



26-5-2025

Proyecto Final: Aprendizaje por Refuerzo y Q-Learning

Grado de Ingeniería Informática e
Inteligencia Artificial



JAVIER REVUELTA FERNÁNDEZ

Aprendizaje Automático No Supervisado
Y Por Refuerzo – 2ºA

~ ÍNDICE ~

Resumen:	0
Caso de uso 1: Cliff Walking	1
Descripción del entorno:	1
Exploración y selección de hiperparámetros:	1
Entrenamiento final del agente y análisis de la Q-table:	2
Evolución del aprendizaje y análisis de resultados:	3
Conclusión:	4
Caso de uso 2: Taxi	5
Descripción del entorno:	5
Exploración y selección de hiperparámetros:	6
Entrenamiento final del agente y análisis de la Q-table:	7
Evolución del aprendizaje y análisis de resultados:	8
Conclusión:	9
Conclusión Final:	10

Resumen:

En este documento se presenta el desarrollo del proyecto final de la asignatura de "Aprendizaje Automático No Supervisado y por Refuerzo", correspondiente al segundo curso del grado en ingeniería informática e inteligencia artificial en la Universidad San Jorge. El objetivo principal del trabajo es aplicar técnicas de aprendizaje por refuerzo, y en particular el algoritmo Q-Learning, en entornos de la librería Gymnasium, con el fin de entrenar agentes capaces de aprender comportamientos óptimos a través de la interacción con su entorno.

A lo largo del proyecto se abordan dos entornos: CliffWalking-v0, un entorno determinista con fuertes penalizaciones que representa un desafío típico en problemas de exploración versus explotación; y Taxi-v3, un entorno más complejo donde el agente debe aprender a recoger y dejar a un pasajero en ubicaciones específicas, penalizando decisiones incorrectas o inefficientes.

El desarrollo del proyecto se ha centrado en entrenar agentes utilizando Q-Learning desde cero, diseñando e implementando la lógica de aprendizaje mediante funciones modulares en Python. Además, se ha hecho especial énfasis en la exploración y justificación de los hiperparámetros más relevantes (learning rate, gamma, decay rate y número de episodios), evaluando distintas combinaciones y analizando su impacto en el rendimiento del agente.

Además del entrenamiento, se ha realizado un seguimiento de la evolución del aprendizaje a través de métricas como la recompensa media y el número de pasos por episodio, generando gráficas que permiten interpretar la calidad de la política aprendida. Finalmente, se ha exportado la Q-table entrenada de cada agente y se ha generado un GIF que muestra el comportamiento final del agente, permitiendo una comprensión visual de la solución aprendida.

Este proyecto ha permitido consolidar los conceptos teóricos impartidos en la asignatura, aplicándolos en entornos prácticos donde se puede apreciar la influencia de los distintos componentes del algoritmo Q-Learning. Asimismo, ha reforzado la comprensión de los retos típicos del aprendizaje por refuerzo, como el equilibrio entre exploración y explotación, y la sensibilidad del agente a los valores de los hiperparámetros, sentando una base sólida para futuras aplicaciones y trabajos en el ámbito de la ingeniería informática e inteligencia artificial.

Caso de uso 1: Cliff Walking

Descripción del entorno:

El primer entorno del proyecto ha sido “CliffWalking-v0”, un entorno de la librería Gymnasium cuyo objetivo es entrenar a un agente para que aprenda a moverse por una cuadrícula evitando grandes penalizaciones.

El entorno consiste en una cuadrícula de 4 filas por 12 columnas (48 estados posibles totales) desde la cual el agente comienza en la esquina inferior izquierda y debe conseguir llegar a la esquina inferior derecha superando las casillas calificadas como “acantilado”. Éstas se encuentran en la parte inferior central de la cuadrícula y, si se pisa, se penaliza fuertemente al agente con una recompensa negativa de -100, devolviéndolo a continuación al estado inicial.

El espacio de observación está formado por 48 estados discretos, donde cada número representa una celda de la cuadrícula. Por otro lado, el espacio de acciones está compuesto por cuatro acciones discretas: moverse hacia arriba (0), hacia la derecha (1), hacia abajo (2), o hacia la izquierda (3).

Las recompensas están definidas de la siguiente forma:

- Recompensa por moverse a una celda válida --> -1
- Recompensa por caer al acantilado --> -100 y reinicio al estado inicial.
- El episodio termina cuando el agente alcanza el estado objetivo o "la meta".

