Conceptos básicos de aprendizaje reforzado y sus aplicaciones en robótica

Dr.-Eng. Miguel Solis

13 de noviembre de 2018

Utilizando Jupyterhub en
https://python.innovacionyrobotica.com

Estructura

- Introducción

- Aprendizaje reforzado

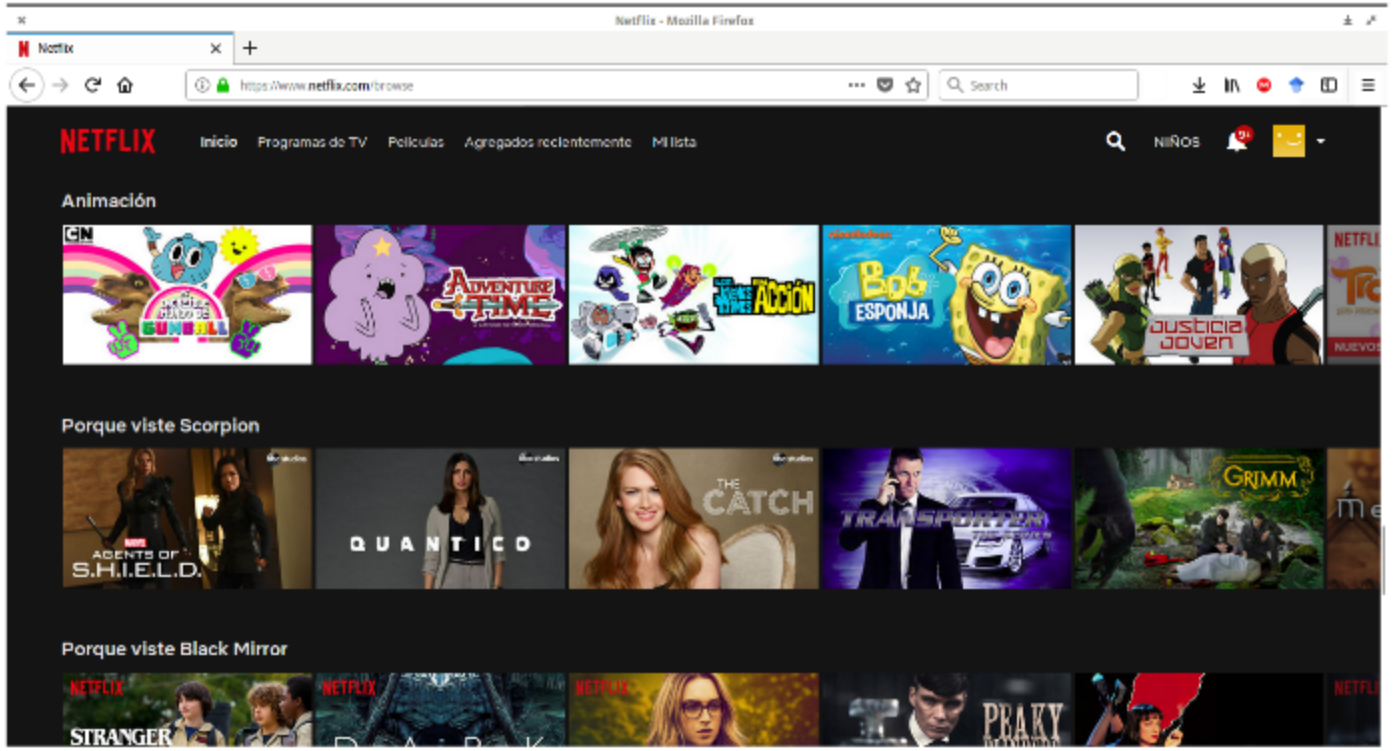
- Métodos

- Aplicaciones en robótica

(algunos) Tipos de aprendizaje

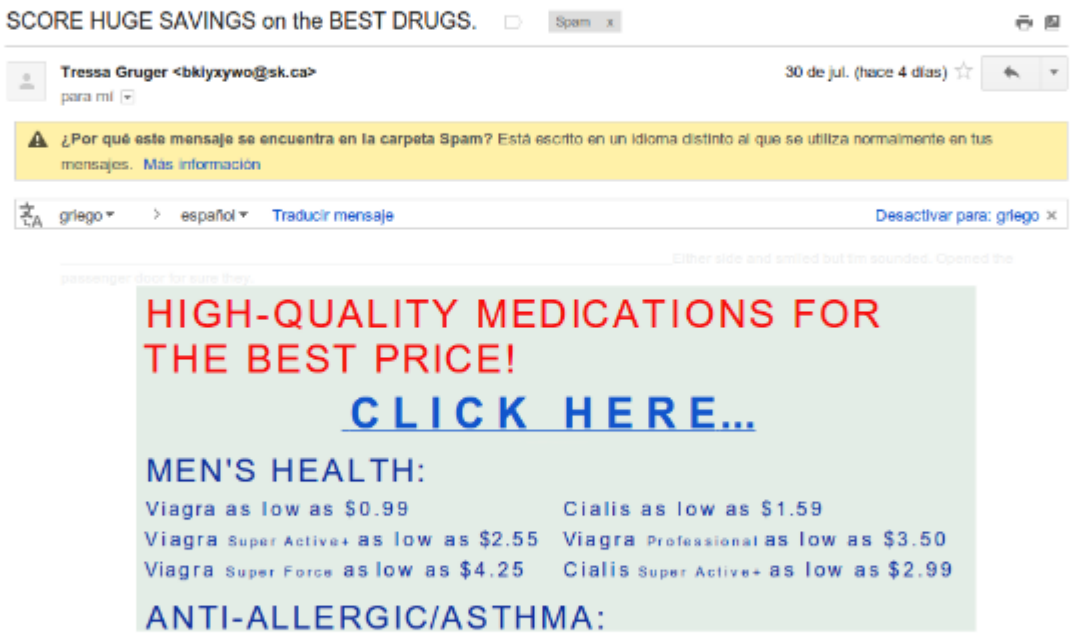
- Aprendizaje Supervisado: encontrar un mapeo de entrada a salida, supervisor provee valores correctos.
  - Regresión: la salida corresponde a un valor numérico.
  - Clasificación: la salida corresponde a una etiqueta de clase.
- Aplicaciones:
  - Reconocimiento de rostros, escritura, gestos
  - Diagnóstico médico
  - Bioinformática, Quimioinformática
  - ...

Recomendaciones



- Entrada: preferencias de otros usuarios con perfil similar
- Salida: predicción de la etiqueta de clase que me interesa

