Aprendizaje por Refuerzo I

Posgrado en Inteligencia Artificial - FIUBA

Integrantes:

- Ing. Fabricio Denari
- Ing. Marco Joel Isidro
- Ing. Julio Donadello
- Ing. Bruno Masoller

Fecha de entrega: xx/04/2025

Profesor: Ing. Miguel Augusto Azar

Índice

Introducción Teórica

El aprendizaje por refuerzo (Reinforcement Learning, RL) es una rama del aprendizaje automático en la que un agente aprende a tomar decisiones en un entorno para maximizar una recompensa acumulada. A diferencia de otros enfoques supervisados, el agente no recibe respuestas correctas directas, sino que explora acciones y aprende de sus consecuencias (recompensas o penalizaciones) en función del estado actual del entorno.

Q-learning es un algoritmo de aprendizaje por refuerzo libre de modelo (*model-free reinforcement learning*), lo que significa que no requiere un conocimiento previo de la dinámica del entorno (es decir, de las probabilidades de transición entre estados). Su objetivo es aprender una función de acción-valor $Q(s,a)$ que represente la utilidad esperada de tomar una acción a en un estado s y seguir luego la política óptima derivada de dicha función.

La función Q se actualiza según la siguiente ecuación:

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

Donde:

- α es la tasa de aprendizaje, que controla cuánto se actualiza el valor actual con la nueva información.
- γ es el factor de descuento, que determina cuánto se valoran las recompensas futuras respecto a las inmediatas.
- r es la recompensa obtenida al tomar la acción a desde el estado s.
- s' es el nuevo estado alcanzado después de ejecutar la acción.
- $\max_{a'} Q(s',a')$ representa el valor estimado de la mejor acción posible en el nuevo estado s'.

Con una política que asegure suficiente exploración y un horizonte de tiempo suficientemente largo, Q-learning está garantizado a converger hacia una política óptima que maximiza la recompensa acumulada esperada, incluso en entornos con transiciones estocásticas. Esta propiedad fue demostrada formalmente en el trabajo seminal de Christopher Watkins en 1989, en su artículo Technical Note: Q-Learning.

Desarrollo

En esta sección se describe el diseño y la implementación del agente que aprende a desplazarse en una ciudad simulada mediante el algoritmo Q-learning. Para ello se utilizó la biblioteca gym como base del entorno, adaptándolo a nuestras necesidades, y se empleó pygame para su visualización.

Si bien inicialmente se consideró el uso de la librería “gymnasium”, una versión moderna y más modular de “gym”, se optó por descartarla debido a ciertas limitaciones prácticas: no permitía extender fácilmente la grilla a dimensiones superiores como 10x10 ni integrarse de forma flexible con herramientas de búsqueda de hiperparámetros como Optuna.

El entorno simula una ciudad con calles de sentido único, donde el agente (un taxi) debe recoger pasajeros ubicados aleatoriamente y llevarlos a su destino, minimizando penalizaciones asociadas a errores de navegación y al tiempo de recorrido. Este comportamiento se entrena a través de Q-learning, con el objetivo de construir una política óptima que maximice la recompensa acumulada.

Durante el desarrollo se implementaron dos scripts principales:

- train.py, encargado de entrenar la Q-table, ajustar hiperparámetros mediante Optuna y visualizar el aprendizaje.

- test.py, que permite evaluar el desempeño del agente utilizando la política aprendida, con la opción de renderizar o grabar los recorridos.

Implementación

El entorno representa una ciudad cuadriculada de tamaño configurable (por defecto 10x10), en la cual el agente (el taxi) debe cumplir la tarea de recoger pasajeros en ubicaciones aleatorias y llevarlos a sus respectivos destinos.

