Acerca de

Reinforcement Learning (RL)

Evelyn Coronel

Aprendizaje Profundo y Redes Neuronales Artificiales

27 de Noviembre del 2020

- (Mi) Motivación
- ¿Qué es RL?
- Overview de un time-step
- Intro: Policy, Value Function
- Value Function y Q-Learning
- Ejemplo: DQN
- Problemas (?)

¿Por qué me gusta tanto?

AlphaGo (Deep Mind):

Juega Go (juego de mesa)

OpenAI Five :

Juega Dota 2 (MOBA) (Como LoL)

AlphaStar (Deep Mind):

Juega Starcraft II

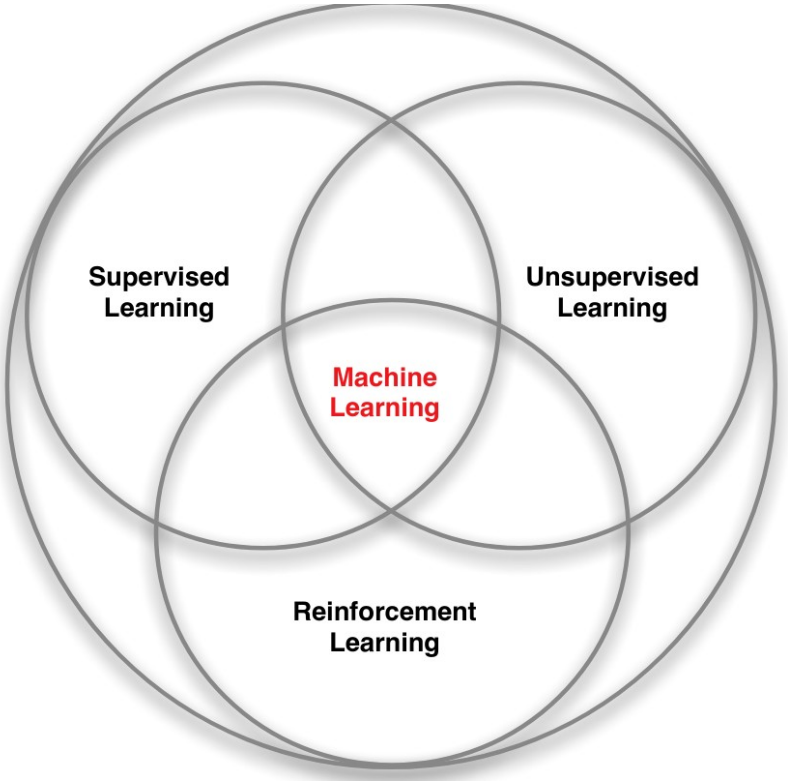
(real-time strategy game)

Multi-Agent (OpenAI):

máquinas jugando las escondidas.

(*El video* es genial)

¿Qué es RL?

