



XXV

**CONGRESO INTERNACIONAL DE
INFORMÁTICA Y SISTEMAS**



Taller: Introducción práctica al aprendizaje reforzado con Gym en Python

Dr. Miguel Solís
Universidad Andrés Bello, Chile

Áreas en Robótica

- Robótica móvil
- Robótica bioinspirada
- Robótica cognitiva
- Robótica evolutiva
- Interacción humano-robot
- Microrrobótica
- Robótica tele-operada
- Robótica de enjambre



Percepción y Acción

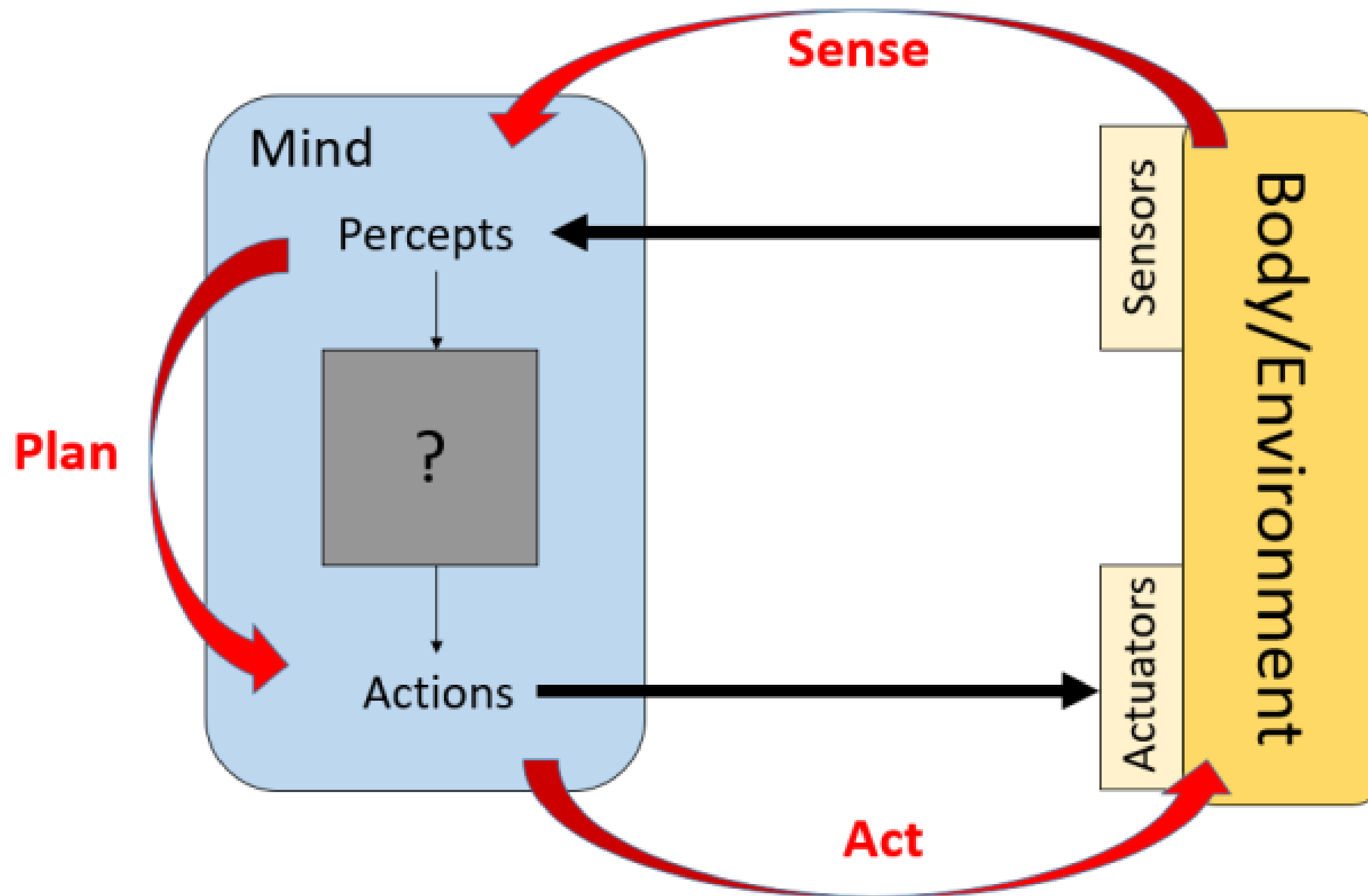


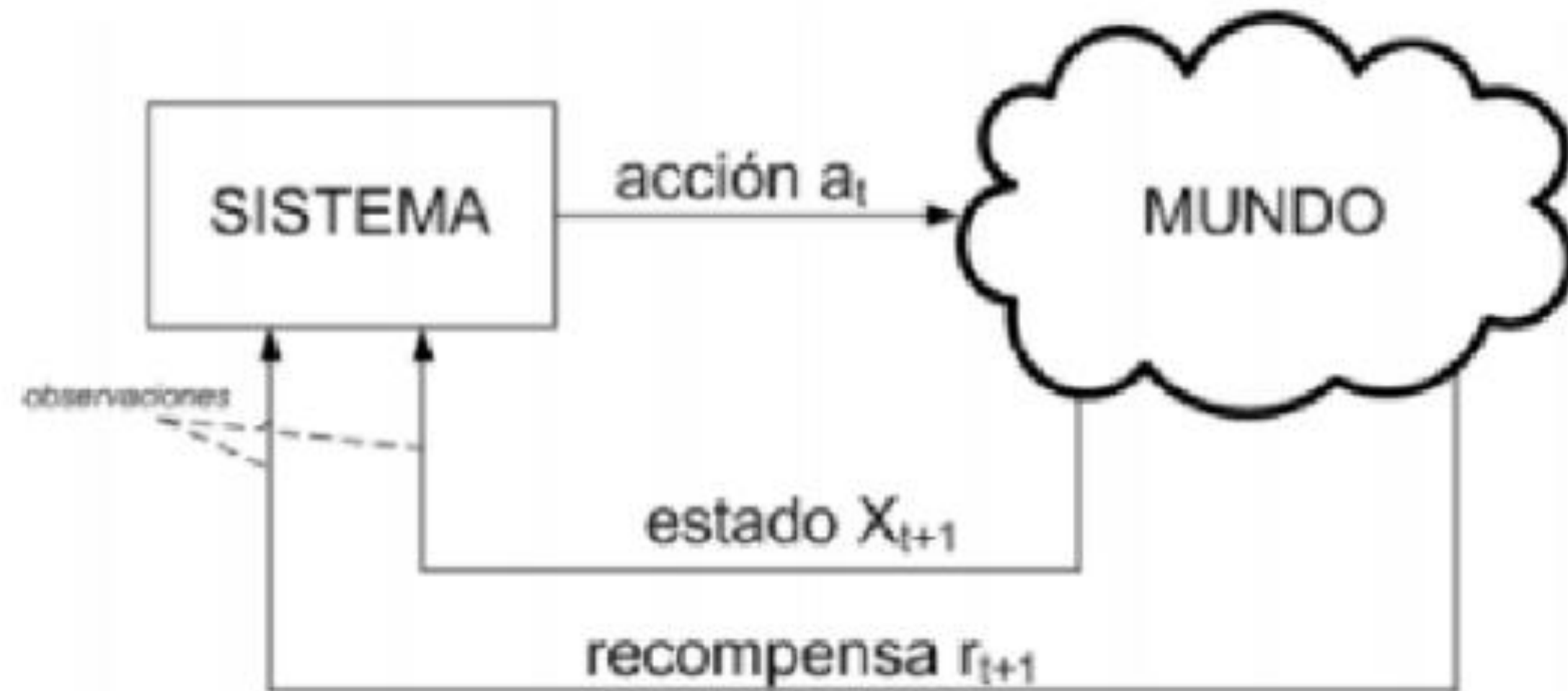
Foto extraída de "Robotic Systems" de Kris Hauser, University of Illinois at Urbana-Champaign.

Toma de decisiones

- de manera reactiva: reaccionando ante ciertos estímulos del entorno. La manera más simple de su implementación es a través de una máquina de estados finitos. (*Si ocurre A estando en X, entonces ejecuto B y quedaré en Y*).
- de manera inteligente: existen técnicas de inteligencia artificial que permiten generar cierto comportamiento en base a heurísticas.
- con aprendizaje incremental: existen técnicas computacionales para dotar de aprendizaje automático a cierto agente, que puede ser un dispositivo con actuaciones físicas.



Aprendizaje reforzado



- Aprende de interacciones con el entorno.
- Aplicaciones:
 - Problemas de decisión secuencial.
 - Sistemas adaptivos.



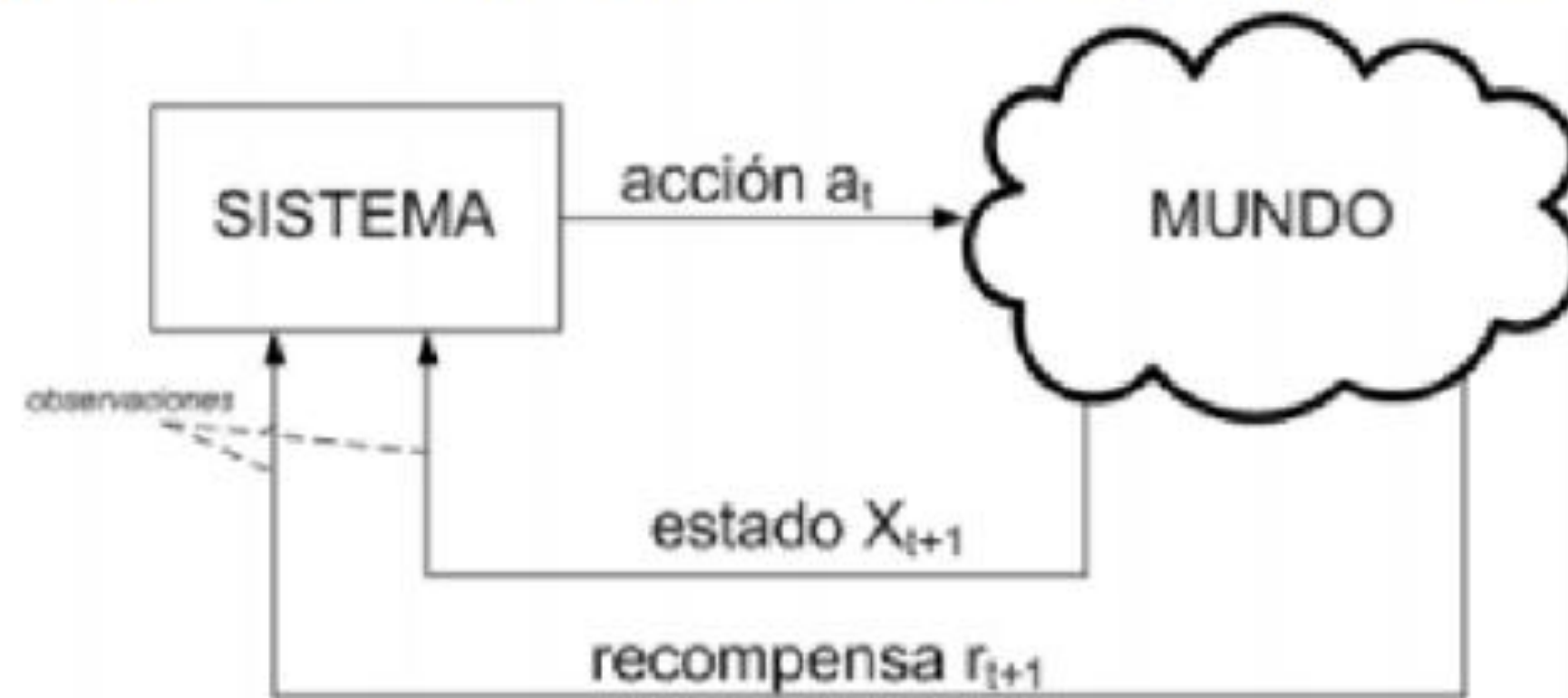
Aprendizaje reforzado y conductismo

- Edward L. Thorndike - Animal Intelligence: An experimental study of the associate processes in animals (1898).
- El animal modifica su conducta según interacción de prueba y error con el medio ambiente.



Aprendizaje reforzado y conductismo

El agente interactúa con el entorno a través de percepciones y acciones.



- Recibe como entrada (percibe), el estado actual del entorno, s .
- Luego, genera una acción (ejecuta) a como salida.
- Recibe una señal de refuerzo r (recompensa).



Proceso de Markov

- Un problema de aprendizaje reforzado se formula de manera formal mediante un MDP.
- MDP: Markov Decision Process.

Un estado s_k se dice que obedece a un Proceso de Markov (de primer orden) si y sólo si:

$$Pr\{s_{k+1}|s_k\} = Pr\{s_{k+1}|s_1, \dots, s_k\}.$$



Elementos en un problema de aprendizaje reforzado

Un problema de aprendizaje reforzado, formulado como un MDP está compuesto por la tupla (S, A, T, R) donde

- S : conjunto de estados
- A : conjunto de acciones
- $T : S \times A \times S \rightarrow [0, 1]$ (*función de transición, desconocida*)
- $R : S \times A \times S \rightarrow \mathbb{R}$ (*función de recompensas*)
- $\pi : S \rightarrow A$ (*política*)



Elementos en un problema de aprendizaje reforzado

- Estado en instante k :

$$s_k \in \mathcal{S}$$

- Acción ejecutada en instante k :

$$a_k \in \mathcal{A}$$

- Función de transición de estado:

$$s_{k+1} = T(s_k, a_k)$$



Recompensas

$$r_k = R(s_k, a_k, s_{k+1})$$

- $r_k > 0$
- $r_k = 0$
- $r_k < 0$

- Política π :

$$\pi : \mathcal{S} \rightarrow \mathcal{A}$$

- Una política es óptima si maximiza la recompensa a largo plazo.



Retorno

$$\begin{aligned} V^{\pi}(s_k) &= r_k + \gamma r_{k+1} + \gamma^2 r_{k+2} + \dots \\ &= \sum_{i=0}^{\infty} \gamma^i r_{k+i}. \end{aligned}$$

Restricciones:

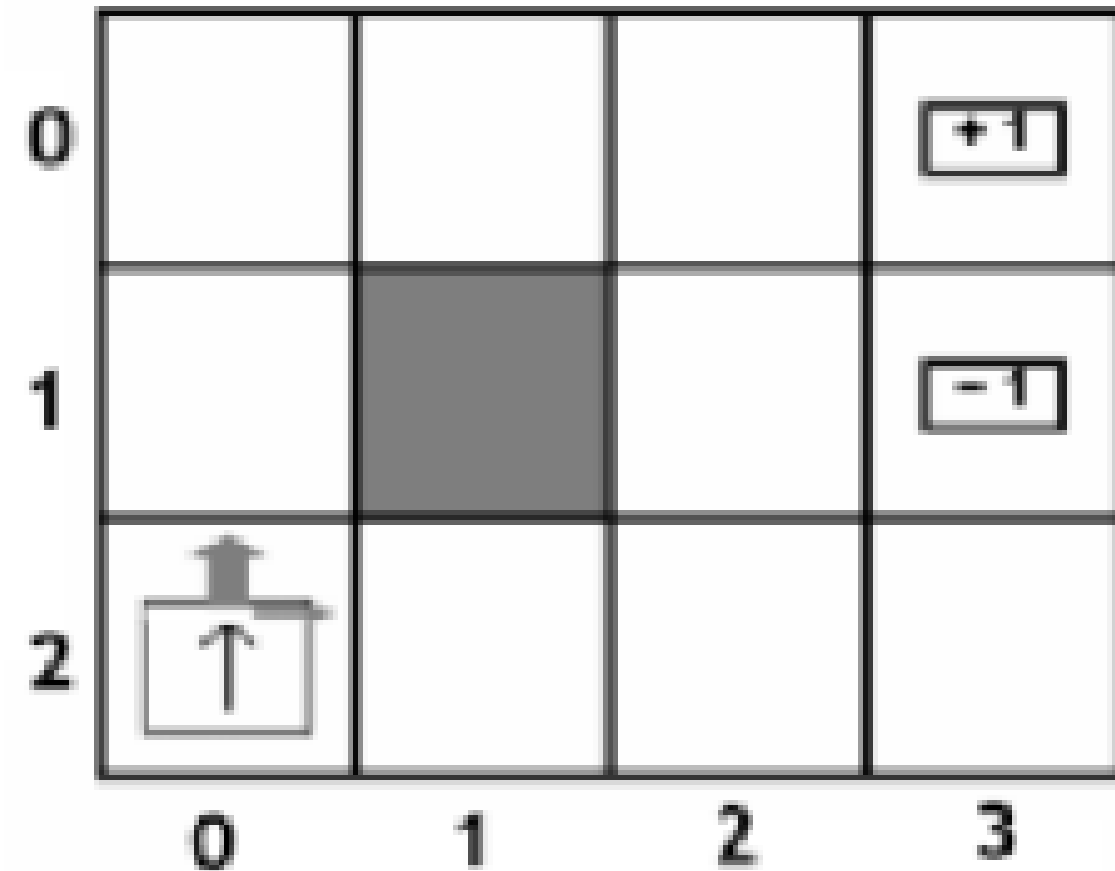
- $0 \leq \gamma < 1$.
- r_k acotado.

Una política π^* es óptima si la función de valor obtenida para esa política es óptima:

$$V^*(s) = V^{\pi^*}(s) \geq V^{\pi}(s) \quad \forall s, \pi$$



Ejemplo: Grid World



- Estados: ubicación en la grilla.
- Acciones: arriba, izquierda, derecha, abajo.
- Recompensa: +1, -1, -0.1.



Algoritmos clásicos

Provenientes de Programación Dinámica:

- Value Iteration
- Policy Iteration
- Temporal Difference
 - Q-Learning
 - SARSA






Value Iteration

Consiste en realizar k iteraciones, hasta que $V_k(s) - V_{k-1}(s)$ es suficientemente pequeño, con actualización según:



$$V_k(s) = \max_a \sum_{s'} Pr\{s', s, a\} \cdot (R(s', s, a) + \gamma V_{k-1}(s'))$$





Ejemplo

| | | | | |
|---|---|---|---|---|
| 0 | | | |  |
| 1 | | | |  |
| 2 |  | | | |
| | 0 | 1 | 2 | 3 |



- Después de 1 iteración:

| | | | | |
|---|-------|------|-------|---|
| 0 | -0.1 | -0.1 | 1 |  |
| 1 | -0.19 | | -0.1 |  |
| 2 | -0.1 | -0.1 | -0.19 | -0.1 |
| | 0 | 1 | 2 | 3 |

- En convergencia:

| | | | | |
|---|------|------|------|---|
| 0 | 0.62 | 0.8 | 1 |  |
| 1 | 0.46 | | 0.8 |  |
| 2 | 0.31 | 0.46 | 0.62 | 0.46 |
| | 0 | 1 | 2 | 3 |

- Después de 2 iteraciones:

| | | | | |
|---|-------|-------|------|---|
| 0 | 0.62 | 0.8 | 1 |  |
| 1 | 0.46 | | 0.8 |  |
| 2 | -0.27 | -0.19 | 0.62 | -0.27 |
| | 0 | 1 | 2 | 3 |



Valor Q

- Así como $V(s)$ corresponde a la función que valoriza el estado, $Q(s, a)$ corresponde a la función que valoriza el tomar cierta acción en ese estado.
- Para una política óptima π^* , se cumple

$$Q^{\pi^*}(s, a) \geq Q^{\pi}(s, a) \quad \forall s, a, \pi$$



Q-learning

- Consiste en iterar sobre cada par (estado, acción), para α y γ fijos.
- Regla de actualización:

$$Q(s_k, a_k) \leftarrow (1 - \alpha)Q(s_k, a_k) + \alpha \left(r_{k+1} + \gamma \max_a Q(s_{k+1}, a) \right)$$

Suponiendo $\hat{Q} = Q^*$, entonces la acción óptima para cada estado se puede obtener maximizando:

$$\pi^*(s_k) = \arg \max_a Q^*(s_k, a_k)$$



Imagen extraída de "Introduction to Q-learning with OpenAI Gym", Gelana Tostaeva, Medium.



Ejemplo

- No se realizan iteraciones sobre todo el espacio de estados, sólo cuando el estado se visita.
- Este ejemplo es repetitivo (cuando se llega a un estado final, vuelve al inicio).
- **atención** con los óptimos locales.

| | | | | |
|---|------|------|---|--------|
| 0 | der. | izq. | | meta |
| 1 | arr. | | | prohib |
| 2 | arr. | izq. | | |
| | 0 | 1 | 2 | 3 |



Estancamiento en óptimo local:

Exploración/Explotación

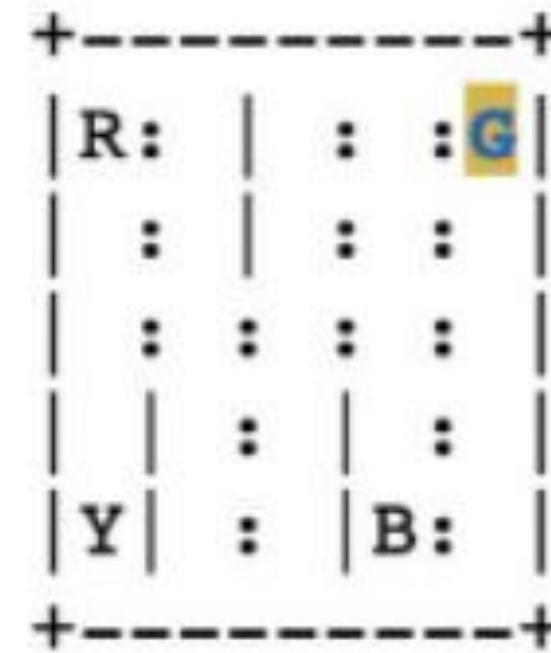
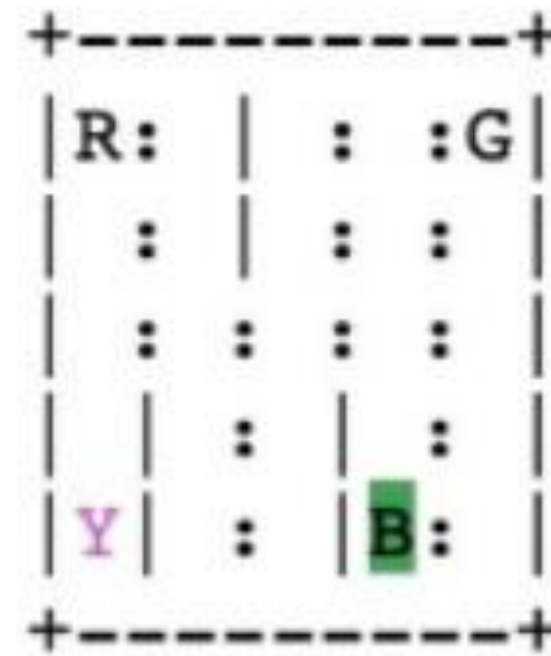
| | | | | |
|---|------|------|---|--------|
| 0 | der. | izq. | | meta |
| 1 | arr. | | | prohib |
| 2 | arr. | izq. | | |
| | 0 | 1 | 2 | 3 |

Incorporando exploración

| | | | | |
|---|------|------|------|--------|
| 0 | der. | der. | der. | meta |
| 1 | arr. | | | prohib |
| 2 | arr. | izq. | | |
| | 0 | 1 | 2 | 3 |



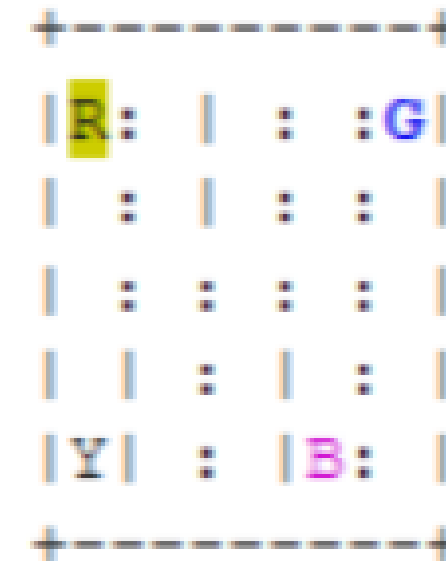
Ejemplo a programar



- R, G, Y, B: ubicaciones
- rectángulo amarillo: taxi vacío
- rectángulo verde: taxi ocupado
- azul: ubicación de pasajero
- morado: ubicación de destino



Acciones



- 0: mover al sur
- 1: mover al norte
- 2: mover al este (derecha)
- 3: mover al oeste (izquierda)
- 4: recoger pasajero
- 5: dejar pasajero



Espacio de estados y acciones

- Acciones: 6
- Estados: 500 (ubicacion_pasajero, ubicacion_taxi, destino)
 - 4 posibles destinos
 - ubicacion_pasajero:
 - 4 ubicaciones para tomar el taxi
 - o pasajero se encuentra en misma ubicación de taxi
 - ubicacion_taxi:
 - 25 ubicaciones posibles según el mapa
- Verificación en Gym:
 - env.action_space
 - env.observation_space

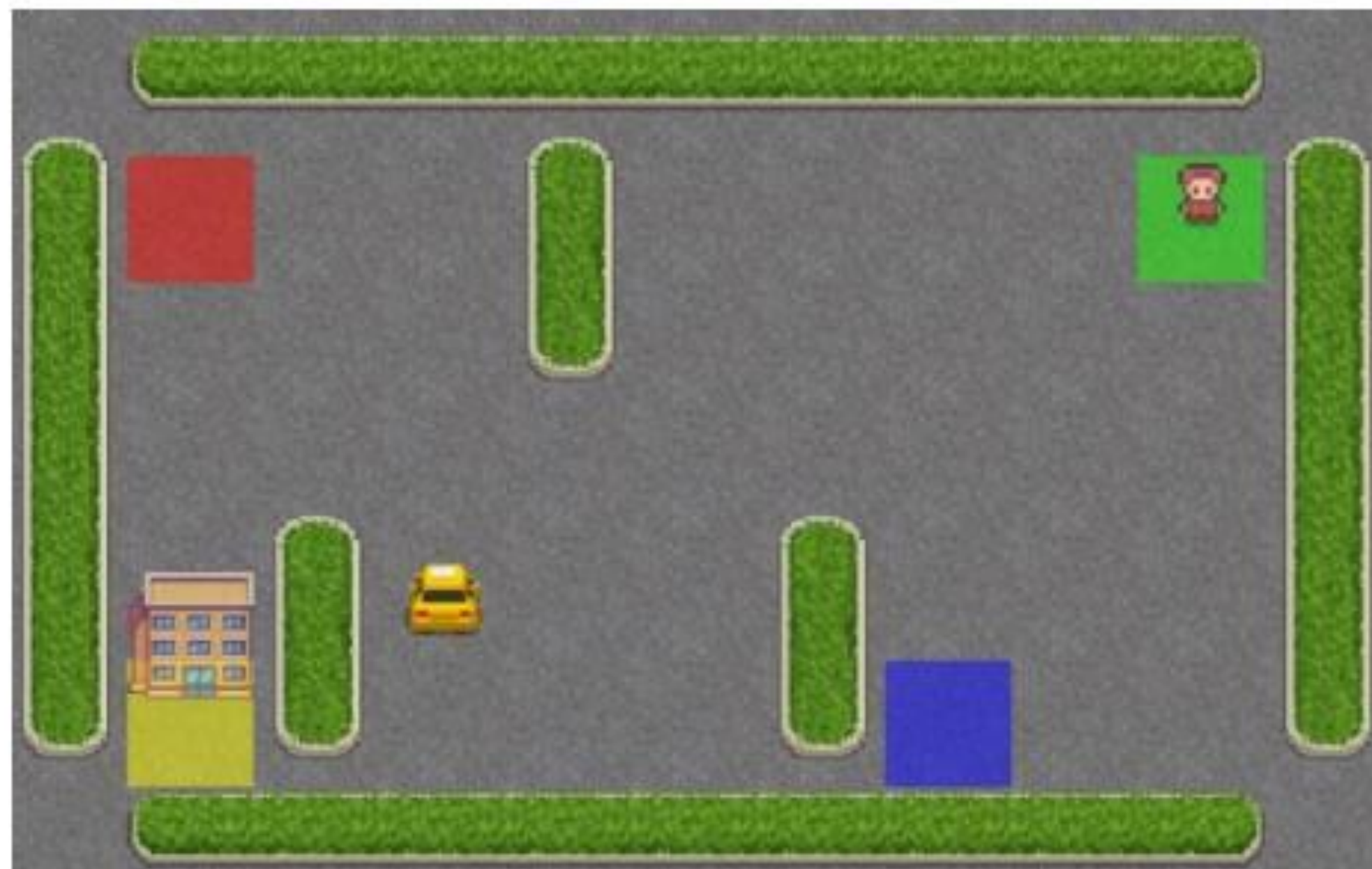


Recompensas

- dejar a pasajero en ubicación correcta: 20 puntos
- descuento de 1 punto cada vez que se mueve con pasajero y no llega a destino
- descuento de 10 puntos por dejar a pasajero en ubicación incorrecta/ilegal



Python Notebook

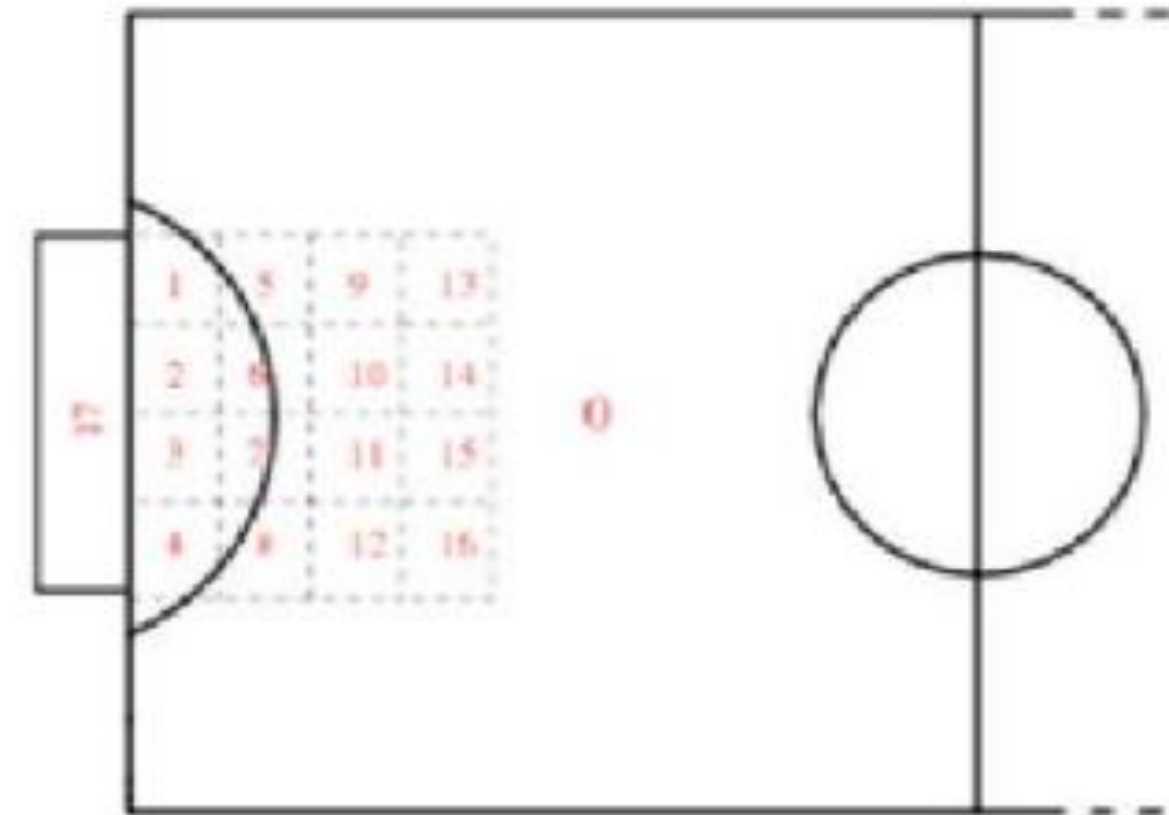


Abrir con visor de Python Notebook o
colab.research.google.com



Más ejemplos

- G.A. Ahumada, C.J. Nettle and M.A. Solis, 'Accelerating Q-learning through Kalman Filter Estimations applied in a RoboCup SSL Simulation', **Proceedings** of the 10th IEEE Latin American Robotics Symposium, 2013.



Más ejemplos

Generación de estrategia defensiva



estado compuesto por:

- ▶ $\text{dist}(K_i, \text{pelota}), \text{dist}(T_j, \text{pelota})$
- ▶ $\text{dist}(K_i, K_j)$
- ▶ $\text{dist}(K_i, T_j)$
- ▶ $\text{angle}(K_i, T_j)$

Ollino, F., Solis, M. A., & Allende, H. (2018). Batch reinforcement learning on a RoboCup Small Size League keepaway strategy learning problem. In 4th Congress on Robotics and Neuroscience, CRONe 2018. CEUR-WS.



Algunos desafíos

- diseño de recompensas.
- retardo en la ejecución de las acciones.
- **representación tabular.**



Algunos temas actuales

- affordances

(Objeto, Acción, Efecto)

Dado un objeto y cierta acción, ¿cuál es el efecto?

Dado un objeto y cierto efecto deseado, ¿cuál es la acción requerida?



- continual reinforcement learning (open-ended)





XXV

**CONGRESO INTERNACIONAL DE
INFORMÁTICA Y SISTEMAS**



Gracias por ver