UIMP
Universidad Internacional Menéndez Pelayo

AEPIA
ASOCIACIÓN ESPAÑOLA PARA LA INTELIGENCIA ARTIFICIAL

Práctica 2: Aprendizaje por Refuerzo Profundo

Máster Universitario en Investigación en Inteligencia Artificial

Aprendizaje por Refuerzo

Javier Vela Tambo
100012966@alumnos.uimp.es

28 de mayo de 2025

Introducción

En esta práctica se resuelven tres ejercicios de Aprendizaje por Refuerzo Profundo (ARP) utilizando los entornos de Gymnasium [1] y los algoritmos de ARP de la librería Stable Baselines 3 (SB3) [2]. El objetivo es explorar diferentes entornos de Gymnasium y utilizar diferentes algoritmos para resolverlos. El primer ejercicio consiste en implementar un agente para resolver el entorno de CartPole (Sección 1). El segundo ejercicio implica entrenar un agente para resolver el entorno LunarLander (Sección 2) utilizando tres algoritmos diferentes de SB3. Por último, en el tercer ejercicio se entrena un agente para jugar al juego Breakout de Atari (Sección 3).

1. Optimización de CartPole

El objetivo de este ejercicio es desarrollar un agente capaz de mantener el péndulo en equilibrio durante los 500 pasos máximos del entorno `CartPole-v1` de Gymnasium [3]. Este entorno consiste en un péndulo invertido que debe ser equilibrado en posición vertical. El agente recibe una recompensa de +1 por cada paso que el péndulo se mantenga en equilibrio, y el episodio termina cuando el péndulo cae o se alcanza el límite de pasos.

Para evaluar el rendimiento del agente durante el entrenamiento, se ha implementado un callback utilizando las clases `StopTrainingOnRewardThreshold` y `EvalCallback` de la librería SB3. Este callback permite evaluar periódicamente el desempeño del agente y detener el proceso una vez que el agente alcanza el objetivo de maximizar la recompensa.

Inicialmente, se ha utilizado un agente basado en el algoritmo Deep Q-Networks (DQN) siguiendo el tutorial de la práctica. Aunque se han observado mejoras en el rendimiento mediante el ajuste de hiperparámetros, el agente no ha logrado de manera efectiva mantener el péndulo en equilibrio durante los 500 pasos.

[1] `https://gymnasium.farama.org/`
[2] `https://stable-baselines3.readthedocs.io`
[3] `https://gymnasium.farama.org/environments/classic_control/cart_pole/`

La Figura 1 muestra la recompensa media obtenida por el agente DQN durante el entrenamiento y las evaluaciones periódicas. Durante los 1.000.000 pasos de entrenamiento, el agente no logra alcanzar de forma consistente el objetivo deseado. Aunque en algunos momentos parece acercarse al rendimiento esperado, su desempeño varía mucho, lo que muestra que no ha aprendido una estrategia estable y efectiva para resolver el entorno. Esto podría deberse a varios factores, como una exploración limitada o hiperparámetros que no están bien ajustados para este problema.

Para superar esta limitación, se ha empleado el algoritmo Proximal Policy Optimization (PPO) de SB3. A diferencia de DQN, PPO logra un agente capaz de equilibrar el péndulo en tan solo 25.000 pasos. En este caso, se han utilizado los hiperparámetros por defecto de PPO en SB3, que incluyen un *learning rate* de 0.0003, un *batch size* de 64, un *gamma* de 0.99 y 2048 pasos por actualización. En PPO, tanto la red de la política (actor) como la de valor (crítico) utilizan dos capas ocultas de tamaño 64.

La Figura 2 muestra la recompensa media obtenida por el agente PPO durante el entrenamiento y evaluación. Se observa una mejora constante en el rendimiento, alcanzando un agente que puede mantener el péndulo en equilibrio durante los 500 pasos. Aunque la recompensa media no alcanza el máximo de 500 durante el entrenamiento debido a la exploración del entorno, en el modo de evaluación determinista el agente toma las mejores acciones posibles, logrando así el objetivo.

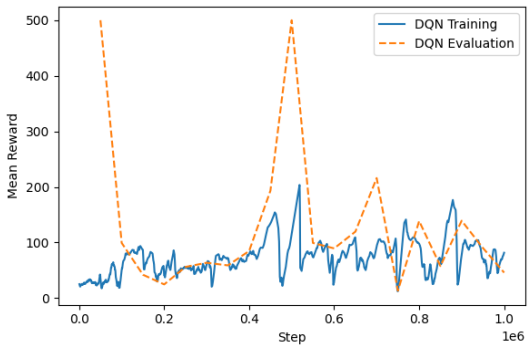

Figura 1: Recompensa media del agente DQN en CartPole.

