



# **Visión por Computador II**

**CEAI, FIUBA**

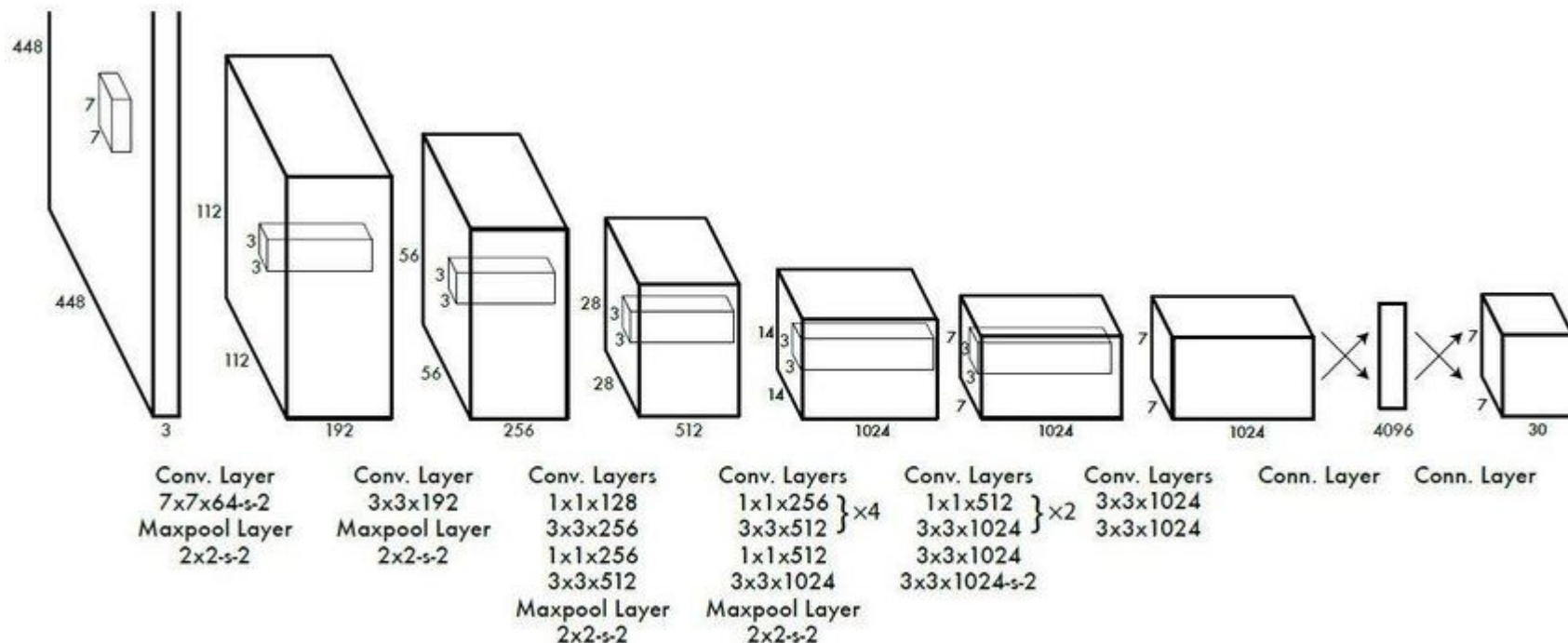
Profesor: Javier Kreiner, [javkrei@gmail.com](mailto:javkrei@gmail.com)



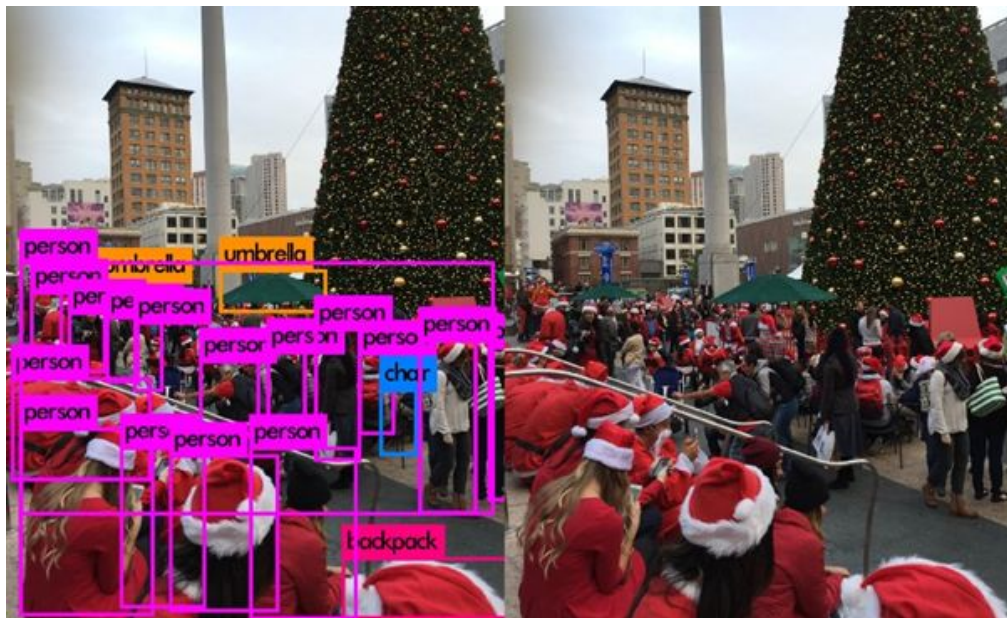
## Quinta clase:

- Usar algoritmo YOLO
  - ejemplo de programación
- Introducción a Reinforcement Learning

# Arquitectura de YOLO v1



# Limitaciones de YOLO v1



# Limitaciones de YOLO v1



- El algoritmo original utiliza una grilla de 7x7
- Cada celda puede reconocer sólo un objeto, entonces el máximo número de objetos es 49
- Si una celda contiene más de uno, el modelo no va a poder detectar todos, y ese es el problema de detección objetos cercanos que tiene YOLO
- Además tiene un error de localización relativamente alto

