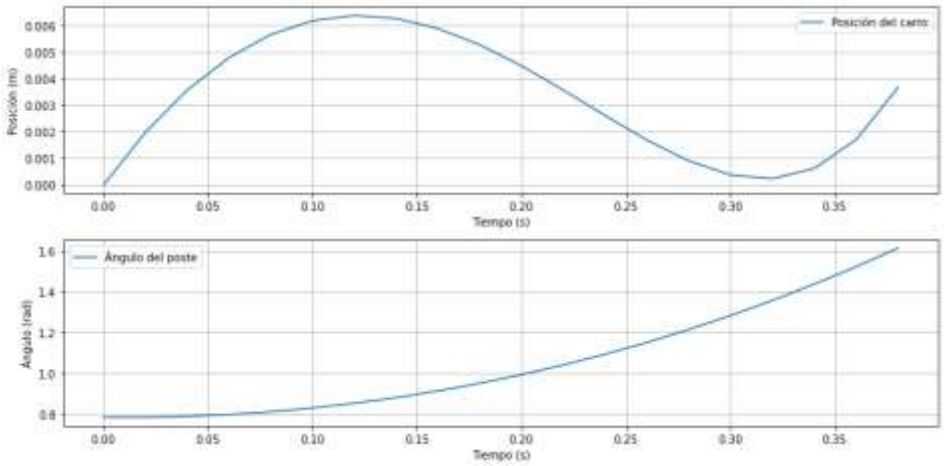
```
iteración 0-[ 1.96711136e-03  1.96514426e-01  7.85396909e-01 -1.25375740e-04]
iteración 50-[ 5.11764872 10.02885172  0.76769028 -0.08597444]
iteración 100-[20.0268162  18.22529945 -0.98485363 -5.38222423]
iteración 150-[37.81885633 18.27286444 -7.1339723  -5.13884912]
iteración 200-[ 55.60990179  18.30742556 -13.28957108  -4.91941127]
iteración 250-[ 73.40025888  18.3314357  -19.45096608  -4.72522064]
iteración 300-[ 91.19016649  18.34726103 -25.61742202  -4.55828056]
iteración 350-[108.97980082  18.35706188 -31.78814162  -4.42094956]
iteración 400-[126.76928369  18.3626926  -37.96225849  -4.3155666 ]
iteración 450-[144.55869325  18.36560688 -44.13884866  -4.24416394]
iteración 500-[162.34807128  18.36680524 -50.31693878  -4.20822087]
iteración 550-[180.13744083  18.36679655 -56.49552446  -4.20850717]
iteración 600-[197.92681825  18.36557392 -62.6735943   -4.24503099]
iteración 650-[215.71622722  18.36262409 -68.85014509  -4.31698987]
iteración 700-[233.50571153  18.35693908 -75.02420023  -4.42290719]
iteración 750-[251.29534866  18.34705171 -81.19483434  -4.56076777]
iteración 800-[269.08526152  18.33110516 -87.3611854   -4.72821279]
iteración 850-[286.87562742  18.30693099 -93.52245375  -4.92289332]
iteración 900-[304.66668685  18.27215526 -99.67790255  -5.14282464]
iteración 950-[ 322.45874662   18.22432243 -105.82685006   -5.38670456]
Resultado esperado en Iteración 88
Posicion del carro = 15.3 metros
Angulo de poste = 0.139 radianes
Tiempo de 1000 episodios      = 161.2548828125 segundos
```



