Tema 5: Aprendizaje por Refuerzo

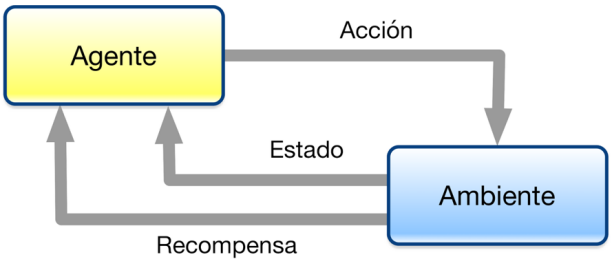
Introducción al Aprendizaje por Refuerzo

Prof. Wladimir Rodriguez

wladimir@ula.ve

Departamento de Computación

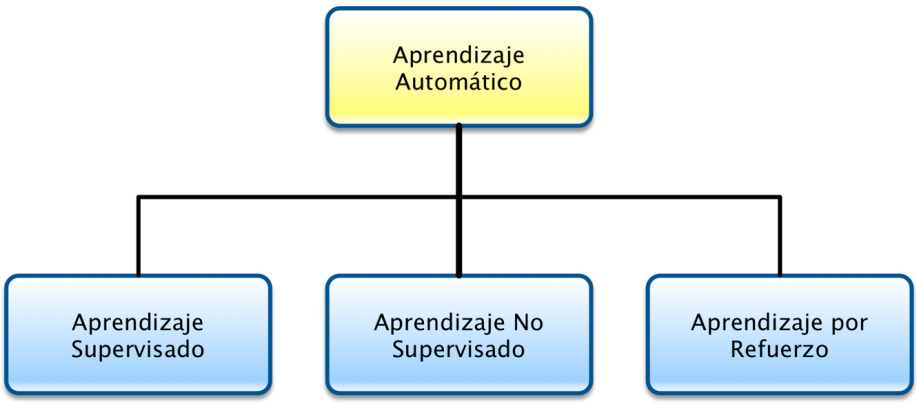
Aprendizaje por Refuerzo



El aprendizaje por refuerzo es aprender qué hacer, dada una situación y un conjunto de posibles acciones para elegir, con el fin de maximizar una recompensa. Al alumno, que llamaremos agente, no se le dice qué hacer, debe descubrirlo por sí mismo a través de la interacción con el ambienta. El objetivo es elegir sus acciones de tal manera que la recompensa acumulada se maximice. Entonces, elegir la mejor recompensa ahora, podría no ser la mejor decisión, a la larga.

- Una definición formal

> El aprendizaje por refuerzo es un marco para resolver tareas de control (también llamados problemas de decisión) mediante la creación de agentes que aprenden del ambiente interactuando con él a través de prueba y error y recibiendo recompensas (positivas o negativas) como su única retroalimentación.



El aprendizaje por refuerzo es diferente del aprendizaje supervisado, el tipo de aprendizaje estudiado en la mayoría de las aplicaciones en el campo del aprendizaje automático. El aprendizaje supervisado es aprender de un conjunto de entrenamiento de ejemplos etiquetados proporcionados por un supervisor externo. Cada ejemplo es una descripción de una situación junto con una especificación, la etiqueta, de la acción correcta que el sistema debería tomar en esa situación, que a menudo es identificar la categoría a la que pertenece la situación El objetivo de este tipo de aprendizaje es que el sistema extrapole o generalice sus respuestas para que actúe correctamente en situaciones que no están presentes en el conjunto de entrenamiento. Este es un tipo importante de aprendizaje, pero por si solo no es adecuado para aprender de la interacción. En problemas interactivos, a menudo no es práctico obtener ejemplos del comportamiento deseados que sean correctos y representativos de todas las situaciones en las que el agente tiene que actuar. En territorio inexplorado, donde uno esperaría que el aprendizaje sea más beneficioso, un agente debe ser capaz de aprender de su propia experiencia.

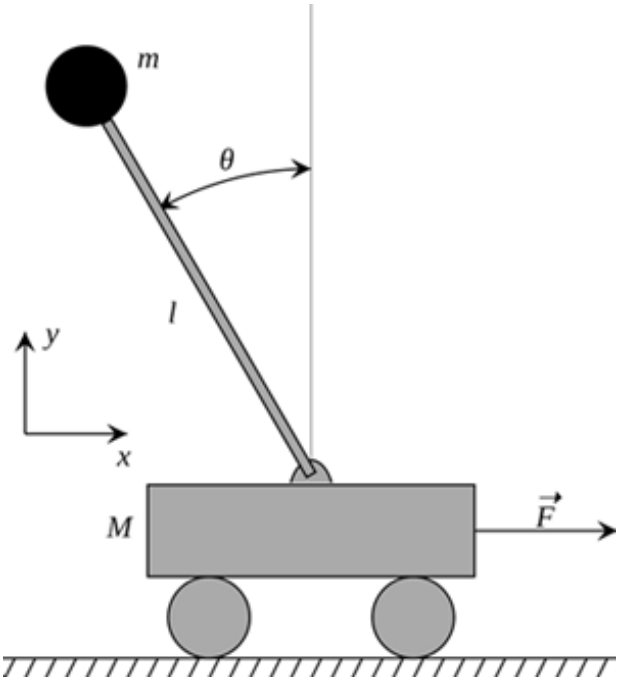
El aprendizaje por refuerzo también es diferente del aprendizaje no supervisado, que generalmente trata de encontrar la estructura oculta en colecciones de datos sin etiqueta. Los términos aprendizaje supervisado y aprendizaje no supervisado parecerían clasificar exhaustivamente los paradigmas de aprendizaje automático, pero no es así. Aunque uno podría estar tentado a pensar en el aprendizaje por refuerzo como una especie de aprendizaje no supervisado porque no se basa en ejemplos de comportamiento correcto, el aprendizaje por refuerzo está intentando maximizar una señal de recompensa en lugar de tratar de encontrar la estructura oculta. Descubriendo la estructura en la experiencia de un agente ciertamente puede ser útil en el aprendizaje por refuerzo, pero por sí mismo no aborda el problema de aprendizaje por resfuerzo de

maximizar la señal de recompensa.

Por lo tanto, se puede considerar que el aprendizaje por refuerzo es un tercer paradigma de aprendizaje automático, junto con el aprendizaje supervisado y el aprendizaje no supervisado.