Este entorno es ideal para estudiar el equilibrio entre exploración y explotación, ya que para alcanzar la meta con éxito y eficiencia el agente debe aprender a tomar rutas arriesgadas sin caer, gestionando adecuadamente la penalización frente a la recompensa.

Además, en esta versión (`is_slippery=False`), el entorno es determinista, es decir, cada vez que el agente ejecuta una acción desde un estado, el resultado es siempre el mismo. No hay aleatoriedad ni incertidumbre en las transiciones. Esto facilita el aprendizaje, ya que el agente puede predecir con precisión las consecuencias de sus acciones, permitiéndonos centrarnos en el impacto del algoritmo y de los hiperparámetros sin introducir ruido estocástico adicional.

Exploración y selección de hiperparámetros:

Para obtener un agente capaz de desenvolverse de manera óptima en el entorno indicado, se llevó a cabo un proceso de búsqueda y ajuste de hiperparámetros clave del algoritmo Q-Learning: “learning rate”, “gamma” o factor de descuento y “decay rate” de épsilon para la política épsilon-greedy.

Para ello, se diseñó una primera exploración de 27 combinaciones posibles (3 valores para cada hiperparámetro), evaluando cada una con 5 ejecuciones distintas mediante semillas controladas. De este modo, se garantizó la consistencia estadística y se obtuvo una estimación robusta del rendimiento del agente, evitando que las decisiones dependiesen de una única ejecución aleatoria. La evaluación de cada combinación se basó en la recompensa media de los últimos 100 episodios del entrenamiento ya que proporciona una visión precisa del comportamiento del agente una vez finalizado el proceso de aprendizaje,

evitando que las recompensas negativas asociadas a las primeras fases de exploración distorsionen el análisis.

En esta primera fase, los valores explorados fueron:

- Learning_rate: [0.1, 0.3, 0.7]
- Gamma: [0.90, 0.95, 0.99]
- Decay_rate: [0.0005, 0.001, 0.005]

La combinación que obtuvo el mejor resultado fue: "learning_rate" = 0.7, "gamma" = 0.90 , y decay_rate = 0.005; con una recompensa media de -25.66 y una desviación estándar de 2.10.

En cuanto al learning rate, las configuraciones con learning rate más bajos (0.1 o 0.3) mostraron un rendimiento inferior, mientras que 0.7 permitió un ajuste más rápido de la Q-table, confirmando el mejor funcionamiento de un aprendizaje más agresivo en este tipo de entornos penalizantes.

En cuanto al factor de descuento "gamma", un valor moderado de 0.9 resultó el más adecuado, ya que el entorno penaliza errores inmediatos como caer en el acantilado, por lo que priorizar el corto plazo fue más efectivo que valorar en exceso las recompensas futuras.

Finalmente, el mejor valor de "decay_rate" fue 0.005, lo cual nos indica que en este entorno determinista es más beneficioso reducir rápidamente la exploración aleatoria para explotar cuanto antes lo aprendido. Esta rápida transición de exploración a explotación es clave en entornos deterministas y simples donde la política óptima puede consolidarse con relativa rapidez.

Para asegurar que esta combinación no correspondía simplemente a un máximo local, se llevó a cabo una segunda exploración más exhaustiva, extendiendo los rangos de los hiperparámetros en función de los resultados obtenidos anteriormente. A pesar de ello, no se encontró ninguna combinación que mejorase el rendimiento obtenido previamente, confirmando así la solidez de la solución seleccionada.

Por último, el número de episodios de entrenamiento se fijó en 5000 tras observar que el agente alcanzaba un comportamiento estable antes de ese umbral. En este entorno, continuar el entrenamiento más allá de ese punto no ofrecía mejoras significativas en la recompensa media, pero aumentaba enormemente el coste computacional, por lo que se optó por un equilibrio eficiente entre rendimiento y recursos.

Entrenamiento final del agente y análisis de la Q-table:

Una vez seleccionados los hiperparámetros óptimos, se procedió al entrenamiento final del agente utilizando la implementación del algoritmo Q-Learning desarrollada en Python. Para este entorno, se utilizó una política ϵ -greedy con decaimiento exponencial, la cual permite al agente explorar acciones aleatorias en las primeras fases del entrenamiento y, progresivamente, centrarse en la explotación de aquellas acciones que maximizan el valor Q aprendido. Este enfoque favorece la construcción de una política sólida, ya que primero garantiza la diversidad de experiencias y posteriormente refuerza las decisiones más eficaces descubiertas durante el proceso de aprendizaje.

