UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Sistemas Inteligentes Distribuidos

Agente de aprendizaje por refuezo para el entorno Cliff Walking

Lluc Martínez Busquets
Eric Medina León
Àlex Rodríguez Rodríguez

2024/2025 Q2

Resumen

Este trabajo presenta un estudio detallado sobre la implementación de cuatro algoritmos de aprendizaje por refuerzo en el entorno *CliffWalking-v0* de la libreria de Python *Gymnasium*: *Value Iteration*, *Direct Estimation*, *Q-Learning* y *REINFORCE*. El entorno se configura con el modo *is_slippery* activado, lo cual introduce estocasticidad en las transiciones de estado.
El objetivo es evaluar el rendimiento de cada algoritmo, así como qué parámetros e hiperparámetros son los óptimos para su funcionamiento.

Índice

1. Introducción

El aprendizaje por refuerzo es un paradigma de aprendizaje automático en el que un agente aprende a tomar decisiones interactuando con un entorno y recibiendo recompensas o penalizaciones.
Este trabajo se centra en la implementación y evaluación experimental de cuatro algoritmos de aprendizaje por refuerzo en el entorno *CliffWalking-v0*, de la librería de Python *Gymnasium*, con el objetivo de comparar su rendimiento y eficiencia en dicho entorno. Los algoritmos implementados son: *Value Iteration*, *Direct Estimation*, *Q-Learning* y *REINFORCE*.

1.1. Caracterización del entorno

El entorno de *Cliff Walking*, propuesto originalmente por *Sutton & Barto*, es un entorno clásico para evaluar algoritmos de aprendizaje por refuerzo. En este entorno, el agente debe navegar por una cuadrícula de dimensiones 4x12 evitando caer en un acantilado, lo que representa una penalización significativa. El objetivo del agente es llegar a la esquina inferior derecha de la cuadrícula de la forma más eficiente posible, maximizando la recompensa acumulada a lo largo del tiempo y minimizando el número de pasos necesarios para alcanzar la meta.
A continuación se caracteriza de forma detallada el entorno:

- **Observabilidad**: Totalmente observable. El agente recibe en cada paso su posición exacta en la cuadrícula, sin ruido ni información oculta, por lo que tiene acceso total al estado relevante.
- **Número de agentes**: Un único agente.
- **Determinismo**: Estocástico, ya que está activado el modo `is_slippery=True`. En este caso, por cada acción que el agente toma, hay una probabilidad de aproximadamente el 66.7 % de que el agente se resvale hacia una dirección perpendicular a la acción deseada.
- **Atomicidad**: Secuencial. Las decisiones del agente tienen consecuencias que dependen de toda la historia de acciones y percepciones, y los efectos futuros de las acciones importan para maximizar la recompensa acumulada.
- **Dinamicidad**: Estático. El estado del entorno sólo cambia cuando el agente toma una acción; no hay cambios "por sí mismos" mientras el agente razona.
- **Continuidad**: Discreto. Tanto el espacio de estados (posiciones en la cuadrícula) como el de acciones (arriba, abajo, izquierda, derecha) y el tiempo de decisión son discretos.
- **Conocimiento**: Conocido. Las reglas de transición (aunque estocásticas) y la función de recompensa están definidas de antemano y son accesibles al agente.

1.1.1. Espacio de acciones

El agente dispone de un conjunto finito de acciones

$$\mathcal{A} = \{\text{Arriba, Derecha, Abajo, Izquierda}\},$$

cada una de las cuales intenta desplazar al agente una celda en la dirección indicada.

1.1.2. Estados inicial y terminal

- **Estado inicial** $s_0 = (3, 0)$, correspondiente a la esquina inferior izquierda de la cuadrícula.
- **Estado terminal** $s_T = (3, 11)$, la meta en la esquina inferior derecha; al llegar aquí, el episodio termina.

Si durante el episodio el agente cae en el acantilado, este regresa al estado inicial s_0 y continua con el episodio.

1.1.3. Función de recompensa

La señal de recompensa $R(s, a, s')$ se define como:

$$R(s, a, s') = \begin{cases} -100, & \text{si } s' \text{ es una celda de acantilado (cliff),} \\ -1, & \text{en cada transición válida que no alcance la meta ni el cliff,} \\ 0, & \text{al alcanzar el estado terminal } s_T. \end{cases}$$

De este modo, el agente está incentivado a llegar cuanto antes a la meta evitando caer en el precipicio.

1.2. Entorno experimental

Componente	**Descripción**
Sistema operativo	Ubuntu 24.04.2 LTS
Kernel Linux	6.10.3-061003-generic
CPU	AMD Ryzen 7 5825U (8 núcleos, 16 hilos, hasta 4.5 GHz)
GPU integrada	AMD Radeon Graphics (Barcelo)
Memoria RAM	16 GB DDR4
Intérprete de Python	Python 3. 9. 18

Cuadro 1: Entorno de hardware y software utilizado en los experimentos

2. Value Iteration

2.1. Descripción del algoritmo

La iteración por valor es un método de programación dinámica para resolver un Proceso de Decisión de Markov (MDP) y encontrar simultáneamente la función valor óptima V^* y la política óptima π^*. Se basa en la relación de Bellman óptima:

$$V^*(s) \;=\; \max_{a \in A} \sum_{s'} P(s' \mid s, a) \left[R(s, a, s') + \gamma \, V^*(s') \right],$$

donde:

- S es el conjunto de estados.
- A es el conjunto de acciones.
- $P(s' \mid s, a)$ es la probabilidad de transición de s a s' dado a.
- $R(s, a, s')$ es la recompensa recibida al transitar.
- $\gamma \in [0, 1)$ es el factor de descuento.

A continuación se presenta el pseudocódigo genérico de Value Iteration que se ha implementado en este proyecto, seguido de las decisiones de diseño adoptadas en la implementación de Python.

Algorithm 1 Value Iteration

Require: Conjunto de estados S, conjunto de acciones A, $P(s' \mid s, a)$ y $R(s, a, s')$, factor de descuento $\gamma \in [0, 1)$, umbral de convergencia $\varepsilon > 0$

Ensure: Función valor V y política óptima π

Inicializar $V(s) \leftarrow 0, \ \forall s \in S$
repeat
    $\Delta \leftarrow 0$
    for all $s \in S$ **do**
        $V_{\text{old}} \leftarrow V(s)$
        $V(s) \leftarrow \max_{a \in A} \sum_{s'} P(s' \mid s, a)\big[R(s, a, s') + \gamma \, V(s')\big]$
        $\Delta \leftarrow \max\big(\Delta, \ |V(s) - V_{\text{old}}|\big)$
    end for
until $\Delta \leq \varepsilon$
for all $s \in S$ **do**
    $\pi(s) \leftarrow \arg\max\limits_{a \in A} \sum\limits_{s'} P(s' \mid s, a)\big[R(s, a, s') + \gamma \, V(s')\big]$
end for
return V, π

Decisiones de diseño en la implementación Python

- **Cálculo de $Q(s,a)$ con manejo del estado terminal**: para calcular el valor de cada acción consideramos que si se ha llegado a un estado terminal, el término de arranque posterior (*bootstrap*) se anula:

$$Q(s,a) \;=\; \sum_{s'} p\left[r + \gamma\, V(s')\right] \quad \longrightarrow \quad \text{bootstrap} = 0 \text{ si } s' \text{ es terminal.}$$

  De esta forma, se garantiza que al terminar el episodio, no se incorporen erróneamente estimaciones de valor posteriores a la terminación.

- **Evaluación periódica de la política**: tras cada iteración de valor calculamos la recompensa media de la política actual en $N = 100$ episodios de longitud máxima $T_{\text{máx}} = 200$ (método `check_improvements`), tanto para monitorizar progresos como para registrar la mejor recompensa y la iteración en que ocurre. Se fijan los valores de N y $T_{\text{máx}}$ para evitar que el algoritmo se detenga por un número excesivo de episodios, lo que podría ocurrir si la política converge a una política subóptima. En este caso, el algoritmo se detendría sin haber explorado adecuadamente el espacio de estados.

2.2. Experimentación

En esta sección se presentan los experimentos realizados para evaluar el rendimiento del algoritmo de Iteración de Valor en el entorno. Se analiza cómo diferentes parámetros del algoritmo afectan su capacidad para encontrar políticas óptimas, su convergencia y su eficiencia.

2.2.1. Experimento factor de descuento & umbral de convergencia

2.2.1.1 Diseño experimental

El objetivo de este experimento es analizar cómo los parámetros factor de descuento (γ) y umbral de convergencia (ϵ) afectan el rendimiento del algoritmo de iteración de valor.

Observación	El rendimiento y óptimalidad de la política encontrada por *Value Iteration* se ven afectados por los valores de γ y ϵ.
Planteamiento	Para cada pareja de valores de γ y ϵ, se compara la tasa de acierto (llegar al estado final), la recompensa media, número de pasos y tiempo de entrenamiento del algoritmo.
Hipótesis	Se espera que un mayor valor de γ conduzca a una política más óptima, mientras que un menor valor de ϵ permita una convergencia más rápida con una menor precisión.
Método	▪ Se elige un conjunto de valores para γ y ϵ: $\gamma \in \{0.5, 0.7, 0.9, 0.95, 0.99\}$ y $\epsilon \in \{1 \times 10^{-1}, 1 \times 10^{-2}, 1 \times 10^{-4}, 1 \times 10^{-8}\}$. ▪ Para cada combinación de γ y ϵ, se ejecuta el algoritmo *Value Iteration* en el entorno. ▪ Se evalúa la política obtenida probándola con 500 episodios. ▪ A diferencia de otros algoritmos que requieren de múltiples ejecuciones por su naturaleza estocástica, para Value Iteration basta con una única ejecución por combinación de parámetros, ya que es un algoritmo determinista que siempre converge a la misma política óptima para unos valores dados de γ y ϵ.