# YOLO v2

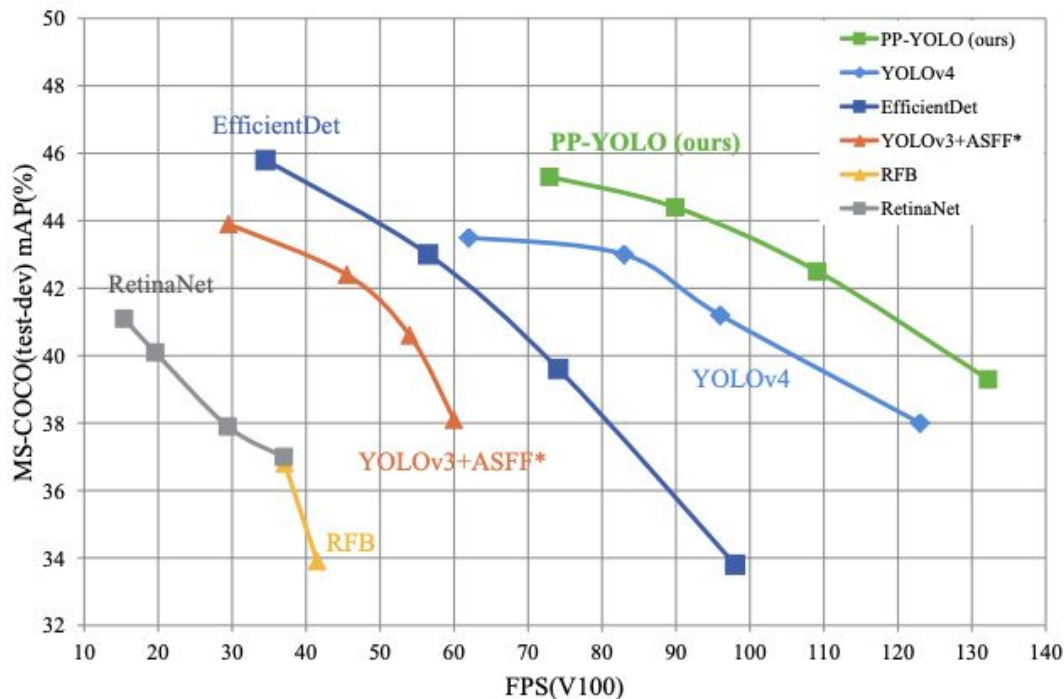


## Mejoras:

- Batch Normalization
- Clasificador de mayor resolución: el tamaño de la entrada fue aumentado de 224x224 a 448x448 (en un paso de tuning del entrenamiento)
- Anchor boxes: permiten detectar varios objetos en la misma celda. Se eligen utilizando un algoritmo de k-meas clustering
- se aumenta la grilla a 13x13, lo que permite identificar y localizar objetos más pequeños
- entrenamiento multi-escala: si YOLO es entrenado con imágenes pequeñas de un objeto tiene dificultad de detectarlo cuando el objeto aparece más grande en la imagen. Para eso se entrena con imágenes aleatorias con diferentes dimensiones. Como el modelo sólo tiene capas convolucionales y de pooling la entrada puede ser redimensionada 'on the fly'. Como resultado la red es más robusta para detectar objetos en diferentes resoluciones
- Darknet 19: Utiliza la arquitectura Darknet 19 con 19 capas convolucionales, 5 max pool y softmax para clasificación. La arquitectura es muy veloz.

# Más YOLOs vs lo demás, estado del arte

- Competencia por velocidad de detección y mAP





# Ejemplo

- colab:  
<https://colab.research.google.com/drive/1MeZzcfUMGfaAmslwPAxluN5ISq1ZR0Vg?usp=sharing>
- Otros repos de python y tf2:
  - yolo v4: <https://github.com/emadbocctorx/yolo-tf2>
  - yolo v3: <https://github.com/zzh8829/yolov3-tf2>
- Ejemplo de construcción de yolo v1 simple:
  - colab: [https://colab.research.google.com/drive/1PCqlip7wt3fZzm4vUtLUXGAqp\\_xD82hg?usp=sharing](https://colab.research.google.com/drive/1PCqlip7wt3fZzm4vUtLUXGAqp_xD82hg?usp=sharing)  
(obtenido de: <https://www.maskaravivek.com/post/yolov1/>)





# Competición de Metadata

- <https://metadata.fundacionsadosky.org.ar/competition/22/>
- Slack: [https://join.slack.com/t/desafios-agtech/shared\\_invite/zt-iu7s07km-slaD~P00PlhHjr6OgaDf3Q](https://join.slack.com/t/desafios-agtech/shared_invite/zt-iu7s07km-slaD~P00PlhHjr6OgaDf3Q)
- Winn: <https://bcrwinn.app.link/rw18caVjKab>
- <https://desafiosagtech.com/>



# Fuentes

## Libros:

- **Reinforcement Learning: An Introduction.** Richard S. Sutton and Andrew G. Barto, Second Edition, MIT Press, Cambridge, MA, 2018 (<http://incompleteideas.net/book/the-book-2nd.html>)
- **Algorithms for Reinforcement Learning.** Csaba Szepesvári, Synthesis Lectures On Artificial Intelligence And Machine Learning #9, Morgan & Claypool publishers, 2010



# Fuentes

Cursos online:

- UCL RL course: <http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html>
- [https://github.com/yandexdataschool/Practical\\_RL/tree/coursera](https://github.com/yandexdataschool/Practical_RL/tree/coursera)
- <https://yadi.sk/d/loPpY45J3EAYfU>



# Plan de la parte de Aprendizaje Reforzado

- Qué es Aprendizaje Reforzado (Reinforcement Learning)
- Relación con otros campos
- Diferencias con otros paradigmas de ML
- Características propias de Aprendizaje Reforzado
- Definición del problema
- Ejemplos
- Review de matemática
- Ejercicios de programación
- Software
- Intro a Open AI Gym
- Bonus: Lecturas recomendadas