Ejemplos de Aprendizaje por Refuerzo

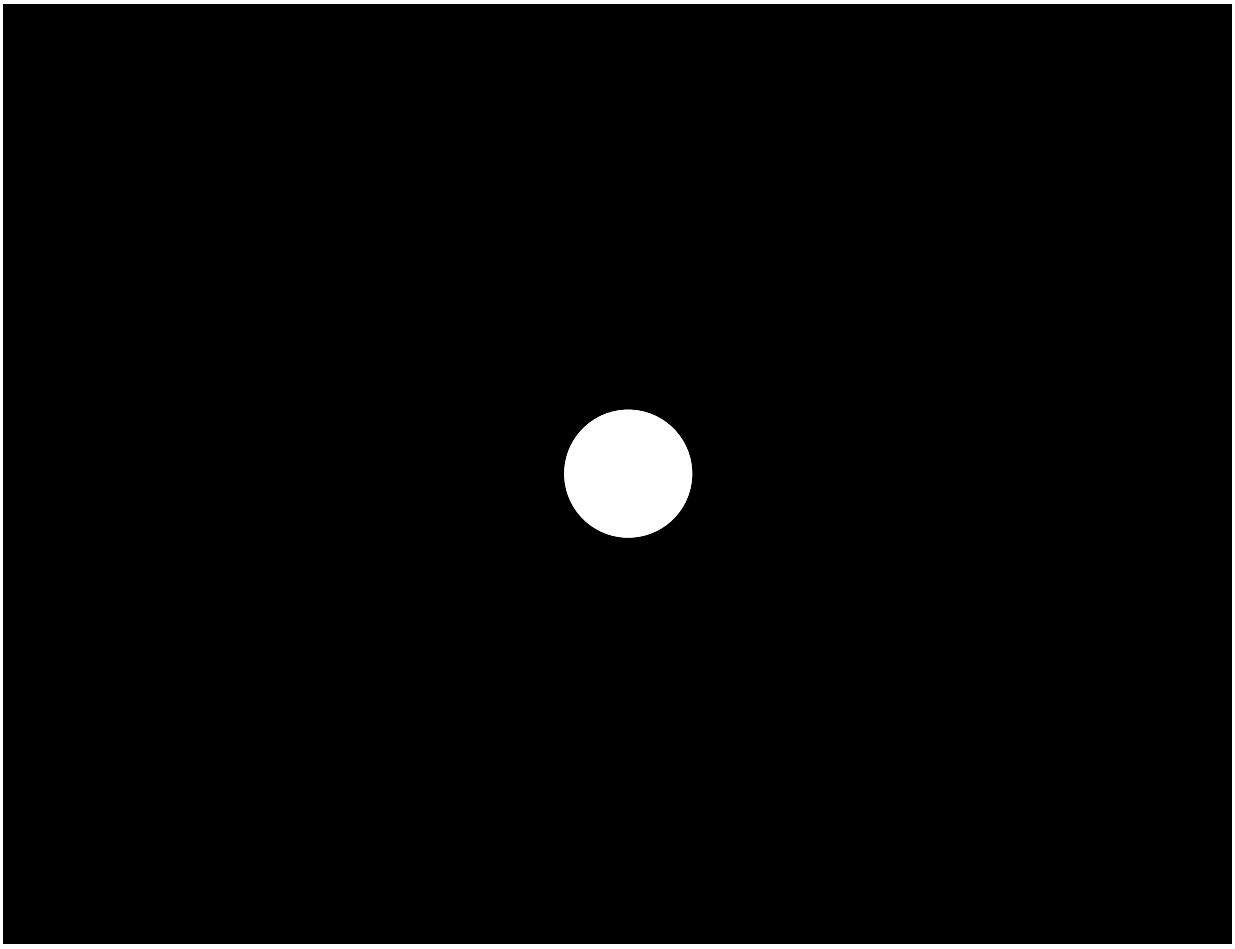
1. Balance Carro-Poste



- **Objetivo**: equilibrar el poste sobre un carro en movimiento
- **Estado**: ángulo, velocidad angular, posición, velocidad horizontal
- **Acciones**: fuerza horizontal al carro
- **Recompensa**: 1 en cada paso de tiempo si el poste está en posición vertical

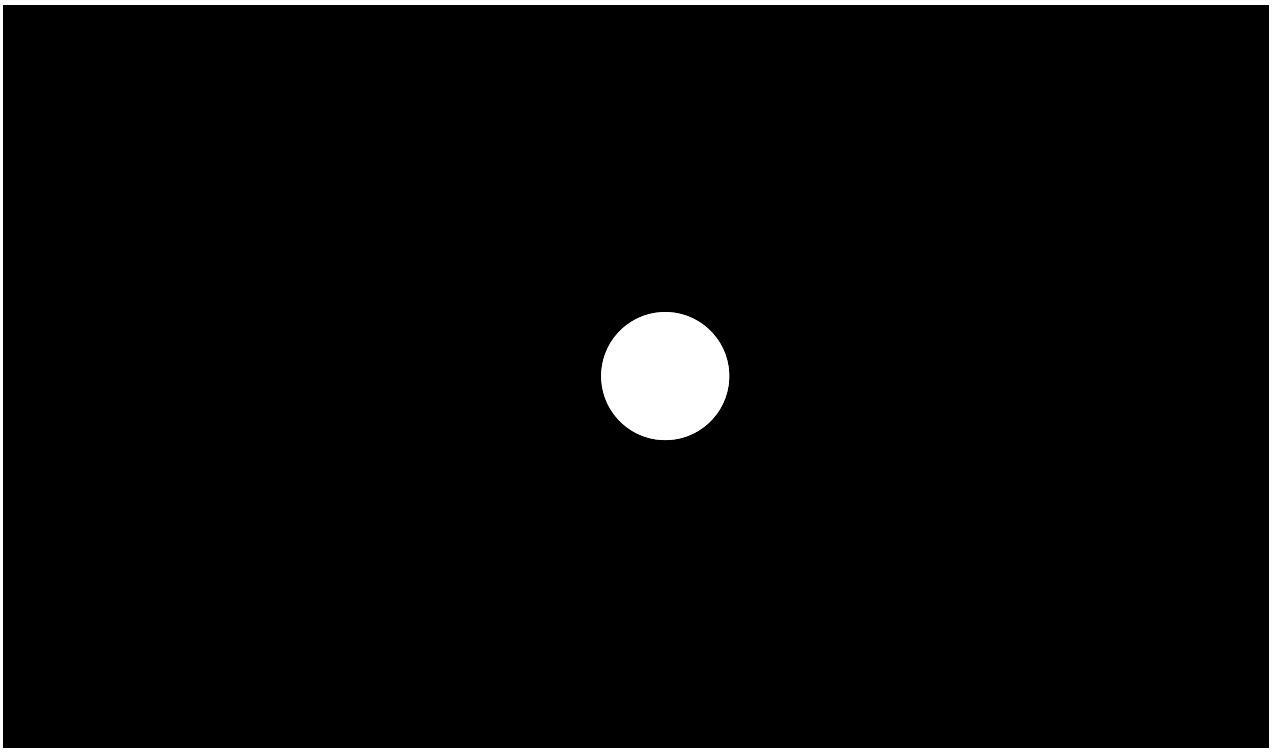
2. Juegos de Atari

Breakout



- **Objetivo**: Ganar el juego con el puntaje más alto
- **Estado**: Píxeles de la pantalla del juego
- **Acciones**: arriba, abajo, izquierda, derecha, etc.
- **Recompensa** - Puntuación proporcionada por el juego

Pac-Mac



https://github.com/tychovdo/PacmanDQN

- **Objetivo**: Ganar el juego con el puntaje más alto
- **Estado**: Píxeles de la pantalla del juego
- **Acciones**: arriba, abajo, izquierda, derecha, etc.
- **Recompensa** - Puntuación proporcionada por el juego

3. Entrenar Robots para el embalaje

- **Objetivo**: Elegir un dispositivo de una caja y ponerlo en un contenedor
- **Estado**: Píxeles brutos del mundo real
- **Acciones**: Posibles acciones del robot
- **Recompensa**: Positiva al colocar un dispositivo con éxito; de lo contrario, negativo

La hipótesis de la recompensa: la idea central del Aprendizaje por Refuerzo

¿Por qué el objetivo del agente es maximizar el rendimiento esperado?

Porque el Aprendizaje por Refuerzo se basa en la hipótesis de la recompensa, que es que todos los objetivos pueden describirse como la maximización del rendimiento esperado (recompensa acumulada esperada).

Es por eso que en el Aprendizaje por Refuerzo, para tener el mejor comportamiento, necesitamos maximizar la recompensa acumulada esperada.