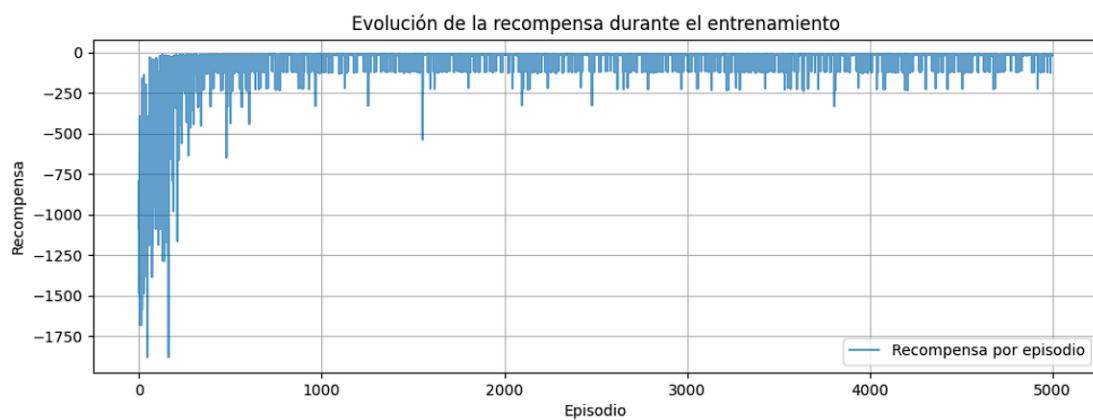
Tras 5000 episodios de entrenamiento, se obtuvo una Q-table final de dimensiones (48, 4), donde cada fila representa un estado del entorno y cada columna el valor estimado para una de las cuatro acciones posibles. Esta tabla recoge el conocimiento aprendido por el agente a lo largo del proceso de entrenamiento, y permite extraer la política óptima seleccionando en cada estado la acción con mayor valor Q.

En términos generales, los valores numéricos de la Q-table reflejan la calidad de las acciones en cada estado: valores negativos más cercanos a cero indican trayectorias eficientes hacia la meta, mientras que valores muy negativos corresponden a decisiones que conducen al acantilado o a rutas ineficientes. Esta matriz contiene el conocimiento estratégico adquirido por el agente, y constituye la base de su comportamiento dentro del entorno.

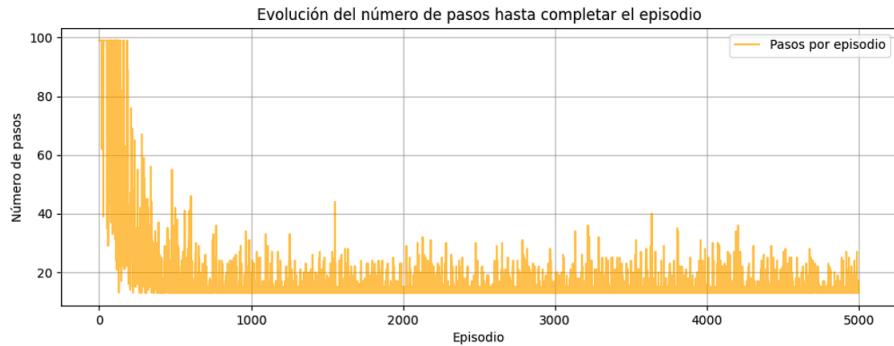
Evolución del aprendizaje y análisis de resultados:

Para evaluar el proceso de aprendizaje del agente en el entorno CliffWalking-v0, se representaron diversas gráficas que nos permiten analizar su rendimiento, estabilidad y convergencia a lo largo del entrenamiento. Estas visualizaciones nos aportan una perspectiva analítica del comportamiento del agente, permitiéndonos interpretar en qué momento comienza a aprender y cómo evoluciona su política en función del tiempo.

