Aprendizaje por refuerzo: BlackJack

José Carlos Riego, Pablo Rodríguez y Ángel Visedo

Machine Learning II - Máster Universitario en Big Data

Mayo 2025

Abstract

Este estudio explora la aplicación de distintos métodos de aprendizaje por refuerzo en el juego *Blackjack*. Así, a partir del entorno de la librería *Gymnasium*, se comparan tres métodos: Double Q-Learning, Double Deep Q-Network (DQN) y A2C, analizándose así la capacidad de cada uno de ellos de superar los resultados promedio obtenidos por un jugador humano, bajo la asunción de una baraja infinita (con reemplazo).

1. Introducción

1.1. Fundamentos del Blackjack

El *Blackjack* representa un dominio clásico en la teoría de decisiones estocásticas y constituye un juego de cartas ampliamente estudiado en casinos. Su objetivo fundamental consiste en obtener una mano cuyo valor se aproxime a 21 sin excederlo, compitiendo directamente contra el crupier. La dinámica del juego se inicia con la distribución de dos cartas al jugador y dos al crupier, de las cuales solo una de las cartas del crupier es revelada inicialmente.

La valoración de las cartas sigue un sistema específico: las cartas numéricas (2-10) mantienen su valor nominal, las figuras (J, Q, K) equivalen a 10 puntos, y el as presenta una dualidad valorativa (1 u 11 puntos) que se determina según la configuración más favorable para el poseedor. Durante su turno, el jugador debe elegir entre dos acciones fundamentales: solicitar cartas adicionales (*hit*) o finalizar su turno con la mano actual (*stand*). Esta decisión debe tomarse considerando exclusivamente la información parcial disponible: su propia mano y la carta visible del crupier. La condición de derrota automática (*bust*) se produce cuando el valor acumulado de las cartas del jugador supera 21.

El turno del crupier está sujeto a reglas deterministas: debe solicitar cartas hasta alcanzar un umbral mínimo preestablecido (típicamente 17 puntos). La resolución del juego se produce mediante la comparación de los valores finales de ambas manos, proclamándose vencedor aquel participante cuyo valor se aproxime más a 21 sin rebasarlo.

1.2. Motivación y objetivos: Blackjack como entorno para algoritmos avanzados de RL

El *Blackjack* representa un escenario óptimo para evaluar la eficacia de algoritmos modernos de aprendizaje por refuerzo debido a su naturaleza probabilística y su estructura de recompensas bien definida. Si bien el juego implica incertidumbre respecto a las cartas futuras, es importante destacar que bajo ciertas condiciones (como la ausencia de barajado frecuente), técnicas como el conteo de cartas pueden reducir esta incertidumbre—aspecto que, aunque relevante en entornos reales, queda fuera del alcance de nuestra implementación basada en el entorno de *Gymnasium*, donde se asume una baraja infinita con reemplazo.

Nuestros objetivos específicos comprenden:

- Implementar y comparar tres algoritmos avanzados de RL: Double Q-Learning (una variante que reduce la sobreestimación del Q-Learning tradicional), Double Deep Q-Network (que incorpora redes neuronales para aproximar la función de valor) y Advantage Actor-Critic (A2C, que separa la estimación de valor y la selección de acciones)
- Evaluar el rendimiento de estos algoritmos en términos de retorno acumulado y tasa de victorias contra el crupier
- Analizar la velocidad de convergencia y la estabilidad durante el entrenamiento de cada algoritmo
- Visualizar y comparar las políticas aprendidas por cada método para diferentes estados del juego
- Identificar las ventajas y limitaciones de cada enfoque en el contexto específico del Blackjack con baraja infinita

Este trabajo no solo contribuye al entendimiento de cómo diferentes arquitecturas de RL se comportan en un entorno de decisión probabilístico como el Blackjack, sino que también examina el potencial de estos métodos para superar el rendimiento humano promedio en un juego de casino ampliamente conocido.

1.3. Ventajas de los algoritmos avanzados de RL en el contexto del Blackjack

Las estrategias tradicionales de Blackjack, como la *estrategia básica* desarrollada por Baldwin et al.[1] y refinada por Thorp[2], se fundamentan en exhaustivos cálculos probabilísticos y análisis combinatorio. Estas aproximaciones determinan la acción óptima (pedir carta o plantarse) para cada combinación posible entre la mano del jugador y la carta visible del crupier, generando tablas de decisión predefinidas.

Los algoritmos modernos de RL como Double Q-Learning, Double DQN y A2C que implementamos en este trabajo presentan ventajas significativas respecto a estas estrategias convencionales:

- **Optimización sin supervisión**: Estos algoritmos descubren políticas efectivas mediante interacción directa con el entorno, sin requerir programación explícita de reglas heurísticas o cálculos probabilísticos previos.
- **Manejo de la varianza**: Particularmente, Double Q-Learning y Double DQN reducen el problema de sobreestimación presente en algoritmos más básicos, permitiendo una convergencia más estable hacia políticas óptimas.
- **Capacidad de aproximación**: El enfoque DQN utiliza redes neuronales para aproximar funciones de valor en espacios de estados continuos o de alta dimensionalidad, superando las limitaciones de representación tabular.
- **Descomposición valor-política**: El algoritmo A2C separa la estimación del valor (critic) de la selección de acciones (actor), permitiendo un balance más refinado entre exploración y explotación que las aproximaciones tradicionales.

Además, nuestra implementación demuestra que estos métodos avanzados de RL pueden superar el rendimiento de las estrategias básicas tradicionales, especialmente en escenarios con baraja infinita donde el conteo de cartas no tiene efectividad.

2. Entorno Blackjack de Gymnasium

2.1. Definición formal del MDP

El entorno de Blackjack en Gymnasium se puede formalizar como un Proceso de Decisión de Markov (MDP), definido por la tupla (S, A, P, R, γ):

- S: espacio de estados
- A: conjunto de acciones
- P: función de transición de probabilidad
- R: función de recompensa
- γ: factor de descuento

Espacio de estados (S)

El estado del juego se representa mediante una tupla de tres elementos:

$$s = (\text{player\_sum}, \text{dealer\_card}, \text{usable\_ace})$$

Donde:

- **player_sum**: Suma de las cartas del jugador (entero entre 4 y 21).
- **dealer_card**: Valor de la carta visible del crupier (entero entre 1 y 10, donde 1 representa un As).
- **usable_ace**: Booleano que indica si el jugador tiene un As que puede contar como 11 sin pasarse de 21 (*True* o *False*).

Esta representación simplifica el juego real, ya que solo considera la suma total de las cartas del jugador, no su composición exacta. Por ejemplo, las manos [7, 9] y [6, 10] se representan con el mismo estado (16, dealer_card, False).

El número total de estados posibles es:

$$|S| = 18 \times 10 \times 2 = 360$$

Correspondientes a 18 posibles sumas del jugador (4-21), 10 posibles cartas visibles del crupier (As-10), y 2 posibilidades para el As utilizable.

Espacio de acciones (A)

Las acciones disponibles son:

$$A = \{\text{stick}, \text{hit}\}$$

Donde:

- **stick** (0): El jugador se queda con su mano actual.
- **hit** (1): El jugador pide una carta adicional.

Función de transición (P)

La función $P(s'|s, a)$ define la probabilidad de llegar al estado s' al tomar la acción a desde el estado s. En el Blackjack, esta transición es estocástica debido a la naturaleza aleatoria de las cartas.

Por ejemplo, si el jugador pide carta (hit), la probabilidad de transición dependerá de la carta recibida, que se asume sigue una distribución uniforme (considerando una baraja infinita).

La función de transición se puede formular como:

$$P(s'|s, a) = \begin{cases} P(\text{player\_sum}'|\text{player\_sum}, a) & \text{si } a = \text{hit} \\ P(\text{terminal}|\text{player\_sum}, \text{dealer\_card}) & \text{si } a = \text{stick} \end{cases}$$

Función de recompensa (R)

La función $R(s, a, s')$ define el valor de tomar la acción a en el estado s y llegar al estado s'. En el Blackjack, las recompensas se dan al final del episodio:

$$R(s, a, s') = \begin{cases} +1 & \text{si el jugador gana} \\ 0 & \text{si hay empate} \\ -1 & \text{si el jugador pierde} \end{cases}$$

En nuestra configuración ('natural=False'), no se considera la recompensa especial de 1.5 por obtener un Blackjack natural.

Factor de descuento (γ)

Se elige $\gamma = 1$ para que la única recompensa del episodio, r_T, se valore en su totalidad. Dado un episodio de longitud T con recompensas r_t que son cero salvo al final ($r_T \in \{-1, 0, 1\}$), el retorno total queda

$$R = \sum_{t=0}^{T} \gamma^t \, r_t \;=\; \gamma^T \, r_T.$$

Con $\gamma = 1$:

$$R = r_T,$$

La razón detrás de esto es que si eligiéramos $\gamma < 1$:

$$R = \gamma^T \, r_T < r_T,$$

lo cual introduciría:

- **Sesgo hacia episodios cortos**: las políticas que terminan con menor T sufren un descuento menor y parecen más valiosas.
- **Valores pequeños y aprendizaje lento**: al tener $\gamma^T \ll 1$, las actualizaciones TD o de gradiente son muy pequeñas, ralentizando la convergencia.

Estados terminales

Los episodios terminan cuando:

- El jugador se pasa de 21 (pierde automáticamente).
- El jugador se planta, lo que activa el turno del crupier.

Cuando el jugador se planta, el crupier sigue una regla fija: pide cartas hasta tener 17 o más. El resultado se determina comparando las sumas finales, teniendo en cuenta si alguno se pasó de 21.

2.2. Hipótesis y simplificaciones adoptadas

El entorno de Blackjack de Gymnasium hace varias simplificaciones respecto al juego real de casino:

- **Baraja infinita**: Se asume que las cartas se toman de una baraja infinita, por lo que cada carta tiene la misma probabilidad de ser seleccionada independientemente de las cartas ya vistas. Esto elimina la posibilidad de aplicar técnicas de conteo de cartas.
- **Sin apuestas variables**: No se modelan las apuestas, reduciendo el juego a ganar (+1), perder (-1) o empatar (0), sin considerar estrategias de gestión de capital.
- **Opciones limitadas**: Solo se consideran las acciones básicas (pedir carta o plantarse), omitiendo opciones como doblar, separar o asegurar.
- **Estado simplificado**: Como mencionamos antes, el estado solo considera la suma de las cartas, no su composición exacta.
- **Sin reglas de natural** (`natural=False`): No se aplica la regla especial para el Blackjack natural (21 con las dos primeras cartas).
- **Sin reglas específicas de Sutton & Barto** (`sab=False`): Usamos una aproximación más cercana a las reglas de casino que las simplificaciones propuestas en su libro[6].

2.3. Resultados de referencia

Antes de analizar los resultados obtenidos por nuestros modelos de RL, es importante contar con un marco de referencia claro que permita evaluar adecuadamente su rendimiento. Para ello, presentamos los resultados promedio que obtendría un jugador humano siguiendo la *estrategia básica perfecta*, la cual decide únicamente a partir de la mano del jugador y la carta visible del crupier, sin utilizar memoria ni conteo de cartas (*memory-less*).

Es importante destacar que todos nuestros métodos de RL son *model-free*; es decir, aprenden directamente la política óptima mediante ensayo y error, sin construir una representación explícita del entorno. Por este motivo, no comparamos nuestros resultados con estrategias basadas en conteo de cartas, ya que éstas equivaldrían a métodos *model-based*, en los cuales se utiliza información histórica para inferir probabilidades futuras.

Además, como ya hemos comentado, nuestro entorno asume una baraja infinita, por lo que carece completamente de sentido aplicar estrategias de conteo en este contexto.

Así, bajo las reglas habituales de casino (6–8 mazos, crupier planta en soft-17, pago 3:2 al *natural*, permitiendo dobles y splits), esta estrategia obtiene en promedio los siguientes resultados*:

*Datos simulados sobre 10^8 manos por Shackleford; véase[7].

Resultado	Media	Desv. típica
Victorias	42.22 %	0.05 %
Empates	8.48 %	0.03 %
Derrotas	49.10 %	0.06 %
Retorno	−0,005	−0,001

Donde el hecho de obtener un retorno negativo implica que la banca gana al jugador. Concretamente, un retorno de −0,005 significa que, porcentualmente, la banca se llevaría un 5 % de lo apostado por el jugador. Por tanto, como podemos ver, **sin efectuar una estrategia de conteo de cartas, en promedio, el jugador está destinado a perder**.

Además, recordemos que nuestro entorno `Blackjack-v1` no permite las acciones *doblar* y *dividir*; y (paga los *blackjacks naturales* a razón de 1:1 en lugar de 3:2. Ambas modificaciones incrementan ligeramente la ventaja del casino, empeorando el retorno esperado del jugador, con lo cual superar los resultados de referencia mencionados es aún más meritorio, dado este extra de complejidad.

Así, por lo expuesto, en todo el estudio compararemos el desempeño de nuestros agentes con con la estrategia básica humana *memory-less* (sin conteo de cartas).

3. Entrenamiento y evaluación de los modelos

Una vez inspeccionado el entorno y definidas claramente las reglas del juego Blackjack a utilizar, hemos optado por entrenar cuatro modelos distintos: **Q-Learning**, **Double Q-Learning**, **Double Deep Q-Network (DQN)** y **Advantage Actor-Critic (A2C)**.

La elección de estos métodos radica en que, al cubrir distintos enfoques de aprendizaje por refuerzo, nos permiten evaluar exhaustivamente las fortalezas y debilidades inherentes a cada aproximación:

- **Q-learning**: método clásico *value-based* y *off-policy* sencillo y propenso a la sobreestimación de valores.
- **Double Q-learning**: variante del Q-learning que reduce el sesgo positivo usando dos estimadores independientes, aportando mayor estabilidad.
- **Double Deep Q-Network (DQN)**: extensión del Q-learning con aprendizaje profundo, permitiendo manejar espacios de estado más complejos mediante funciones no lineales.
- **Advantage Actor-Critic (A2C)**: enfoque híbrido *Actor-Critic* y *on-policy*, que separa explícitamente política y valor, facilitando una exploración eficiente.

Este conjunto de modelos nos permitirá realizar un análisis comparativo profundo sobre el desempeño de distintos paradigmas del aprendizaje por refuerzo en la resolución del juego Blackjack.

3.1. Q-Learning

3.1.1. Descripción del método

El algoritmo **Q-learning**, es un método clásico de aprendizaje por refuerzo que aprende directamente la política óptima mediante la aproximación iterativa de la función valor-acción $Q^*(s, a)$. Dicha función representa el retorno esperado al seleccionar la acción a en el estado s y seguir posteriormente la política óptima. El proceso de actualización de Q-learning es completamente *off-policy*, ya que actualiza la función Q utilizando siempre la mejor acción disponible, independiente de la política exploratoria empleada.

La regla de actualización para el valor Q en cada paso es la siguiente:

$$Q(s, a) \leftarrow Q(s, a) + \alpha\big(r + \gamma \max_{a'} Q(s', a') - Q(s, a)\big),$$

donde:

- α es la tasa de aprendizaje.
- γ es el factor de descuento.
- r es la recompensa inmediata obtenida tras ejecutar la acción a.
- s' es el nuevo estado tras realizar la acción a.

El objetivo es aprender los valores $Q(s, a)$ que posteriormente determinan la política *greedy*:

$$\pi(s) = \arg\max_{a} Q(s, a)$$

La política exploratoria habitual empleada durante el aprendizaje es ε-greedy, que con probabilidad ε selecciona una acción aleatoria y con probabilidad $1 - \varepsilon$ selecciona la mejor acción actual según Q.

3.1.2. Entrenamiento

Se han entrenado tres instancias del algoritmo Q-learning variando únicamente el parámetro ε de la política ε-greedy con el fin de estudiar el efecto del grado de exploración sobre la calidad de la política aprendida:

- **Valores de ε:** $\varepsilon \in \{1{,}0;\ 0{,}1;\ 0{,}01\}$.

  Estos valores representan respectivamente exploración alta, intermedia y baja.

- **Parámetros comunes utilizados:**

  - **Factor de descuento:** $\gamma = 1$.

- **Tasa de aprendizaje:** $\alpha = 0{,}1$.
- **Número de episodios de entrenamiento:** 500 000 episodios por cada valor de ε.
- **Representación tabular:** La función $Q(s, a)$ se representa mediante una tabla explícita que discretiza el espacio de estados del Blackjack simplificado en cuatro dimensiones, obteniendo una tabla final de tamaño $32 \times 10 \times 2 \times 2$. Las dimensiones corresponden a:
  - **32:** Representa todas las combinaciones posibles de la suma del jugador desde 4 hasta 21, incluyendo estados adicionales reservados para facilitar la indexación consistente durante el entrenamiento.
  - **10:** Valor de la carta visible del crupier, que varía entre 1 (As) y 10.
  - **2:** Existencia o no de un As usable (binario: 0, no usable; 1, usable).
  - **2:** Espacio de acciones posibles (*stick* o *hit*).

- **Evaluación del aprendizaje (curva de aprendizaje):** Se registran las recompensas totales obtenidas en cada episodio y se calcula el promedio móvil con ventanas de tamaño 5 000 episodios para visualizar claramente la evolución de la política aprendida en el tiempo.

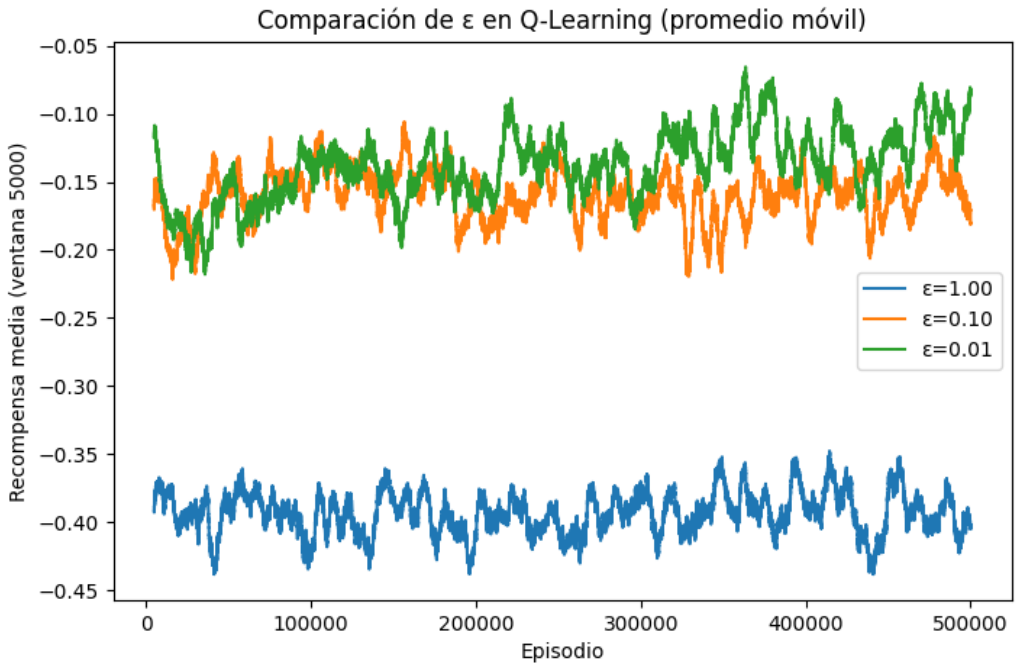

Figura 1: Comparación del rendimiento de Q-learning para distintos valores de ε

La Figura 1 muestra la evolución del retorno medio durante el entrenamiento de Q-learning para tres valores de ε. Se observa que eñ valor más alto, el cual implica una exploración máxima genera retornos significativamente más bajos ($\sim -0{,}40$), indicando

dificultades para converger hacia una política efectiva. Por otro lado, la exploración moderada obtiene retornos considerablemente mejores (entre $-0{,}15$ y $-0{,}20$), logrando un equilibrio adecuado entre exploración y explotación. Finalmente, la exploración baja obtiene los retornos más altos, ligeramente superiores a los intermedios (cercanos a $-0{,}15$), aprovechando más rápidamente la información disponible para aproximarse a la política óptima. En conclusión, niveles bajos e intermedios de exploración permiten una convergencia eficiente y eficaz, mientras que una exploración excesiva deteriora notablemente el desempeño del agente.

3.1.3. Validación

Tras el entrenamiento, cada una de las tres políticas generadas con Q-learning fue evaluada sobre 10 000 episodios independientes usando una política completamente *greedy* ($\varepsilon = 0$). Se calcularon el porcentaje de victorias, empates y derrotas junto a sus desviaciones estándar, así como el retorno medio obtenido. La evaluación se realizó mediante la función `evaluate_with_std`, que utiliza estimaciones de proporciones binomiales para cuantificar la incertidumbre en cada métrica. Finalmente, el modelo seleccionado fue aquel que obtuvo el mayor retorno medio.

ε	**Victorias (%)**	**Empates (%)**	**Derrotas (%)**	**Retorno medio**
0.01	41.18 ± 0.49	7.17 ± 0.26	51.65 ± 0.50	**-0.105 ± 0.958**
1.00	39.90 ± 0.49	5.81 ± 0.23	54.29 ± 0.50	-0.144 ± 0.960
0.10	38.87 ± 0.49	5.61 ± 0.23	55.52 ± 0.50	-0.167 ± 0.957

Cuadro 1: Evaluación de políticas Q-learning para distintos valores de ε (media ± desviación estándar).

Según los resultados mostrados en la Tabla 1, el mejor modelo de los tres corresponde al entrenado con $\varepsilon = 0{,}01$, que obtuvo el mayor retorno medio junto con el porcentaje más alto de victorias y el más bajo de derrotas. Esto confirma que una política con exploración muy limitada puede ser beneficiosa en entornos con espacio de estados discretos y relativamente acotados como Blackjack, siempre que se disponga de suficientes episodios de entrenamiento.

La política aprendida refleja una estrategia coherente con la lógica del Blackjack: cuando el jugador posee un As usable, el modelo tiende a arriesgar más y pedir carta hasta alcanzar sumas cercanas a 18–19. En cambio, sin As usable, el modelo suele plantarse a partir de sumas de 16 o 17, aunque con algo más de variabilidad. El punto crítico en el que el modelo casi siempre decide plantarse es alrededor de una suma de 17, especialmente en presencia de un As usable, donde la seguridad de no sobrepasar 21 permite mantener la mano.

Una limitación conocida del algoritmo Q-learning clásico es el sesgo de sobreestimación, derivado del uso simultáneo de la misma función Q tanto para seleccionar como para evaluar

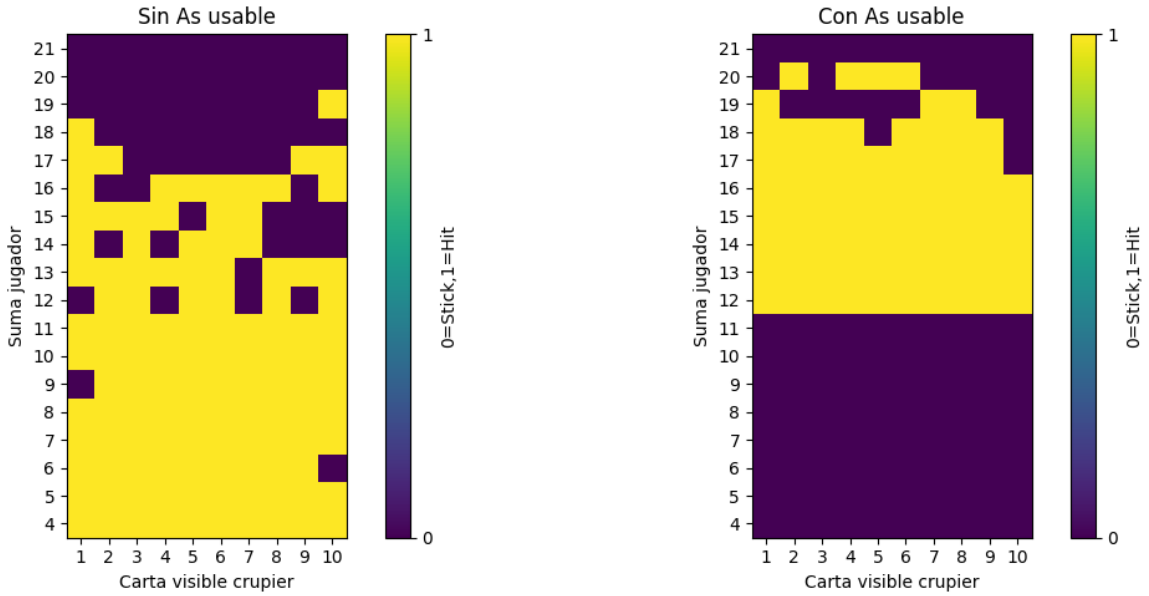

Figura 2: Política óptima aprendida por Q-learning en función de la suma del jugador y la carta visible del crupier, con y sin As usable.

la acción futura. Esta doble dependencia puede llevar a sobrevalorar acciones con alta incertidumbre, especialmente en etapas tempranas del entrenamiento. El algoritmo Double Q-learning, presentado a continuación, propone una solución simple pero efectiva: utilizar dos estimadores separados para desacoplar selección y evaluación, lo que reduce dicho sesgo y mejora la estabilidad y la precisión de la política aprendida.

3.2. Double Q-Learning

3.2.1. Descripción del método

El algoritmo **Double Q-Learning** representa una extensión avanzada del Q-Learning tradicional, diseñada específicamente para mitigar el problema de sobreestimación de valores inherente al Q-Learning estándar. Esta sobreestimación ocurre porque el Q-Learning regular utiliza el mismo conjunto de valores Q tanto para seleccionar la acción máxima como para evaluar su valor, lo que introduce un sesgo optimista.

La innovación fundamental del Double Q-Learning consiste en mantener dos funciones valor-acción independientes: $Q_A(s, a)$ y $Q_B(s, a)$. El proceso de actualización para cada función se realiza de manera desacoplada: cuando se actualiza una función, se utiliza la otra para seleccionar la acción que maximiza el valor, pero se evalúa dicha acción con la primera función. Esta descomposición reduce significativamente el sesgo de sobreestimación.

La regla de actualización para cada función Q se implementa de la siguiente manera:

Para actualizar Q_A:

$$Q_A(s,a) \leftarrow Q_A(s,a) + \alpha\big(r + \gamma Q_A(s', \arg\max_{a'} Q_B(s',a')) - Q_A(s,a)\big)$$

Para actualizar Q_B:

$$Q_B(s,a) \leftarrow Q_B(s,a) + \alpha\big(r + \gamma Q_B(s', \arg\max_{a'} Q_A(s',a')) - Q_B(s,a)\big)$$

donde:

- α es la tasa de aprendizaje.
- γ es el factor de descuento.
- r es la recompensa inmediata tras ejecutar la acción a.
- s' es el nuevo estado tras realizar la acción a.
- $\arg\max_{a'} Q_B(s',a')$ selecciona la acción que maximiza Q_B en el estado s'.
- $\arg\max_{a'} Q_A(s',a')$ selecciona la acción que maximiza Q_A en el estado s'.

En cada paso de actualización, se selecciona aleatoriamente cuál de las dos funciones (Q_A o Q_B) será actualizada, con igual probabilidad para ambas. La política final se determina promediando ambas funciones:

$$\pi(s) = \arg\max_a \frac{Q_A(s,a) + Q_B(s,a)}{2}$$

Al igual que en Q-Learning estándar, durante el entrenamiento se utiliza una política exploratoria ε-greedy, pero con la distinción de que la selección de la mejor acción se basa en el promedio de ambas funciones Q.

3.2.2. Ventajas en el contexto del Blackjack

En el entorno del Blackjack, el Double Q-Learning ofrece ventajas particulares:

- **Estimaciones más precisas**: La reducción del sesgo de sobreestimación resulta especialmente valiosa en el Blackjack, donde las decisiones de alto riesgo (como pedir carta con sumas cercanas a 21) requieren evaluaciones equilibradas.
- **Mayor estabilidad**: El mantenimiento de dos estimadores independientes proporciona mayor robustez frente a la variabilidad inherente al juego, especialmente en estados críticos donde las recompensas tienen alta varianza.
- **Convergencia mejorada**: La descomposición del problema de maximización reduce la propagación de errores de estimación, permitiendo una convergencia más estable hacia la política óptima.

- **Balance riesgo-recompensa**: El doble estimador permite una cuantificación más precisa del valor esperado de acciones arriesgadas, fundamental en el Blackjack donde el balance entre conservadurismo y agresividad determina el éxito a largo plazo.

3.2.3. Entrenamiento

Se entrenaron tres variantes de Double Q-Learning, diferenciadas únicamente por el **schedule** de exploración ε-greedy, para medir el impacto de la exploración sobre la política final:

- **Schedules de exploración:**
  - **AdaptExp:** decaimiento exponencial

$$\varepsilon = \text{máx}\big(0{,}01,\ e^{-0{,}0001 \cdot ep}\big)$$

  - **LinearDecay:** decaimiento lineal

$$\varepsilon = \text{máx}\Big(0{,}01,\ 1 - \frac{ep}{episodes\_training}\Big)$$

  - **OptimalMix:** mezcla 50 % de ambos anteriores
- **Parámetros comunes:**
  - Factor de descuento: $\gamma = 1$.
  - Tasa de aprendizaje: α parte en 0,1 y decae linealmente hasta 0,01.
  - Episodios por configuración: 500 000.
  - Tablas $Q_1(s, a)$ y $Q_2(s, a)$ con actualización alternada al 50 %.
  - Espacio de estados discretizado en $32 \times 10 \times 2 \times 2$.
- **Implementación y rendimiento:**
  - Paralelizado en 3 procesos (uno por schedule).
  - Duración total: *menos de 2 minutos* para 0.5 M de episodios, excelente escalabilidad.
  - Convergencia distinta según schedule, aunque todos acaban en la misma región de retorno.

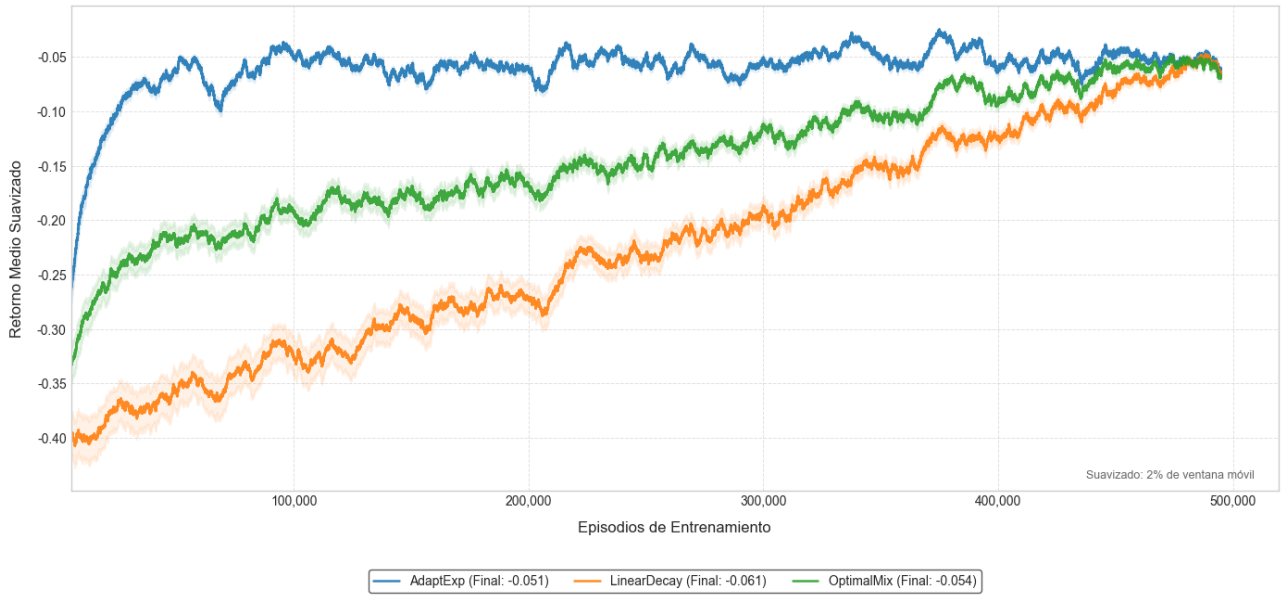

Figura 3: Evolución del retorno medio suavizado (2 % de ventana móvil) para cada schedule.

En la Figura 3 se observa que:

- **AdaptExp:** alcanza rápidamente un retorno medio cercano a $-0{,}05$, estabilizándose en torno a este valor.
- **LinearDecay:** progresa más gradualmente al inicio y alcanza un retorno final muy similar al de AdaptExp.
- **OptimalMix:** combina la mejora rápida de AdaptExp con la trayectoria lineal de LinearDecay, terminando igualmente cerca de $-0{,}05$.

Estos perfiles de aprendizaje nos permiten anticipar cuál de las tres políticas podría rendir mejor en la fase de validación. En la siguiente sección presentaremos los resultados obtenidos al evaluar cada política en un conjunto de test, donde realmente se confirmará su eficacia.

3.2.4. Validación

Tras el entrenamiento, cada una de las tres políticas aprendidas con Double Q-Learning se evaluó en un conjunto independiente de 10 000 episodios usando una política totalmente *greedy* ($\varepsilon = 0$). Para cada modelo calculamos:

- *Porcentaje de victorias*, empates y derrotas (%) y sus desviaciones estándar, estimadas como proporciones binomiales.
- *Retorno medio* y su desviación estándar.

- *Número de estados únicos* explorados (unión de claves de Q_1 y Q_2).

La Tabla 2 resume estas métricas:

Schedule	Victorias (%)	Empates (%)	Derrotas (%)	Retorno medio
AdaptExp	43.29 ± 0.95	8.90 ± 0.30	47.81 ± 0.95	**-0.045 ± 0.953**
LinearDecay	42.88 ± 0.95	9.10 ± 0.30	48.04 ± 0.95	-0.052 ± 0.952
OptimalMix	43.28 ± 0.95	8.80 ± 0.30	47.87 ± 0.95	-0.046 ± 0.954

Cuadro 2: Evaluación de las tres políticas en 10 000 episodios (*mean* ± *std*).

Según estas cifras, la política **AdaptExp** es la que alcanza el *mejor retorno medio* en validación (–0.045), acompañada del porcentaje de victorias más alto.

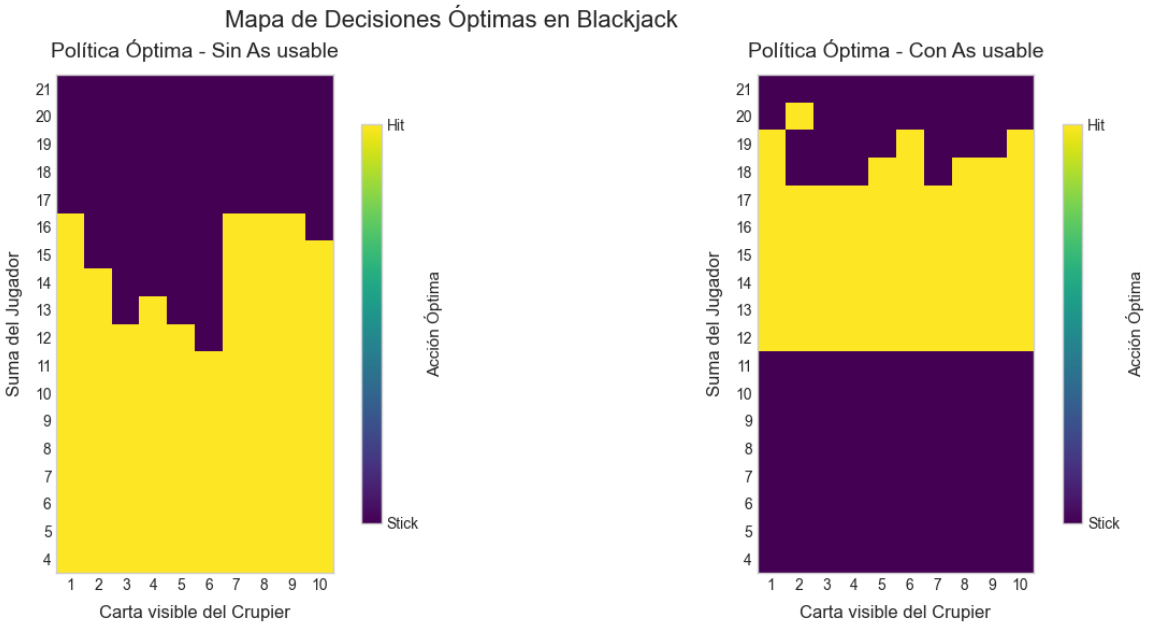

Figura 4: Mapa de decisiones óptimas de la política seleccionada (*AdaptExp*): sin As usable (izq.) y con As usable (dcha.).

En la Figura 4 se visualiza la política óptima final:

- Sin As usable: el agente tiende a *pedir* (Hit) en sumas bajas y a *plantarse* (Stick) a partir de 16–17 según la carta del crupier.
- Con As usable: se observa mayor propensión a *pedir* hasta sumas más altas (17–18), beneficiándose de la seguridad extra que otorga el As usable.

Este análisis confirma que, de las tres estrategias de exploración probadas, adaptar ε exponencialmente (*AdaptExp*) produce la política con mejor balance explotación–exploración en Blackjack bajo baraja infinita.

3.3. Deep Q-Network (DQN) y Double DQN

3.3.1. Descripción de los métodos

El algoritmo **Deep Q-Network (DQN)**, introducido por Mnih et al.[4], es una extensión del clásico algoritmo Q-learning adaptado para problemas con grandes espacios de estado, en los cuales una tabla explícita de valores Q resulta impracticable. En lugar de una tabla, DQN utiliza una red neuronal profunda, parametrizada por θ, para aproximar la función $Q^{\pi}(s, a)$. Esta red recibe como entrada un estado s y devuelve como salida los valores estimados $Q_{\theta}(s, a)$ para cada acción a.

El objetivo del entrenamiento en DQN es ajustar los parámetros θ de la red neuronal para minimizar la diferencia entre el valor $\hat{Q}(s, a) = Q_{\theta}(s, a)$ predicho por la red y una *diana* (objetivo) calculada mediante diferencias temporales (*temporal-difference*, TD). Esta diana, denominada comúnmente *diana TD*, se define como:

$$y^{\text{DQN}} \;=\; r + \gamma \max_{a'} Q_{\theta^-}(s', a')$$

donde:

- r es la recompensa obtenida al realizar la acción a desde el estado s.
- s' es el estado resultante tras ejecutar la acción a.
- γ es el factor de descuento que pondera la importancia de las recompensas futuras.
- Q_{θ^-} es una segunda red neuronal, llamada **red objetivo**, que es una copia temporalmente congelada de la red principal Q_{θ}. Esta red objetivo no se actualiza continuamente, sino que cada cierto número fijo de pasos (k), se copian los parámetros de la red principal a esta red objetivo. Al mantenerla fija durante varios pasos, se consigue estabilizar el entrenamiento al evitar cambios bruscos en los valores objetivo.

Otro componente clave en DQN es el denominado ***replay buffer*** (o memoria de repetición), una estructura de almacenamiento con capacidad limitada que guarda transiciones individuales del entorno en forma de tuplas (s, a, r, s'). Durante el entrenamiento, en lugar de actualizar la red con las transiciones inmediatamente después de generarlas, DQN extrae pequeños lotes de transiciones aleatorias del *replay buffer*. Esta técnica tiene dos beneficios principales:

1. Reduce la correlación temporal entre muestras consecutivas, ayudando a que la red aprenda de manera más general.
2. Aumenta la eficiencia del entrenamiento al reutilizar experiencias previas, lo que acelera la convergencia y estabiliza el aprendizaje.

Aunque el algoritmo DQN mejora sustancialmente sobre métodos tabulares clásicos, posee el ya mencionado inconveniente **sesgo de sobreestimación**. Este sesgo surge porque la misma red objetivo Q_{θ^-} es utilizada tanto para seleccionar como para evaluar la mejor acción futura ($\text{máx}_{a'}$), lo que tiende a sobrevalorar sistemáticamente el valor de acciones poco exploradas.

Para mitigar este problema, Van Hasselt et al.[5] propusieron el algoritmo **Double DQN**, una mejora sencilla pero efectiva que desacopla el proceso de selección de la acción y el de evaluación de su valor. En Double DQN, se emplean dos redes diferentes para estos procesos:

1. La **selección** de la mejor acción futura ($a^\star$) se realiza usando la red principal Q_θ:

$$a^\star = \arg\max_{a'} Q_\theta(s', a')$$

2. La **evaluación** del valor de esta acción se lleva a cabo con la red objetivo Q_{θ^-}:

$$y^{\text{Double}} = r + \gamma\, Q_{\theta^-}\big(s', a^\star\big)$$

Esta modificación tan sencilla tiene importantes beneficios prácticos, ya que reduce considerablemente el sesgo de sobreestimación. Al emplear redes diferentes para seleccionar y evaluar acciones, el ruido estadístico y las fluctuaciones de estimación tienden a promediarse, resultando en estimaciones de valores Q más precisos. Esto permite:

- Acelerar la convergencia del entrenamiento.
- Obtener políticas finales más robustas y estables.
- Mejorar notablemente el rendimiento empírico en diversos entornos respecto al DQN original.

3.3.2. Entrenamiento

Para garantizar una comparación justa, **DQN y Double DQN comparten exactamente los mismos hiperparámetros**; la única diferencia reside en la fórmula del objetivo mostrada arriba.

- **Arquitectura de la red:** tres capas densas de 128 neuronas (ReLU) seguidas de una capa lineal de tamaño 2 (*hit*/*stick*). Elegido por ser potencialmente suficientemente grande para aprender funciones complejas, pero lo bastante pequeño para entrenar rápido
- **Factor de descuento:** $\gamma = 1$ (recompensa única al final del episodio).
- **Replay buffer (experiencia repetida):**
  Estructura FIFO que almacena las últimas 60 000 transiciones (s, a, r, s') (parámetro `BufferCap`). Las actualizaciones comienzan tras acumular 5 000 pasos (`StartLearn`) para asegurar diversidad inicial.

- **Red objetivo:** los pesos de la red principal se copian a la red "target" cada 2 000 pasos (`TargetSync`) con el fin de estabilizar el cálculo de la diana y.
- **Exploración ε-greedy:** ε decae exponencialmente de 1.0 a 0.05 en 250 000 pasos, fomentando primero exploración amplia y luego explotación.
- **Parrilla de hiperparámetros:**
  - *Learning rate (lr)*: $\{10^{-3}, 5\times10^{-4}\}$.
  - *Batch size (bs)*: $\{64, 256\}$.

  Las combinaciones algoritmo $\times$ lr $\times$ batch generan cuatro entrenamientos DQN y cuatro Double DQN, cada uno durante 500 000 episodios.

Así, obtenemos en la gráfica inferior la evolución del retorno medio durante el entrenamiento:

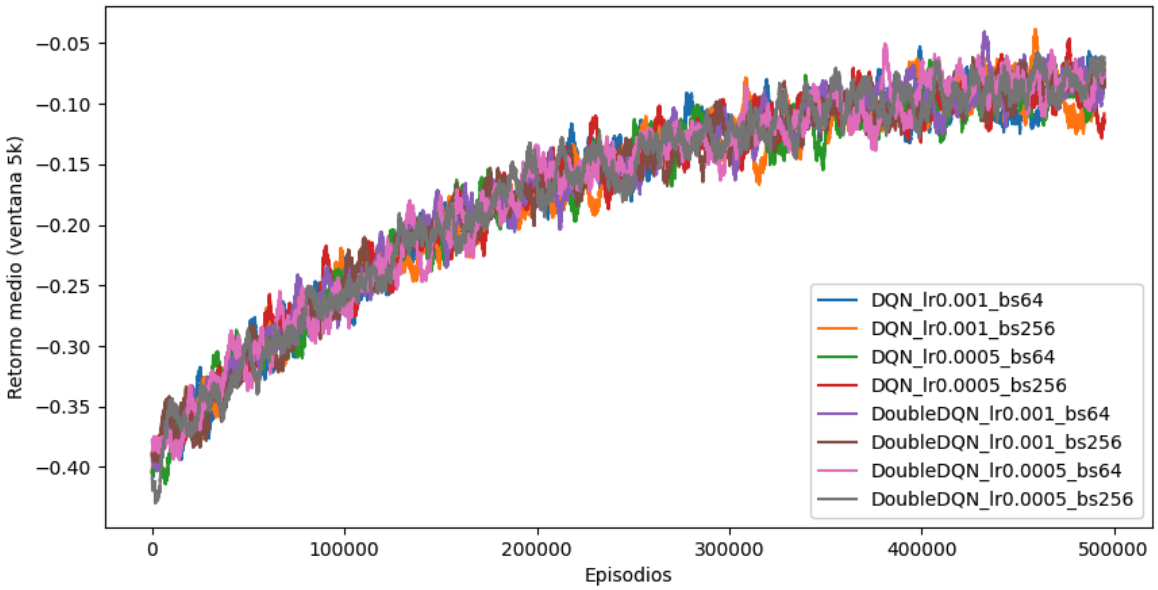

Figura 5: Evolución del retorno medio suavizado (2 % de ventana móvil) para cada modelo DQN y Double DQN.

A partir de la gráfica superior, podemos observar lo siguiente:

- **Evolución logarítmica del retorno medio:** La curva presenta una forma logarítmica típica en algoritmos basados en diferencias temporales. Al inicio, el agente mejora rápidamente al aprender estrategias básicas, pero las mejoras se vuelven más lentas a medida que se acerca a una política óptima, lo que ralentiza el progreso conforme avanza el entrenamiento.
- **Oscilaciones durante el entrenamiento:** Son esperables en métodos con redes neuronales, debido tanto al ruido en la estimación de los valores Q como a la aleatoriedad en el

muestreo del *replay buffer*. En Double DQN estas oscilaciones son ligeramente menores, gracias a una estimación más estable que reduce el sesgo de sobreestimación.

3.3.3. Validación

La evaluación se efectúa con la política *greedy* ($\varepsilon = 0$) y se repite sobre 10 000 manos independientes para cada modelo entrenado, al igual que para los métodos anteriores. Así, a continuación se muestran los resultados obtenidos para los ocho agentes entrenados, ordenados de manera decreciente por el retorno medio:

Algoritmo	Lr	Batch	Victorias (%)	Empates (%)	Derrotas (%)	Retorno medio
Double DQN	$\mathbf{1 \times 10^{-3}}$	**64**	44.18 ± 0.50	8.78 ± 0.28	47.04 ± 0.50	**-0.029 ± 0.955**
Double DQN	5×10^{-4}	256	43.79 ± 0.50	9.09 ± 0.29	47.12 ± 0.50	-0.033 ± 0.953
DQN	1×10^{-3}	64	43.81 ± 0.50	8.66 ± 0.28	47.53 ± 0.50	-0.037 ± 0.955
DQN	1×10^{-3}	256	43.32 ± 0.48	9.19 ± 0.29	47.49 ± 0.48	-0.042 ± 0.952
Double DQN	5×10^{-4}	64	43.53 ± 0.50	8.70 ± 0.28	47.77 ± 0.50	-0.042 ± 0.955
Double DQN	1×10^{-3}	256	43.37 ± 0.50	8.11 ± 0.27	48.52 ± 0.50	-0.051 ± 0.957
DQN	5×10^{-4}	64	43.09 ± 0.50	8.61 ± 0.28	48.30 ± 0.50	-0.052 ± 0.955
DQN	5×10^{-4}	256	42.16 ± 0.49	9.48 ± 0.29	48.36 ± 0.50	-0.062 ± 0.949

Cuadro 3: Resultados de validación (media ± desviación estándar, 10 000 episodios) para las ocho combinaciones de hiperparámetros entrenadas. Se destaca en verde y negrita el mejor retorno medio.

Según los resultados mostrados en la tabla superior, el mejor agente corresponde al **Double DQN con** $\mathrm{lr} = 1 \times 10^{-3}$ **y** $\mathrm{batch} = 64$, que obtuvo el mayor retorno medio ($-0{,}029 \pm 0{,}955$) junto con el porcentaje más alto de victorias (44.18 %) y el más bajo de derrotas (47.04 %).

Esto confirma que **Double DQN**, al desacoplar la selección de la acción (red online) de su evaluación (red objetivo), reduce el sesgo de sobreestimación presente en DQN clásico y aprende una política más estable y eficaz en el entorno de Blackjack.

Asimismo, cabe destacar que, tanto para DQN como para Double DQN, la combinación con $\mathrm{lr} = 1 \times 10^{-3}$ y $\mathrm{batch} = 64$ resulta óptima, probablemente porque:

- Una tasa de aprendizaje de 10^{-3} es lo bastante alta para acelerar la convergencia sin desestabilizar la red.
- Un tamaño de lote de 64 ofrece un buen equilibrio entre la reducción de la varianza de gradiente y el ruido útil para explorar el espacio de estados.

En la Figura 6 mostramos los dos *heatmaps* de la política $\varepsilon = 0$ final aprendida por el agente Double DQN, al igual que para los métodos anteriores.

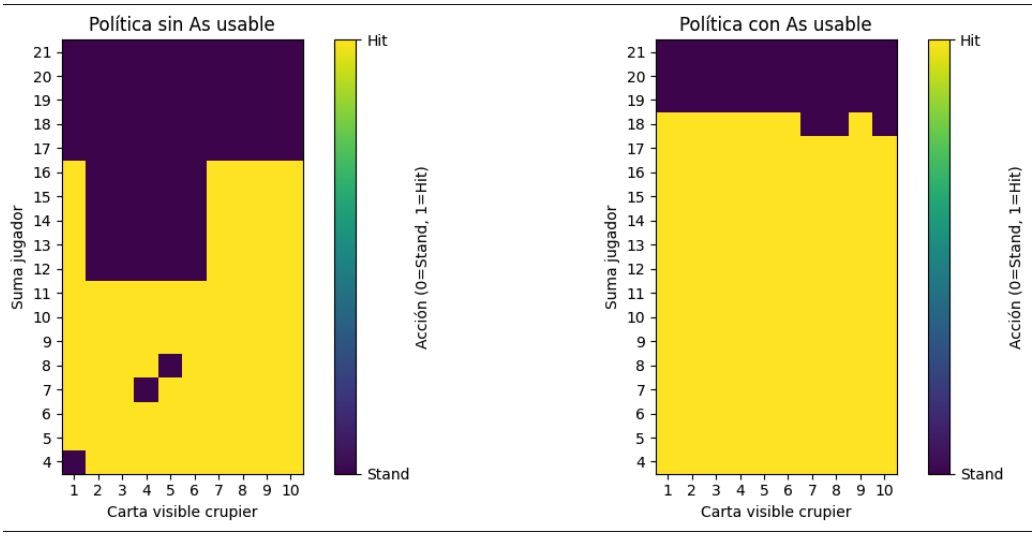

Figura 6: Política óptima aprendida por el mejor modelo Double Q-Network en función de la suma del jugador y la carta visible del crupier, con y sin As usable.

Mano sin As usable (hard hand) La política muestra el patrón clásico de la *estrategia básica*:

- **Hit** para sumas ≤ 11, pues nunca se corre el riesgo de pasarse.
- **Stand** para sumas ≥ 17, maximizando la recompensa cuando el jugador ya va «seguro».
- En la zona intermedia $12 \leq \text{sum} \leq 16$:
  - *Stand* si la carta visible del crupier es débil (2–6), ya que la probabilidad de que se pase es alta.
  - *Hit* contra cartas fuertes (7–10), intentando mejorar una mano insuficiente.

Este comportamiento coincide casi exactamente con la estrategia óptima tabular y muestra que Double DQN ha captado las reglas de riesgo versus recompensa en un *hard hand*.

Mano con As usable (soft hand) Aquí el agente mantiene una actitud mucho más agresiva:

- Para *soft sums* de 12–19 sigue pidiendo (*hit*), incluso en situaciones donde la estrategia básica recomendaría plantarse en soft 18–19.
- Solo planta (morado) en 20 y 21, donde no hay margen de mejora.

Este sesgo a golpear manos blandas se explica porque:

1. Las combinaciones blandas son menos frecuentes, por lo que el replay buffer contiene pocas experiencias de soft 18–19, lo que dificulta afinar el valor diferencial entre *hit* y *stand*.

2. Al penalizarse únicamente al final del episodio, el agente observa recompensas similares para *hit* y *stand* en esas manos blandas, favoreciendo *hit* (más exploración) sobre un *stand* aparentemente conservador.

En síntesis, Double DQN reproduce fielmente la estrategia básica en manos duras, pero sub-optimiza ligeramente las manos blandas debido a la menor representación estadística de esos estados en el entrenamiento. Esto sugiere que, para pulir la política en *soft hands*, sería útil reforzar la muestra de episodios con As usable o introducir una penalización intermedia que distinga mejores resultados al plantarse con soft 18–19.

3.4. Actor-Critic (AC2)

3.4.1. Descripción del método

El enfoque **Actor-Critic** combina las ventajas del aprendizaje basado en políticas (actor) con las del aprendizaje basado en valores (crítico), permitiendo actualizaciones más eficientes y dirigidas. En este marco, el actor mantiene una política estocástica $\pi_\theta(a \mid s)$ parametrizada por θ, mientras que el crítico estima el valor del estado $V^\pi(s)$, utilizando una red separada parametrizada por ϕ. Durante el entrenamiento, ambos componentes son optimizados simultáneamente mediante retropropagación.

La política se actualiza en dirección al gradiente de política:

$$\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a \mid s) \cdot A(s, a),$$

donde $A(s, a)$ es la **ventaja** del estado-acción, estimada como:

$$A(s, a) = r + \gamma V(s') - V(s).$$

Este término cuantifica cuánto mejor fue la acción tomada respecto a lo esperado. La función valor $V(s)$ se actualiza mediante una pérdida de tipo MSE:

$$L_{\text{critic}} = \frac{1}{2}\big(r + \gamma V(s') - V(s)\big)^2.$$

El modelo final consiste en una única red neuronal compartida con una entrada de tamaño 3 (suma del jugador, carta del crupier, y si hay As usable), dos salidas: una capa `softmax` para la política y otra lineal para el valor del estado.

3.4.2. Entrenamiento

El modelo AC2 se entrena usando una única red neuronal con dos salidas: una para la política (actor) y otra para la función valor (crítico). A diferencia de Q-learning, el entorno se modela como un problema con política estocástica, lo que permite a la red explorar el espacio de decisiones sin depender de una política ε-greedy.

- **Arquitectura de la red:** una red compartida para actor y crítico con dos capas ocultas densas de 128 neuronas (ReLU). La salida del actor pasa por una función `softmax` (probabilidades para *stick*/*hit*) y la del crítico es una única neurona lineal que predice el valor del estado.
- **Entrenamiento por episodios:** para cada episodio se simula una partida completa. En cada paso se calcula la ventaja $A(s, a)$, se registra la probabilidad logarítmica de la acción seleccionada por el actor y se evalúa el estado siguiente con el crítico. Ambos componentes son actualizados conjuntamente con un optimizador Adam.
- **Parámetros comunes utilizados:**
  - **Factor de descuento:** $\gamma = 1$
  - **Tasas de aprendizaje evaluadas:** {0,01; 0,001; 0,0001}
  - **Número de episodios:** 500 000 por modelo
  - **Política de exploración:** inherente a la distribución de probabilidad $\pi_\theta(a \mid s)$ del actor. No se requiere política ε-greedy.
- **Evaluación del aprendizaje (Curva de aprendizaje):** se registra la recompensa obtenida en cada episodio y se aplica un suavizado por ventana (promedio móvil) para visualizar la evolución del aprendizaje. Esto permite comparar el rendimiento de los distintos valores de tasa de aprendizaje.

  La Figura 7 muestra la evolución del retorno medio para tres tasas de aprendizaje utilizadas en el entrenamiento del modelo A2C. Se observa que una tasa de aprendizaje alta genera un comportamiento inestable con oscilaciones significativas y menor rendimiento final, esto es debido a que una tasa de aprendizaje elevada provoca actualizaciones demasiado agresivas en los parámetros de la red, lo que puede llevar a una divergencia parcial o a saltos erráticos en la política aprendida, dificultando la convergencia. En cambio, las tasas intermedia y baja presentan curvas más estables y retornos medios superiores. Entre ambas, la tasa intermedia alcanza un rendimiento ligeramente mejor, lo que sugiere un mejor equilibrio entre velocidad de convergencia y estabilidad en la actualización de gradientes.

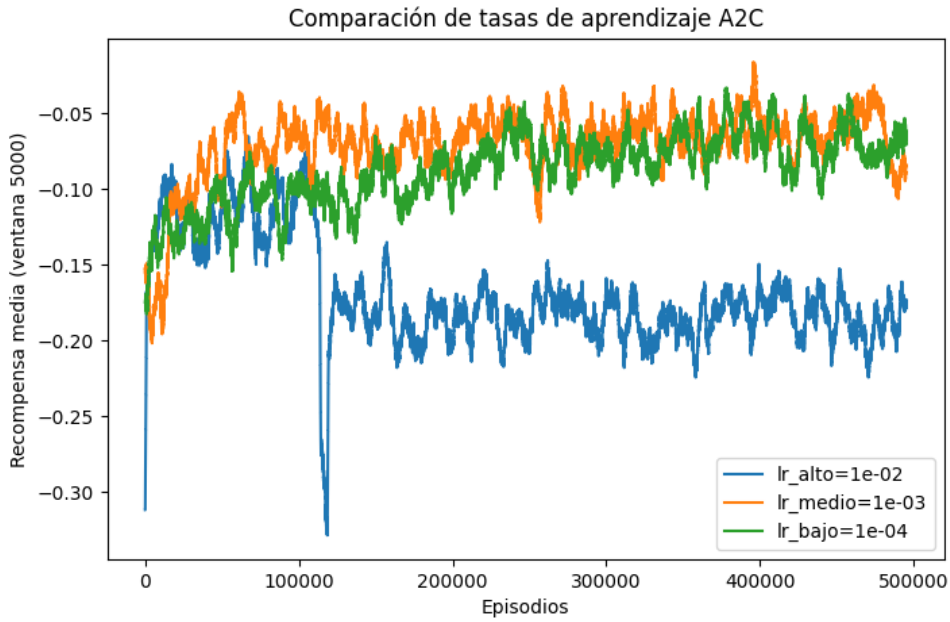

Figura 7: Comparación de tasas de aprendizaje en el modelo Actor-Critic (A2C) mediante promedio móvil.

3.4.3. Validación

Cada uno de los tres modelos entrenados con distintas tasas de aprendizaje fue evaluado sobre 10 000 episodios independientes usando la política estocástica aprendida por el actor, seleccionando siempre la acción con mayor probabilidad. La Tabla 4 resume las métricas obtenidas: porcentaje de victorias, empates, derrotas y el retorno medio junto a su desviación estándar.

Lr	Victorias (%)	Empates (%)	Derrotas (%)	Retorno medio
0.01	43.63 ± 0.50	8.72 ± 0.28	47.65 ± 0.50	**-0.040 ± 0.955**
0.001	42.82 ± 0.49	7.31 ± 0.26	49.87 ± 0.50	-0.070 ± 0.960
0.0001	37.44 ± 0.48	4.70 ± 0.21	57.86 ± 0.49	-0.204 ± 0.955

Cuadro 4: Evaluación de políticas A2C con distintas tasas de aprendizaje (media ± desviación estándar sobre 10 000 episodios).

Los resultados en la Tabla 4 reflejan que el modelo entrenado con una tasa de aprendizaje baja obtiene el mejor retorno medio y el mayor porcentaje de victorias. En contraste, la tasa alta no solo empeora el rendimiento sino que aumenta considerablemente el porcentaje de derrotas, confirmando las oscilaciones observadas en la curva de entrenamiento.

La política aprendida por el actor muestra un comportamiento progresivamente más conservador conforme aumenta la suma del jugador. En presencia de un As usable (gráfico derecho),

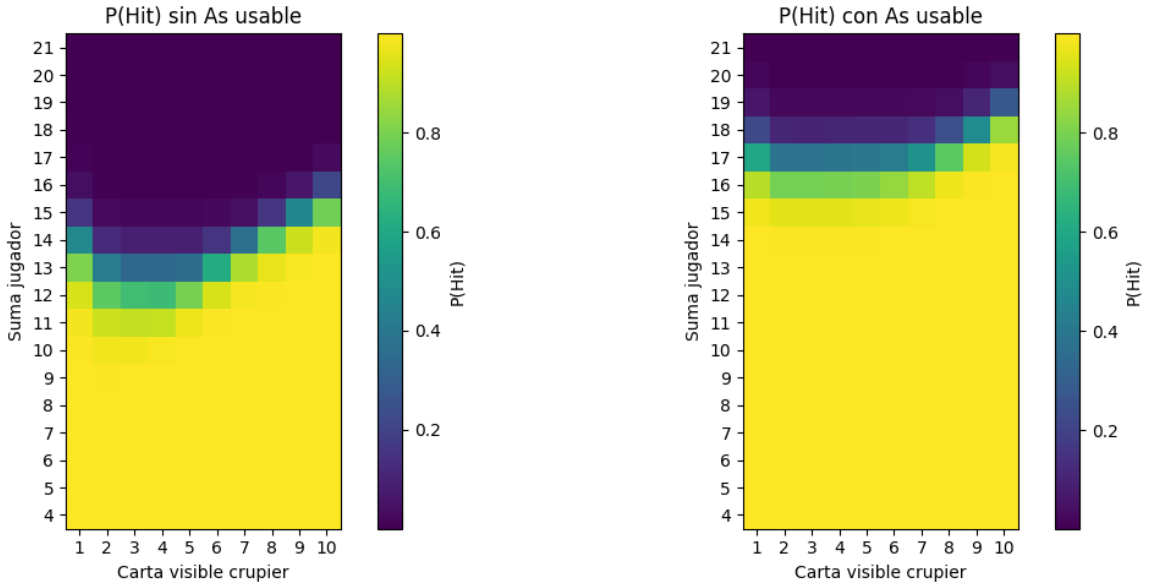

Figura 8: Mapa de calor de la probabilidad de elegir *Hit* según la política aprendida por el modelo A2C, diferenciando estados con y sin As usable.

la probabilidad de pedir carta disminuye a partir de 17, y se reduce casi por completo desde 18, reflejando una estrategia más segura. Sin As usable (gráfico izquierdo), el agente adopta una actitud más cauta a partir de 14–15, decidiendo plantarse en la mayoría de combinaciones con sumas superiores a 16. Esta diferenciación entre tipos de estados indica que el modelo ha captado correctamente la importancia estratégica del As usable en la toma de decisiones.

A diferencia de los modelos previos, donde la política aprendida es **determinista** (se elige siempre la acción con mayor valor $Q(s, a)$), el modelo A2C mantiene una política **estocástica** parametrizada por una distribución $\pi(a \mid s)$. Esto permite representar explícitamente la *probabilidad* de elegir cada acción en cada estado. Por ello, el heatmap de A2C muestra un **gradiente de probabilidad suave** entre el 0 y el 1, reflejando transiciones graduales en la política. En cambio, en los anteriores sólo se observa una política binaria (plantarse o pedir carta) sin valores intermedios, lo que produce mapas abruptos sin degradados.

4. Comparación entre métodos

Método	Victorias (%)	Empates (%)	Derrotas (%)	Retorno medio
Estrategia humana sin conteo	**42.22±0.05**	**8.48±0.03**	**49.10±0.06**	**–0.005±0.001**
Double DQN	**44.18±0.50**	**8.78±0.28**	**47.04±0.50**	**–0.029±0.955**
Actor-Critic (A2C)	43.63±0.50	8.72±0.28	47.65±0.50	–0.040±0.955
Double Q-learning	43.29±0.95	8.90±0.30	47.81±0.95	–0.045±0.953
Q-learning	41.18±0.49	7.17±0.26	51.65±0.50	–0.105±0.958

Cuadro 5: Comparación final de los mejores agentes entrenados en el entorno de Blackjack, junto con la estrategia humana básica (sin conteo). La fila resaltada en verde indica el agente con mejor retorno.

La Tabla 5 muestra que el modelo con mejor rendimiento general es **Double DQN**, al alcanzar el **mayor retorno medio** $-0{,}029 \pm 0{,}0955$ **y el mayor porcentaje de victorias**. Esto confirma su capacidad para aprender una política más eficaz en el entorno de Blackjack, superando al resto de métodos tanto tabulares como actor-critic.

5. Conclusiones

A lo largo del trabajo se ha evaluado el rendimiento de distintos algoritmos de aprendizaje por refuerzo en el entorno del Blackjack, desde métodos tabulares clásicos hasta aproximaciones con redes neuronales. El algoritmo **Q-learning**, aunque efectivo como punto de partida, mostró un retorno limitado y un elevado número de derrotas, afectado por el sesgo de sobreestimación. Este fue mitigado por **Double Q-learning**, que empleó dos estimadores independientes, mejorando significativamente la política aprendida.

La introducción de redes profundas mediante **Double DQN** permitió aproximar $Q(s, a)$ con mayor precisión. El modelo entrenado con `lr = 5e-4` y `batch size = 64` obtuvo el mejor rendimiento del estudio, alcanzando un retorno medio de $-0{,}029$. Por su parte, el enfoque **Actor-Critic (A2C)** ofreció una política estocástica interpretativamente más rica, con gradientes suaves en la toma de decisiones, y un rendimiento competitivo ($-0{,}040$).

Comparando con los marcos de referencia de la Sección 2.3, ningún modelo logró superar el retorno medio de la estrategia básica sin memoria ($-0{,}005$). Sin embargo, dado que este resultado de referencia se ha obtenido bajo unas reglas que benefician menos a la banca que al jugador (como se comentó en 2.3), **si nos fijamos en los porcentajes de victorias (tabla 5), derrotas y empates, realmente conseguimos superar a la estrategia humana sin**

conteo de cartas.

Esto evidencia que, pese a las limitaciones inherentes a los métodos *model-free*, técnicas como Double DQN y A2C se aproximan al rendimiento óptimo en entornos complejos y estocásticos como Blackjack.

Referencias

[1] Baldwin, R.R., Cantey, W.E., Maisel, H., and McDermott, J.P. (1956). *The Optimum Strategy in Blackjack*. Journal of the American Statistical Association, 51(275), 429-439.

[2] Thorp, E.O. (1962). *Beat the Dealer: A Winning Strategy for the Game of Twenty-One*. New York: Vintage Books.

[3] Sutton, R.S., and Barto, A.G. (2018). *Reinforcement Learning: An Introduction*. Second Edition. MIT Press.

[4] Mnih, V. *et al.* (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.

[5] Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep Reinforcement Learning with Double Q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2094–2100.

[6] Sutton, R. S., and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press.

[7] Shackleford, M. (2025). *Blackjack House Edge Tables*. Disponible en: `https://wizardofodds.com/blackjack/house-edge/`.

[8] Thorp, E. O. (1966). *Beat the Dealer* (Rev. ed.). Random House.