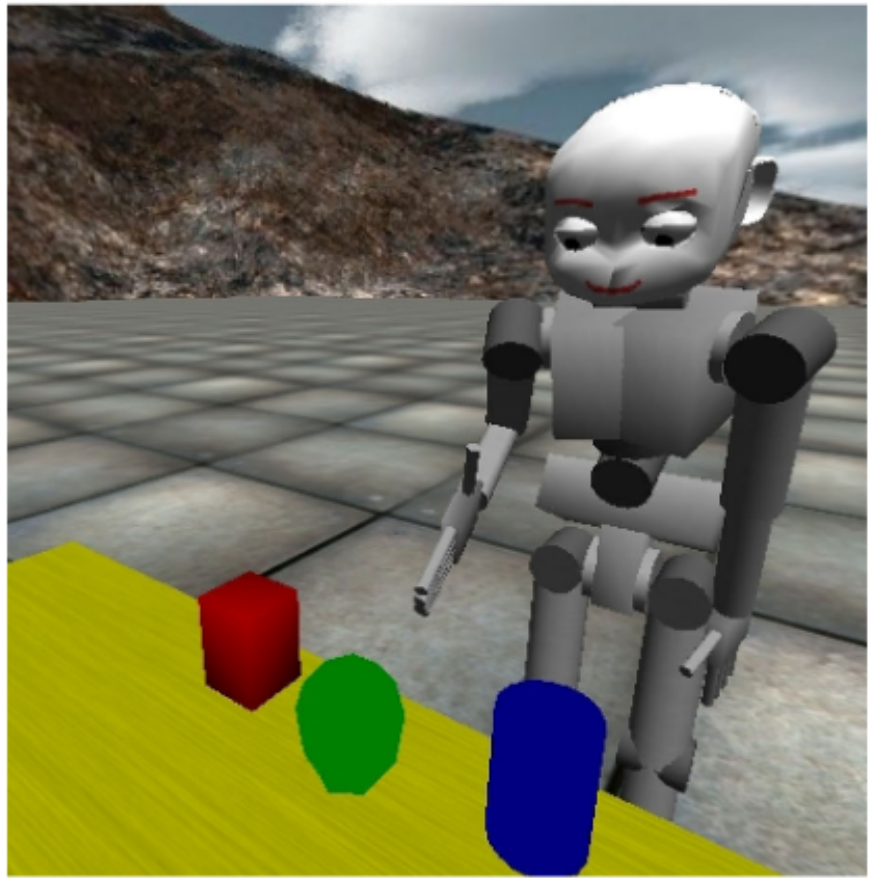
Detección de Spam



- **Entrada**: palabras en correo, junto con la cantidad de repeticiones
- **Salida**: Spam / No-Spam