## Qué es Aprendizaje Reforzado

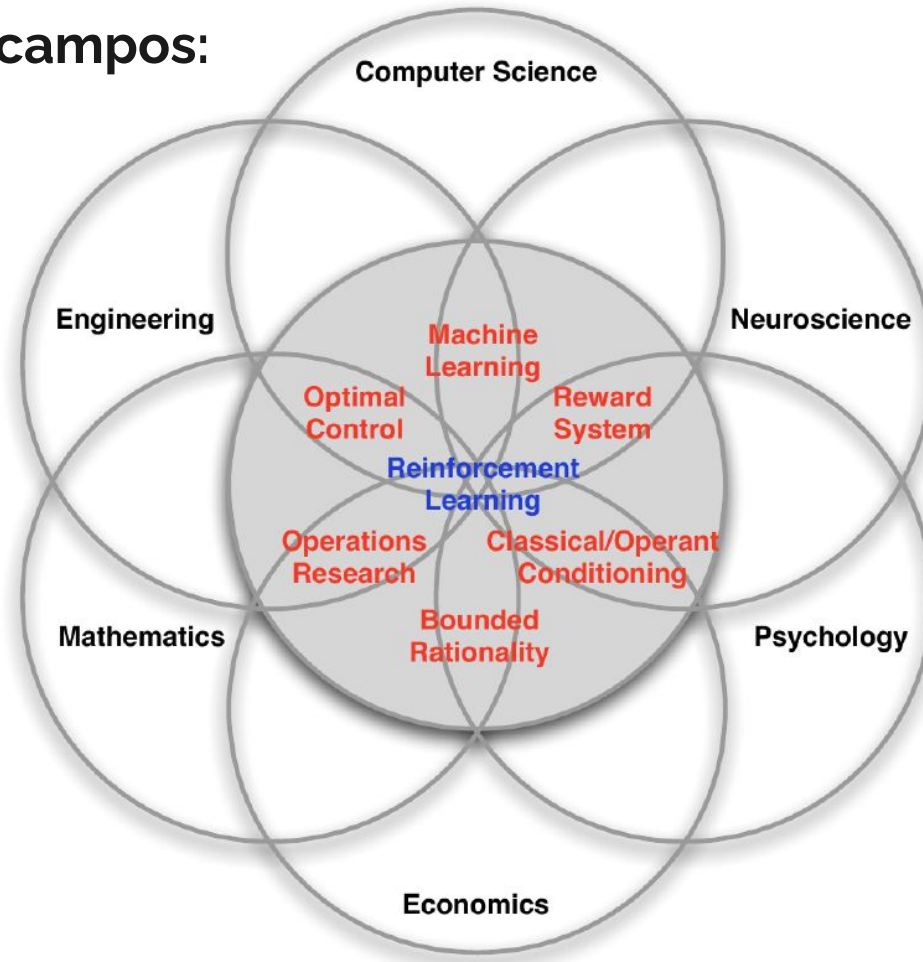
- El problema central: aprendizaje por prueba y error a través de la interacción con el ambiente
- El objetivo es aprender a 'qué hacer' -, o sea cómo mapear acciones a situaciones - para maximizar una señal de recompensa
- El nombre se refiere a un problema, una clase de métodos de solución, el campo de estudio del problema
- Al agente no se le dice lo que debe hacer, debe aprenderlo interactuando con el ambiente y recibiendo recompensas
- Se deben tomar decisiones secuenciales de manera óptima



## Qué es Aprendizaje Reforzado

- El agente obtiene información o ‘sensa’ u obtiene información a través de sus ‘sentidos’ o aparatos de medición
- A través de esa interacción con el ambiente el agente obtiene información y aprende sobre el comportamiento del mundo
- Con esa información decide qué acciones realizar en cada momento
- El comportamiento es guiado por un objetivo que se modela en forma de una recompensa numérica
- De nuevo, el objetivo la toma de decisiones secuenciales de manera óptima, esto es: maximizar la recompensa acumulada

## Relación con otros campos:





## Diferencias con otros paradigmas de ML, RL vs Aprendizaje supervisado

- No viene dado un dataset con inputs y targets
- No hay un 'supervisor', o sea input y targets, hay una señal de recompensa
- El feedback se recibe con retraso, no es instantáneo
- Las decisiones son secuenciales, los datos no son i.i.d.
- Las acciones del agente modifican los datos que va recibiendo



## Diferencias con otros paradigmas de ML, RL vs aprendizaje supervisado



| <b>Aprendizaje supervisado</b>                   | <b>Aprendizaje reforzado</b>   |
|--|--|
| Aprende a aproximar respuestas de referencia     | Aprende la estrategia óptima a través de prueba y error                        |
| Necesita un dataset con las respuestas correctas | Necesita el feedback de las propias acciones del agente                        |
| El modelo no afecta los datos de entrada         | El agente puede modificar sus propias observaciones (dependen de las acciones) |

## Diferencias con otros paradigmas de ML, RL vs Aprendizaje no supervisado



| <b>Aprendizaje no supervisado</b>                        | <b>Aprendizaje reforzado</b>   |
|--|--|
| Aprende la estructura subyacente de un conjunto de datos | Aprende la estrategia óptima a través de prueba y error                        |
| No hay feedback  | Necesita el feedback de las propias acciones del agente                        |
| El modelo no afecta los datos de entrada                 | El agente puede modificar sus propias observaciones (dependen de las acciones) |



## Aprendizaje Reforzado

- Considera el problema integral de un agente con un objetivo interactuando con el ambiente y aprendiendo a tomar decisiones
- En ese sentido tanto aprendizaje supervisado como no supervisado son en general parte de sistemas mayores y resuelven un subproblema
- El campo de RL desarrolla técnicas y busca principios generales de aprendizaje que funcionen en este setup general de interacción con el entorno
- Marco general: procesos de decisión markovianos



## Aprendizaje Reforzado

- Objetivo: aprender a tomar decisiones de tal manera de maximizar las recompensas futuras acumuladas
- Aspectos distintivos:
  - Las acciones pueden tener consecuencias de largo plazo
  - Las recompensas pueden llegar con retraso y ser esparsas
  - Puede ser bueno sacrificar recompensa ahora para obtener más en el futuro
  - El agente afecta el ambiente a medida que toma decisiones



# Explotación vs Exploración

- Uno de los elementos fundamentales es el dilema de exploración vs explotación
- Explotación significa tomar la acción que creemos dado nuestro modelo actual nos dará más recompensa
- Exploración es tomar acciones de tal manera de aprender más respecto al ambiente y las recompensas de cada acción
- La dificultad de equilibrar entre exploración y explotación es otro problema distintivo de aprendizaje reforzado que no aparece en otros campos de machine learning



