Aprendizaje automatico - Laboratorio 2

Bryan Salamone
Universidad de la República
Montevideo, Uruguay
bryan.salamone@fing.edu.uy

Alejandro Orellano
Universidad de la República
Montevideo, Uruguay
alejandro.orellano@fing.edu.uy

Emiliano Gonzalez
Universidad de la República
Montevideo, Uruguay
gonzalez.emiliano@fing.edu.uy

***Abstract*—Este trabajo implementa y compara dos enfoques de aprendizaje por refuerzo [1] —Q-Learning tabular y Deep Q-Network (DQN)— aplicados al entorno Frozen Lake de Gymnasium. Ambos agentes fueron entrenados en la versión no determinista del problema, con el objetivo de aprender una política que maximice la tasa de éxito. Los resultados muestran que, si bien Q-Learning proporciona una solución eficiente y estable, DQN presenta una mayor capacidad de generalización hacia entornos más complejos.**

I. Introducción

Este trabajo presenta el desarrollo de la segunda tarea entregable del curso Aprendizaje Automático. El mismo consiste en la resolución del problema Frozen Lake del entorno Gymnasium mediante dos enfoques de aprendizaje por refuerzo: Q-Learning tabular [2] y Deep Q-Network (DQN) [3]. El objetivo es desarrollar agentes capaces de aprender una política efectiva para atravesar el lago congelado evitando caer en los agujeros, en un entorno no determinista. Se describen los procesos de implementación y calibración de parámetros de ambos métodos, así como los resultados obtenidos en términos de tasa de éxito y estabilidad del aprendizaje. Finalmente, se realiza un contraste entre las dos soluciones, destacando las ventajas del enfoque tabular por su simplicidad y menor costo computacional, frente a la capacidad del modelo DQN para generalizar y representar entornos más complejos.

II. Q-Learning

A. Descripción de la solución implementada para Q-Learning

El agente Q-Learning desarrollado implementa un enfoque *tabular* de aprendizaje por refuerzo. Se modela el entorno de *Frozen Lake* como un proceso de decisión de Markov (MDP), donde el agente aprende una función de valor de acción $Q(s, a)$ que estima la recompensa esperada a largo plazo para cada par estado–acción.

La implementación se basa en una tabla Q de tamaño $(n_{\text{estados}} \times n_{\text{acciones}})$ inicializada en ceros. En cada episodio, el agente recorre el entorno eligiendo acciones según una política ε*-greedy*, que equilibra la exploración y la explotación: con probabilidad ε elige una acción aleatoria y con probabilidad $1 - \varepsilon$ selecciona la acción con mayor valor Q actual.

Tras cada interacción con el entorno, se actualiza el valor Q correspondiente mediante la regla de actualización de Q-Learning:

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

donde:

- α es la tasa de aprendizaje,
- γ es el factor de descuento de futuras recompensas,
- r es la recompensa inmediata obtenida, y
- s' es el nuevo estado alcanzado.

El entrenamiento consiste en repetir este proceso durante 10 000 episodios, permitiendo que la tabla Q converja hacia valores estables. Una vez finalizado, el agente puede jugar explotando su política aprendida, eligiendo siempre la acción con el mayor valor Q para cada estado.

La elección de 10 000 episodios no fue arbitraria: durante las pruebas observamos que a partir de este punto el proceso de aprendizaje deja de mostrar mejoras significativas, ya que la recompensa promedio comienza a oscilar entre 0.5 y 0.6. Consideramos que este comportamiento se debe a la naturaleza no determinista del entorno, donde el deslizamiento inherente (*slippery*) introduce un componente de aleatoriedad que impide alcanzar una política completamente óptima.