Reconocimiento de objetos

iCub Simulator

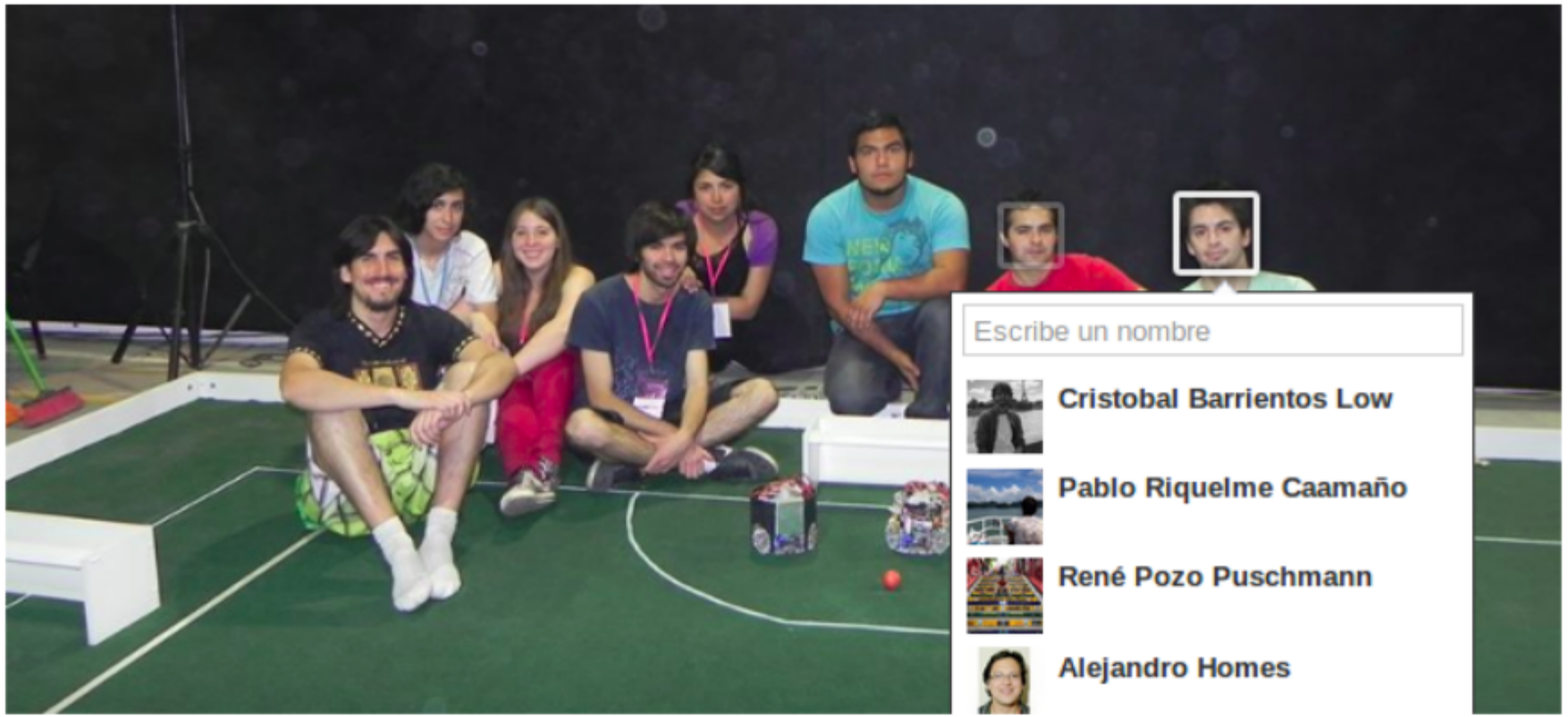


(algunos) Tipos de aprendizaje

- Aprendizaje no supervisado: de los datos de entrenamiento, sólo se conocen las entradas
- Aplicaciones:
  - Extracción de características
  - Reducción de la dimensionalidad

Reconocimiento de amigos en Facebook

Sugerencia en el etiquetado de amigos



Aprendizaje reforzado

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

Introducción

Aprendizaje reforzado

- Fundamentos
- Definición de un problema en RL

Métodos

- Plataforma para ejemplos
- Value Iteration
- Policy Iteration
- Q-learning

Aplicaciones en robótica

Psicología

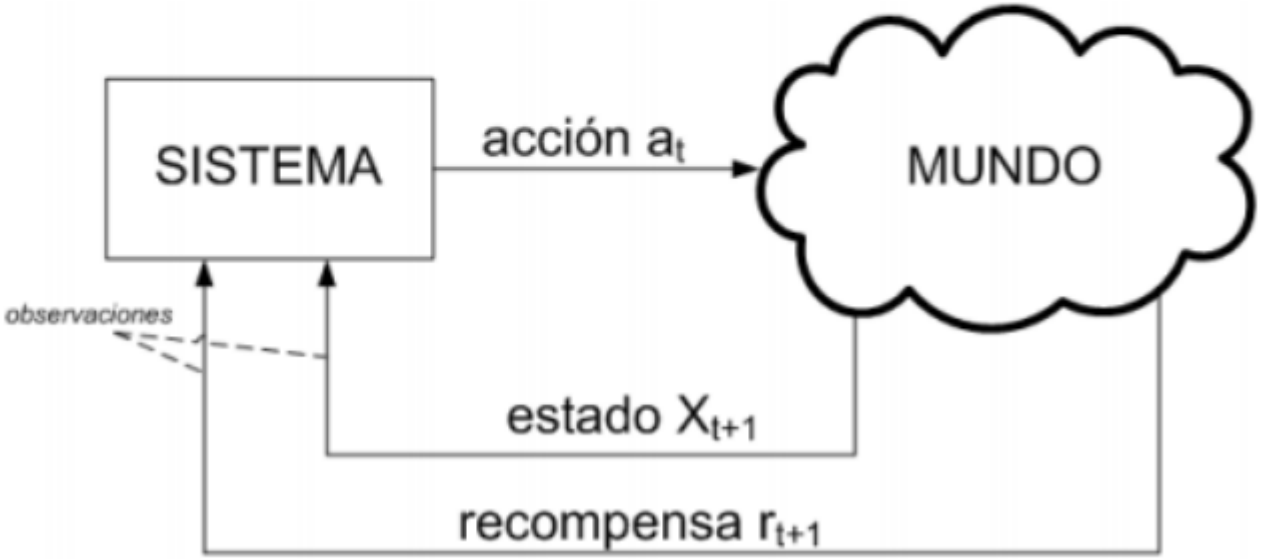
- Edward L. Thorndike - Animal Intelligence: An experimental study of the associate processes in animals (1898)
- El animal modifica su conducta según interacción de prueba y error con el medio ambiente

Interacción

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 (recompensa).

Esquema


SISTEMA
acción a_t
MUNDO
observaciones
estado X_{t+1}
recompensa r_{t+1}

Elementos

Un problema de aprendizaje reforzado (RL - Reinforcement Learning), formulado como un MDP (Proceso de Decisión Markoviano) está compuesto por $(S, A, \mathbf{T}, R)$ donde...

- ¿Markoviano?

Proceso de Markov

Un estado s_k se dice que obedece a un proceso de Markov (de 1er orden) ssi:

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

Elementos

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

- S: Conjunto de estados
- A: Conjunto de acciones
- T: $S \times A \times S \rightarrow [0, 1]$ (desconocido)
- R: $S \times A \times S \rightarrow \mathbb{R}$.
- π: $S \rightarrow A$.

Aprendizaje Reforzado

Estado $s_k \in S$: Acción $a_k \in A$

$$s_k \rightarrow^{a_k} s_{k+1}$$

Función de transición de estado T:

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

Recompensas

$$r_k = r(s_{k-1}, a_k, s_k)$$

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

Politica π: $S \rightarrow A$

Una politica es óptima si maximiza la recompensa a largo plazo

Retorno

Función de Valor:

$$V^{\pi}(s_k) = r_k + \gamma r_{k+1} + \gamma^2 r_{k+2} + \ldots = \sum_{i=0}^{\infty} \gamma^i r_{k+i}$$

$$0 \leq \gamma < 1, \qquad r_k \text{ acotado}$$

Una politica π^* es óptima si:

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

Introducción

Aprendizaje reforzado

Fundamentos

Definición de un problema en RL

Métodos

Plataforma para ejemplos

Value Iteration

Policy Iteration

Q-learning

Aplicaciones en robótica

Jupyterhub

https://python.innovacionyrobotica.com

- Creación libre de cuentas hasta 14-nov-2018
  - **pass:** ciistacna

Jupyterhub

a) Abrir python notebook disponible

b) Guardar copia local para poder modificar el código (guardar como)

c) Cerrar notebook actual y abrir copia creada

Value Iteration

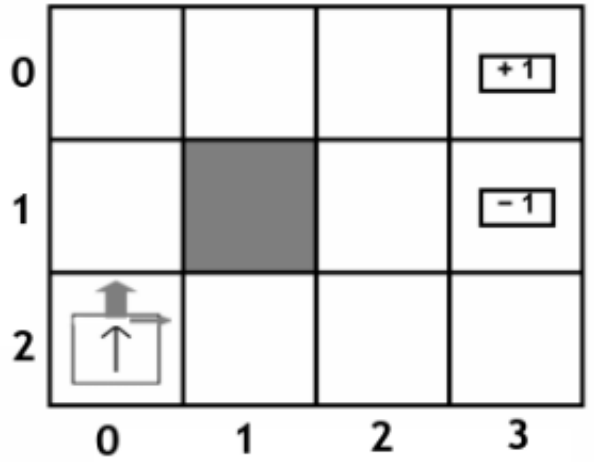
Consiste en realizar k iteraciones, hasta que $V_k(s) - V_{k-1}(s)$ es suficientemente pequeño, con actualización según

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

Value Iteration

1: $k = 0$
2: $\hat{V}_0(x) = 0 \qquad \forall s \in \mathcal{S}$
3: **while** $\Delta > \epsilon$ **do** ▷ (para ϵ pequeño)
4: **for** $s \in \mathcal{S}$ **do**
5: $\hat{V}_{k+1}(x) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} Pr\{(s, a, s')\} (R(s, a) + \gamma V(s'))$
6: **end for**
7: $\Delta = \| \hat{V}_{k+1} - \hat{V}_k \|$
8: $k = k + 1$
9: **end while**
10: **return** π

Ejemplo 1 (ver Python Notebook)



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

Ejemplo 1 (ver Python Notebook)

Después de 1 iteración:

0	-0.1	-0.1	1	+1
1	-0.19		-0.1	-1
2	-0.1	-0.1	-0.19	-0.1
	0	1	2	3

Ejemplo 1 (ver Python Notebook)

Después de 2 iteraciones:

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

Ejemplo 1 (ver Python Notebook)

En convergencia:

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

Toma de decisiones

Entonces...

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

- debo ejecutar necesariamente la acción que maximiza este argumento?
  - comportamiento greedy
  - comportamiento ϵ-greedy

Policy Iteration

1: $\hat{\pi}_0(s) =$ acción aleatoria $\quad \forall s \in \mathcal{S}$
2: $\hat{V}_0(s) = 0 \quad \forall s \in \mathcal{S}$ ▷ arbitrariamente
3: evaluar_politica()
4: mejorar_politica()

Policy Iteration

evaluar_politica():

1: $k = 0$
2: **while** $\Delta > \epsilon$ **do** ▷ (para ϵ pequeño)
3: **for** $s \in \mathcal{S}$ **do**
4: $\hat{V}_{k+1}(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} Pr\{(s, a, s')\} (R(s, a) + \gamma V(s'))$
5: **end for**
6: $\Delta = \| \hat{V}_{k+1} - \hat{V}_k \|$
7: $k = k + 1$
8: **end while**

Policy Iteration

mejorar_politica() :

1: $politica\_estable = true$
2: $k = 0$
3: **for** $s \in \mathcal{S}$ **do**
4: $\quad \hat{\pi}_{k+1}(s) = argmax_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} Pr\{(s, a, s')\} \left(R(s, a) + \gamma V(s')\right)$
5: $\quad$ **if** $\hat{\pi}_k(s) \neq \hat{\pi}_{k+1}(s)$ **then**
6: $\quad\quad politica\_estable = false$
7: $\quad$ **end if**
8: **end for**
9: **if** *politica_estable* **then**
10: $\quad$ **return** $\hat{\pi}_k$
11: **else**
12: $\quad$ evaluar_politica()
13: **end if**

Ejemplo 2 (ver Python Notebook)

- Se encuentra de forma directa la política
- De todas formas se evalúa la función de valor

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

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

Q-learning

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

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

Q-learning

1: $\hat{Q}(s_0, a_0) = 0 \qquad \forall s \in \mathcal{S}, \quad a \in \mathcal{A}$
2: **for** $k = 1, 2, ..., n$ **do**
3: **Observar** $s_k, a_k, R(s_k, a_k), s_{k+1}$
4: $\hat{Q}(s_k, a_k) = \hat{Q}(s_k, a_k) + \alpha \left(R_{k+1} + \gamma \max_{a_{k+1}} \hat{Q}(s_{k+1}, a_{k+1}) - \hat{Q}(s_k, a_k) \right)$
5: **end for**
6: **return** $\hat{Q}$

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

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

Ejemplo 3 (ver Python Notebook)