## Aprendizaje Reforzado - puntos clave

- Aprendizaje a través de interacción con el ambiente
- Hay aleatoriedad en el comportamiento del mundo
- Comportamiento guiado por una señal de recompensa
- Equilibrio entre exploración y explotación
- El comportamiento afecta los datos que el agente va observando



## Videos de algunos ejemplos

- robot humanoide: <https://www.youtube.com/watch?v=No-JwwPbSLA>
- helicoptero: <https://www.youtube.com/watch?v=0JL04JJjocc>
- blackout: <https://www.youtube.com/watch?v=eG1Ed8PTJ18>
- space invaders: <https://www.youtube.com/watch?v=W2CAghUiofY>
- arquero robotico: <https://www.youtube.com/watch?v=CIF2SBVY-J0>



## Definición del problema en más detalle

- Un agente interactúa con el ambiente y recibe observaciones
- El agente elige una acción de un conjunto en cada paso
- El mundo responde cambiando de estado
- El agente recibe feedback en forma de una recompensa numérica y esta es la manera en que se va a guiar al agente a tomar comportamientos deseables (reward hypothesis)
- Al tomar acciones también modifica las partes del mundo que explora (“active learning”)
- El objetivo es aprender la política óptima, el mapa observación -> acción, de manera de maximizar la recompensa acumulada



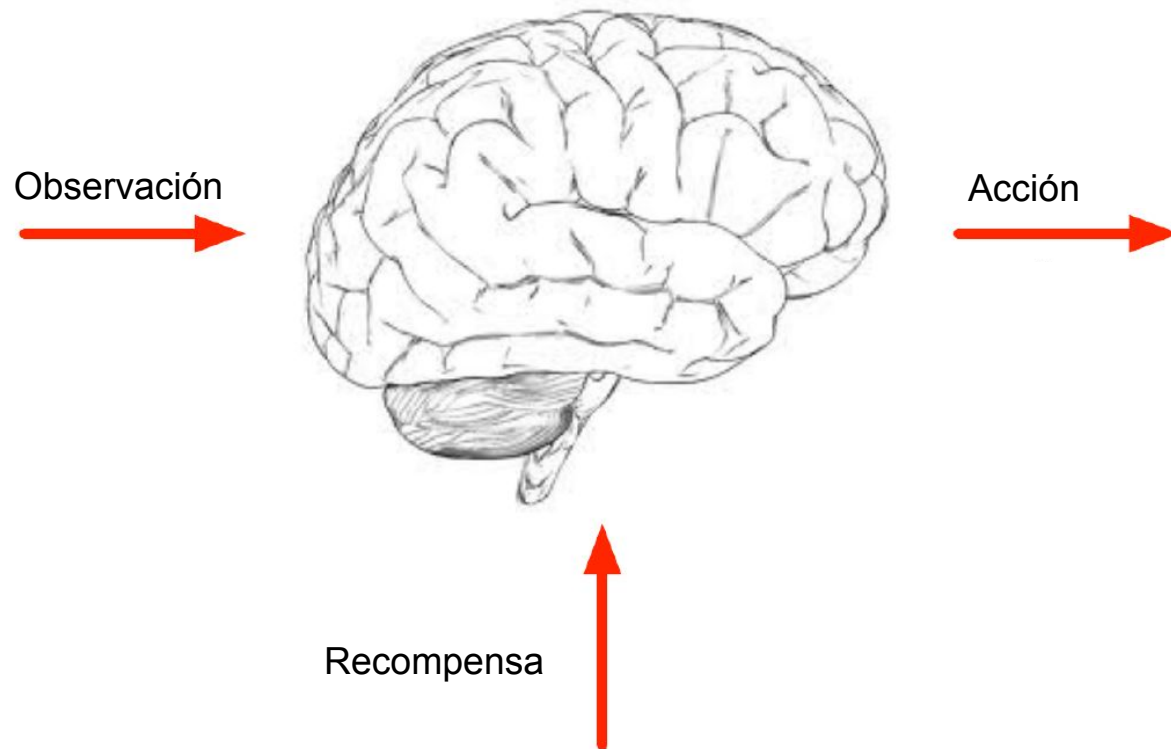


## Definición del problema - componentes

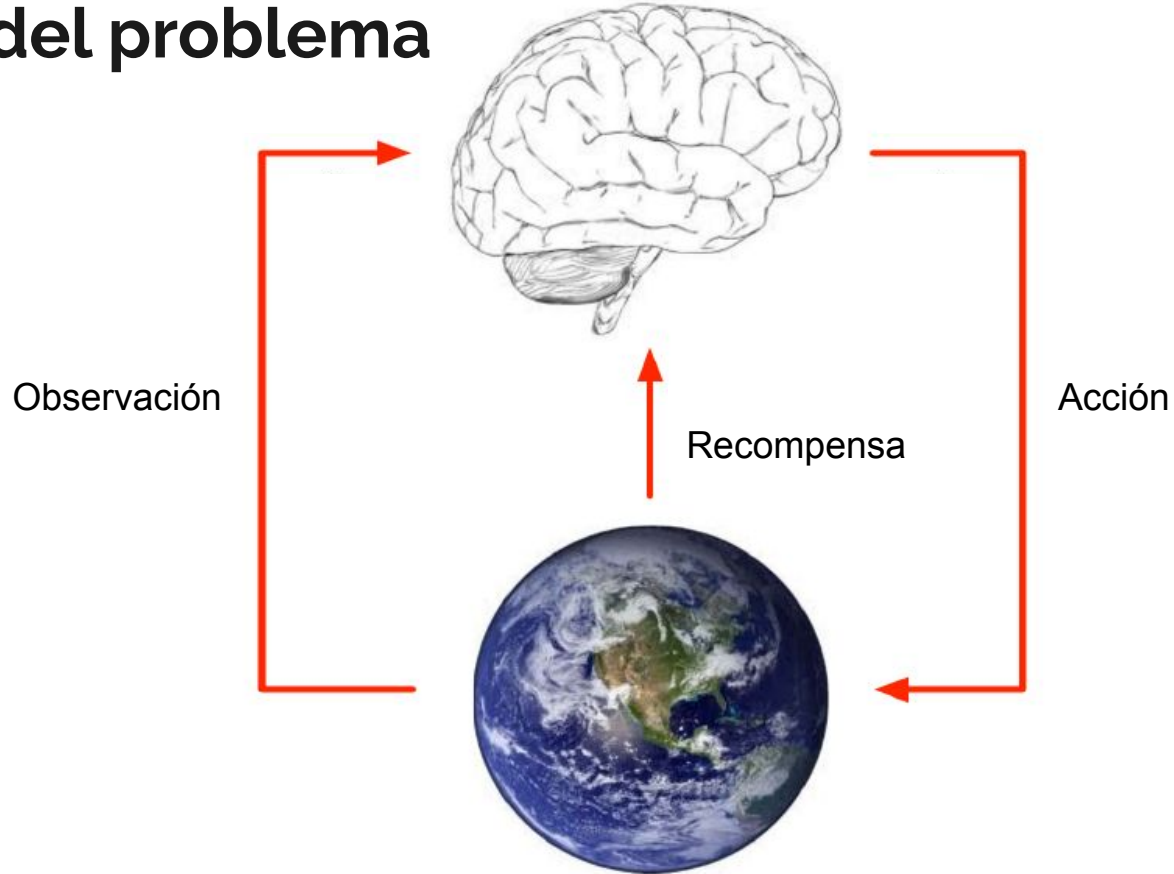
- El ambiente - estados, dinámica de transición
- Observaciones/mediciones del ambiente
- Acciones posibles en cada estado del ambiente
- Un agente
  - Una política (de acción): mapa de estados a acciones
  - Una señal de recompensa
  - Una función de valor (cuál es el valor de estar en cada estado/estado-acción)
  - Un modelo del ambiente (opcionalmente)