B. Calibración de parámetros

El proceso de calibración de los hiperparámetros del agente Q-Learning se realizó de manera empírica, buscando alcanzar un equilibrio entre exploración, convergencia y estabilidad del aprendizaje. El agente fue inicializado con los siguientes valores:

- Tasa de aprendizaje: $\alpha = 0.1$
- Factor de descuento: $\gamma = 0.95$
- Tasa inicial de exploración: $\epsilon = 1.0$
- Tasa mínima de exploración: $\epsilon_{\text{min}} = 0.01$
- Factor de decaimiento de la exploración: $\epsilon_{\text{decay}} = 0.9999$

Durante el entrenamiento, el valor de ϵ se redujo de forma multiplicativa en cada paso del agente, siguiendo la relación $\epsilon \leftarrow \epsilon \cdot \epsilon_{\text{decay}}$, lo que permite un proceso de exploración controlado: al comienzo el agente explora ampliamente el entorno, y gradualmente tiende hacia la explotación de las mejores acciones aprendidas.

En la tabla I se observan los resultados a lo largo del entrenamiento. Se observó un incremento en la recompensa promedio y en la longitud de los episodios, lo que indica una mejora progresiva en la política aprendida. Por ejemplo, hacia el episodio 1 000 la recompensa promedio era de aproximadamente 0.04 con una longitud media de 10 pasos, mientras que hacia el episodio 10 000 la recompensa promedio superó el umbral de 0.60 con una longitud media cercana a 37 pasos.

Este comportamiento confirma que el decaimiento de ϵ fue suficientemente lento como para permitir una exploración efectiva durante las primeras fases del entrenamiento, favoreciendo la convergencia posterior.

En síntesis, los valores seleccionados para los hiperparámetros demostraron ser adecuados para el entorno `FrozenLake-v1`, logrando un aprendizaje estable en un escenario no determinista sin caer prematuramente en la explotación.

C. Evaluación del proceso de aprendizaje y desempeño final

A continuación se presenta la evaluación del agente Q-Learning tras el entrenamiento de 10 000 episodios. Primero se resumen los indicadores observados durante el entrenamiento y, posteriormente, los resultados de varias evaluaciones independientes realizadas en modo de explotación.

a) Resumen del entrenamiento: Durante el entrenamiento se registraron estadísticas por bloques de 1000 episodios.

TABLE I
EVOLUCIÓN DE LA RECOMPENSA Y LONGITUD PROMEDIO POR EPISODIO EN Q-LEARNING 4X4

Episodio	Recompensa promedio	Longitud promedio
1000	0.040	10.0
2000	0.300	25.6
3000	0.611	36.5
4000	0.618	37.3
5000	0.623	35.4
6000	0.599	36.5
7000	0.644	38.8
8000	0.588	38.0
9000	0.670	38.0
10000	0.667	36.8

Como se observa en la tabla I, el agente mejora rápidamente en los primeros miles de episodios (la recompensa promedio crece notablemente entre 1000 y 3000), y que posteriormente la recompensa se estabiliza con oscilaciones alrededor de ≈ 0.60. La longitud promedio de los episodios aumenta conforme la política mejora, lo que indica que los episodios exitosos (los que llegan a la meta) requieren más pasos en promedio debido a la dinámica estocástica del entorno (`is_slippery=True`).

b) Resultados de las evaluaciones: Tras finalizar el entrenamiento se realizaron cinco evaluaciones independientes de 1 000 episodios cada una en modo de explotación. Los resultados se muestran en la tabla II

c) Resumen estadístico: De las cinco evaluaciones la tasa de victorias (en %) tiene las siguientes estadísticas:

- Media de tasa de victorias: $\overline{p} = 68.82\%$.
- Desviación estándar muestral de las tasas: $s \approx 1.18\%$.
- Rango observado: 67.1% – 70.3% .

Las desviaciones estándar reportadas en cada evaluación (aproximadamente 0.45–0.47) corresponden a la desviación de la variable binaria "éxito en el episodio" y son coherentes con las tasas observadas (para $p \approx 0.7$ la desviación teórica de una Bernoulli es $\sqrt{p(1-p)} \approx 0.458$).