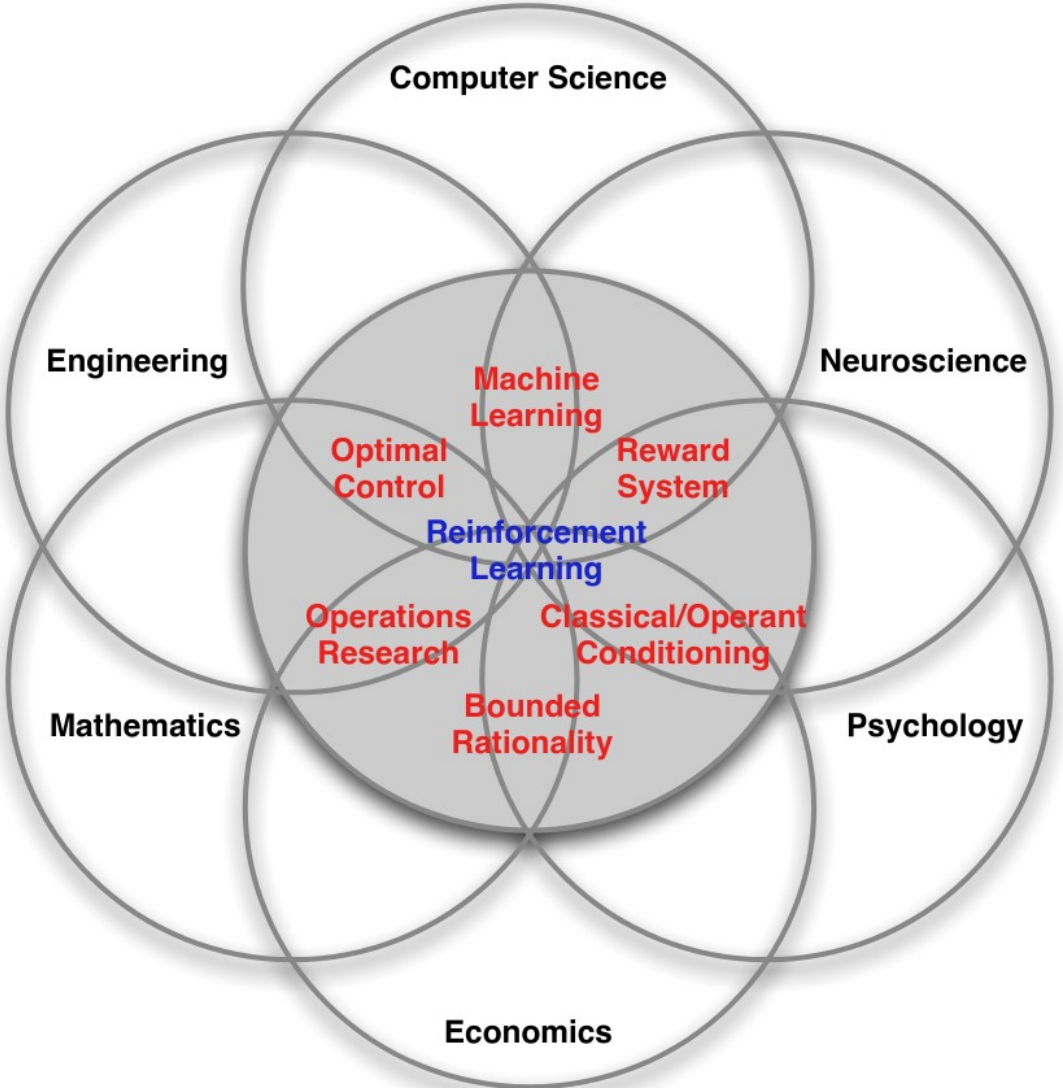


Es una de las distintas formas de enfrentar un problema de ML.

La diferencia con otros paradigmas es que no hay ningún supervisor, sólo una señal de recompensa o castigo (reward): Trial and Error.

Los datos tienen una correlación temporal: no es óptimo para datos independientes en el tiempo.

Las decisiones que toma el programa afecta los datos que recibe. Online Learning.