Objetivo: maximizar la esperanza recompensas acumuladas

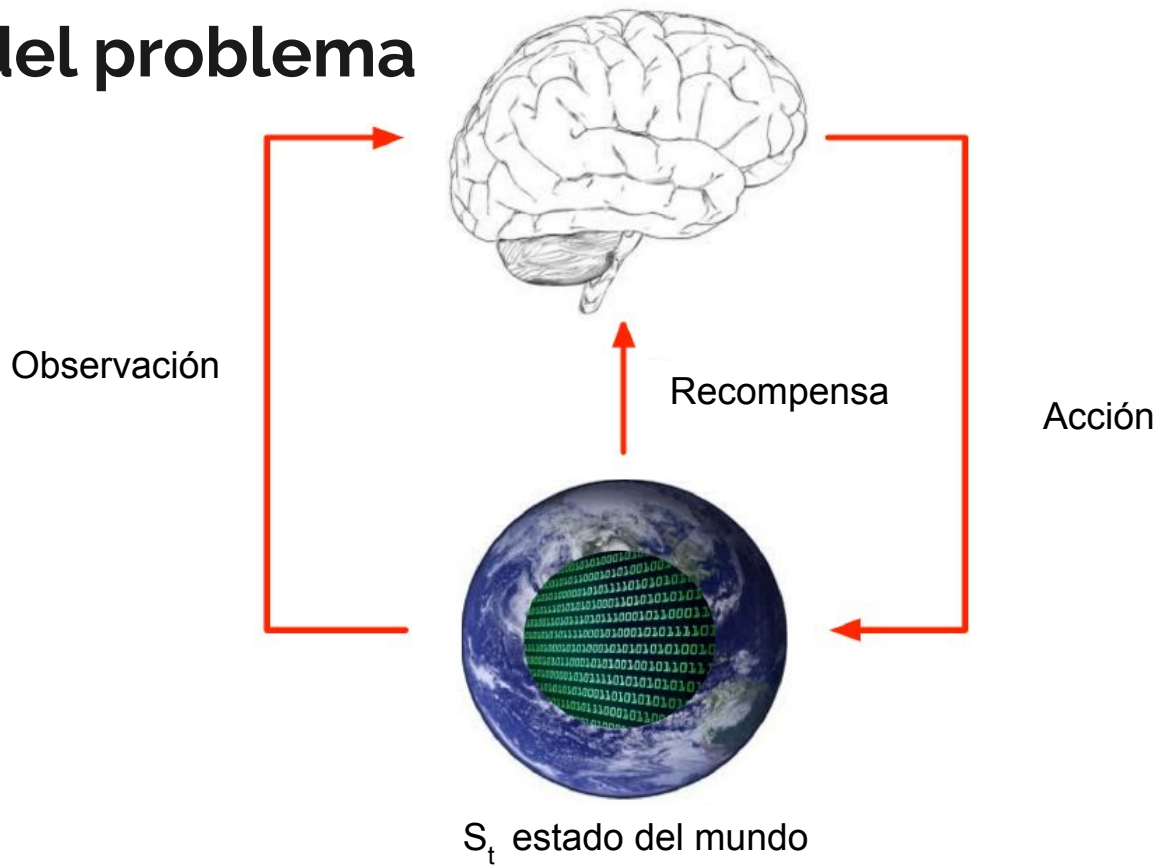
# Marco del problema



# Marco del problema

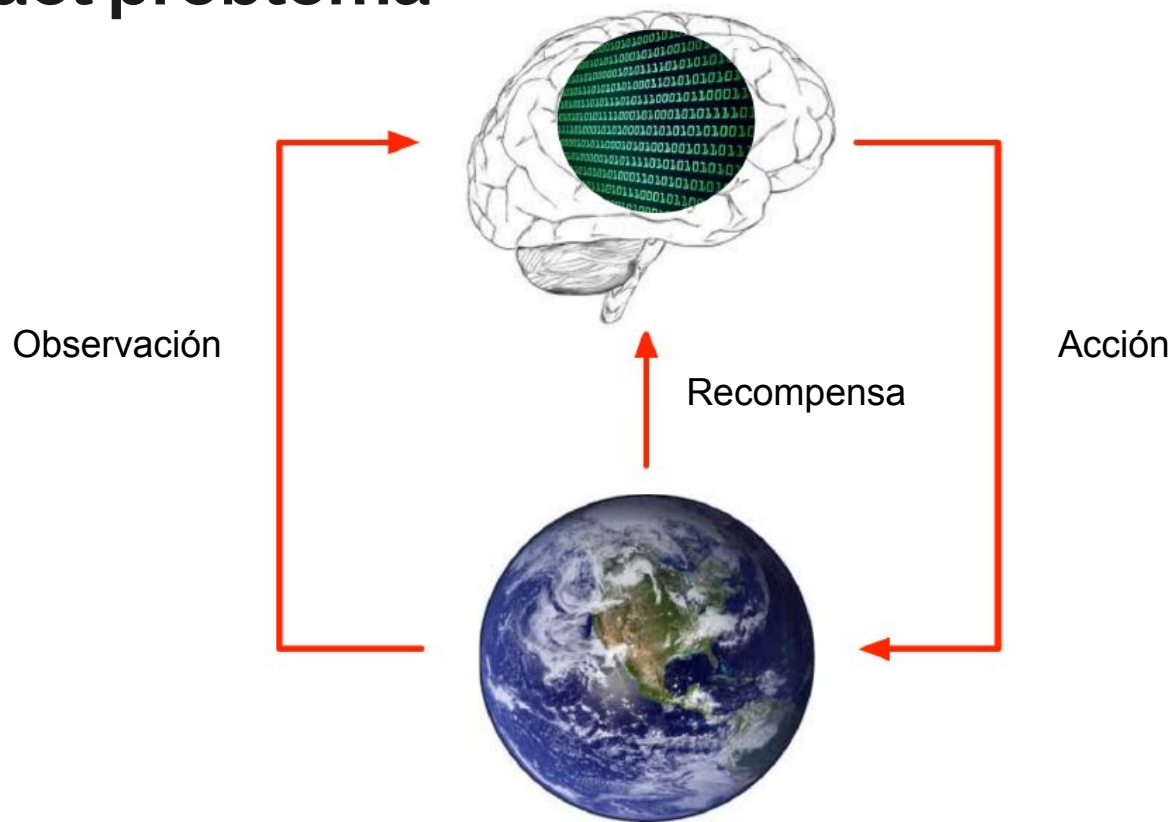


# Marco del problema



# Marco del problema

$S_t$  estado del agente





## Ejemplos (en cada uno de estos, recompensa, acciones, observaciones?)

- Un jugador de ajedrez, Teg, go, Backgammon, etc.
- Un helicóptero debe realizar piruetas
- Diseñar landing page para maximizar retención
- Chatbots con diversos objetivos: psicólogos, servicio al cliente
- Tratamiento médico personalizado
- Administración de una cartera de acciones
- Robots
- Asistentes de navegación

## Más ejemplos (en cada uno de estos, recompensa, acciones, observaciones?)



- Elegir ads para maximizar la ganancia
- Controlar una central de energía
- Jugar juegos de computadora mirando las imágenes
- Una gacela aprende a caminar/correr
- Preparar el desayuno
- El banco central toma decisiones en función de la economía
- Una bacteria se comporta en función del ambiente químico a su alrededor
- Vehículos autónomos