Formalizando el problema del Aprendizaje por Refuerzo

El Proceso de Decisión de Markov (MDP por sus siglas en inglés) es una formulación matemática del problema del Aprendizaje por Refuerzo. Que satisfacen la propiedad de Markov:

Propiedad de Markov: el estado actual representa por completo el estado del ambiente (mundo). Es decir, el futuro depende solo del presente.

Un MDP puede definirse por (S, A, R, P, γ) donde:

- **S**: conjunto de posibles estados
- **A**: conjunto de posibles acciones
- **R**: distribución de la probabilidad de la recompensa dado el par (estado, acción)
- **P**: distribución de probabilidad de que tan posible que alguno de los estados sea el nuevo estado, dado el par (estado, acción). También conocido como probabilidad de transición.
- γ: factor de descuento de la recompensa

Espacio de Observaciones/Estados

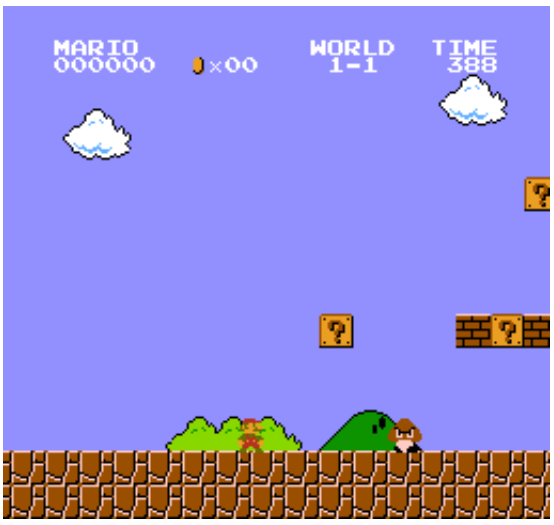
Las observaciones/estados son la información que nuestro agente obtiene del entorno. En el caso de un videojuego, puede ser un fotograma (una captura de pantalla). En el caso del agente comercial, puede ser el valor de una determinada acción, etc.

Hay que hacer una diferenciación entre observación y estado:

- Estado s: es una descripción completa del estado del mundo (no hay información oculta). En un entorno completamente observado.

En el juego de ajedrez, recibimos un estado del entorno ya que tenemos acceso a toda la información del tablero. Por esto en un juego de ajedrez, estamos en un entorno completamente observado.

- Observación o: es una descripción parcial del estado. En un entorno parcialmente observado.

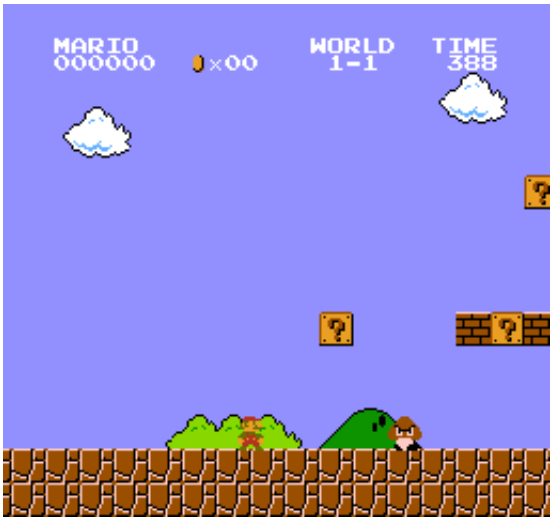


En Super Mario Bros, estamos en un entorno parcialmente observado, solo vemos una parte del nivel cerca del jugador, por lo que recibimos una observación.

Espacio de acción

El espacio de acción es el conjunto de todas las acciones posibles en un entorno. Las acciones pueden provenir de un espacio discreto o continuo:

- *Espacio discreto*: el número de acciones posibles es finito.



En Super Mario Bros, tenemos un conjunto finito de acciones ya que solo tenemos 4 direcciones y salto.

- *Espacio continuo*: el número de acciones posibles es infinito.

Un agente de Carro Autónomo tiene infinidad de acciones posibles ya que puede girar 20° a la izquierda, 21,1°, 21,2°, tocar la bocina, girar 20° a la derecha...

Recompensas y descuentos

La recompensa es fundamental en RL porque es la única retroalimentación para el agente. Gracias a ella, nuestro agente sabe si la acción realizada fue buena o no.

La recompensa acumulada en cada paso de tiempo t se puede escribir como:

$$R(\tau) = \sum_{t \geq 0}^{\infty} r_t$$

$$R_t = r_t + r_{t+1} + \cdots + r_n$$

Sin embargo, en realidad, no podemos simplemente agregarlos así. Las recompensas que llegan antes (al comienzo del juego) tienen más probabilidades de suceder, ya que son más predecibles que las recompensas futuras a largo plazo. Por lo que se descontaran las recompenzas a largo plazo

Para descontar las recompensas, procedemos así:

- Definimos una tasa de descuento llamada gamma (γ). Debe estar entre 0 y 1. La mayoría de las veces entre 0,99 y 0,95.
- Cuanto mayor sea la gamma, menor será el descuento. Esto significa que nuestro agente se preocupa más por la recompensa a largo plazo.
- Por otro lado, cuanto menor sea la gamma, mayor será el descuento. Esto significa que nuestro agente se preocupa más por la recompensa a corto plazo.
- Luego, cada recompensa será descontada por gamma al exponente del paso de tiempo, por lo que la recompensa futura es cada vez menos probable.

$$R(\tau) = \sum_{t \geq 0}^{\infty} \gamma^k r_t$$

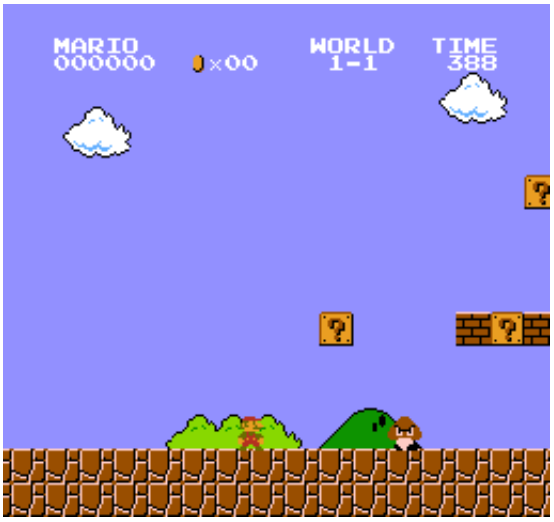
$$R_t = r_t + \gamma r_{t+1} + \cdots + \gamma^{n-t} r_n = r_t + \gamma R_{t+1}$$