Espacio de estados

Cada estado se codifica como una combinación de:

- La posición del taxi en la grilla: (taxi_row, taxi_col)
- El índice que indica dónde se encuentra el pasajero (en una ubicación fija o dentro del taxi): passenger_idx
- Un flag booleano que indica si el pasajero está dentro del taxi: in_taxi

Esta codificación permite capturar toda la información relevante para que el agente tome decisiones informadas en cada paso del episodio.

Espacio de acciones

El taxi puede ejecutar una de las siguientes acciones discretas:

- 0: moverse hacia abajo
- 1: moverse hacia arriba
- 2: moverse hacia la derecha
- 3: moverse hacia la izquierda

Cada acción actualiza la posición del taxi, siempre que no se excedan los límites de la grilla.

Política de recompensa

El sistema de recompensas fue diseñado para incentivar trayectorias óptimas y penalizar comportamientos ineficientes:

Políticas de movimiento:

- Penalización de -1 por cada movimiento, lo cual empuja al agente a encontrar caminos más cortos.
- Penalización adicional de -5 por moverse en contramano, es decir, en dirección opuesta al sentido permitido de la calle. Esta penalización no aplica si el taxi se encuentra en una ubicación de pickup o dropoff.

Políticas de pickup:

Requiere que el taxi NO tenga un pasajero a bordo (not self.in_taxi) y que esté parado en la posición de pickup del pasajero asignado.

- Si ambas condiciones se cumplen: marca que el pasajero está ahora en el taxi (self.in_taxi = 1) y da una recompensa positiva de +15 (por hacer el pickup correctamente).
- Si no: penaliza con -15 (porque intentó hacer pickup donde no debía o ya llevaba alguien).

Políticas de dropoff:

Requiere que el taxi tenga un pasajero a bordo (self.in_taxi) y que esté en una de las ubicaciones de dropoff válidas.

- Si ambas condiciones se cumplen: Libera al pasajero (self.in_taxi = 0), da una gran recompensa de +30 (por completar el viaje) y marca el episodio como terminado (done = True).
- Si no: penaliza con -15 (porque intentó dejar un pasajero sin llevar a alguien o en un lugar inválido).

Resumen

Tipo de política	Condiciones	Resultado	Recompensa
Movimiento válido	Movimiento en sentido permitido	Movimiento aceptado	**-1**
Movimiento inválido (contramano)	Movimiento en dirección opuesta a la calle *(excepto en pickup/dropoff)*	Penalización adicional	**-5** *(además del -1 base)*
Pickup válido	No lleva pasajero **y** está en la posición de pickup del pasajero	Se sube el pasajero al taxi (in_taxi = 1)	**+15**
Pickup inválido	Ya lleva pasajero **o** no está en la ubicación correcta	Acción incorrecta	**-15**
Dropoff válido	Lleva pasajero **y** está en una ubicación válida de dropoff	Se baja el pasajero (in_taxi = 0), termina el episodio	**+30**
Dropoff inválido	No lleva pasajero **o** está en lugar incorrecto	Acción incorrecta	**-15**

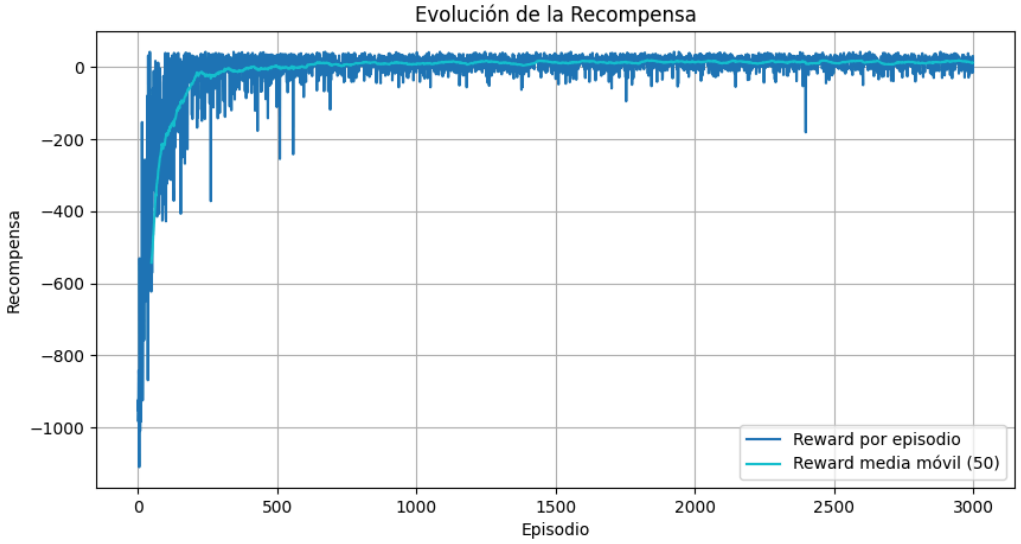
Resultados

Para evaluar el desempeño del agente entrenado con Q-Learning, se realizaron múltiples experimentos y se registraron las métricas más relevantes del proceso de aprendizaje.
A continuación, se presentan los principales resultados obtenidos:

Curvas de Aprendizaje

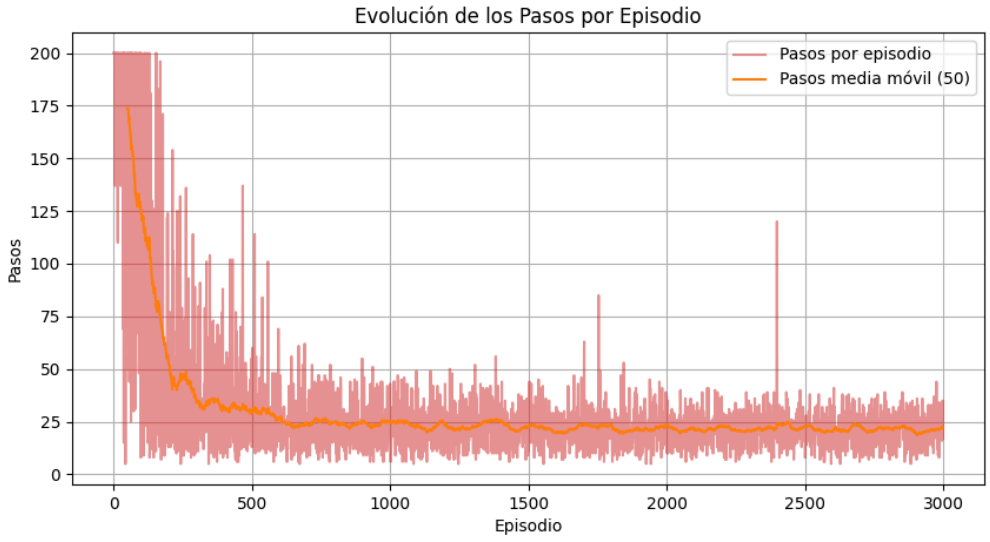
Se graficaron tres métricas clave a lo largo de los episodios:

Recompensa acumulada por episodio



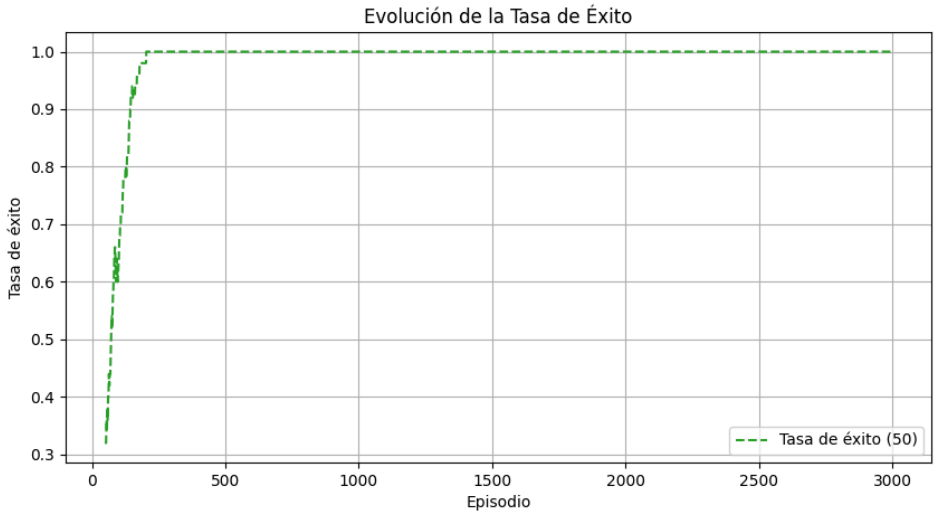
Muestra cómo el agente fue mejorando su desempeño, alcanzando mayores recompensas conforme aprendía a evitar penalizaciones innecesarias y a completar correctamente los viajes.

Cantidad de pasos por episodio



Se observa una tendencia decreciente, indicando que el agente encontró trayectorias más eficientes hacia los objetivos.

Tasa de éxito por episodio

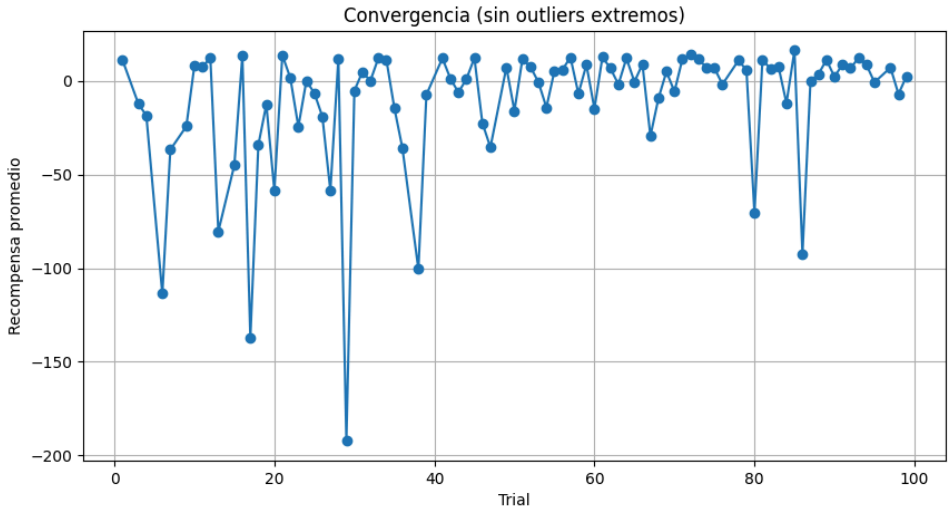


Refleja el porcentaje de episodios en los que el agente logró completar el viaje con éxito (pickup + dropoff), alcanzando una estabilidad después de cierto punto del entrenamiento.

Optimización de Hiperparámetros con Optuna

Se utilizó la biblioteca Optuna para realizar una búsqueda eficiente de los hiperparámetros óptimos (*alpha, gamma y epsilon*). Los siguientes gráficos muestran cómo evolucionó la optimización:

Convergencia de los trials

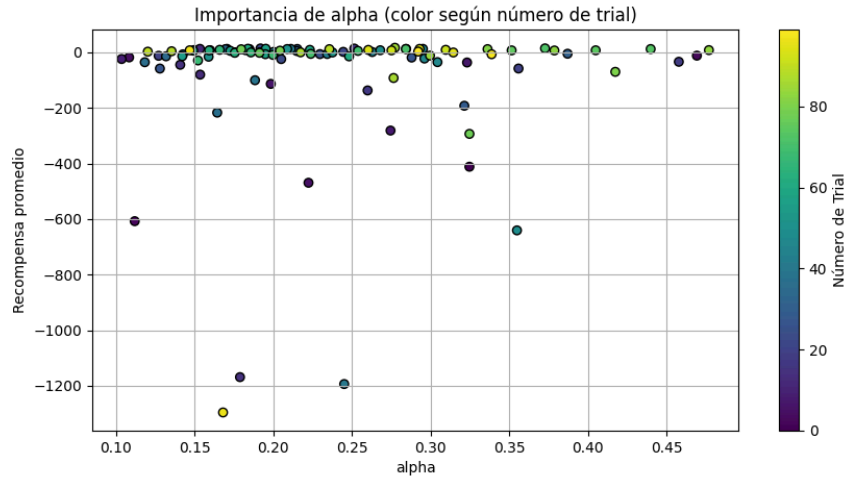


Se realizó un filtrado de outliers para mejorar la visualización.Se realizó un filtrado de outliers para mejorar la visualización. Se observa que, a medida que avanzan los trials, las combinaciones de hiperparámetros evaluadas tienden a converger hacia valores que generan un mejor desempeño del agente. Esto indica que el proceso de optimización fue

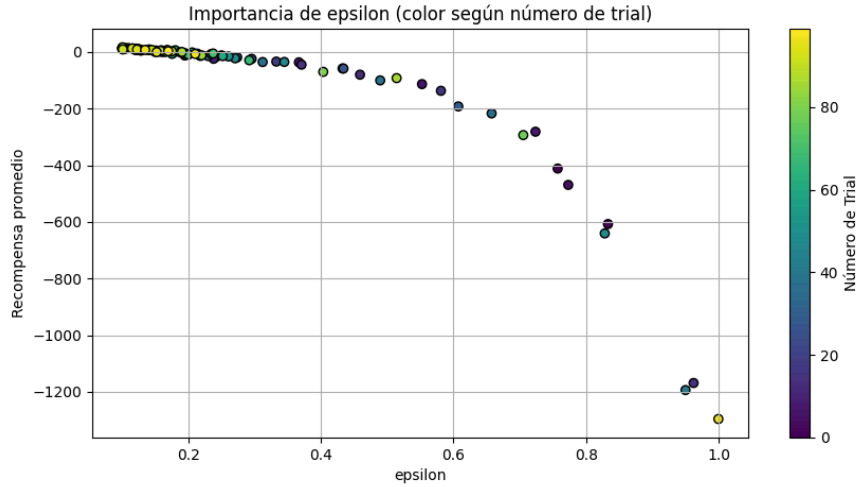
efectivo para encontrar configuraciones estables y eficientes, permitiendo mejorar significativamente la política aprendida respecto a los valores iniciales.

Distribución de hiperparámetros en los mejores resultados

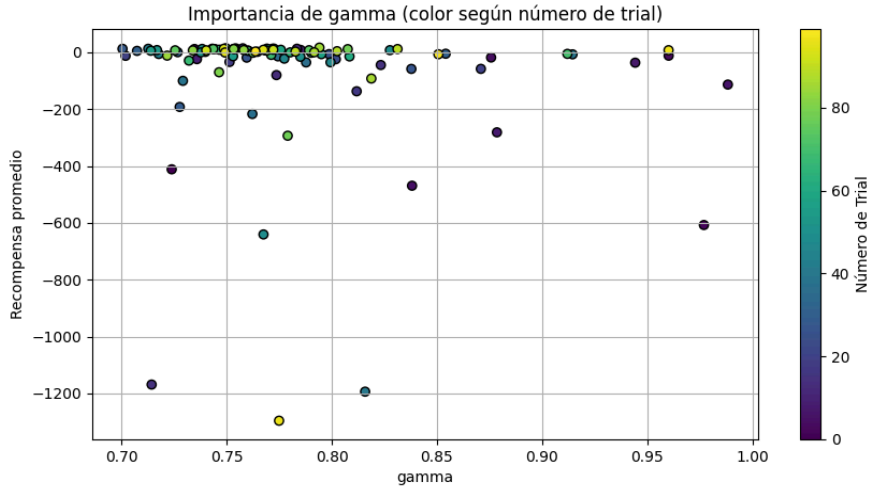
- *alpha*



- *epsilon*



- *gamma*



Los gráficos muestran cómo evolucionaron los valores de los hiperparámetros *alpha*, *epsilon* y *gamma* a lo largo de los trials, reflejando su impacto en el rendimiento del agente.

Finalmente se exportó un archivo .csv (trial_results.csv) con los resultados completos de cada combinación evaluada.

Resultados finales

Q-Table final

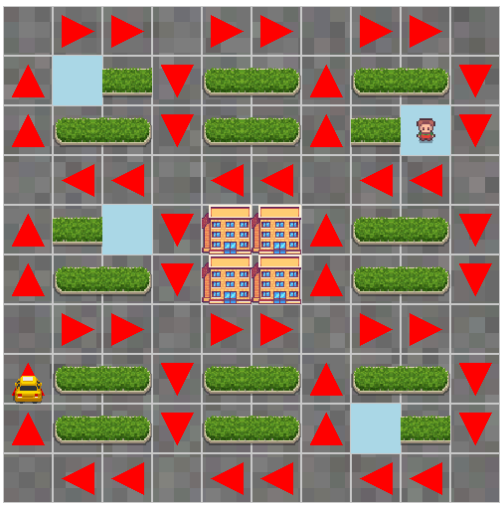
Se guardaron dos versiones de la tabla Q:

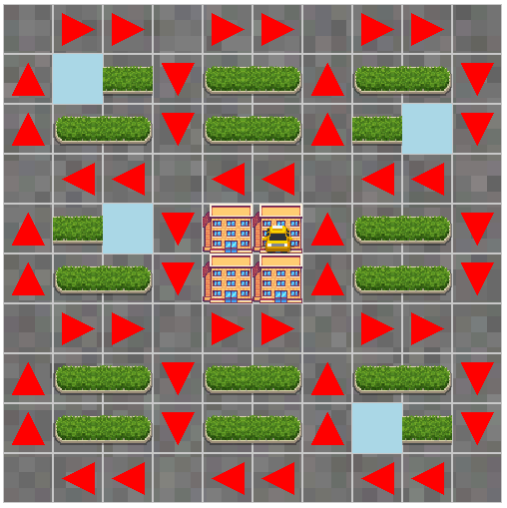
- final_q_table.npy: Q-table resultante del último entrenamiento.
- best_q_table.npy: Q-table correspondiente a la mejor combinación de hiperparámetros encontrada por Optuna.

Estas tablas pueden utilizarse para simular el comportamiento del agente sin necesidad de reentrenar.

Simulación del mejor resultado

La animación (results/best_result/simulacion.gif) muestra una simulación del agente utilizando la mejor política aprendida. En ella, se observa cómo el taxi recoge al pasajero y lo transporta hasta el destino, respetando las direcciones de las calles y minimizando penalizaciones:





Conclusiones

…

En ref se observa que:

- Alpha se mantiene constante y estable en los mejores resultados, lo que sugiere que existe un valor óptimo de tasa de aprendizaje que facilita la convergencia sin provocar oscilaciones en la política.

- Epsilon presenta una tendencia decreciente, lo cual es coherente con la estrategia de exploración-explotación: al inicio se exploran más acciones y, a medida que el agente aprende, se va consolidando una política más explotativa.

- Gamma muestra mayor variabilidad, lo cual indica que el valor óptimo de descuento a futuro depende más de la interacción con otros hiperparámetros y del entorno específico, aunque tiende a estabilizarse en un rango que favorece la planificación a largo plazo.

Esta distribución reafirma la importancia de una correcta selección y ajuste fino de los hiperparámetros para maximizar la eficacia del aprendizaje.