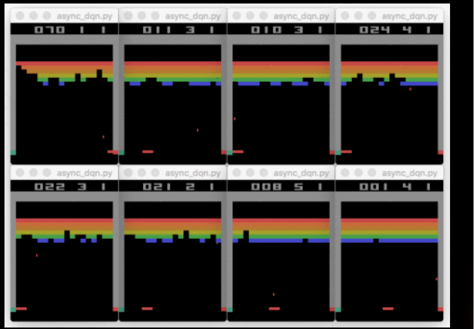


Robótica inteligente mediante Reinforcement Learning

Juan G Victores
RoboticsLab – Universidad Carlos III de Madrid

@jgvictores
http://roboticslab.uc3m.es
jcgvicto@ing.uc3m.es

SUMMER SCHOOL
CEA Comité Español de Automática
ALMA

Robótica inteligente mediante Reinforcement Learning

- ❑ **Reinforcement Learning dentro de la IA**
- ❑ **Conceptos Principales en RL**
- ❑ **Ejemplos de Algoritmos en RL**
- ❑ **Robótica: Expert Rules hasta End-to-end**
- ❑ **Conclusiones y recursos**

Artificial Intelligence

- Artificial General Intelligence (AGI)
- Narrow AI
  - Optimization (Gradient Descent, Evol. Comput...)
  - Good Old-Fashioned AI (Cognitivist/Symbolical)
  - Machine Learning (Connectionist/Statistical)

Artificial Intelligence

- Artificial General Intelligence (AGI)
- Narrow AI
  - Optimization (Gradient Descent, Evol. Comput...)
  - Good Old-Fashioned AI (Cognitivist/Symbolical)
  - Machine Learning (Connectionist/Statistical)

Machine Learning (Tipos de Problemas)

Supervised Learning

Unsupervised Learning

Reinforcement Learning

Aún más vía Deep Learning
(p.ej. Selfsupervised Learning, Style Transfer, GANs…)

Machine Learning (Tipos de Problemas)

Supervised Learning

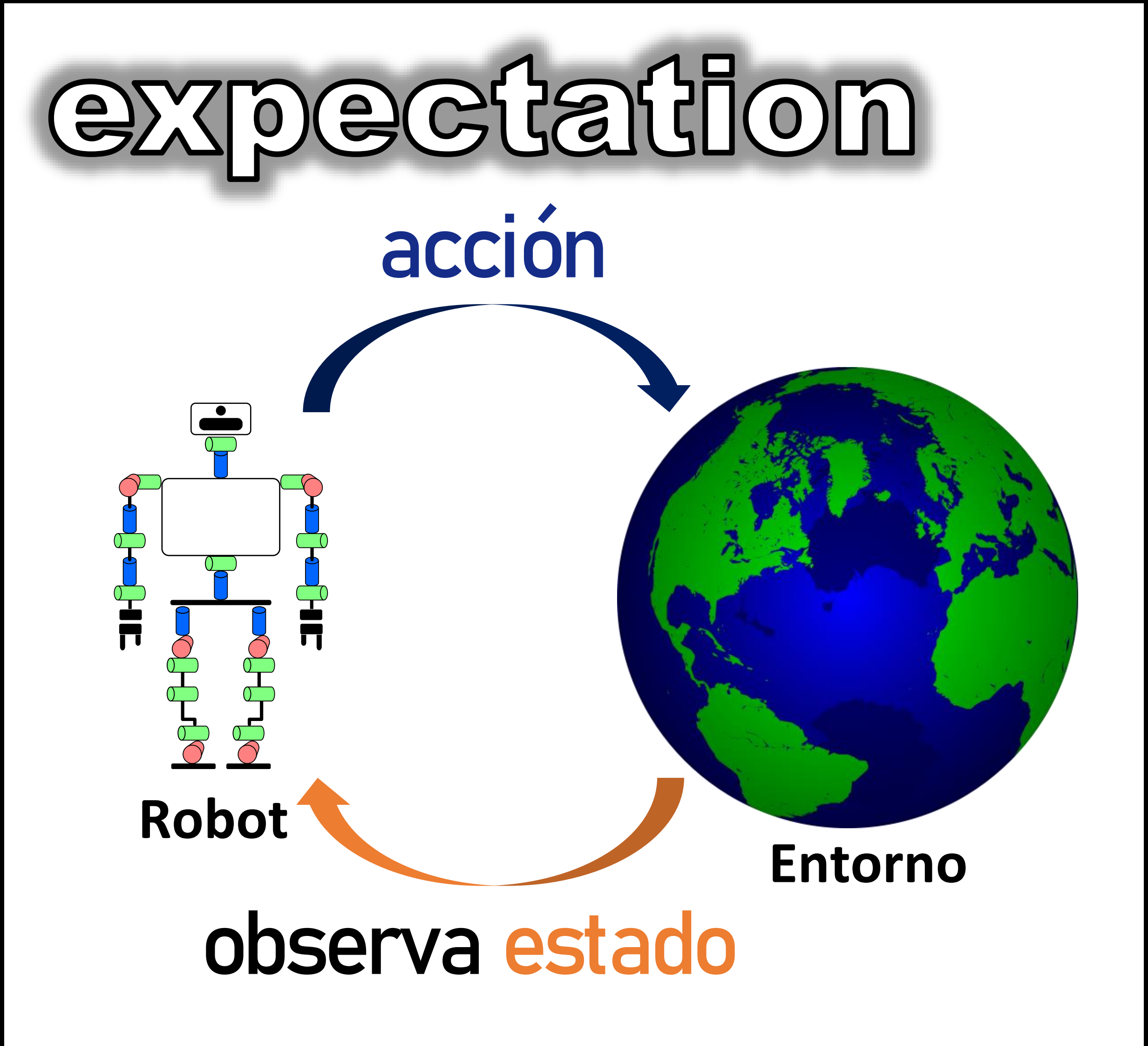
Unsupervised Learning

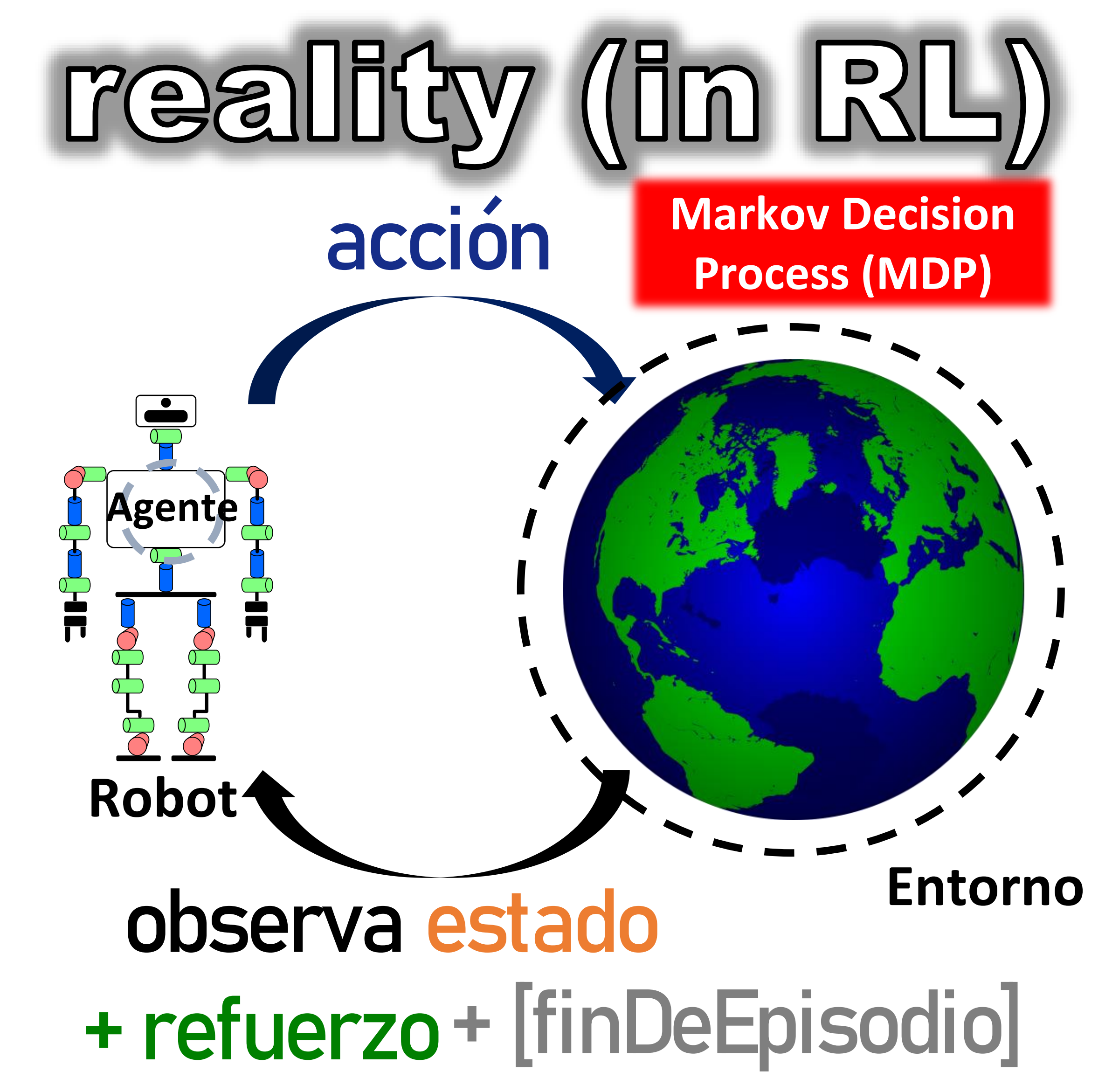
Reinforcement Learning

Aún más vía Deep Learning
(p.ej. Selfsupervised Learning, Style Transfer, GANs...)