Tipo de tareas

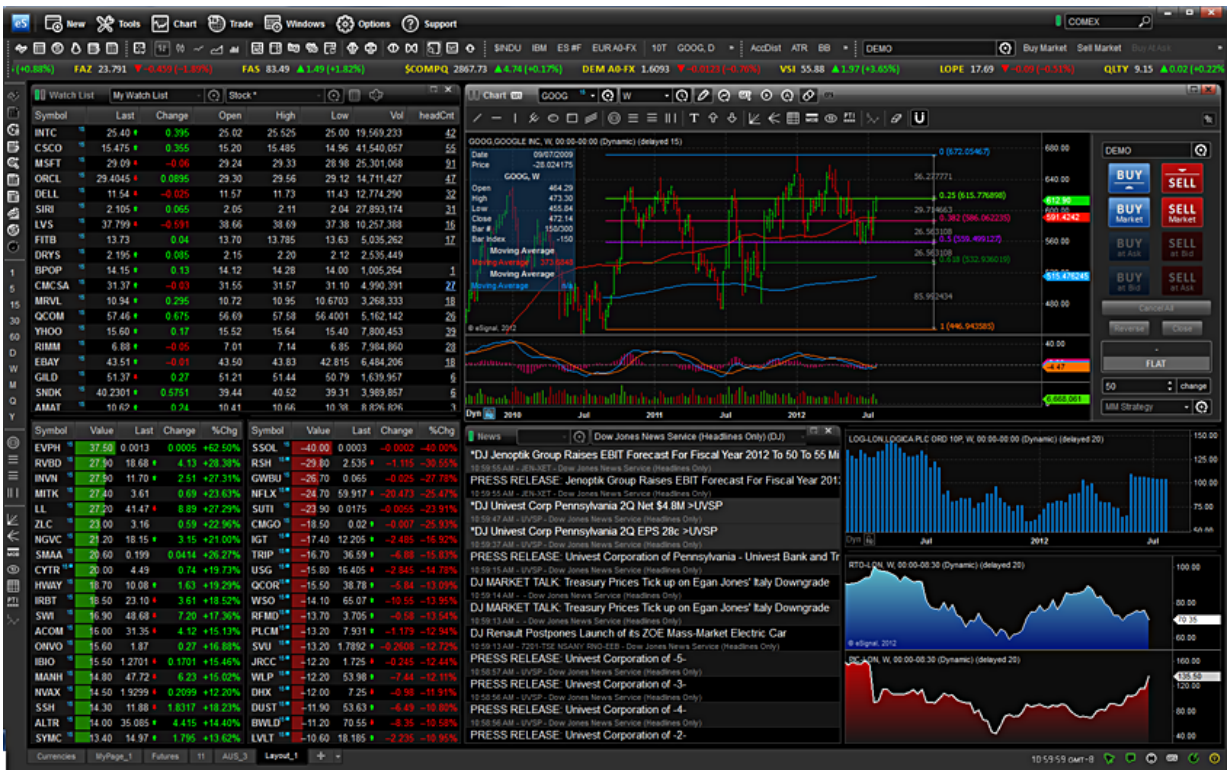
Una tarea es una instancia de un problema de aprendizaje por refuerzo. Podemos tener dos tipos de tareas: episódicas y continuas.

- *Tarea episódica*: En este caso, tenemos un punto de partida y un punto final (un estado terminal). Esto crea un episodio: una lista de Estados, Acciones, Recompensas y nuevos Estados. Por ejemplo, piensa en Super Mario Bros: un

episodio comienza con el lanzamiento de un nuevo nivel de Mario y termina cuando te matan o llegas al final del nivel.



- *Tareas continuas*: Estas son tareas que continúan para siempre (sin estado terminal). En este caso, el agente debe aprender a elegir las mejores acciones y simultáneamente interactuar con el entorno. Por ejemplo, un agente que realiza transacciones bursátiles automatizadas. Para esta tarea, no hay un punto de partida ni un estado terminal. El agente sigue corriendo hasta que decidimos detenerlo.



Compromiso entre exploración/explotación

Finalmente, antes de ver los diferentes métodos para resolver problemas de aprendizaje por refuerzo, debemos cubrir otro tema muy importante: la compensación de exploración/explotación.

- La exploración es explorar el entorno al intentar acciones aleatorias para encontrar más información sobre el entorno.
- La explotación es explotar información conocida para maximizar la recompensa.

Existen dos enfoques principales para resolver problemas de Aprendizaje por Refuerzo

Ahora que aprendimos el enfoque de Aprendizaje por Refuerzo, ¿cómo resolvemos el problema de Aprendizaje por Refuerzo?

En otros términos, ¿cómo construir un agente de Aprendizaje por Refuerzo que pueda seleccionar las acciones que maximicen su recompensa acumulada esperada?

La Política π: el cerebro del agente

La Política π es el cerebro de nuestro Agente, es la función que nos dice qué acción tomar dado el estado en el que nos encontramos. Por lo que define el comportamiento del agente en un momento dado.

Esta Política es la función que queremos aprender, nuestro objetivo es encontrar la política óptima π^*, la política que maximiza la rentabilidad esperada cuando el agente actúa de acuerdo con ella. Encontramos esta π^* a través del entrenamiento.

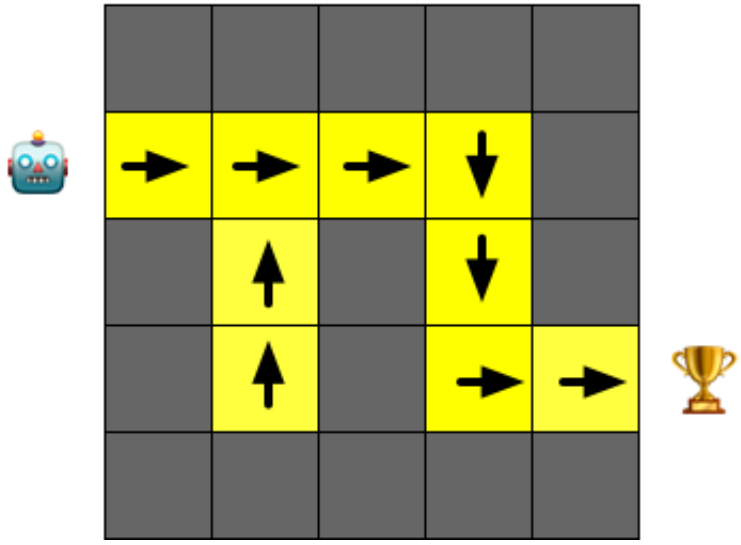
Hay dos enfoques para entrenar a nuestro agente para encontrar esta política óptima π^* :

- *Directamente*, enseñando al agente a aprender qué acción tomar, dado el estado en el que se encuentra: Métodos Basados en Políticas.
- *Indirectamente*, enseñe al agente a aprender qué estado es más valioso y luego tome la acción que lo lleve a los estados más valiosos: Métodos Basados en Valores.

Métodos Basados en Políticas.

En los métodos basados en políticas, aprendemos una función de política directamente.

Esta función mapeará desde cada estado a la mejor acción correspondiente en ese estado. O una distribución de probabilidad sobre el conjunto de acciones posibles en ese estado.



Tenemos dos tipos de políticas:

- *Determinista*: una política en un estado determinado siempre devolverá la misma acción.

$$a = \pi(s)$$

- Estocástica: genera una distibución de probabilidad sobre las acciones.

$$\pi(a|s) = P[A|s]$$

política(acciones | estado) = distribución de probabilidad sobre el conjunto de acciones dado el estado actual

Métodos basados en valores

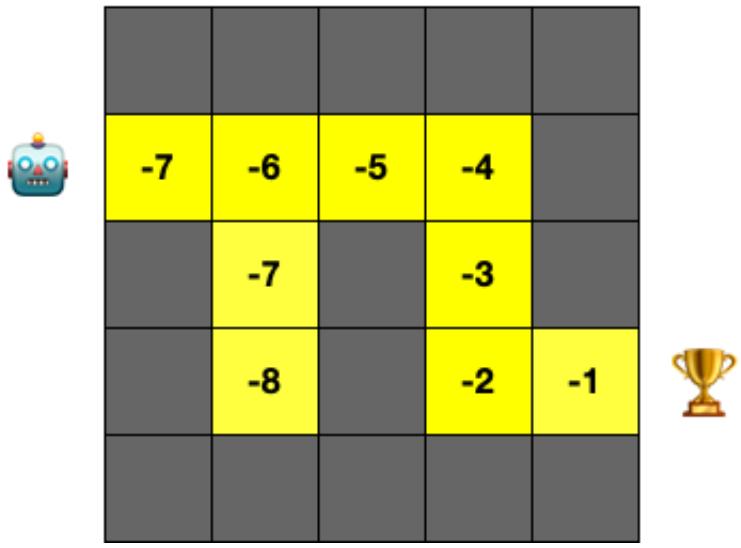
En los métodos basados en valores, en lugar de entrenar una función de política, entrenamos una función de valor que asigna un estado al valor esperado de estar en ese estado.