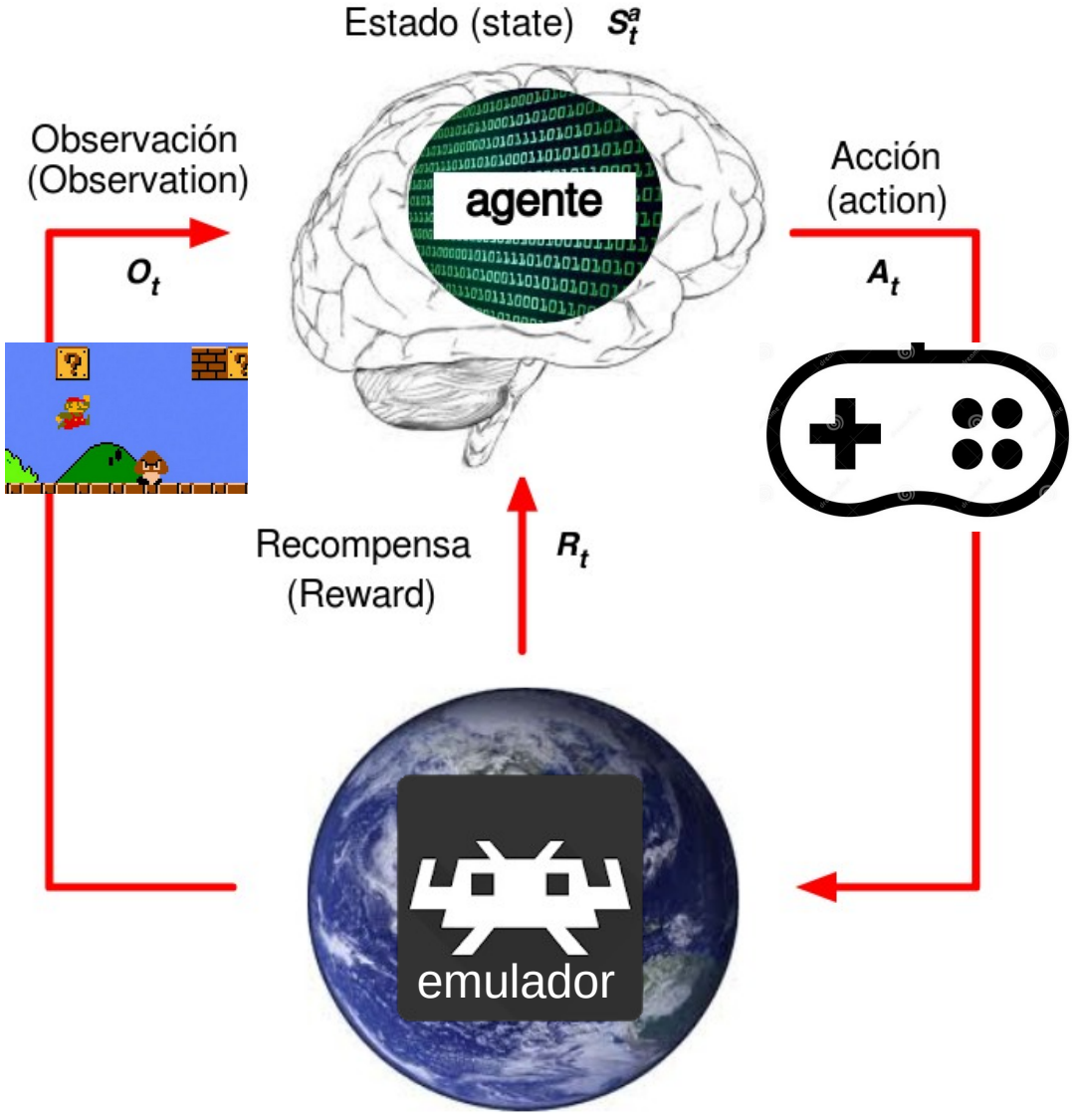


¿Por qué lo usaríamos?

Ramas de distintas áreas estudian el mismo problema:

Como tomar de decisiones

Control: Forma óptima de controlar un sistema.
Neuro: Dopamina!
Psicología: Pavlov
Economía: Como la gente toma decisiones y Teoría de Juegos (según wikipedia)



Ambiente: La "cosa" que nos da toda la información. Su estado es un MS* (eg. pantalla)

Observación: La información que toma el agente del ambiente.

Agente: La red que hicimos.

Estado (del agente): Información que sirve para la siguiente acción. MS (o no si elegimos mal).

Reward: ***positivo*** *$* *o* ***negativo*** *$*

*MS==Markov State

Policy π (Política): el comportamiento del agente. Maximiza el valor de expectación del reward. Puede ser random o determinista. Decide la **acción**.

Policy Based

Actor Critic

Value Function (Función de valor): ¿cuánto reward espero obtener?

$$v_\pi(s) = E_\pi(R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} \ldots | S_t = s) \text{ [MRP*]}$$

Value Based

Model: Una representación del ambiente. Be smart. Podemos predecir que va a pasar en el ambiente o el próximo reward. *No es necesario para hacer RL.*

Model Free

* Markov Reward Process

Policy y Value Function

(teorema [MDP]) Ambos tiene una función óptima que maximiza el reward, llamémoslos π^* y $v_{\pi^*} = v^*$.

Puedo armar π^* a partir v^*, así que nos vamos a concentrar en v_π

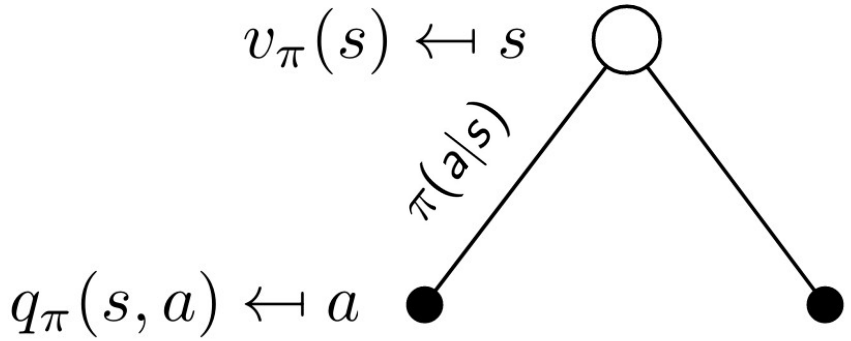
Existen dos tipos de value functions:

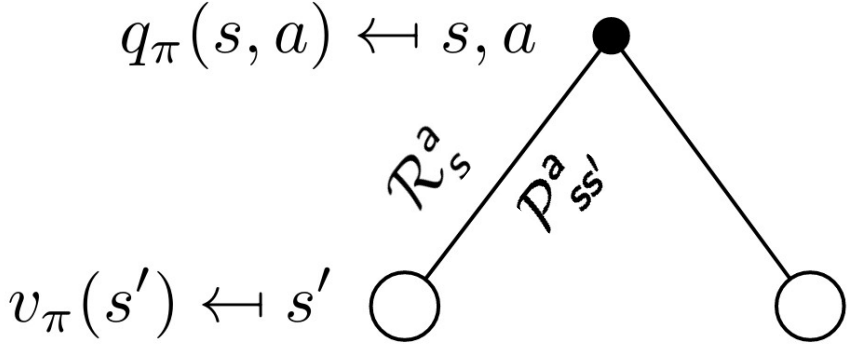
1- State value function: $v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s \right]$ (V-values)

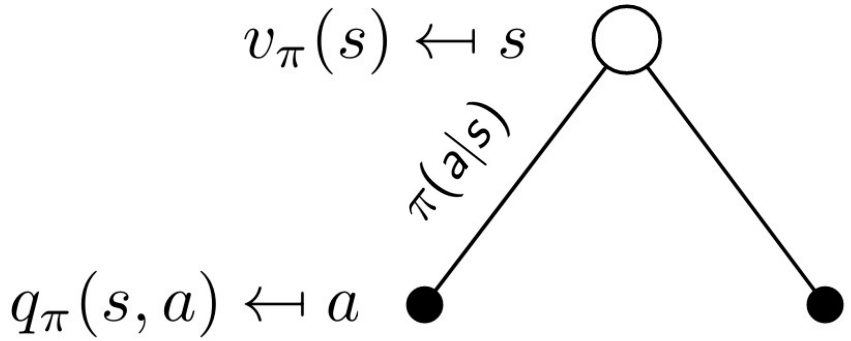
2- Action value function: $q_\pi(s, a) = \mathbb{E}_\pi \left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a \right]$
(Q-values)

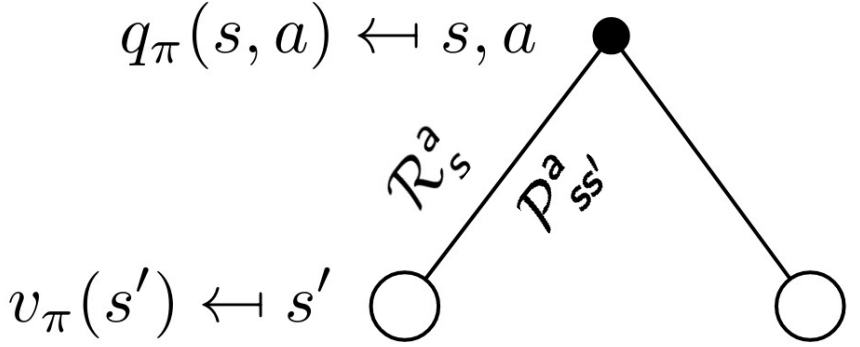
Estas dos funciones están relacionadas entre sí

A su vez, puedo armarme v* a partir de q* y obtener una ecuación para resolver v*/q*


$v_\pi(s) \leftarrow\!\shortmid\ s$
$\pi(a|s)$
$q_\pi(s,a) \leftarrow\!\shortmid\ a$


$q_\pi(s,a) \leftarrow\!\shortmid\ s,a$
$\mathcal{R}^a_s$
$\mathcal{P}^a_{ss'}$
$v_\pi(s') \leftarrow\!\shortmid\ s'$





$$v_*(s) = \max_a q_*(s,a)$$

$$q_*(s,a) = \mathcal{R}^a_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} v_*(s')$$

$$q_*(s,a) = \mathcal{R}^a_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \max_{a'} q_*(s',a')$$

Q-Learning

$$q_{t+1}(s,a) \approx \mathcal{R}_s^a + \gamma \max_{a'} q_t(s',a')$$

$$q_{t+1}(s,a) \leftarrow (1-\alpha)\, q_t(s,a) + (\alpha)\left[\, \mathcal{R}_s^a + \gamma \max_{a'} q_t(s',a') \,\right]$$

$$\boxed{\mathcal{R}_s^a + \gamma \max_{a'} q_t(s',a') - q_t(s,a)} \longrightarrow$$

MSE, Huber

Problemas

Sabemos características sobre el ambiente (RL)

vs.