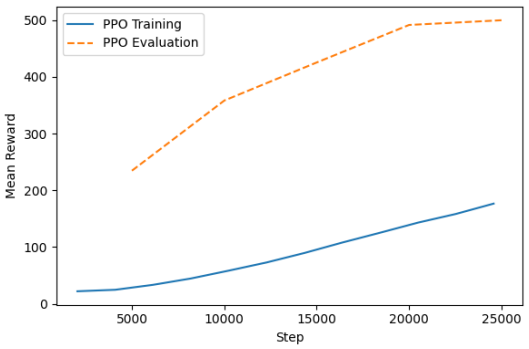

Figura 2: Recompensa media del agente PPO en CartPole.

2. Entrenamiento de LunarLander

El objetivo de este ejercicio es entrenar un agente para resolver un entorno *Box2D* de Gymnasium utilizando 3 algoritmos proporcionados por SB3. Entre los tres entornos disponibles (Bipedal Walker, Car Racing y Lunar Lander) se ha elegido el entorno `LunarLander-v3` [4].

2.1. Descripción del Entorno

El entorno `LunarLander-v3` simula el aterrizaje controlado de un cohete lunar en una plataforma situada en las coordenadas (0,0). El agente dispone de cuatro acciones discretas: no hacer nada, encender el motor principal, o encender los motores laterales izquierdo o derecho. El estado del entorno es un vector de 8 dimensiones que incluye la posición y velocidad del cohete, su ángulo y velocidad angular, y dos indicadores booleanos que señalan si cada pata está en contacto con el suelo.

La recompensa en cada paso depende de la proximidad y velocidad respecto a la plataforma, la inclinación del cohete, y el contacto de las patas con el suelo. Se penaliza el uso de los motores (principal y laterales) y se otorgan recompensas adicionales por aterrizajes exitosos (+100) o penalizaciones por accidentes (-100). Un episodio se considera resuelto si se alcanzan al menos 200 puntos de recompensa total. El episodio termina si el cohete se estrella, sale de la pantalla o queda inactivo.

2.2. Descripción de los Algoritmos Seleccionados

Para resolver el entorno `LunarLander-v3`, se han seleccionado tres de los algoritmos más populares de SB3: DQN, PPO y A2C. Estos algoritmos son adecuados para entornos con espacios de acción discretos y continuos, y han demostrado ser efectivos en una amplia variedad de tareas de ARP.

- *Deep Q-Networks* (DQN) es un algoritmo basado en Q-learning que utiliza redes neuronales profundas para aproximar la función de valor de acción. DQN es adecuado para entornos con espacios de acción discretos.
- *Proximal Policy Optimization* (PPO) es un algoritmo de optimización de políticas que utiliza una función de pérdida para actualizar la política del agente. PPO es conocido por su estabilidad y eficiencia en el entrenamiento, y es adecuado tanto para espacios de acción discretos como continuos.
- *Advantage Actor-Critic* (A2C) es un algoritmo que combina la aproximación de políticas y el aprendizaje por valor. A2C utiliza dos redes neuronales: una para la política (actor) y otra para la función de valor (crítico). Este enfoque permite una actualización más eficiente de la política del agente.

El uso de estos algoritmos se justifica por su popularidad y eficacia probada en una amplia variedad de entornos, así como por su implementación en la librería SB3, lo que facilita su uso. Además, permiten comparar diferentes enfoques de aprendizaje por refuerzo profundo bajo condiciones similares, proporcionando una visión más completa sobre el rendimiento y las características de cada uno.

[4] `https://gymnasium.farama.org/environments/box2d/lunar_lander/`

2.3. Elección de Hiperparámetros

Para la selección de los hiperparámetros de los 3 algoritmos, se ha realizado una búsqueda de hiperparámetros utilizando la librería Optuna [5]. Esta librería permite realizar una búsqueda eficiente de hiperparámetros mediante técnicas de optimización bayesiana, lo que facilita encontrar combinaciones óptimas para cada algoritmo.

Se han realizado 100 iteraciones de búsqueda para cada algoritmo, por cada iteración se ha entrenado un agente durante 50.000 pasos y se ha evaluado su rendimiento en 5 episodios. La recompensa media obtenida en estos episodios se ha utilizado como métrica para evaluar el rendimiento del agente y guiar la búsqueda de hiperparámetros. Las Tablas 1, 2 y 3 muestran los rangos de hiperparámetros seleccionados a explorar para cada algoritmo, junto con los mejores valores encontrados durante la búsqueda.