INTERACCIÓN ROBOT-ENTORNO

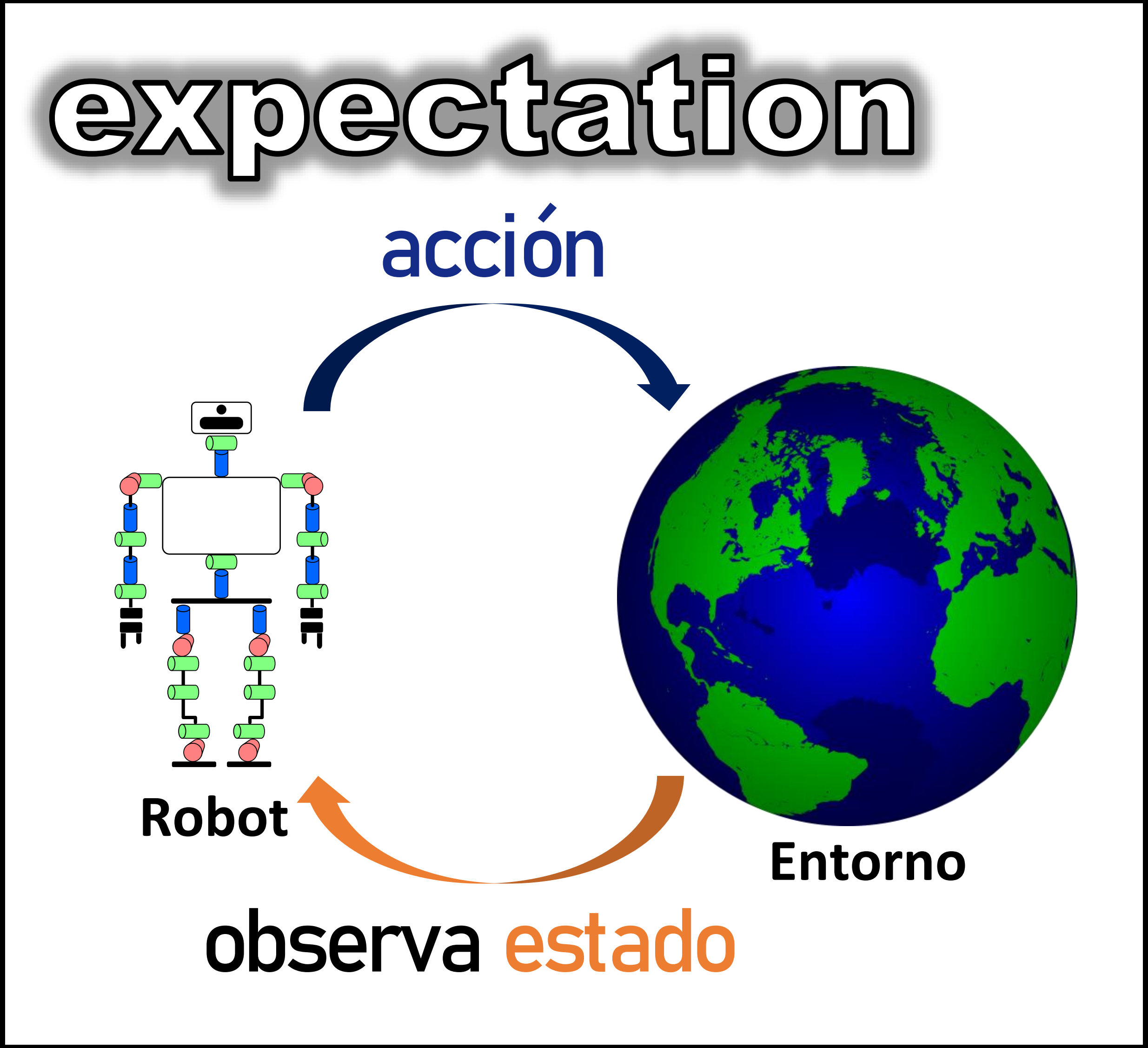
@jgvictores

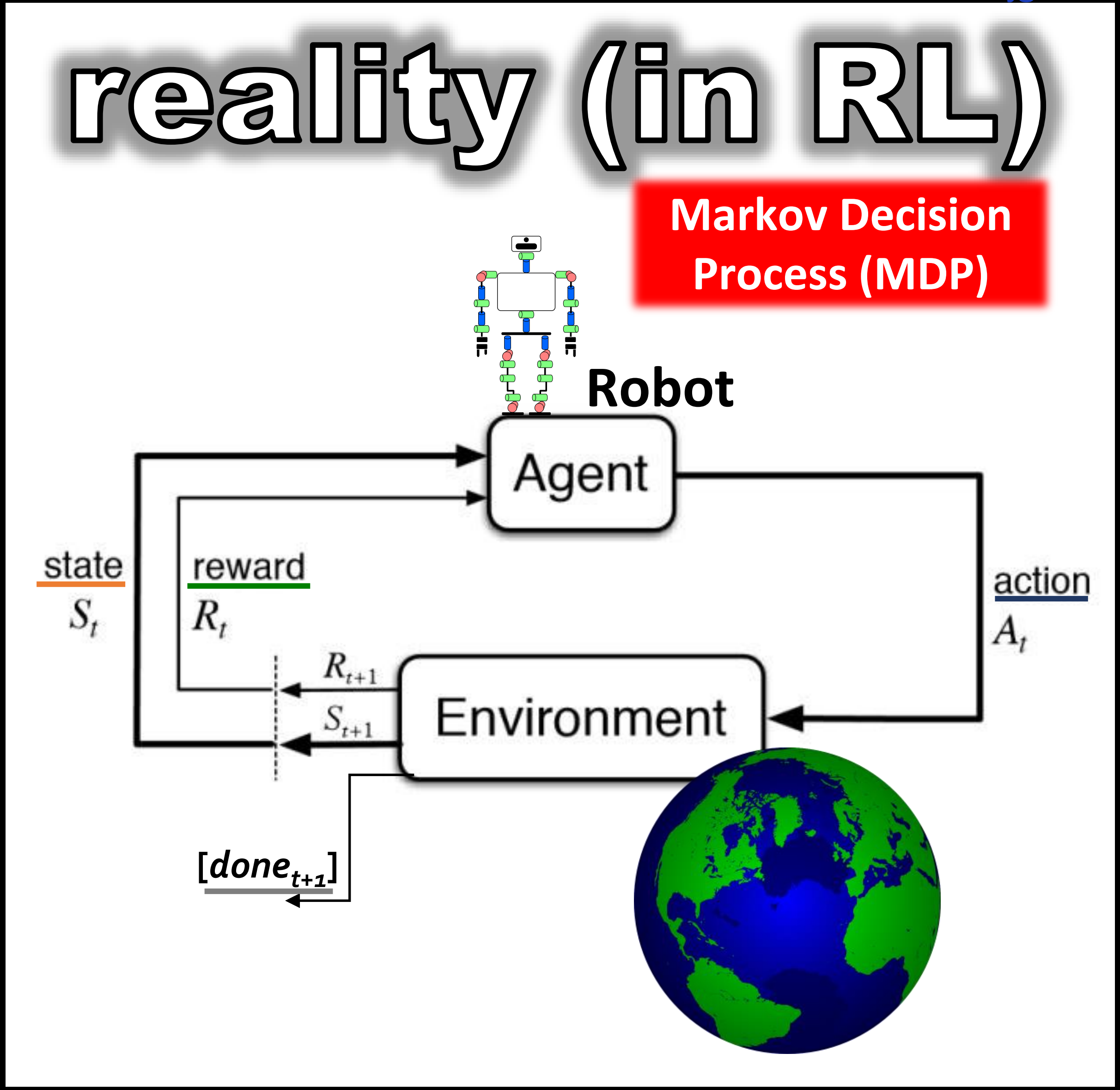




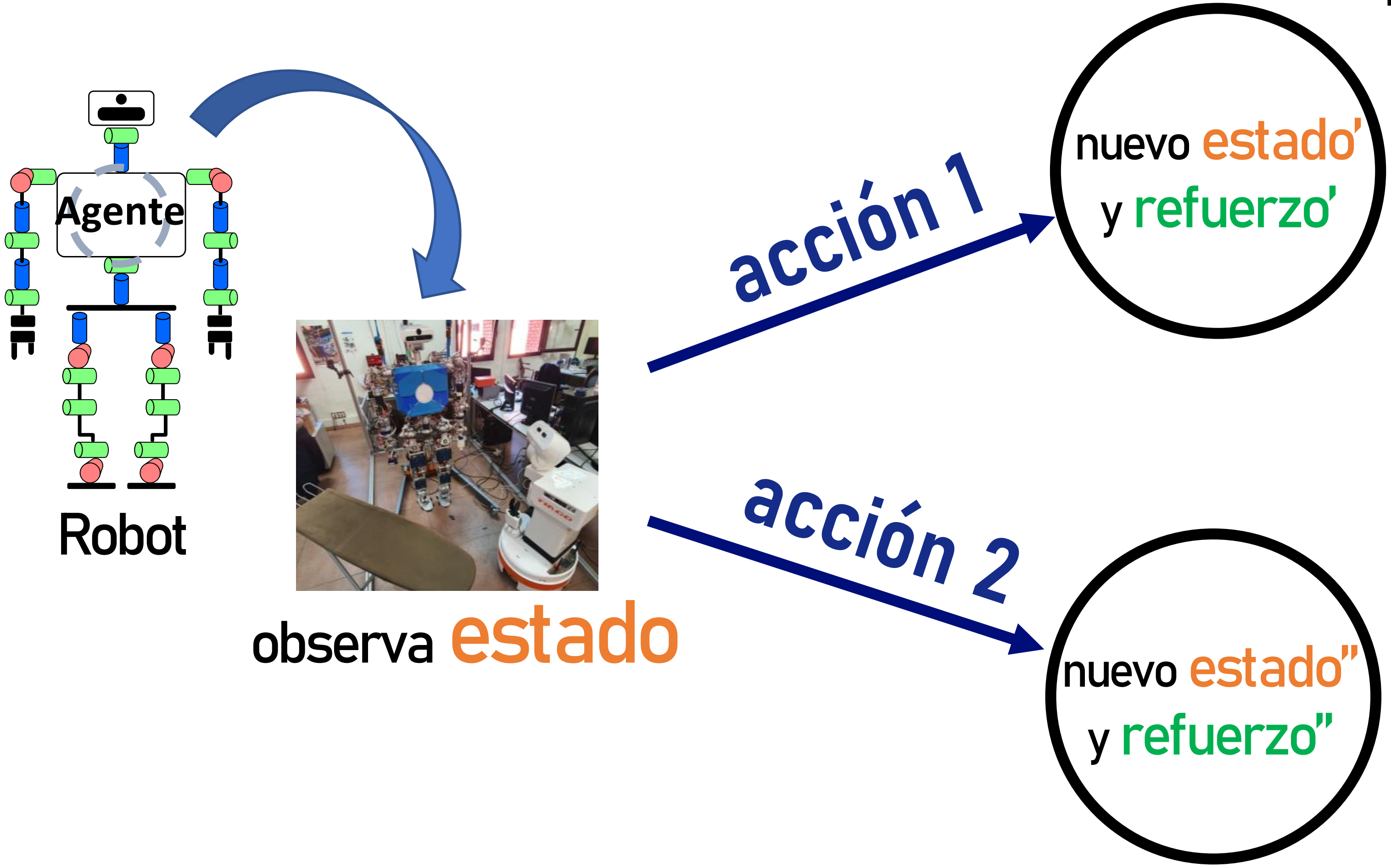
INTERACCIÓN ROBOT-ENTORNO

@jgvictores





Reinforcement Learning (Tipo de Problema)



Toma de decisiones

Objetivo: ley de control

acción = π(estado)

Maximiza: f(refuerzo)

Supuestos:

Markov Assumption

Markov Decision Process

π directa o vía V/Q

Reinforcement Learning: Técnicas

Watkins (1989): Q-Learning

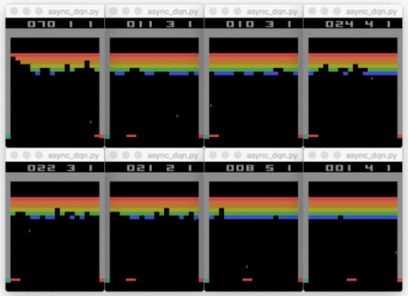
- π(s) / V(s) / Q(s,a) / Actor-Critic

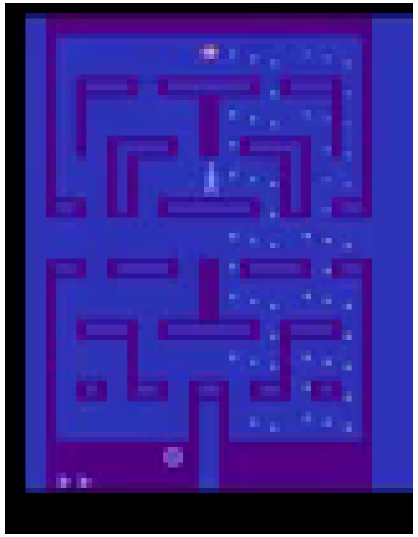
Mnih (2015-): Deep Q-Network (DQN)

- Tabular / Function Approximator
- f(reward): Average / Discounted / (Return)
- Model-free / Model-based
- MDP / POMDP / …
- On-policy / Off-policy

Mnih (2015-): Deep Q-Network (DQN)

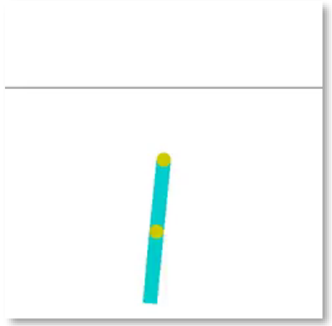
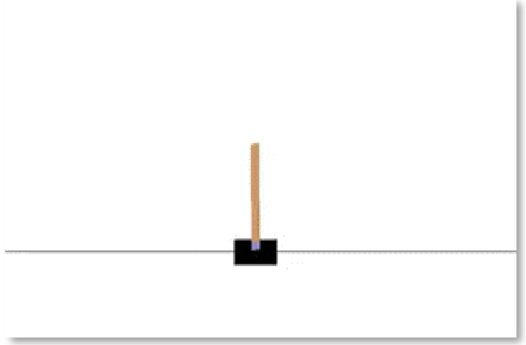
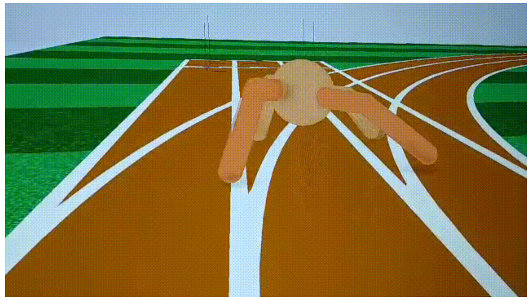
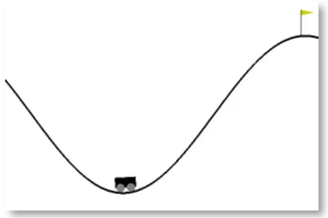
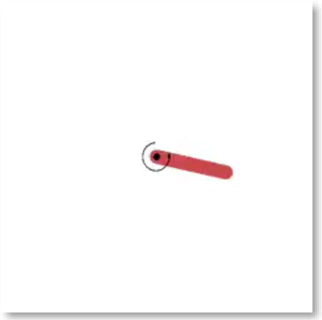
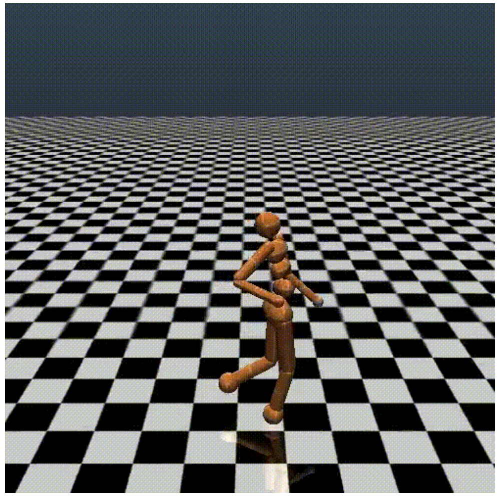
RL+DeepL=DRL: en videojuegos (2015-)