Tenemos un modelo del ambiente (Planning)

Exploración vs Explotación

Predicción (evaluar) vs. Control (optimizar)

$100

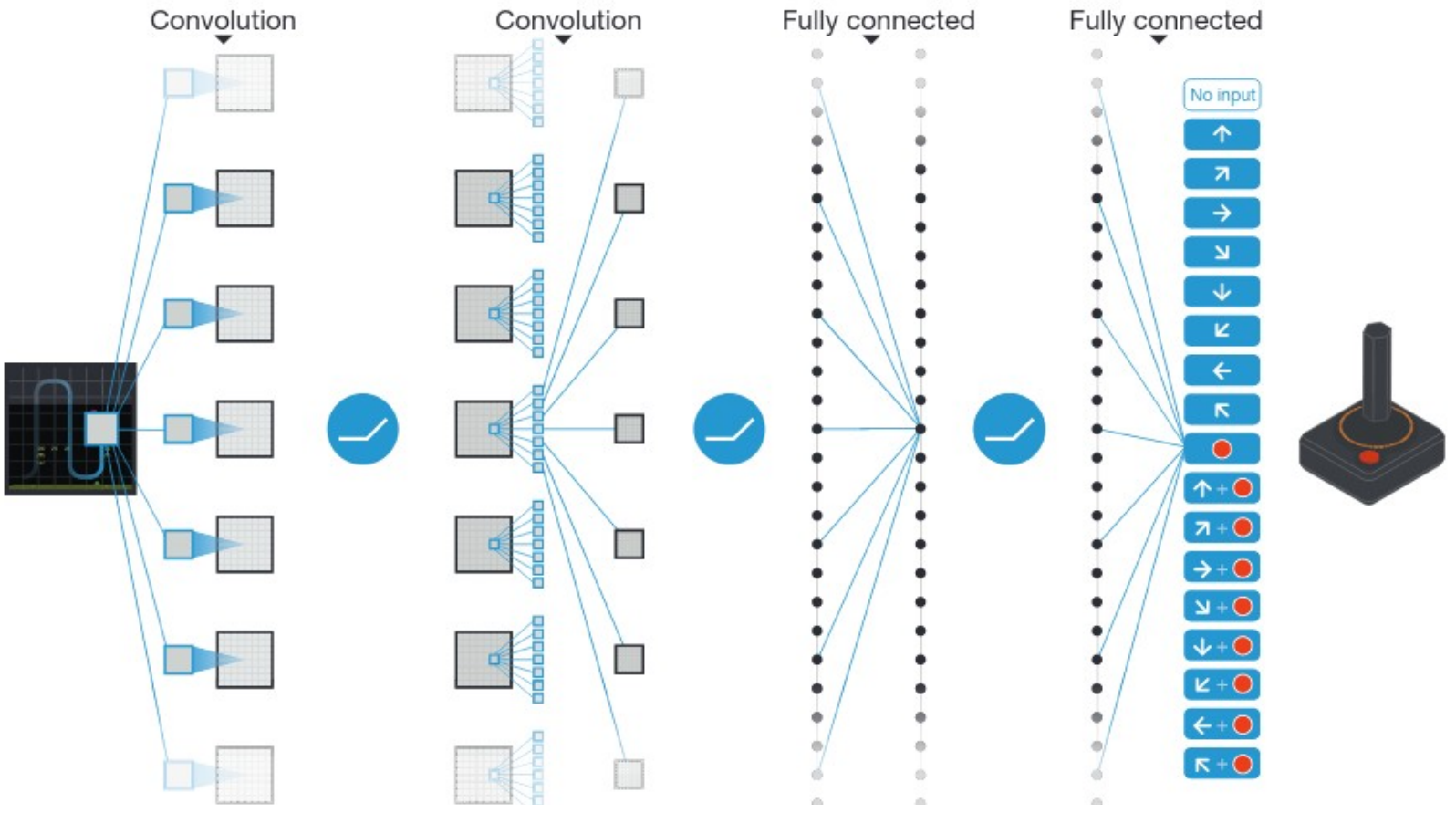
P: 1

$200

P: 0.75

DQN (Deep Q-Learning Network)

(DeepMind 2015)



~Extras~

Más info en code/README.txt

CS188 Berkeley Intro to AI

Página del práctico: Proyecto 3

Página de la materia: CS188

Clases del 2013 (me gustan más estas que las más nuevas)

Files you'll edit:	
`valueIterationAgents.py`	A value iteration agent for solving known MDPs.
`qlearningAgents.py`	Q-learning agents for Gridworld, Crawler and Pacman.
`analysis.py`	A file to put your answers to questions given in the project.

VizDoom: Github y Página

Markov: MS, MRP, MDP

La teoría estadística detrás de RL:

- Markov State: $\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, ..., S_t]$
- Markov Process (Cadenas de Markov): S, P
- Markov Reward Process: S, P, R, γ
- Markov Decision Process : S, $P^{(A)}$,A, $R^{(A)}$, γ

S: Conjunto de estados, P: Matriz de Prob., R: Reward, A: Conj. de Acciones