Caso No converge a la solución con Fuerza inicial = 12.<u>75N</u>

```
iteración 0-[1.96542362e-03 1.96345812e-01 7.85398701e-01 5.37373391e-05]
iteración 50-[0.28012742 0.05722024 5.26352997 2.30262451]
iteración 100-[-2.67609439 -7.58144462  5.02097693 -2.17479746]
iteración 150-[-15.05895253 -15.16130293  -0.89282083  -4.84118513]
iteración 200-[-30.51116075 -15.10933506  -6.89823891  -4.2692278 ]
iteración 250-[-45.96548248 -15.09600266 -12.93125657  -3.85628705]
iteración 300-[-61.42031269 -15.09637568 -18.9822215   -3.6310785 ]
iteración 350-[-76.87517326 -15.09663005 -25.04005648  -3.61247243]
```

```
iteración 400-[-92.33002208 -15.09562685 -31.09306709  -3.8021055 ]
iteración 450-[-107.7845114   -15.10500412  -37.12999628   -4.18386191]
iteración 500-[-123.23727911  -15.14796554  -43.14089929   -4.72982466]
iteración 550-[-138.68539919  -15.25532382  -49.11739606   -5.41550168]
iteración 600-[-154.12415886  -15.45846924  -55.05182651   -6.23952778]
iteración 650-[-169.54734028  -15.77815173  -60.93572219   -7.22583611]
iteración 700-[-184.94860295  -16.19229582  -66.75991182   -8.35110318]
iteración 750-[-200.32630046  -16.54413776  -72.52374945   -9.27712114]
iteración 800-[-215.69279586  -16.53858138  -78.25910651   -9.26249949]
iteración 850-[-231.07100622  -16.18157058  -84.02423759   -8.32272063]
iteración 900-[-246.47287346  -15.76868529  -89.84998164   -7.19888719]
iteración 950-[-261.89652707  -15.45194589  -95.73523198   -6.21682312]
Resultado esperado en Iteración 19
Posicion del carro = 0.0 metros
Angulo de poste = 1.615 radianes
Tiempo de 1000 episodios      = 162.358828125 segundos
```



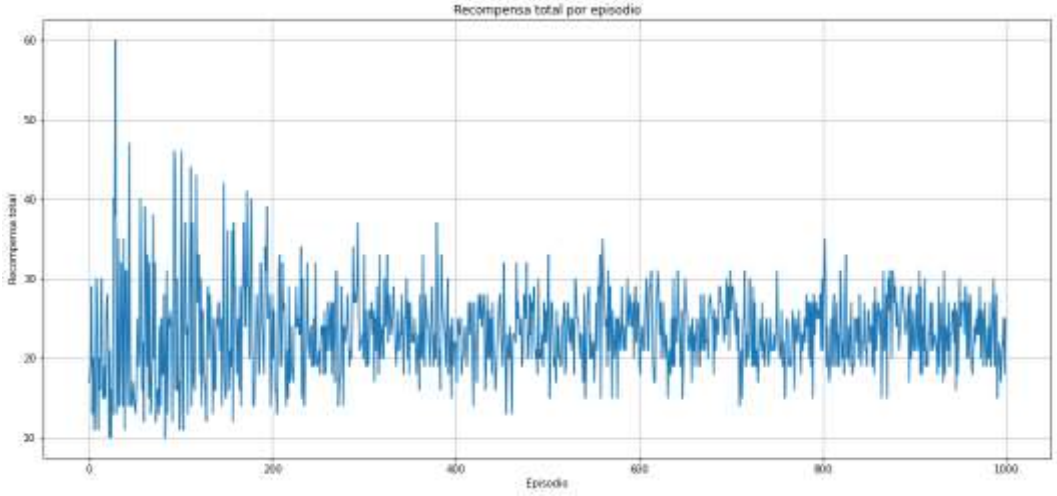
ii. Implementación en Gymnasium de Q-Learning

