

# **APRENDIZAJE POR REFUERZO**

Métodos basados en muestreo: Monte Carlo

---

Antonio Manjavacas

[manjavacas@ugr.es](mailto:manjavacas@ugr.es)

# CONTENIDOS

---

- 1. Predicción Monte Carlo**
- 2. Control Monte Carlo**
- 3. Métodos *on-policy***
- 4. Métodos *off-policy***

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*

Los **métodos basados en *sampling*** (muestreo) 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

---

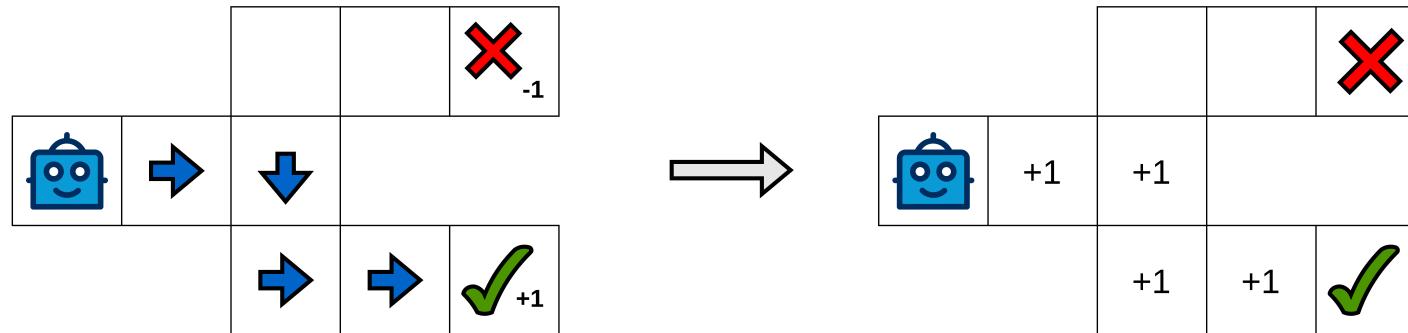
## Método Monte Carlo

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

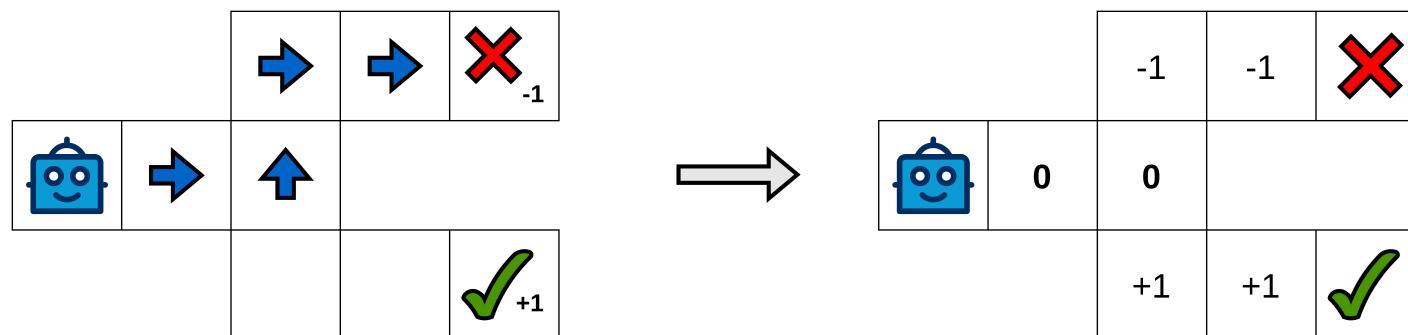
La idea básica detrás de la **predicción Monte Carlo** (*Monte Carlo prediction*) es seguir múltiples trayectorias aleatorias desde diferentes estados.

- 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**).

## EPISODIO 1



## EPISODIO 2



Monte Carlo requiere solamente **experiencia**.

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



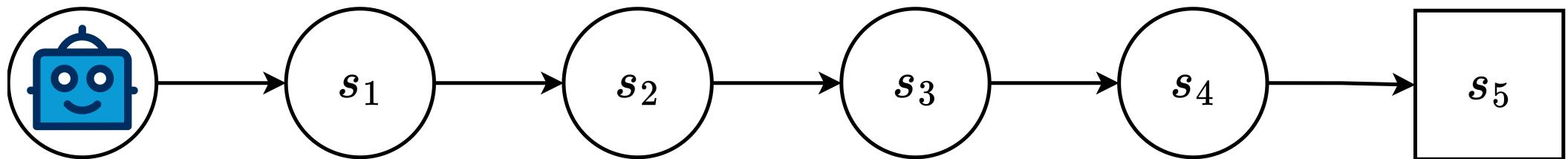
Permite la resolución de problemas de RL a partir del **promedio de las recompensas finales obtenidas** (*average sample returns*).

- 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).

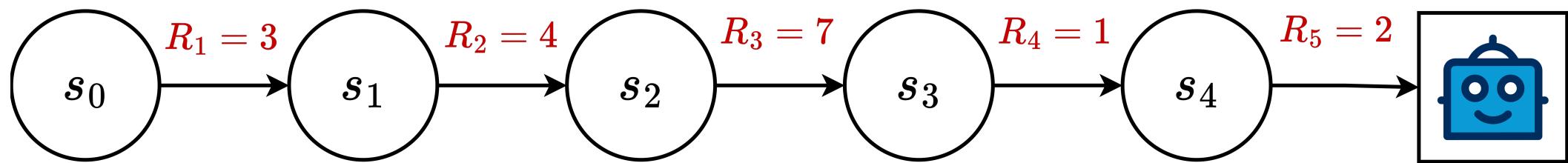
Tratamos de estimar  $v(s)$ , con  $\gamma = 0.5$

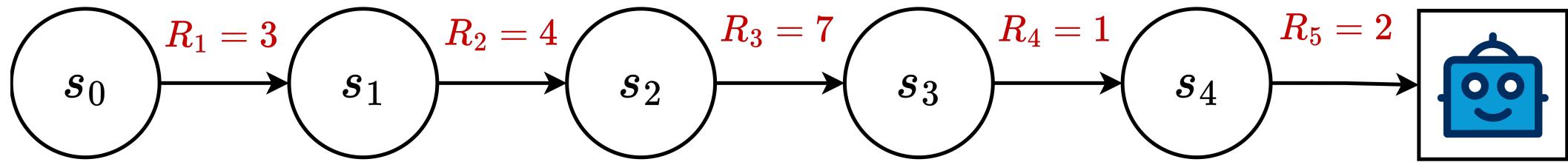
- Recordemos que  $v(s) = \mathbb{E}[G_t \mid S_t = s]$

Comenzamos realizando una **trayectoria aleatoria**:



Acumulamos **experiencia**...





Calculamos el **retorno** para cada estado:

$$G_0 = R_1 + \gamma G_1$$

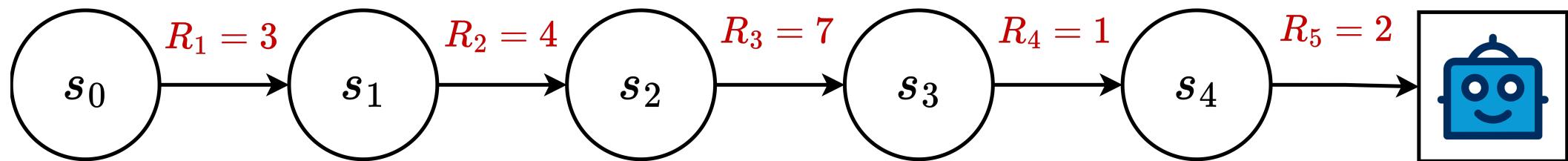
$$G_3 = R_4 + \gamma G_4$$

$$G_1 = R_2 + \gamma G_2$$

$$G_4 = R_5 + \gamma G_5$$

$$G_2 = R_3 + \gamma G_3$$

$$G_5 = 0$$



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

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

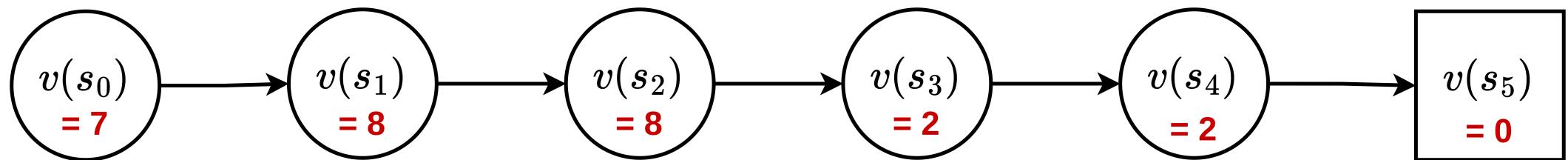
$$G_2 = R_3 + \gamma G_3 = 7 + 0.5 \cdot 2 = 8$$

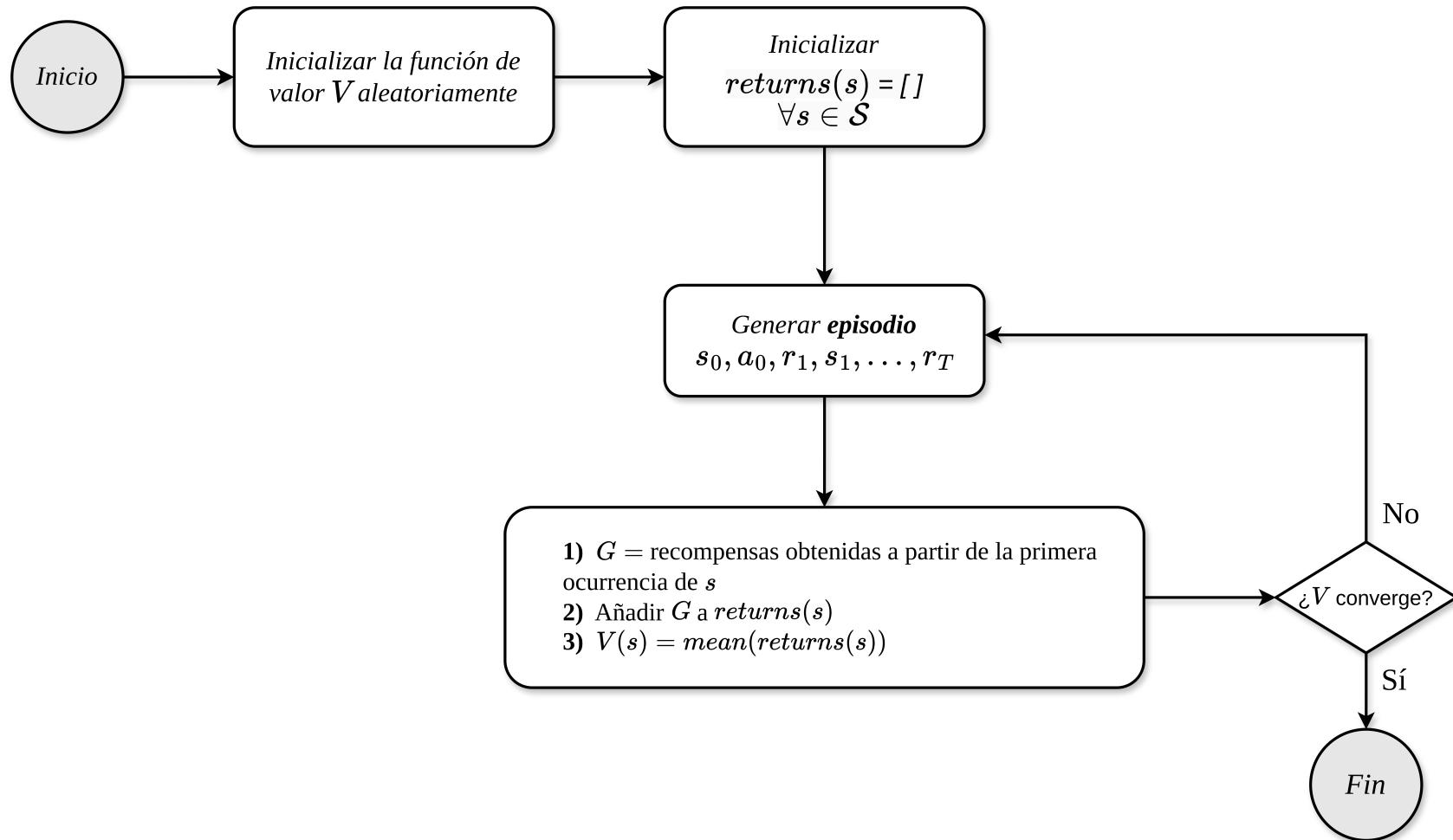
$$G_3 = R_4 + \gamma G_4 = 1 + 0.5 \cdot 2 = 2$$

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

$$G_5 = 0$$

Valores estimados:





- 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...

**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  $V$ 

```

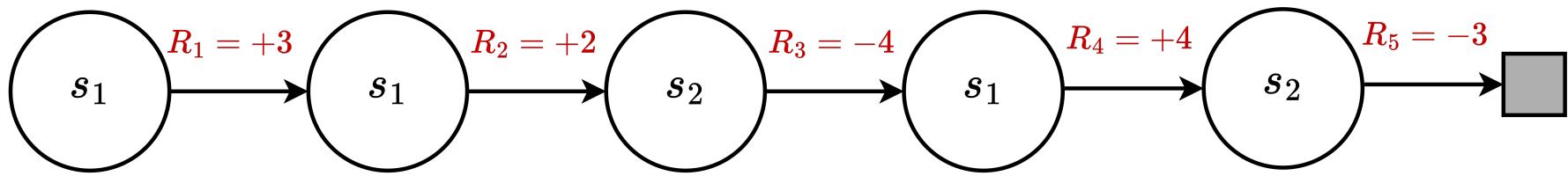
**Algorithm 2:** Every-Visit MC Prediction

```

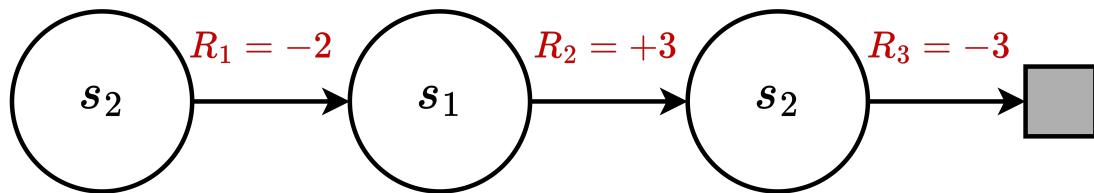
Input: policy  $\pi$ , positive integer  $num\_episodes$ 
Output: value function  $V$  ( $\approx v_\pi$ , if  $num\_episodes$  is large enough)
Initialize  $N(s) = 0$  for all  $s \in \mathcal{S}$ 
Initialize  $Returns(s) = 0$  for all  $s \in \mathcal{S}$ 
for episode  $e \leftarrow 1$  to  $e \leftarrow num\_episodes$  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
    end
     $V(s) \leftarrow \frac{Returns(s)}{N(s)}$  for all  $s \in \mathcal{S}$ 
return  $V$ 

```

**EPISODIO 1**

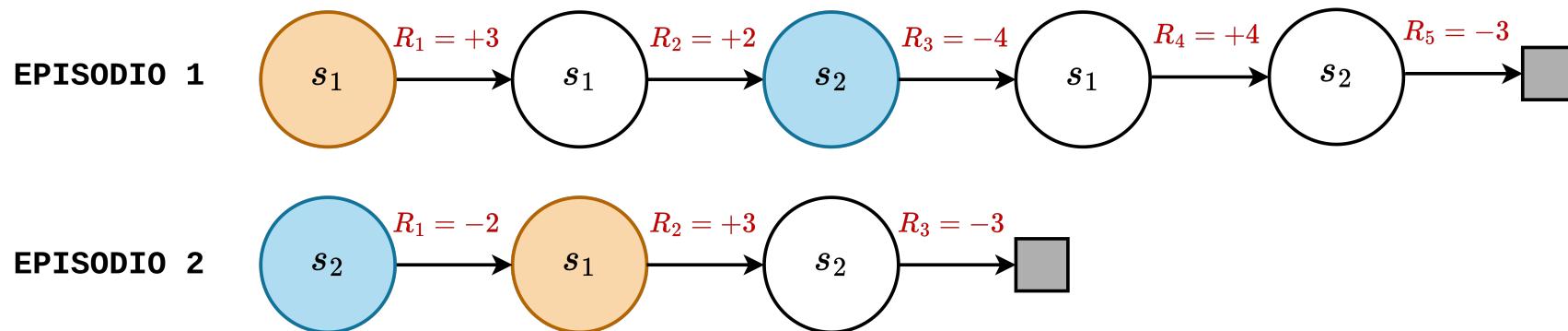


**EPISODIO 2**



**FIRST-VISIT MC**

Consideramos las recompensas acumuladas a partir de la primera visita.

**Episodio 1:**

$$V(s_1) = 3 + 2 + -4 + 4 - 3 = 2$$

$$V(s_2) = -4 + 4 - 3 = -3$$

**Episodio 2:**

$$V(s_1) = 3 - 3 = 0$$

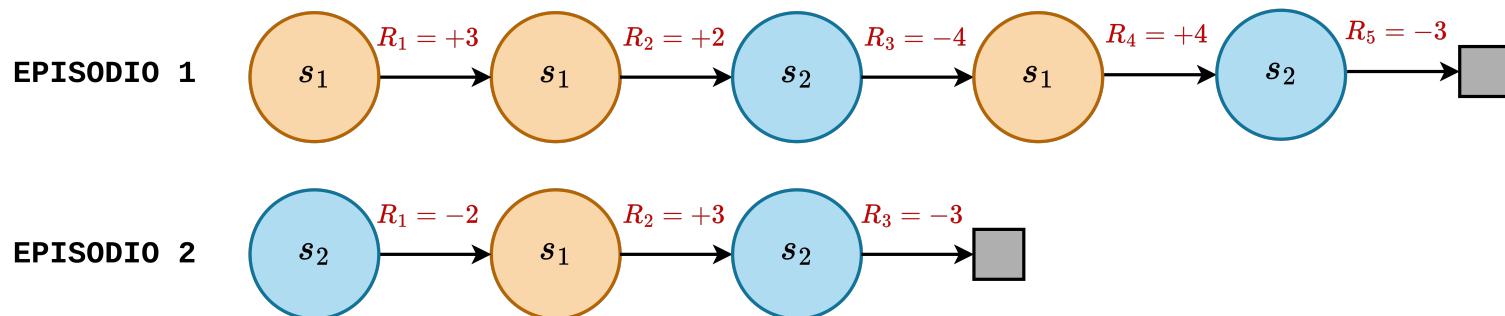
$$V(s_2) = -2 + 3 - 3 = -2$$

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

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

## EVERY-VISIT MC

Consideramos las recompensas acumuladas a partir todas las visitas.



### Episodio 1:

$$V(s_1) = (3 + 2 - 4 + 4 - 3) + (2 - 4 + 4 - 3) + (4 - 3) = 2$$

$$V(s_2) = (-4 + 4 - 3) + (-3) = 0$$

### Episodio 2:

$$V(s_1) = 3 - 3 = 0$$

$$V(s_2) = (-2 + 3 - 3) + (-3) = 1$$

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

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

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 se emplea **bootstrapping (estimaciones a partir de estimaciones)**.

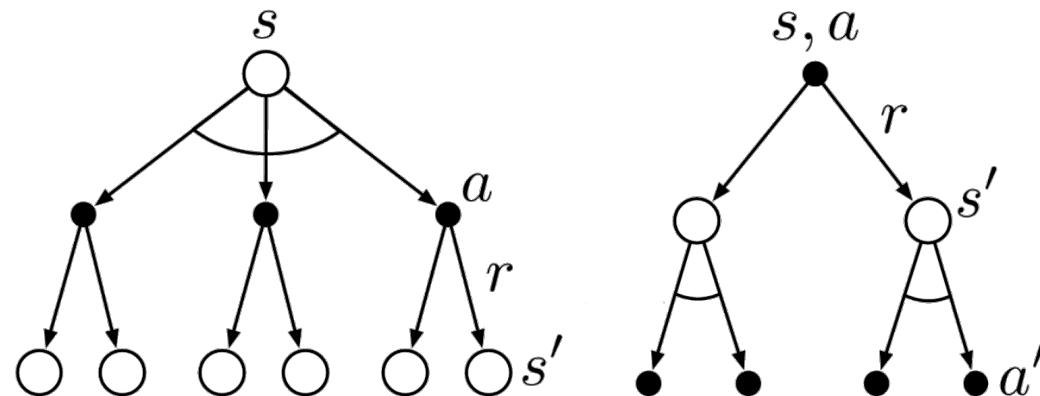
Esto puede ser útil cuando solamente queremos saber el valor de un subconjunto de estados (ignorando el resto).

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.



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



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

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

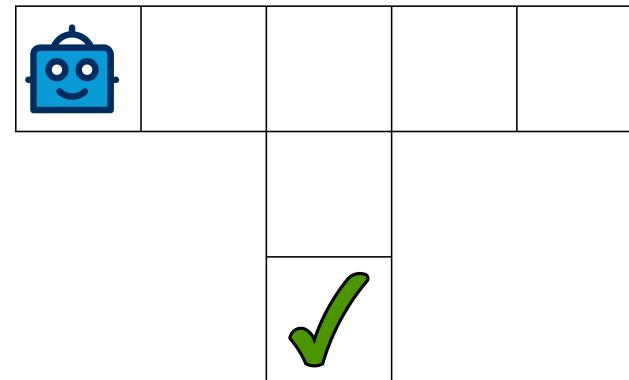
**Con un modelo del entorno**, los valores de los estados son suficientes para saber qué política seguir.

- 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**.

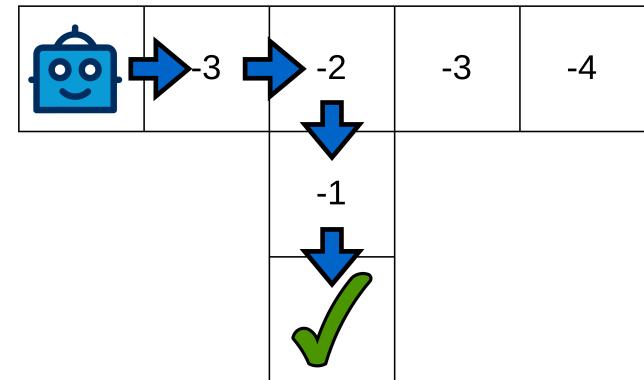
**Con un modelo del entorno**, el agente sólo tiene que estimar los valores de los estados y actuar de forma *greedy* para alcanzar su objetivo.



Robot icon	-3	-2	-3	-4
	-1			

Primero estimamos los valores  
de los estados...

...y después obtenemos la  
política óptima



**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

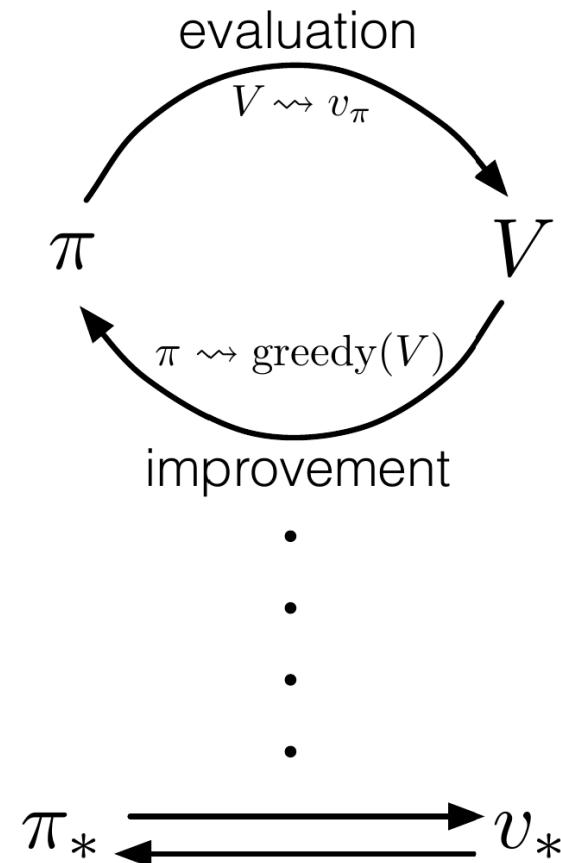
$$\pi_0 \rightarrow q_0 \rightarrow \pi_1 \rightarrow q_1 \rightarrow \dots \rightarrow \pi_* \rightarrow q_*$$

## 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...**

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

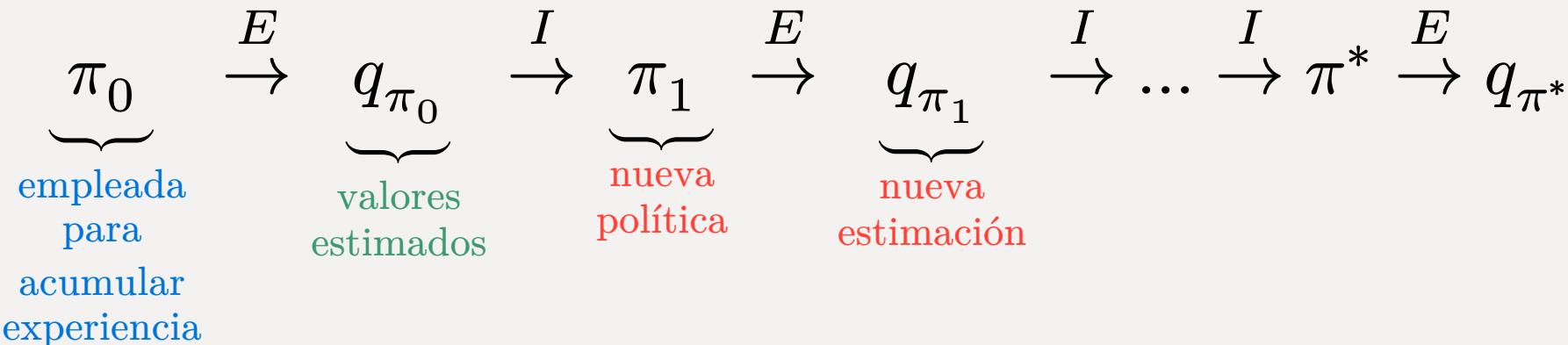
Para cada función de valor  $q$ , la política *greedy* correspondiente es aquella tal que  $\forall s \in \mathcal{S}$  elige de forma **determinista** una acción con valor máximo:

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

Es decir,  $\pi_{k+1}$  es siempre la política **greedy** con respecto a  $q_{\pi_k}$ .



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



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

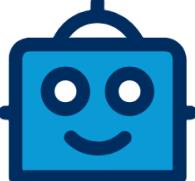
- La convergencia se alcanza cuando la política y la función de valor son óptimas.

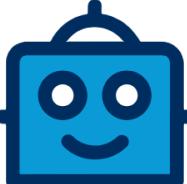
# Ejemplo

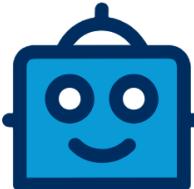
Las recompensas son los valores en cada casilla. Asumimos  $\gamma = 1$ .

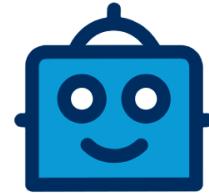
$s_1$ -1	$s_2$ -1	$s_3$ -1
$s_4$ -1	$s_5$ -1	$s_6$ -1
$s_7$ -1	$s_8$ <b>-10</b>	$s_9$ <b>+10</b>

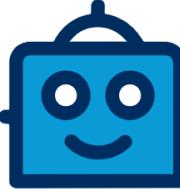
	-1	-1
-1	-1	-1
-1	<b>-10</b>	<b>+10</b>

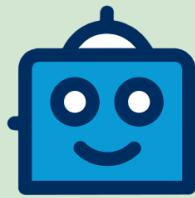
-1	-1	-1
	-1	-1
-1	<b>-10</b>	<b>+10</b>

-1	-1	-1
-1		-1
-1	<b>-10</b>	<b>+10</b>

-1		-1
-1	-1	-1
-1	<b>-10</b>	<b>+10</b>

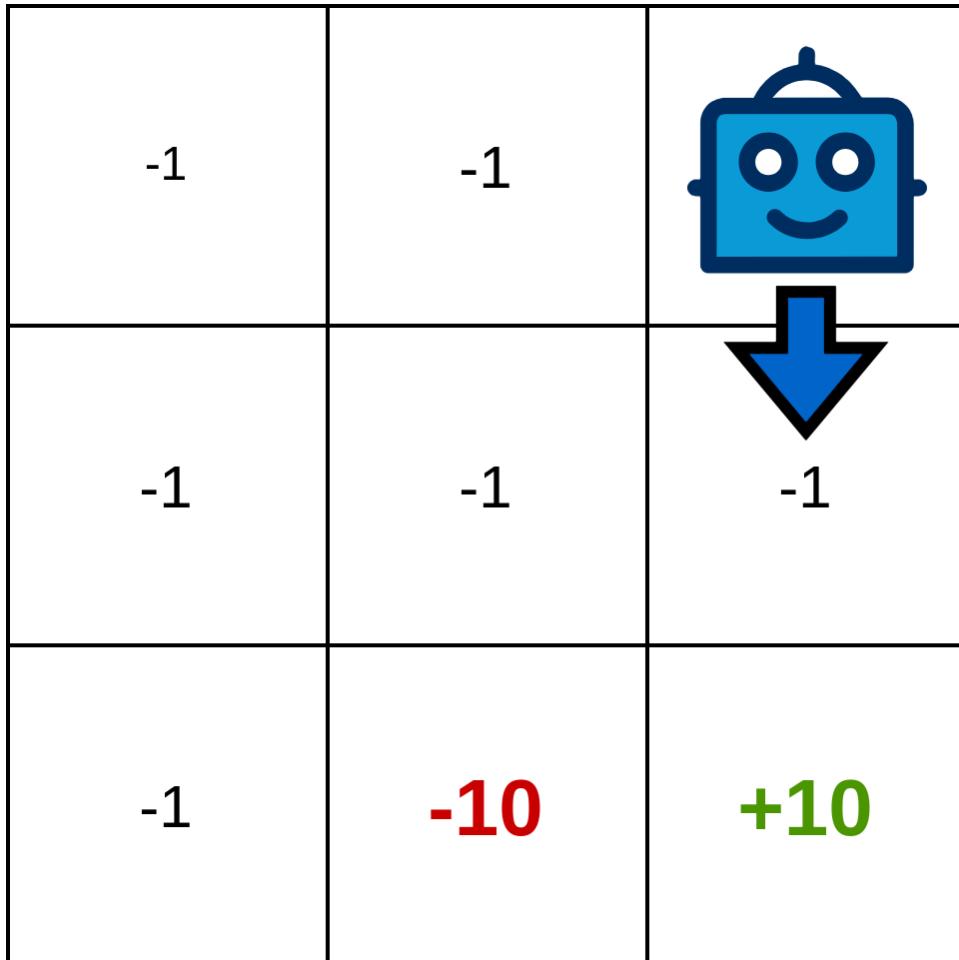
-1	-1	
-1	-1	-1
-1	<b>-10</b>	<b>+10</b>

-1	-1	-1
-1	-1	
-1	<b>-10</b>	<b>+10</b>

-1	-1	-1
-1	-1	-1
-1	<b>-10</b>	

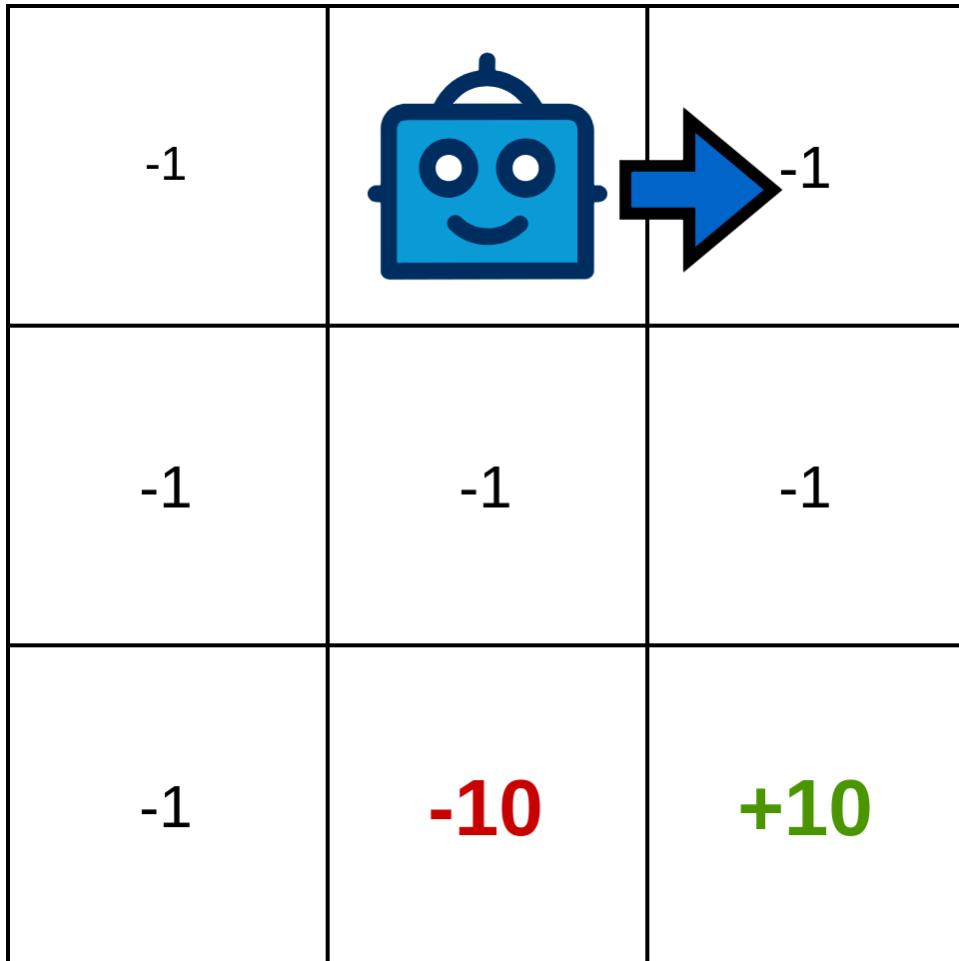
-1	-1	-1
-1	-1	
-1	<b>-10</b>	<b>+10</b>

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



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

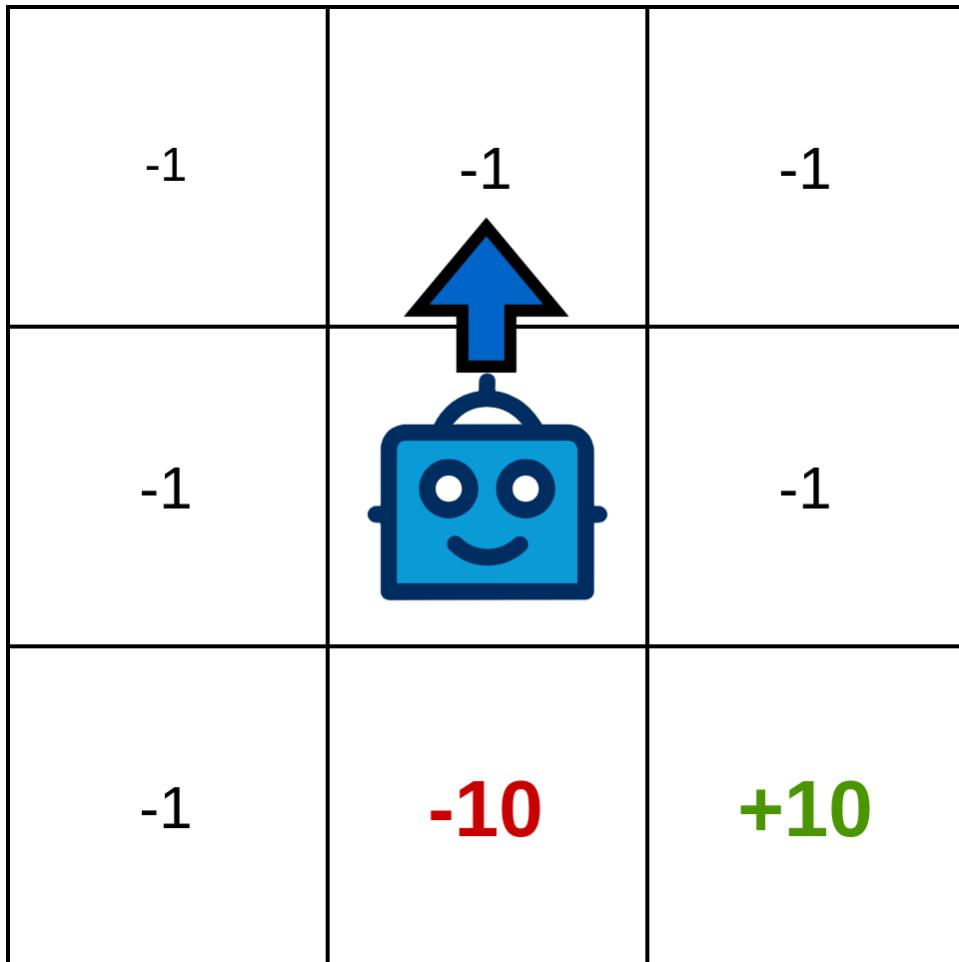
$$q(s_3, \downarrow) = (-1) + 10 = 9$$



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

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

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

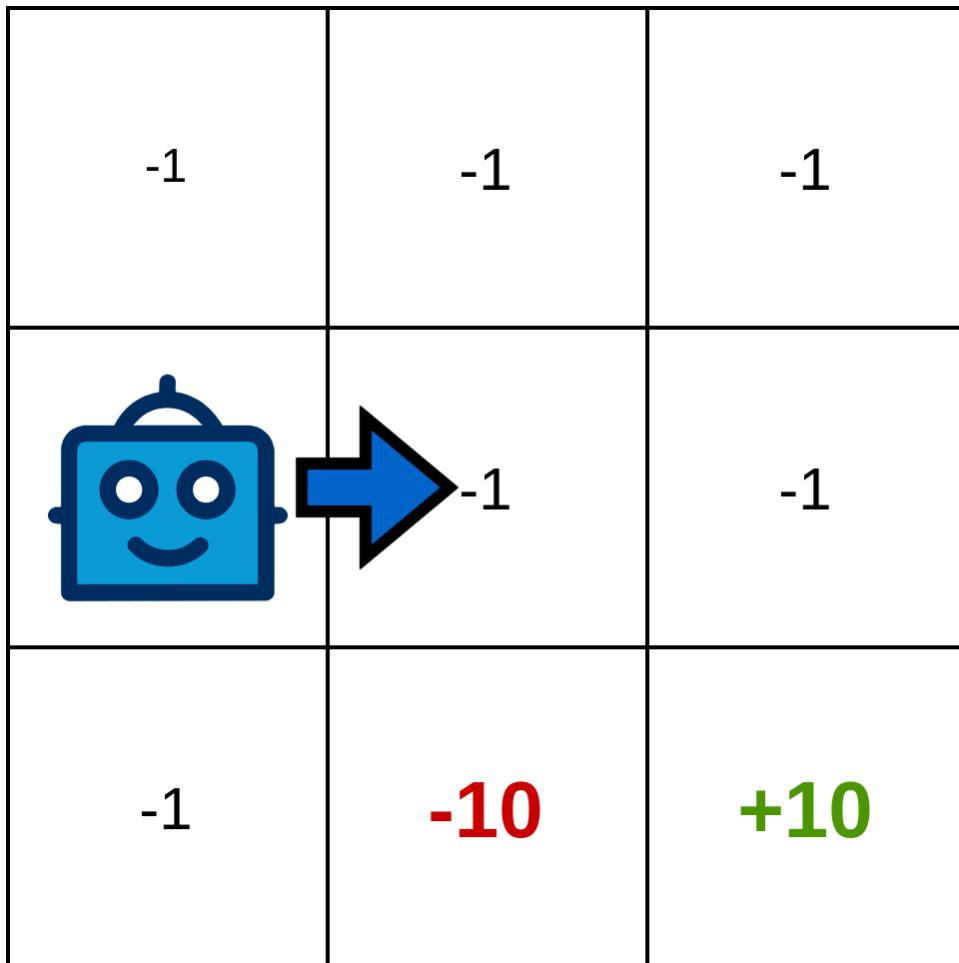


$$q(s_6, \downarrow) = 10$$

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

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

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



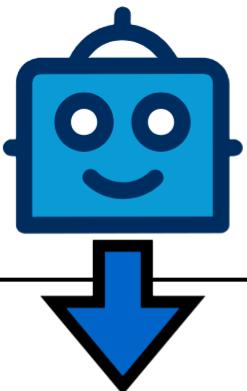
$$q(s_6, \downarrow) = 10$$

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

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

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

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



-1	-1	-1
-1	-1	-1
-1	<b>-10</b>	<b>+10</b>

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

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

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

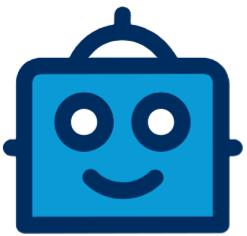
$$q(s_5, \uparrow) = 7$$

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

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

# Ejemplo

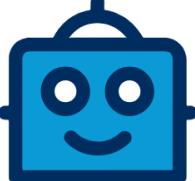
47/131

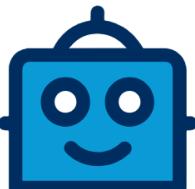
	-1	-1
-1	-1	-1
-1	<b>-10</b>	<b>+10</b>

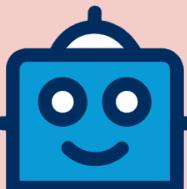
				
$s_1$		<b>5</b>		
$s_2$				<b>8</b>
$s_3$		<b>9</b>		
$s_4$				<b>6</b>
$s_5$	<b>7</b>			
$s_6$		<b>10</b>		
$s_7$				
...	...	...	...	...

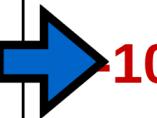
*Siguiente iteración...*

	-1	-1
-1	-1	-1
-1	<b>-10</b>	<b>+10</b>

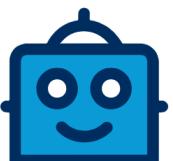
-1	-1	-1
	-1	-1
-1	<b>-10</b>	<b>+10</b>

-1	-1	-1
-1	-1	-1
	<b>-10</b>	<b>+10</b>

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

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

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

-1	-1	-1
 	-1	-1
-1	<b>-10</b>	<b>+10</b>

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

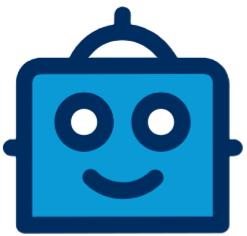
$$q(s_4, \downarrow) = (-1) + (-10) = -11$$

 	-1	-1
-1	-1	-1
-1	<b>-10</b>	<b>+10</b>

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

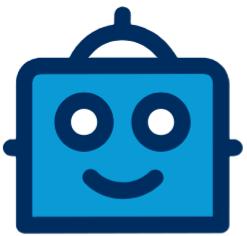
$$q(s_4, \downarrow) = -11$$

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

	-1	-1
-1	-1	-1
-1	<b>-10</b>	<b>+10</b>

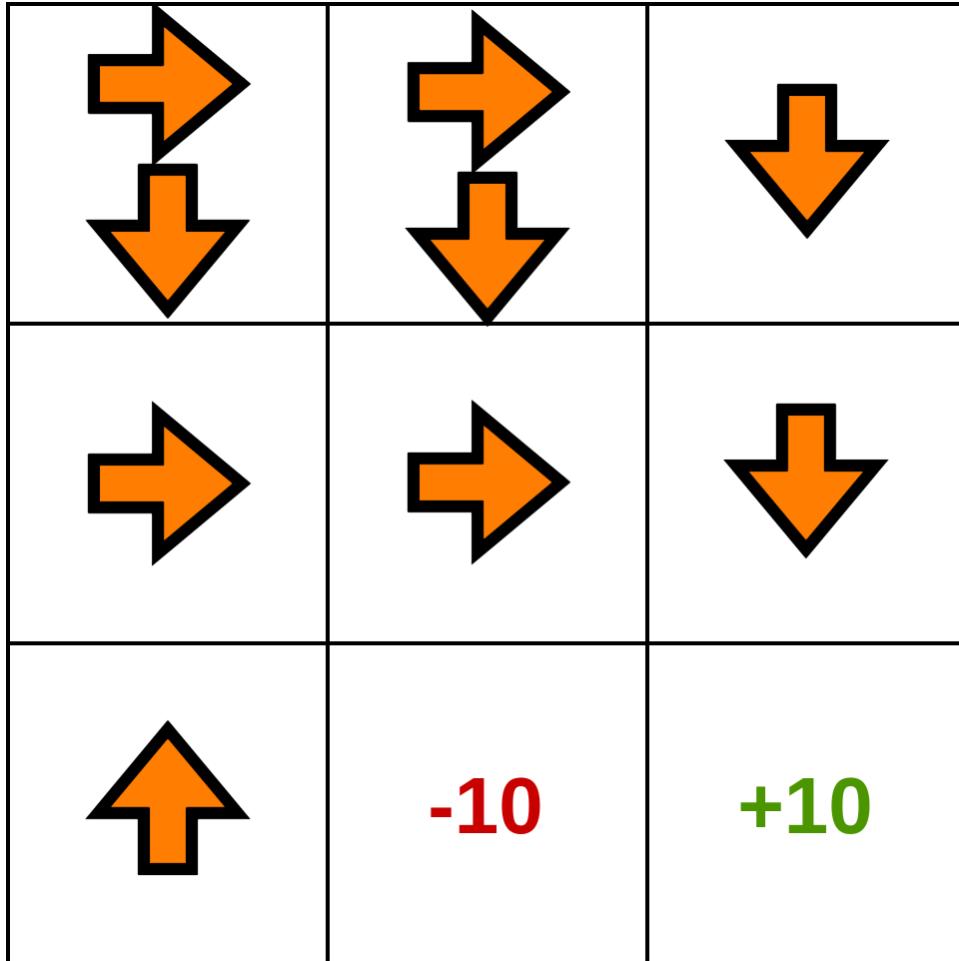
				
$s_1$		$\frac{5+(-12)}{2} = -3.5$		
$s_2$				8
$s_3$		9		
$s_4$		<b>-11</b>		6
$s_5$	7			
$s_6$		10		
$s_7$				<b>-10</b>
...	...	...	...	...



	-1	-1
-1	-1	-1
-1	<b>-10</b>	<b>+10</b>

Los **valores** convergen...

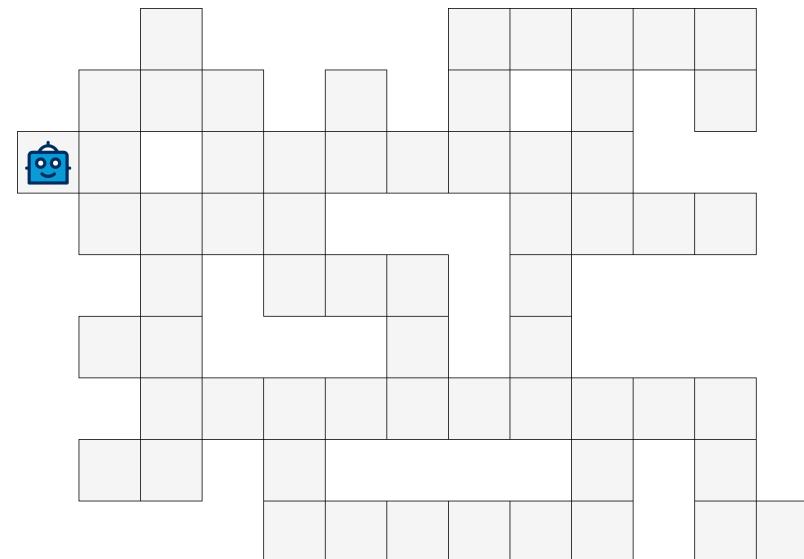
	↑	↓	←	→
$s_1$	6	7	6	7
$s_2$	7	8	6	8
$s_3$	8	9	7	8
$s_4$	6	6	7	8
$s_5$	7	-10	7	9
$s_6$	8	10	8	9
$s_7$	7	6	6	-10
$s_8$	-	-	-	-
$s_9$	-	-	-	-



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
$s_8$	-	-	-	-
$s_9$	-	-	-	-

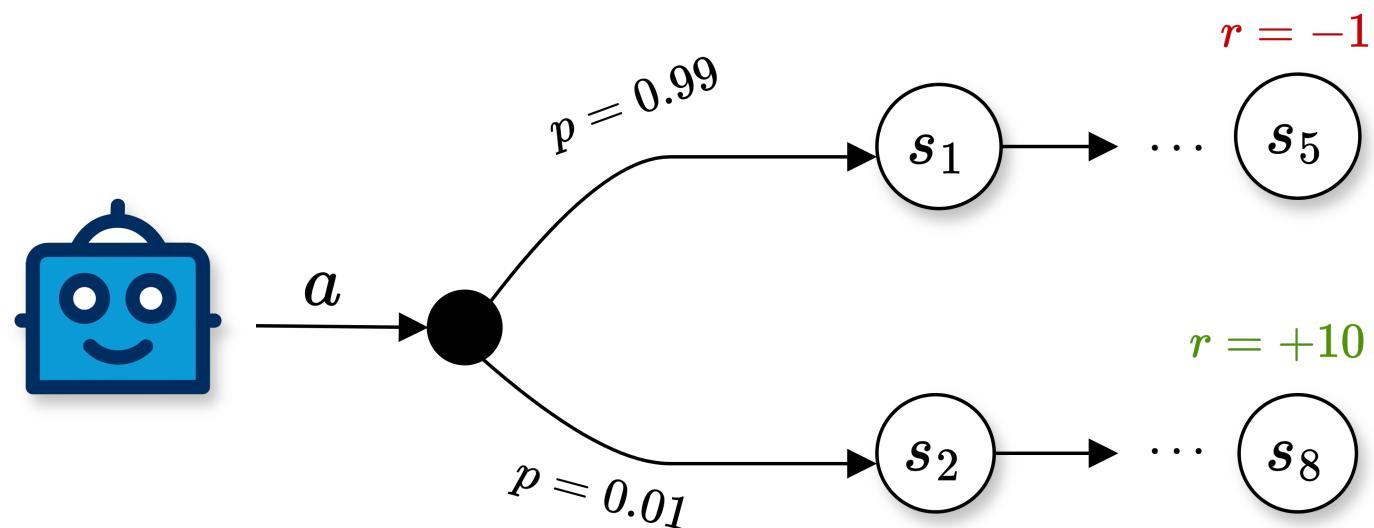
Un problema de MC a la hora de estimar valores es que algunos pares acción-estado **podrían no visitarse nunca...**



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

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**...





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  $s \in \mathcal{S}$  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*).

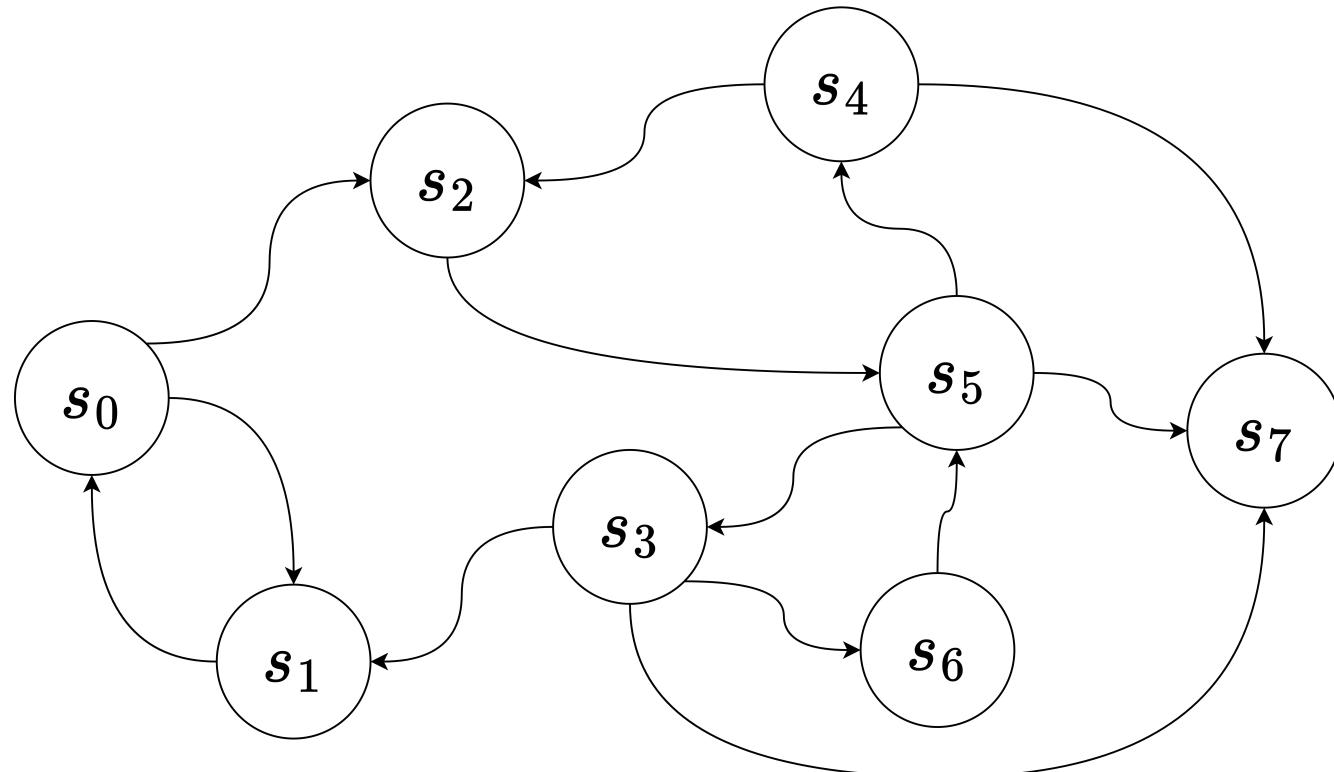
Se trata de una forma de asegurar **exploración continua**.

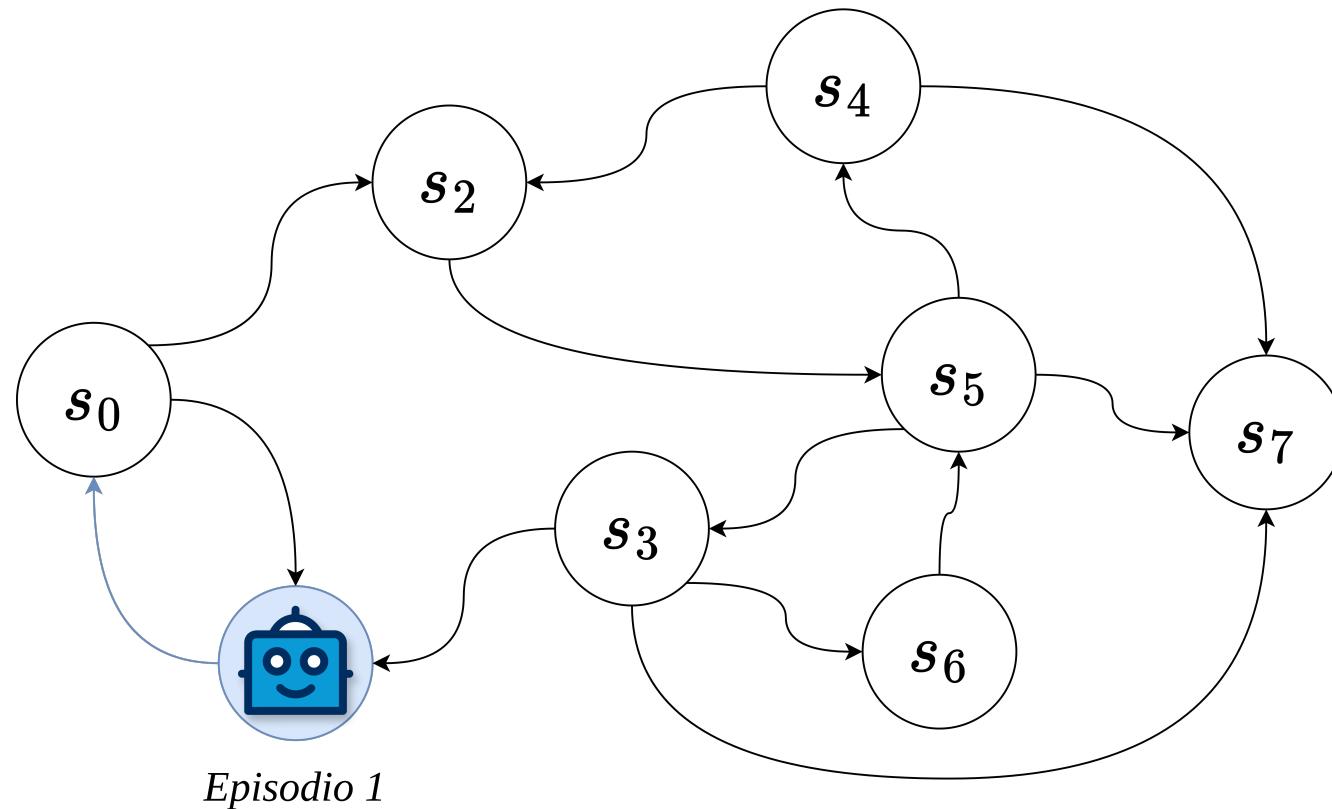
- Todo episodio empieza desde un **par estado-acción aleatorio**:

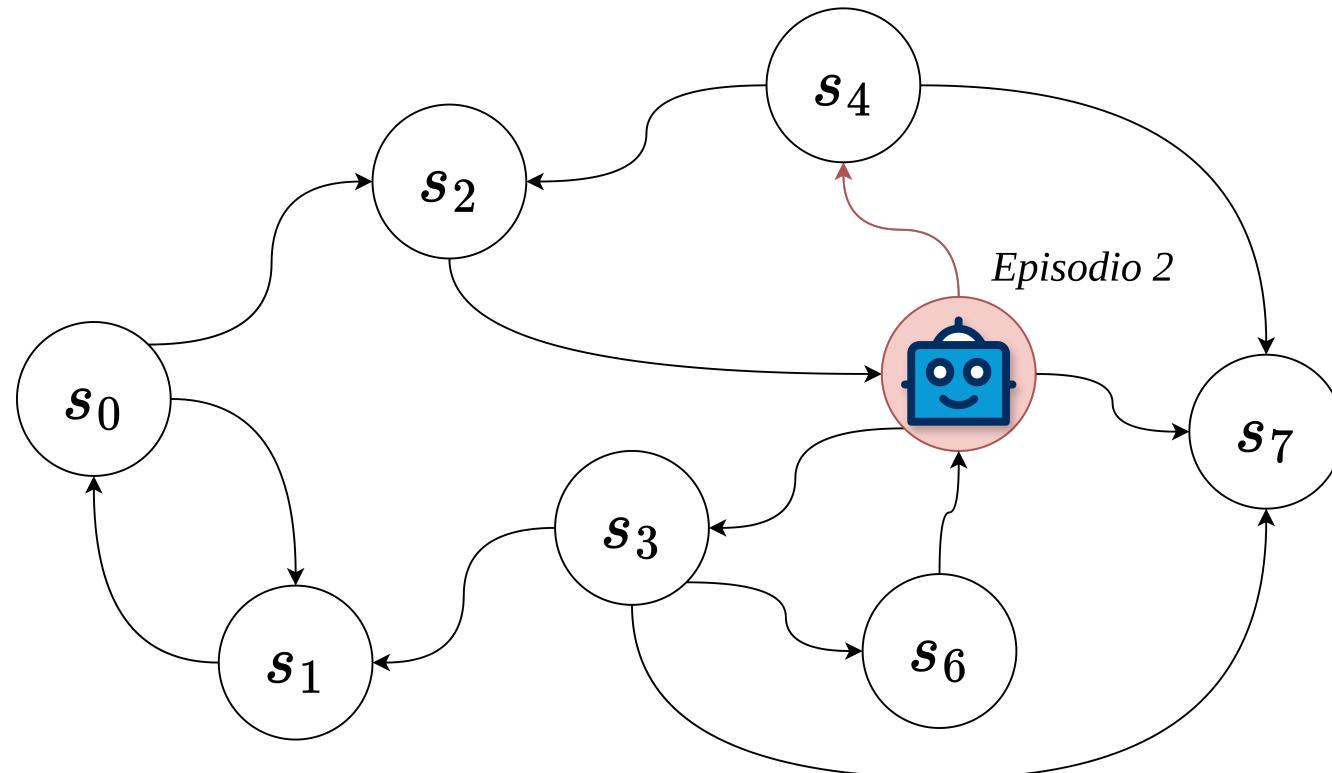
$$\underbrace{s_0, a_0}_{\text{Aleatorios}}, \underbrace{s_1, a_1, s_2, a_2, \dots}_{\text{Dependientes de } \pi, p}$$

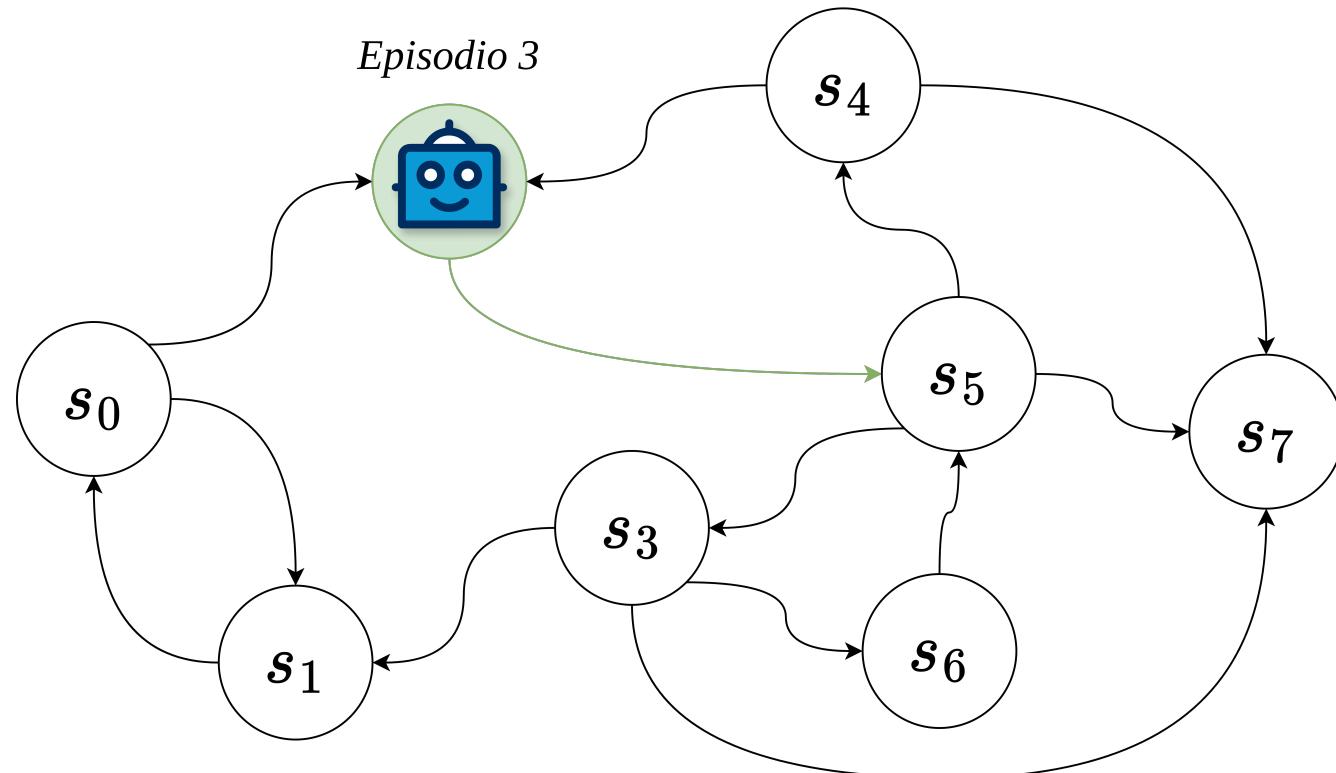
- Cada par estado-acción tiene una probabilidad **no nula** de ser seleccionado **al principio de un episodio**:

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









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

Initialize:

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

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

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

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  $\pi$ :  $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

$$G \leftarrow 0$$

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  $Returns(S_t, A_t)$

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

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

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 o acciones **continuos**.
- Ineficiencia: estados o acciones **inaccesibles** desde el estado inicial real.
- ...

**¿SOLUCIÓN?**



Control MC sin inicios de  
exploración

Vamos a estudiar dos alternativas a MC con inicios de exploración...

## Métodos *on-policy*

- 👤 Se emplea **una única política** que mejora progresivamente, permitiendo siempre cierta exploración.
  - Mejoran y evalúan constantemente la misma política.

## Métodos *off-policy*

- 👥 El agente aprende una **política objetivo** (*target policy*) a partir de datos generados por otra **política exploratoria** (*behaviour policy*).
  - La política que empleamos para aprender/explorar «está fuera» (*off*) de la que empleamos para seleccionar acciones.

# **MÉTODOS ON-POLICY**

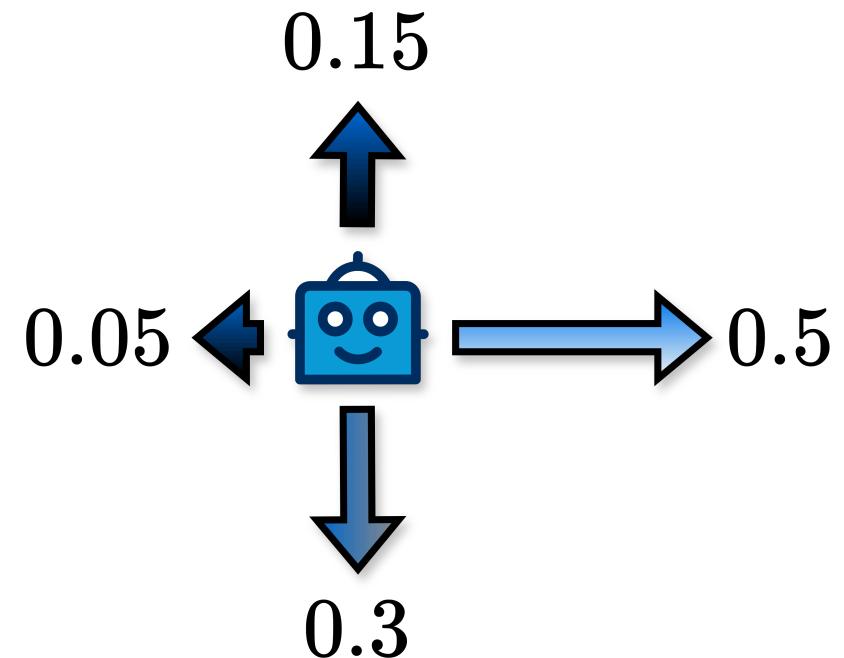
---

Los métodos  ***on-policy*** emplean **una única política**.

Esta política *aspira* a un comportamiento óptimo, pero siempre debe reservar **cierta probabilidad de explorar**.

Las políticas empleadas generalmente son ***soft*** («suaves»), es decir:

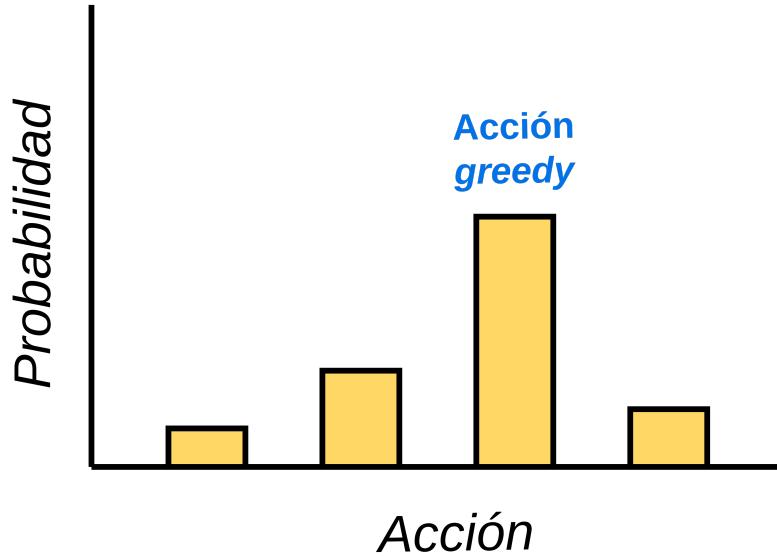
$$\pi(a|s) > 0 \quad \forall s \in \mathcal{S}, a \in \mathcal{A}$$



MC con inicios de exploración es *on-policy* ...  
... aunque poco viable, como hemos adelantado.

Una opción más apropiada son las **políticas  $\varepsilon$ -greedy**.

Las políticas  $\varepsilon$ -*greedy* son políticas estocásticas que siempre permiten cierta probabilidad  $\varepsilon > 0$  de explorar.

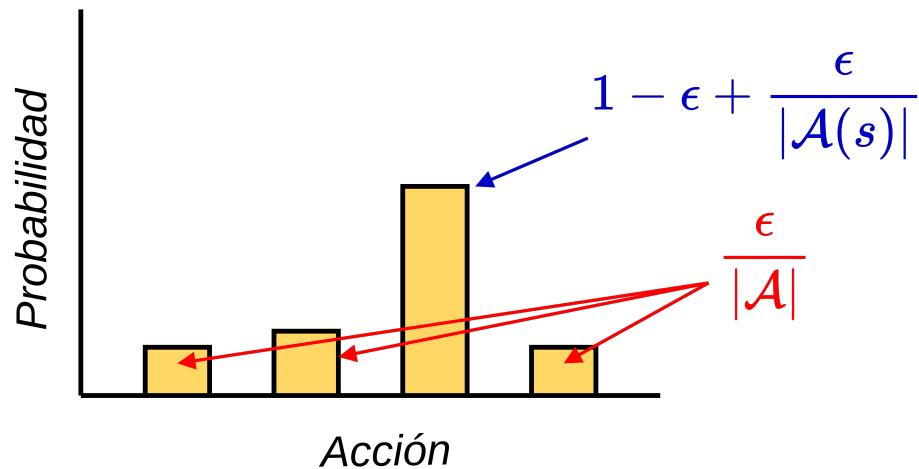


La **acción greedy** es aquella que se elige con mayor probabilidad.

Eventualmente, el resto de acciones (no óptimas) podrían explorarse con probabilidad  $\varepsilon$ .

- El valor de  $\varepsilon$  puede reducirse gradualmente, hasta que la política sea prácticamente determinista.

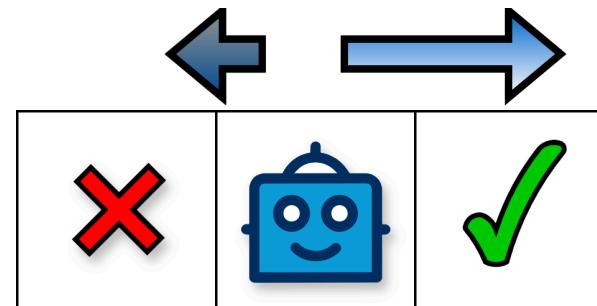
$\varepsilon$ -greedy es un subconjunto de las políticas conocidas como  **$\varepsilon$ -soft**.



Siempre permiten cierta exploración.  
En el caso de  $\varepsilon$ -greedy:

$$\pi(a|s) \geq \frac{\varepsilon}{|\mathcal{A}(s)|}$$

- Si  $\varepsilon > 0$  estas políticas nunca pueden ser óptimas. Esto se debe a que **siempre existe cierta probabilidad de realizar acciones sub-óptimas** (explorar).
- No convergen en una política óptima, pero sí en una **muy aproximada**. Además, evitan emplear inicios de exploración.



On-policy first-visit MC control (for  $\varepsilon$ -soft policies), estimates  $\pi \approx \pi_*$

Algorithm parameter: small  $\varepsilon > 0$

Initialize:

$\pi \leftarrow$  an arbitrary  $\varepsilon$ -soft policy

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

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

Repeat forever (for each episode):

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

$G \leftarrow 0$

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  $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow$  average( $Returns(S_t, A_t)$ )

$A^* \leftarrow \operatorname{argmax}_a Q(S_t, a)$  (with ties broken arbitrarily)

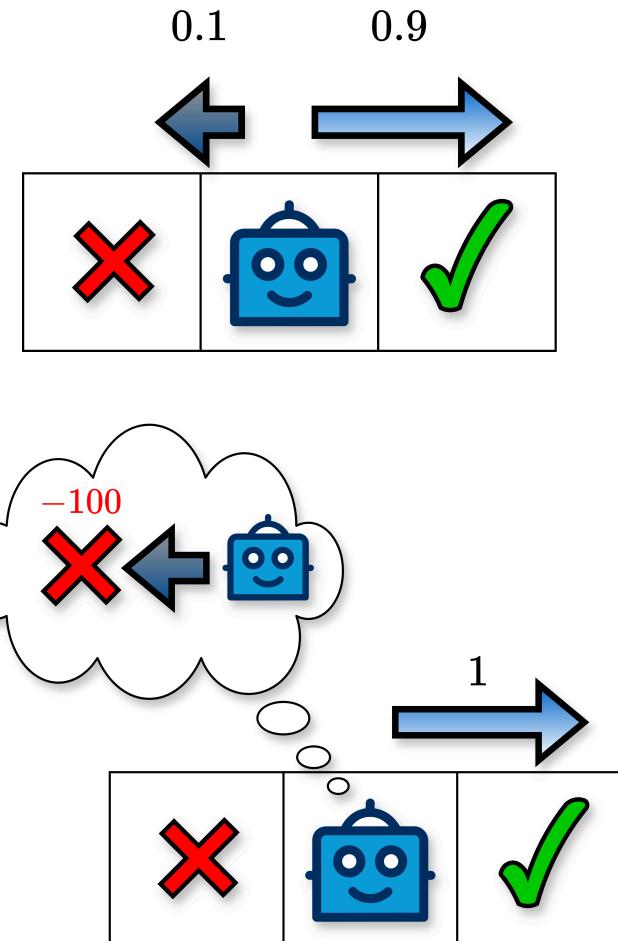
For all  $a \in \mathcal{A}(S_t)$ :

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

? ¿Existe alguna **alternativa** a mantener siempre cierta probabilidad de explorar?

- MC *on-policy* supone aprender una política **muy cercana a la óptima**, pero siempre existe cierta probabilidad de elegir acciones sub-óptimas.

Los métodos **off-policy** son una alternativa.



# **MÉTODOS OFF-POLICY**

---

Los métodos  ***off-policy*** hacen uso de **dos políticas**:

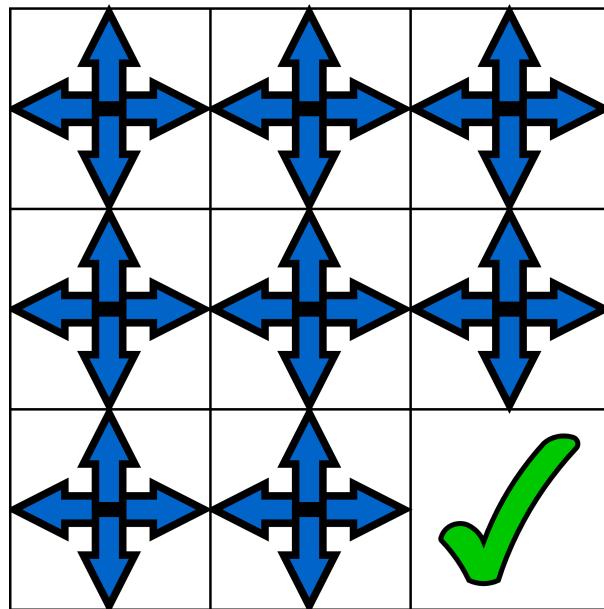
**Política objetivo** (*target policy*). Destinada a ser óptima.

**Política de comportamiento** (*behaviour policy*). Política exploratoria empleada para «generar comportamiento» (muestrear, acumular experiencia).

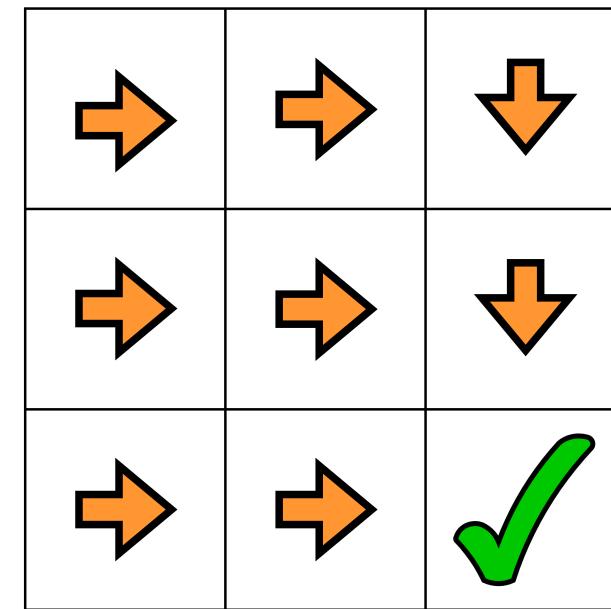
En este caso, decimos que **el aprendizaje de la política óptima se hace a partir de datos/resultados «fuera» (*off*) de la política objetivo**.

- Es decir, mediante información obtenida por la política de comportamiento.

*Política de  
comportamiento*



*Política objetivo*



Una analogía...



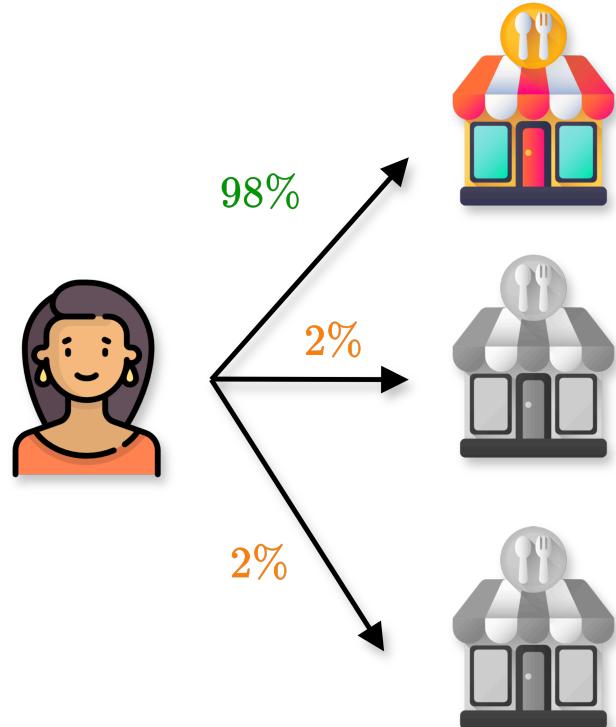
## On-policy

Imagina que tienes un restaurante favorito al que sueles ir a comer (**acción greedy**).

- Al principio, es posible que tu criterio no sea muy preciso pero, a medida que visitas todos los restaurantes varias veces, cada vez repites más el mismo.

Algunos días vuelves a otros restaurantes que consideras peores para ver si la calidad ha mejorado (**exploración**).

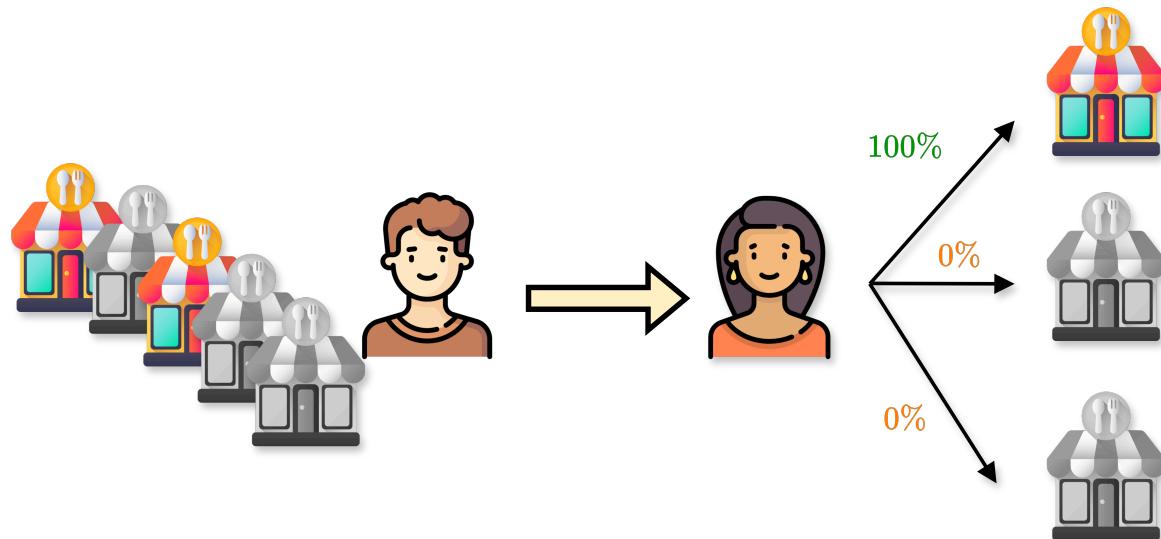
- Eventualmente estás abierto a dar una nueva oportunidad a restaurantes peores.



## Off-policy

Dejas que otra persona pruebe todos los restaurantes de la ciudad durante un tiempo (**política de comportamiento**). En base a su experiencia, eliges ir siempre al restaurante que te recomiende (**política objetivo**).

- Tú no pruebas nuevos restaurantes (no exploras), lo hace alguien por ti.





Los métodos **on-policy** son más simples, porque sólo se requiere una política.



Los métodos **off-policy** requieren más tiempo para converger y presentan una mayor varianza (cambios de comportamiento más bruscos).

- No obstante, son más potentes y generales.
- De hecho, **son una generalización de los métodos on-policy**, en el caso concreto en que las políticas objetivo y de comportamiento sean las mismas.

 Los algoritmos *off-policy* suelen emplearse para aprender a partir de datos generados por otro controlador/algoritmo, o por humanos (***imitation learning***).

*¿Cuándo es preferible el aprendizaje off-policy?*

- Aprendizaje a partir de datos generados por **humanos u otros agentes**.
- Aprendizaje a partir de la experiencia generada por **políticas anteriores**.
  - Reutilización de experiencia proveniente de versiones anteriores de la misma política.
- Aprendizaje de una política óptima **determinista empleando otra política exploratoria**.
- Aprendizaje a partir de la experiencia de **múltiples políticas combinadas**.

# Predicción *off-policy*

**Objetivo:** estimar  $v_\pi$  o  $q_\pi$  dadas las políticas  $\pi$  (objetivo) y  $b$  (comportamiento).

Si queremos emplear episodios de  $b$  para estimar valores para  $\pi$ , es necesario que cada acción tomada por  $b$  la pueda tomar también  $\pi$  (al menos, eventualmente).

Denominamos a esto **supuesto de cobertura**:

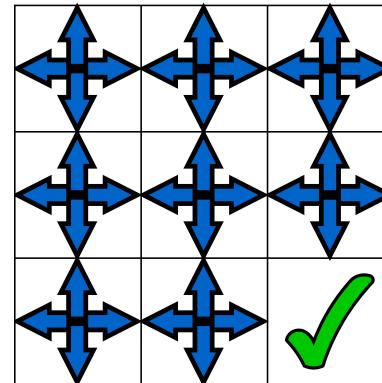
Si  $\pi(a|s) > 0$ , entonces  $b(a|s) > 0$

- $b$  es una política **estocástica** (no tiene por qué serlo al 100%, puede ser  $\varepsilon$ -greedy).
- $\pi$  puede ser **determinista** o **estocástica**.

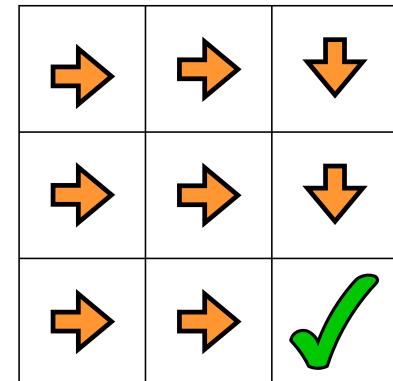
Generalmente consideraremos políticas objetivo  $\pi$  **deterministas**.

Aunque existen problemas donde puede ser útil que  $\pi$  sea **estocástica**.

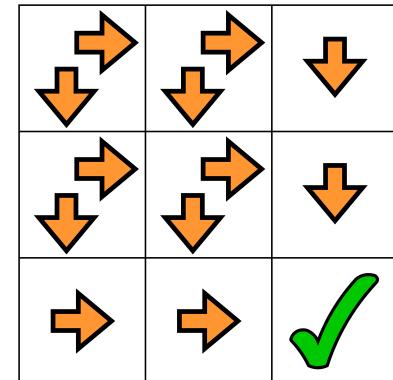
*Política de comportamiento*



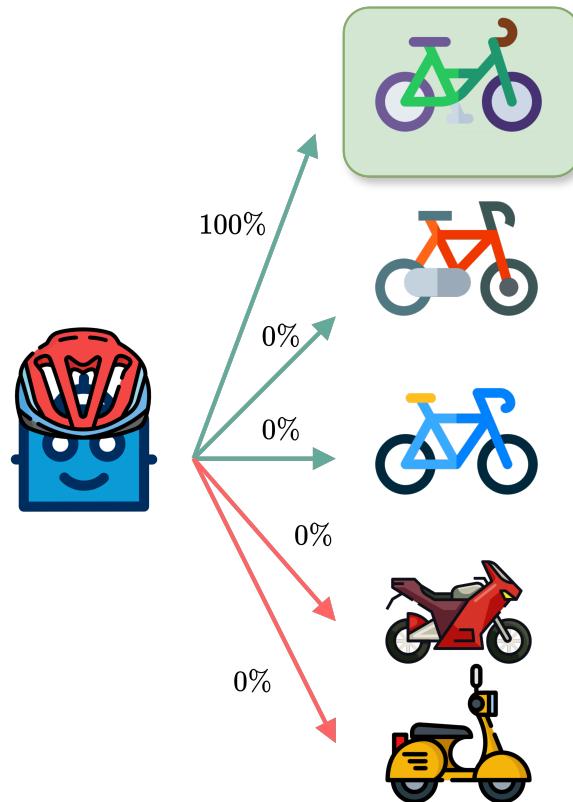
*Política objetivo determinista*



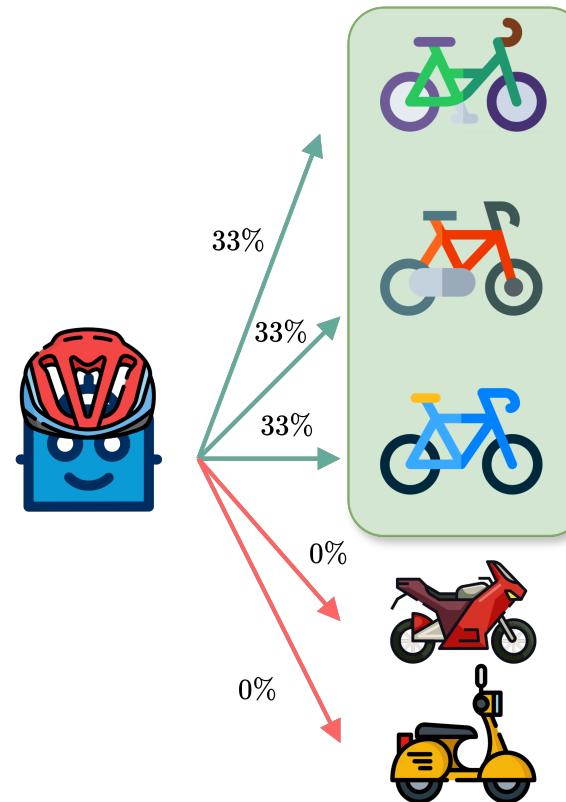
*Política objetivo estocástica*



*Política objetivo determinista*



*Política objetivo estocástica*

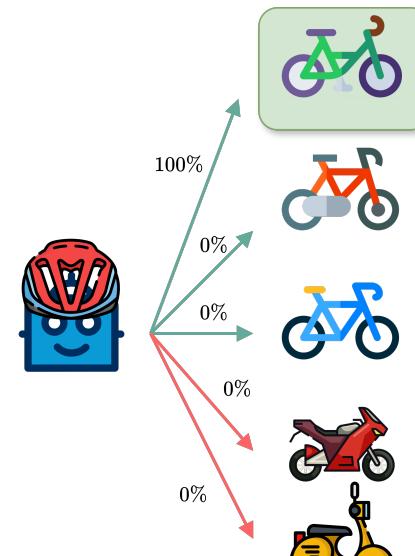


## ¿Para qué una política objetivo $\pi$ estocástica?

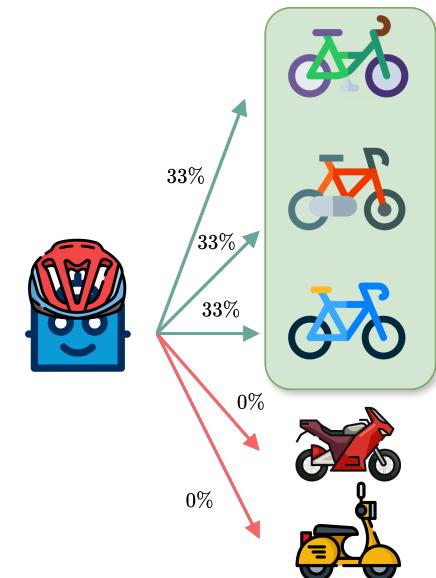
En este ejemplo, si  $\pi$  es **determinista** sólo contemplará una acción por estado, incluso si hay varias acciones óptimas.

Pero si es **estocástica**, tenemos una política que permite una **mayor variedad de acciones óptimas** para un mismo estado.

Política objetivo **determinista**



Política objetivo **estocástica**



Sea como sea la política objetivo  $\pi$ , estamos tratando de obtener  $v_\pi(s)$  **a partir de experiencia generada por una política  $b$  diferente**. Es decir, lo que tenemos es:

$$v_b(s) = \mathbb{E}[G_t \mid S_t = s]$$

El **problema** es que las distribuciones de estados y acciones bajo  $b$  y  $\pi$  pueden ser diferentes, dando lugar a un **sesgo**.

Si un subconjunto de estados es más frecuente siguiendo  $b$ , entonces  $\pi$  únicamente contará con información sobre esos estados, ignorando el resto.

 Una forma de solucionar esto es emplear **importance sampling**.

# *Importance sampling*

## *Importance sampling*

El **muestreo por importancia**, o ***importance sampling*** es una técnica empleada en estadística para estimar el valor esperado de una distribución en base a ejemplos muestreados de una distribución diferente.

Veamos de forma intuitiva en qué consiste...

Quiero obtener:

$$\mathbb{E}[g(X)]$$

Genero muestras aleatorias de una distribución:

$$X_1, X_2, \dots, X_n \sim \mathcal{D}$$

Aproximo el valor esperado con Monte Carlo:

$$\mathbb{E}[g(X)] \simeq \frac{1}{n} \sum_{i=1}^n g(X_i)$$

Valores de  $g(X)$  podrían ser **poco probables** pero con una **contribución muy significativa** sobre  $\mathbb{E}[g(X)]$ .

Si no se muestran durante la estimación Monte Carlo,  $\mathbb{E}[g(X)]$  se estimará mal.

El **muestreo por importancia** consiste en emplear una distribución «modificada» donde los valores más importantes (los que más afectan a la estimación de  $\mathbb{E}[g(X)]$ ) se vuelven **más probables**.

- Aseguramos así que sean muestrados y formen parte de la estimación Monte Carlo de  $\mathbb{E}[g(X)]$ .
- Para paliar el efecto de este aumento de probabilidad, los valores se escalan dándoles un menor peso al ser muestrados.

Muestreamos valores que provienen de la distribución *modificada*:

$$Y_1, Y_2, \dots, Y_n \sim \mathcal{D}'$$

Y aproximamos de la siguiente manera:

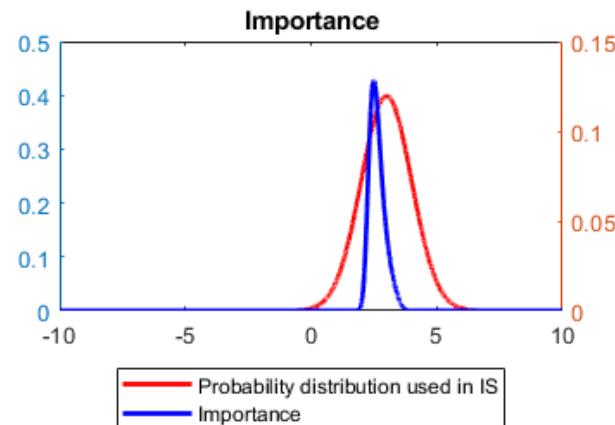
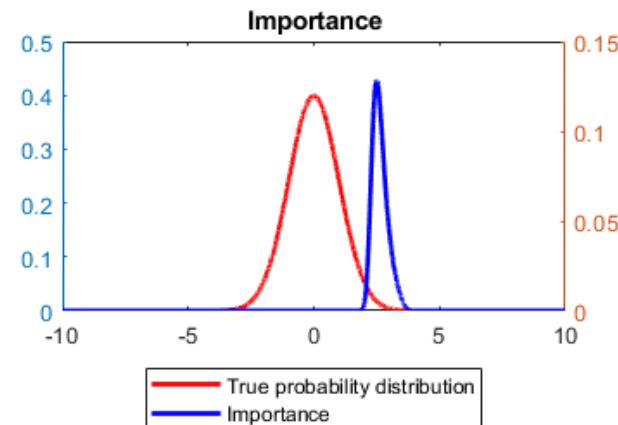
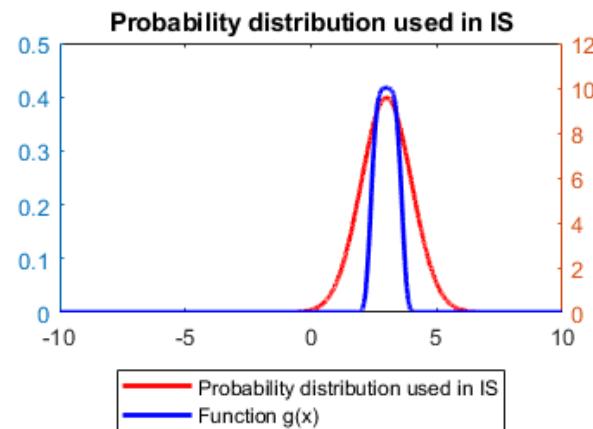
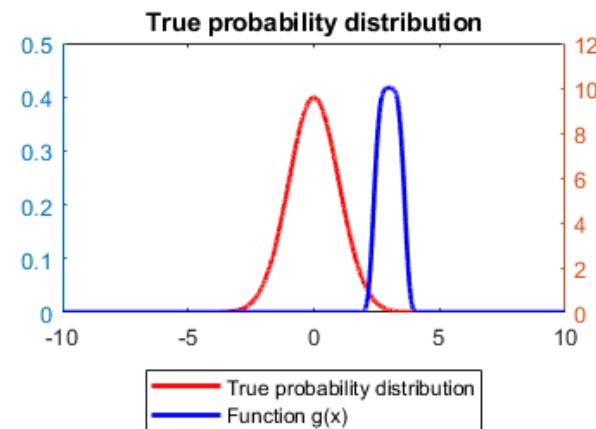
$$\mathbb{E}[g(Y)] = \frac{1}{n} \sum_{i=1}^n \frac{p_{\mathcal{D}}(Y_i)}{p_{\mathcal{D}'}(Y_i)} g(Y_i)$$

Siendo:

$$\mathbb{E}[g(Y)] \simeq \mathbb{E}[g(X)]$$

# Importance sampling

100/131



*Intuitivamente:*

- **Ampliamos** (*scale up*) los eventos que son raros en  $\mathcal{D}'$  pero comunes en  $\mathcal{D}$ .
- **Reducimos** (*scale down*) los eventos que son comunes en  $\mathcal{D}'$  pero raros en  $\mathcal{D}$ .

$$\mathbb{E}[g(Y)] = \frac{1}{n} \sum_{i=1}^n \frac{p_{\mathcal{D}}(Y_i)}{p_{\mathcal{D}'}(Y_i)} g(Y_i)$$

$$\mathbb{E}[g(Y)] = \frac{1}{n} \sum_{i=1}^n \frac{p_{\mathcal{D}}(Y_i)}{p_{\mathcal{D}'}(Y_i)} g(Y_i)$$

$\frac{p_{\mathcal{D}}(Y_i)}{p_{\mathcal{D}'}(Y_i)} = 1$	Misma probabilidad en $\mathcal{D}$ y $\mathcal{D}'$ . La aportación no varía.
$\frac{p_{\mathcal{D}}(Y_i)}{p_{\mathcal{D}'}(Y_i)} > 1$	$Y_i$ es más probable en la distribución original $\mathcal{D}$ , por lo que su peso es mayor.
$\frac{p_{\mathcal{D}}(Y_i)}{p_{\mathcal{D}'}(Y_i)} < 1$	$Y_i$ es más probable en la distribución modificada $\mathcal{D}'$ , por lo que su peso es menor.
$\frac{p_{\mathcal{D}}(Y_i)}{p_{\mathcal{D}'}(Y_i)} = 0$	$Y_i$ no aporta nada, porque no puede obtenerse en la distribución original.
$p_{\mathcal{D}'}(Y_i) = 0$	No se cumple el principio de cobertura.

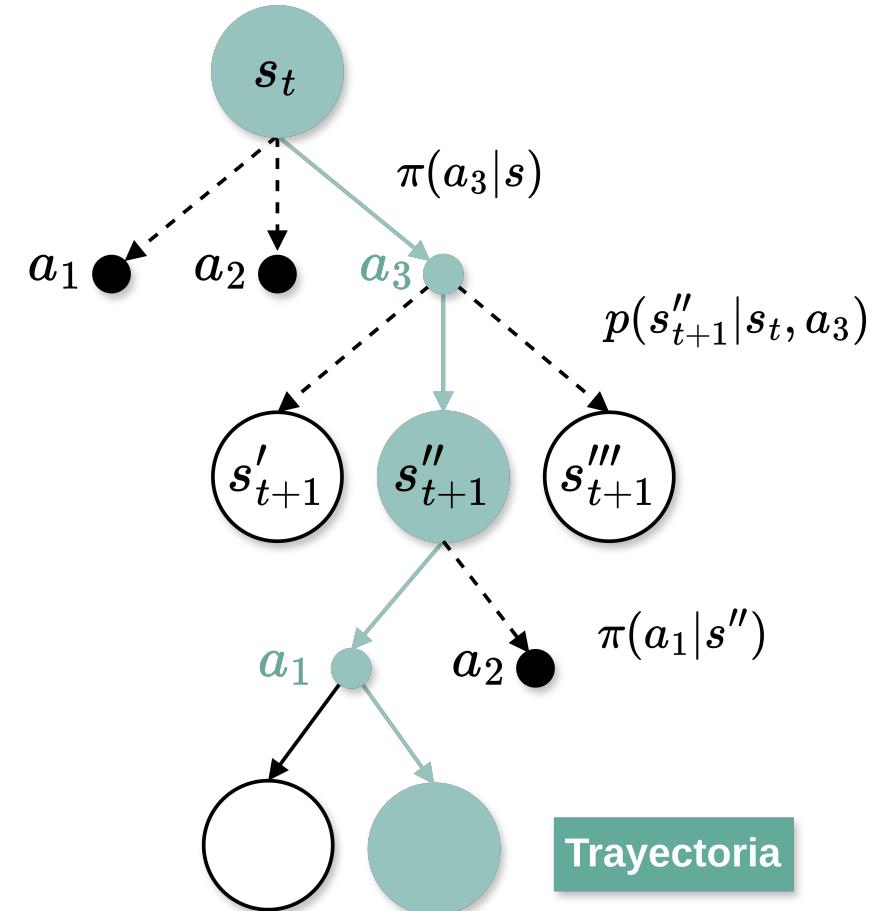
¿Cómo se aplica en  
predicción *off-policy*?

Recordemos el concepto de **trayectoria**:

$$\tau = \{S_t, A_t, S_{t+1}, A_{t+1}, \dots, S_T\}$$

La **probabilidad de realizar una trayectoria  $\tau$  bajo una política  $\pi$**  es:

$$\Pr(\tau_\pi) = \prod_{k=t}^{T-1} \pi(A_k | S_k) p(S_{k+1} | S_k, A_k)$$



Utilizamos  $\rho$  para **ponderar las recompensas finales** obtenidas en cada trayectoria.

$\rho = 1$	El valor de $G$ obtenido se mantiene, ya que es igual de probable con $b$ y $\pi$ .
$\rho > 1$	El valor de $G$ obtenido por $b$ tiene mayor peso, porque es una trayectoria probable con $\pi$ .
$\rho < 1$	El valor de $G$ obtenido por $b$ se reduce porque es una trayectoria poco probable con $\pi$ .
$\rho = 0$	El valor de $G$ se anula porque no es una trayectoria que podamos obtener con $\pi$ .

Finalmente, lo que tenemos es:

$$\mathbb{E}[\underbrace{\rho_{t:T-1} G_t}_{\text{Retorno obtenido}} \mid S_t = s] = \underbrace{v_\pi(s)}_{\text{Función de valor correspondiente a la política objetivo } \pi}$$

tras una serie  
de trayectorias  
siguiendo  $b$

a la política  
objetivo  $\pi$

Ponderamos la recompensa acumulada obtenida  
tras una trayectoria en base a su probabilidad.

# Ejemplo

La política  $b$  interactúa con el entorno durante un episodio y obtiene un retorno  $G = 10$ .

Si la trayectoria seguida por  $b$  es 3 veces **menos** probable que ocurra empleando  $\pi$ , aumentamos  $G \times 3$ :

$$\pi(a|s) > b(a|s) \text{ (ej. } \times 3) \longrightarrow G = 10 \times 3 = 30$$

Si, por el contrario, es **más** probable que ocurra en  $b$  que en  $\pi$ , reducimos el valor de  $G$ :

$$\pi(a|s) < b(a|s) \text{ (ej. } \times 0.25) \rightarrow G = 10 \times 0.25 = 2.5$$

# Predicción Monte Carlo de $v_\pi$ con *importance sampling*

Dado un conjunto (*batch*) de episodios observados a partir de una política  $b$ , procedemos a estimar  $v_\pi$ . Tenemos dos opciones:

- *Importance sampling ordinario*:

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}$$

- *Importance sampling ponderado*:

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$$

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}$$

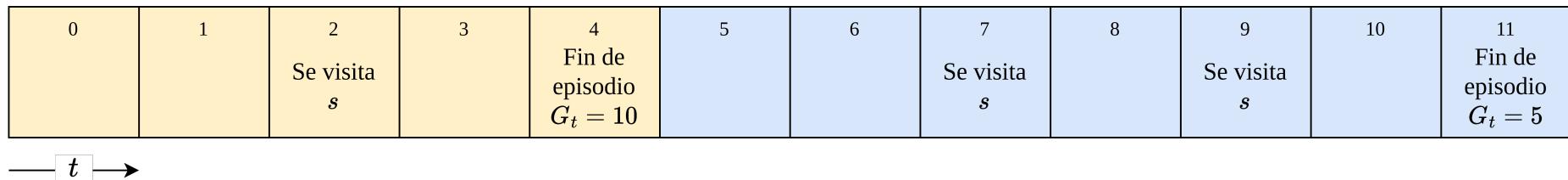
- $t$  = *time step* donde se visita  $s$ .
- $T(t)$  = siguiente terminación de episodio tras  $t$ .
- Retorno obtenido al final de la trayectoria.
- *Time steps* en los que se ha visitado  $s$ .

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}$$

- **First-visit:**  $\mathcal{T}(s)$  sólo considera los *time steps* de la primera visita a  $s$  en cada episodio.
- **Every-visit:**  $\mathcal{T}(s)$  incluye todas las visitas a  $s$ .

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}$$

Batch de 2 episodios:



**First-visit:**  $\mathcal{T}(s) = \{1, 7\}$

$$V(s) = \frac{\rho_{1:3} \cdot 10 + \rho_{7:10} \cdot 5}{2}$$

**Every-visit:**  $\mathcal{T}(s) = \{1, 7, 9\}$

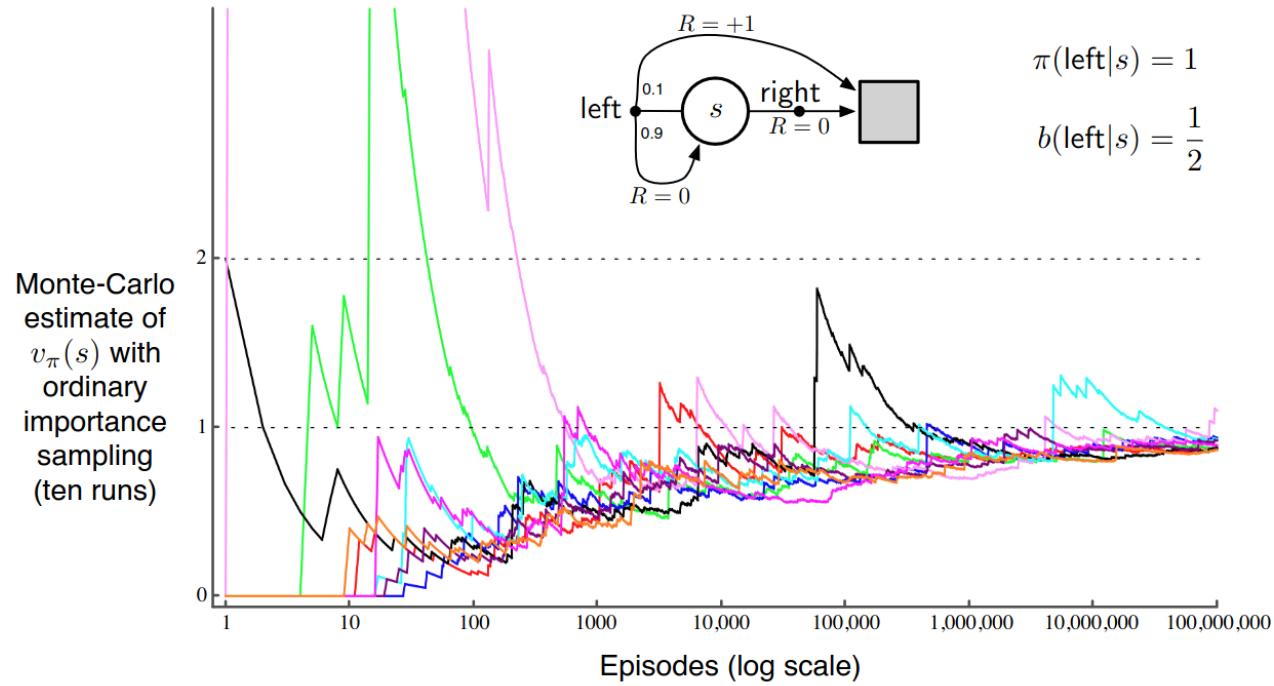
$$V(s) = \frac{\rho_{1:3} \cdot 10 + \rho_{7:10} \cdot 5 + \rho_{9:10} \cdot 5}{3}$$

El ***importance sampling* ordinario** no presenta sesgos (*bias*) en favor de la política de comportamiento  $b$ , pero puede ser demasiado extremo.

Por ejemplo, si una trayectoria es  $\times 10$  veces más probable bajo  $\pi$  que bajo  $b$ , la estimación será diez veces  $G$ , que se aleja bastante del realmente observado.

Se plantea como alternativa el ***importance sampling* ponderado**:

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$$



**Figure 5.4:** Ordinary importance sampling produces surprisingly unstable estimates on the one-state MDP shown inset (Example 5.5). The correct estimate here is 1 ( $\gamma = 1$ ), and, even though this is the expected value of a sample return (after importance sampling), the variance of the samples is infinite, and the estimates do not converge to this value. These results are for off-policy first-visit MC.

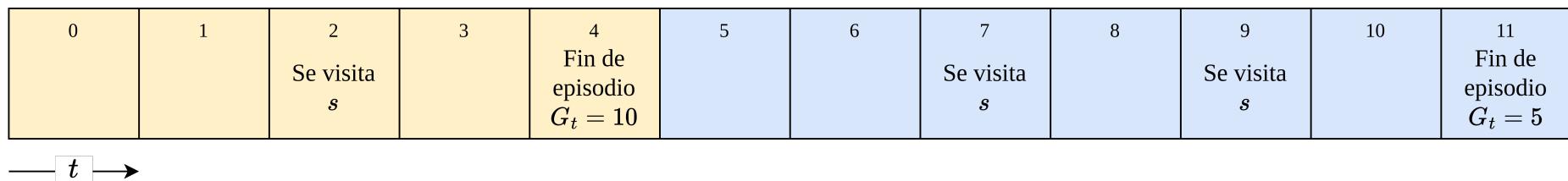
$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$$

- Presenta **mayor sesgo**:
  - Si sólo se visita un estado una vez,  $v_\pi(s) = v_b(s)$ , el *importance sampling ratio* se cancela y hay un sesgo hacia la política de comportamiento.
  - Sin embargo, **la varianza es menor** (actualizaciones menos extremas).
  - Suele ser una mejor opción

# Ejemplo

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$$

Batch de 2 episodios:

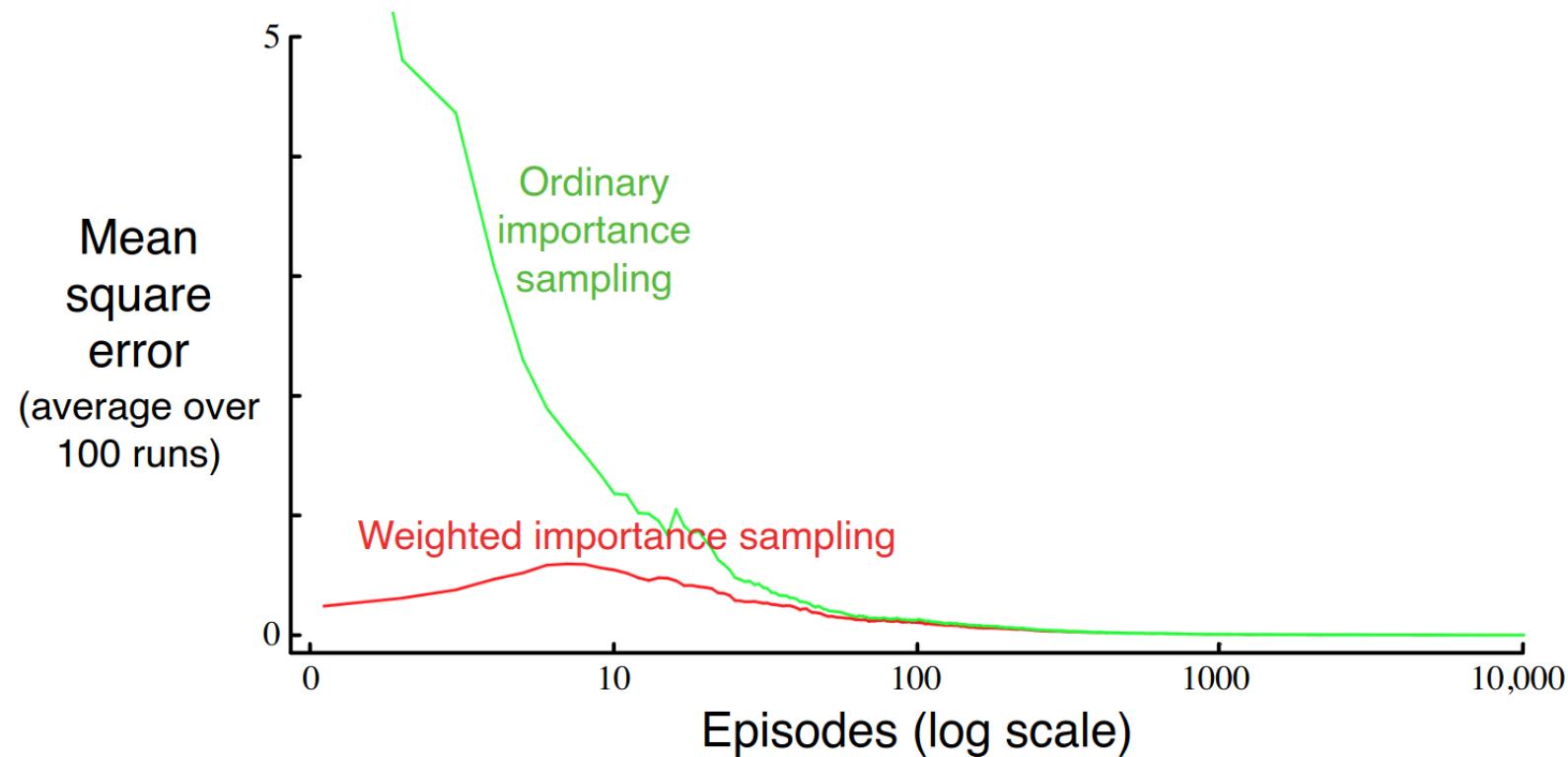


**First-visit:**  $\mathcal{T}(s) = \{1, 7\}$

**Every-visit:**  $\mathcal{T}(s) = \{1, 7, 9\}$

$$V(s) = \frac{\rho_{1:3} \cdot 10 + \rho_{7:10} \cdot 5}{\rho_{1:3} + \rho_{7:10}}$$

$$V(s) = \frac{\rho_{1:3} \cdot 10 + \rho_{7:10} \cdot 5 + \rho_{9:10} \cdot 5}{\rho_{1:3} + \rho_{7:10} + \rho_{9:10}}$$



**Figure 5.3:** Weighted importance sampling produces lower error estimates of the value of a single blackjack state from off-policy episodes. ■

# Predicción Monte Carlo incremental

La predicción Monte Carlo puede realizarse de forma **incremental**.

- Supongamos una secuencia de retornos:  $G_1, G_2, \dots, G_{n-1}$  obtenidos partiendo de un mismo estado.
- Cada retorno tiene una ponderación  $W_i = \rho_{t_i:T(t_i)-1}$
- Empleando *importance sampling* ponderado tenemos:

$$V_n = \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}, \quad n \geq 2$$

$$V_n = \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}, \quad n \geq 2$$

Buscamos **actualizar incrementalmente**  $V_n$  cada vez que se obtiene un nuevo retorno  $G_n$ .

- Para cada estado consideramos  $C_n$ , que es la suma acumulada de los pesos de los  $n$  primeros retornos:

$$C_{n+1} = C_n + W_{n+1}$$

- El cálculo de  $W_{n+1}$  es recursivo:

$$W_1 \leftarrow \rho_{T-1}$$

$$W_2 \leftarrow \rho_{T-1}\rho_{T-2}$$

$$W_3 \leftarrow \rho_{T-1}\rho_{T-2}\rho_{T-3}$$

...

$$W_{n+1} \leftarrow W_n \rho_n$$

Así, la **regla de actualización** empleada es:

$$V_{n+1} = V_n + \frac{W_n}{C_n}[G_n - V_n], \quad n \geq 1$$

$V$  puede ser el valor de un estado o de un par acción-estado.

- ✓ Si bien esta implementación se corresponde con el algoritmo de **predicción off-policy con importance sampling ponderado**, también se aplica al caso **on-policy** si  $\pi = b$  ( $W$  es siempre = 1).

Off-policy MC prediction (policy evaluation) for estimating  $Q \approx q_\pi$

Input: an arbitrary target policy  $\pi$

Initialize, for all  $s \in \mathcal{S}$ ,  $a \in \mathcal{A}(s)$ :

$$Q(s, a) \in \mathbb{R} \text{ (arbitrarily)}$$

$$C(s, a) \leftarrow 0$$

Loop forever (for each episode):

$b \leftarrow$  any policy with coverage of  $\pi$

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

$$G \leftarrow 0$$

$$W \leftarrow 1$$

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

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

$$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$$

$$W \leftarrow W \frac{\pi(A_t | S_t)}{b(A_t | S_t)}$$

# Control Monte Carlo *off-policy*

## Off-policy MC control, for estimating $\pi \approx \pi_*$

Initialize, for all  $s \in \mathcal{S}$ ,  $a \in \mathcal{A}(s)$ :

$$Q(s, a) \in \mathbb{R} \text{ (arbitrarily)}$$

$$C(s, a) \leftarrow 0$$

$$\pi(s) \leftarrow \operatorname{argmax}_a Q(s, a) \quad (\text{with ties broken consistently})$$

Loop forever (for each episode):

$$b \leftarrow \text{any soft policy}$$

Generate an episode using  $b$ :  $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

$$G \leftarrow 0$$

$$W \leftarrow 1$$

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

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

$$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$$

$$\pi(S_t) \leftarrow \operatorname{argmax}_a Q(S_t, a) \quad (\text{with ties broken consistently})$$

If  $A_t \neq \pi(S_t)$  then exit inner Loop (proceed to next episode)

$$W \leftarrow W \frac{1}{b(A_t | S_t)}$$

**En resumen...**

- Los métodos **Monte Carlo** aprenden funciones de valor y políticas óptimas a partir de experiencia procedente de **episodios muestrados / aleatorios**.
- Las políticas óptimas se aprenden directamente, **sin requerir un modelo del entorno** ni emplear *bootstrapping* (estimaciones a partir de estimaciones).
- El esquema general de **GPI** también se aplica a estos métodos (evaluación + mejora de la política).
- La aproximación de las funciones de valor se realiza en base al **retorno promedio** desde cada estado.

- Para garantizar la exploración, empleamos técnicas como **inicios de exploración**, aunque presenta algunas limitaciones.
- Los métodos **on-policy** permiten asegurar la exploración y alcanzar una política muy cercana a la óptima.
- Los métodos **off-policy** emplean dos políticas: una para explorar, y otra para actuar.
- El uso de dos políticas requiere aplicar **importance sampling** ordinario o ponderado para que las estimaciones no estén sesgadas.

# **APRENDIZAJE POR REFUERZO**

Métodos basados en muestreo: Monte Carlo

---

Antonio Manjavacas

[manjavacas@ugr.es](mailto:manjavacas@ugr.es)