Cuadro 2: Value Iteration - Experimento 1 - Factor de descuento & umbral de convergencia

2.2.1.2 Resultados

A continuación se presenta un análisis detallado de las diferentes métricas evaluadas en el experimento. La Tabla 3 muestra un resumen de los resultados más relevantes:

γ	ϵ	Tasa éxito	Recompensa	Pasos	Tiempo (s)
0.99	10^{-8}	1.000	-63.422	63.4	31.77
0.95	10^{-4}	1.000	-64.190	64.2	13.74
0.90	10^{-2}	1.000	-63.920	63.9	7.24
0.70	10^{-1}	0.666	-144.702	144.7	2.44
0.50	10^{-1}	0.594	-152.758	152.8	1.61

Cuadro 3: Resultados representativos del experimento con Value Iteration

Tasa de éxito

La Figura 1 muestra la tasa de éxito para las diferentes combinaciones de parámetros:

- Valores altos de γ (≥ 0.9) logran tasas de éxito perfectas (100 %) o casi perfectas.
- Con $\gamma = 0.7$:
  - 100 % de éxito con $\epsilon \leq 10^{-4}$
  - Cae al 89.6 % con $\epsilon = 10^{-2}$
  - Solo 66.6 % con $\epsilon = 10^{-1}$
- Con $\gamma = 0.5$:
  - 100 % de éxito solo con $\epsilon = 10^{-8}$
  - Deterioro progresivo: 96.4 % (10^{-4}), 77.4 % (10^{-2}), 59.4 % (10^{-1})

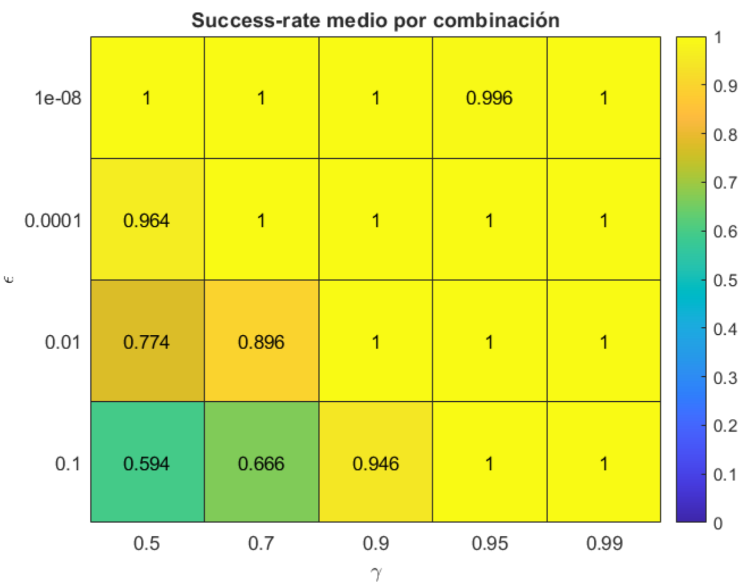


Figura 1: Tasa de éxito para diferentes valores de γ y ϵ en el algoritmo de iteración de valor.

Recompensa media

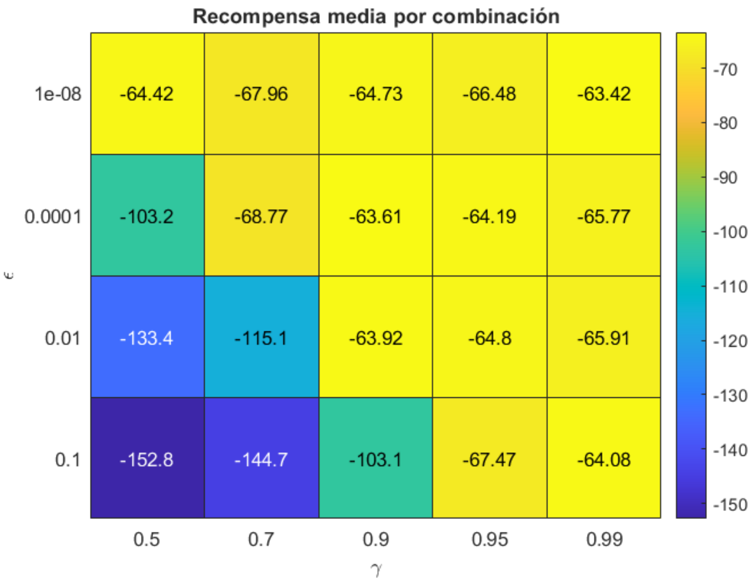


Figura 2: Recompensa media para diferentes valores de γ y ϵ.

El análisis de las recompensas medias muestra patrones claros:

- Con $\gamma = 0.99$:
  - Mejor recompensa global (-63.422) con $\epsilon = 10^{-8}$
  - Rendimiento consistente en todo el rango de ϵ: -65.766 (10^{-4}), -65.906 (10^{-2}), -64.080 (10^{-1})
- Con $\gamma = 0.9$:
  - Recompensas similares con $\epsilon \leq 10^{-2}$: -64.734 (10^{-8}), -63.608 (10^{-4}), -63.920 (10^{-2})
  - Deterioro significativo con $\epsilon = 10^{-1}$: -103.114
- Valores bajos de γ:
  - $\gamma = 0.7$: recompensas entre -67.962 y -144.702
  - $\gamma = 0.5$: peor rendimiento, llegando a -152.758 con $\epsilon = 10^{-1}$

Número de pasos

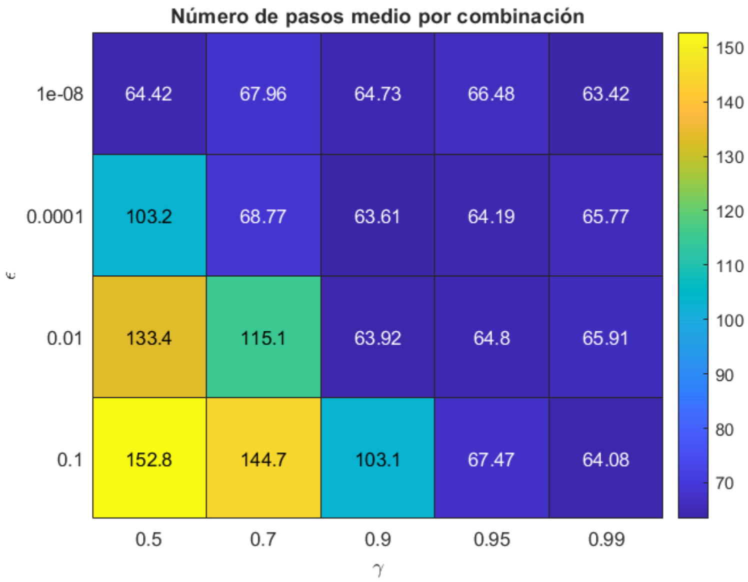


Figura 3: Número medio de pasos para diferentes valores de γ y ϵ.

El análisis de los pasos necesarios revela patrones similares a las recompensas:

- Mejores resultados con $\gamma \geq 0.9$:
  - $\gamma = 0.99$: consistentemente eficiente (63.4-65.9 pasos)
  - $\gamma = 0.95$: rendimiento similar (64.2-67.5 pasos)
  - $\gamma = 0.90$: eficiente con $\epsilon \leq 10^{-2}$ (63.6-64.7 pasos)
- Deterioro con valores bajos de γ:
  - $\gamma = 0.7$: aumento significativo (68.0-144.7 pasos)
  - $\gamma = 0.5$: peor rendimiento (64.4-152.8 pasos)
- Efectos de ϵ:
  - Mayor impacto en γ bajos
  - Menor influencia en γ altos (≥ 0.9)

Tiempo de entrenamiento

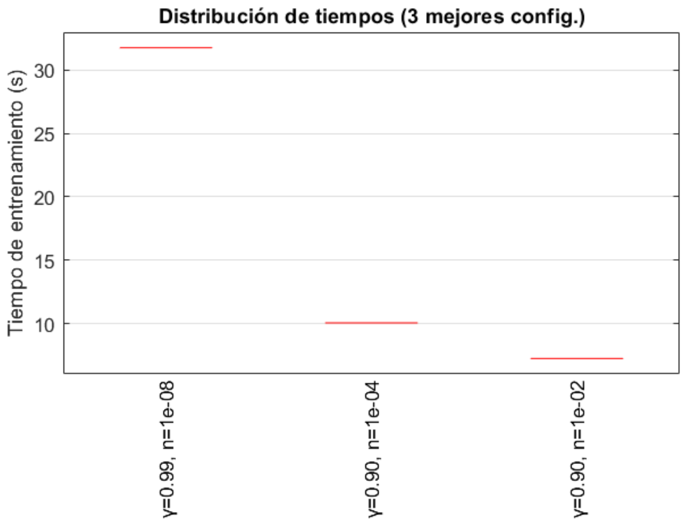


Figura 4: Tiempo de entrenamiento para diferentes valores de γ y ϵ.

El análisis del tiempo de entrenamiento revela patrones importantes:

- Impacto de ϵ:
  - $\epsilon = 10^{-8}$: tiempos más altos (5.03-31.77 segundos)
  - $\epsilon = 10^{-4}$: tiempos moderados (3.53-20.23 segundos)
  - $\epsilon = 10^{-2}$: tiempos bajos (2.38-14.90 segundos)
  - $\epsilon = 10^{-1}$: tiempos mínimos (1.61-11.95 segundos)
- Efecto de γ:
  - $\gamma = 0.99$: mayor rango de tiempos (11.95-31.77 segundos)
  - $\gamma = 0.95$: rango intermedio (8.60-21.52 segundos)
  - $\gamma = 0.90$: rango moderado (5.60-15.19 segundos)
  - Valores bajos ($\gamma \leq 0.7$): tiempos menores en general
- Relaciones observadas:
  - Crecimiento exponencial del tiempo al reducir ϵ
  - Incremento aproximadamente lineal al aumentar γ
  - Mayor sensibilidad al valor de ϵ que al de γ

2.2.1.3 Conclusiones del experimento

Del análisis experimental se pueden extraer las siguientes conclusiones:

1. El factor de descuento (γ) tiene un impacto crítico en el rendimiento del algoritmo:
    - Valores altos ($\gamma \geq 0.9$) son necesarios para políticas óptimas
    - Mejora significativa en todas las métricas al aumentar γ
2. El umbral de convergencia (ϵ) afecta principalmente al tiempo de entrenamiento:
    - Valores más bajos producen mejores políticas pero requieren más tiempo
    - Relación exponencial entre precisión y tiempo de computación
3. La mejor combinación considerando todas las métricas es $\gamma = 0.99$ y $\epsilon = 10^{-8}$:
    - 100 % de tasa de éxito
    - Mejor recompensa media (-63.422)
    - Menor número de pasos (63.422)
    - Tiempo de entrenamiento aceptable (31.77 segundos)
4. Alternativas según prioridades:
    - Balance rendimiento-tiempo: $\gamma = 0.99$, $\epsilon = 10^{-4}$
    - Mínimo tiempo: $\gamma = 0.99$, $\epsilon = 10^{-2}$

Estos resultados demuestran que el algoritmo de Iteración de Valor es capaz de encontrar políticas óptimas para el entorno Cliff Walking, especialmente cuando se utiliza un factor de descuento alto y un umbral de convergencia suficientemente bajo, permitiendo al agente considerar adecuadamente las recompensas futuras y converger a una solución precisa.

3. Direct Estimation

3.1. Descripción del algoritmo

La versión de Estimación Directa que se ha implementado corresponde a un *Método Monte Carlo basado en modelo*, en el cual:

1. Se recolectan muestras de transición (s, a, s', r) jugando acciones aleatorias.

2. Se estiman empíricamente

$$\hat{T}(s, a, s') = \frac{N(s, a, s')}{\sum_u N(s, a, u)},$$
$$\hat{R}(s, a, s') = \frac{\sum_{i=1}^{N(s,a,s')} r_i(s, a, s')}{N(s, a, s')}.$$

   donde $N(s, a, s')$ representa el número de veces que se ha observado la transición del estado s al estado s' tomando la acción a, y $r_i(s, a, s')$ es la recompensa obtenida en la i-ésima transición de s a s' mediante la acción a.