D. Análisis de la política obtenida

Tras el entrenamiento, se analizó la política aprendida por el agente Q-Learning en el entorno `FrozenLake-v1` con `is_slippery=True`. La política óptima aprendida se puede representar como la siguiente tabla de acciones por estado:

S	$\uparrow$	$\downarrow$	$\uparrow$
$\leftarrow$	H	$\rightarrow$	H
$\uparrow$	$\downarrow$	$\leftarrow$	H
H	$\rightarrow$	$\uparrow$	G

a) Leyenda:

- S = Estado inicial
- G = Meta
- H = Hoyo
- $\uparrow$ = Mover arriba
- $\downarrow$ = Mover abajo
- $\leftarrow$ = Mover izquierda
- $\rightarrow$ = Mover derecha

b) Interpretación de la política:

1) El agente ha aprendido a evitar los hoyos (H) en la medida de lo posible, eligiendo acciones que incrementan la probabilidad de llegar a la meta (G).
2) Observando la política, se evidencia un comportamiento conservador en los estados cercanos a H, priorizando movimientos seguros.
3) La política no siempre sigue el camino más corto posible debido a la naturaleza estocastica del entorno (`is_slippery=True`), que obliga al agente a considerar rutas alternativas para minimizar el riesgo de caer en un hoyo.
4) La secuencia de acciones desde el estado inicial (S) hacia la meta (G) refleja la adaptación del agente a la dinámica probabilística del entorno, lo que explica la longitud promedio de episodios observada durante el entrenamiento (aproximadamente 38 pasos).

c) Conclusión: La política obtenida muestra que el agente Q-Learning es capaz de identificar rutas con mayor probabilidad de éxito frente a la incertidumbre del entorno. Aunque no garantiza la trayectoria más corta debido al resbalamiento, la política maximiza la recompensa esperada considerando la probabilidad de resbalar en cada acción.

d) Interpretación:

1) **Convergencia y estabilidad:** El agente muestra una convergencia clara durante los primeros 3 000 episodios y luego se estabiliza alrededor de una tasa de éxito cercana al 0.60–0.70 en evaluaciones independientes. La variabilidad entre evaluaciones es baja (desviación de 2.25 puntos porcentuales), lo que indica comportamiento consistente de la política aprendida.
2) **Impacto del no determinismo:** El entorno `is_slippery=True` introduce resbalones (probabilidad de acción tomada $1/3$), por lo que es esperable que la longitud media de episodios exitosos sea mucho mayor que el mínimo determinista (6 pasos).

TABLE II
RESULTADOS DE LAS EVALUACIONES DEL AGENTE Q 4X4

Evaluación	Tasa de victorias (%)	Recompensa promedio	Desviación estándar	Victorias
1	67.1	0.671	0.470	671
2	69.2	0.692	0.462	692
3	70.3	0.703	0.457	703
4	69.1	0.691	0.462	691
5	68.4	0.684	0.465	684

La longitud promedio observada (aproximadamente 33.4) es coherente con la trayectoria de aprendizaje en un entorno con transiciones ruidosas.

3) **Relación éxito / longitud:** Dado que la recompensa es 1 únicamente al alcanzar la meta, una recompensa promedio 0.60 implica que alrededor del 60% de los episodios alcanzan la meta. La longitud promedio creciente indica que los episodios exitosos requieren varias correcciones por resbalones; esto no es necesariamente indicador de un mal algoritmo, sino de la dificultad impuesta por la dinámica estocástica.

e) Conclusión: El agente Q-Learning entrenado durante 10 000 episodios alcanza un desempeño consistente y robusto frente al no determinismo del entorno, con una tasa de éxito promedio de aproximadamente 68.82% en evaluaciones independientes. La longitud media de los episodios es elevada debido a la dinámica de resbalamiento, por lo que la comparación con el óptimo del proceso de decisión de Markov es necesaria para decidir si el desempeño es cercano al óptimo o si hay margen de mejora.

III. DEEP Q-NETWORK

A. Descripción de la implementación

El segundo agente implementado utiliza Deep Q-Network (DQN), un algoritmo que combina Q-Learning con redes neuronales profundas para aproximar la función $Q(s, a)$ en lugar de usar una tabla. En esencia, DQN mantiene una red neuronal parametrizada $Q(s, a)$ cuyos pesos θ se entrenan para minimizar el error de Bellman: la red toma como entrada una representación del estado s y estima los valores Q para cada acción a. El entrenamiento ajusta θ de modo que se cumpla aproximadamente $Q(s, a) \approx r + \gamma \max_{a'} Q(s', a')$ para cada transición (s, a, r, s') observada, imitando la actualización tabular.

B. Arquitectura de la red

En nuestro agente se empleó una red fully-connected sencilla con dos capas ocultas. Los estados del entorno se representan mediante codificación one-hot, es decir, cada estado discreto se transforma en un vector binario de 16 componentes (para el problema Frozen Lake 4x4), donde únicamente la posición correspondiente al estado actual toma el valor 1 y las demás son 0. Esta representación evita introducir relaciones ordinales artificiales entre estados y permite que la red aprenda directamente sobre el espacio discreto de estados.

La primera capa oculta tiene 32 neuronas con activación ReLU, y la segunda capa oculta también 32 neuronas ReLU. Finalmente, la capa de salida es lineal con 4 neuronas (correspondientes a las 4 acciones posibles), produciendo $Q(s, a)$ para $a = 0, 1, 2, 3$. En notación compacta, la red es de tamaño $16 \to 32 \to 32 \to 4$.

Esta topología (32 neuronas ocultas) se eligió con el fin de no sobredimensionar demasiado la red para un problema con solo 16 estados discretos. De hecho, se experimentó usando 128 neuronas por capa y se notó que ralentiza el aprendizaje y que induce oscilaciones por sobrecapacidad. Con 32 neuronas, la red tiene suficiente capacidad para aproximar la función Q óptima sin incurrir en sobreajuste.

C. Mecanismos de estabilidad

Se implementó una estructura **ReplayBuffer** para almacenar experiencias recientes $(s, a, r, s', \text{done})$. Su capacidad, definida por el hiperparámetro *buffer_size* del agente DQN, determina la cantidad máxima de transiciones que pueden conservarse en memoria antes de sobrescribir las más antiguas. En cada paso de entrenamiento, en lugar de actualizar la red inmediatamente con la última experiencia observada, las transiciones se acumulan en el buffer y se entrenan en mini-lotes aleatorios de tamaño *batch_size* muestreados del mismo. De esta forma, el aprendizaje se desacopla temporalmente de la secuencia de estados del entorno, reduciendo la correlación entre muestras consecutivas y mejorando la estabilidad del entrenamiento.

Se mantienen dos redes neuronales con la misma arquitectura: la red principal Q_θ que se entrena directamente, y una red objetivo Q_{θ^-} que se utiliza para calcular los valores objetivo $\left(r + \gamma \max_{a'} Q_{\theta^-}(s', a')\right)$ en la ecuación de Bellman. La red objetivo se sincroniza periódicamente con la principal, copiando sus pesos cada *target_update* pasos. Durante esos pasos, Q_{θ^-} permanece fijo, proporcionando un objetivo relativamente estable para que Q_θ se acerque. Cada *target_update* iteraciones, $\theta^- \leftarrow \theta$, actualizando la red objetivo. Este procedimiento de hard update evita la retroalimentación instantánea de la red consigo misma y reduce las oscilaciones.

D. Algoritmo de optimización

Además de estos mecanismos principales, se utilizó el optimizador **Adam** [4] con tasa de aprendizaje inicial ajustable por el parametro *lr* para entrenar la red. Adam se escogió por su efectividad en ajustar parámetros de redes profundas con ruido en gradientes. Se probó inicialmente un lr del orden de 10^{-3} y se observó cierta inestabilidad (las recompensas

fluctuaban ampliamente), por lo que se redujo a 5×10^{-4} para tener pasos de gradiente más pequeños y estables. Con este valor, la red tardó más en aprender pero evitó grandes oscilaciones en los valores Q.

E. Calibracion de parámetros

Por otro lado, los demás hiperparámetros del agente DQN se fijaron en:

- Tasa de aprendizaje: $lr = 5 \times 10^{-4}$
- Factor de descuento: $\gamma = 0.99$
- Tamaño de lotes: $batch\_size = 32$
- Tamaño del replay buffer: $buffer\_size = 10000$
- Frecuencia de actualización de la red objetivo: $target\_update = 100$
- Tasa inicial de exploración: $\epsilon = 1.0$
- Tasa mínima de exploración: $\epsilon_{\text{min}} = 0.01$
- Factor de decaimiento de la exploración: $\epsilon_{\text{decay}} = 0.9999$

F. Desempeño y convergencia del agente DQN

La tabla III muestra la evolución de la recompensa promedio por episodio para el agente DQN entrenado durante 10.000 episodios (promediada en ventana de 1000 episodios, similar a Q-Learning). En los primeros 1000 episodios, la recompensa promedio fue cercana a cero y comenzó a aumentar gradualmente, alcanzando alrededor de 0.4 hacia el episodio 2000. Entre los episodios 3000 y 5000 se observaron oscilaciones, y recién después del episodio 6000 la curva se estabilizó alrededor de 0.65. Aunque el desempeño final fue aceptable, la variabilidad entre ejecuciones fue mayor que en el método tabular, con tasas de éxito que fluctuaron entre 52% y 67% según la semilla utilizada por el agente.

En términos de velocidad de aprendizaje, tanto Q-Learning como DQN alcanzaron el umbral del 50% de éxito en los 3000 episodios. Si bien DQN posee capacidad de generalización, en un entorno pequeño como FrozenLake 4x4 esa ventaja no se manifiesta claramente. Finalmente, aunque la función de valor Q_θ no converge completamente, la política resultante tiende a estabilizarse hacia una estrategia efectiva. En la tabla IV podemos observar las tasas de éxito obtenidas, mayores al 60%, que indican que el agente fue capaz de aprender una política viable sin memorizar estados, sino generalizando desde la experiencia acumulada.

TABLE III
EVOLUCIÓN DE LA RECOMPENSA Y LONGITUD PROMEDIO DURANTE EL ENTRENAMIENTO DQN 4X4

Episodio	Recompensa Promedio	Longitud Promedio
1000	0.052	11.3
2000	0.383	33.1
3000	0.658	43.7
4000	0.632	42.3
5000	0.666	43.1
6000	0.647	43.0
7000	0.673	42.8
8000	0.654	43.1
9000	0.672	41.2
10000	0.652	44.7

TABLE IV
RESULTADOS DE LAS EVALUACIONES DEL AGENTE DQN 4X4

Evaluación	Tasa de victorias (%)	Recompensa promedio	Desviación estándar
1	70.2	0.702	0.457
2	72.4	0.724	0.447
3	65.7	0.657	0.475
4	68.2	0.682	0.466
5	66.6	0.666	0.472

G. Análisis de la política aprendida

En la siguiente tabla se muestra la política aprendida por el agente DQN:

S	↑	↑	↑
←	H	→	H
↑	↓	←	H
H	→	→	G

La politica coincidió en gran medida con la obtenida por Q-Learning en los estados clave del entorno, especialmente en lo que refiere a evitar hoyos y trazar rutas seguras hacia la meta. Aunque se observaron algunas diferencias puntuales en decisiones individuales —producto de la generalización que introduce la red neuronal— estas no afectaron significativamente el desempeño. En varios estados, DQN eligió acciones conservadoras distintas a las de la tabla, probablemente debido a cómo propaga valor entre estados similares, pero mantuvo una tasa de éxito comparable.

IV. RESOLUCIÓN DEL ENTORNO FROZENLAKE 8×8

En la versión con *Q-Learning* tabular, el entorno *FrozenLake 8x8* introduce una mayor complejidad debido al incremento del espacio de estados (64 posibles posiciones) y a la mayor probabilidad de caer en estados terminales no deseados. Se utilizó una tabla $Q(s, a)$ de dimensión 64×4, inicializada en cero y actualizada siguiendo la regla de Bellman con un factor de descuento $\gamma = 0.99$, tasa de aprendizaje $\alpha = 0.1$ y política ϵ-greedy con decaimiento exponencial de ϵ desde 1.0 hasta 0.01. Para estabilizar el aprendizaje en un entorno estocástico, se aumentó el número de episodios de entrenamiento a $50{,}000$ y se aplicó una política de exploración más lenta, permitiendo al agente descubrir caminos seguros hacia el objetivo. El resultado fue un incremento gradual en la tasa de éxito, aunque con oscilaciones típicas del aprendizaje tabular en espacios de gran tamaño.

Por otro lado, el agente *Deep Q-Network (DQN)* resuelve el mismo entorno empleando una red neuronal como aproximador de la función $Q(s, a)$. Los estados se codifican mediante *one-hot encoding* de dimensión 64, y la red utilizada tiene una arquitectura ampliada de $64 \to 128 \to 128 \to 4$ con activaciones ReLU en las capas ocultas. Este aumento en la capacidad del modelo permite capturar mejor las transiciones del entorno y suavizar el aprendizaje respecto al enfoque tabular. Se mantuvo un factor de descuento $\gamma = 0.99$, tasa de aprendizaje del optimizador Adam $\alpha = 0.001$, tamaño de lote 64 y *replay buffer* de $10{,}000$ transiciones. El uso de *experience replay* y de una red objetivo actualizada cada 100 pasos proporcionó mayor estabilidad y permitió al agente alcanzar

políticas óptimas de forma más consistente, reduciendo la desviación en las recompensas obtenidas durante la evaluación. En la tabla V se contrastan los resultados de evaluacion de ambos agentes, tanto el Q tabular como DQN, para el problema Frozen Lake 8x8.

TABLE V
COMPARARTIVA DE RESULTADOS DE LAS EVALUACIONES DE LOS AGENTES PARA FROZEN LAKE 8X8

Evaluación	Agente	Tasa de victorias (%)	Recompensa promedio	Desviación estándar
1	Q	53.3	0.533	0.499
	DQN	56.1	0.561	0.496
2	Q	49.5	0.495	0.500
	DQN	51.9	0.519	0.500
3	Q	51.2	0.512	0.500
	DQN	49.2	0.492	0.500
4	Q	51.5	0.515	0.500
	DQN	56.8	0.568	0.495
5	Q	50.2	0.502	0.500
	DQN	52.3	0.523	0.499

V. CONCLUSION

Ambos agentes desarrollados, Q-Learning tabular y Deep Q-Network, lograron superar el umbral del 50% de éxito en el entorno FrozenLake 4x4 y tambien el 8x8 ajustando hiper-parametros, cumpliendo con los objetivos del laboratorio. Q-Learning se destacó por su simplicidad, estabilidad y rapidez de convergencia, alcanzando tasas cercanas al 70% con poca variabilidad entre ejecuciones. DQN, en cambio, mostró un desempeño más sensible a la inicialización y a los hiperparámetros, logrando en promedio un 68% de éxito pero con mayor desviación. Su proceso de aprendizaje fue más complejo y costoso computacionalmente, aunque demostró capacidad de generalización y flexibilidad frente a problemas de mayor dimensión.

La elección entre ambos métodos depende del dominio del problema. Q-Learning es ideal para entornos discretos y pequeños. DQN, por su parte, permite escalar a espacios grandes o continuos, incorporando representaciones complejas a través de redes neuronales, aunque requiere mayor calibración o fine-tuning. En este entorno puntual, el método tabular resultó más eficiente, pero DQN es mas escalable en escenarios más complejos.

REFERENCES

[1] Presentación Aprendizaje por refuerzo. Disponible en: https://eva.fing.edu.uy/pluginfile.php/292856/mod_resource/content/6/11-refuerzos.pdf.

[2] T. M. Mitchell, *Machine Learning*, Chapter 13: Reinforcement Learning. New York: McGraw-Hill, 1997.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, "Playing Atari with Deep Reinforcement Learning", arXiv preprint arXiv:1312.5602, 19 Dec 2013. Disponible en: https://arxiv.org/pdf/1312.5602

[4] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, Chapter 8: "Optimization for Training Deep Models", Section 8.5.3 (Adam). Cambridge, MA: MIT Press, 2016.