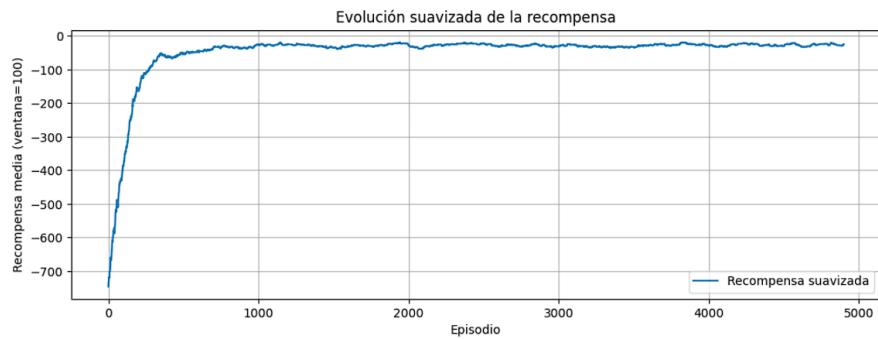
1. Evolución de la recompensa por episodio: En esta primera gráfica se observa cómo el agente, durante los primeros episodios, comete numerosos errores que lo llevan a caer repetidamente en el acantilado, acumulando recompensas muy negativas. Sin embargo, a medida que el entrenamiento avanza, se aprecia una mejora progresiva y sostenida en la recompensa, estabilizándose finalmente en valores cercanos a cero. Esto indica que el agente ha aprendido a evitar las zonas de alto castigo y a alcanzar la meta con un comportamiento más eficiente.



2. Evolución del número de pasos por episodio: Esta gráfica complementa la anterior mostrando cuántos pasos necesita el agente para completar cada episodio. Inicialmente, el número de pasos es muy variable, y en muchas ocasiones elevado, lo cual se debe a trayectorias ineficientes o episodios que terminan abruptamente por caer al acantilado. A partir de aproximadamente el episodio 750, se aprecia una notable estabilización en torno a los 20 pasos por episodio, lo cual nos sugiere que el agente ha aprendido una trayectoria segura y directa hacia el objetivo, minimizando tanto errores como rutas innecesarias.



3. Recompensa suavizada (media móvil): La última visualización aplica una media móvil sobre la serie de recompensas para reducir el ruido y resaltar la tendencia global. Esta técnica permite observar con mayor claridad la convergencia del agente. En esta gráfica se puede identificar una mejora progresiva a lo largo de los primeros episodios, seguida de una meseta que comienza alrededor del episodio 1000, indicando que el agente ha alcanzado una política razonablemente estable y eficaz.



En conjunto, estas tres gráficas muestran que el agente ha sido capaz de aprender de forma progresiva, constante y robusta. A través del entrenamiento, ha dejado de tomar decisiones aleatorias o penalizadas y ha consolidado una política que le permite alcanzar su objetivo minimizando riesgos y tiempo. Este comportamiento evidencia que los hiperparámetros seleccionados han permitido una buena convergencia del algoritmo Q-Learning en este entorno, y validan el éxito del entrenamiento realizado.

Conclusión:

El entorno de Cliff Walking ha permitido ilustrar de forma clara los desafíos característicos del aprendizaje por refuerzo en entornos deterministas con penalizaciones severas. A través del uso del algoritmo Q-Learning y una política ϵ -greedy con decaimiento, el agente ha sido capaz de aprender a evitar el acantilado y alcanzar la meta de forma eficiente tras un proceso de entrenamiento estructurado.

La exploración sistemática de hiperparámetros ha demostrado ser clave para obtener un comportamiento óptimo, y los resultados visuales que apreciamos en el notebook adjunto confirman una convergencia clara hacia una política estable. Este primer caso de uso ha servido como base para afianzar tanto los conceptos teóricos como las habilidades prácticas necesarias para abordar entornos más complejos, como se verá en el siguiente apartado.

Caso de uso 2: Taxi

Descripción del entorno:

El segundo entorno abordado en este proyecto ha sido Taxi-v3, también incluido en la librería Gymnasium. Este entorno presenta un nivel de complejidad mayor respecto a Cliff Walking, tanto en el tamaño del espacio de estados como en la tarea que el agente debe realizar.

El entorno consiste en una cuadrícula de 5x5 donde un taxi debe recoger a un pasajero en una de cuatro ubicaciones posibles y llevarlo a su destino, que también será una de esas cuatro posiciones. El agente debe aprender a navegar por la cuadrícula, recoger correctamente al pasajero y dejarlo en su destino, evitando penalizaciones. El espacio de observación de este nuevo entorno también es discreto y finito, con un total de 500 estados posibles. Cada estado representa una combinación entre la posición del taxi (25 posiciones posibles (5x5)), la ubicación del pasajero (4 ubicaciones de recogida + 1 posición si está dentro del taxi), y la ubicación de destino (4 posibles ubicaciones).

Para lograr el objetivo definido, el agente dispone de un conjunto de 6 acciones posibles que corresponden a:

- 0: Mover hacia el sur (abajo) 
- 1: Mover hacia el norte (arriba) 
- 2: Mover hacia el este (derecha) 
- 3: Mover hacia el oeste (izquierda) 
- 4: Recoger al pasajero
- 5: Dejar al pasajero

Cada acción tiene un efecto determinista sobre el estado del entorno. El taxi se desplazará siempre que no se encuentre con una pared. En ese caso, el estado no cambia, pero se penaliza igualmente con una recompensa de -1.

El sistema de recompensas en este caso está diseñado con el objetivo de impulsar decisiones eficientes, es decir, que se cumpla el objetivo en los menores pasos posibles. Es por esto que el agente obtendrá:

- -1 por cada acción válida o inválida (aunque no se desplace si se encuentra con una pared, el tiempo sigue contando negativamente, por ejemplo)
- -10 por intentar recojer o dejar al pasajero en una ubicación incorrecta
- +20 por completar correctamente el trayecto (dejar al pasajero correctamente en su destino)

El episodio termina únicamente cuando el pasajero ha sido recogido y dejado correctamente en su lugar de destino. Este diseño recompensa la efectividad y la eficiencia, penalizando las rutas innecesariamente largas.

A diferencia del entorno anterior, aquí el agente debe aprender a realizar una secuencia específica de acciones (navegar, recoger, navegar y dejar) respetando las reglas del entorno. Esto introduce un componente adicional de razonamiento y planificación en comparación con entornos puramente de navegación.

Exploración y selección de hiperparámetros:

La exploración de hiperparámetros para este entorno siguió un procedimiento muy similar que en el entorno anterior. En una primera fase, se evaluaron 27 combinaciones posibles de valores para “learning_rate”, “gamma” y “decay_rate”, replicando la misma metodología: cinco ejecuciones por combinación, semillas distintas para garantizar consistencia estadística y evaluación basada en la recompensa media de los últimos 100 episodios.

Tras esta primera búsqueda, se observó que varias combinaciones obtenían resultados similares, pero destacaba especialmente la configuración de “learning_rate” = 0.3, “gamma” = 0.95 y “decay_rate” = 0.005, que ofrecía una de las mejores recompensas medias, junto con un buen equilibrio entre rendimiento y estabilidad.

A continuación, se procedió a determinar el número óptimo de episodios necesarios para el entrenamiento. Para ello, se mantuvo la mejor combinación de hiperparámetros encontrada hasta el momento y se evaluaron 5 configuraciones de episodios: 1000, 2000, 5000, 7500 y 10000. Los resultados mostraron que, a partir de los 5000 episodios, el agente ya alcanzaba una política razonablemente eficiente. Sin embargo, con 10 000 episodios se lograba una mayor recompensa media y una menor variabilidad, por lo que se decidió establecer este valor como estándar para el entrenamiento final del agente. No obstante, debido al elevado coste computacional de entrenar con 10 000 episodios por combinación, se optó por utilizar 5000 episodios durante las fases de exploración extendidas de hiperparámetros, ya que los resultados obtenidos con esta cantidad de episodios eran suficientemente buenos para identificar tendencias y diferencias de rendimiento.

Con esta estrategia, se realizó una segunda exploración más precisa, acotando los rangos de los hiperparámetros más prometedores:

- Learning_rate: [0.3, 0.4, 0.5]
- Gamma: [0.90, 0.95]
- Decay_rate: [0.005, 0.0075, 0.01]

Esta búsqueda reveló que la mejor combinación era learning_rate = 0.4, gamma = 0.95, decay_rate = 0.01, alcanzando una recompensa media de 5.876, la más alta registrada en todo el proceso. Aunque presentaba una ligera desviación estándar, esta se mantenía dentro de rangos razonables y aceptables.