# Éxitos

- TD-Gammon (1992)
- Atari Games (DQN, 2015)
- AlphaGo(2015/2016)/AlphaGo Zero(2017)/AlphaZero(2017)
- Dota 2 (2018)
- Starcraft 2 (2019)
- Manipulación Robótica (2018)

<https://ai.googleblog.com/2018/06/scalable-deep-reinforcement-learning.html>

El campo no ha hecho un impacto económico significativo aún, pero está comenzando a ser usado en diferentes industrias (grandes oportunidades). Tal vez el problema más importante: necesita ingentes cantidades de datos. Leer <https://www.oreilly.com/ideas/practical-applications-of-reinforcement-learning-in-industry>





# Compañías utilizando Aprendizaje Reforzado

- Deepmind: AlphaGo, AlphaZero, Atari Games, <https://deepmind.com/>
- Trading algorítmico: Hihedge, <https://www.hihedge.com/>, <https://pit.ai/>
- Ambientes de cultivo controlables: Optimal Labs: <http://optimal.ag/>
- Aprendizaje de robots/vehículos autónomos: <http://covariant.ai/>,  
<https://www.latentlogic.com/>, <https://www.osaro.com/>, <http://prowler.io/>,  
<https://www.fanuc.com/>
- Análisis de datos: <http://intelligentlayer.com/>
- Chatbots: <https://rasa.com/>