Converge a la solución

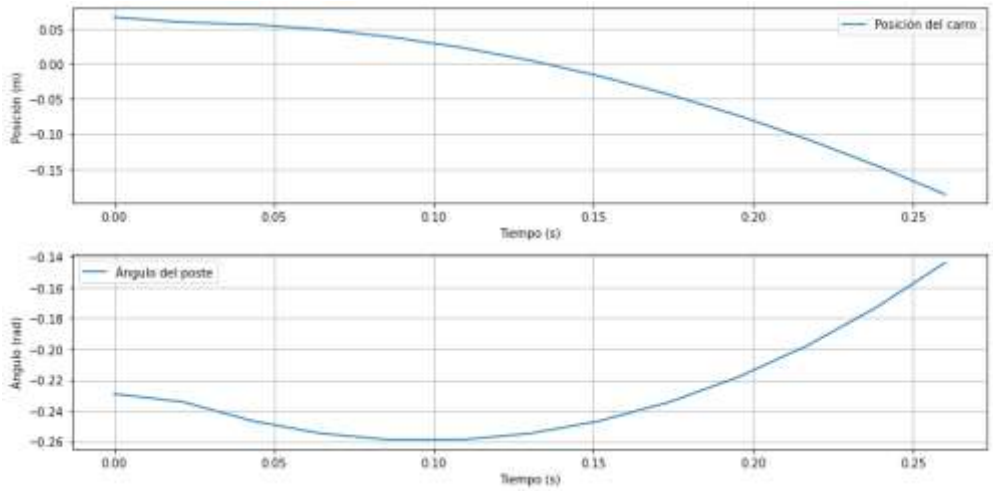
```
Episodio 0, Estado final: (4, 3, 7, 6), Recompensa total: 17.0
Episodio 20, Estado final: (4, 5, 2, 3), Recompensa total: 28.0
Episodio 40, Estado final: (4, 4, 2, 3), Recompensa total: 31.0
Episodio 60, Estado final: (4, 5, 2, 2), Recompensa total: 12.0
Episodio 80, Estado final: (4, 4, 2, 3), Recompensa total: 18.0
Episodio 100, Estado final: (4, 4, 2, 3), Recompensa total: 25.0
Episodio 120, Estado final: (4, 4, 2, 3), Recompensa total: 33.0
Episodio 140, Estado final: (4, 4, 2, 3), Recompensa total: 24.0
Episodio 160, Estado final: (4, 4, 2, 4), Recompensa total: 29.0
Episodio 180, Estado final: (4, 4, 2, 3), Recompensa total: 14.0
Episodio 200, Estado final: (4, 4, 2, 3), Recompensa total: 28.0
Episodio 220, Estado final: (4, 4, 2, 3), Recompensa total: 24.0
Episodio 240, Estado final: (4, 5, 2, 3), Recompensa total: 20.0
Episodio 260, Estado final: (4, 4, 2, 3), Recompensa total: 21.0
Episodio 280, Estado final: (4, 4, 2, 3), Recompensa total: 23.0
Episodio 300, Estado final: (4, 4, 2, 4), Recompensa total: 33.0
Episodio 320, Estado final: (4, 4, 2, 3), Recompensa total: 24.0
Episodio 340, Estado final: (4, 4, 2, 4), Recompensa total: 23.0
Episodio 360, Estado final: (4, 4, 6, 5), Recompensa total: 16.0
Episodio 380, Estado final: (4, 4, 2, 3), Recompensa total: 30.0
Episodio 400, Estado final: (4, 4, 2, 3), Recompensa total: 21.0
Episodio 420, Estado final: (4, 4, 2, 3), Recompensa total: 27.0
Episodio 440, Estado final: (4, 4, 2, 3), Recompensa total: 27.0
Episodio 460, Estado final: (4, 4, 2, 3), Recompensa total: 20.0
Episodio 480, Estado final: (4, 4, 2, 4), Recompensa total: 23.0
Episodio 500, Estado final: (4, 4, 2, 4), Recompensa total: 27.0
Episodio 520, Estado final: (4, 4, 2, 4), Recompensa total: 19.0
Episodio 540, Estado final: (4, 4, 2, 3), Recompensa total: 28.0
Episodio 560, Estado final: (4, 4, 2, 4), Recompensa total: 35.0
Episodio 580, Estado final: (4, 4, 2, 3), Recompensa total: 24.0
```

```
Episodio 600, Estado final: (4, 4, 2, 4), Recompensa total: 19.0
Episodio 620, Estado final: (4, 4, 2, 3), Recompensa total: 31.0
Episodio 640, Estado final: (4, 4, 2, 3), Recompensa total: 22.0
Episodio 660, Estado final: (4, 4, 2, 3), Recompensa total: 23.0
Episodio 680, Estado final: (4, 4, 2, 4), Recompensa total: 25.0
Episodio 700, Estado final: (4, 4, 2, 3), Recompensa total: 26.0
Episodio 720, Estado final: (4, 4, 2, 3), Recompensa total: 30.0
Episodio 740, Estado final: (4, 4, 2, 3), Recompensa total: 22.0
Episodio 760, Estado final: (4, 4, 2, 3), Recompensa total: 24.0
Episodio 780, Estado final: (4, 4, 2, 4), Recompensa total: 25.0
Episodio 800, Estado final: (4, 4, 2, 4), Recompensa total: 21.0
Episodio 820, Estado final: (4, 4, 2, 3), Recompensa total: 28.0
Episodio 840, Estado final: (4, 4, 2, 3), Recompensa total: 28.0
Episodio 860, Estado final: (4, 4, 2, 4), Recompensa total: 27.0
Episodio 880, Estado final: (4, 4, 2, 3), Recompensa total: 24.0
Episodio 900, Estado final: (4, 4, 2, 4), Recompensa total: 21.0
Episodio 920, Estado final: (4, 4, 2, 4), Recompensa total: 21.0
Episodio 940, Estado final: (4, 4, 2, 4), Recompensa total: 23.0
Episodio 960, Estado final: (4, 4, 2, 3), Recompensa total: 28.0
Episodio 980, Estado final: (4, 4, 2, 3), Recompensa total: 24.0
```