Para confirmar la validez de este resultado, se realizó una última búsqueda centrada exclusivamente en valores superiores de decay_rate: [0.01, 0.015, 0.02, 0.03]. Los resultados confirmaron que decay_rate = 0.01 seguía siendo el más competitivo, ya que valores más altos penalizaban el rendimiento por provocar una explotación demasiado temprana.

Por todo esto, finalmente se escogieron los siguientes parámetros para el entrenamiento final del agente tras haber demostrado ser la más eficaz en términos de rendimiento, estabilidad y eficiencia de aprendizaje en este entorno complejo:

- Learning_rate: 0.4
- Gamma: 0.95
- Decay_rate: 0.01
- Número de episodios: 10.000

Entrenamiento final del agente y análisis de la Q-table:

Una vez identificada la mejor combinación de hiperparámetros (“learning_rate” = 0.4, “gamma” = 0.95, “decay_rate” = 0.01), se procedió al entrenamiento final del agente durante 10 000 episodios. Como en el caso anterior, se utilizó la política ϵ -greedy con decaimiento exponencial, que permite al agente explorar el entorno al principio del entrenamiento y centrarse progresivamente en explotar las decisiones que han demostrado ser más eficaces.

El agente fue entrenado desde cero utilizando el algoritmo Q-Learning, generando una Q-table final de dimensiones (500, 6). Cada fila de esta matriz representa uno de los 500 estados posibles del entorno, definidos por la combinación de la ubicación del taxi, la posición del pasajero y el destino; y cada columna corresponde a una de las seis acciones disponibles (moverse en las cuatro direcciones, recoger y dejar al pasajero).

La Q-table refleja el conocimiento adquirido por el agente a lo largo del entrenamiento. Los valores Q almacenados representan la calidad esperada de ejecutar cada acción en cada estado concreto. Tras completar el proceso de entrenamiento, el agente puede determinar su política óptima seleccionando en cada estado la acción con mayor valor Q.

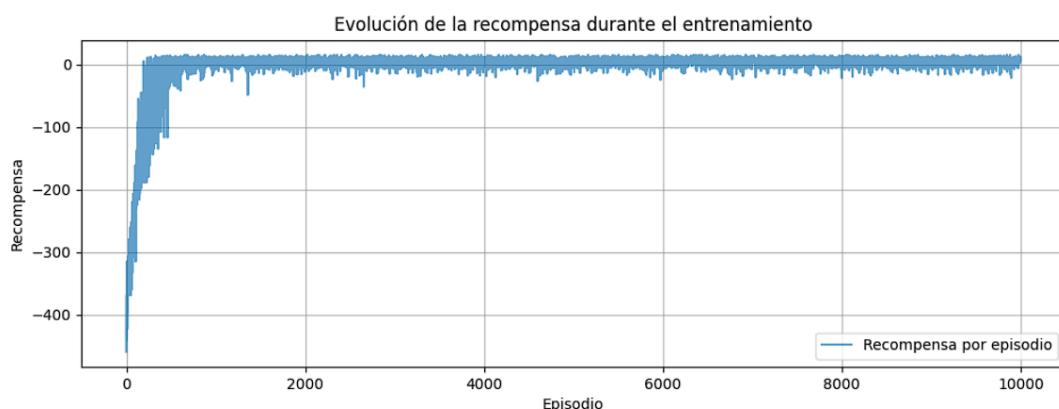
A nivel general, se observan valores Q altos en aquellos estados cercanos a la entrega exitosa del pasajero, y valores bajos o negativos en acciones que conducen a penalizaciones, como intentar recoger o dejar al pasajero en ubicaciones incorrectas. Esto nos sugiere que el agente ha aprendido correctamente a realizar la secuencia de acciones requeridas por el entorno.

Este entrenamiento demuestra que la política aprendida permite al agente desenvolverse con eficiencia en un entorno más complejo y estructurado, gestionando tanto la navegación como las decisiones contextuales.

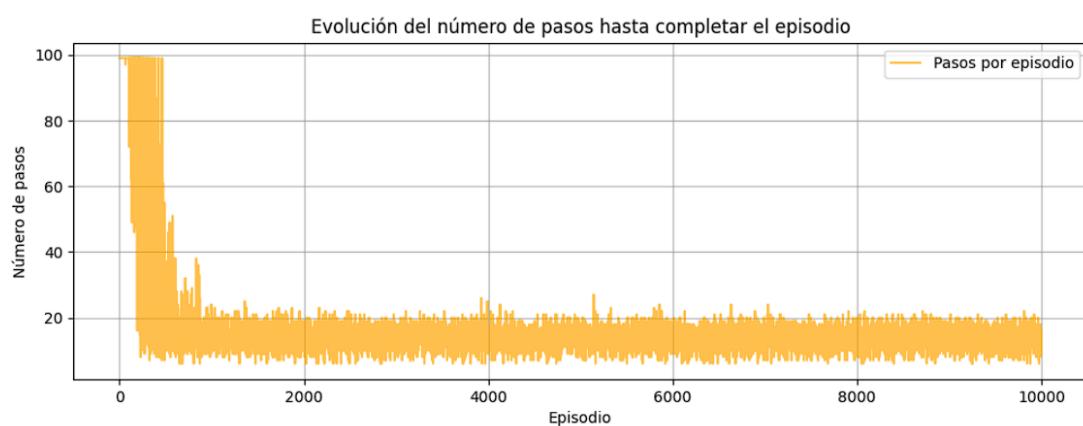
Evolución del aprendizaje y análisis de resultados:

Para evaluar el rendimiento y la convergencia del agente entrenado en el entorno Taxi-v3, se representaron diversas gráficas que muestran la evolución de su comportamiento a lo largo de los 10 000 episodios de entrenamiento. Estas visualizaciones nos permiten observar tanto la mejora progresiva en la recompensa como la progresión en eficiencia del agente a la hora de completar su tarea.

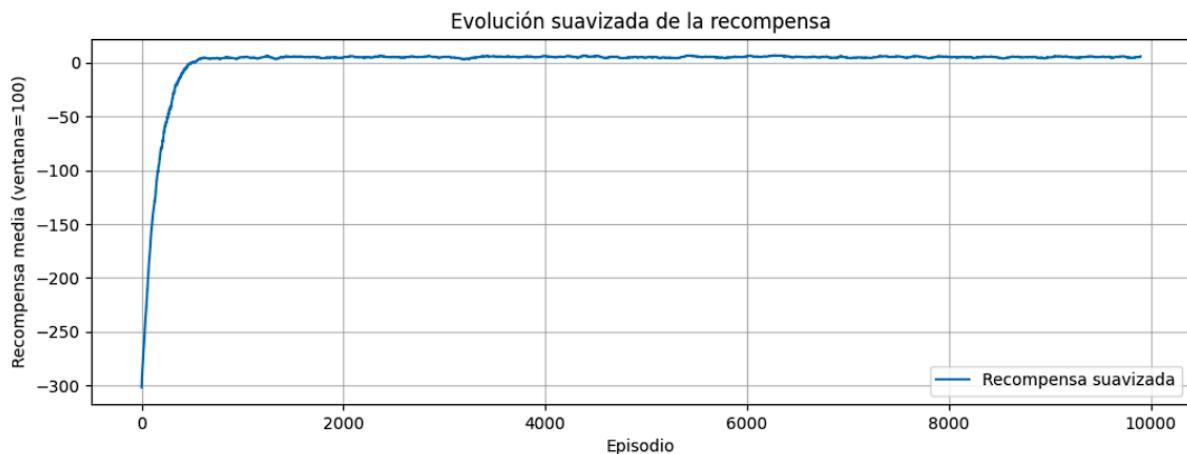
1. Evolución de la recompensa por episodio: En esta gráfica se observa cómo, durante las primeras etapas del entrenamiento, el agente obtiene recompensas negativas con alta variabilidad, debido a la exploración y a las frecuentes penalizaciones por ejecutar acciones incorrectas. Sin embargo, a medida que progresa el entrenamiento, las recompensas mejoran de forma significativa, estabilizándose en torno a valores positivos. Esta estabilización indica que el agente ha aprendido a realizar las secuencias de acciones correctas para completar los episodios con éxito: recoger al pasajero y dejarlo en su destino minimizando errores.



2. Evolución del número de pasos por episodio: Esta gráfica muestra una tendencia descendente en la cantidad de pasos necesarios para completar cada episodio. Al inicio, el agente actúa de manera ineficiente, con rutas largas o decisiones incorrectas que alargan la duración del episodio. A partir de aproximadamente el episodio 1000, se aprecia una caída clara y sostenida en el número de pasos, que se estabiliza en torno a los 13–15 pasos, lo cual indica que el agente ha aprendido a completar su tarea de forma más directa y eficaz.



3. Recompensa suavizada (media móvil): Esta última gráfica aplica una media móvil para eliminar el ruido de las recompensas individuales. La tendencia global nos muestra una mejora clara durante las primeras fases, seguida de una meseta estable en torno a valores positivos a partir del episodio 1000 - 1500. Esta evolución representa que el agente ha convergido hacia una política estable y eficaz que maximiza la recompensa y minimiza las penalizaciones.



En conjunto, estas gráficas reflejan un proceso de aprendizaje progresivo, en el que el agente ha pasado de comportamientos aleatorios e ineficientes a tomar decisiones estratégicas y optimizadas. La combinación final de hiperparámetros seleccionada ha demostrado ser adecuada para guiar el aprendizaje del agente en este entorno complejo, permitiéndole adquirir una política eficaz tanto en términos de recompensa como de eficiencia temporal.

Conclusión:

El entorno Taxi-v3 ha permitido abordar un escenario más complejo que el de navegación básica, en el que el agente debía aprender a ejecutar correctamente una secuencia de acciones: navegar, recoger y dejar al pasajero en el destino correcto. Esta dinámica ha requerido que el agente no solo evite penalizaciones, sino que también comprenda y ejecute acciones contextuales dependientes del estado.

La exploración rigurosa de los hiperparámetros del algoritmo Q-Learning (“learning_rate” = 0.4, “gamma” = 0.95 y “decay_rate” = 0.01; junto con 10 000 episodios de entrenamiento) permitieron maximizar el rendimiento y alcanzar una política estable y eficiente para el agente.

Finalmente, el análisis visual del aprendizaje confirma que el agente ha evolucionado desde un comportamiento aleatorio e ineficaz hasta completar los episodios con trayectorias consistentes y recompensas positivas sostenidas. Este segundo caso de uso ha servido para consolidar el conocimiento práctico sobre aprendizaje por refuerzo en entornos discretos, demostrando la efectividad del enfoque seguido incluso en tareas secuenciales más exigentes.

Conclusión Final:

El desarrollo de este proyecto ha permitido aplicar de forma práctica los conceptos fundamentales del aprendizaje por refuerzo estudiados en la asignatura, poniendo en práctica el algoritmo Q-Learning en dos entornos discretos de distinta complejidad: CliffWalking-v0 y Taxi-v3. Ambos casos de uso han sido abordados desde cero, diseñando un sistema modular en Python para entrenar y evaluar agentes capaces de aprender a través de la interacción con su entorno.

El entorno CliffWalking ha servido como introducción al comportamiento del agente en entornos deterministas con penalizaciones severas, donde la planificación a corto plazo y la rápida consolidación de la política son fundamentales. En cambio, el entorno Taxi-v3 ha planteado un reto de mayor complejidad, tanto por el tamaño del espacio de estados como por la naturaleza de la tarea. Aquí el agente no solo debe navegar, sino también tomar decisiones dependientes del contexto, como cuándo recoger o dejar al pasajero. A pesar de ello, la metodología seguida ha permitido alcanzar una política eficaz y consistente para ambos entorno.

El uso de una política ϵ -greedy con decaimiento exponencial ha demostrado ser una estrategia adecuada para equilibrar exploración y explotación en ambos casos. Asimismo, la elección de la recompensa media en los últimos episodios como métrica principal ha permitido evaluar de manera realista la eficacia final de la política aprendida.

Este trabajo ha servido para consolidar no solo los conocimientos teóricos sobre aprendizaje por refuerzo, sino también competencias relacionadas con el análisis experimental, la justificación técnica de decisiones y la interpretación de resultados, entre otras. Además, nos ha mostrado la sensibilidad del algoritmo Q-Learning a sus hiperparámetros, y la necesidad de adaptar sus configuraciones a las particularidades de cada entorno.

En definitiva, el proyecto ha cumplido sus objetivos, sentando una base sólida para abordar con seguridad nuevos entornos y algoritmos más avanzados dentro del campo de la aprendizaje automático no supervisado y por refuerzo, y, en definitiva, la inteligencia artificial.