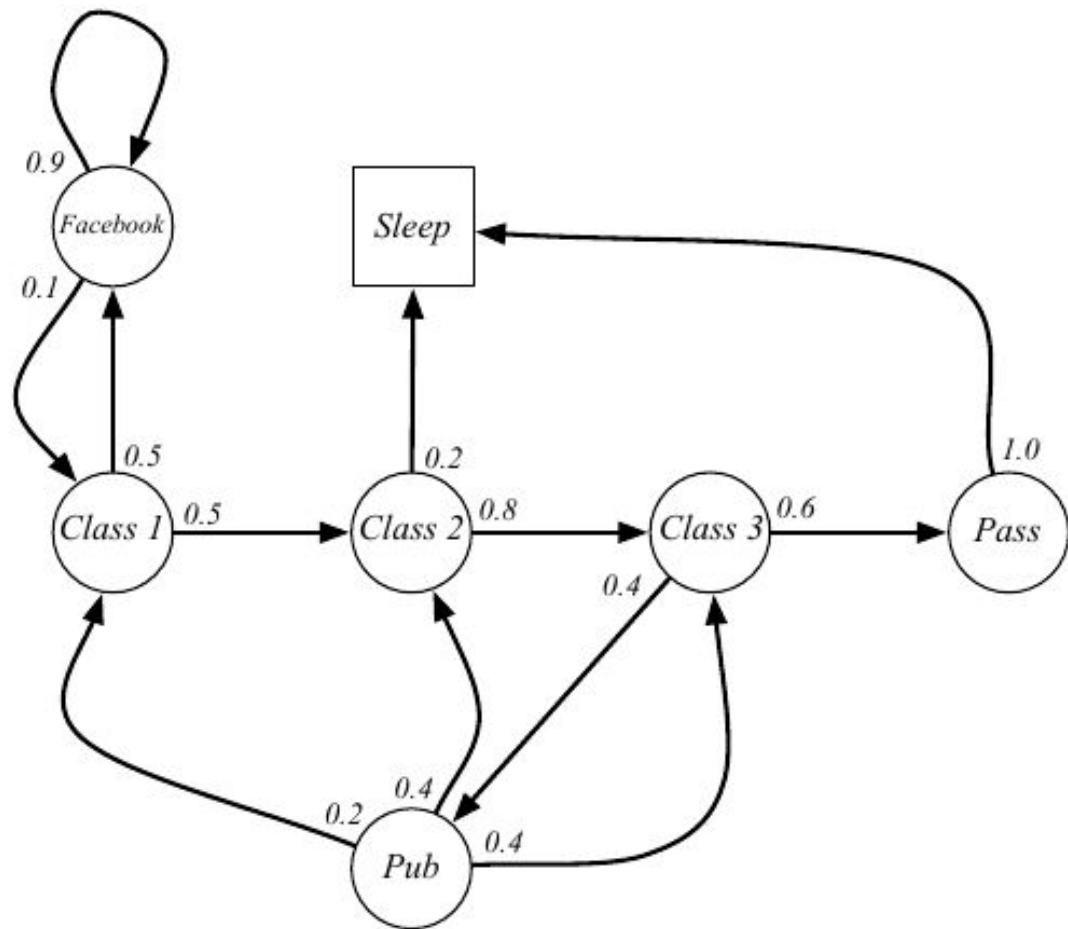
# Introducción a OpenAI Gym



## Repaso de Matemática:

- Cadenas de Markov

$$P(X_{n+1} = s' | X_n = s) = p_{ss'}$$





## Propiedad de Markov

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, \dots, S_t]$$

# Proceso o Cadena de Markov



A Markov process is a memoryless random process, i.e. a sequence of random states  $S_1, S_2, \dots$  with the Markov property.

## Definition

A *Markov Process* (or *Markov Chain*) is a tuple  $\langle \mathcal{S}, \mathcal{P} \rangle$

- $\mathcal{S}$  is a (finite) set of states
- $\mathcal{P}$  is a state transition probability matrix,  
$$\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$$

# Proceso de Recompensa de Markov

A Markov reward process is a Markov chain with values.

## Definition

A *Markov Reward Process* is a tuple  $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$  is a finite set of states
- $\mathcal{P}$  is a state transition probability matrix,  
$$\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$$
- $\mathcal{R}$  is a reward function,  $\mathcal{R}_s = \mathbb{E}[R_{t+1} \mid S_t = s]$
- $\gamma$  is a discount factor,  $\gamma \in [0, 1]$

# Proceso de recompensa de Markov

  
*Función de recompensa*

$$\mathcal{R}_s := E[R_{t+1} | S_t = s]$$

**Retorno**

$$G_t := R_{t+1} + \gamma R_{t+2} + \dots$$

*Función de Valor*

$$v(s) = E[G_t | S_t = s]$$



# Ecuación de Bellman

Podemos descomponer la función de valor en dos partes:

- La recompensa inmediata
- La función de valor en el próximo paso descontada

$$\begin{aligned}v(s) &= \mathbb{E}[G_t \mid S_t = s] \\&= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s] \\&= \mathbb{E}[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \dots) \mid S_t = s] \\&= \mathbb{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\&= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]\end{aligned}$$

# Ecuación de Bellman



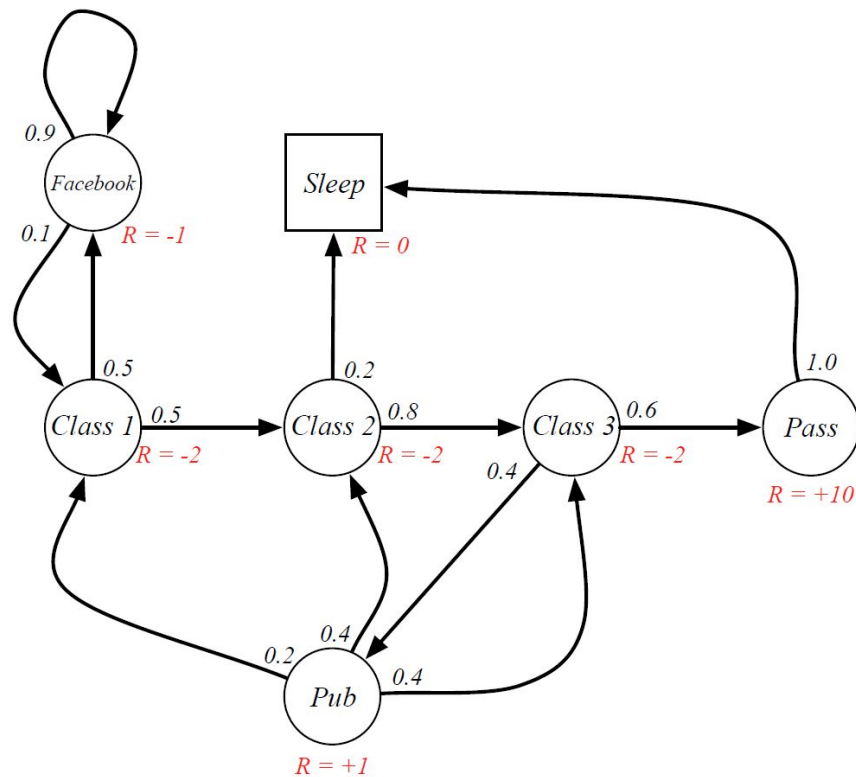
$$v(s) = \mathcal{R}_s + \gamma \sum_{s'} p_{s,s'} v(s')$$

$$v = (v(s_1), \dots, v(s_K))$$

Un sistema de K ecuaciones!

