

Aprendizaje por refuerzo

Métodos basados en muestreo (1)

Antonio Manjavacas Lucas

manjavacas@ugr.es

Índice

1. Predicción Monte Carlo
2. Control Monte Carlo

Limitaciones de la programación dinámica

Hemos visto que los algoritmos de **programación dinámica** son **poco escalables** a medida que el tamaño del **espacio de estados/acciones** aumenta.

Además, asumíamos algo que rara vez se da en la práctica: la disponibilidad y conocimiento íntegro de un **modelo del entorno**.

- Son métodos *model-based*.

Limitaciones de la programación dinámica

Hemos visto que los algoritmos de **programación dinámica** son **poco escalables** a medida que el tamaño del **espacio de estados/acciones** aumenta.

Además, asumíamos algo que rara vez se da en la práctica: la disponibilidad y conocimiento íntegro de un **modelo del entorno**.

- Son métodos *model-based*.

?

¿Existe una forma más sencilla/escalable de obtener el valor de los diferentes estados/acciones?

Recordemos que **valor de un estado = retorno esperado**, esto es:

$$v_{\pi}(s) = \mathbb{E}_{\pi}[G_t \mid S_t = s]$$

Métodos basados en muestreo

Sample-based learning methods

Métodos basados en *sampling*

Los **métodos *sampling*** ofrecen una serie de ventajas que facilitan la estimación de valores/obtención de políticas óptimas en MDPs.

- Su principal ventaja es que **no requieren un modelo del entorno**.

En términos generales, su funcionamiento consiste en:

1. Acumular **experiencia** mediante la **interacción** con el entorno.
2. **PREDICCIÓN**: estimar el valor de estados/acciones.
3. **CONTROL**: obtener la política óptima asociada.

Comenzaremos estudiando los métodos Monte Carlo, y posteriormente veremos otras alternativas.

Predicción Monte Carlo

Predicción Monte Carlo

Método Monte Carlo

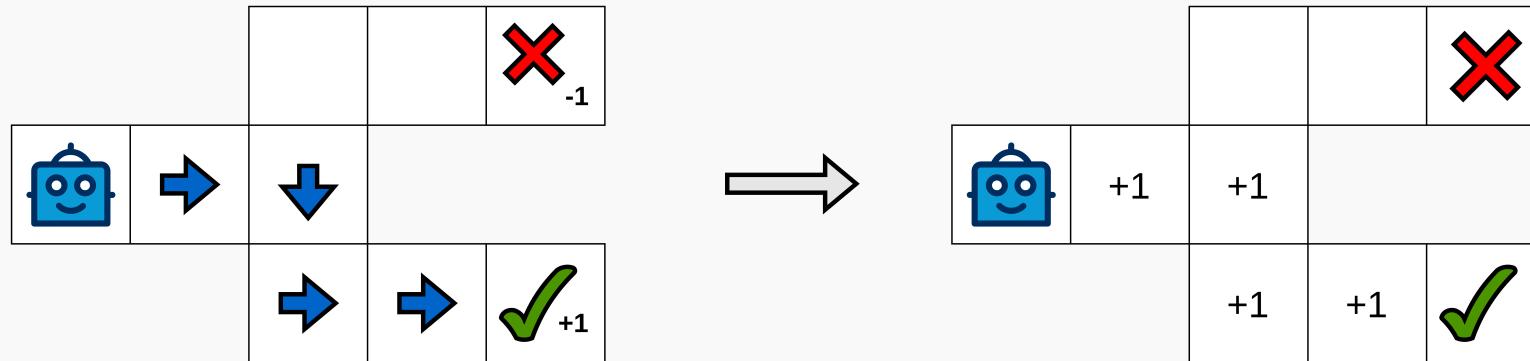
Técnica computacional empleada para estimar resultados mediante la generación de múltiples muestras aleatorias.

La idea básica detrás de la **predicción Monte Carlo** (*Monte Carlo prediction*) es seguir múltiples trayectorias aleatorias desde diferentes estados.

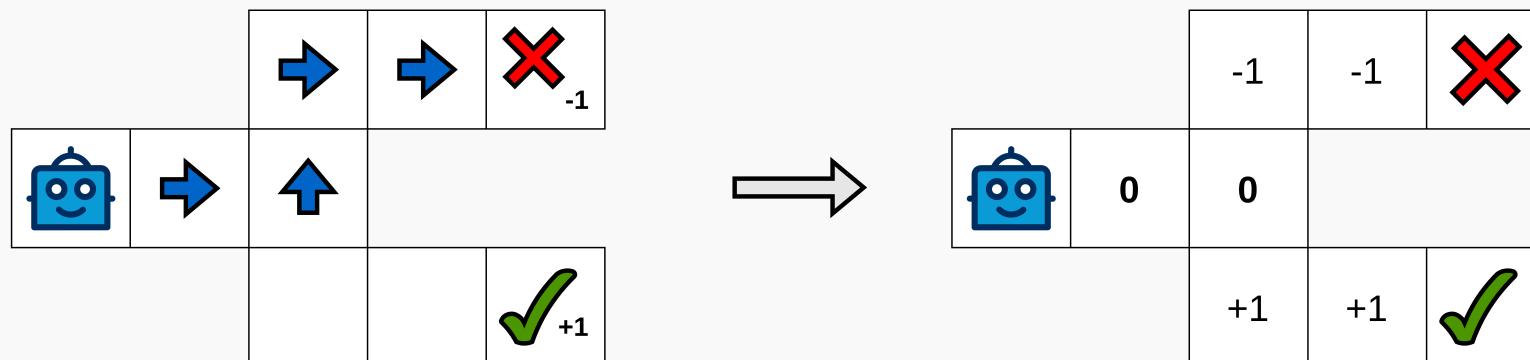
- Para aproximar el valor de un estado, calculamos la **media** de las recompensas acumuladas obtenidas cada vez que se visitó.
- No requiere un **modelo** del entorno (es **model-free**).

Ejemplo

EPISODIO 1



EPISODIO 2



Predicción Monte Carlo

Monte Carlo requiere solamente **experiencia**.

- No asume conocimiento alguno de las dinámicas del entorno... $p(s', r|s, a)$.
- No emplea valores esperados \mathbb{E} , sino **resultados empíricos**.



Permite la resolución de problemas de RL a partir del **promedio de las recompensas finales obtenidas** (*average sample returns*).

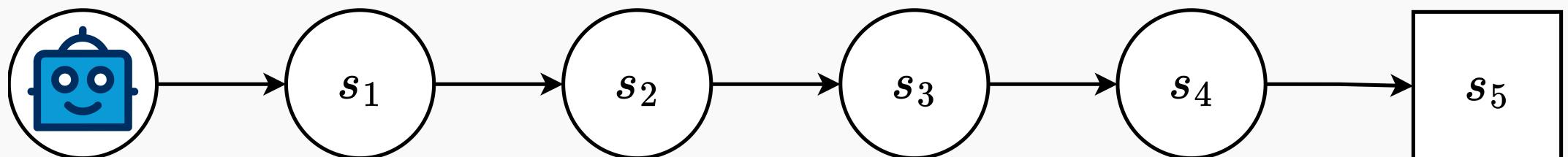
- Se trata de un **proceso de mejora episodio-a-episodio**, ya que solamente conocemos el *return* (recompensa acumulada final) al terminar un episodio.
- Decimos que es un aprendizaje **basado en resultados completos** (vs. parciales).

Ejemplo

Tratamos de estimar $v(s)$, con $\gamma = 0.5$

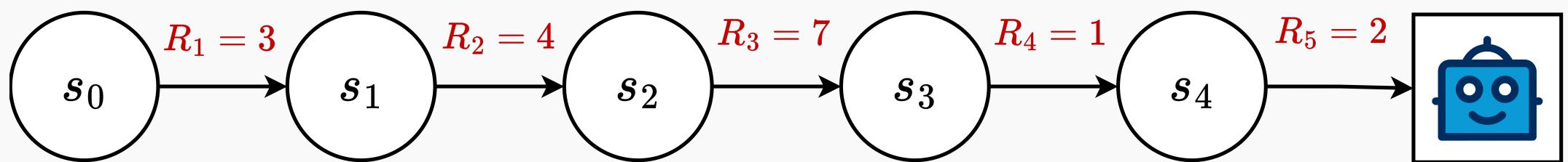
- Recordemos que $v(s) = \mathbb{E}[G_t | S_t = s]$

Comenzamos realizando una **trayectoria aleatoria**:

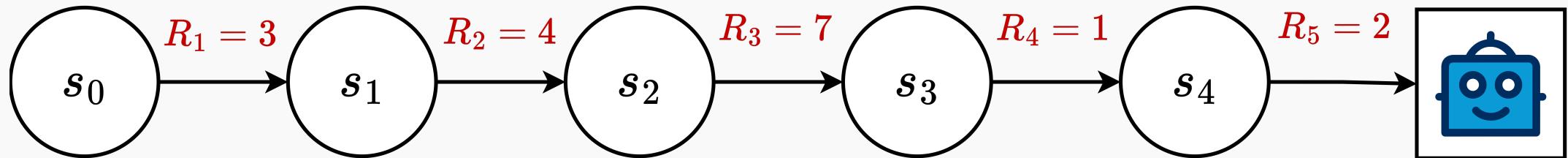


Ejemplo

Acumulamos experiencia...



Ejemplo



Calculamos el **retorno** para cada estado:

$$G_0 = R_1 + \gamma G_1$$

$$G_3 = R_4 + \gamma G_4$$

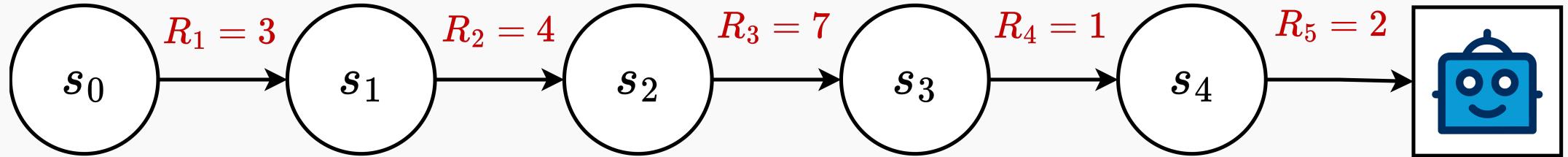
$$G_1 = R_2 + \gamma G_2$$

$$G_4 = R_5 + \gamma G_5$$

$$G_2 = R_3 + \gamma G_3$$

$$G_5 = 0$$

Ejemplo



$$G_0 = R_1 + \gamma G_1$$

$$G_3 = R_4 + \gamma G_4$$

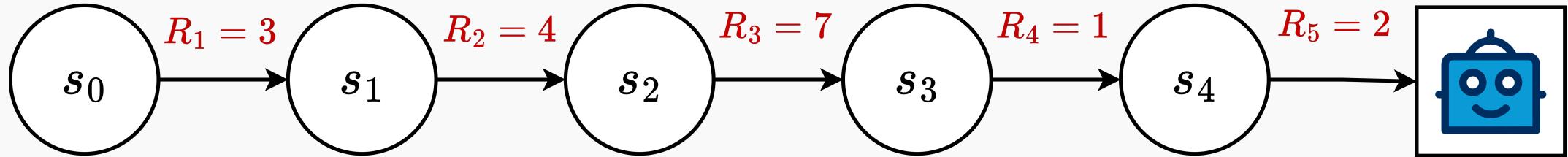
$$G_1 = R_2 + \gamma G_2$$

$$G_4 = R_5 + \gamma G_5 = 2 + 0.5 \cdot 0 = 2$$

$$G_2 = R_3 + \gamma G_3$$

$$G_5 = 0$$

Ejemplo



$$G_0 = R_1 + \gamma G_1$$

$$G_3 = R_4 + \gamma G_4 = 1 + 0.5 \cdot 2 = 2$$

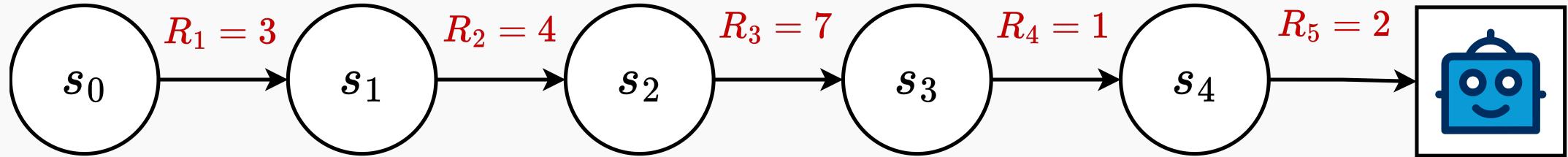
$$G_1 = R_2 + \gamma G_2$$

$$G_4 = R_5 + \gamma G_5 = 2 + 0.5 \cdot 0 = 2$$

$$G_2 = R_3 + \gamma G_3$$

$$G_5 = 0$$

Ejemplo



$$G_0 = R_1 + \gamma G_1$$

$$G_3 = R_4 + \gamma G_4 = 1 + 0.5 \cdot 2 = 2$$

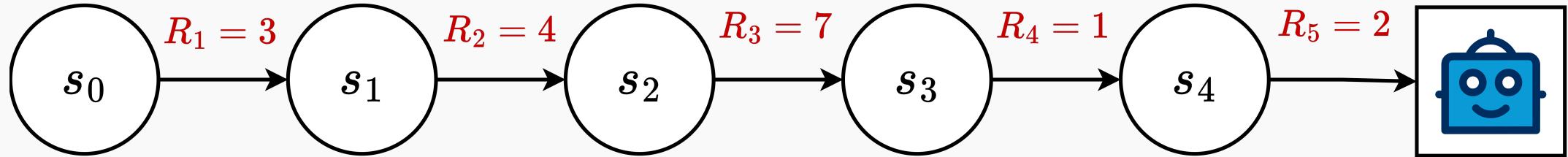
$$G_1 = R_2 + \gamma G_2$$

$$G_4 = R_5 + \gamma G_5 = 2 + 0.5 \cdot 0 = 2$$

$$G_2 = R_3 + \gamma G_3 = 7 + 0.5 \cdot 2 = 8$$

$$G_5 = 0$$

Ejemplo



$$G_0 = R_1 + \gamma G_1$$

$$G_3 = R_4 + \gamma G_4 = 1 + 0.5 \cdot 2 = 2$$

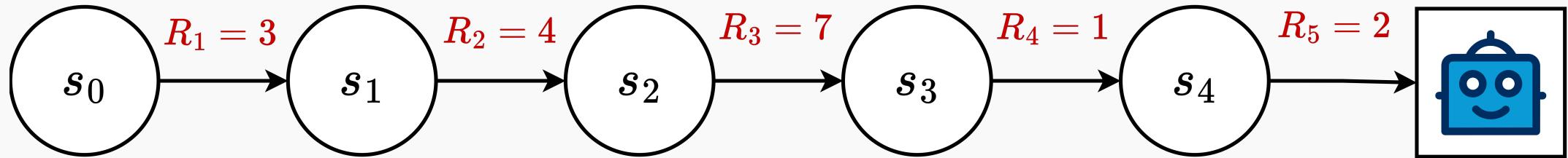
$$G_1 = R_2 + \gamma G_2 = 4 + 0.5 \cdot 8 = 8$$

$$G_4 = R_5 + \gamma G_5 = 2 + 0.5 \cdot 0 = 2$$

$$G_2 = R_3 + \gamma G_3 = 7 + 0.5 \cdot 2 = 8$$

$$G_5 = 0$$

Ejemplo



$$G_0 = R_1 + \gamma G_1 = 3 + 0.5 \cdot 8 = 7$$

$$G_1 = R_2 + \gamma G_2 = 4 + 0.5 \cdot 8 = 8$$

$$G_2 = R_3 + \gamma G_3 = 7 + 0.5 \cdot 2 = 8$$

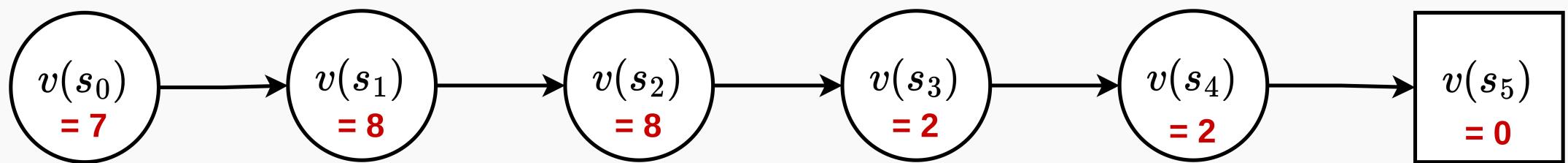
$$G_3 = R_4 + \gamma G_4 = 1 + 0.5 \cdot 2 = 2$$

$$G_4 = R_5 + \gamma G_5 = 2 + 0.5 \cdot 0 = 2$$

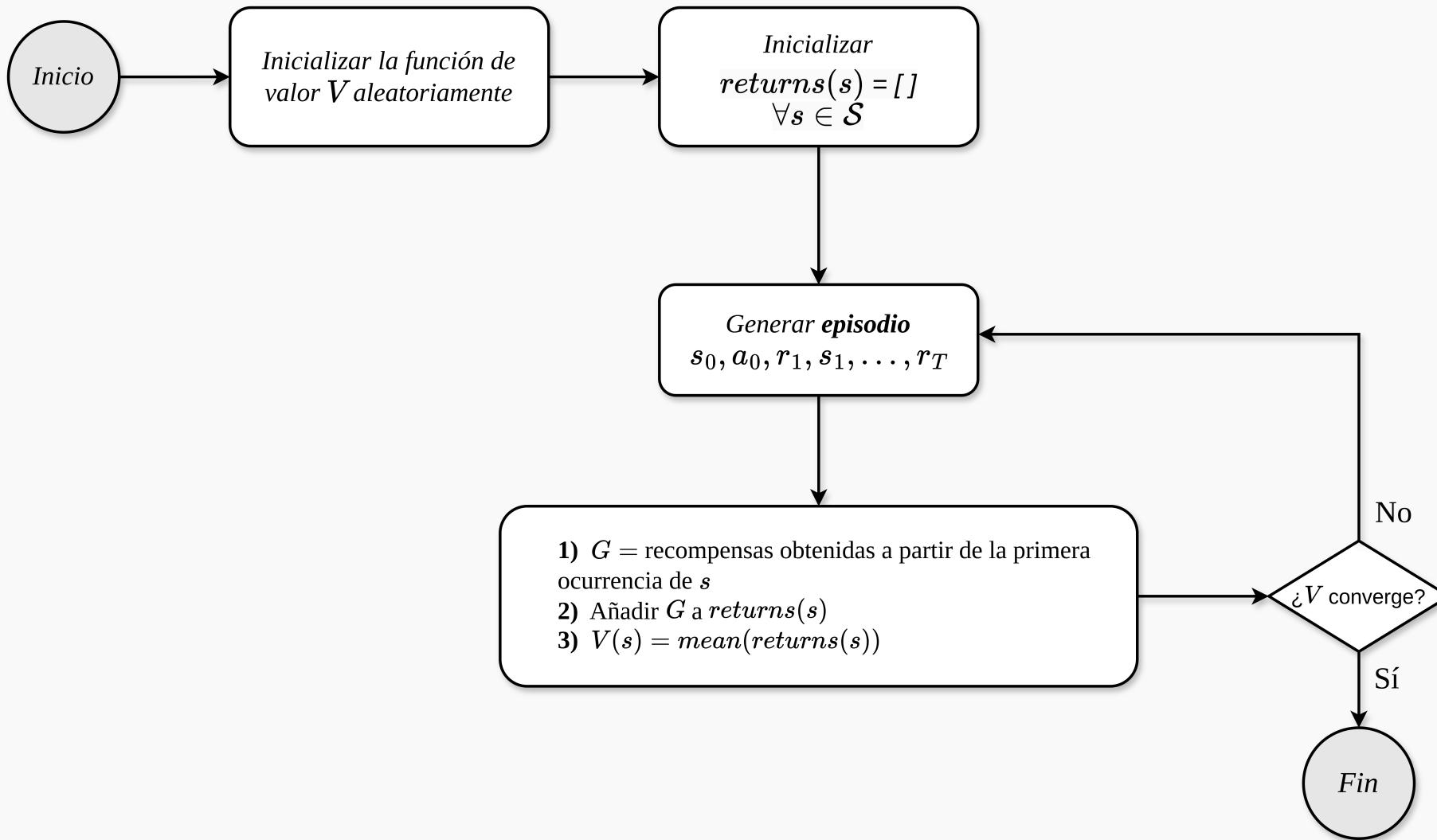
$$G_5 = 0$$

Ejemplo

Valores estimados:



Predicción Monte Carlo



Predictión *First-visit* vs. *Every-visit*

- Denominamos a este método *First-visit Monte Carlo prediction*, porque sólo tenemos en cuenta la recompensa obtenida en la **primera visita** a cada estado.
- Una alternativa es *Every-visit Monte Carlo prediction*, que tiene en cuenta **todas las visitas** a un mismo estado.

Ambos convergen en $v_{\pi}(s)$

Veamos exactamente en qué se diferencian...

Predictión First-visit vs. Every-visit

Algorithm 1: First-Visit MC Prediction

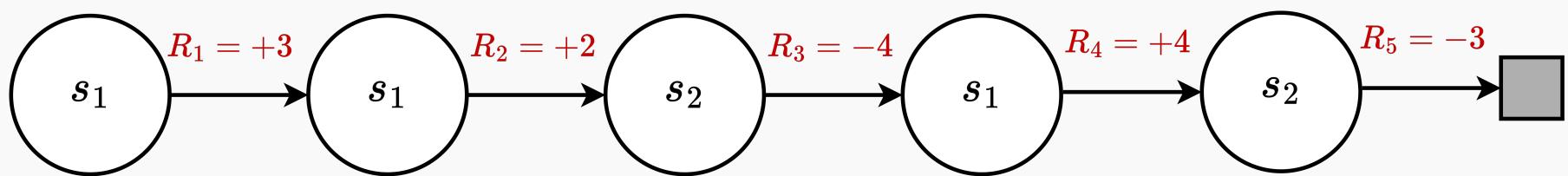
```
Input: policy  $\pi$ , positive integer  $num\_episodes$ 
Output: value function  $V$  ( $\approx v_\pi$ , if  $num\_episodes$  is large enough)
Initialize  $N(s) = 0$  for all  $s \in \mathcal{S}$ 
Initialize  $Returns(s) = 0$  for all  $s \in \mathcal{S}$ 
for episode  $e \leftarrow 1$  to  $e \leftarrow num\_episodes$  do
    Generate, using  $\pi$ , an episode  $S_0, A_0, R_1, S_1, A_1, R_2 \dots, S_{T-1}, A_{T-1}, R_T$ 
     $G \leftarrow 0$ 
    for time step  $t = T - 1$  to  $t = 0$  (of the episode  $e$ ) do
         $G \leftarrow G + R_{t+1}$ 
        if state  $S_t$  is not in the sequence  $S_0, S_1, \dots, S_{t-1}$  then
             $Returns(S_t) \leftarrow Returns(S_t) + G_t$ 
             $N(S_t) \leftarrow N(S_t) + 1$ 
        end
    end
     $V(s) \leftarrow \frac{Returns(s)}{N(s)}$  for all  $s \in \mathcal{S}$ 
return  $V$ 
```

Algorithm 2: Every-Visit MC Prediction

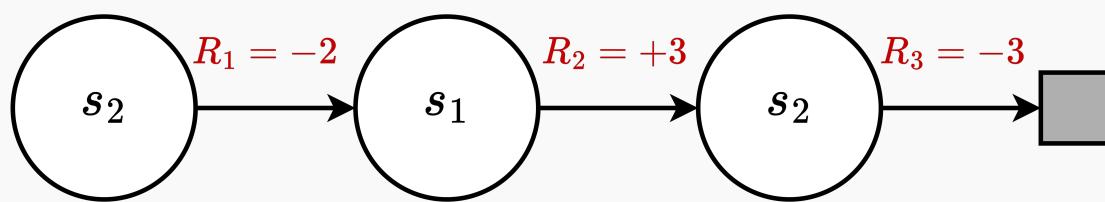
```
Input: policy  $\pi$ , positive integer  $num\_episodes$ 
Output: value function  $V$  ( $\approx v_\pi$ , if  $num\_episodes$  is large enough)
Initialize  $N(s) = 0$  for all  $s \in \mathcal{S}$ 
Initialize  $Returns(s) = 0$  for all  $s \in \mathcal{S}$ 
for episode  $e \leftarrow 1$  to  $e \leftarrow num\_episodes$  do
    Generate, using  $\pi$ , an episode  $S_0, A_0, R_1, S_1, A_1, R_2 \dots, S_{T-1}, A_{T-1}, R_T$ 
     $G \leftarrow 0$ 
    for time step  $t = T - 1$  to  $t = 0$  (of the episode  $e$ ) do
         $G \leftarrow G + R_{t+1}$ 
         $Returns(S_t) \leftarrow Returns(S_t) + G_t$ 
         $N(S_t) \leftarrow N(S_t) + 1$ 
    end
     $V(s) \leftarrow \frac{Returns(s)}{N(s)}$  for all  $s \in \mathcal{S}$ 
return  $V$ 
```

Ejemplo

EPISODIO 1



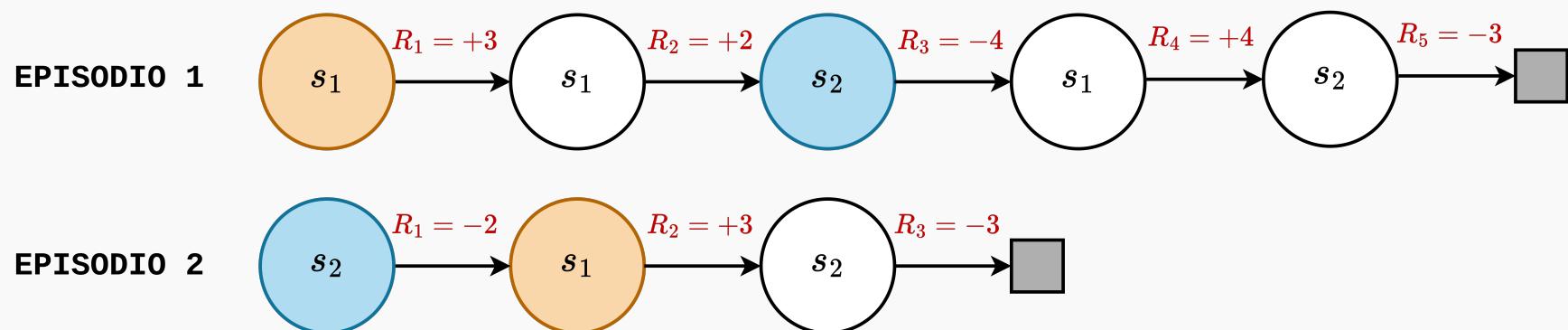
EPISODIO 2



Ejemplo

FIRST-VISIT MC

Consideramos las recompensas acumuladas a partir de la primera visita.



Episodio 1:

$$V(s_1) = 3 + 2 + -4 + 4 - 3 = \mathbf{2}$$

$$V(s_2) = -4 + 4 - 3 = \mathbf{-3}$$

Episodio 2:

$$V(s_1) = 3 - 3 = \mathbf{0}$$

$$V(s_2) = -2 + 3 - 3 = \mathbf{-2}$$

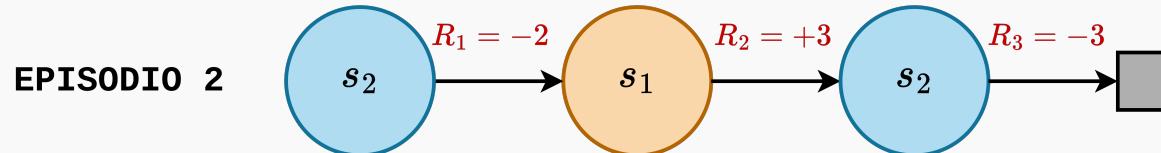
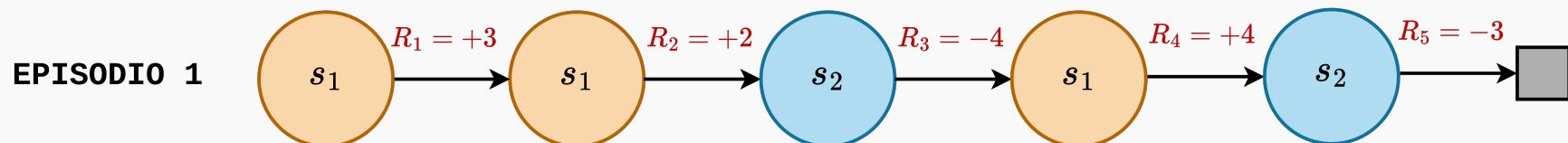
$$V(s_1) = \frac{2+0}{2} = \mathbf{1}$$

$$V(s_2) = \frac{-3-2}{2} = \mathbf{2.5}$$

Ejemplo

EVERY-VISIT MC

Consideramos las recompensas acumuladas a partir todas las visitas.



Episodio 1:

$$V(s_1) = (3 + 2 - 4 + 4 - 3) + (2 - 4 + 4 - 3) + (4 - 3) = \mathbf{2}$$

$$V(s_2) = (-4 + 4 - 3) + (-3) = \mathbf{0}$$

Episodio 2:

$$V(s_1) = 3 - 3 = \mathbf{0}$$

$$V(s_2) = (-2 + 3 - 3) + (-3) = \mathbf{1}$$

$$V(s_1) = \frac{2-1+1+0}{4} = \mathbf{0.5}$$

$$V(s_2) = \frac{-3-3-2-3}{4} = \mathbf{2.75}$$

Predicción Monte Carlo

MC es una solución relativamente simple para aproximar funciones de valor/evaluar políticas.

Un aspecto importante de MC es que la estimación del valor de un estado **NO** depende de las estimaciones de valor de otros estados.

- El valor de cada estado es independiente del resto, y sólo depende de la recompensa acumulada al final del episodio.
- Es decir, no hay *bootstrapping* (estimaciones a partir de estimaciones).

Esto puede ser útil cuando solamente queremos saber el valor de un subconjunto de estados (ignorando el resto).

Diagrama *backup* de MC

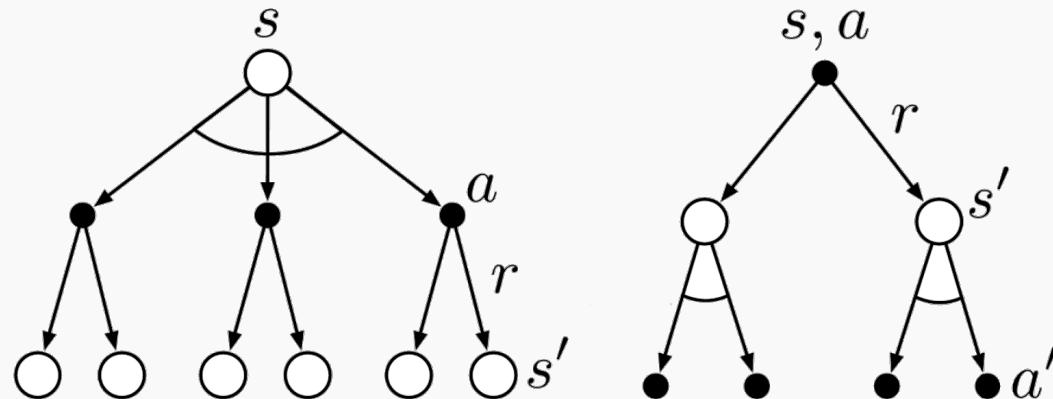
El diagrama *backup* del método Monte Carlo representa cómo las estimaciones de valor de los estados requieren llegar hasta el final del episodio.

Una vez obtenido el **retorno** (es decir, la **recompensa acumulada al final del episodio**), dicha información se propaga hacia atrás.



Diagrama backup de MC

Si comparamos con los diagramas de los algoritmos de programación dinámica...



- MC no emplea *bootstrapping*.
- MC permite estimaciones para subconjuntos de estados.

Estimación de valores de acción con MC

Si no contamos con un modelo del entorno, es particularmente útil tratar de estimar directamente los *action-values* (vs. *state-values*).

Con un modelo del entorno, los valores de los estados son suficientes para saber qué política seguir.

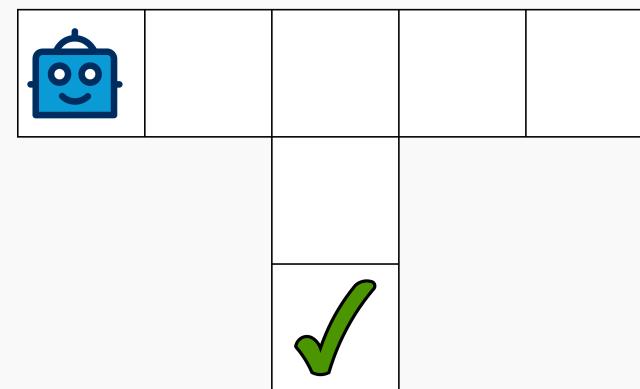
- Se mira “un paso adelante” y se decide a qué nuevo estado ir.

Sin un modelo, los valores de los estados **NO** son suficientes.

- Es necesario estimar de forma explícita el valor de cada acción.
- Saber qué acción realizar en cada estado nos conduce directamente a una **política óptima**.

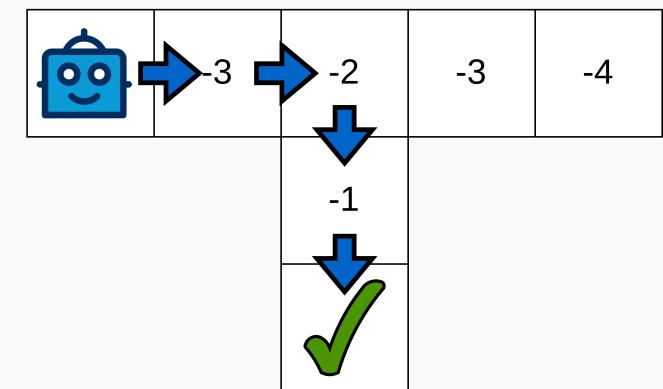
Estimación de valores de acción con MC

Con un modelo del entorno, el agente sólo tiene que estimar los valores de los estados y actuar de forma *greedy* para alcanzar su objetivo.



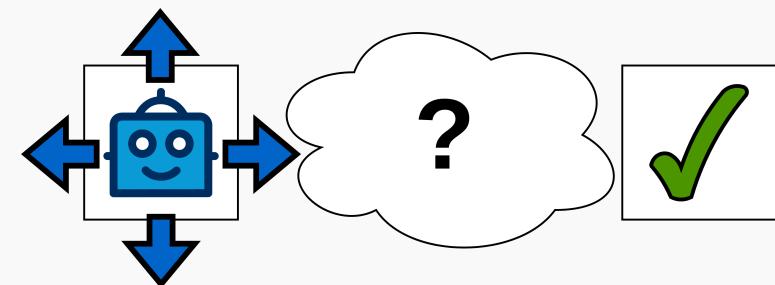
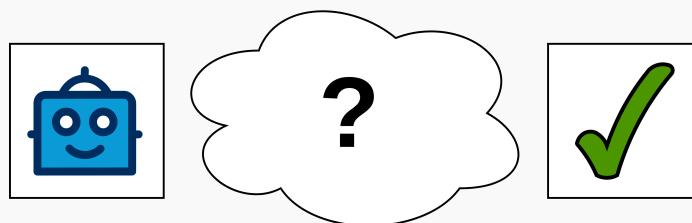
Primero estimamos los valores de los estados...

	-3	-2	-3	-4
	-1			



Estimación de valores de acción con MC

Sin un modelo del entorno, el agente necesita estimar los valores de cada par acción-estado...



El agente aprende a elegir la mejor acción para cada estado → está aprendiendo directamente la política óptima.

Control Monte Carlo

? ¿Cómo utilizar Monte Carlo para aproximar políticas óptimas?

Seguiremos la idea de **GPI** (*Iteración de la Política Generalizada*), aplicando hasta convergencia:

1. **Evaluación** de la política
2. **Mejora** de la política

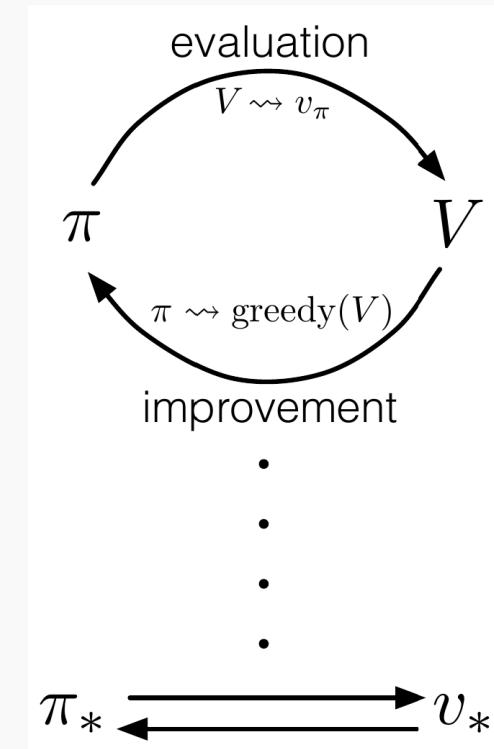
$$\pi_0 \rightarrow q_0 \rightarrow \pi_1 \rightarrow q_1 \rightarrow \dots \rightarrow \pi_* \rightarrow q_*$$

Recordemos...

En GPI se mantiene una **función de valor** y una **política** aproximadas.

 La **función de valor** se actualiza progresivamente hasta aproximarse a la función de valor de la política actual.

 Por otro lado, la **política** siempre se mejora con respecto a la función de valor actual (de forma *greedy*).



Recordemos...

$$\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \xrightarrow{I} \dots \xrightarrow{I} \pi^* \xrightarrow{E} q_{\pi^*}$$

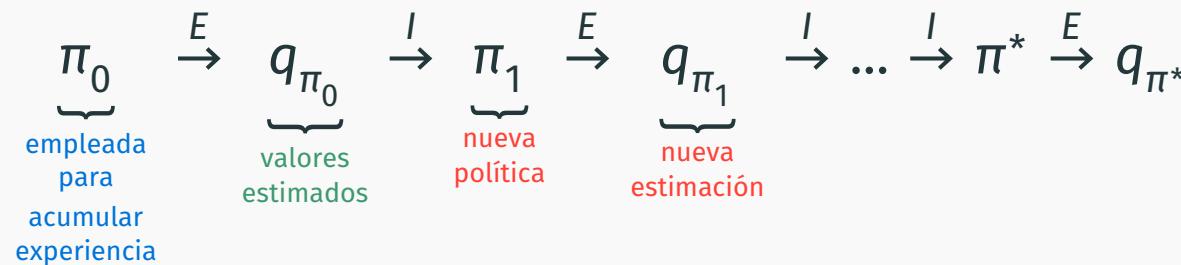
Para cada función de valor q , la política *greedy* correspondiente es aquella tal que $\forall s \in \mathcal{S}$ elige de forma **determinista** una acción con valor máximo:

$$\pi(s) = \operatorname{argmax}_a q(s, a)$$

Es decir, π_{k+1} es siempre la política *greedy* con respecto a q_{π_k} .

 Matemáticamente, se demuestra que cada π_{k+1} será uniformemente mejor que π_k o, al menos, igual (en ese caso, ambas políticas son óptimas).

GPI con Monte Carlo



De esta forma, MC es capaz de obtener **políticas óptimas** a partir de **episodios muestreados** y sin ningún conocimiento del entorno.

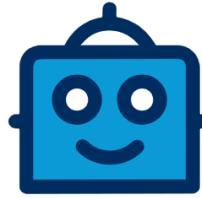
La estabilidad se alcanza cuando la política y la función de valor son óptimas.

Ejemplo

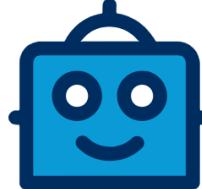
Las recompensas son los valores en cada casilla. Asumimos $\gamma = 1$.

s_1 -1	s_2 -1	s_3 -1
s_4 -1	s_5 -1	s_6 -1
s_7 -1	s_8 -10	s_9 +10

Ejemplo

	-1	-1
-1	-1	-1
-1	-10	+10

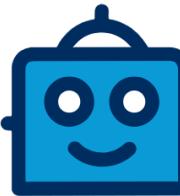
Ejemplo

-1	-1	-1
	-1	-1
-1	-10	+10

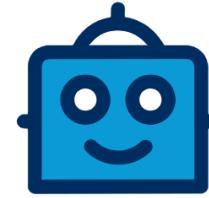
Ejemplo

-1	-1	-1
-1		-1
-1	-10	+10

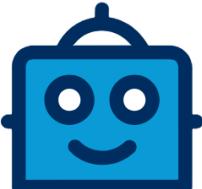
Ejemplo

-1		-1
-1	-1	-1
-1	-10	+10

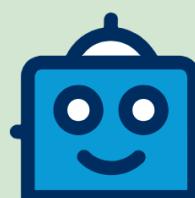
Ejemplo

-1	-1	
-1	-1	-1
-1	-10	+10

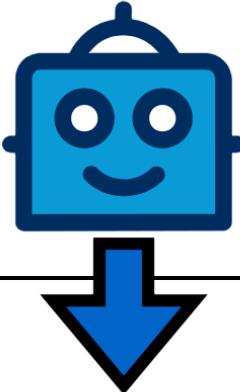
Ejemplo

-1	-1	-1
-1	-1	
-1	-10	+10

Ejemplo

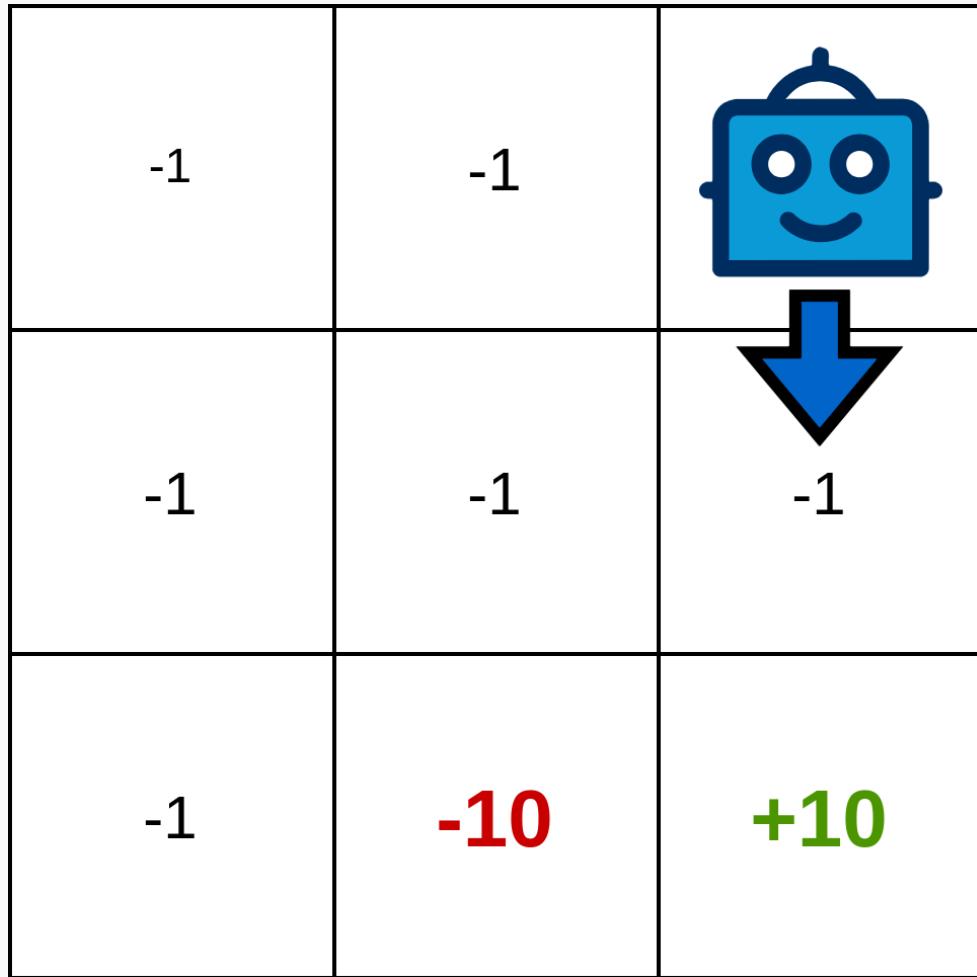
-1	-1	-1
-1	-1	-1
-1	-10	

Ejemplo

-1	-1	-1
-1	-1	
-1	-10	+10

$$q(s_6, \downarrow) = 10$$

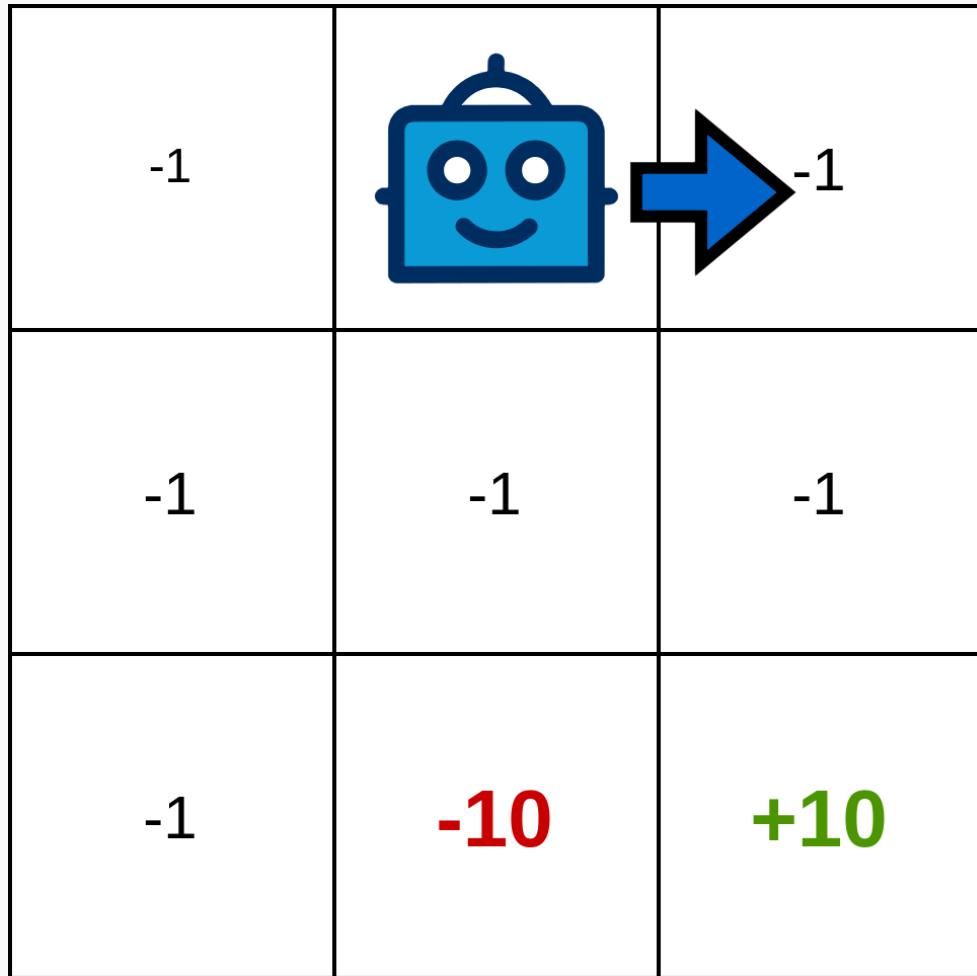
Ejemplo



$$q(s_6, \downarrow) = 10$$

$$q(s_3, \downarrow) = (-1) + 10 = 9$$

Ejemplo

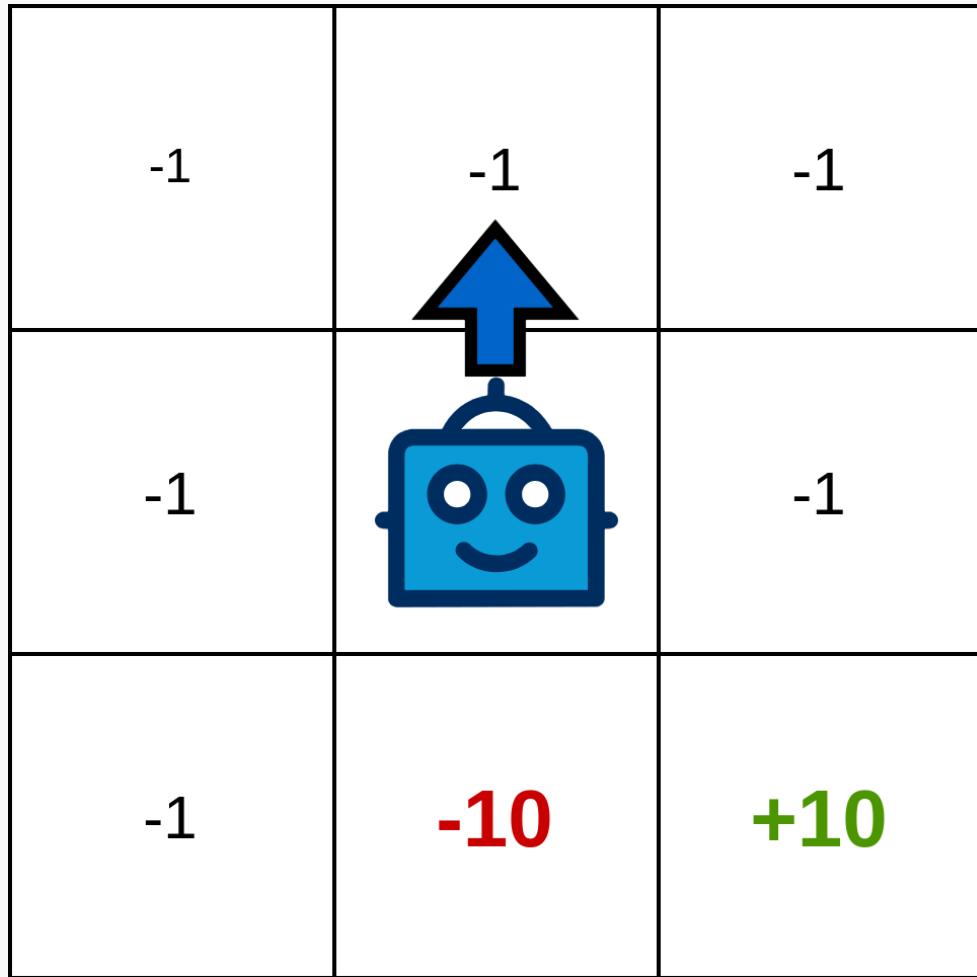


$$q(s_6, \downarrow) = 10$$

$$q(s_3, \downarrow) = 9$$

$$q(s_2, \rightarrow) = (-1) + 9 = 8$$

Ejemplo



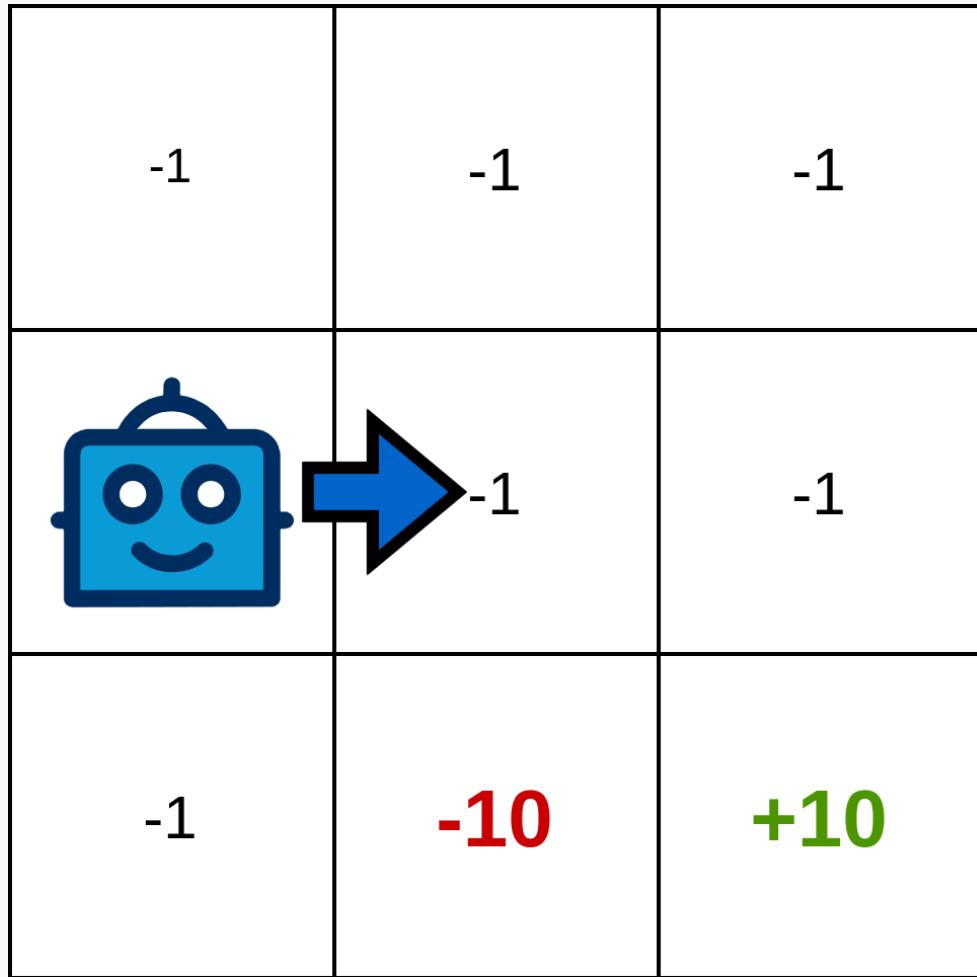
$$q(s_6, \downarrow) = 10$$

$$q(s_3, \downarrow) = 9$$

$$q(s_2, \rightarrow) = 8$$

$$q(s_5, \uparrow) = (-1) + 8 = 7$$

Ejemplo



$$q(s_6, \downarrow) = 10$$

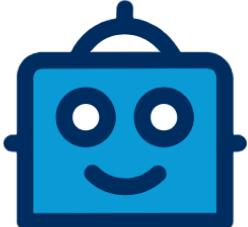
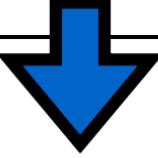
$$q(s_3, \downarrow) = 9$$

$$q(s_2, \rightarrow) = 8$$

$$q(s_5, \uparrow) = 7$$

$$q(s_4, \rightarrow) = (-1) + 7 = 6$$

Ejemplo

 	-1	-1
-1	-1	-1
-1	-10	+10

$$q(s_6, \downarrow) = 10$$

$$q(s_3, \downarrow) = 9$$

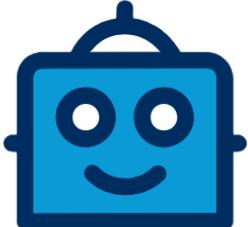
$$q(s_2, \rightarrow) = 8$$

$$q(s_5, \uparrow) = 7$$

$$q(s_4, \rightarrow) = 6$$

$$q(s_1, \downarrow) = (-1) + 6 = 5$$

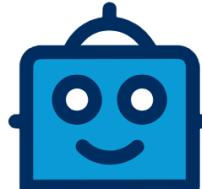
Ejemplo

	-1	-1
-1	-1	-1
-1	-10	+10

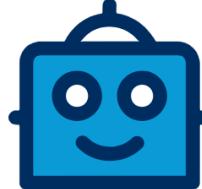
				
s_1		5		
s_2				8
s_3		9		
s_4				6
s_5	7			
s_6		10		
s_7				
...

Ejemplo

Siguiente iteración...

	-1	-1
-1	-1	-1
-1	-10	+10

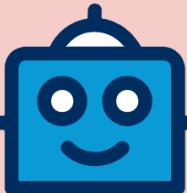
Ejemplo

-1	-1	-1
	-1	-1
-1	-10	+10

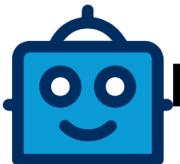
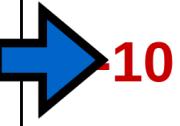
Ejemplo

-1	-1	-1
-1	-1	-1
	-10	+10

Ejemplo

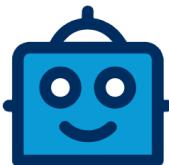
-1	-1	-1
-1	-1	-1
-1		+10

Ejemplo

-1	-1	-1
-1	-1	-1
 	10	+10

$$q(s_7, \rightarrow) = -10$$

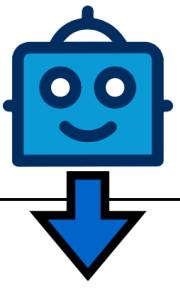
Ejemplo

-1	-1	-1
 	-1	-1
-1	-10	+10

$$q(s_7, \rightarrow) = -10$$

$$q(s_4, \downarrow) = (-1) + (-10) = -11$$

Ejemplo

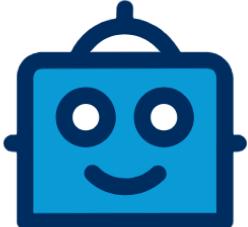
	-1	-1
-1	-1	-1
-1	-10	+10

$$q(s_7, \rightarrow) = -10$$

$$q(s_4, \downarrow) = -11$$

$$q(s_1, \leftarrow) = (-1) + (-11) = -12$$

Ejemplo

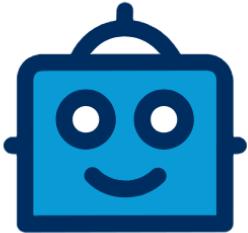
	-1	-1
-1	-1	-1
-1	-10	+10

	↑	↓	←	→
s_1		$\frac{5+(-12)}{2} = -3.5$		
s_2				8
s_3		9		
s_4		-11		6
s_5	7			
s_6		10		
s_7				-10
...

Ejemplo



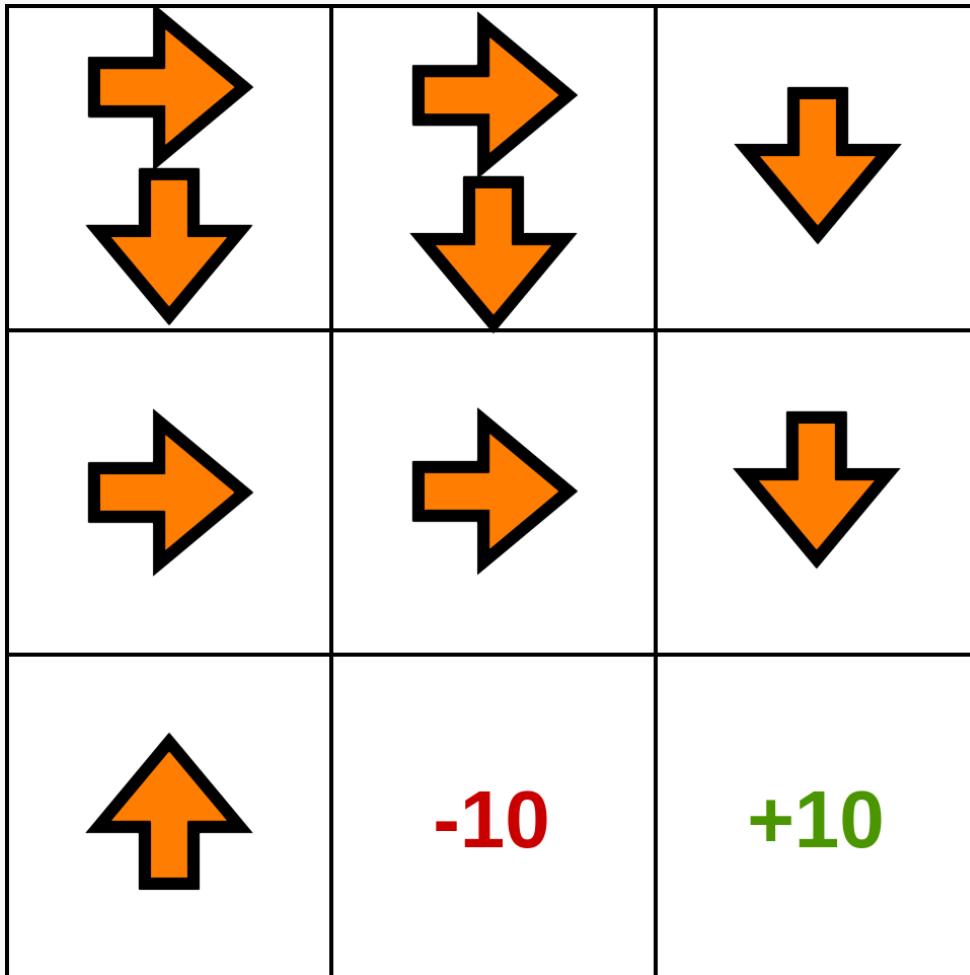
Ejemplo

	-1	-1
-1	-1	-1
-1	-10	+10

Los valores convergen...

				
s_1	6	7	6	7
s_2	7	8	6	8
s_3	8	9	7	8
s_4	6	6	7	8
s_5	7	-10	7	9
s_6	8	10	8	9
s_7	7	6	6	-10
s_8	-	-	-	-
s_9	-	-	-	-

Ejemplo

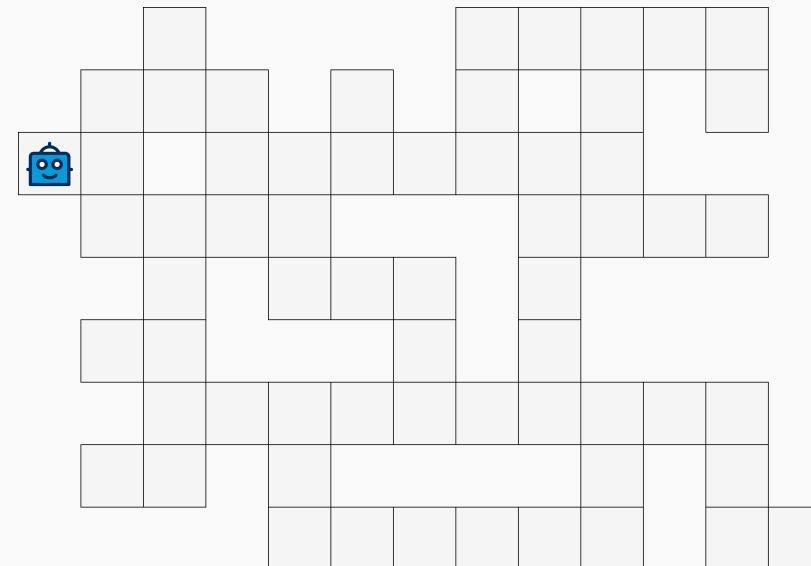


Obtenemos la **política greedy**...

	↑	↓	←	→
s_1	6	7	6	7
s_2	7	8	6	8
s_3	8	9	7	8
s_4	6	6	7	8
s_5	7	-10	7	9
s_6	8	10	8	9
s_7	7	6	6	-10
s_8	-	-	-	-
s_9	-	-	-	-

Exploración en métodos MC

Un problema de MC a la hora de estimar valores es que algunos pares acción-estado podrían no visitarse nunca...

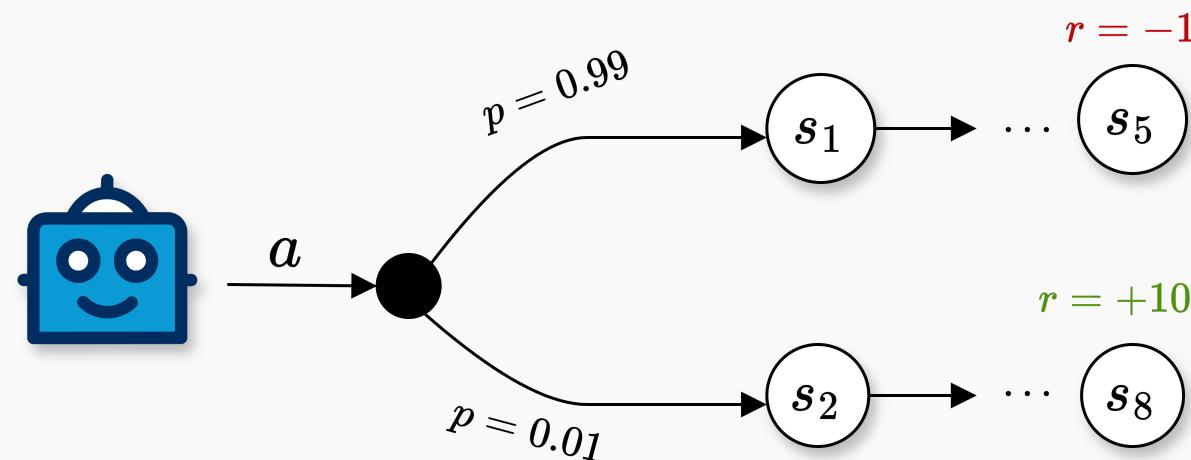


Si el agente sigue una política completamente aleatoria, es posible que ciertas acciones/estados rara vez se seleccionen, especialmente en **entornos complejos**.

Exploración en métodos MC

Este **sesgo en la selección de acciones** conduce a una **exploración desigual** y a una **estimación sesgada** de los valores.

El problema es similar si nos encontramos en un entorno **no determinista**...



Exploración en métodos MC



Es necesario favorecer la **exploración** del agente.

Debemos asegurar que todos los pares acción-estado se acaben visitando. Para ello:

- Las acciones que puedan tomarse partiendo de un estado $s \in S$ nunca tendrán probabilidad = 0 (**política estocástica**).
- Para evitar problemas asociados a entornos no deterministas (donde las transiciones a algunos estados pueden ser poco frecuentes), podemos emplear **inicios de exploración** (*exploring starts*).

Inicios de exploración

Se trata de una forma de asegurar **exploración continua**.

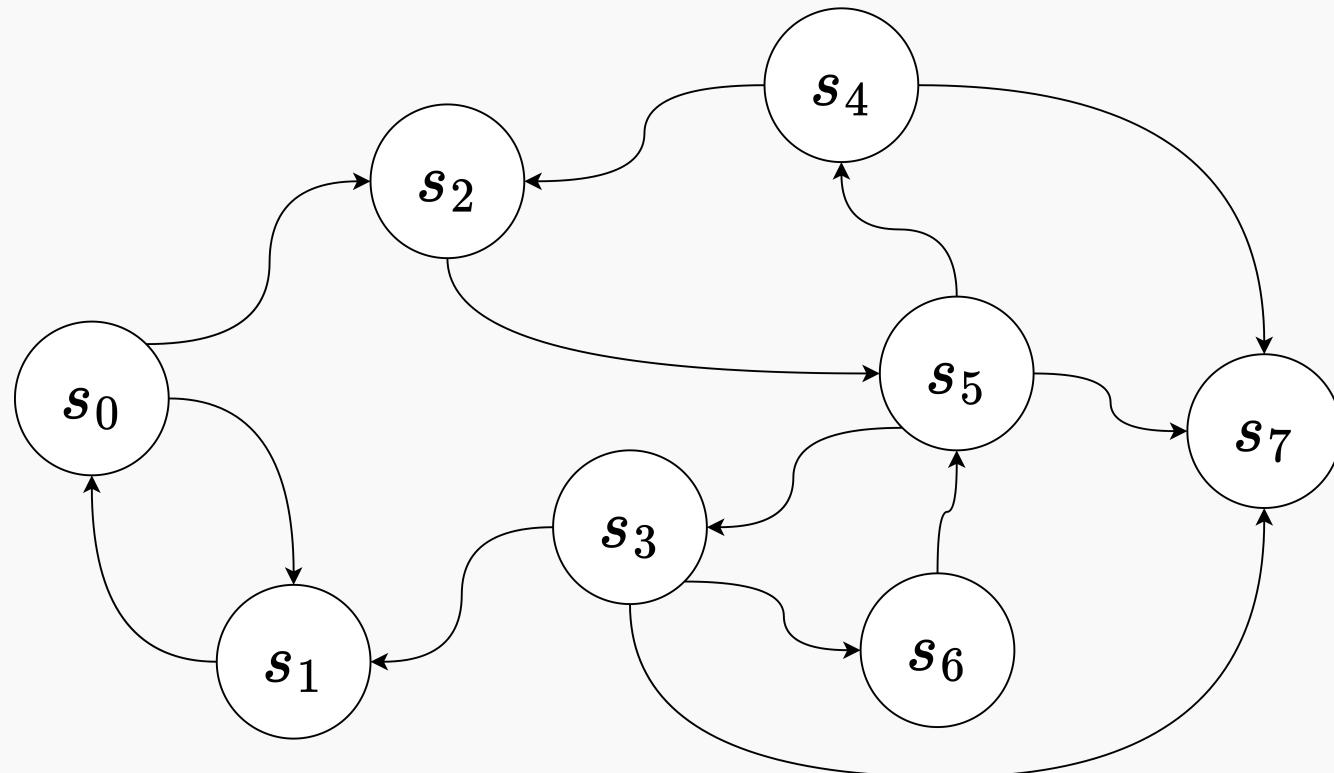
- Todo episodio empieza desde un **par estado-acción aleatorio**:

$$\underbrace{s_0, a_0}_{\text{Aleatorios}}, \underbrace{s_1, a_1, s_2, a_2, \dots}_{\text{Dependientes de } \pi, p}$$

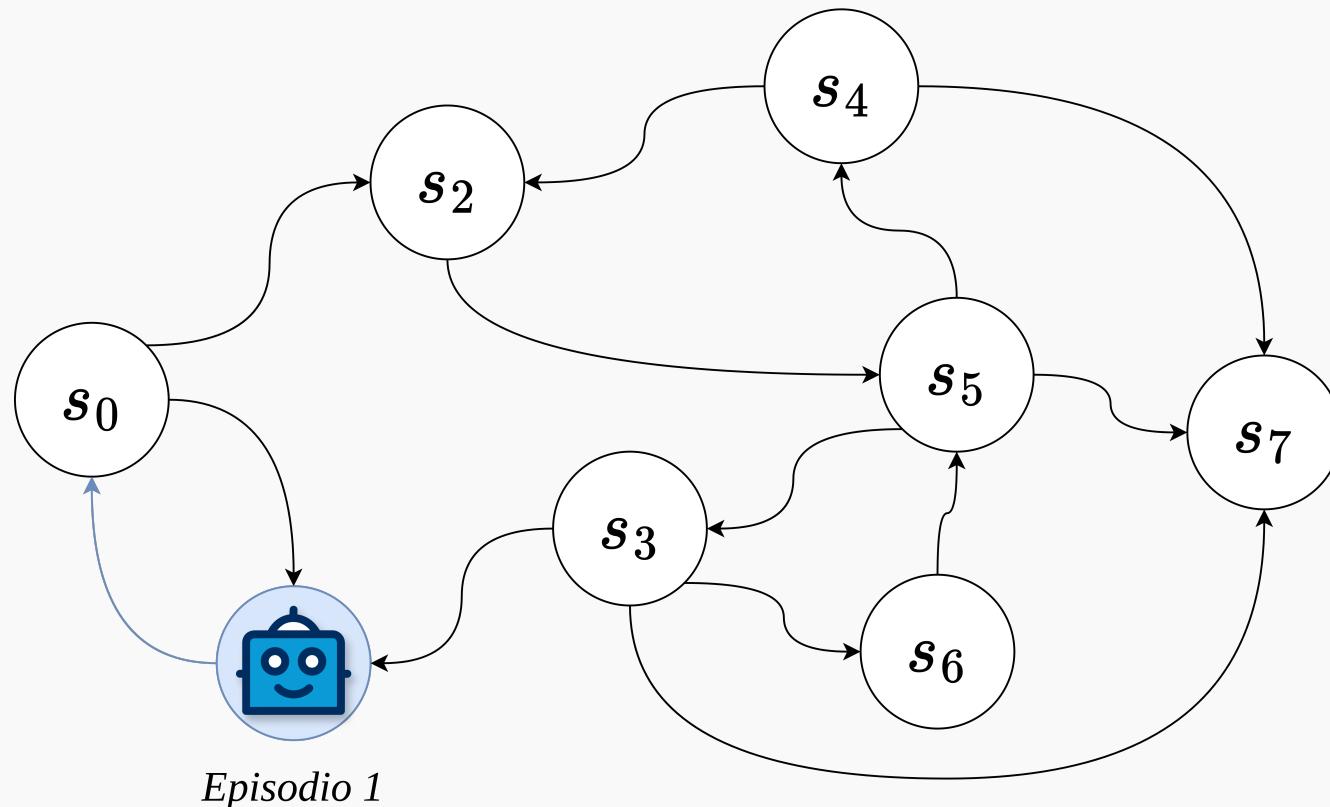
- Cada par estado-acción tiene una probabilidad **no nula** de ser seleccionado **al principio de un episodio**:

$$\mu(s, a) > 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$$

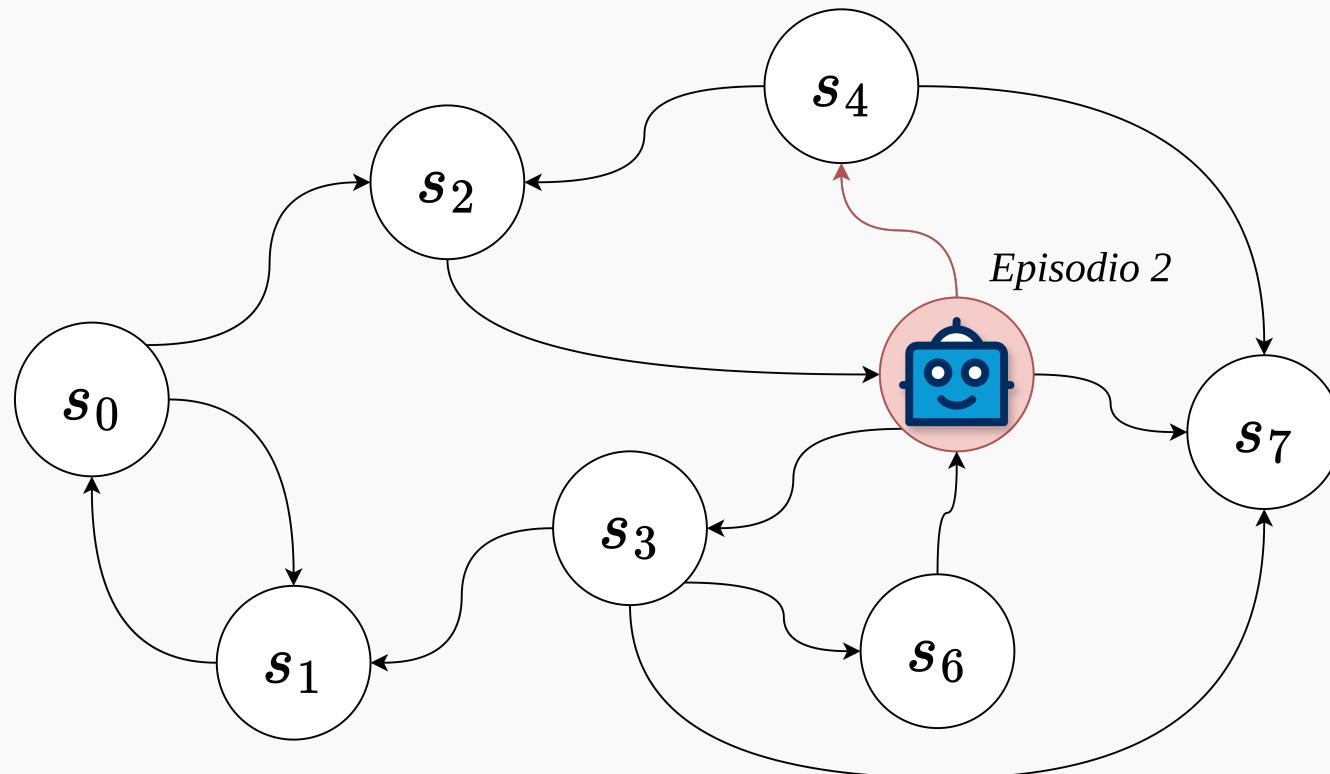
Ejemplo



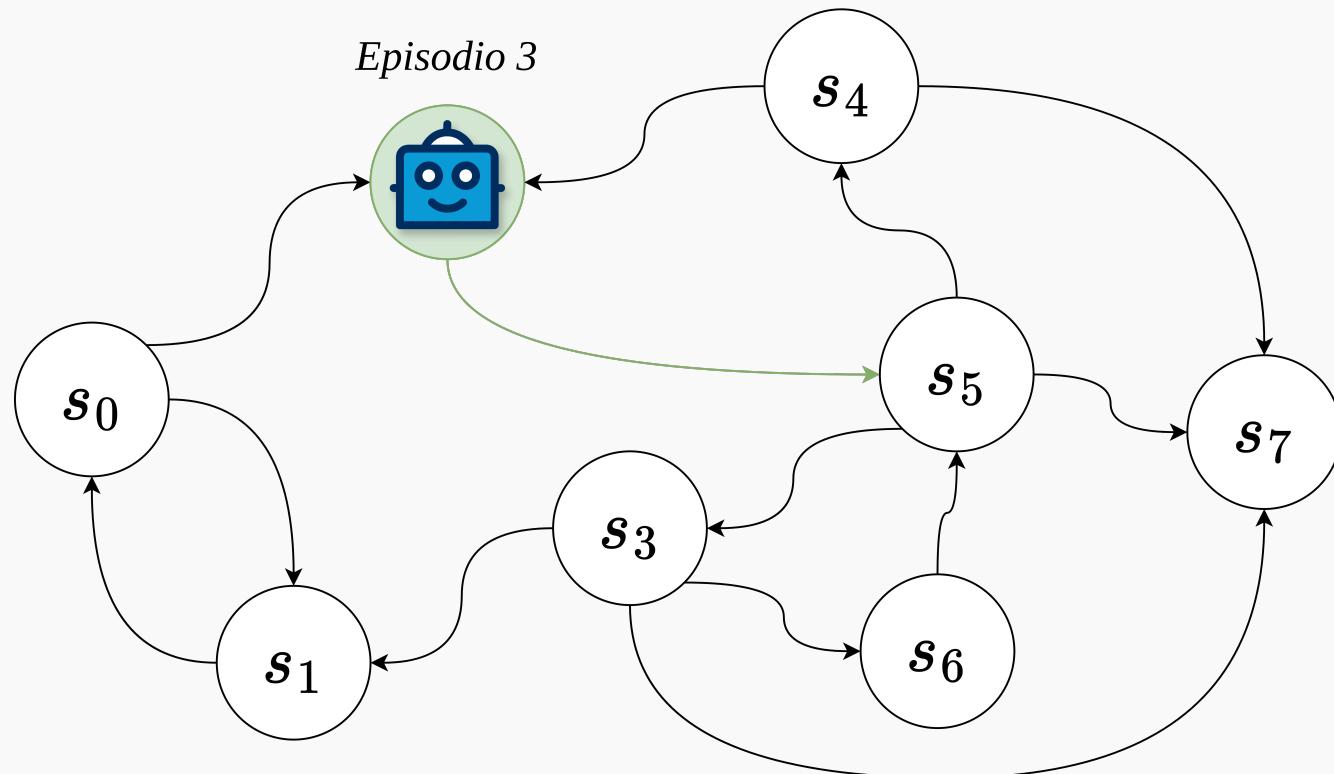
Ejemplo



Ejemplo



Ejemplo



Inicios de exploración

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$

$Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$

$Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}, A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0

Generate an episode from S_0, A_0 , following π : $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1, \dots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \arg \max_a Q(S_t, a)$

Inicios de exploración

El **problema** es que **no siempre podemos emplear inicios de exploración**.

- Existen problemas en los que es difícil comenzar es un par estado-acción aleatorio.

Es difícil asegurar que un agente pueda empezar desde toda configuración posible, especialmente en problemas lo suficientemente complejos.

- Problemas con espacios de estados / acciones **continuos**.
- Ineficiencia: estados / acciones **inaccesibles** desde el estado inicial real.
- ...



Inicios de exploración

El **problema** es que **no siempre podemos emplear inicios de exploración**.

- Existen problemas en los que es difícil comenzar es un par estado-acción aleatorio.

Es difícil asegurar que un agente pueda empezar desde toda configuración posible, especialmente en problemas lo suficientemente complejos.

- Problemas con espacios de estados / acciones **continuos**.
- Ineficiencia: estados / acciones **inaccesibles** desde el estado inicial real.
- ...



¿SOLUCIÓN?

Aprendizaje por refuerzo

Métodos basados en muestreo (1)

Antonio Manjavacas Lucas

manjavacas@ugr.es