Cuadro 1: Hiperparámetros explorados y seleccionados para DQN en LunarLander-v3

Hiperparámetro	**Rango explorado**	**Valor seleccionado**
Tamaño capa oculta	{[64, 64], [128, 128], [256, 256]}	[64, 64]
Learning rate	$[10^{-5}, 10^{-3}]$ (log)	0.00022
`buffer_size`	{10 000, 50 000, 100 000}	10 000
`batch_size`	{32, 64, 128}	128
Gamma	[0,95, 0,999]	0.959
`train_freq`	{1, 4, 8}	1
`exploration_fraction`	[0,1, 0,5]	0.248
`exploration_final_eps`	[0,01, 0,1]	0.041

Cuadro 2: Hiperparámetros explorados y seleccionados para PPO en LunarLander-v3

Hiperparámetro	**Rango explorado**	**Valor seleccionado**
Tamaño capa oculta (pi)	{64, 128, 256}	256
Tamaño capa oculta (vf)	{64, 128, 256}	256
Learning rate	$[10^{-5}, 10^{-3}]$ (log)	0.00080
`n_steps`	{128, 256, 512}	128
Gamma	[0,95, 0,999]	0.976

Cuadro 3: Hiperparámetros explorados y seleccionados para A2C en LunarLander-v3

Hiperparámetro	**Rango explorado**	**Valor seleccionado**
Tamaño capa oculta (pi)	{64, 128, 256}	128
Tamaño capa oculta (vf)	{64, 128, 256}	256
Learning rate	$[10^{-5}, 10^{-3}]$ (log)	0.00070
n_steps	{5, 20, 50, 100}	5
Gamma	[0,95, 0,999]	0.999
`ent_coef`	[0,0, 0,1]	0.0144
`vf_coef`	[0,25, 1,0]	0.922

[5] `https://optuna.org/`

2.4. Entrenamiento de las Políticas

Para entrenar las políticas de los tres algoritmos, se ha utilizado el entorno `LunarLander-v3` con los hiperparámetros seleccionados previamente. Cada agente se ha entrenado durante 500.000 pasos, evaluando su rendimiento cada 5.000 pasos en 50 episodios. La recompensa media de estas evaluaciones ha servido para monitorizar el progreso y determinar si el entorno se resuelve.

La Figura 3 muestran la evolución de la recompensa media y la longitud de los episodios para DQN, PPO y A2C. DQN y PPO presentan un entrenamiento relativamente estable, con incrementos graduales en la recompensa, aunque sin llegar a resolver el entorno de forma consistente. A2C, en cambio, muestra una mayor variabilidad, pero alcanza recompensas medias superiores.

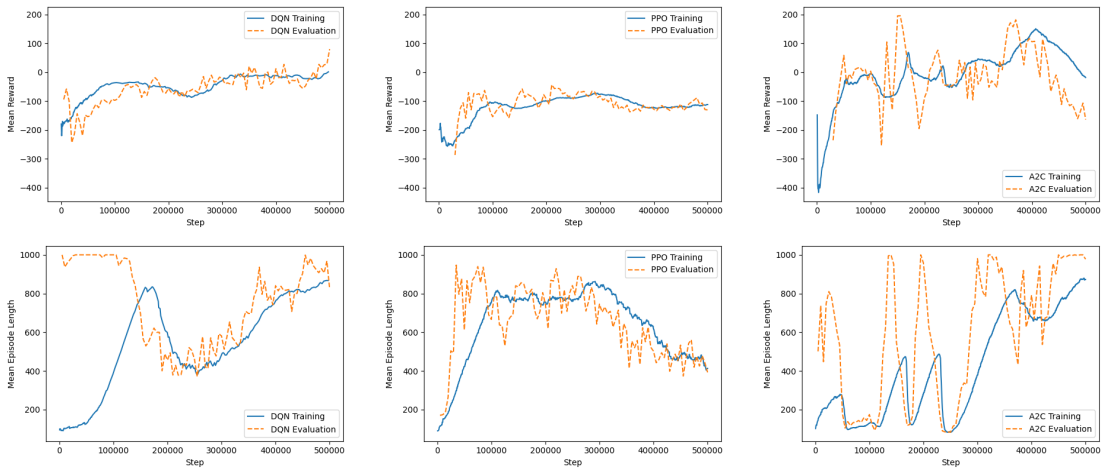

Figura 3: Recompensa media (arriba) y longitud de episodio (abajo) de los agentes DQN (izquierda), PPO (centro) y A2C (derecha) en LunarLander.

2.5. Evaluación de las Políticas

Para evaluar las políticas finales, se ha seleccionado el mejor modelo según la mayor recompensa alcanzada durante el entrenamiento. Cada modelo se ha evaluado en 50 episodios, calculando la recompensa media obtenida. La Tabla 4 muestra que A2C logra la mayor recompensa media, seguido de DQN y PPO. Sin embargo, ninguno de los algoritmos alcanza de forma consistente la recompensa media de 200 necesaria para resolver el entorno.