El valor de un estado es el rendimiento descontado esperado que el agente puede obtener si comienza en ese estado y luego actúa de acuerdo con nuestra política.

"Actuar de acuerdo con nuestra política" simplemente significa que nuestra política es "ir al estado con el valor más alto".

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + R_{t+2} + R_{t+3} + \ldots | S_t = s]$$

Aquí vemos que nuestra función de valor definió el valor para cada estado posible.



Gracias a nuestra función de valor, en cada paso nuestra política seleccionará el estado con el mayor valor definido por la función de valor: -7, luego -6, luego -5 (y así sucesivamente) hasta alcanzar la meta

Ejemplo de Aprendizaje por Refuerzo

En este ejemplo utilizaremos una librería para crear ambientes para el Aprendizaje por Refuerzo llamada Gym. Gym es una librería Python de código abierto para desarrollar y comparar algoritmos de aprendizaje por refuerzo al proporcionar una API estándar para comunicar algoritmos con entornos de aprendizaje, así como un conjunto estándar de entornos que cumplen con esa API.

In [9]:
```
import gym

from stable_baselines3 import PPO
from stable_baselines3.common.evaluation import evaluate_policy
from stable_baselines3.common.env_util import make_vec_env
```

La librería que contiene nuestro ambiente (entorno) se llama Gym.

La biblioteca de Gym proporciona dos cosas:

- Una interfaz que te permite crear ambientes de Aprendizaje por Refuerzo.
- Una colección de ambientes (gym-control, atari, box2D...).

Con Gym:

1. Creamos nuestro entorno usando `gym.make()`
2. Restablecemos el entorno a su estado inicial con `observacion = env.reset()`

En cada paso:

1. Obtén una acción usando nuestro modelo (en nuestro ejemplo tomamos una acción aleatoria)
2. Usando `env.step(action)`, realizamos esta acción en el entorno y obtenemos
    - `observación`: El nuevo estado (s_{t+1})
    - `recompenza`: La recompensa que obtenemos tras ejecutar la acción
    - `listo`: Indica si el episodio terminó
    - `info`: Un diccionario que proporciona información adicional (depende del ambiente).

Si el episodio ha terminado:

- Restablecemos el entorno a su estado inicial con `observation = env.reset()`

In [5]:
```
import gym

# Primero, creamos un ambiente(entorno) llamado LunarLander-v2
ambiente = gym.make("LunarLander-v2")

# Posteriormente reiniciamos este ambiente
observacion = ambiente.reset()

for _ in range(20):
  # Tomar una acción al azar
  accion = ambiente.action_space.sample()
  print("Acción tomada:", accion)

  # Ejecutar la acción en el ambiente y obtener
  # el próximo estado, rencompensa, listo e información adicional
  observacion, rencompensa, listo, info = ambiente.step(accion)

  # Si el juego finalizo (en nuestro caso, aterizamos, nos estrellamos o se
  if listo:
      # Resetear el ambiente
      print("Reiniciar el ambiente")
      observacion = ambiente.reset()
```

```
Acción tomada: 2
Acción tomada: 1
Acción tomada: 0
Acción tomada: 2
Acción tomada: 3
Acción tomada: 3
Acción tomada: 2
Acción tomada: 0
Acción tomada: 2
Acción tomada: 0
Acción tomada: 1
Acción tomada: 2
Acción tomada: 1
Acción tomada: 1
Acción tomada: 0
Acción tomada: 1
Acción tomada: 0
Acción tomada: 0
Acción tomada: 2
Acción tomada: 2
```

El ambiente

En este ejemplo vamos a entrenar a nuestro agente, un `Lunar Lander`, para aterrizar correctamente en la luna. Para hacer eso, el agente necesita aprender a adaptar su velocidad y posición (horizontal, vertical y angular) para aterrizar correctamente.

In [6]:
```
# Crear ambiente con gym.make("<nombre_del_ambiente>")
ambiente = gym.make("LunarLander-v2")
ambiente.reset()
print("_____ESPACIO DE OBSERVACIÓN_____ \n")
print("Forma del Espacio de Observación", ambiente.observation_space.shape)
print("Observación aleatoria", ambiente.observation_space.sample()) # Obtene
```

```
_____ESPACIO DE OBSERVACIÓN_____

Forma del Espacio de Observación (8,)
Observación aleatoria [-0.7987782   1.7987401   0.20531172  1.42423    -1.88
09506   1.0780168
  0.08760991  1.1825439 ]
```

Vemos con `Forma del Espacio de Observación (8,)` que la observación es un vector de tamaño 8, donde cada valor contiene información diferente sobre el módulo de aterrizaje:

- Coordenada horizontal de la plataforma (x)
- Coordenada verticalde plataforma (y)
- Velocidad horizontal (x)
- Velocidad vertical (y)
- Ángulo
- Velocidad angular
- Si la pierna izquierda tiene punto de contacto tocó la tierra
- Si la pierna derecha tiene punto de contacto tocó la tierra

In [7]:
```
print("\n _____ESPACIO_DE_ACCIÓN_____ \n")
print("Forma del Espacio de Acción", ambiente.action_space.n)
print("Acción aleatoria", ambiente.action_space.sample()) # Obtener una acci
```

```
 _____ESPACIO_DE_ACCIÓN_____

Forma del Espacio de Acción 4
Acción aleatoria 3
```

El espacio de acción (el conjunto de acciones posibles que puede realizar el agente) es discreto con 4 acciones disponibles:

- Hacer nada,
- Dispara motor de orientación izquierda,
- Dispara el motor principal,
- Dispara motor de orientación derecha.

Función de recompensa (la función que otorga una recompensa en cada paso de tiempo):

- Moverse desde la parte superior de la pantalla hasta la plataforma de aterrizaje y la velocidad cero es de aproximadamente 100 a 140 puntos.
- El motor principal de disparo es -0.3 cada cuadro
- Cada contacto con el suelo de la pierna es +10 puntos
- El episodio termina si el módulo de aterrizaje se estrella (adicional -100 puntos) o se detiene (+100 puntos)

Ambiente vectorizado

Creamos un ambiente vectorizado (método para apilar múltiples ambientes independientes en un solo ambiente) de 16 ambientes, de esta manera, tendremos experiencias más diversas durante el entrenamiento.

In [18]:
```
# Crear el ambiente(entorno)
ambiente = make_vec_env('LunarLander-v2', n_envs=16)
```

Crear el modelo

- Ahora que estudiamos nuestro ambiente y entendimos el problema: **poder aterrizar correctamente el módulo de aterrizaje lunar en la plataforma de aterrizaje controlando los motores de orientación izquierdo, derecho y principal**. Construyamos el algoritmo que vamos a usar para resolver este problema 🚀.
- Para hacerlo, vamos a utilizar la librería de Aprendizaje por Refuerzo, Stable Baselines3 (SB3).
- SB3 es un conjunto de **implementaciones confiables de algoritmos de aprendizaje por refuerzo en PyTorch**.

Para resolver este problema, vamos a utilizar SB3 **PPO**. PPO (también conocido como optimización de política proximal).

PPO es una combinación de:

- *Método de aprendizaje por refuerzo basado en valores*: aprender una función de acción-valor que nos indicará cuál es la **acción más valiosa a realizar dado un estado y una acción**.
- *Método de aprendizaje por refuerzo basado en políticas*: aprender una política que **nos dará una distribución de probabilidad sobre las acciones**.

Stable-Baselines3 es fácil de configurar:

1. Crear el ambiente (en nuestro caso se hizo arriba)
2. Definir el modelo que se quiere usar y crear una instancia de este modelo
3. Entrenar al agente con `modelo.learn` y defines el número de pasos de tiempo de entrenamiento

```
# Crear ambiente
ambiente = gym.make('LunarLander-v2')

# Instanciar el agente
modelo = PPO('MlpPolicy', env, verbose=1)

# Entrenar al agente
modelo.learn(total_timesteps=int(2e5))
```

In [19]:
```
# Crear modelo y seleccionar parámetros para accelerar el entrenamiento
modelo = PPO(
    policy = 'MlpPolicy',
    env = ambiente,
    n_steps = 1024,
    batch_size = 64,
    n_epochs = 4,
    gamma = 0.999,
    gae_lambda = 0.98,
    ent_coef = 0.01,
    verbose=1)
```

```
Using cpu device
```

Entrenar al modelo (agente) de PPO 🏃

Entrenaremos a nuestro modelo para 500 000 intervalos de tiempo, no olvide usar GPU en Colab. Tomará aproximadamente ~10 minutos, pero puede usar menos intervalos de tiempo si solo desea probarlo.

In [20]:
```
# Entrenar por 500,000 pasos de tiempo
modelo.learn(total_timesteps=500000)
# Save the model
nombre_modelo = "../modelos/ppo-LunarLander-v2"
modelo.save(nombre_modelo)
```

```
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 90.8     |
|    ep_rew_mean     | -167     |
| time/              |          |
|    fps             | 11245    |
|    iterations      | 1        |
|    time_elapsed    | 1        |
|    total_timesteps | 16384    |
---------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 87.2        |
|    ep_rew_mean          | -138        |
| time/                   |             |
|    fps                  | 7661        |
|    iterations           | 2           |
|    time_elapsed         | 4           |
|    total_timesteps      | 32768       |
| train/                  |             |
|    approx_kl            | 0.007708028 |
|    clip_fraction        | 0.0314      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.38       |
|    explained_variance   | 0.000503    |
```

```
|    learning_rate        | 0.0003      |
|    loss                 | 886         |
|    n_updates            | 4           |
|    policy_gradient_loss | -0.0046     |
|    value_loss           | 4.6e+03     |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 101         |
|    ep_rew_mean          | -126        |
| time/                   |             |
|    fps                  | 6482        |
|    iterations           | 3           |
|    time_elapsed         | 7           |
|    total_timesteps      | 49152       |
| train/                  |             |
|    approx_kl            | 0.009654526 |
|    clip_fraction        | 0.0326      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.37       |
|    explained_variance   | -0.00827    |
|    learning_rate        | 0.0003      |
|    loss                 | 725         |
|    n_updates            | 8           |
|    policy_gradient_loss | -0.00432    |
|    value_loss           | 2.11e+03    |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 94.5        |
|    ep_rew_mean          | -118        |
| time/                   |             |
|    fps                  | 6147        |
|    iterations           | 4           |
|    time_elapsed         | 10          |
|    total_timesteps      | 65536       |
| train/                  |             |
|    approx_kl            | 0.0071801716 |
|    clip_fraction        | 0.0583      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.36       |
|    explained_variance   | -0.000293   |
|    learning_rate        | 0.0003      |
|    loss                 | 465         |
|    n_updates            | 12          |
|    policy_gradient_loss | -0.00623    |
|    value_loss           | 1.26e+03    |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 98.4        |
|    ep_rew_mean          | -105        |
| time/                   |             |
```

```
|    fps                  | 5933        |
|    iterations           | 5           |
|    time_elapsed         | 13          |
|    total_timesteps      | 81920       |
| train/                  |             |
|    approx_kl            | 0.009204393 |
|    clip_fraction        | 0.109       |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.35       |
|    explained_variance   | -0.0143     |
|    learning_rate        | 0.0003      |
|    loss                 | 227         |
|    n_updates            | 16          |
|    policy_gradient_loss | -0.0057     |
|    value_loss           | 646         |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 109         |
|    ep_rew_mean          | -85.5       |
| time/                   |             |
|    fps                  | 5574        |
|    iterations           | 6           |
|    time_elapsed         | 17          |
|    total_timesteps      | 98304       |
| train/                  |             |
|    approx_kl            | 0.010268188 |
|    clip_fraction        | 0.0989      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.32       |
|    explained_variance   | 0.00145     |
|    learning_rate        | 0.0003      |
|    loss                 | 229         |
|    n_updates            | 20          |
|    policy_gradient_loss | -0.00529    |
|    value_loss           | 731         |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 100         |
|    ep_rew_mean          | -73.3       |
| time/                   |             |
|    fps                  | 5493        |
|    iterations           | 7           |
|    time_elapsed         | 20          |
|    total_timesteps      | 114688      |
| train/                  |             |
|    approx_kl            | 0.008488523 |
|    clip_fraction        | 0.103       |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.31       |
|    explained_variance   | -0.0169     |
|    learning_rate        | 0.0003      |
```

```
|    loss                 | 209          |
|    n_updates            | 24           |
|    policy_gradient_loss | -0.00671     |
|    value_loss           | 493          |
-----------------------------------------
-----------------------------------------
| rollout/                |              |
|    ep_len_mean          | 107          |
|    ep_rew_mean          | -63.4        |
| time/                   |              |
|    fps                  | 5386         |
|    iterations           | 8            |
|    time_elapsed         | 24           |
|    total_timesteps      | 131072       |
| train/                  |              |
|    approx_kl            | 0.0065523633 |
|    clip_fraction        | 0.0366       |
|    clip_range           | 0.2          |
|    entropy_loss         | -1.3         |
|    explained_variance   | -0.00198     |
|    learning_rate        | 0.0003       |
|    loss                 | 130          |
|    n_updates            | 28           |
|    policy_gradient_loss | -0.00252     |
|    value_loss           | 506          |
-----------------------------------------
-----------------------------------------
| rollout/                |              |
|    ep_len_mean          | 108          |
|    ep_rew_mean          | -52.7        |
| time/                   |              |
|    fps                  | 5361         |
|    iterations           | 9            |
|    time_elapsed         | 27           |
|    total_timesteps      | 147456       |
| train/                  |              |
|    approx_kl            | 0.008149089  |
|    clip_fraction        | 0.0795       |
|    clip_range           | 0.2          |
|    entropy_loss         | -1.28        |
|    explained_variance   | 0.000569     |
|    learning_rate        | 0.0003       |
|    loss                 | 125          |
|    n_updates            | 32           |
|    policy_gradient_loss | -0.00597     |
|    value_loss           | 327          |
-----------------------------------------
-----------------------------------------
| rollout/                |              |
|    ep_len_mean          | 116          |
|    ep_rew_mean          | -42.5        |
| time/                   |              |
|    fps                  | 5319         |
```

```
|    iterations           | 10          |
|    time_elapsed         | 30          |
|    total_timesteps      | 163840      |
| train/                  |             |
|    approx_kl            | 0.008190183 |
|    clip_fraction        | 0.0563      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.26       |
|    explained_variance   | -0.0894     |
|    learning_rate        | 0.0003      |
|    loss                 | 255         |
|    n_updates            | 36          |
|    policy_gradient_loss | -0.00278    |
|    value_loss           | 488         |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 146         |
|    ep_rew_mean          | -34         |
| time/                   |             |
|    fps                  | 5143        |
|    iterations           | 11          |
|    time_elapsed         | 35          |
|    total_timesteps      | 180224      |
| train/                  |             |
|    approx_kl            | 0.009829336 |
|    clip_fraction        | 0.0422      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.21       |
|    explained_variance   | 0.000403    |
|    learning_rate        | 0.0003      |
|    loss                 | 254         |
|    n_updates            | 40          |
|    policy_gradient_loss | -0.00412    |
|    value_loss           | 463         |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 155         |
|    ep_rew_mean          | -26.6       |
| time/                   |             |
|    fps                  | 4813        |
|    iterations           | 12          |
|    time_elapsed         | 40          |
|    total_timesteps      | 196608      |
| train/                  |             |
|    approx_kl            | 0.006112218 |
|    clip_fraction        | 0.0353      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.19       |
|    explained_variance   | 0.000252    |
|    learning_rate        | 0.0003      |
|    loss                 | 211         |
```

```
|    n_updates            | 44          |
|    policy_gradient_loss | -0.00261    |
|    value_loss           | 568         |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 211         |
|    ep_rew_mean          | -18.9       |
| time/                   |             |
|    fps                  | 4503        |
|    iterations           | 13          |
|    time_elapsed         | 47          |
|    total_timesteps      | 212992      |
| train/                  |             |
|    approx_kl            | 0.008938706 |
|    clip_fraction        | 0.0337      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.16       |
|    explained_variance   | -0.00136    |
|    learning_rate        | 0.0003      |
|    loss                 | 253         |
|    n_updates            | 48          |
|    policy_gradient_loss | -0.00256    |
|    value_loss           | 551         |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 269         |
|    ep_rew_mean          | -12.9       |
| time/                   |             |
|    fps                  | 3918        |
|    iterations           | 14          |
|    time_elapsed         | 58          |
|    total_timesteps      | 229376      |
| train/                  |             |
|    approx_kl            | 0.0070185848 |
|    clip_fraction        | 0.0458      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.17       |
|    explained_variance   | -0.00658    |
|    learning_rate        | 0.0003      |
|    loss                 | 321         |
|    n_updates            | 52          |
|    policy_gradient_loss | -0.00299    |
|    value_loss           | 604         |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 333         |
|    ep_rew_mean          | -0.0544     |
| time/                   |             |
|    fps                  | 3592        |
|    iterations           | 15          |
```

```
|    time_elapsed         | 68           |
|    total_timesteps      | 245760       |
| train/                  |              |
|    approx_kl            | 0.0054769106 |
|    clip_fraction        | 0.0387       |
|    clip_range           | 0.2          |
|    entropy_loss         | -1.23        |
|    explained_variance   | -0.033       |
|    learning_rate        | 0.0003       |
|    loss                 | 185          |
|    n_updates            | 56           |
|    policy_gradient_loss | -0.00226     |
|    value_loss           | 433          |
-----------------------------------------
-----------------------------------------
| rollout/                |              |
|    ep_len_mean          | 377          |
|    ep_rew_mean          | 7.9          |
| time/                   |              |
|    fps                  | 3301         |
|    iterations           | 16           |
|    time_elapsed         | 79           |
|    total_timesteps      | 262144       |
| train/                  |              |
|    approx_kl            | 0.0051241606 |
|    clip_fraction        | 0.0282       |
|    clip_range           | 0.2          |
|    entropy_loss         | -1.17        |
|    explained_variance   | 0.186        |
|    learning_rate        | 0.0003       |
|    loss                 | 153          |
|    n_updates            | 60           |
|    policy_gradient_loss | -0.00218     |
|    value_loss           | 363          |
-----------------------------------------
-----------------------------------------
| rollout/                |              |
|    ep_len_mean          | 434          |
|    ep_rew_mean          | 18.4         |
| time/                   |              |
|    fps                  | 3066         |
|    iterations           | 17           |
|    time_elapsed         | 90           |
|    total_timesteps      | 278528       |
| train/                  |              |
|    approx_kl            | 0.003580145  |
|    clip_fraction        | 0.0137       |
|    clip_range           | 0.2          |
|    entropy_loss         | -1.21        |
|    explained_variance   | 0.427        |
|    learning_rate        | 0.0003       |
|    loss                 | 140          |
|    n_updates            | 64           |
```

```
|    policy_gradient_loss | -0.00121    |
|    value_loss           | 305         |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 520         |
|    ep_rew_mean          | 25.2        |
| time/                   |             |
|    fps                  | 2854        |
|    iterations           | 18          |
|    time_elapsed         | 103         |
|    total_timesteps      | 294912      |
| train/                  |             |
|    approx_kl            | 0.0066190967 |
|    clip_fraction        | 0.0292      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.21       |
|    explained_variance   | 0.577       |
|    learning_rate        | 0.0003      |
|    loss                 | 107         |
|    n_updates            | 68          |
|    policy_gradient_loss | -0.00312    |
|    value_loss           | 274         |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 603         |
|    ep_rew_mean          | 32.6        |
| time/                   |             |
|    fps                  | 2673        |
|    iterations           | 19          |
|    time_elapsed         | 116         |
|    total_timesteps      | 311296      |
| train/                  |             |
|    approx_kl            | 0.005822403 |
|    clip_fraction        | 0.0309      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.2        |
|    explained_variance   | 0.737       |
|    learning_rate        | 0.0003      |
|    loss                 | 57.6        |
|    n_updates            | 72          |
|    policy_gradient_loss | -0.000512   |
|    value_loss           | 147         |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 639         |
|    ep_rew_mean          | 41.9        |
| time/                   |             |
|    fps                  | 2578        |
|    iterations           | 20          |
|    time_elapsed         | 127         |
```

```
|    total_timesteps      | 327680       |
| train/                  |              |
|    approx_kl            | 0.0039718943 |
|    clip_fraction        | 0.039        |
|    clip_range           | 0.2          |
|    entropy_loss         | -1.18        |
|    explained_variance   | 0.795        |
|    learning_rate        | 0.0003       |
|    loss                 | 96.4         |
|    n_updates            | 76           |
|    policy_gradient_loss | -0.00123     |
|    value_loss           | 146          |
-----------------------------------------
-----------------------------------------
| rollout/                |              |
|    ep_len_mean          | 687          |
|    ep_rew_mean          | 47.7         |
| time/                   |              |
|    fps                  | 2468         |
|    iterations           | 21           |
|    time_elapsed         | 139          |
|    total_timesteps      | 344064       |
| train/                  |              |
|    approx_kl            | 0.0035180554 |
|    clip_fraction        | 0.0255       |
|    clip_range           | 0.2          |
|    entropy_loss         | -1.15        |
|    explained_variance   | 0.804        |
|    learning_rate        | 0.0003       |
|    loss                 | 174          |
|    n_updates            | 80           |
|    policy_gradient_loss | -0.00198     |
|    value_loss           | 195          |
-----------------------------------------
-----------------------------------------
| rollout/                |              |
|    ep_len_mean          | 680          |
|    ep_rew_mean          | 53           |
| time/                   |              |
|    fps                  | 2375         |
|    iterations           | 22           |
|    time_elapsed         | 151          |
|    total_timesteps      | 360448       |
| train/                  |              |
|    approx_kl            | 0.0065165358 |
|    clip_fraction        | 0.0735       |
|    clip_range           | 0.2          |
|    entropy_loss         | -1.17        |
|    explained_variance   | 0.893        |
|    learning_rate        | 0.0003       |
|    loss                 | 47.3         |
|    n_updates            | 84           |
|    policy_gradient_loss | -0.0025      |
```

```
|    value_loss           | 87.9         |
-----------------------------------------
-----------------------------------------
| rollout/                |              |
|    ep_len_mean          | 722          |
|    ep_rew_mean          | 57.4         |
| time/                   |              |
|    fps                  | 2286         |
|    iterations           | 23           |
|    time_elapsed         | 164          |
|    total_timesteps      | 376832       |
| train/                  |              |
|    approx_kl            | 0.0054524504 |
|    clip_fraction        | 0.0419       |
|    clip_range           | 0.2          |
|    entropy_loss         | -1.16        |
|    explained_variance   | 0.868        |
|    learning_rate        | 0.0003       |
|    loss                 | 108          |
|    n_updates            | 88           |
|    policy_gradient_loss | -0.00245     |
|    value_loss           | 125          |
-----------------------------------------
-----------------------------------------
| rollout/                |              |
|    ep_len_mean          | 739          |
|    ep_rew_mean          | 70.6         |
| time/                   |              |
|    fps                  | 2220         |
|    iterations           | 24           |
|    time_elapsed         | 177          |
|    total_timesteps      | 393216       |
| train/                  |              |
|    approx_kl            | 0.0056241686 |
|    clip_fraction        | 0.0291       |
|    clip_range           | 0.2          |
|    entropy_loss         | -1.17        |
|    explained_variance   | 0.919        |
|    learning_rate        | 0.0003       |
|    loss                 | 24           |
|    n_updates            | 92           |
|    policy_gradient_loss | -0.00127     |
|    value_loss           | 67.8         |
-----------------------------------------
---------------------------------------
| rollout/                |           |
|    ep_len_mean          | 803       |
|    ep_rew_mean          | 79.9      |
| time/                   |           |
|    fps                  | 2161      |
|    iterations           | 25        |
|    time_elapsed         | 189       |
|    total_timesteps      | 409600    |
```

```
| train/                  |           |
|    approx_kl            | 0.0034053 |
|    clip_fraction        | 0.0277    |
|    clip_range           | 0.2       |
|    entropy_loss         | -1.14     |
|    explained_variance   | 0.934     |
|    learning_rate        | 0.0003    |
|    loss                 | 62        |
|    n_updates            | 96        |
|    policy_gradient_loss | -0.0015   |
|    value_loss           | 65.9      |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 811         |
|    ep_rew_mean          | 82.9        |
| time/                   |             |
|    fps                  | 2116        |
|    iterations           | 26          |
|    time_elapsed         | 201         |
|    total_timesteps      | 425984      |
| train/                  |             |
|    approx_kl            | 0.0047242157 |
|    clip_fraction        | 0.0583      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.14       |
|    explained_variance   | 0.95        |
|    learning_rate        | 0.0003      |
|    loss                 | 44.8        |
|    n_updates            | 100         |
|    policy_gradient_loss | -0.00161    |
|    value_loss           | 41          |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 826         |
|    ep_rew_mean          | 86.5        |
| time/                   |             |
|    fps                  | 2074        |
|    iterations           | 27          |
|    time_elapsed         | 213         |
|    total_timesteps      | 442368      |
| train/                  |             |
|    approx_kl            | 0.0039413394 |
|    clip_fraction        | 0.0298      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.12       |
|    explained_variance   | 0.919       |
|    learning_rate        | 0.0003      |
|    loss                 | 27.3        |
|    n_updates            | 104         |
|    policy_gradient_loss | -0.000714   |
|    value_loss           | 74.6        |
```

```
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 784         |
|    ep_rew_mean          | 86.8        |
| time/                   |             |
|    fps                  | 2045        |
|    iterations           | 28          |
|    time_elapsed         | 224         |
|    total_timesteps      | 458752      |
| train/                  |             |
|    approx_kl            | 0.0059618973 |
|    clip_fraction        | 0.0404      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.11       |
|    explained_variance   | 0.94        |
|    learning_rate        | 0.0003      |
|    loss                 | 11.7        |
|    n_updates            | 108         |
|    policy_gradient_loss | -0.0011     |
|    value_loss           | 63.6        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 810         |
|    ep_rew_mean          | 90.2        |
| time/                   |             |
|    fps                  | 2010        |
|    iterations           | 29          |
|    time_elapsed         | 236         |
|    total_timesteps      | 475136      |
| train/                  |             |
|    approx_kl            | 0.0040800227 |
|    clip_fraction        | 0.0181      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.06       |
|    explained_variance   | 0.93        |
|    learning_rate        | 0.0003      |
|    loss                 | 42.1        |
|    n_updates            | 112         |
|    policy_gradient_loss | -0.00181    |
|    value_loss           | 94.8        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 805         |
|    ep_rew_mean          | 92.9        |
| time/                   |             |
|    fps                  | 1982        |
|    iterations           | 30          |
|    time_elapsed         | 247         |
|    total_timesteps      | 491520      |
| train/                  |             |
```

```
|    approx_kl             | 0.0035253982 |
|    clip_fraction         | 0.0207       |
|    clip_range            | 0.2          |
|    entropy_loss          | -1.06        |
|    explained_variance    | 0.943        |
|    learning_rate         | 0.0003       |
|    loss                  | 19.9         |
|    n_updates             | 116          |
|    policy_gradient_loss  | -0.000987    |
|    value_loss            | 59.1         |
------------------------------------------
------------------------------------------
| rollout/                 |              |
|    ep_len_mean           | 828          |
|    ep_rew_mean           | 101          |
| time/                    |              |
|    fps                   | 1952         |
|    iterations            | 31           |
|    time_elapsed          | 260          |
|    total_timesteps       | 507904       |
| train/                   |              |
|    approx_kl             | 0.0061352625 |
|    clip_fraction         | 0.0388       |
|    clip_range            | 0.2          |
|    entropy_loss          | -1.07        |
|    explained_variance    | 0.939        |
|    learning_rate         | 0.0003       |
|    loss                  | 20.8         |
|    n_updates             | 120          |
|    policy_gradient_loss  | -0.0012      |
|    value_loss            | 64.1         |
------------------------------------------
```

Evaluar al agente

- Ahora que nuestro agente Lunar Lander está entrenado, debemos **verificar su rendimiento**.
- Stable-Baselines3 proporciona un método para hacerlo: `evaluate_policy`.
- En el siguiente paso, veremos **cómo evaluar y compartir automáticamente a su agente para competir en una tabla de clasificación, pero por ahora hagámoslo nosotros mismos**

Cuando evalúes a tu agente, no debes usar tu entorno de entrenamiento sino crear un entorno de evaluación.

In [21]:
```
#Evaluar el modelo
ambiente_evaluacion = gym.make("LunarLander-v2")
promedio_recompensa, desviacion_estandar_recompensa = evaluate_policy(modelo
print(f"promedio recompensa = {promedio_recompensa:.2f} +/- {desviacion_esta
```

```
promedio recompensa = 213.25 +/- 21.952831986064314
```

In [ ]: