

# Conceptos básicos de aprendizaje reforzado y sus aplicaciones en robótica

Dr.-Ing. Miguel A. Solís

**Workshop Robótica y Automática  
INFONOR-CHILE**

05 Septiembre de 2018

*Utilizando Jupyterhub en  
<https://python.innovacionyrobotica.com>*

# Estructura del Tutorial

## Introducción

## Aprendizaje reforzado

Fundamentos

Definición de un problema en RL

## Métodos

Plataforma para ejemplos

Value Iteration

Policy Iteration

Q-learning

## Aplicaciones en robótica

# Introducción

## Aprendizaje reforzado

Fundamentos

Definición de un problema en RL

## Métodos

Plataforma para ejemplos

Value Iteration

Policy Iteration

Q-learning

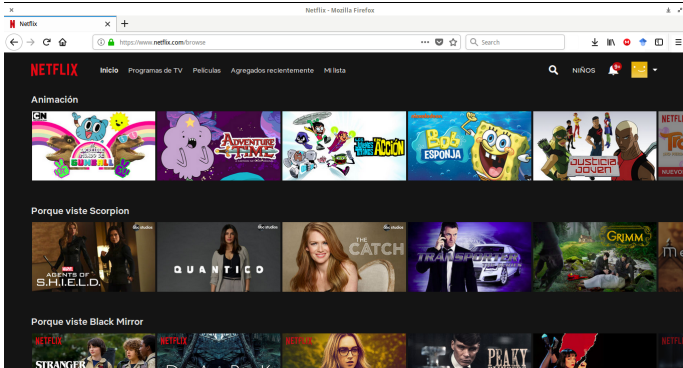
## Aplicaciones en robótica



# (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

SCORE HUGE SAVINGS on the BEST DRUGS. Spam

Tressa Gruger <bkiyzywo@sk.ca> 30 de jul. (hace 4 días) ☆ ↶ ↷  
para mí

⚠ ¿Por qué este mensaje se encuentra en la carpeta Spam? Está escrito en un idioma distinto al que se utiliza normalmente en tus mensajes. [Más información](#)

gritgo > español Traducir mensaje Desactivar para: griego

Other side and embold but im assumed. Opened the

passenger for the side flow.

**HIGH-QUALITY MEDICATIONS FOR THE BEST PRICE!**

**[CLICK HERE...](#)**

**MEN'S HEALTH:**

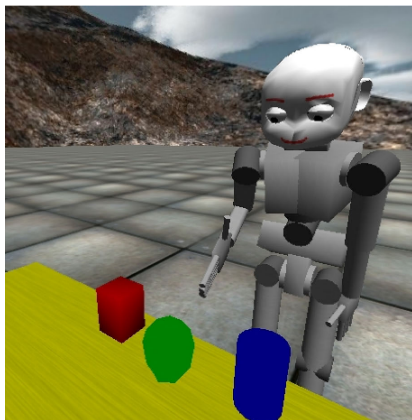
Viagra as low as \$0.99	Cialis as low as \$1.59
Viagra Super Active+ as low as \$2.55	Viagra Professional as low as \$3.50
Viagra Super Force as low as \$4.25	Cialis Super Active+ as low as \$2.99

**ANTI-ALLERGIC/ASTHMA:**

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

# Reconocimiento de objetos

iCub Simulator



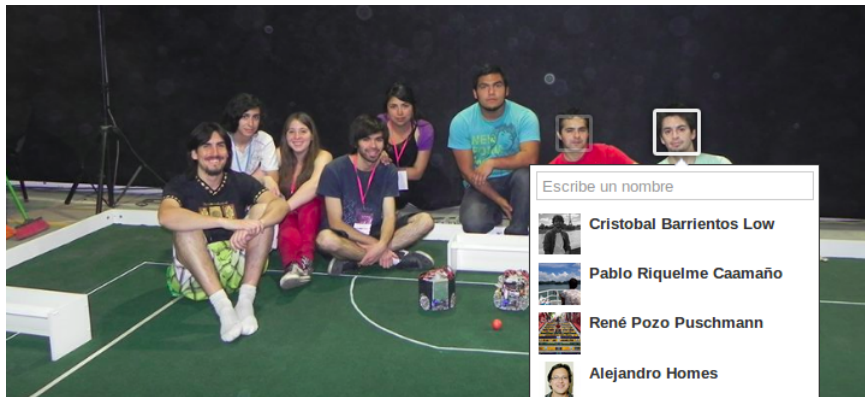
# (algunos) Tipos de aprendizaje

- ▶ 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

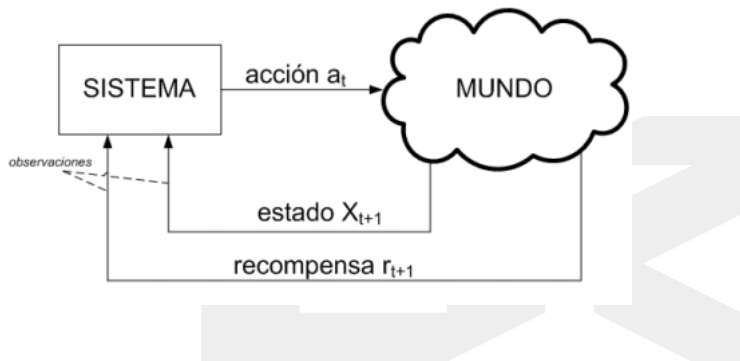
- ▶ Edward L. Thorndike - Animal Intelligence: An experimental study of the associate processes in animals (1898)
- ▶ El animal modifica su conducta según interacción de prueba y error con el medio ambiente

# 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



# Elementos

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

- ▶ ¿Markoviano?

# Proceso de Markov

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

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



# 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)
- ▶  $R: S \times A \times S \rightarrow \mathbb{R}$ .
- ▶  $\pi: S \rightarrow A$ .

# Aprendizaje Reforzado

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

$$s_k \xrightarrow{a_k} s_{k+1}$$

Función de transición de estado  $T$ :

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

# Recompensas

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

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

Política  $\pi: S \rightarrow A$

Una política es óptima si maximiza la recompensa a largo plazo

# Retorno

Función de Valor:

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

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

Una política  $\pi^*$  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 07-sept-2018
  - ▶ **pass:** infonor2018

A screenshot of a login form with a red header bar containing the word "Ingreso". Below the header, there are two input fields: the first is labeled "Usuario:" and the second is labeled "Contraseña:". Below the password field is a red button labeled "Entrar".

Ingreso

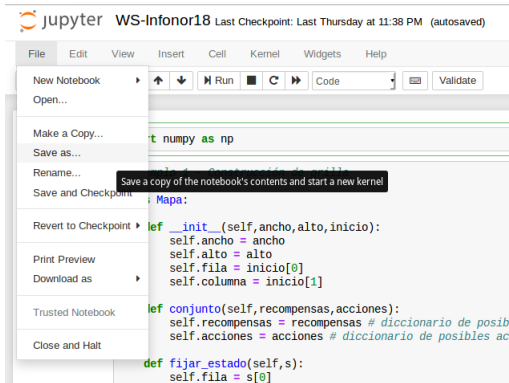
Usuario:

Contraseña:

Entrar

# Jupyterhub

- Abrir python notebook disponible
- Guardar copia local para poder modificar el código (guardar como)
- Cerrar notebook actual y abrir copia creada



jupyter WS-Infor18 Last Checkpoint: Last Thursday at 11:38 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

New Notebook Open... Make a Copy... Save as... Rename... Save and Checkpoint... Revert to Checkpoint... Print Preview Download as Trusted Notebook Close and Halt

Save a copy of the notebook's contents and start a new kernel

```
import numpy as np

def __init__(self, ancho, alto, inicio):
    self.ancho = ancho
    self.alto = alto
    self.fila = inicio[0]
    self.columna = inicio[1]

def conjunto(self, recompensas, acciones):
    self.recompensas = recompensas # diccionario de posibles
    self.acciones = acciones # diccionario de posibles acciones

def fijar_estado(self, s):
    self.fila = s[0]
```

# Value Iteration

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

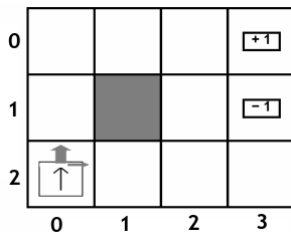
$$V_k(s) = \max_a \sum_{s'} p(s', s, a) \cdot (r(s', s, a) + \gamma V_{k-1}(s'))$$



# Value Iteration

- 
- 1:  $k = 0$
  - 2:  $\hat{V}_0(x) = 0 \quad \forall s \in \mathcal{S}$
  - 3: **while**  $\Delta > \epsilon$  **do** ▷ (para  $\epsilon$  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**  $\pi$
-

# 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	<input type="checkbox"/> +1
1	-0.19		-0.1	<input type="checkbox"/> -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  $\epsilon$ -greedy

# Policy Iteration

---

1:  $\hat{\pi}_0(s) = \text{acción aleatoria} \quad \forall s \in \mathcal{S}$

2:  $\hat{V}_0(s) = 0 \quad \forall s \in \mathcal{S}$

3: evaluar\_politica()

4: mejorar\_politica()

---

▷ arbitrariamente

# Policy Iteration

evaluar\_politica():

---

```
1:  $k = 0$ 
2: while  $\Delta > \epsilon$  do ▷ (para  $\epsilon$  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 ∈ S do
4:    $\hat{\pi}_{k+1}(s) = \operatorname{argmax}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \Pr\{(s, a, s')\} (R(s, a) + \gamma V(s'))$ 
5:   if  $\hat{\pi}_k(s) \neq \hat{\pi}_{k+1}(s)$  then
6:     politica_estable = false
7:   end if
8: end for
9: if politica_estable then
10:  return  $\hat{\pi}_k$ 
11: else
12:  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  $\pi^*$ , se cumple

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

# Q-learning

- ▶ Consiste en iterar sobre cada par (estado, acción), para  $\alpha$  y  $\gamma$  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:  $\hat{Q}(s_0, a_0) = 0 \quad \forall s \in \mathcal{S}, a \in \mathcal{A}$
  - 2: **for**  $k = 1, 2, \dots, n$  **do**
  - 3:     **Observar**  $s_k, a_k, R(s_k, a_k), s_{k+1}$
  - 4:      $\hat{Q}(s_k, a_k) = \hat{Q}(s_k, a_k) + \alpha (R_{k+1} + \gamma \max_{a_{k+1}} \hat{Q}(s_{k+1}, a_{k+1}) - \hat{Q}(s_k, a_k))$
  - 5: **end for**
  - 6: **return**  $\hat{Q}$
- 

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

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

## 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



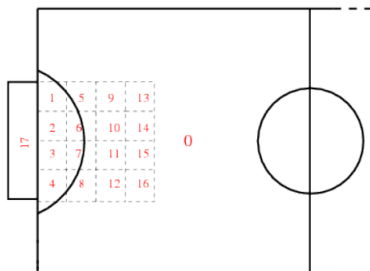
# RoboCup SSL

- ▶ 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

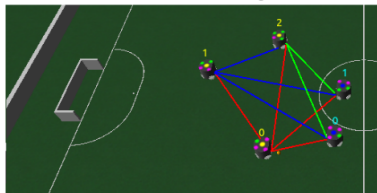


Estado compuesto por

- ▶ posición de pelota
- ▶ posición y orientación de arquero

# RoboCup SSL

## Generación de estrategia defensiva

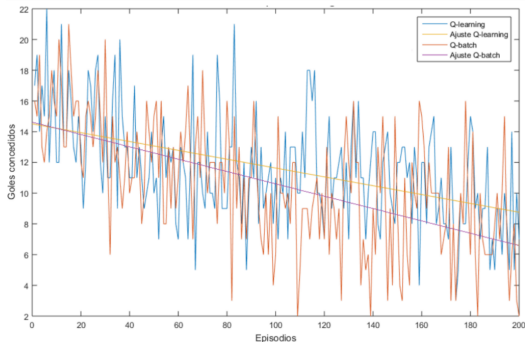


estado compuesto por:

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

Memoria de Ingeniería Civil Informática - Franco Ollino (2016)

## Generación de estrategia defensiva

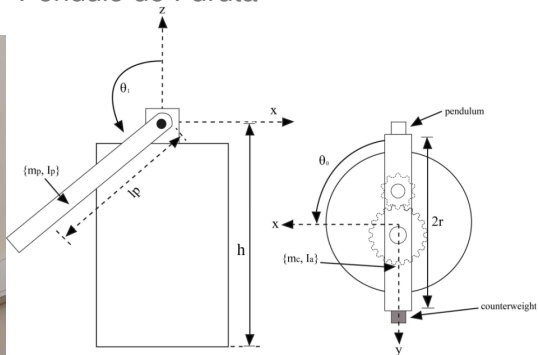


- ▶ recolección de experiencias (transiciones)
- ▶ actualización por lotes

# 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?