Recompensa por episodio



Convergencia al menor ángulo del poste

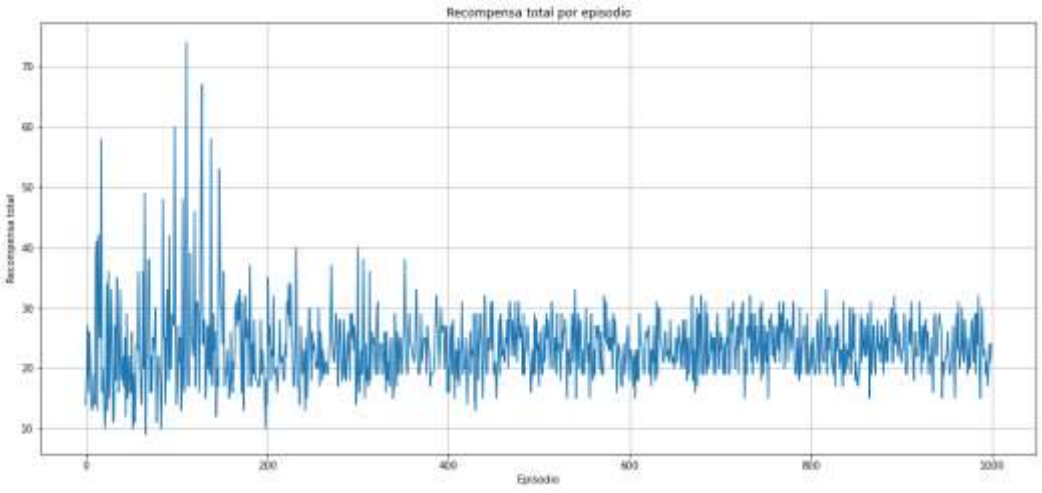


iii. Implementación en Gymnasium de SARSA

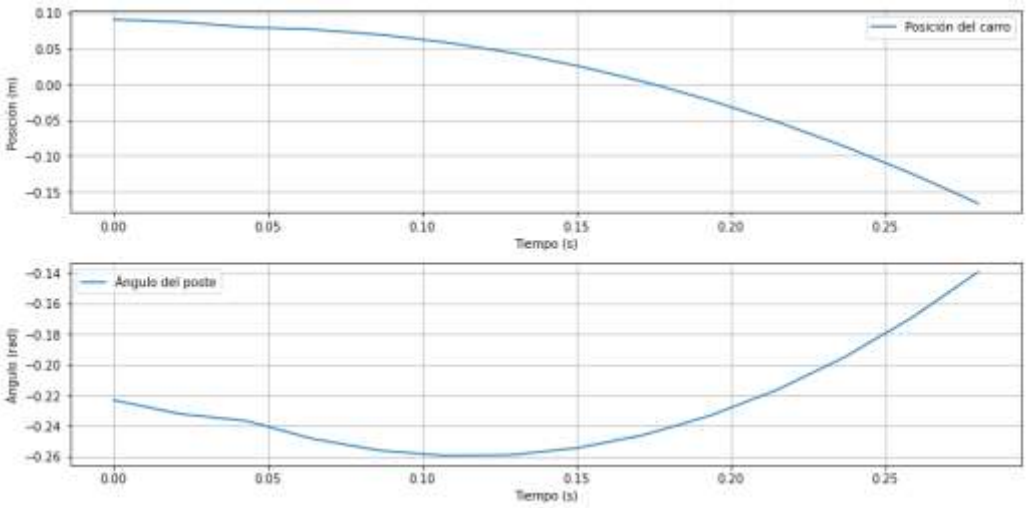
Converge a la solución:

```
Episodio 0, Estado final: (4, 2, 7, 7), Recompensa total: 14.0
Episodio 20, Estado final: (4, 3, 7, 6), Recompensa total: 15.0
Episodio 40, Estado final: (4, 3, 6, 6), Recompensa total: 18.0
Episodio 60, Estado final: (4, 3, 6, 6), Recompensa total: 20.0
Episodio 80, Estado final: (4, 3, 6, 6), Recompensa total: 22.0
Episodio 100, Estado final: (4, 5, 2, 2), Recompensa total: 14.0
Episodio 120, Estado final: (4, 4, 2, 3), Recompensa total: 46.0
Episodio 140, Estado final: (4, 3, 6, 5), Recompensa total: 17.0
Episodio 160, Estado final: (4, 4, 6, 5), Recompensa total: 26.0
Episodio 180, Estado final: (4, 4, 2, 3), Recompensa total: 17.0
Episodio 200, Estado final: (4, 4, 6, 5), Recompensa total: 14.0
Episodio 220, Estado final: (4, 4, 6, 5), Recompensa total: 24.0
Episodio 240, Estado final: (4, 4, 6, 4), Recompensa total: 20.0
Episodio 260, Estado final: (4, 4, 2, 4), Recompensa total: 23.0
Episodio 280, Estado final: (4, 4, 2, 3), Recompensa total: 28.0
Episodio 300, Estado final: (4, 4, 2, 4), Recompensa total: 40.0
Episodio 320, Estado final: (4, 4, 2, 4), Recompensa total: 19.0
Episodio 340, Estado final: (4, 3, 6, 6), Recompensa total: 17.0
Episodio 360, Estado final: (4, 4, 2, 3), Recompensa total: 23.0
Episodio 380, Estado final: (4, 4, 2, 3), Recompensa total: 17.0
Episodio 400, Estado final: (4, 4, 2, 3), Recompensa total: 20.0
Episodio 420, Estado final: (4, 4, 2, 3), Recompensa total: 20.0
Episodio 440, Estado final: (4, 4, 2, 3), Recompensa total: 32.0
Episodio 460, Estado final: (4, 4, 2, 3), Recompensa total: 22.0
Episodio 480, Estado final: (4, 4, 2, 3), Recompensa total: 27.0
Episodio 500, Estado final: (4, 4, 2, 4), Recompensa total: 17.0
Episodio 520, Estado final: (4, 4, 2, 4), Recompensa total: 29.0
Episodio 540, Estado final: (4, 4, 2, 4), Recompensa total: 33.0
Episodio 560, Estado final: (4, 4, 2, 3), Recompensa total: 18.0
Episodio 580, Estado final: (4, 4, 2, 4), Recompensa total: 25.0
Episodio 600, Estado final: (4, 4, 2, 3), Recompensa total: 18.0
Episodio 620, Estado final: (4, 4, 2, 3), Recompensa total: 26.0
Episodio 640, Estado final: (4, 4, 2, 3), Recompensa total: 26.0
Episodio 660, Estado final: (4, 4, 2, 3), Recompensa total: 20.0
Episodio 680, Estado final: (4, 4, 2, 3), Recompensa total: 28.0
Episodio 700, Estado final: (4, 4, 2, 3), Recompensa total: 26.0
Episodio 720, Estado final: (4, 4, 2, 4), Recompensa total: 19.0
Episodio 740, Estado final: (4, 4, 2, 4), Recompensa total: 27.0
Episodio 760, Estado final: (4, 4, 2, 3), Recompensa total: 21.0
Episodio 780, Estado final: (4, 4, 2, 4), Recompensa total: 29.0
Episodio 800, Estado final: (4, 4, 2, 3), Recompensa total: 24.0
Episodio 820, Estado final: (4, 4, 2, 3), Recompensa total: 23.0
Episodio 840, Estado final: (4, 4, 2, 3), Recompensa total: 20.0
Episodio 860, Estado final: (4, 4, 2, 3), Recompensa total: 27.0
Episodio 880, Estado final: (4, 4, 2, 4), Recompensa total: 19.0
Episodio 900, Estado final: (4, 4, 2, 4), Recompensa total: 21.0
Episodio 920, Estado final: (4, 4, 2, 4), Recompensa total: 31.0
Episodio 940, Estado final: (4, 4, 2, 3), Recompensa total: 24.0
Episodio 960, Estado final: (4, 4, 2, 4), Recompensa total: 25.0
Episodio 980, Estado final: (4, 4, 2, 3), Recompensa total: 21.0
```

Recompensa por episodio



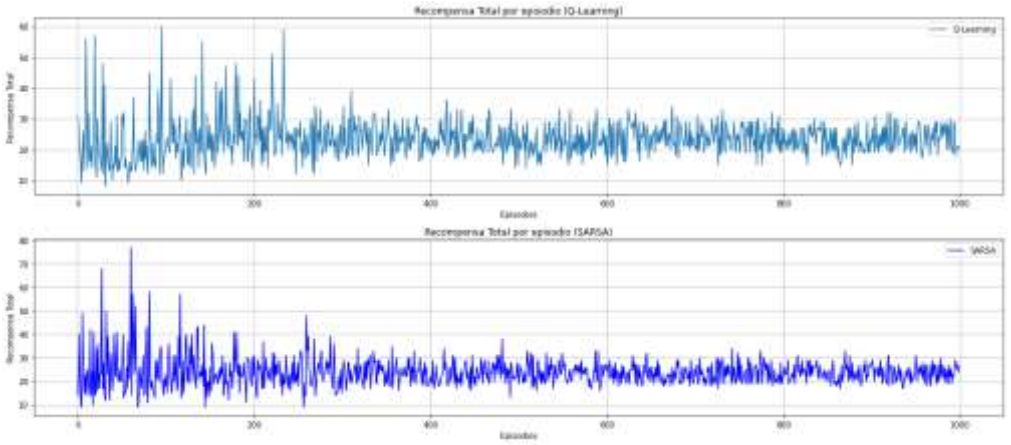
Convergencia al menor ángulo del poste



iv. Comparación de los algoritmos Q-Learning y Sarsa

```
[Q-Learning] Episode: 0, Total Reward: 31.0, Epsilon: 0.995
[Q-Learning] Episode: 100, Total Reward: 22.0, Epsilon: 0.6027415843082742
[Q-Learning] Episode: 200, Total Reward: 43.0, Epsilon: 0.36512303261753626
[Q-Learning] Episode: 300, Total Reward: 24.0, Epsilon: 0.2211807388415433
[Q-Learning] Episode: 400, Total Reward: 23.0, Epsilon: 0.13398475271138335
[Q-Learning] Episode: 500, Total Reward: 24.0, Epsilon: 0.0811640021330769
[Q-Learning] Episode: 600, Total Reward: 15.0, Epsilon: 0.04916675299948831
[Q-Learning] Episode: 700, Total Reward: 21.0, Epsilon: 0.029783765425331846
[Q-Learning] Episode: 800, Total Reward: 22.0, Epsilon: 0.018042124582040707
[Q-Learning] Episode: 900, Total Reward: 22.0, Epsilon: 0.010929385683282892
[SARSA] Episode: 0, Total Reward: 14.0, Epsilon: 0.995
[SARSA] Episode: 100, Total Reward: 23.0, Epsilon: 0.6027415843082742
[SARSA] Episode: 200, Total Reward: 31.0, Epsilon: 0.36512303261753626
[SARSA] Episode: 300, Total Reward: 24.0, Epsilon: 0.2211807388415433
[SARSA] Episode: 400, Total Reward: 22.0, Epsilon: 0.13398475271138335
[SARSA] Episode: 500, Total Reward: 19.0, Epsilon: 0.0811640021330769
[SARSA] Episode: 600, Total Reward: 27.0, Epsilon: 0.04916675299948831
[SARSA] Episode: 700, Total Reward: 22.0, Epsilon: 0.029783765425331846
[SARSA] Episode: 800, Total Reward: 23.0, Epsilon: 0.018042124582040707
[SARSA] Episode: 900, Total Reward: 28.0, Epsilon: 0.010929385683282892
Tiempo Q-Learning = 1.1438782215118408 segundos
Tiempo SARSA      = 0.97505784034729 segundos
Sumarización de Recompensas:
Total Recompensa (Q-Learning): 23404.0
Total Recompensa (SARSA): 23815.0
```

Recompensa por episodio



6. Análisis de Resultados

En el caso de Cart-Pole

Observación	Calculo Tradicional	Q-Learning	SARSA
Cambio de estado inicial	Alta probabilidad de perder la convergencia	Converge	Converge
Tiempo de 1000 episodios (segundos)	162.35	1.14	0.97
Acumulación de recompensas	---	23404.0	23815.0

7. Conclusión

- En los métodos de aprendizaje por reforzamiento el riesgo de perder la convergencia en una búsqueda es mínimo, en comparación con operaciones o calculos matemáticos.
- En gasto de tiempo, SARSA es más rápido que Q-Learning
- Analizando las recompensas, en este caso SARSA supera a Q-Learning, normalmente Q-Learning debería haber sido mas alto, porque es mas goloso y va principalmente en busca de las recompensas, en cambio SARSA va en busca de las mejores políticas.

8. Referencias

- **Gymnasium Cart-Pole Environment**:
  - Cart-Pole Environment in Gymnasium https://gymnasium.farama.org/environments/classic_control/cart_pole/
  - Gymnasium Documentation https://gymnasium.farama.org/

- **Aprendizaje por Refuerzo**:
  - Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto.