Comparación entre Q-Learning y Deep Q-Learning en FrozenLake

Ing. Ignacio de Pedro Mermier

25 de agosto de 2025

Resumen

En este trabajo se estudia y compara el rendimiento de los algoritmos de **Q-Learning** y **Deep Q-Learning** en el entorno `FrozenLake` de la librería `Gymnasium`. En particular para el caso de **Q-Learning**, se implementa además un mecanismo de *grid search* para la optimización los hiperparámetros. Los resultados serán evaluados bajo distintas métricas que permitirán establecer una comparación entre los mismos.

1. Introducción

Dentro de los métodos de aprendizaje por refuerzo existen los métodos de diferencia temporal. Estos se caracterizan por tener un agente compara la recompensa obtenida mediante una suposición y la recompensa realmente obtenida luego de ejecutar la acción. Entre los algoritmos más representativos se encuentran "Q-Learning", un método basado en tomar la mejor acción siguiente posible según una tabla "Q", y "Deep Q-Learning (DQN)", que emplea redes neuronales para aproximar la función "Q".

En este trabajo, ambos enfoques son comparados en el clásico entorno `FrozenLake`, lo cual permite analizar las ventajas y limitaciones de cada método, así como el impacto de la optimización de hiperparámetros mediante *grid search*.

2. Fundamentos Teóricos

2.1. Q-Learning

Q-Learning es un algoritmo basado en valores, que busca aprender una función $Q(s, a)$ que aproxima el valor esperado de tomar una acción a en un estado s, siguiendo una política óptima.

La actualización de los valores Q se realiza de manera iterativa a partir de la siguiente ecuación:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \Big[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \Big]$$

donde:

- s_t es el estado actual,
- a_t la acción ejecutada,

- r_{t+1} la recompensa recibida,
- s_{t+1} el estado siguiente,
- α la tasa de aprendizaje,
- γ el factor de descuento.

Q-Learning se basa en una *Q-table*, una estructura que almacena los valores de recompensa esperada para cada par estado-acción. Esta tabla se actualiza de manera iterativa mediante el algoritmo, permitiendo elegir en cada estado la acción que maximiza la recompensa esperada.

Si bien Q-Learning funciona adecuadamente en entornos pequeños y discretos, presenta importantes limitaciones en espacios de estado-acción grandes:

- La **Q-table** crece exponencialmente con el tamaño del estado y las acciones, lo que implica altos requerimientos de memoria.
- El tiempo de entrenamiento se vuelve poco práctico a medida que aumenta la dimensionalidad del problema.
- No generaliza entre estados similares: cada estado debe ser aprendido explícitamente.

Por estas razones, Q-Learning tabular no es escalable a problemas más complejos, y se requieren métodos aproximadores de funciones como **Deep Q-Learning**.

2.2. Deep Q-Learning

Deep Q-Learning extiende el algoritmo Q-Learning al reemplazar la Q-table por una red neuronal capaz de aprenden la relación entre estado, acción y recompensa esperada, permitiendo generalizar y abordar problemas complejos y de alta dimensionalidad en el espacio de estados. Las principales características de este método son:

- **Aproximación de la función Q:** se utiliza una red neuronal para estimar los valores $Q(s, a)$, lo que permite manejar espacios de estado mucho más grandes.
- **Replay buffer:** se almacena un conjunto de experiencias pasadas (s, a, r, s') que se reutilizan de manera aleatoria durante el entrenamiento, reduciendo la correlación entre muestras y mejorando la eficiencia.
- **Target networks:** se mantiene una copia de la red principal con parámetros actualizados de forma periódica, lo que proporciona objetivos más estables y evita oscilaciones en el aprendizaje.

2.3. FrozenLake

FrozenLake es un entorno clásico de aprendizaje por refuerzo incluido en la librería `Gymnasium` (anteriormente OpenAI Gym). El escenario consiste en una grilla donde el agente debe desplazarse desde un punto inicial hasta un objetivo, evitando caer en casillas peligrosas que representan agujeros en el hielo. El desafío principal es que las acciones no siempre resultan en el movimiento esperado, ya que el entorno incorpora estocasticidad (el hielo es resbaladizo).

Este problema es útil para experimentar con algoritmos de aprendizaje por refuerzo, ya que combina exploración, manejo de incertidumbre y planificación a largo plazo en un entorno relativamente simple pero no trivial.

3. Q-Learning

3.1. Entorno: FrozenLake 4x4

En este primer experimento se aplicó Q-Learning sobre el entorno *FrozenLake 4x4*. Al finalizar el entrenamiento se evaluaron las métricas principales:

- El tiempo de entrenamiento fue muy bajo, únicamente **1.12 segundos**, debido a que el ambiente es pequeño y cuenta con solo 16 estados.
- La **tasa de éxito alcanzó un 88 %**, lo que muestra que el agente aprendió la política óptima en mucho menos de los 20,000 episodios establecidos.
- El **promedio de pasos por episodio fue bajo**, indicando que el modelo converge rápidamente hacia trayectorias cercanas a la óptima.

Si se analiza la evolución de la tasa de éxito a lo largo del entrenamiento, se observa que a partir de los **7,500 episodios** el valor converge hacia un desempeño cercano al 100 %. Esto refleja que el algoritmo ya ha aprendido correctamente la estructura del mapa, lo cual queda representado en la tabla Q.

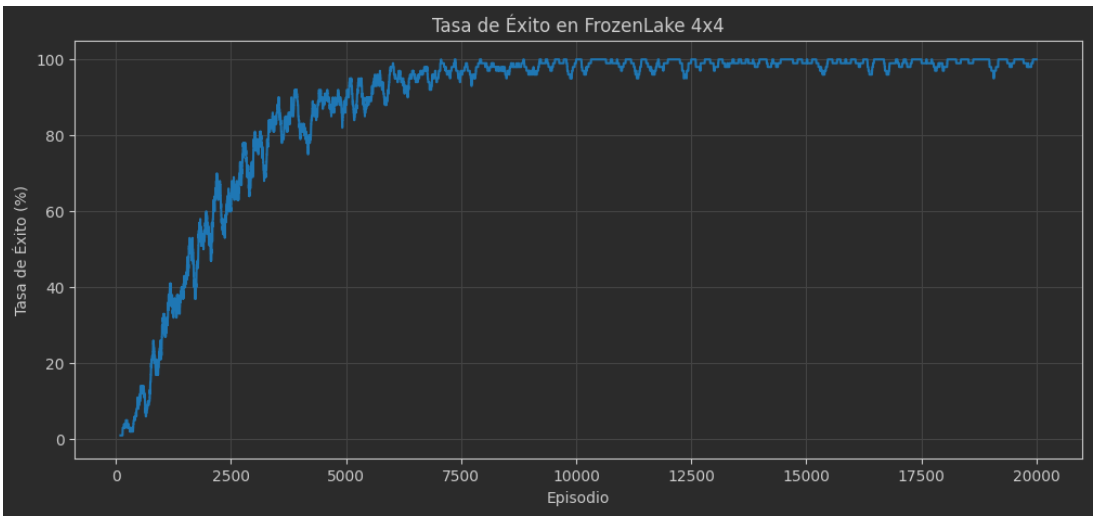

Figura 1: Evolución de la tasa de éxito de Q-Learning en FrozenLake 4x4.

3.2. Entorno: FrozenLake 4x4 con Optimización de Hiperparámetros

Dos de los hiperparámetros más importantes en el caso de **Q-Learning** son:

- **Learning Rate** (α): controla cuánto se actualiza la función Q en cada paso de aprendizaje.

    - Valores muy altos pueden hacer que el agente aprenda de forma inestable.
    - Valores muy bajos pueden ralentizar el aprendizaje.
- **Discount Factor** (γ): determina cuánto peso se le da a las recompensas futuras frente a las inmediatas.
    - Un valor cercano a 0 hace que el agente se enfoque en recompensas inmediatas.
    - Un valor cercano a 1 hace que planifique a más largo plazo.

Dado que no existe un valor óptimo universal para estos parámetros, se empleó la técnica de búsqueda sistemática **Grid Search**, que consiste en evaluar distintas combinaciones de valores y seleccionar aquella que maximiza el desempeño del agente.

Al analizar los resultados obtenidos en el entorno *FrozenLake 4x4*, se observaron los siguientes hallazgos:

- La mejor combinación fue $\alpha = 0{,}01$ y $\gamma = 0{,}9$, alcanzando:
    - **Recompensa promedio en los últimos 100 episodios:** $\approx 0{,}73$
    - **Recompensa acumulada:** $\approx 7{,}02$
- Otras combinaciones también mostraron un rendimiento competitivo, por ejemplo:
    - $\alpha = 0{,}1, \gamma = 0{,}8$ con promedio $\approx 0{,}71$.
    - $\alpha = 0{,}5, \gamma = 0{,}99$ con promedio $\approx 0{,}71$.

Es importante destacar que para estas pruebas se utilizaron únicamente **2500 episodios**, dado que el objetivo era identificar la combinación que converge más rápido hacia un desempeño cercano al 100 %. Si se emplearan muchos más episodios, los últimos 100 tenderían a ser todos exitosos, lo que dificultaría una comparación efectiva entre configuraciones.

La optimización de hiperparámetros permitió mejorar el desempeño del agente Q-Learning, alcanzando una **tasa de éxito cercana al 73 % en los últimos episodios**. Estos resultados ponen de manifiesto la relevancia de ajustar correctamente α y γ, ya que valores inadecuados pueden reducir significativamente la capacidad de aprendizaje del agente.

3.3. Entorno: FrozenLake 8x8

Para analizar el comportamiento de Q-Learning de manera más exhaustiva, se utiliza una expansión del ambiente *FrozenLake 4x4* a *FrozenLake 8x8*, con el objetivo de observar la performance del modelo en un entorno significativamente más grande.

En este caso, como la cantidad de episodios necesarios crece de manera significativa respecto al caso anterior (debido al aumento en el número de estados posibles), el valor de epsilon (ϵ) se obtiene como la inversa de la cantidad de episodios, en lugar de mantenerlo constante como en el caso de *4x4*. Esta estrategia permite realizar pruebas manuales, aumentando gradualmente la cantidad de episodios hasta alcanzar una tasa de éxito razonable, sin necesidad de fijar un valor único de ϵ.

Al aumentar el tamaño del entorno de *FrozenLake* de 4×4 a 8×8, se observa que el aprendizaje del agente se vuelve mucho más lento y desafiante:

- En los primeros 10000 episodios, la tasa de éxito es muy baja (por debajo del 7 %).
- Recién hacia los 20000 episodios el agente alcanza alrededor de un 25 % de éxito, lo que indica que requiere una gran cantidad de interacciones con el entorno para mejorar su desempeño.
- Los pasos promedio por episodio disminuyen progresivamente de aproximadamente 32 a 15, lo que muestra que el agente efectivamente está aprendiendo trayectorias más cortas hacia el objetivo, aunque de manera muy gradual.

Estos resultados reflejan que, al escalar el entorno, el espacio de estados se vuelve mucho más grande y complejo, lo que exige **un mayor número de episodios y un ajuste más cuidadoso de los hiperparámetros** para que el agente logre converger.

3.4. Entorno: FrozenLake 8x8 con Optimización de Hiperparámetros

Al igual que en el caso anterior, se realiza la optimización de los hiperparámetros más importantes:

- **Learning Rate (α)**: controla cuánto se actualiza la función Q en cada paso de aprendizaje.
- **Discount Factor (γ)**: determina cuánto peso se le da a las recompensas futuras frente a las inmediatas.

Tras aplicar *Grid Search* sobre los hiperparámetros α y γ, se obtuvieron los siguientes resultados:

- La mejor combinación encontrada fue $\alpha = 0{,}05$ y $\gamma = 0{,}95$, alcanzando una recompensa promedio en los últimos 100 episodios del 37 %.
- Otras configuraciones también mostraron rendimientos cercanos, como:
  - $\alpha = 0{,}01, \gamma = 0{,}8$ con promedio $\approx 0{,}36$.
  - $\alpha = 0{,}1, \gamma = 0{,}99$ con promedio $\approx 0{,}35$.

La búsqueda sistemática de hiperparámetros confirmó que el desempeño del agente mejora de forma considerable al elegir correctamente α y γ. En el caso de *FrozenLake 8x8*, la mejor combinación fue $\alpha = 0{,}05, \gamma = 0{,}95$, que permitió obtener una recompensa promedio superior en comparación con configuraciones no optimizadas.

Al finalizar se observaron las siguientes métricas:

- Tiempo de aprendizaje: 38.78 segundos, considerablemente mayor debido a los 64 estados del entorno.
- Tasa de éxito: 9 %, lo que indica que el modelo no alcanzó una buena solución en los 200,000 episodios ejecutados.
- Promedio de pasos: relativamente alto, reflejando que el agente aún no converge hacia trayectorias óptimas.

Si graficamos la evolución de la tasa de éxito, se observa que el modelo **podría seguir aprendiendo**, ya que incluso tras 200,000 episodios (10 veces más que en el caso de 4×4), la tasa de éxito recién alcanza valores cercanos al 30 %.

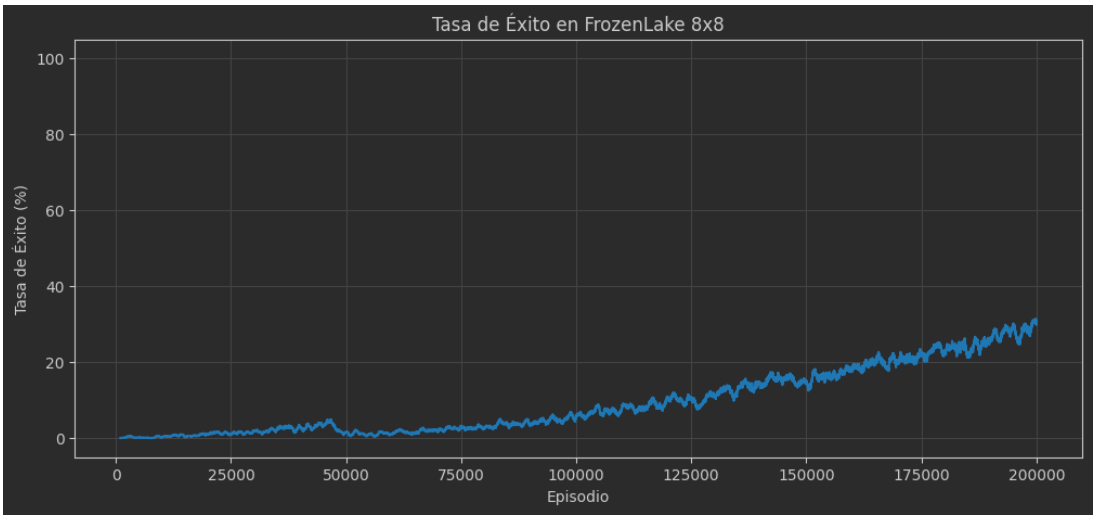

Figura 2: Evolución de la tasa de éxito de Q-Learning en FrozenLake 8x8.

4. Deep Q-Learning (DQN)

4.1. Entorno: FrozenLake 4x4

Al entrenar un agente mediante *Deep Q-Learning* en el entorno *FrozenLake 4x4*, se obtuvieron los siguientes resultados:

- El agente logra una **tasa de éxito muy alta (93 %)**, mostrando que DQN es capaz de aprender una política eficiente en este entorno sencillo.
- El **número promedio de pasos por episodio** fue de 9.6, lo que indica que el agente encontró trayectorias relativamente cortas y consistentes hacia el objetivo.
- Aunque el **tiempo de entrenamiento** fue mayor que en Q-Learning tabular, el desempeño final es claramente superior en términos de **estabilidad y tasa de éxito**.

Al graficar la tasa de éxito del modelo, se observa que a partir de los 4000 episodios la misma converge a un valor cercano al 100 %, lo que indica que el algoritmo ya aprendió correctamente el mapa y lo tiene reflejado en la red neuronal.

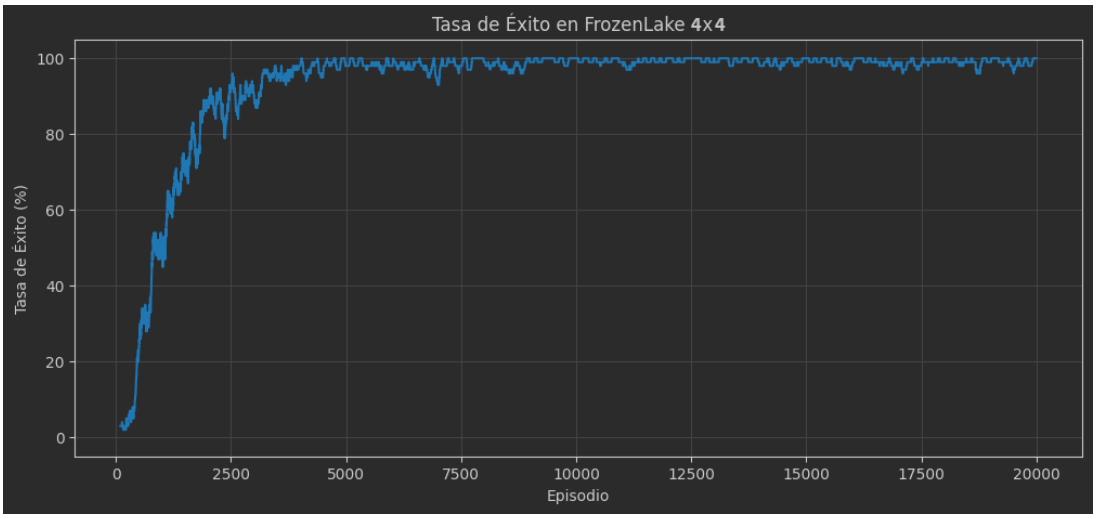

Figura 3: Evolución de la tasa de éxito de Deep Q-Learning en FrozenLake 4x4.

4.2. Entorno: FrozenLake 8x8

Al aplicar *Deep Q-Learning* (DQN) en el entorno *FrozenLake 8x8* y entrenar por solo 5000 episodios (un cuarto de los 20000 episodios usados en el caso de *FrozenLake 4x4*), se obtuvieron los siguientes resultados finales:

- **Tasa de éxito:** 67.02 %
- **Promedio de pasos por episodio:** 2435.1
- **Tiempo total de entrenamiento:** 39.02 segundos

Durante el proceso de aprendizaje, el agente mostró una clara mejora progresiva:

- En los primeros 1000 episodios la tasa de éxito era prácticamente nula, con trayectorias extremadamente largas (más de 10,000 pasos en promedio).
- A partir de los 1250 episodios, el agente comenzó a estabilizar sus trayectorias y alcanzó un 39 % de éxito, con episodios más cortos (∼45 pasos).
- Desde los 2000 episodios en adelante, la tasa de éxito superó el 85 %, y las trayectorias se redujeron a menos de 20 pasos en promedio.
- En las últimas iteraciones (5000 episodios) el agente alcanzó un 98.8 % de éxito, resolviendo consistentemente el entorno con trayectorias óptimas (∼14 pasos).

En comparación con *FrozenLake 4x4*, en *FrozenLake 8x8*, con solo 5,000 episodios (un cuarto de la cantidad usada en 4x4), el agente alcanzó un 98.8 % de éxito en la fase final del entrenamiento.

Esto evidencia que, aunque el entorno más grande es inicialmente más complejo y el agente necesita explorar mucho más, DQN logra escalar exitosamente a entornos de mayor dimensión, algo que no es posible con Q-Learning tabular.

En resumen, DQN no solo resuelve *FrozenLake 4x4* de manera eficiente, sino que también muestra un fuerte potencial de generalización y escalabilidad en el caso de *FrozenLake 8x8*.

Al graficar la tasa de éxito del modelo, se observa que a partir de los 2000 episodios la misma converge a un valor cercano al 100 %, lo que indica que el algoritmo ya aprendió correctamente el mapa y lo tiene reflejado en la red neuronal.

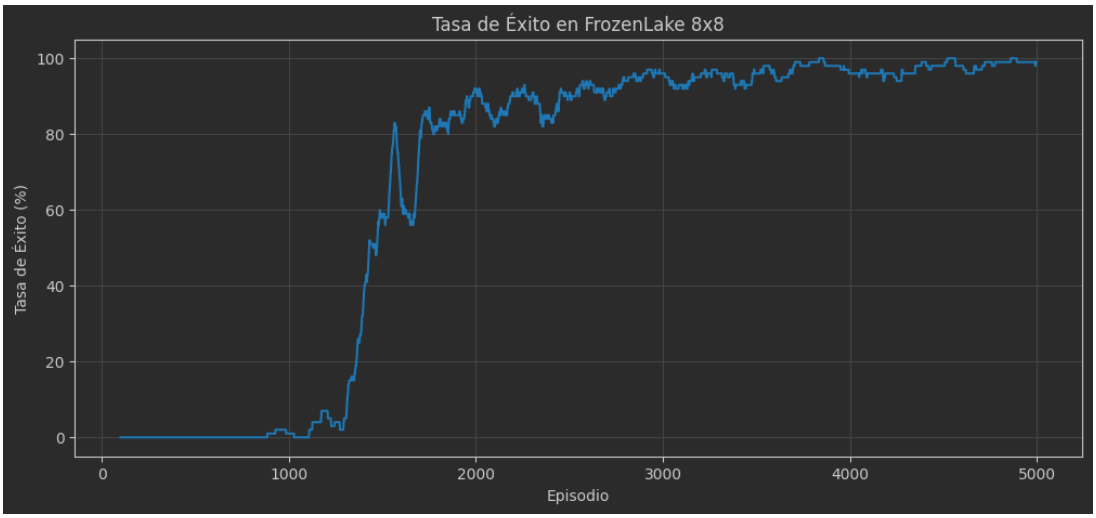

Figura 4: Evolución de la tasa de éxito de Deep Q-Learning en FrozenLake 8x8.

4.3. Comparación de resultados

A continuación se presentan los resultados obtenidos utilizando los métodos *Q-learning* y *Deep Q-learning* sobre los ambientes *FrozenLake 4x4* y *FrozenLake 8x8*:

Método / Entorno	Hiperparámetros	Tasa de éxito final	Episodios hasta convergencia	Tiempo total
Q-Learning 4x4	Sin optimizar ($\alpha = 0{,}1, \gamma = 0{,}9$)	~100 %	7500	1.12 s
Q-Learning 4x4	Optimizado ($\alpha = 0{,}01, \gamma = 0{,}9$)	~100 %	5000	1.09 s
Q-Learning 8x8	Sin optimizar ($\alpha = 0{,}1, \gamma = 0{,}9$)	24.77 %	sin convergencia	36.85 s
Q-Learning 8x8	Optimizado ($\alpha = 0{,}05, \gamma = 0{,}95$)	28.29 %	sin convergencia	38.78 s
DQN 4x4	Sin optimizar ($\alpha = 0{,}001, \gamma = 0{,}9$)	~100 %	3500	176.34 s
DQN 8x8	Sin optimizar ($\alpha = 0{,}001, \gamma = 0{,}9$)	~100 %	4000	39.02 s

Cuadro 1: Comparación de resultados entre Q-Learning y Deep Q-Learning en entornos FrozenLake 4x4 y 8x8.

5. Conclusiones

A partir de los resultados obtenidos en los experimentos realizados con Q-Learning y Deep Q-Learning sobre los entornos *FrozenLake 4x4* y *FrozenLake 8x8*, es posible extraer una serie de conclusiones que resumen las ventajas, limitaciones y diferencias principales entre ambos enfoques:

- **Q-Learning sin optimización:** funciona adecuadamente en entornos pequeños como *FrozenLake 4x4*, alcanzando tasas de éxito aceptables con bajo costo computacional.
- **Optimización de hiperparámetros:** mejora el rendimiento de Q-Learning, aunque en entornos más grandes como *FrozenLake 8x8* sigue siendo insuficiente: la tasa de éxito se mantiene baja y el tiempo de entrenamiento aumenta considerablemente.

- **Deep Q-Learning (DQN):** muestra claras ventajas frente al método tabular:
  - En *FrozenLake 4x4* alcanza una tasa de éxito cercana al 100 %, con trayectorias más consistentes y estables que Q-Learning.
  - En *FrozenLake 8x8* escala mucho mejor, logrando un 98.8 % de éxito con menos episodios que los usados en Q-Learning, evidenciando la capacidad de las redes neuronales para generalizar en espacios de estados más grandes.
- **Resumen:** Q-Learning es adecuado únicamente para entornos pequeños, mientras que DQN, aunque más costoso en términos computacionales, resulta mucho más robusto y escalable en entornos complejos.