3. Se aplica iteración de valor sobre el MDP estimado $(\hat{T}, \hat{R}, \gamma)$ para obtener

$$V^*(s) = \max_a \sum_{s'} \hat{T}(s, a, s') \Big[\hat{R}(s, a, s') + \gamma\, V^*(s')\Big],$$

   y de ahí la política óptima

$$\pi^*(s) = \arg\max_a \sum_{s'} \hat{T}(s, a, s') \Big[\hat{R}(s, a, s') + \gamma\, V^*(s')\Big].$$

A continuación se presenta el pseudocódigo genérico de Direct Estimation que se ha implementado en este proyecto, seguido de las decisiones de diseño adoptadas en la implementación de Python.

Algorithm 2 Estimación Directa (Model-based Monte Carlo)

Require: Factor de descuento γ, número de trayectorias N, tolerancia ε, máximo de iteraciones K

Inicializar contadores de transición y recompensa vacíos
Inicializar $V(s) \leftarrow 0$ para todo estado s
for $t = 1, \dots, K$ **do**
  Recolectar datos:
  for $i = 1, \dots, N$ **do**
    Jugar un paso aleatorio, obtener (s, a, s', r)
    Incrementar $N(s, a, s')$ y acumular recompensa r para el par (s, a, s')
  end for
  Ajustar modelo:
  for all (s, a) **do**
    $\hat{T}(s, a, s') \leftarrow \dfrac{N(s, a, s')}{\sum_u N(s, a, u)}$
    $\hat{R}(s, a, s') \leftarrow \dfrac{\sum_{i=1}^{N(s,a,s')} r_i(s, a, s')}{N(s, a, s')}$
  end for
  Iteración de valor:
  $\Delta \leftarrow 0$
  for all s **do**
    for all a **do**
      $Q(s, a) \leftarrow \sum_{s'} \hat{T}(s, a, s') \big[\hat{R}(s, a, s') + \gamma V(s') \big]$
    end for
    $V_{\text{nuevo}}(s) \leftarrow \text{máx}_a Q(s, a)$
    $\Delta \leftarrow \text{máx}\{\Delta, |V_{\text{nuevo}}(s) - V(s)|\}$
    $V(s) \leftarrow V_{\text{nuevo}}(s)$
  end for
  if $\Delta < \varepsilon$ **then break**
  end if
end for
return V, y derivar $\pi^*(s) = \arg \text{máx}_a Q(s, a)$

Decisiones de diseño en la implementación Python

- **Criterio de parada por paciencia.** Además de la tolerancia en la iteración de valor, detenemos el entrenamiento si no hay mejora en la recompensa media durante `PATIENCE` iteraciones, midiendo esto con la función `check_improvements()`.

3.2. Experimentación

En el caso de estimación directa se analizan diferentes parámetros como el factor de descuento, el número de trayectorias, episodios de entrenamiento y el número máximo de iteraciones sin mejora para detener el entrenamiento (*patience*).

3.2.1. Experimento factor de descuento & número de trayectorias

3.2.1.1 Diseño experimental

El objetivo de este experimento es analizar el rendimiento del algoritmo *Direct Estimation* en función del factor de descuento y el número de trayectorias.

Observación	El rendimiento del algoritmo varía con el factor de descuento y el número de trayectorias.
Planteamiento	Para cada combinación de γ y *número de trayectorias*, se compara la tasa de acierto (llegar al estado final), la recompensa media, número de pasos y tiempo de entrenamiento del algoritmo.
Hipótesis	Un mayor factor de descuento y un mayor número de trayectorias mejorarán el rendimiento del algoritmo.
Método	▪ Se fijan 500 episodios de entrenamiento y un número máximo de iteraciones sin mejora para detener el entrenamiento (*patience*) de 100. ▪ Se eligen los siguientes valores para γ y *número de trayectorias*: $\gamma \in \{0.5, 0.7, 0.9, 0.95, 0.99\}$ y *número de trayectorias* $\in \{10, 100, 500, 1000\}$. ▪ Para cada combinación de γ y *número de trayectorias*, se ejecuta el algoritmo *Direct Estimation* en el entorno. ▪ Se evalúa la política obtenida probándola con 500 episodios. ▪ Se repite el proceso para cada combinación de γ y *número de trayectorias* 10 veces.

Cuadro 4: Direct Estimation - Experimento 1 - Factor de descuento & número de trayectorias

3.2.1.2 Resultados

3.2.2. Experimento número de episodios de entrenamiento

3.2.2.1 Diseño experimental

El objetivo de este experimento es analizar el rendimiento del algoritmo *Direct Estimation* en función del número de episodios de entrenamiento.

Observación	El rendimiento del algoritmo varía con el número de episodios de entrenamiento.
Planteamiento	Se compara la tasa de acierto (llegar al estado final), la recompensa media, número de pasos y tiempo de entrenamiento del algoritmo para diferentes números de episodios de entrenamiento.
Hipótesis	Un mayor número de episodios de entrenamiento mejorará el rendimiento del algoritmo.
Método	▪ Se fijan los mejores valores para γ y *número de trayectorias* del experimento anterior. ▪ Se eligen los siguientes valores para *número de episodios de entrenamiento*: *número de episodios de entrenamiento* $\in \{100, 500, 1000, 5000\}$. ▪ Para cada *número de episodios*, se ejecuta el algoritmo *Direct Estimation* en el entorno. ▪ Se evalúa la política obtenida probándola con 500 episodios. ▪ Se repite el proceso para cada *número de episodios* 10 veces.

Cuadro 5: Direct Estimation - Experimento 2 - Número de episodios de entrenamiento

3.2.2.2 Resultados

3.2.3. Experimento Patience

3.2.3.1 Diseño experimental

El objetivo de este experimento es ver si el algoritmo *Direct Estimation* puede detenerse antes de que se alcance el número máximo de episodios de entrenamiento, lo que podría reducir el tiempo de entrenamiento sin afectar el rendimiento.

Observación	El algoritmo *Direct Estimation* tiene fases largas de entrenamiento sin mejora.
Planteamiento	Se compara la tasa de acierto (llegar al estado final), la recompensa media, número de pasos y tiempo de entrenamiento del algoritmo para diferentes valores de iteraciones sin mejora para detener el entrenamiento (*patience*).
Hipótesis	Un valor de *patience* más bajo permitirá al algoritmo detenerse antes y reducir el tiempo de entrenamiento sin afectar significativamente el rendimiento.
Método	▪ Se fijan los mejores valores para γ, *número de trayectorias* y *número de episodios de entrenamiento* de los experimentos anteriores. ▪ Se eligen los siguientes valores para *patience*: *patience* $\in \{10, 100, 1000\}$. ▪ Para cada *patience*, se ejecuta el algoritmo *Direct Estimation* en el entorno. ▪ Se evalúa la política obtenida probándola con 500 episodios. ▪ Se repite el proceso para cada *patience* 10 veces.

Cuadro 6: Direct Estimation - Experimento 3 - Patience

3.2.3.2 Resultados

4. Q-learning

4.1. Descripción del algoritmo

Q-learning es un algoritmo de aprendizaje por refuerzo. La característica fundamental de Q-learning es su capacidad para aprender de forma off-policy, es decir, puede aprender la política óptima mientras sigue una política de exploración diferente (como ε-greedy). El algoritmo actualiza iterativamente sus estimaciones $Q(s,a)$ utilizando la ecuación de Bellman. A medida que el aprendizaje progresa, las estimaciones de Q convergen hacia los valores óptimos, permitiendo derivar la política óptima como $\pi^*(s) = \arg\max_a Q(s,a)$.

A continuación se presenta el pseudocódigo genérico de Direct Estimation que se ha implementado en este proyecto, seguido de las decisiones de diseño adoptadas en la implementación de Python.

Algorithm 3 Q-Learning

Inicializar $Q(s,a) \leftarrow 0$ para todo $s \in S$, $a \in A$
for episodio $\leftarrow 1$ **to** $N_{\text{episodios}}$ **do**
  $\varepsilon \leftarrow \text{máx}(\varepsilon_{\text{mín}}, \varepsilon_0 \cdot \text{decay}^{\text{episodio}})$
  Inicializar $s \leftarrow s_0$
  for $t \leftarrow 1$ **to** $T_{\text{máx}}$ **do**
    if rand() $\leq \varepsilon$ **then**
      $a \leftarrow$ acción aleatoria
    else
      $a \leftarrow \arg\text{máx}_{a'} Q(s,a')$
    end if
    Ejecutar a, observar r, s'
    td_target $\leftarrow r + \gamma\, \text{máx}_{a''} Q(s',a'')$
    td_error $\leftarrow$ td_target $- Q(s,a)$
    $Q(s,a) \mathrel{\widehat{\leftarrow}} Q(s,a) + \alpha$ td_error
    if s' es terminal **then**
      break
    end if
    $s \leftarrow s'$
  end for
end for

Decisiones de diseño en la implementación Python

- **Política ε-greedy con decaimiento:**

$$\pi(a \mid s) = \begin{cases} \frac{1}{|\mathcal{A}|}, & \text{con probabilidad } \varepsilon, \\ 1, & \text{si } a = \arg\text{máx}_{a'} Q(s,a'), \\ 0, & \text{en otro caso.} \end{cases}$$

Al inicio de cada episodio:

$$\varepsilon \leftarrow \text{máx}\big(\varepsilon_{\text{mín}},\ \varepsilon \cdot (\text{decay})^{\text{episodio}}\big).$$

- **Wrapper de recompensas customizado:** Se penaliza la acción `Izquierda` añadiendo una recompensa peor que la original, ya que en ningún caso interesa que el agente se desplace hacia la izquierda.

4.2. Experimentación

En esta sección se presentan los experimentos realizados para evaluar el rendimiento del algoritmo Q-Learning en el entorno. Se analiza cómo diferentes parámetros del algoritmo afectan su capacidad para encontrar políticas óptimas, su convergencia y su eficiencia.

4.2.1. Experimento factor de descuento & tasa de aprendizaje

4.2.1.1 Diseño experimental

El objetivo de este experimento es analizar cómo el factor de descuento y la tasa de aprendizaje afectan el rendimiento del algoritmo Q-Learning.

Observación	El factor de descuento (γ) y la tasa de aprendizaje (α) de exploración son parámetros críticos en el algoritmo Q-Learning.
Planteamiento	Para cada pareja de valores de γ y α, se compara la tasa de acierto (llegar al estado final), la recompensa media, número de pasos y tiempo de entrenamiento del algoritmo.
Hipótesis	Un mayor factor de descuento y una tasa de aprendizaje más lenta mejorarán el rendimiento del algoritmo.
Método	▪ Se fijan 1000 episodios de entrenamiento, ϵ inicial de 0.9, coeficiente de decaimiento de ϵ de 0.95, un coeficiente de decaimiento de la tasa de aprendizaje de 0.99 y una penalización de la acción "moverse a la izquierda" de -1. ▪ Se eligen los siguientes valores para γ y α: $\gamma \in \{0.5, 0.7, 0.9, 0.95, 0.99\}$ y $\alpha \in \{0.1, 0.2, 0.5, 0.8\}$. ▪ Para cada combinación de γ y α, se ejecuta el algoritmo Q-Learning en el entorno. ▪ Se evalúa la política obtenida probándola con 500 episodios. ▪ Se repite el proceso para cada combinación de γ y *decay* 20 veces para obtener una muestra representativa (debido a la estocasticidad del entorno).

Cuadro 7: Q-Learning - Experimento 1 - Factor de descuento & tasa de aprendizaje

4.2.1.2 Resultados

Los resultados del experimento se han representado mediante heatmaps para facilitar la visualización de los datos. En cada gráfico, el eje X representa el factor de descuento (γ) y el eje Y representa la tasa de aprendizaje (α). Los colores indican el valor de la métrica correspondiente. Cada valor representa la media de las 20 ejecuciones del algoritmo (cada ejecución está representada por la media de 500 episodios) de la correspondiente combinación de parámetros.

Tasa de Éxito

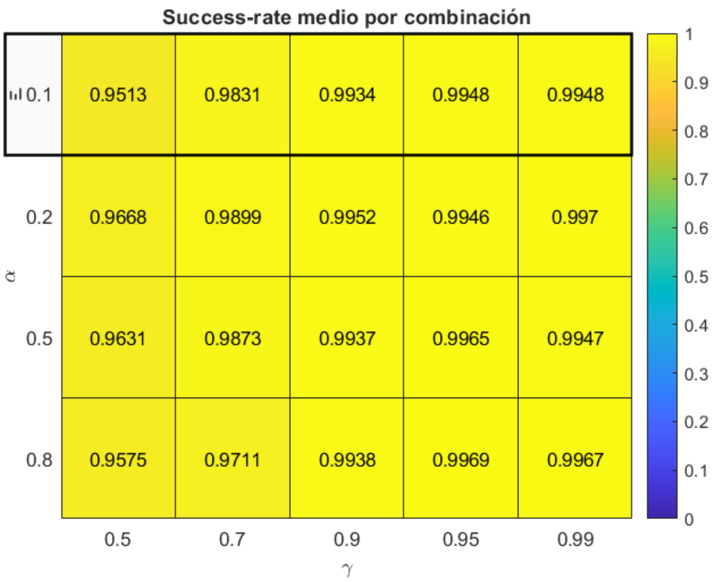


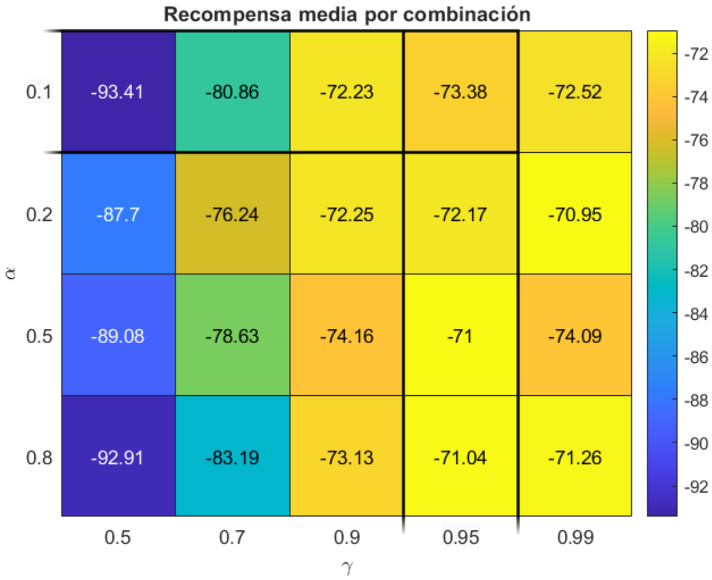
Figura 5: Tasa de éxito para diferentes combinaciones de α y γ

En general, la tasa de éxito en este caso es bastante alta para todas las combinaciones de valores. Sin embargo, puede observarse una pequeña tendencia. Conforme aumenta el factor de descuento (γ), el porcentaje de éxito se aproxima a 1.

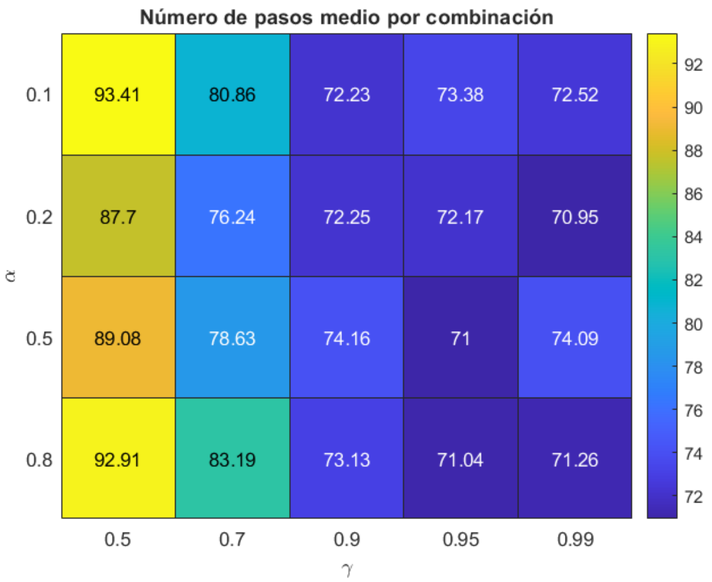
- Con $\gamma = 0.5$, la tasa de éxito se mantiene alrededor del 95-96 %
- Para $\gamma \geq 0.9$, la tasa de éxito supera consistentemente el 99 %

La mejor combinación se alcanza con $\gamma = 0.99$ y $\alpha = 0.2$, logrando una tasa de éxito del 99.7 %

Recompensa media y número de pasos medios



(a) Recompensa media para diferentes combinaciones de α y γ



(b) Número de pasos para diferentes combinaciones de α y γ

Figura 6: Análisis de la recompensa media y número de pasos

El análisis de la recompensa media y el número medio de pasos revela que:

- Las recompensas mejoran significativamente (son menos negativas) al aumentar γ y, por lo tanto, el número de pasos también se reduce.
- La tasa de aprendizaje no tiene mucha influencia en esta cambinación de parámetros.
- Con $\gamma = 0.5$, las recompensas medias oscilan entre -93 y -87, representando trayectorias más largas.

- Las mejores recompensas se obtienen con $\gamma \geq 0.9$, alcanzando valores cercanos a -70.

La combinación óptima es $\gamma = 0.99$, $\alpha = 0.2$ y logra una recompensa media de -70.95. Corresponde con la combinación que obtiene mejor tasa de éxito.

Se puede observar que el número medio de pasos corresponde exactamente con la recompensa media en cada configuración. Esto nos indica que el agente no cae por el barranco en ningún momento, ya que sino la recompensa media sería más negativa (-100 de recompensa por caer en 1 paso, perderia la correspondencia de -1 de recompensa por cada paso).

Tiempo de entrenamiento

Para el tiempo de entrenamiento, se han elegido las tres combinaciones que mejor rendimiento han dado.

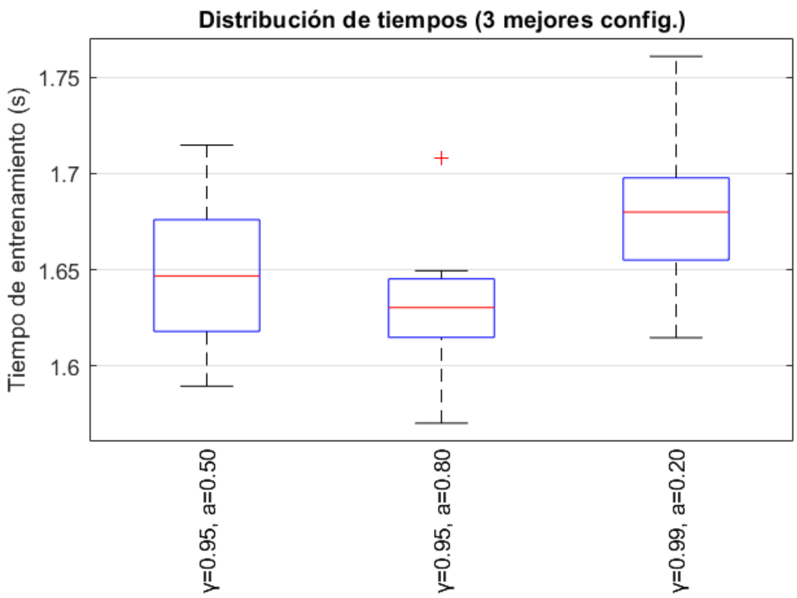


Figura 7: Tiempo de entrenamiento para diferentes combinaciones de α y γ

El análisis del tiempo de ejecución muestra que para las tres combinaciones, el tiempo se mueve entorno a los 1.65 segundos. Hay poca diferencia entre ellos, con una pequeña variabilidad en las colas.

Tabla resumen

γ	α	Métrica	Media	Intervalo de Confianza del 95 %
0.95	0.50	Success-rate	0.997	[0.993, 1.000]
		Rew. media	-71.000	[-74.340, -67.660]
		Steps medios	71.0	[67.7, 74.3]
		Time (s)	1.65	[1.62, 1.67]
0.95	0.80	Success-rate	0.997	[0.994, 1.000]
		Rew. media	-71.039	[-73.977, -68.101]
		Steps medios	71.0	[68.1, 74.0]
		Time (s)	1.63	[1.61, 1.65]
0.99	0.20	Success-rate	0.997	[0.995, 0.999]
		Rew. media	-70.945	[-73.840, -68.051]
		Steps medios	70.9	[68.1, 73.8]
		Time (s)	1.68	[1.66, 1.71]

Cuadro 8: Resultados para combinaciones seleccionadas de γ y α

Conclusiones

Del análisis experimental se pueden extraer las siguientes conclusiones:

1. El factor de descuento (γ) tiene un impacto más significativo que la tasa de aprendizaje (α) en el rendimiento del algoritmo con la configuración de parámetro definida.

2. Los valores óptimos se encuentran en el rango de $\gamma \geq 0.9$.

3. La tasa de aprendizaje óptima parece estar entre 0.2 y 0.5.

4. La mejor combinación general considerando todas los métricas es $\gamma = \mathbf{0.99}$ y $\alpha = \mathbf{0.2}$, que proporciona:

   - Una tasa de éxito del 99.7 %.
   - Una recompensa media de -70.95.
   - Un tiempo de entrenamiento de 1.68 segundos.

5. Los resultados obtenidos confirman la hipótesis inicial de que un mayor factor de descuento y una tasa de aprendizaje más lenta mejoran el rendimiento del algoritmo.

Aunque los parámetros seleccionados no minimizan el tiempo de entrenamiento, las diferencias temporales observadas son mínimas. En este estudio se ha priorizado la optimización de la tasa de aciertos y la recompensa sobre el tiempo de entrenamiento, ya que estos criterios son más relevantes para evaluar la efectividad del algoritmo.

4.2.2. Experimento tasa de exploración (ϵ) & decaimiento de la tasa de exploración (ϵ)

4.2.2.1 Diseño experimental

El objetivo de este experimento es analizar cómo la tasa de exploración y su decaimiento afectan el rendimiento del algoritmo Q-Learning.

Observación	El rendimiento y óptimalidad de la política encontrada por Q-Learning se ven afectados por la tasa de exploración y su decaimiento.
Planteamiento	Para cada combinación de ϵ y decaimiento de ϵ, se compara la tasa de acierto (llegar al estado final), la recompensa media, número de pasos y tiempo de entrenamiento del algoritmo.
Hipótesis	Un mayor valor de ϵ y un decaimiento moderado mejorarán el rendimiento del algoritmo.
Método	▪ Se fijan 1000 episodios de entrenamiento, un coeficiente de decaimiento de la tasa de aprendizaje de 0.99, una penalización de la acción "moverse a la izquierda" de -1. y los mejores valores para γ y α del experimento anterior. ▪ Se eligen los siguientes valores para ϵ y decaimiento de ϵ: $\epsilon \in \{0.7, 0.9, 0.95, 0.99\}$ y decaimiento de $\epsilon \in \{0.8, 0.9, 0.95, 0.99\}$. ▪ Para cada combinación de ϵ y decaimiento de ϵ, se ejecuta el algoritmo Q-Learning en el entorno. ▪ Se evalúa la política obtenida probándola con 500 episodios. ▪ Se repite el proceso para cada combinación de ϵ y decaimiento de ϵ 20 veces para obtener una muestra representativa (debido a la estocasticidad del entorno).

Cuadro 9: Q-Learning - Experimento 2 - Tasa de exploracion & decaimiento de la tasa de exploracion

4.2.2.2 Resultados

Los resultados del experimento se han representado mediante heatmaps para facilitar la visualización de los datos. En cada gráfico, el eje X representa la tasa de exploración (ϵ) y el eje Y representa el decaimiento de la tasa de exploración. Los colores indican el valor de la métrica correspondiente. Cada valor representa la media de las 20 ejecuciones del algoritmo (cada ejecución está representada por la media de 500 episodios) de la correspondiente combinación de parámetros.

Tasa de Éxito

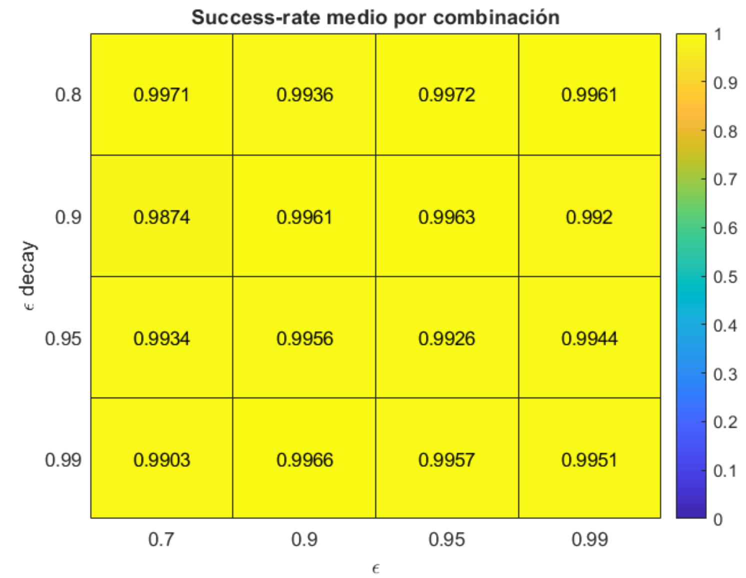
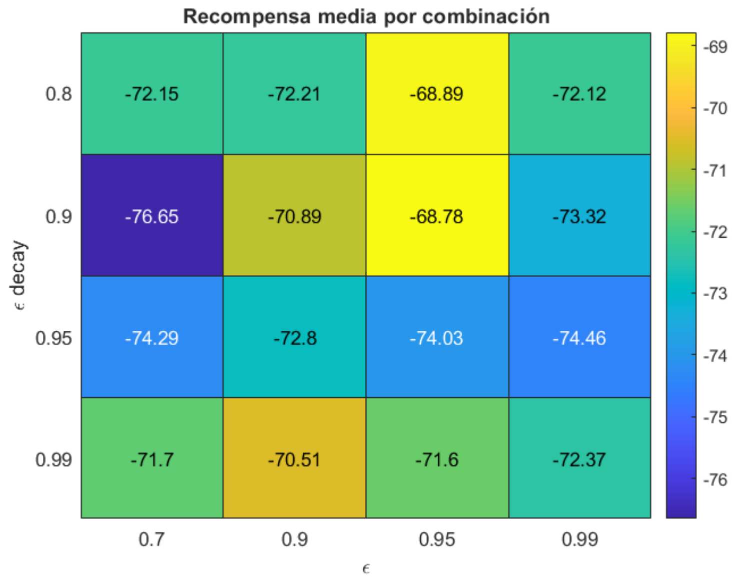


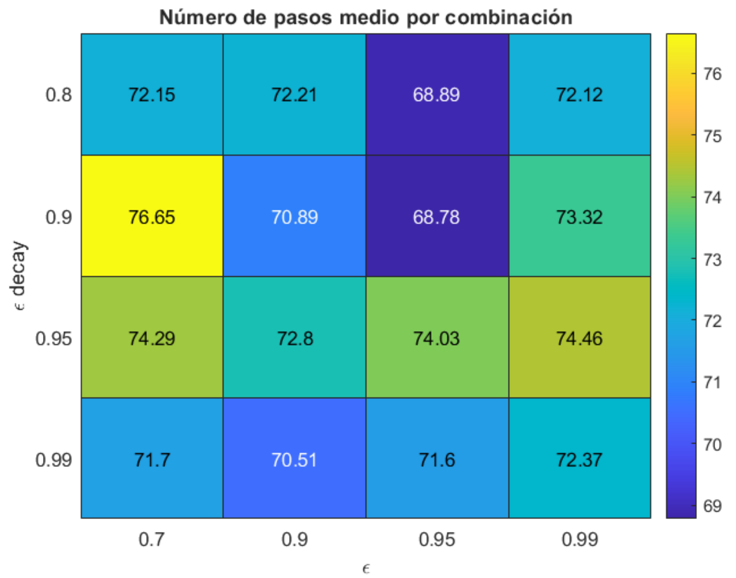
Figura 8: Tasa de éxito para diferentes combinaciones de ϵ y su decaimiento

La tasa de éxito también es bastante elevada en este caso y no se observa ninguna tendencia clara. Todos los valores se encuentran entre el 98.7 % y el 99.7 %.
La mejor combinación se alcanza con $\epsilon = 0.95$ y su decaimiento $= 0.8$, logrando una tasa de éxito del 99.72 %.

Recompensa media y número de pasos medios



(a) Recompensa media para diferentes combinaciones de ϵ y su decaimiento



(b) Número de pasos para diferentes combinaciones de ϵ y su decaimiento

Figura 9: Análisis de la recompensa media y número de pasos

El análisis de la recompensa media y el número medio de pasos tampoco revela una tendencia clara. Lo que si que se puede observar es que todas las configuraciones se encuentran entre -76.65 y -68.79 de recompensa, lo que indica que ninguna de ellas se desvia ni obtiene resultados muy negativos.

Se puede observar también que el número medio de pasos corresponde exactamente con la recompensa media en cada configuración. Esto nos indica que el agente no cae por el barranco en ningún momento, ya que sino la recompensa media sería más negativa.

Las combinaciones óptimas son $\epsilon = 0.95$ y su decaimiento $= 0.8$, logrando una recompensa media de -68.89, y $\epsilon = 0.95$ y su decaimiento $= 0.9$, logrando una recompensa media de -68.78.

Tiempo de entrenamiento

Para el tiempo de entrenamiento, se han elegido las tres combinaciones que mejor rendimiento han dado.

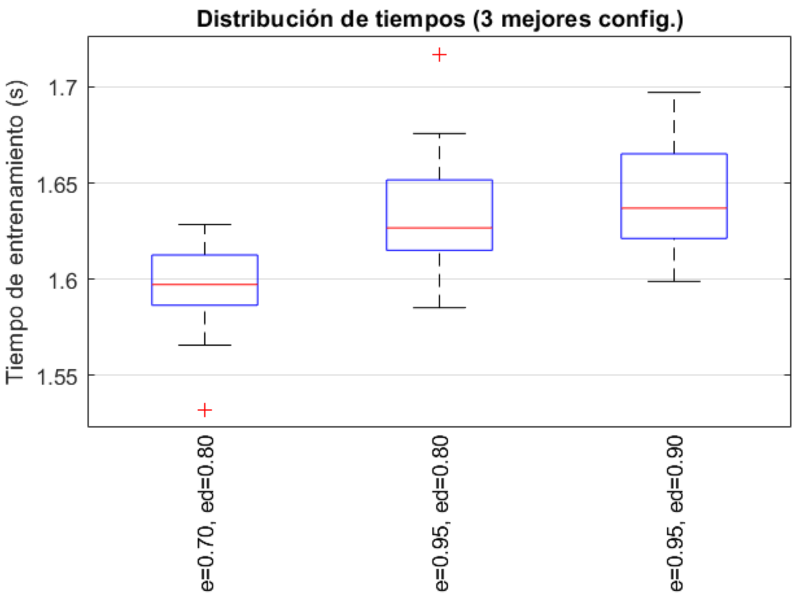


Figura 10: Tiempo de entrenamiento para diferentes combinaciones de ϵ y su decaimiento

El análisis del tiempo de ejecución muestra que para las tres combinaciones, el tiempo se mueve entorno a los 1.62 segundos. Hay poca diferencia entre ellos, con una pequeña variabilidad en las colas.

Tabla resumen

ϵ	*decay*	Métrica	Media	Intervalo de Confianza del 95 %
0.70	0.80	Success-rate	0.997	[0.995, 0.999]
		Rew. media	-72.150	[-75.391, -68.910]
		Steps medios	72.2	[68.9, 75.4]
		Time (s)	1.60	[1.58, 1.61]
0.95	0.80	Success-rate	0.997	[0.995, 0.999]
		Rew. media	-68.889	[-70.869, -66.909]
		Steps medios	68.9	[66.9, 70.9]
		Time (s)	1.63	[1.61, 1.66]
0.95	0.90	Success-rate	0.996	[0.990, 1.003]
		Rew. media	-68.782	[-72.091, -65.473]
		Steps medios	68.8	[65.5, 72.1]
		Time (s)	1.64	[1.62, 1.66]

Cuadro 10: Resultados para combinaciones seleccionadas de ϵ y *decay*

Conclusiones

Del análisis experimental se pueden extraer las siguientes conclusiones:

1. La tasa de exploración y su decaimiento no parecen tener un impacto significativo en el rendimiento del algoritmo Q-Learning.

2. La mejor combinación general considerando todas los métricas es $\epsilon = \mathbf{0.95}$ y su decaimiento $= \mathbf{0.8}$, que proporciona:

   - Una tasa de éxito del 99.72 %.
   - Una recompensa media de -68.89.
   - Un tiempo de entrenamiento de 1.63 segundos.

3. Los resultados obtenidos confirman la hipótesis inicial de que un mayor valor de ϵ y un decaimiento moderado mejoran el rendimiento del algoritmo.

Aunque los parámetros seleccionados no minimizan el tiempo de entrenamiento, las diferencias temporales observadas son mínimas. En este estudio se ha priorizado la optimización de la tasa de aciertos y la recompensa sobre el tiempo de entrenamiento, ya que estos criterios son más relevantes para evaluar la efectividad del algoritmo.

4.2.3. Experimento tasa de aprendizaje & decaimiento de la tasa de aprendizaje

4.2.3.1 Diseño experimental

El objetivo de este experimento es analizar cómo la tasa de aprendizaje y su decaimiento afectan el rendimiento del algoritmo Q-Learning.

Observación	La tasa de aprendizaje (α) y su decaimiento son parámetros que influyen en la convergencia del algoritmo Q-Learning.
Planteamiento	Para cada combinación de α y su decaimiento, se compara la tasa de acierto (llegar al estado final), la recompensa media, número de pasos y tiempo de entrenamiento del algoritmo.
Hipótesis	Una tasa de aprendizaje lenta con un decaimiento gradual mejorará el rendimiento del algoritmo.
Método	▪ Se fijan 1000 episodios de entrenamiento, una penalización de la acción “moverse a la izquierda” de -1. valores de γ, y los mejores valores de γ, ϵ y su decaimiento de los experimentos anteriores. ▪ Se eligen los siguientes valores para α y su decaimiento: $\alpha \in \{0.1, 0.2, 0.5, 0.8\}$ y decaimiento de $\alpha \in \{0.95, 0.99, 0.995, 0.999\}$. ▪ Para cada combinación de α y su decaimiento, se ejecuta el algoritmo Q-Learning en el entorno. ▪ Se evalúa la política obtenida probándola con 500 episodios. ▪ Se repite el proceso para cada combinación de α y su decaimiento 20 veces para obtener una muestra representativa (debido a la estocasticidad del entorno). ▪ (en este experimento se refina el valor de α)

Cuadro 11: Q-Learning - Experimento 3 - Tasa de aprendizaje & decaimiento de la tasa de aprendizaje

4.2.3.2 Resultados

Los resultados del experimento se han representado mediante heatmaps para facilitar la visualización de los datos. En cada gráfico, el eje X representa la tasa de aprendizaje (α) y el eje Y representa el decaimiento de la tasa de aprendizaje. Los colores indican el valor de la métrica correspondiente. Cada valor representa la media de las 20 ejecuciones del algoritmo (cada ejecución está representada por la media de 500 episodios) de la correspondiente combinación de parámetros.

Tasa de Éxito

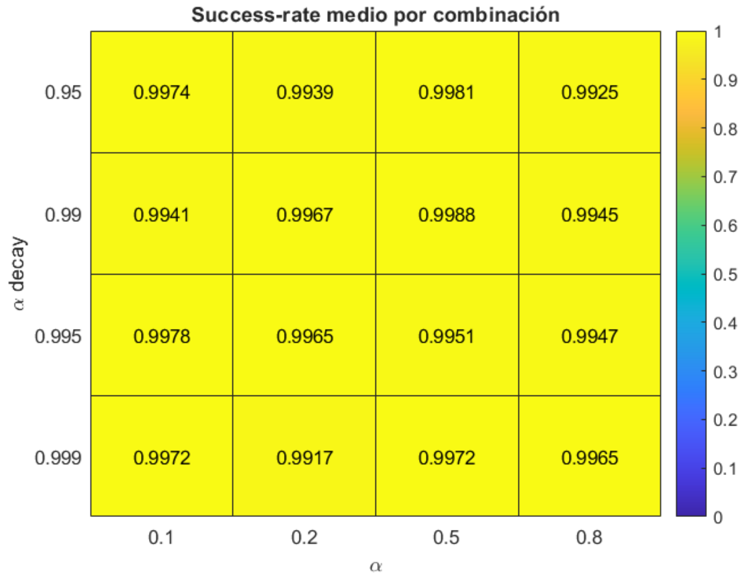
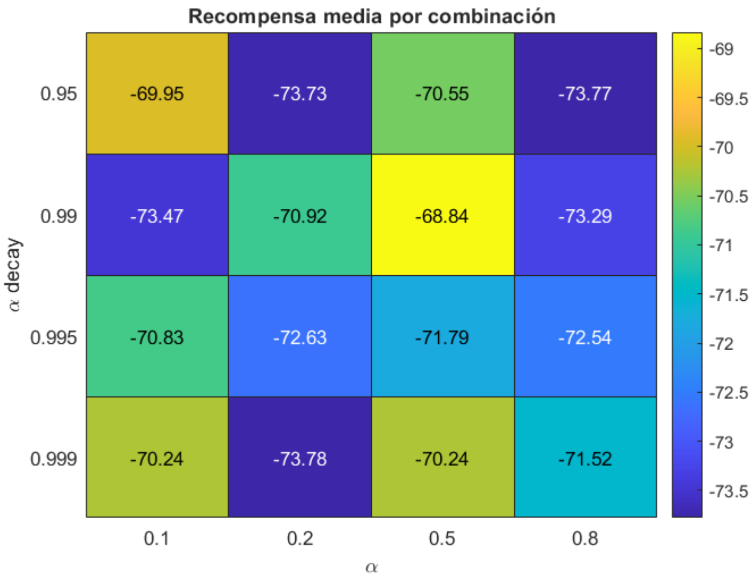


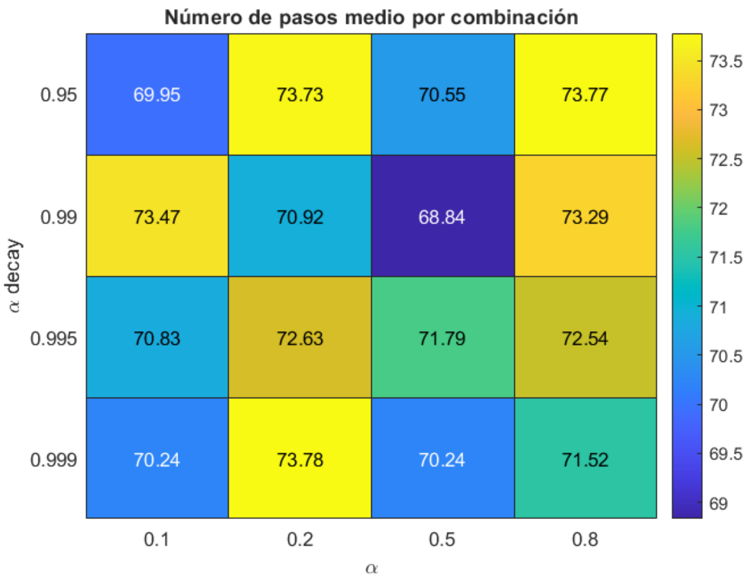
Figura 11: Tasa de éxito para diferentes combinaciones de α y su decaimiento

La tasa de éxito también es muy elevada en este caso y no se observa ninguna tendencia clara. Todos los valores se encuentran entre el 99.17 % y el 99.88 %.
A pesar de la poca diferencia, la mejor combinación se alcanza con $\alpha = 0.5$ y su decaimiento $= 0.99$, logrando una tasa de éxito del 99.88 %.

Recompensa media y número de pasos medios



(a) Recompensa media para diferentes combinaciones de α y su decaimiento



(b) Número de pasos para diferentes combinaciones de α y su decaimiento

Figura 12: Análisis de la recompensa media y número de pasos

El análisis de la recompensa media y el número medio de pasos tampoco revela una tendencia clara. Lo que si que se puede observar es que todas las configuraciones se encuentran entre -73.78 y -68.84 de recompensa, lo que indica que ninguna de ellas se desvia ni obtiene resultados muy negativos.

Se puede observar también que el número medio de pasos corresponde exactamente con la recompensa media en cada configuración. Esto nos indica que el agente no cae por el barranco en ningún momento, ya que sino la recompensa media sería más negativa.

Las combinación óptima es $\alpha = 0.5$ y su decaimiento $= 0.99$, logrando una recompensa media de -68.84.

Tiempo de entrenamiento

Para el tiempo de entrenamiento, se han elegido las tres combinaciones que mejor rendimiento han dado.

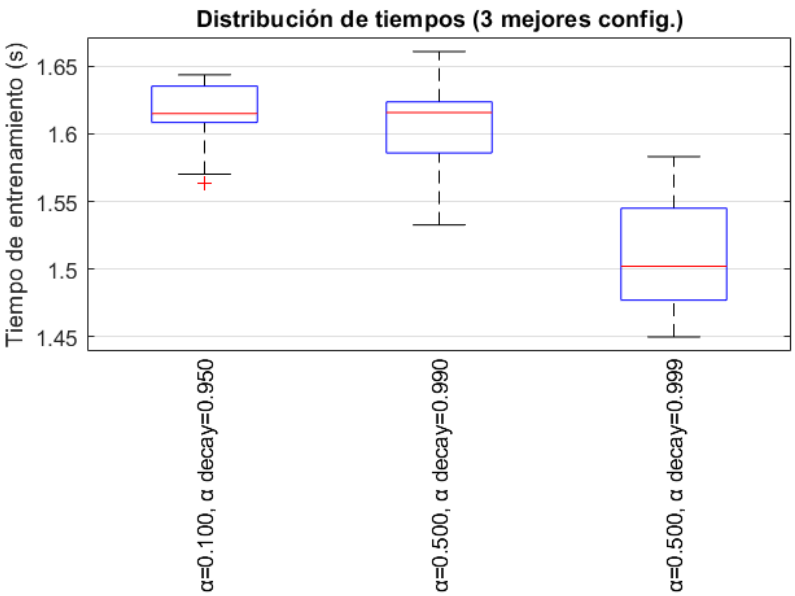


Figura 13: Tiempo de entrenamiento para diferentes combinaciones de α y su decaimiento

En este caso se puede observar como una de las combinaciones tiene un tiempo de entrenamiento más bajo, concretamente la que tiene un decaimiento más lento. Esto puede ser debdido a que αdisminuye más lentamente y por lo tanto el algoritmo converge más rápido. Aún así, el tiempo de entrenamiento varia en pocas decimas de segundo.

Tabla resumen

α	*decay*	Métrica	Media	Intervalo de Confianza del 95 %
0.10	0.95	Success-rate	0.997	[0.995, 1.000]
		Rew. media	-69.945	[-72.926, -66.964]
		Steps medios	69.9	[67.0, 72.9]
		Time (s)	1.62	[1.60, 1.63]
0.50	0.99	Success-rate	0.999	[0.997, 1.000]
		Rew. media	-68.836	[-71.366, -66.305]
		Steps medios	68.8	[66.3, 71.4]
		Time (s)	1.61	[1.58, 1.63]
0.50	0.999	Success-rate	0.997	[0.995, 1.000]
		Rew. media	-70.240	[-73.155, -67.326]
		Steps medios	70.2	[67.3, 73.2]
		Time (s)	1.51	[1.48, 1.54]

Cuadro 12: Resultados para combinaciones seleccionadas de α y *decay*

Conclusiones

Del análisis experimental se pueden extraer las siguientes conclusiones:

1. La tasa de aprendizaje refinada y su decaimiento no parecen tener un impacto significativo en el rendimiento del algoritmo Q-Learning.

2. La mejor combinación general considerando todas los métricas es $\alpha = \mathbf{0.5}$ y su decaimiento $= \mathbf{0.99}$, que proporciona:

   - Una tasa de éxito del 99.88 %.
   - Una recompensa media de -68.84.
   - Un tiempo de entrenamiento de 1.61 segundos.

3. Los resultados obtenidos difieren ligeramente de la hipótesis inicial, ya que ha dado mejor resultado una tasa de aprendizaje moderada con un decaimiento lento.

Aunque los parámetros seleccionados no minimizan el tiempo de entrenamiento, las diferencias temporales observadas son mínimas. En este estudio se ha priorizado la optimización de la tasa de aciertos y la recompensa sobre el tiempo de entrenamiento, ya que estos criterios son más relevantes para evaluar la efectividad del algoritmo.

4.2.4. Experimento número de episodios

4.2.4.1 Diseño experimental

El objetivo de este experimento es analizar cómo el número de episodios de entrenamiento afectan el rendimiento del algoritmo Q-Learning.

Observación	El número de episodios de entrenamiento es un parámetro crítico en el algoritmo Q-Learning.
Planteamiento	Se compara la tasa de acierto (llegar al estado final), la recompensa media, número de pasos y tiempo de entrenamiento del algoritmo para diferentes números de episodios de entrenamiento.
Hipótesis	Un mayor número de episodios de entrenamiento mejorará el rendimiento del algoritmo.
Método	▪ Se fijan los mejores valores para γ, α, decaimiento de α, ϵ, decaimiento de ϵ de los experimentos anteriores y una penalización de la acción "moverse a la izquierda" de -1. ▪ Se eligen los siguientes *números de episodios de entrenamiento*: $\{500, 1000, 5000, 10000\}$. ▪ Para cada *número de episodios*, se ejecuta el algoritmo Q-Learning en el entorno. ▪ Se evalúa la política obtenida probándola con 500 episodios. ▪ Se repite el proceso para cada número de episodios 20 veces para obtener una muestra maś representativa (debido a la estocasticidad del entorno).

Cuadro 13: Q-Learning - Experimento 4 - Número de episodios

4.2.4.2 Resultados

En este experimento, los resultados se han representado mediante boxplots ya que solo se está variando un parámetro.

Tasa de Éxito

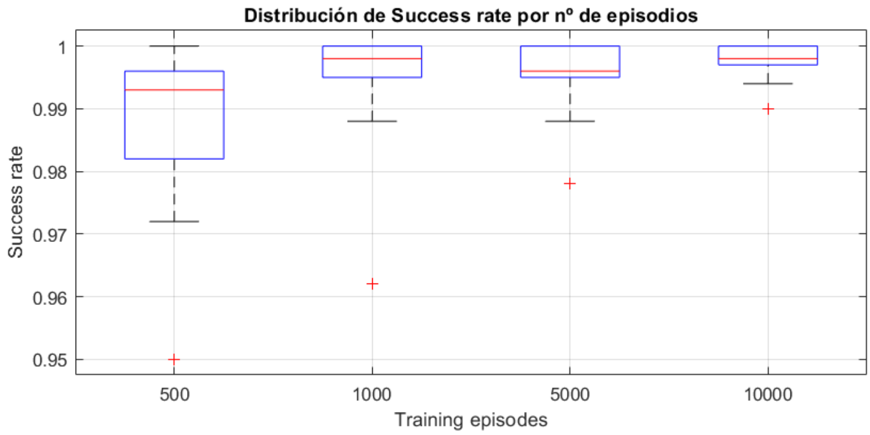
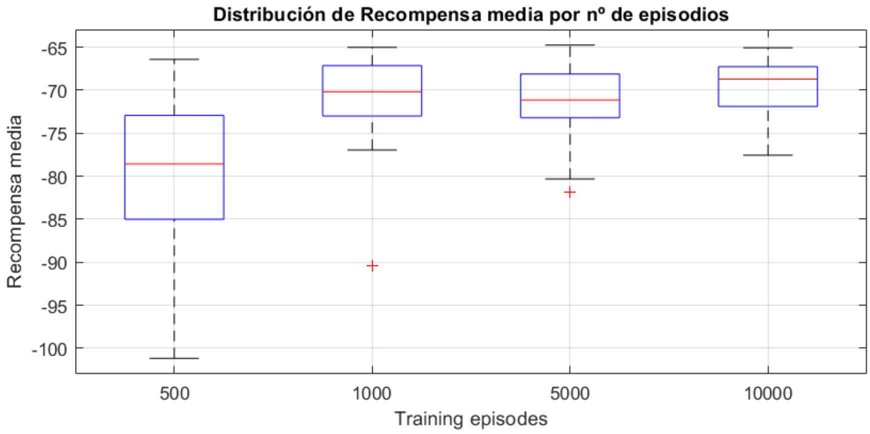


Figura 14: Tasa de éxito para diferentes números de episodios de entrenamiento

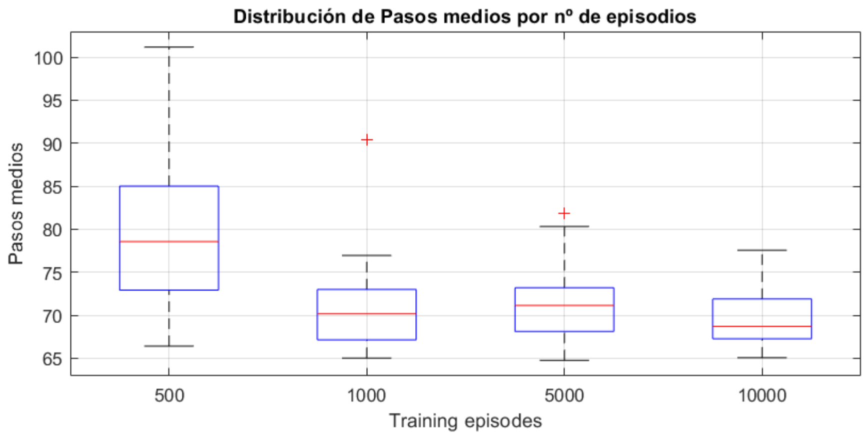
Se puede observar una ligera tendencia a mejorar la tasa de éxito al aumentar el número de episodios de entrenamiento. Además, se observa que la variabilidad de los valores también disminuye al aumentar el número de episodios.

Según estos resultados, el mejor número de episodios de entrenamiento es 10000, con una tasa media de éxito del 99.87 %.

Recompensa media y número de pasos medios



(a) Recompensa media para diferentes números de episodios de entrenamiento



(b) Número de pasos para diferentes números de episodios de entrenamiento

Figura 15: Análisis de la recompensa media y número de pasos

Con estos resultados se observa la misma tendencia que en la tasa de éxito. La recompensa media mejora conforme aumentan los episodios de entrenamiento y el número de pasos disminuyen. También se observa el ”salto”significativo de 500 a 1000 episodios de entrenamiento, donde los resultados mejora notablemente. A partir de 1000 episodios, la mejora es más gradual.

Tiempo de entrenamiento

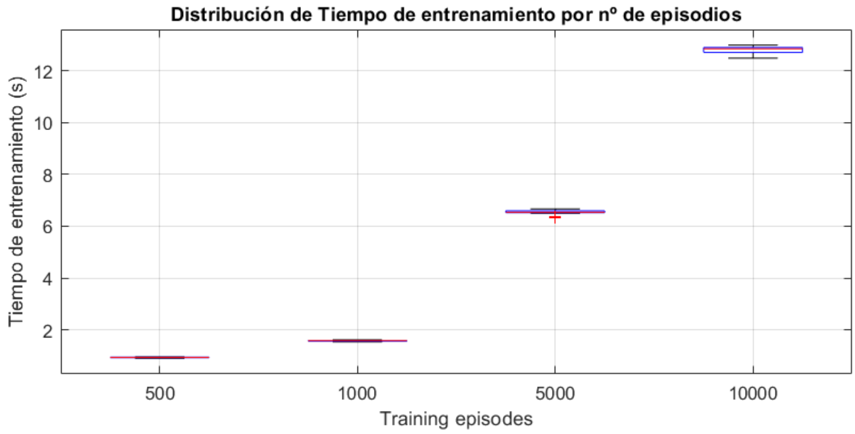


Figura 16: Tiempo de entrenamiento para diferentes números de episodios de entrenamiento

Como es lógico, el tiempo de entrenamiento aumenta conforme aumentan los episodios de entrenamiento. El más costoso es el de 10000 episodios, con un tiempo medio de 12.8 segundos, el doble que el de 5000 (6.54 segundos).

Tabla resumen

Episodios	Métrica	Media	Intervalo de Confianza del 95 %
500	Success-rate	0.988	[0.984, 0.992]
	Rew. media	-79.584	[-82.496, -76.671]
	Pasos medios	79.6	[76.7, 82.5]
	Tiempo (s)	0.93	[0.93, 0.94]
1000	Success-rate	0.996	[0.993, 0.998]
	Rew. media	-71.112	[-72.871, -69.354]
	Pasos medios	71.1	[69.4, 72.9]
	Tiempo (s)	1.58	[1.57, 1.59]
5000	Success-rate	0.996	[0.994, 0.998]
	Rew. media	-71.358	[-72.781, -69.934]
	Pasos medios	71.4	[69.9, 72.8]
	Tiempo (s)	6.54	[6.52, 6.57]
10000	Success-rate	0.998	[0.997, 0.999]
	Rew. media	-69.626	[-70.662, -68.591]
	Pasos medios	69.6	[68.6, 70.7]
	Tiempo (s)	12.80	[12.75, 12.84]

Cuadro 14: Resultados del algoritmo cantidades de episodios de entrenamiento

Conclusiones

Del análisis experimental se pueden extraer las siguientes conclusiones:

1. El número de episodios de entrenamiento afecta considerablemente al rendimiento del algoritmo.
2. El número de episodios de entrenamiento afecta considerablemente al tiempo de entrenamiento.
3. El número de episodios con mejor trade off entre rendimiento y tiempo de entrenamiento es **5000**, que proporciona:
   - Una tasa de éxito del 99.6 %.
   - Una recompensa media de -71.112.
   - Un tiempo de entrenamiento de 6.54 segundos.
4. Los resultados obtenidos confirman la hipótesis inicial de que un mayor número de episodios de entrenamiento mejora el rendimiento del algoritmo.

4.2.5. Experimento penalización de la acción izquierda

4.2.5.1 Diseño experimental

El objetivo de este experimento es analizar si penalizar acciones poco favorables afectan el rendimiento del algoritmo Q-Learning.

Observación	Moverse a la izquierda no es deseable para el agente en ningún momento, ya que no le acerca al objetivo. Por lo tanto, penalizarla con una recompensa menor que las demás acciones puede alterar el comportamiento del agente.
Planteamiento	Se compara la tasa de acierto (llegar al estado final), la recompensa media, número de pasos y tiempo de entrenamiento del algoritmo para diferentes penalizaciones a la acción “moverse a la izquierda”.
Hipótesis	Penalizar la acción “moverse a la izquierda” mejorará el rendimiento del algoritmo.
Método	▪ Se fijan los mejores valores para γ, α, decaimiento de α, ϵ, decaimiento de ϵ de los experimentos anteriores. ▪ Se eligen los siguientes valores para la penalización de la acción “moverse a la izquierda”: {-1, -2, -10, -50}. ▪ Para cada penalización, se ejecuta el algoritmo Q-Learning en el entorno. ▪ Se evalúa la política obtenida probándola con 500 episodios. ▪ Se repite el proceso para cada penalización 20 veces para obtener una muestra más significativa (debido a la estocasticidad del entorno).

Cuadro 15: Q-Learning - Experimento 5 - Penalización de la acción “moverse a la izquierda”

4.2.5.2 Resultados

En este experimento, los resultados se han representado mediante boxplots ya que solo se está variando un parámetro.

Tasa de Éxito

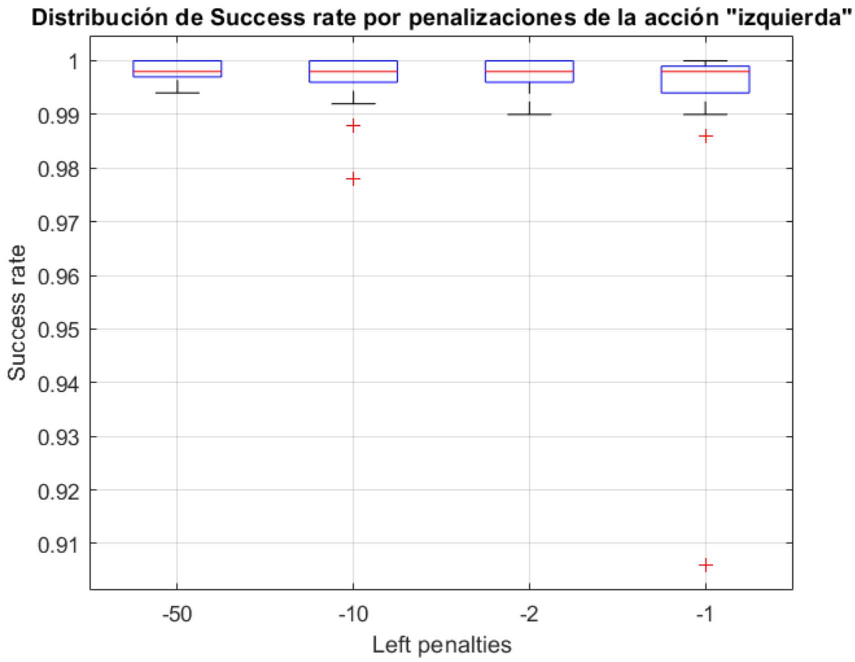
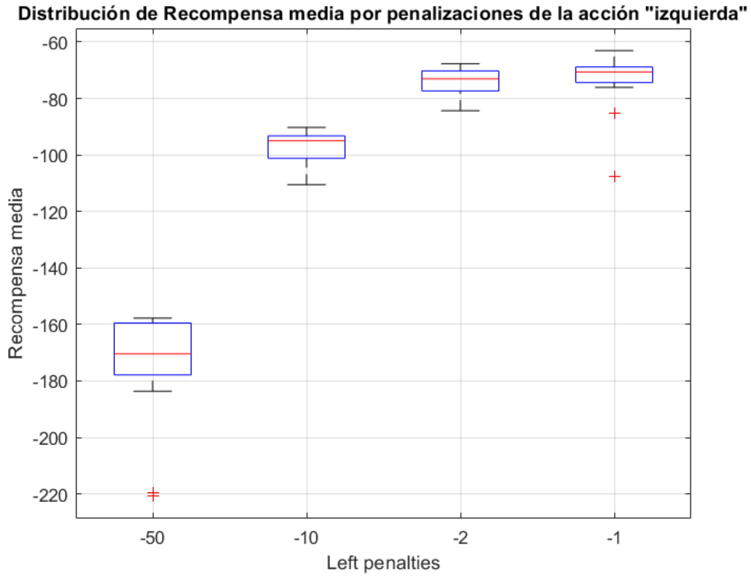


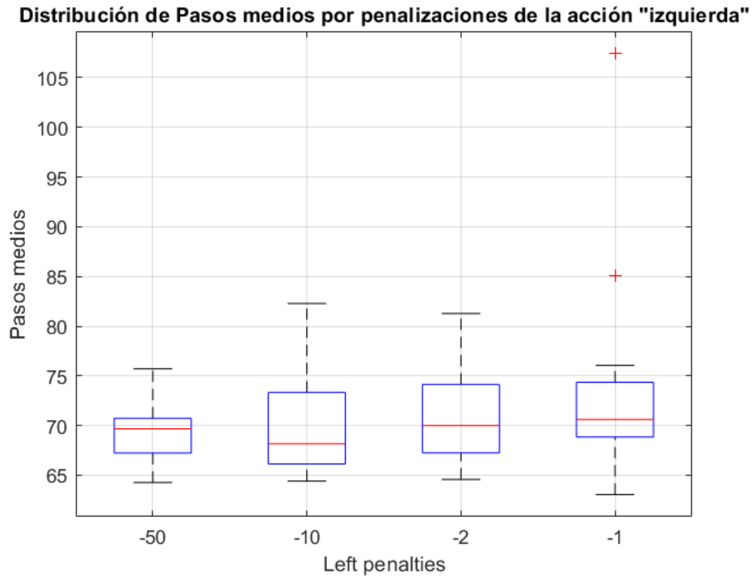
Figura 17: Tasa de éxito para diferentes valores de penalización a la acción “moverse a la izquierda”

No se aprecia una gran diferencia en cuanto a la tasa de éxito. Todas las medias se encuentran por encima del 99 % y la variabilidad es muy baja.

Recompensa media y número de pasos medios



(a) Recompensa media para diferentes valores de penalización a la acción “moverse a la izquierda”



(b) Número de pasos para diferentes valores de penalización a la acción “moverse a la izquierda”

Figura 18: Análisis de la recompensa media y número de pasos

Para seguir con la estructura de los experimentos anteriores, se muestran los boxplots de la recompensa media y el número de pasos medios. En este caso, del boxplot de la recompensa media no se puede deducir nada, ya que al haber modificado la penalización, cada recompensa media es relativa a la penalización. Sin embargo, se puede medir el rendimiento mediante el número de pasos medios. En este caso, se observa una muy ligera tendencia a mejorar el rendimiento al aumentar la penalización.

Tiempo de entrenamiento

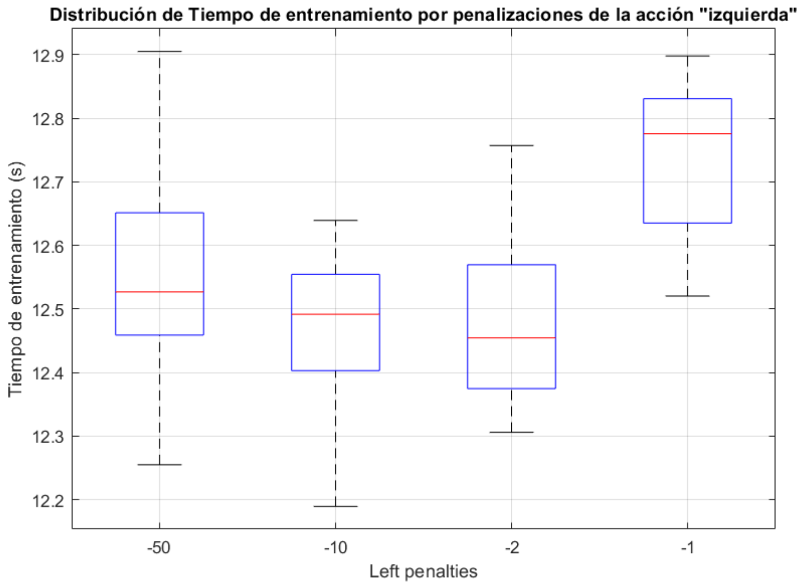


Figura 19: Tiempo de entrenamiento para diferentes valores de penalización a la acción “moverse a la izquierda”

El tiempo de se un poco reducido con penalizaciones más altas que 1, pero no se observa una tendencia clara.

Tabla resumen

Left penalty	Métrica	Media	Intervalo de Confianza del 95 %
-50	Success-rate	0.998	[0.998, 0.999]
	Rew. media	-173.343	[-179.002, -167.685]
	Pasos medios	69.4	[68.4, 70.3]
	Tiempo (s)	12.56	[12.51, 12.61]
-10	Success-rate	0.997	[0.995, 0.998]
	Rew. media	-97.021	[-98.710, -95.332]
	Pasos medios	69.8	[68.2, 71.3]
	Tiempo (s)	12.46	[12.43, 12.50]
-2	Success-rate	0.997	[0.996, 0.998]
	Rew. media	-74.077	[-75.488, -72.666]
	Pasos medios	71.0	[69.6, 72.5]
	Tiempo (s)	12.48	[12.44, 12.53]
-1	Success-rate	0.992	[0.986, 0.998]
	Rew. media	-73.278	[-76.204, -70.352]
	Pasos medios	73.3	[70.4, 76.2]
	Tiempo (s)	12.74	[12.71, 12.78]

Cuadro 16: Resultados del algoritmo para diferentes valores de penalización por ir a la izquierda

Conclusiones

Del análisis experimental se pueden extraer las siguientes conclusiones:

1. La penalización de la acción “moverse a la izquierda” no parece tener un impacto significativo en el rendimiento del algoritmo.
2. El mejor valor de penalización obtenido es **-50**, que proporciona:
   - Una tasa de éxito del 99.8 %.
   - Un número medio de pasos de 69.4.
   - Un tiempo de entrenamiento de 12.56 segundos.
3. Los resultados obtenidos difieren ligeramente de la hipótesis inicial, ya que no se observa una mejora muy significativa al penalizar la acción “moverse a la izquierda”.

5. Reinforce

5.1. Descripción del algoritmo

5.2. Experimentación

Observación	
Planteamiento	
Hipótesis	
Método	▪

Cuadro 17: Experimento 1

6. Conclusiones

7. Bibliografia

8. Apéndices