Fundamentos Matemáticos del Aprendizaje Reforzado: Q-learning

Miguel Ángel Luquín Guerrero

UNIVERSIDAD DE ZARAGOZA
GRADO DE MATEMÁTICAS

Junio, 2025

Índice

Introducción

Aprendizaje Reforzado

El **Aprendizaje Reforzado (Reinforcement Learning)** es una rama del Machine Learning donde un agente aprende a tomar decisiones en una tarea sin recibir instrucciones explícitas.

¿Cómo aprende el agente?

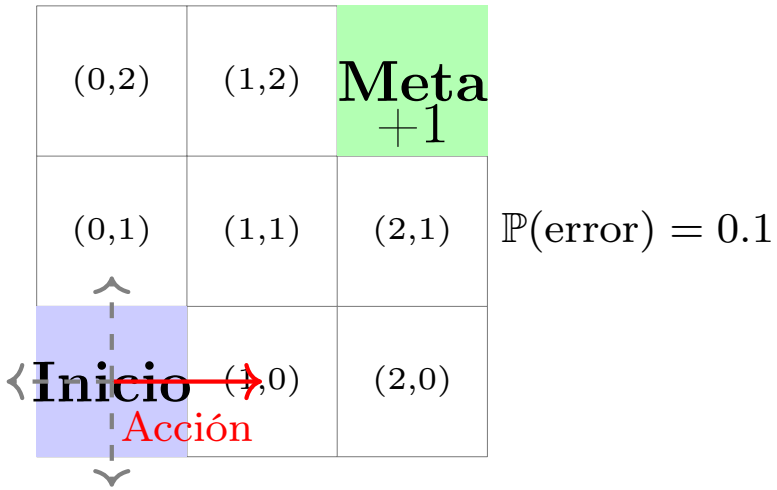
Interactuando con un entorno, recibe **recompensas** que guían su comportamiento hacia una estrategia óptima.

Ejemplo básico de un entorno en RL

Escenario

Un pingüino quiere llegar desde la **casilla de inicio** hasta la **meta**, moviéndose por una cuadrícula.

- El **agente** es el pingüino.
- Cada celda representa un **estado**.
- Puede realizar **acciones** para moverse (ej. derecha, arriba...).
- A veces se equivoca con probabilidad $\mathbb{P}(\text{error}) = 0.1$.
- Recibe una **recompensa** de $+1$ al llegar a la meta.



Marco Formal y Ecuaciones de Bellman

Procesos de decisión de Markov

Definición 1.1

Un Proceso de Decisión de Markov (MDP, por sus siglas en inglés) es una tupla (S, A, T, R) donde S y A son conjuntos finitos cuyos elementos son los estados y acciones del proceso respectivamente. Además,

- $T : S \times A \times S \to [0, 1]$ con $T(s, a, s') = \mathbb{P}(s' \mid s, a)$, la función de mide la probabilidad de que la acción a en el estado s conduzca al estado s', y
- $R : S \times A \times S \to \mathbb{R}$ la función de recompensa por pasar del estado s a s' mediante la acción a.

Episodios y políticas

Definición 1.2

Dado un MDP, un *episodio* es una sucesión de pasos temporales:

$$(s_0, a_0, s_1, r_1, a_1, s_2, r_2, \ldots, s_{F-1}, r_{F-1}, a_{F-1}, s_F, r_F).$$

Definición 1.3

Dado un MDP (S, A, T, R), una *política determinista* es una función $\pi : S \to A$ que a cada estado $s \in S$ le asigna una acción $a \in A$. Denotamos Π al conjunto de todas las políticas posibles. La función $\pi^* \in \Pi$ que maximice el valor esperado de la recompensa acumulada futura se llama *política óptima*.

Criterio de optimalidad

Definición 1.4

Dado un MDP (S, A, T, R) y un factor de descuento $\gamma \in (0, 1)$, el criterio de optimalidad usado para modelos con horizonte infinito es

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right].$$

Teorema 1.5

El criterio de optimalidad converge $\forall \gamma \in (0, 1)$ si las recompensas del MDP asociado están acotadas.

Demostración

$$\sum_{t=0}^{\infty} \gamma^t r_t \leq \sum_{t=0}^{\infty} \gamma^t |r_t| \leq \sum_{t=0}^{\infty} \gamma^t M = M \sum_{t=0}^{\infty} \gamma^t = M \frac{1}{1-\gamma} < \infty.$$

Definición 1.6

Dado un MDP (S, A, T, R) y una política cualquiera $\pi \in \Pi$, se define el valor de un estado con el criterio de optimalidad anterior como la función $V^\pi : S \to \mathbb{R}$ tal que:

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_k \middle| s_0 = s \right],$$

donde:

$$\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| a_t = \pi(s_t) \right].$$

Además definimos la función $Q^\pi : S \times A \to \mathbb{R}$ como

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_k \middle| s_0 = s, a_0 = a \right].$$

Ecuación de Bellman

Idea central

El valor de un estado bajo una política π se puede expresar de forma recursiva: es la recompensa esperada a corto plazo más el valor descontado del siguiente estado.

$$
\begin{aligned}
V^{\pi}(s) &= \mathbb{E}_{\pi}\left[r_0 + \gamma r_1 + \gamma^2 r_2 + \dots \,\middle|\, s_0 = s \right] \\
&= \mathbb{E}_{\pi}\left[r_0 + \gamma V^{\pi}(s_1) \,\middle|\, s_0 = s \right] \\
&= \sum_{s' \in S} T(s, \pi(s), s') \Big(R(s, \pi(s), s') + \gamma V^{\pi}(s') \Big).
\end{aligned}
$$

Esto es la *Ecuación de Bellman* para la función de valor estado.

Existencia de política óptima

Teorema 1.7

Dado un MDP (S, A, T, R) con S y A finitos, recompensas acotadas y un factor de descuento $\gamma \in (0, 1)$, existe una política óptima $\pi^* : S \to A$ que para cualquier estado inicial $s_0 \in S$ maximiza la recompensa esperada y cuya función de valor V^* verifica la ecuación de optimalidad de Bellman:

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} T(s, a, s') \left(R(s, a, s') + \gamma V^*(s') \right), \quad \forall s \in S.$$

Además, la política óptima puede definirse como:

$$\pi^*(s) = \arg\max_{a \in A} \sum_{s' \in S} T(s, a, s') \left(R(s, a, s') + \gamma V^*(s') \right).$$

Función Q: Acción óptima en cada estado

Definición de la función Q^*

La función de valor estado-acción óptima nos indica el valor esperado de realizar una acción a en un estado s, y luego seguir la política óptima:

$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left(R(s,a,s') + \gamma \max_{a'} Q^*(s',a') \right)$$

¿Cómo elegir la política óptima?

Basta con seleccionar en cada estado la acción con mayor valor Q:

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

Si conocemos Q^*, entonces conocer π^* es inmediato.

Algoritmo de Q-learning

Actualización de la función Q

Definición 2.1

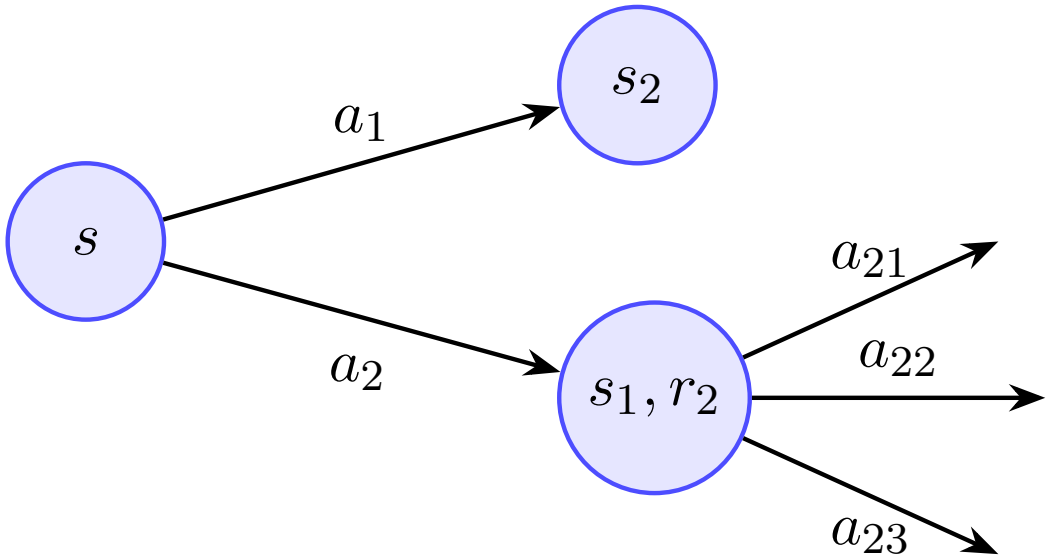
Sea (S, A, T, R) un MDP y $\gamma \in (0, 1)$ un factor de descuento. El algoritmo Q-learning actualiza iterativamente la función de valor acción-estado $Q : S \times A \to \mathbb{R}$ mediante

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

donde:

- $\alpha \in (0, 1]$ es la tasa de aprendizaje,
- $r = R(s, a, s')$ es la recompensa inmediata obtenida,
- $s' \sim T(\cdot|s, a)$ es el siguiente estado.

Actualización de la función Q



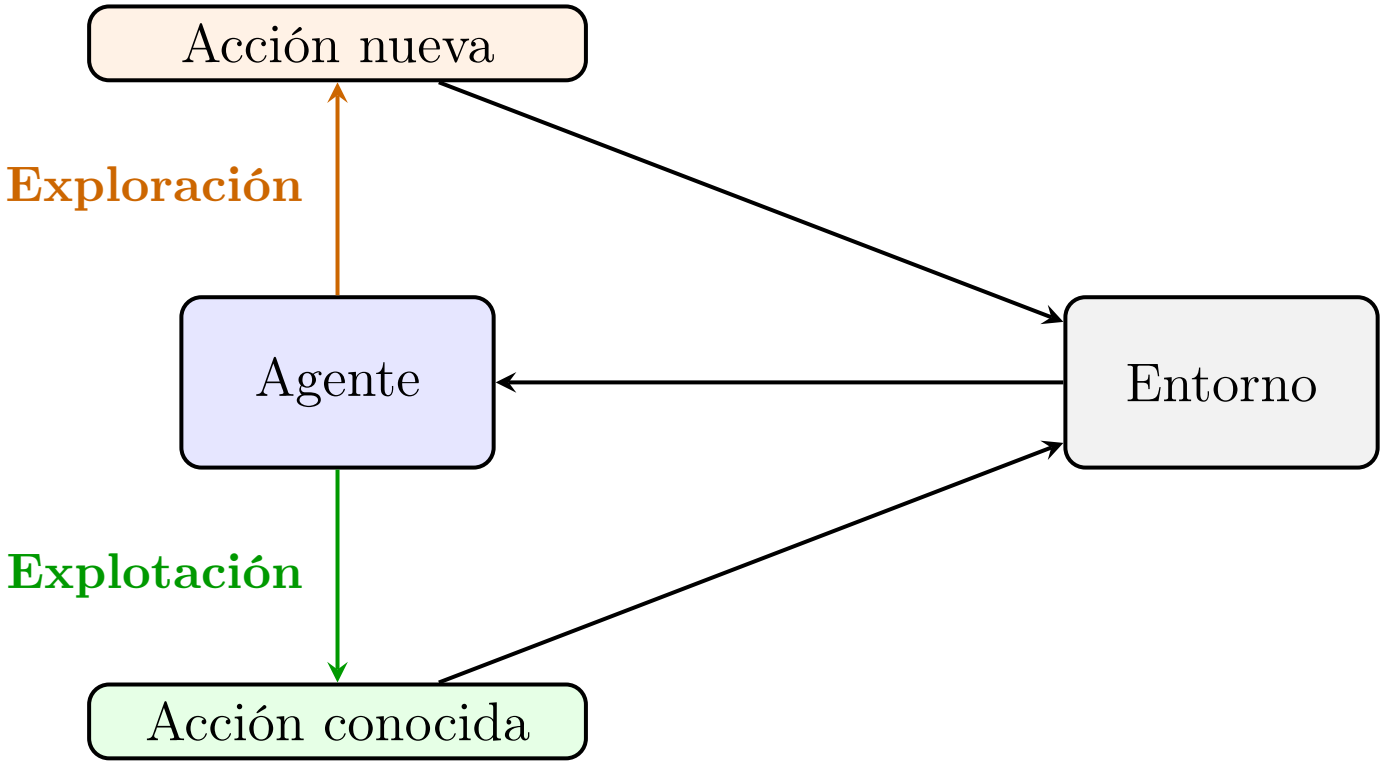
$$Q(s, a_2) \leftarrow Q(s, a_2) + \alpha \Big[r_2 + \gamma \max_{a' \in \{a_{21}, a_{23}, a_{23}\}} Q(s_1, a') - Q(s, a_2) \Big]$$

Solo se modifica el par (estado-acción) que se acaba de visitar.

Aspectos a tener en cuenta

1. **Estrategia de selección de acciones:** ¿Cómo elegir las acciones a tomar durante el entrenamiento?
2. **Inicialización de la función Q:** ¿Cómo se establece el valor inicial de las estimaciones Q?
3. **Hiperparámetros α y γ:** ¿Cómo afectan la tasa de aprendizaje y el factor de descuento al comportamiento del agente?

Exploración vs. Explotación



Exploración: probar acciones desconocidas.

Explotación: elegir la mejor opción conocida.

Estrategia ε-greedy

Objetivo: Balancear **exploración** y **explotación** durante el aprendizaje por refuerzo.

La política ε-greedy con $\varepsilon \in [0, 1]$ resuelve este dilema mediante:

$$\pi(s) = \begin{cases} \text{argmax}_a Q(s, a), & \text{con probabilidad } 1 - \varepsilon \text{ (explotación)} \\ \text{acción aleatoria}, & \text{con probabilidad } \varepsilon \text{ (exploración)} \end{cases}$$

Ventajas:

- Fácil de implementar.
- Garantiza que todas las acciones sean eventualmente exploradas.

Nota: Comúnmente, ε decrece con el tiempo, favoreciendo la explotación al final del entrenamiento.

Métodos de Inicialización de la Función Q

La forma en que se inicializa la función Q puede influir en el comportamiento del algoritmo, existen varios métodos:

Método	**Convergencia**	**Exploración inicial**	**Requiere conocimiento**
Cero	Lenta	Baja	No
Optimista	Rápida	Alta	Sí ($R_{\text{máx}}$)
Aleatoria	Moderada	Media	No

La estrategia más usada por su simplicidad en la práctica es la inicialización cero.

Hiperparámetros en Q-learning

- **Factor de descuento** $\gamma \in (0, 1)$
  Controla cuánto se valoran las recompensas futuras.
- **Tasa de aprendizaje** $\alpha \in (0, 1)$
  Determina cuánto se actualiza el valor Q con nueva información.
  Estrategias comunes:
  - Constante: $\alpha_n = \alpha_0$
  - Lineal: $\alpha_n = \alpha_0 \cdot (1 - \frac{n}{N})$
  - Exponencial: $\alpha_n = \alpha_0 \cdot e^{-\lambda n}$
- **Interacción γ–α y recomendaciones prácticas**
  - Es clave equilibrar γ y α para mejorar eficiencia.
  - Valores típicos recomendados:
    - $\gamma \in (0.9,\ 0.99)$
    - $\alpha \in (0.05,\ 0.25)$

Lago helado con Q-learning

Objetivo

Estudiar la **convergencia del algoritmo Q-learning** según los valores de γ y α.

Condiciones fijas

- Matriz inicial: $Q_0 = 0$.
- Estrategia ε-greedy con decaimiento: $\varepsilon_{n+1} = \varepsilon_n \cdot 0.999$, con $\varepsilon_0 = 1$.
- Estados S, acciones A, transición T y recompensa R como en la introducción.
- Número de episodios: $N_{episodios} = \{100, 200, \ldots, 1000\}$.

Agentes y configuración de hiperparámetros

Agentes evaluados

- **Agente 1:** $\gamma = 0.25$, $\alpha = 0.9$

  Valores no recomendados

- **Agente 2:** $\gamma = 0.95$, $\alpha = 0.1$

  Valores recomendados fijos

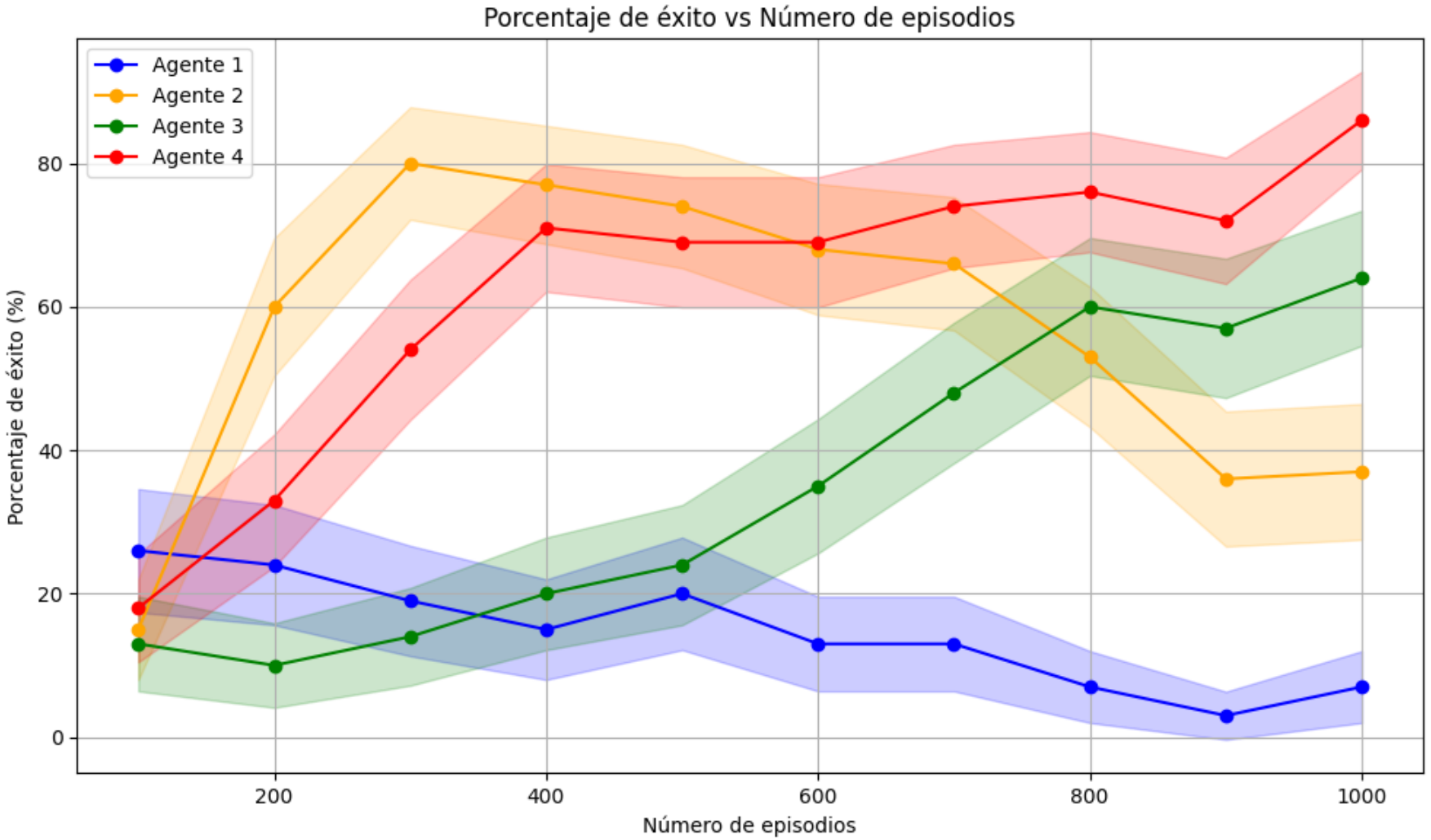
- **Agente 3:** $\gamma = 0.25$, $\alpha_n = 0.1 \cdot \left(1 - \frac{n}{N_{\text{episodios}}}\right)$

  Mal γ y decaimiento lineal de α

- **Agente 4:** $\gamma = 0.95$, $\alpha_n = 0.1 \cdot \left(1 - \frac{n}{N_{\text{episodios}}}\right)$

  Buen γ y decaimiento lineal de α

Resultados: Porcentaje de éxito de los agentes



Deep Q-learning

Fitted Q-learning: Aprendizaje Reforzado Escalable

Problema con Representación Tabular

En espacios de estado y acción de alta dimensión o continuos, la representación tabular de la función de valor-acción $Q(s, a)$ es inviable.

Solución: Fitted Q-learning

Fitted Q-learning emplea un modelo paramétrico $Q(s, a; \theta)$, donde θ son los parámetros de un **aproximador funcional** (usualmente una red neuronal).

Objetivo: Ajustar θ para que $Q(s, a; \theta) \approx Q^*(s, a)$, es decir, aproximar la función óptima de valor-acción.

Deep Q-Network (DQN) y Aproximación Universal

Deep Q-Network (DQN)

DQN integra el **Fitted Q-learning** con redes neuronales profundas, permitiendo resolver tareas de control en entornos con espacios de estado de alta dimensión.

Fundamento teórico: Aproximación Universal

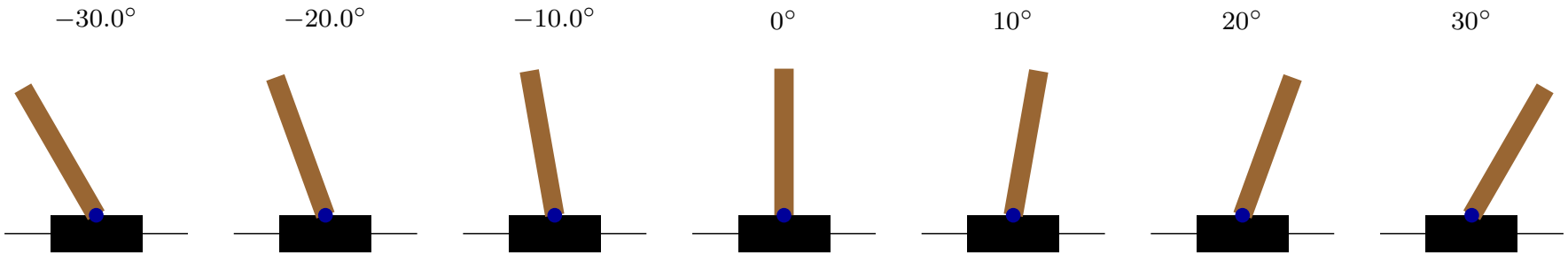
El **Teorema de Aproximación Universal** establece que una red neuronal con una sola capa oculta puede aproximar cualquier función continua con precisión arbitraria.

En el contexto de DQN, esto implica que una red neuronal $Q(s, a; \theta)$ puede aproximar la función de valor óptima $Q^*(s, a)$.

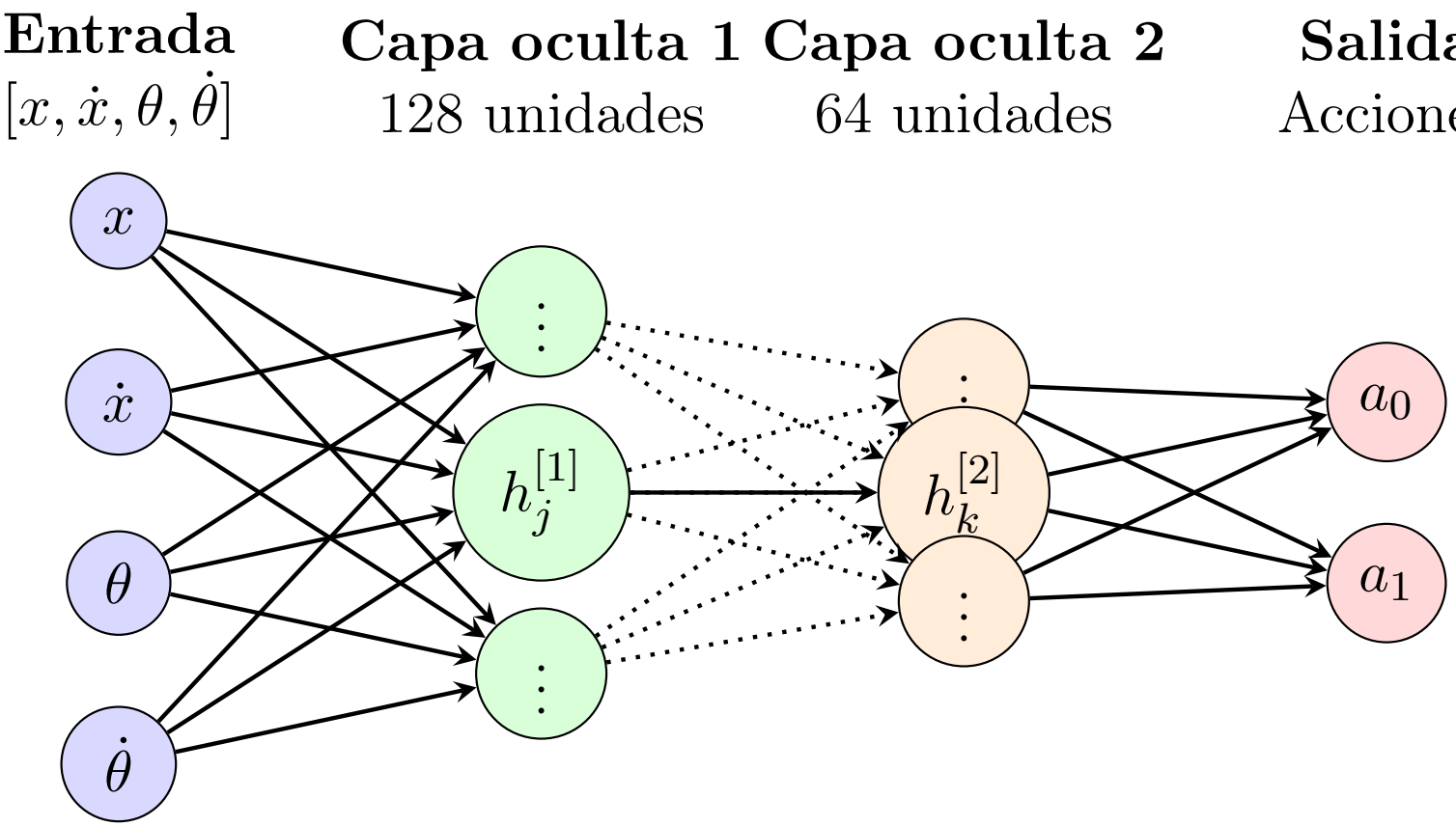
Cart Pole: Un ejemplo imposible para Q-learning tabular

Vamos a mostrar un ejemplo de entrenamiento de un agente que **no sería viable** con Q-learning tabular.

Objetivo: entrenar un agente para balancear un palo moviendo un carrito (*Cart Pole*).



Red neuronal profunda



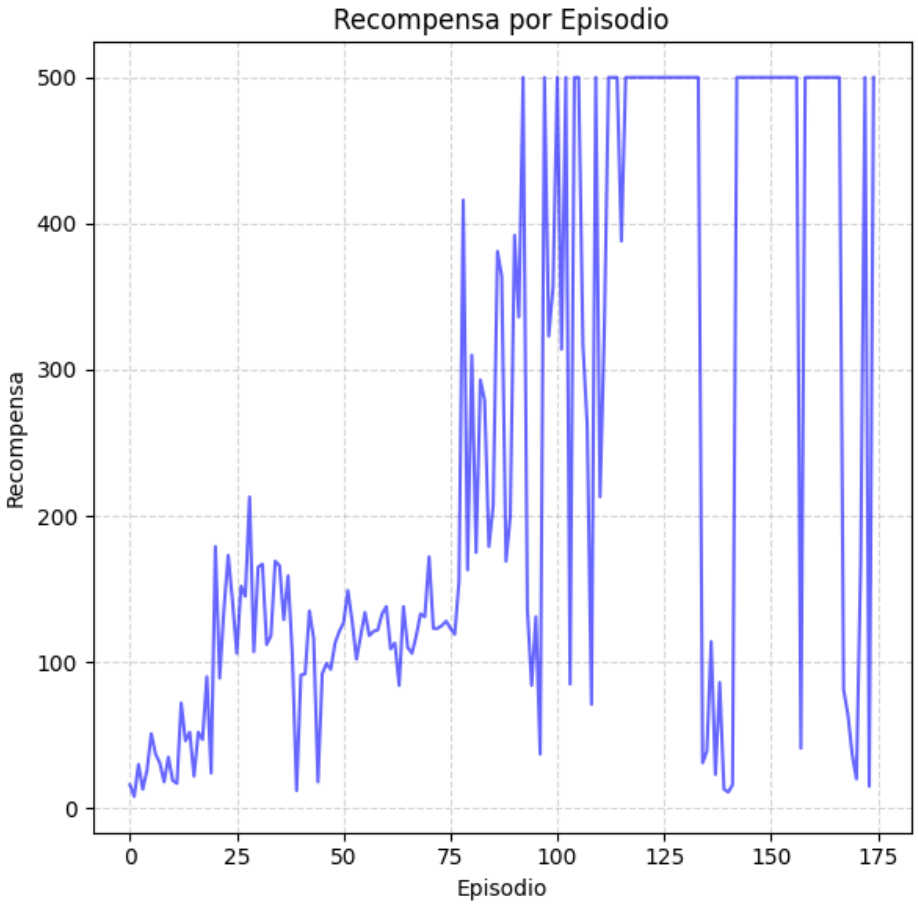
Entrenamieto y prueba

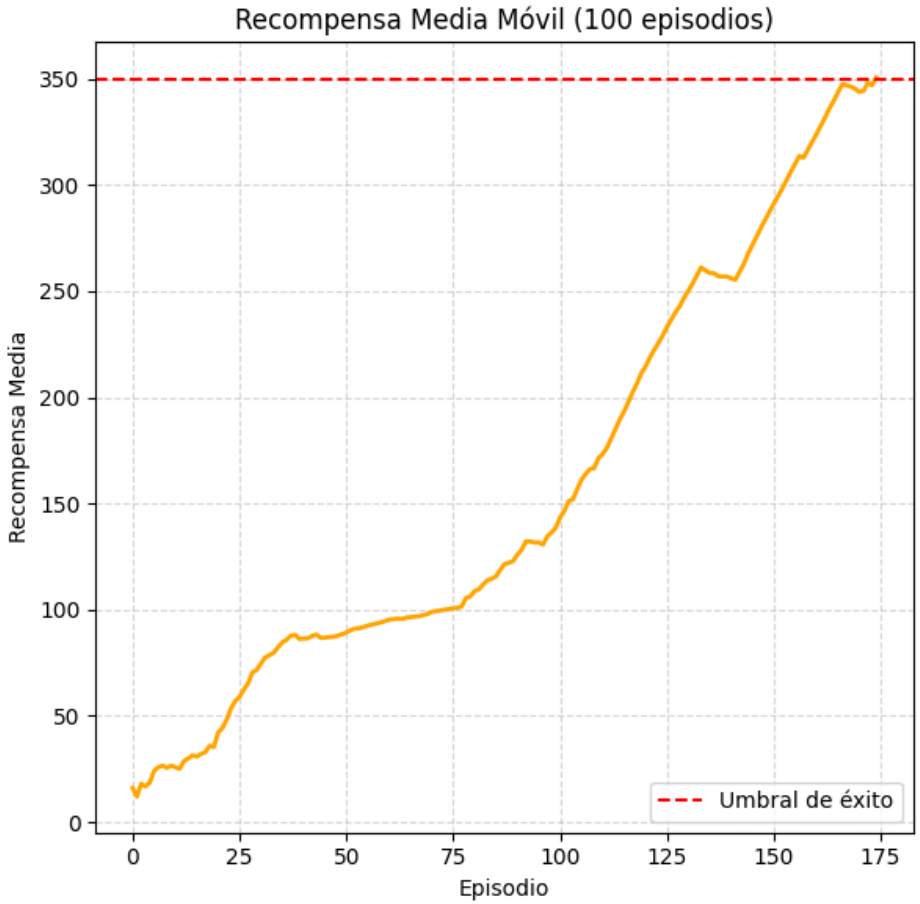
Videos del entrenamiento y prueba del algoritmo:

▸ Reproducir Video de Entrenamiento

▸ Reproducir Video de Prueba

Resultados


Recompensa por Episodio
Recompensa
Episodio
0
100
200
300
400
500
0
25
50
75
100
125
150
175


Recompensa Media Móvil (100 episodios)
Recompensa Media
Episodio
Umbral de éxito
0
50
100
150
200
250
300
350
0
25
50
75
100
125
150
175

Conclusiones

Partimos de:
Procesos de decisión de Markov y ecuaciones de Bellman.

Hemos llegado a:
Algoritmos de Q-learning tabular y aproximación de la función Q por medio de redes neuronales.

¿Qué más hay?
Se están desarrollando actualmente:
Robots móviles, estrategias de trading, gestión de tráfico en redes inalámbricas...

Agradecimientos