Dado que el objetivo es maximizar la recompensa media y acercarse lo máximo posible a resolver el entorno, se elige A2C como política final. A2C obtiene la mayor recompensa media (178.00) y demuestra una mayor capacidad de aprendizaje en este problema.

Cuadro 4: Recompensa media y desviación estándar de las políticas finales en LunarLander-v3

Algoritmo	**Recompensa media**	**Desviación estándar**
DQN	52.32	113.04
PPO	-72.82	69.14
A2C	178.00	95.20

3. Entrenamiento de Atari Breakout

En el último ejercicio se aborda el entorno `ALE/Breakout-v5` de Gymnasium[6], uno de los clásicos de Atari. En este juego, el agente controla una pala situada en la parte inferior de la pantalla y debe evitar que la pelota caiga, destruyendo los ladrillos de la parte superior. El entorno presenta un espacio de acciones discreto de 4 posibles movimientos: `NOOP` (0), `FIRE` (1), mover a la `DERECHA` (2) y a la `IZQUIERDA` (3). El estado se representa como una imagen RGB de $210 \times 160 \times 3$ píxeles. La recompensa se otorga al destruir ladrillos, variando según el color del ladrillo.

Para abordar este entorno, se ha empleado el algoritmo A2C de SB3, utilizando la política *CnnPolicy*, que incorpora redes convolucionales para procesar la información visual. Además, se ha aplicado la técnica de apilado de frames (`VecFrameStack`) para proporcionar al agente una secuencia de imágenes, permitiendo capturar la dinámica temporal del entorno y facilitando el aprendizaje de estrategias más complejas.

El entrenamiento se ha realizado durante 1.000.000 de pasos, evaluando el rendimiento del agente cada 5.000 pasos mediante la clase `EvalCallback` de SB3. En este caso, no se ha utilizado un umbral de recompensa para detener el entrenamiento, ya que el objetivo es maximizar la recompensa media alcanzada. Se han mantenido los hiperparámetros por defecto de A2C: *learning rate* de 0.0007, `n_steps` de 5 y *gamma* de 0.99.

La Figura 4 muestra la evolución de la recompensa media y la longitud de los episodios durante el entrenamiento. Se observa una mejora progresiva en el rendimiento del agente, aunque tras 1 millón de pasos y varias horas de entrenamiento la recompensa media obtenida es de 12.60 $\pm$ 4.48 puntos. Esto indica que, si bien el agente aprende a interactuar con el entorno, aún no alcanza un nivel de juego avanzado. Dada la complejidad del entorno y la naturaleza visual del problema, sería necesario un entrenamiento más prolongado o el ajuste de hiperparámetros para lograr un desempeño significativamente mejor.

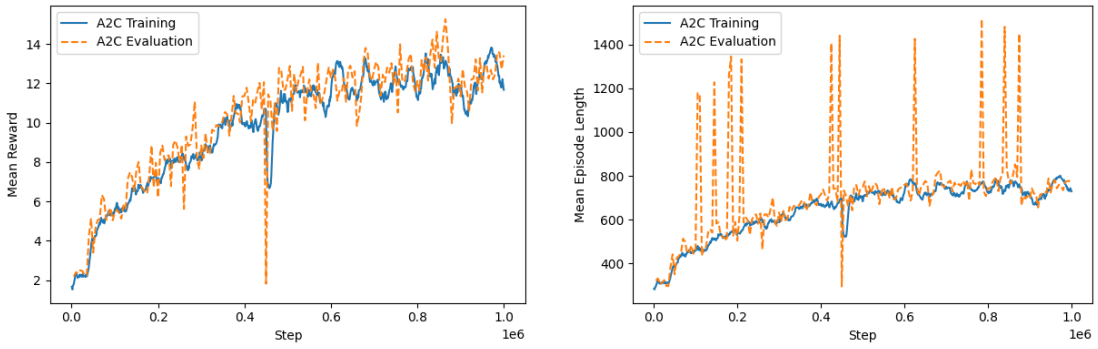

Figura 4: Recompensa media (izquierda) y longitud de episodio (derecha) del agente A2C en Breakout.

[6] `https://ale.farama.org/environments/breakout/`

A. Entorno de Ejecución

Los experimentos se han realizado en un entorno Jupyter Notebook ejecutado desde Visual Studio Code, utilizando un equipo personal con procesador Intel i9-13900K (24 núcleos, 32 hilos), 94 GB de memoria RAM y una tarjeta gráfica NVIDIA RTX A4000 con 16 GB de VRAM.

B. Recursos Adicionales

El código asociado a esta práctica se encuentra disponible en el repositorio de GitHub `javiervela/grid-world-q-learning-agent`[7]. A continuación se muestran algunos los enlaces relevantes al repositorio:

- Vídeos de los entornos
- Jupyter Notebooks con el código de los ejercicios

[7] `https://github.com/javiervela/grid-world-q-learning-agent`