Schulman (2015-): TRPO

DRL: en simuladores de robots (2015-)

DRL: en robots (2015-)

Levine (2015-): End-to-End Training of Deep Visuomotor Policies

Conceptos RL (agente y entorno)

s: estado = x

a: acción = u (управление)

r: refuerzo (recompensa)

Ecuaciones de Bellman (fundamentos y proofs matemáticos)

Conceptos RL (entorno)

Algoritmos "Model-Based": Intentan hallar modelo de P
Algoritmos "Model-Free": El resto (muy habitual)

P: dinámica=transiciones

En un MDP, el entorno devuelve estado completo.
En un Partialy Observable MDP (POMDP), sólo devuelve una parte: la observación

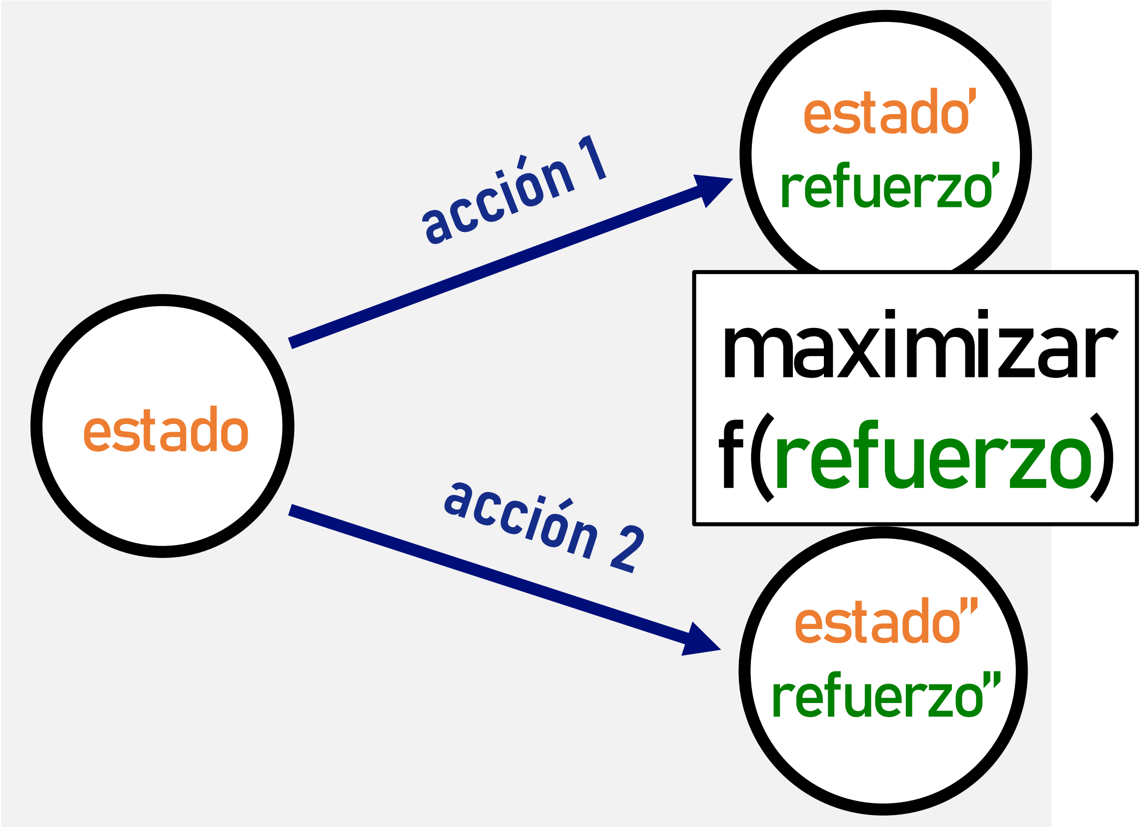
Determinista	Probabilística
s',r',fin = P(s,a)	P(s',r',fin | s,a)

fin: "episodic" (vs. "continuing"="infinite")

isd(s): distribución inicial de estados

Conceptos RL (agente)

π: Ley de control (policy, cómo actúa agente)



Determinista

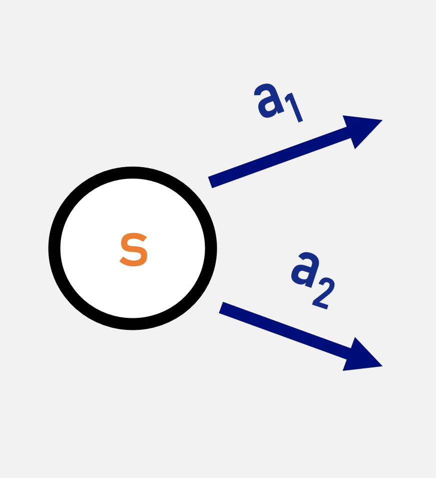
acción=π(estado)

Probabilística

π(acción|estado)

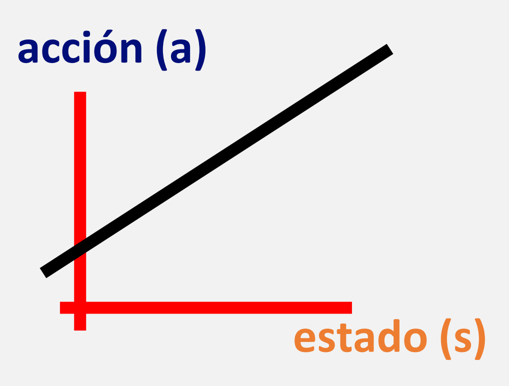
Conceptos RL (agente)

π: Ley de control, acción=π(estado)



- Modelo explícito
  - Tabla
  - Función
- Modelo que depende de Q o V

estado (s)	acción (a)
s_1	...
s_2	...
s_3	...
...	...



Conceptos RL (agente)

Q(s,a): Q-Value, action-value. Modela el valor de cada acción para cada estado.

Es opcional. Tablular / function approx.

π "greedy"
$\pi = \operatorname{argmax}_a(Q(s,a))$

π "ε-greedy"
$\begin{cases} (1-\varepsilon): \operatorname{argmax}_a(Q(s,a)) \\ (\varepsilon): \text{random } a \end{cases}$

Conceptos RL (agente)

V(s): Value, state value. Modelo valor de cada estado.

Es opcional. Tablular / function approx.

Una π “greedy” sería aquella que siempre escoja la a que lleve al s’ (estado contiguo) de mayor V(s’).

Conceptos RL (agente)

- **γ: Discount [0, 1]. Opcional. Pondera importancia del refuerzo futuro.**
- **G: Return. Refuerzos ponderados según discount y caso de uso específico.**

$$G_t = \sum_t^T r_{t+1} + r_{t+2} + \ \dots + r_T$$

$$G_t = \sum_t^\infty r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots$$

Robótica inteligente mediante Reinforcement Learning

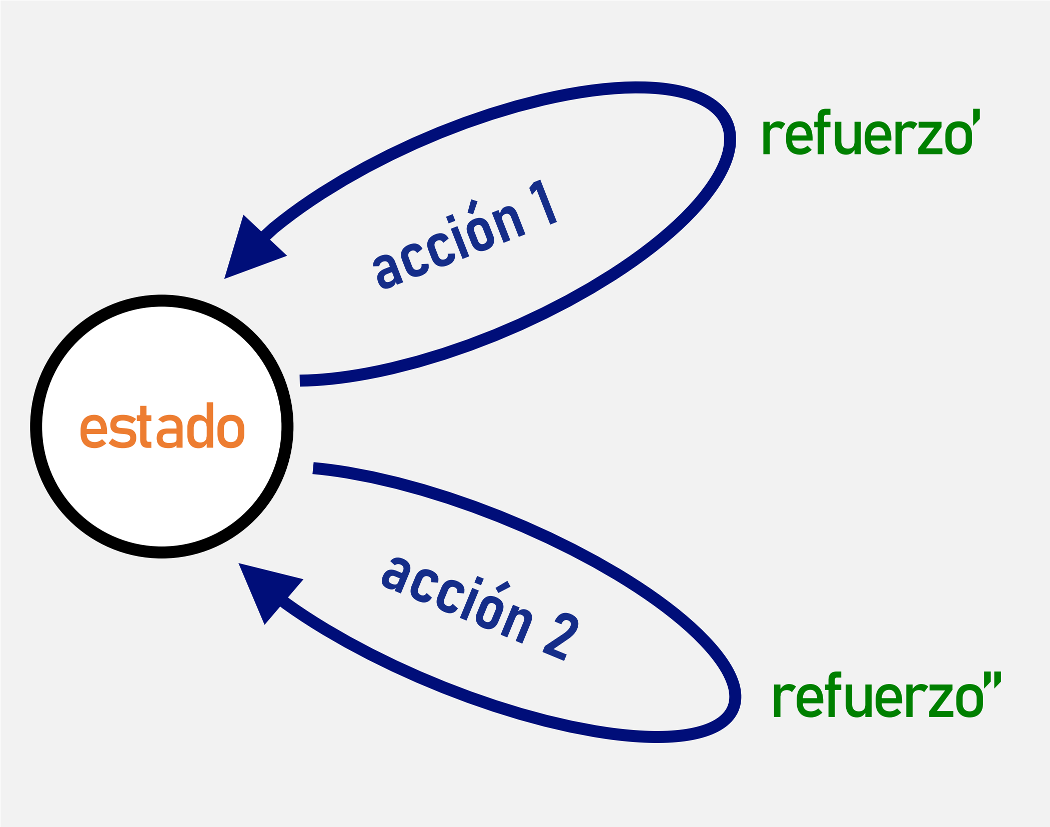
- ❑ **Reinforcement Learning dentro de la IA**
- ❑ **Conceptos Principales en RL**
- ❑ **Ejemplos de Algoritmos en RL**
- ❑ **Robótica: Expert Rules hasta End-to-end**
- ❑ **Conclusiones y recursos**

Robótica inteligente mediante Reinforcement Learning

- ❑ Reinforcement Learning dentro de la IA
- ❑ Conceptos Principales en RL
- ❑ **Ejemplos de Algoritmos en RL**
- ❑ Robótica: Expert Rules hasta End-to-end
- ❑ Conclusiones y recursos

k-armed Bandit: ¡un entorno sin estado!

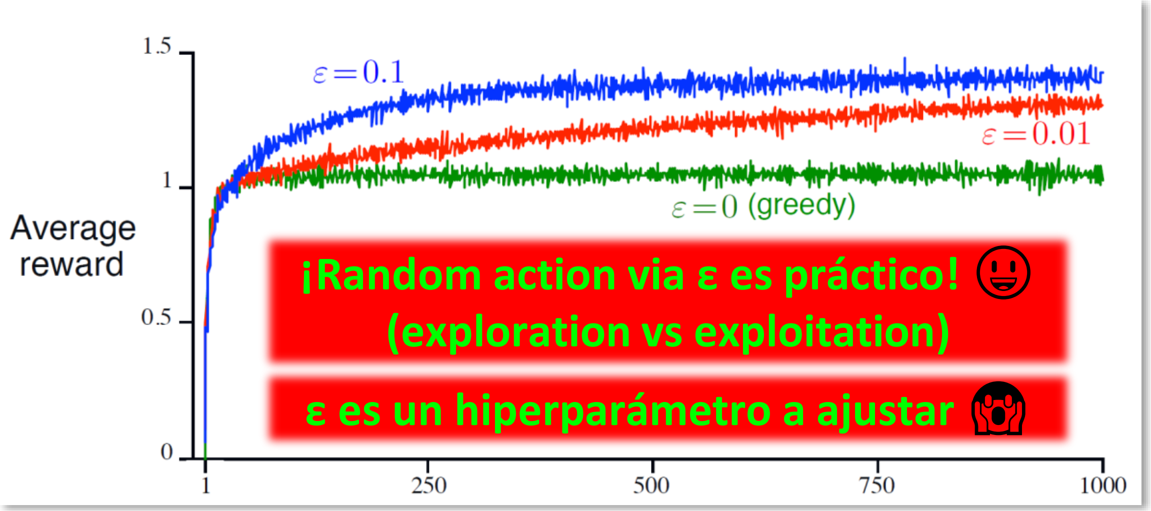
k acciones (c/u proporciona r correspondiente)



k-armed Bandit: una π solución vía Q

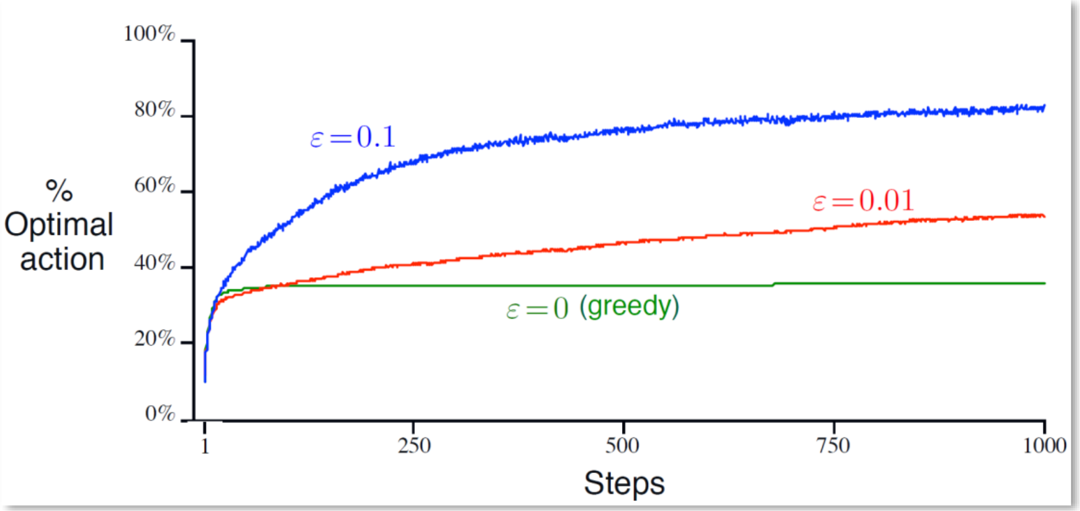
Caso particular: Q(s,a) = Q(a)

Algoritmo: $Q(a) = \dfrac{\Sigma\ r \text{ habiendo seleccionado } a}{\#\text{ veces seleccionado } a}$



¡Random action via ε es práctico! 😃
(exploration vs exploitation)

ε es un hiperparámetro a ajustar 😱



Sutton, Barto, "Reinforcement Learning: An Introduction", 2018

Un algoritmo muy conocido: Q-Learning

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $\boxed{Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]}$
    $S \leftarrow S'$
  until S is terminal

Sutton, Barto, "Reinforcement Learning: An Introduction", 2018

@jgvictores

Un algoritmo muy conocido: Q-Learning

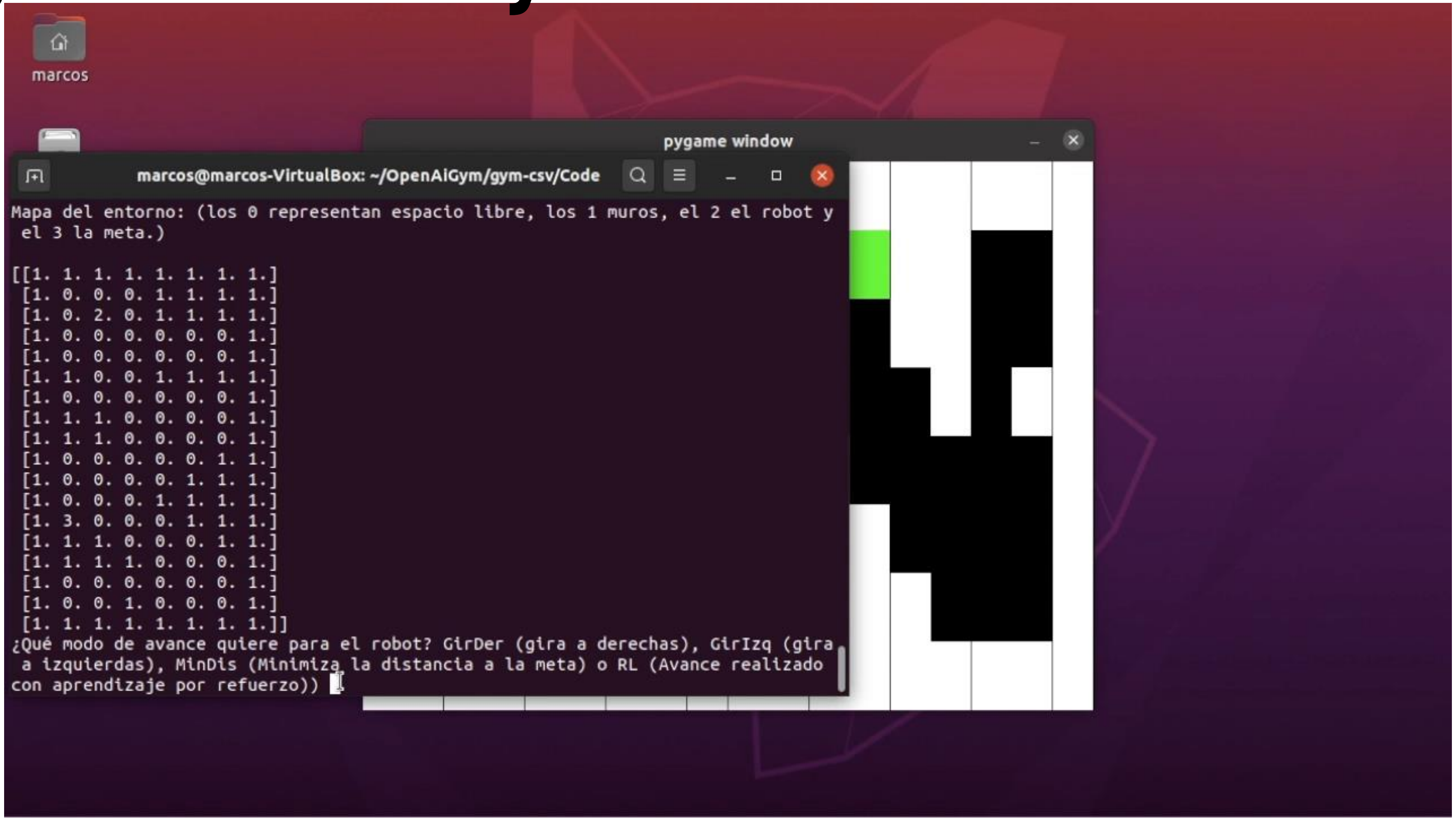
$$Q(S,A) \leftarrow Q(S,A) + \alpha \left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$$

Es un método tabular, s y a valores discretos.

Incorpora hiperparámetro α (learning rate), que se puede ajustar para optimizar en *convergencia* o en *rapidez de aprendizaje*.

```
Q[s,a] = Q[s,a] + alpha*(r + gamma*np.max(Q[s1,:]) - Q[s,a])
```

Un algoritmo muy conocido: Q-Learning

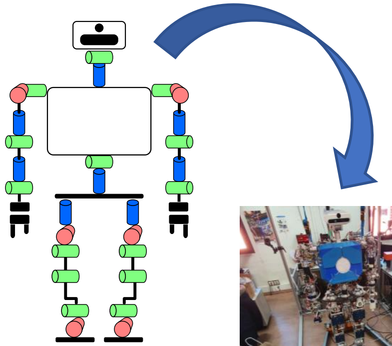


Algoritmos "Deep Reinforcement Learning"

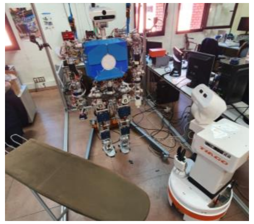
Mnih (2015-): Deep Q-Network (DQN) for Atari

Levine (2015-): End-to-End Training of Deep Visuomotor Policies

entrada = estado (imagen, pose…)



Agente



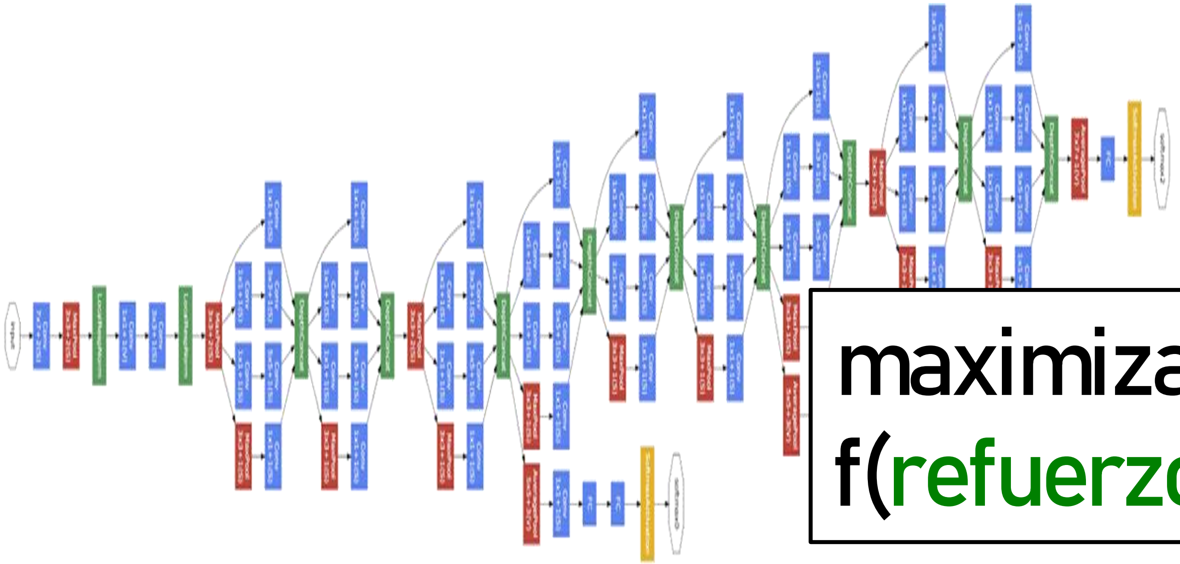
observa estado

acción 1

nuevo estado' y refuerzo'

acción 2

nuevo estado'' y refuerzo''



maximizar f(refuerzo)

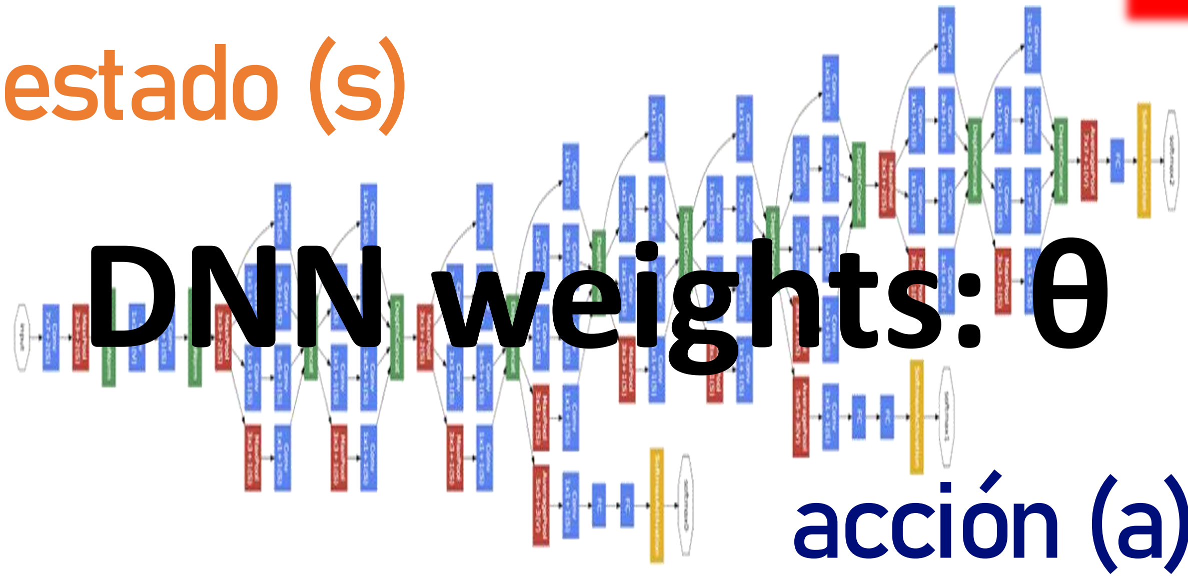
salida = π/V/Q

para acción = π(state)

Algoritmos "Deep Reinforcement Learning"

Modelado directo de π (policy gradient, p.ej. REINFORCE)

estado (s)



DNN weights: θ

acción (a)

Probabilística

π_θ(**acción** | **estado**)

Supervised Learning (DL)

$$\nabla_\theta J_{\mathrm{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right)$$

Reinforcement Learning (DL)

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

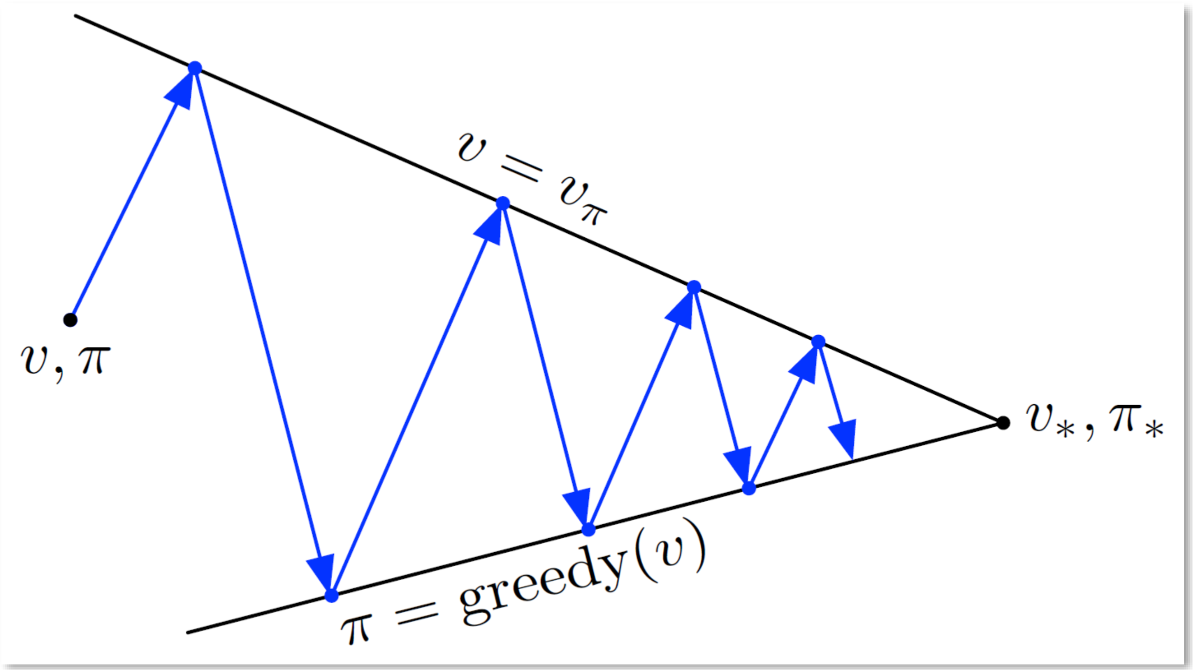
Supervised & Reinforcement Learning Objective Eqs from "Berkeley CS 285 – Fall 2020 (Instructor: Sergey Levine)"

Más posibilidades dentro de RL

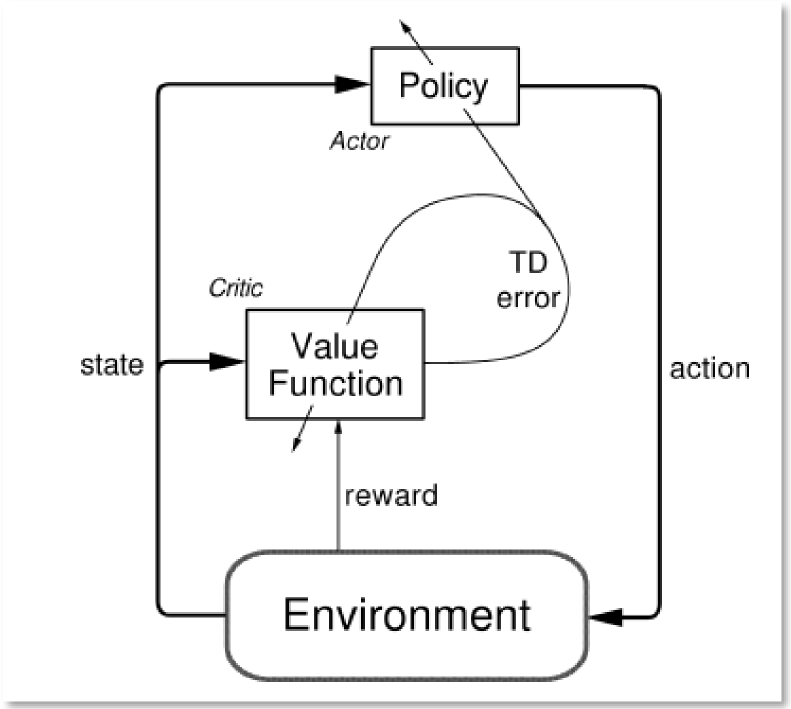
- En “on-policy”, el agente mejora sus parámetros a través de su interacción.
- En “off-policy”, es capaz de mejorar sus parámetros en base a los datos de interacción de otro agente (herramienta: “importance sampling”).

Más posibilidades dentro de RL

Modelos simultáneos: Generalized Policy Iteration (GPI)



Actor-Critic: A3C, TRPO, SAC…



Más posibilidades dentro de RL (Avanzado)

Inverse Reinforcement Learning

Metalearning

AutoML

…

…

…

Robótica inteligente mediante Reinforcement Learning

- ❑ Reinforcement Learning dentro de la IA
- ❑ Conceptos Principales en RL
- ❑ Ejemplos de Algoritmos en RL
- ❑ **Robótica: Expert Rules hasta End-to-end**
- ❑ **Conclusiones y recursos**

Robótica: Expert Rules hasta End-to-End

- Component Based Software Engineering (CBSE)
  - Sistemas Expertos (Rule-Based: IF-ELSE)
  - Planificación, Cinemática, Control
  - Learning w/hand-crafted Features
  - All Learned Features
- End-to-End (p.ej. Via DRL)

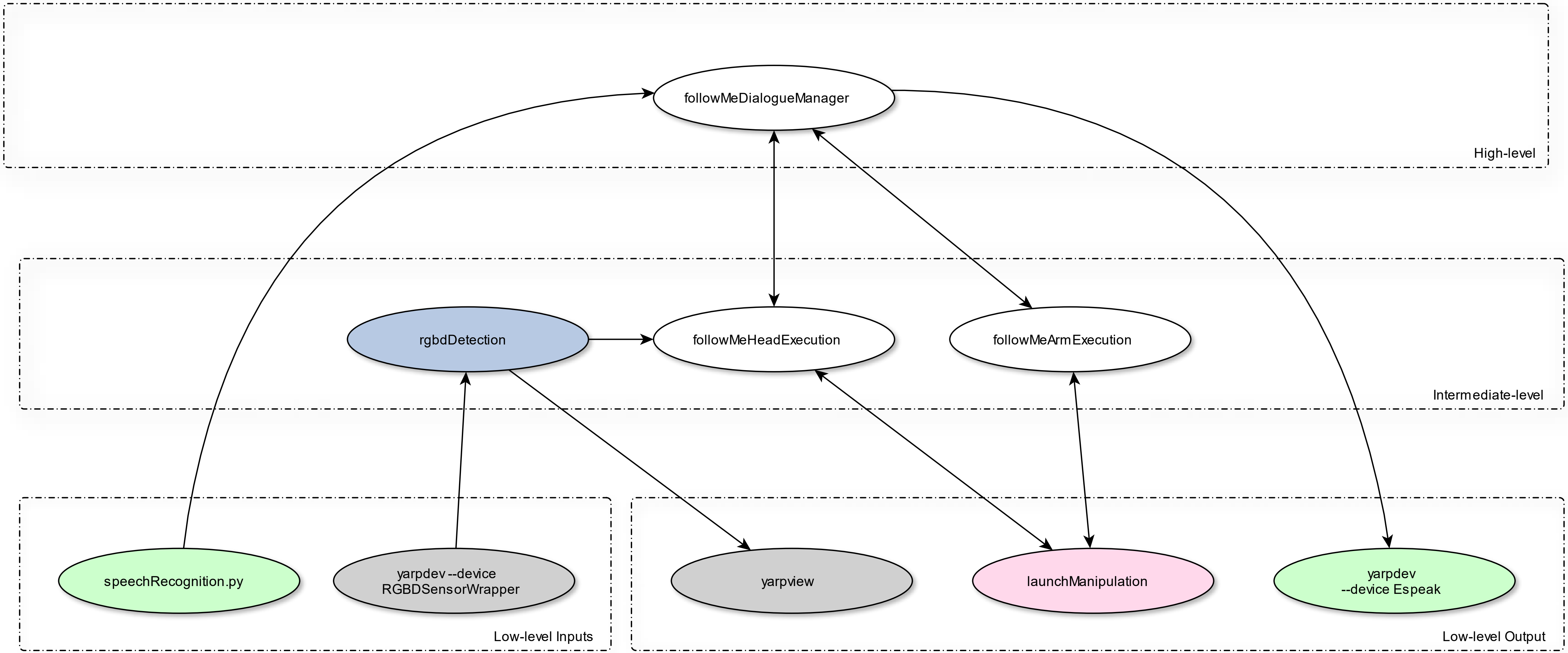
Hyperparameters
- Hand-crafted
- Automated search

Expert (Coding)

"generic/ agnostic" Algorithms

Learning (Data+Model +Algorthms)

Robótica: CBSE (rules & generic & learning)



Robótica: Expert Rules hasta End-to-End

- Component Based Software Engineering (CBSE)
  - Sistemas Expertos (Rule-Based: IF-ELSE)
  - Planificación, Cinemática, Control
  - Learning w/hand-crafted Features
  - All Learned Features
- End-to-End (p.ej. Via DRL)

Hyperparameters
- Hand-crafted
- Automated search

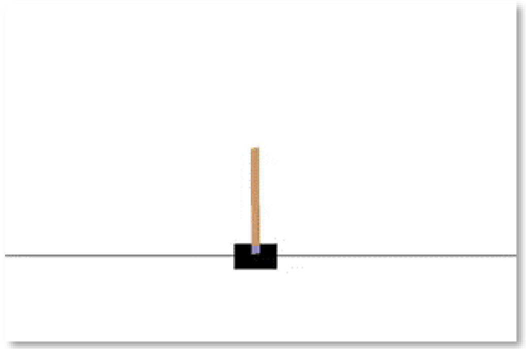
Expert (Coding)

"generic/ agnostic" Algorithms

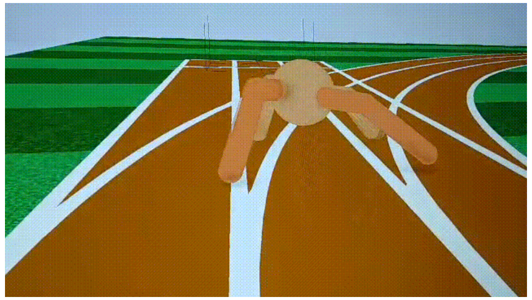
Learning (Data+Model +Algorthms)

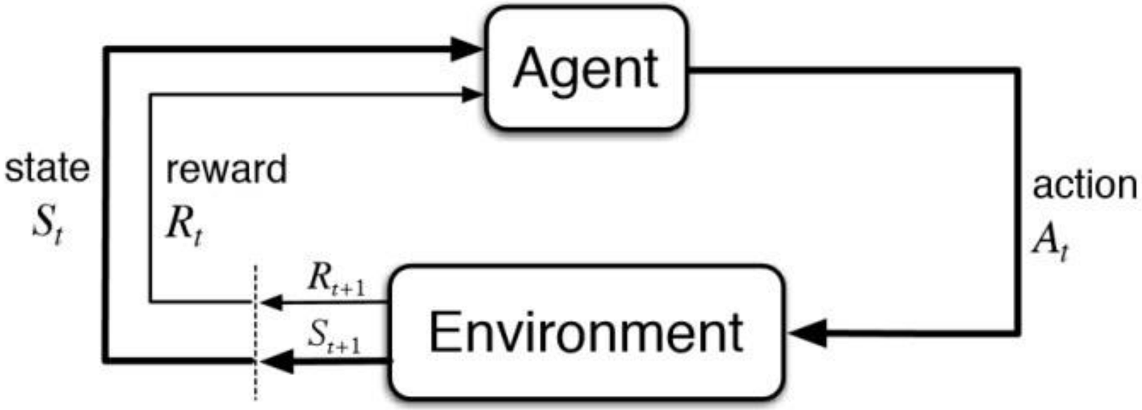
Recursos (software): OpenAI Gym

- Python API: Utilizar entornos [1]



```
import gym
env = gym.make('CartPole-v0')
env.reset()
for _ in range(1000):
    env.render()
    env.step(env.action_space.sample()) # take a random action
```

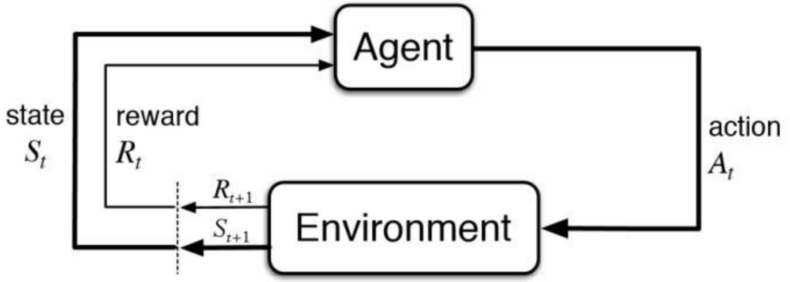




Recursos (software): OpenAI Gym

- Python API: Utilizar entornos [1]

```
import gym
env = gym.make('CartPole-v0')
for i_episode in range(20):
    observation = env.reset()
    for t in range(100):
        env.render()
        print(observation)
        action = env.action_space.sample()
        observation, reward, done, info = env.step(action)
        if done:
            print("Episode finished after {} timesteps".format(t+1))
            break
```



Recursos (software): OpenAI Gym

- **Python API: Crear entornos [1]**

```
import gym
from gym import error, spaces, utils
from gym.utils import seeding

class FooEnv(gym.Env):
  metadata = {'render.modes': ['human']}

  def __init__(self):
    ...
  def step(self, action):
    ...
  def reset(self):
    ...
  def render(self, mode='human'):
    ...
  def close(self):
    ...
```

Recursos (software): OpenAI Gym

- **Python API: Crear entornos [1]**

```python
class DiscreteEnv(Env):

    """
    Has the following members
    - nS: number of states
    - nA: number of actions
    - P: transitions (*)
    - isd: initial state distribution (**)

    (*) dictionary dict of dicts of lists, where
      P[s][a] == [(probability, nextstate, reward, done), ...]
    (**) list or array of length nS


    """
```

```python
def __init__(self, nS, nA, P, isd):
    self.P = P
    self.isd = isd
    self.lastaction = None # for rendering
    self.nS = nS
    self.nA = nA

    self.action_space = spaces.Discrete(self.nA)
    self.observation_space = spaces.Discrete(self.nS)

    self.seed()
    self.s = categorical_sample(self.isd, self.np_random)
    self.lastaction=None

def seed(self, seed=None):
    self.np_random, seed = seeding.np_random(seed)
    return [seed]

def reset(self):
    self.s = categorical_sample(self.isd, self.np_random)
    self.lastaction = None
    return self.s

def step(self, a):
    transitions = self.P[self.s][a]
    i = categorical_sample([t[0] for t in transitions], self.np_random)
    p, s, r, d= transitions[i]
    self.s = s
    self.lastaction = a
    return (s, r, d, {"prob" : p})
```

Recursos (software):

1. https://gym.openai.com
2. https://github.com/openai/gym
3. https://github.com/openai/gym/blob/master/docs/environments.md#third-party-environments
4. https://github.com/jgvictores/awesome-deep-reinforcement-learning#rldrl-gyms
5. https://github.com/hill-a/stable-baselines

Recursos (libros, cursos, proyectos)

Guili, “Deep Learning with TensorFlow 2 and Keras”, 2019

https://www.coursera.org/specializations/deep-learning

Sutton, Barto, “Reinforcement Learning: An Introduction”, 2018

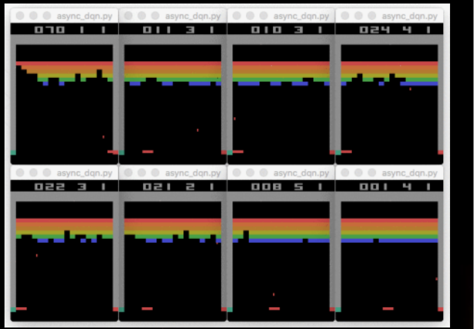
https://www.coursera.org/specializations/reinforcement-learning

http://rail.eecs.berkeley.edu/deeprlcourse

Máster Universitario en Robótica y Automatización (UC3M)
(p.ej. Planificación, Aprendizaje, Simuladores…)

ALMA: Human Centric Algebraic Machine Learning (H2020-EIC-FETPROACT-2019)

https://github.com/jgvictores/awesome-deep-reinforcement-learning



Robótica inteligente mediante Reinforcement Learning

Juan G Victores
RoboticsLab – Universidad Carlos III de Madrid

SUMMER SCHOOL
CEA Comité Español de Automática
ALMA

@jgvictores
http://roboticslab.uc3m.es
jcgvicto@ing.uc3m.es