- 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

Introducción

Aprendizaje reforzado

- Fundamentos
- Definición de un problema en RL

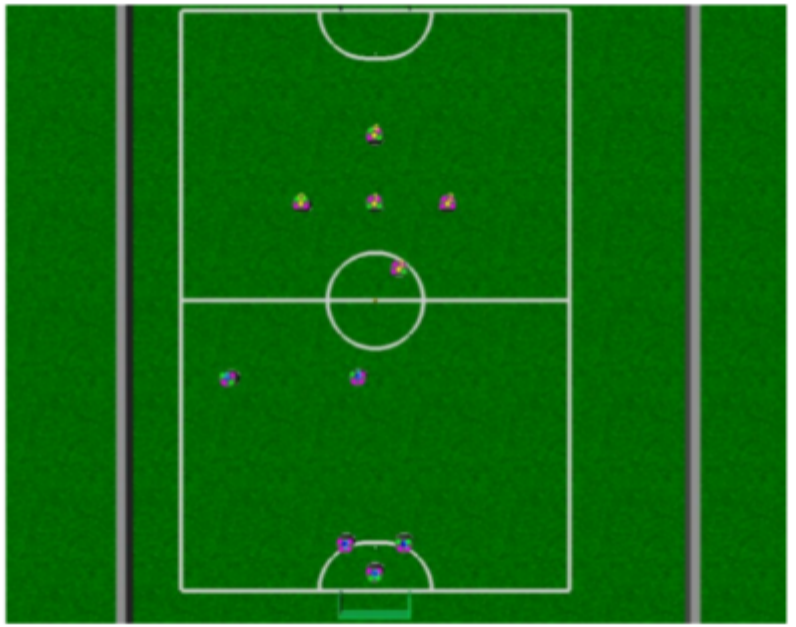
Métodos

- Plataforma para ejemplos
- Value Iteration
- Policy Iteration
- Q-learning

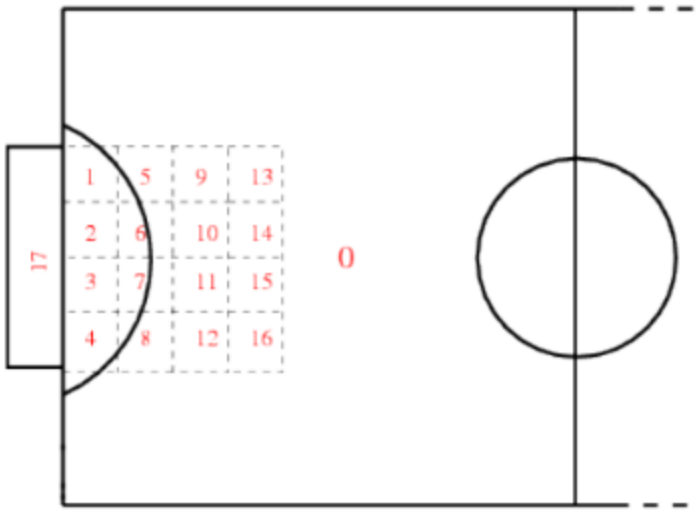
Aplicaciones en robótica

RoboCup SSL

- 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.



RoboCup SSL

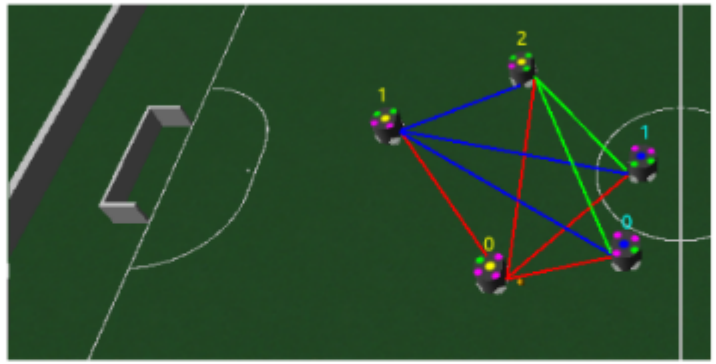


Estado compuesto por

- posición de pelota
- posición y orientación de arquero

RoboCup SSL

Generación de estrategia defensiva



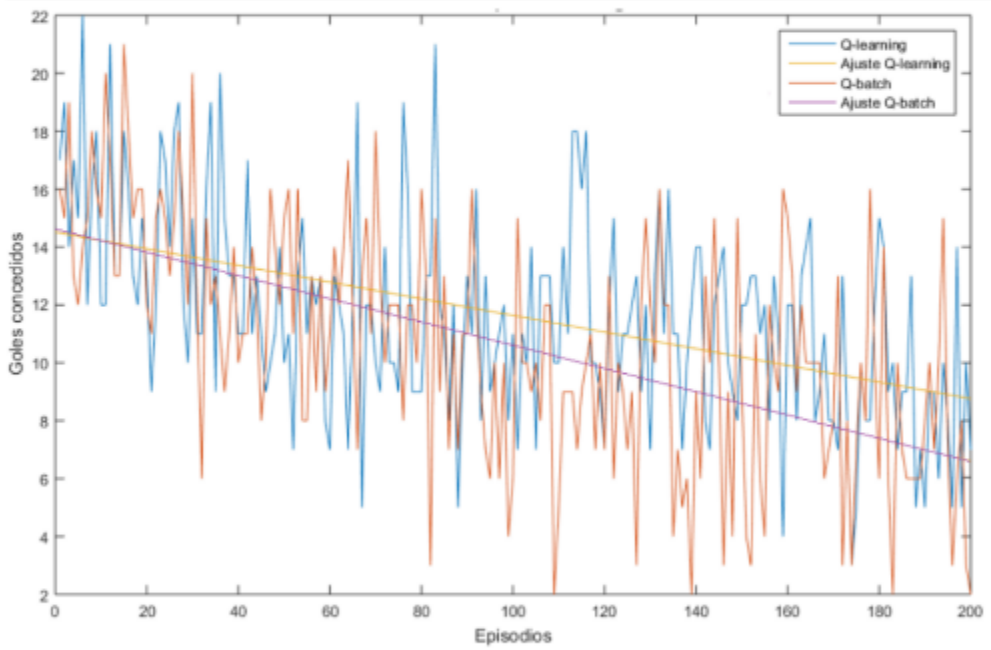
estado compuesto por:

- dist(K_i,pelota), dist(T_j,pelota)
- dist(K_i,K_j)
- dist(K_i,T_j)
- angle(K_i,T_j)

Memoria de Ingeniería Civil Informática - Franco Ollino (2016)

RoboCup SSL

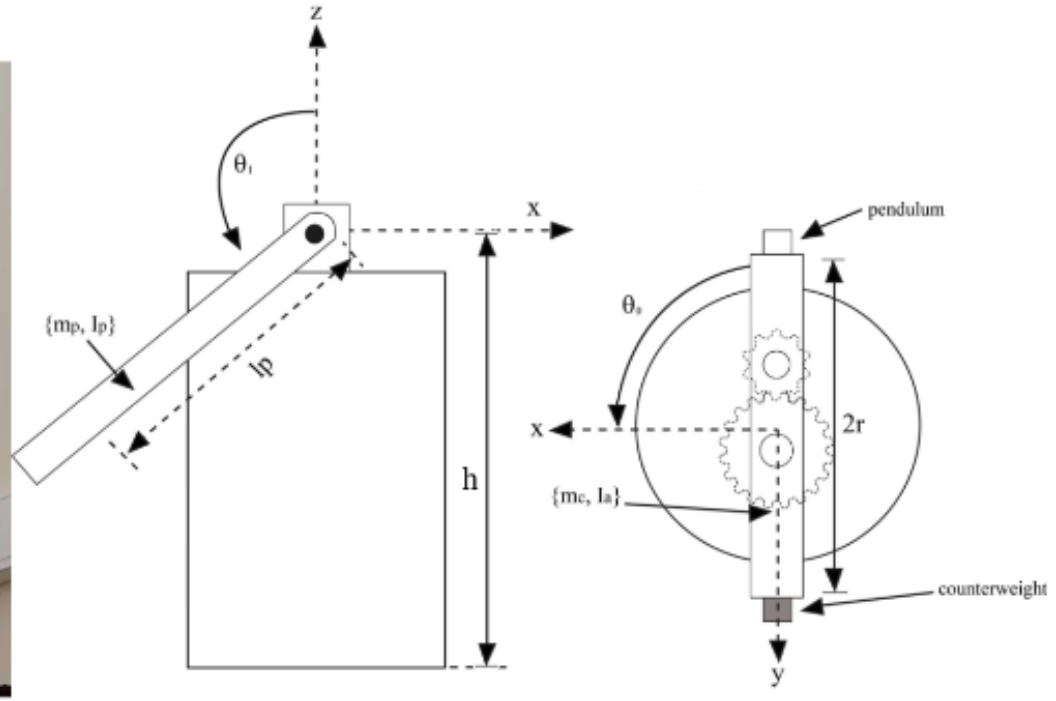
Generación de estrategia defensiva



- recolección de experiencias (transiciones)
- actualización por lotes

Péndulo invertido rotatorio

Péndulo de Furuta



- Sintonización de parámetros del controlador

Lectura recomendada

- Richard S. Sutton and Andrew G. Barto. **Reinforcement Learning: An Introduction**, 1998.
- Jens Kober, J Andrew Bagnell, and Jan Peters. **Reinforcement learning in robotics: A survey**. The International Journal of Robotics Research, 32(11):1238–1274, 2013.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. **Human-level control through deep reinforcement learning**. Nature, 518(7540):529-533, 2015.

Preguntas?

CONGRESO INTERNACIONAL DE INFORMÁTICA Y SISTEMAS

"Gestión del conocimiento e innovación tecnológica"