- Con matrices: 
$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$
- Escrito en una línea:  $v = \mathcal{R} + \gamma p v$

# Un ejemplo



# Proceso de Decisión de Markov



Se agrega un *acción* la cual modifica el ambiente.

$$p_{s,s'}^a := P(S_{t+1} = s' | S_t = s, A_t = a)$$

$$\mathcal{R}_s^a := E[R_{t+1} | S_t = s, A_t = a]$$

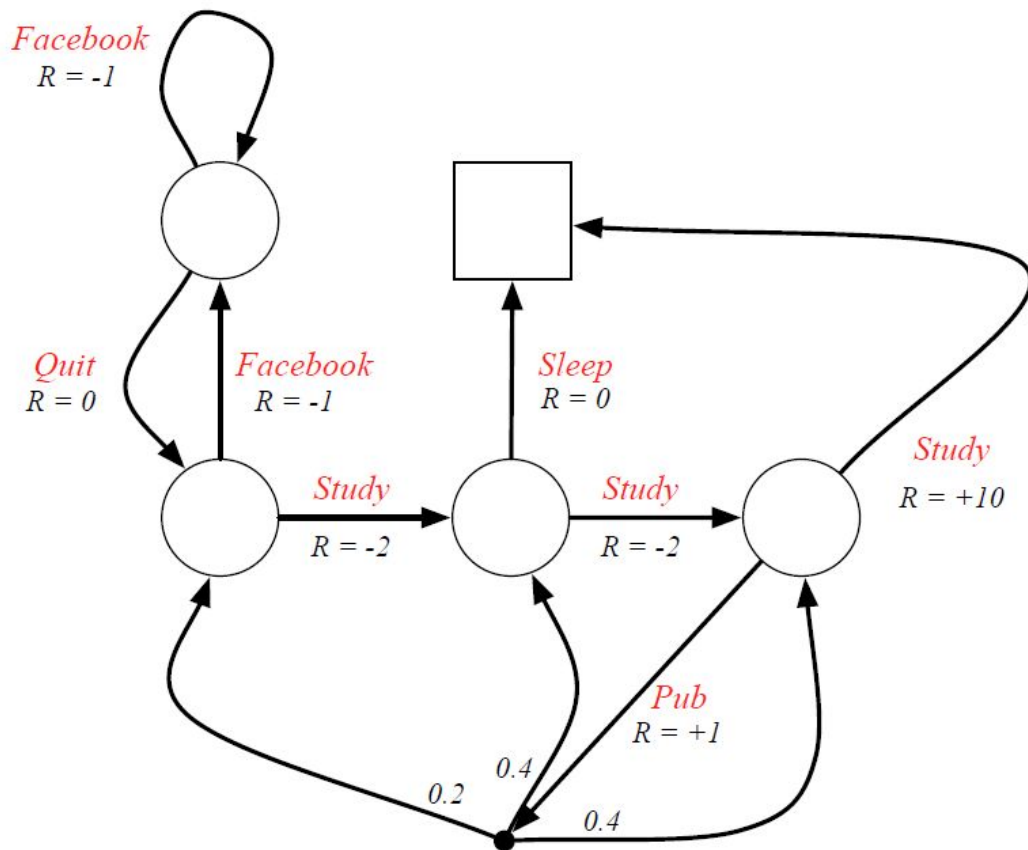
# Proceso de Decisión de Markov

## Definition

A *Markov Decision Process* is a tuple  $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$  is a finite set of states
- $\mathcal{A}$  is a finite set of actions
- $\mathcal{P}$  is a state transition probability matrix,  
$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$
- $\mathcal{R}$  is a reward function,  $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
- $\gamma$  is a discount factor  $\gamma \in [0, 1]$ .

# Ejemplo



# Política

$$\pi(a|s) = \mathbb{P}[A_t = a \mid S_t = s]$$

- Es una distribución de probabilidad sobre las acciones dado un estado
- Define el comportamiento del agente
- Dado un proceso de decisión de Markov y una política fija, la secuencia

$S_1, R_2, S_2, \dots$  es un proceso de recompensa de Markov  $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$

Donde:

$$\mathcal{P}_{s,s'}^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}_{ss'}^a$$
$$\mathcal{R}_s^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}_s^a$$



## Funciones de Valor:

- Función de valor de estado:

$$v_{\pi}(s) = \mathbb{E}_{\pi} [G_t \mid S_t = s]$$

$$v_{\pi}(s) = \mathbb{E}_{\pi} [R_{t+1} + \gamma v_{\pi}(S_{t+1}) \mid S_t = s]$$

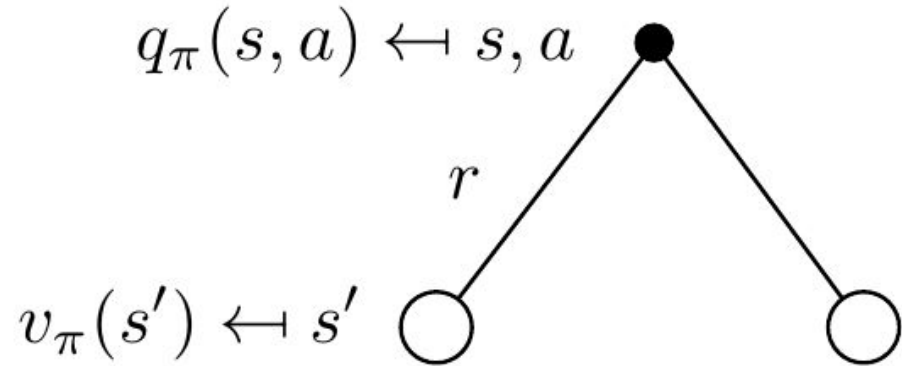
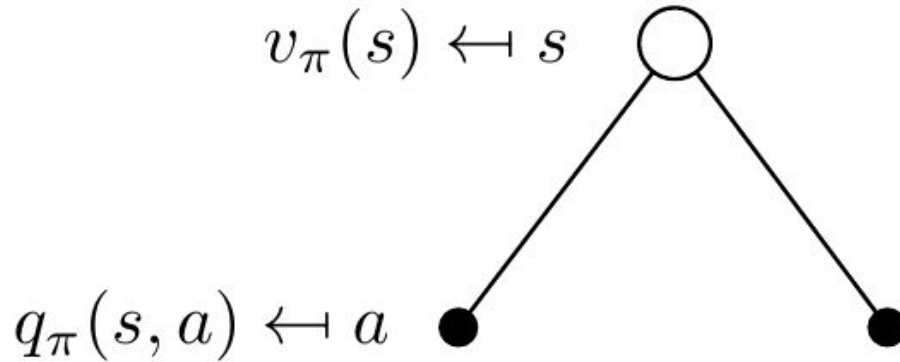
- Función de valor de estado-acción:

$$q_{\pi}(s, a) = \mathbb{E}_{\pi} [G_t \mid S_t = s, A_t = a]$$

$$q_{\pi}(s, a) = \mathbb{E}_{\pi} [R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$$

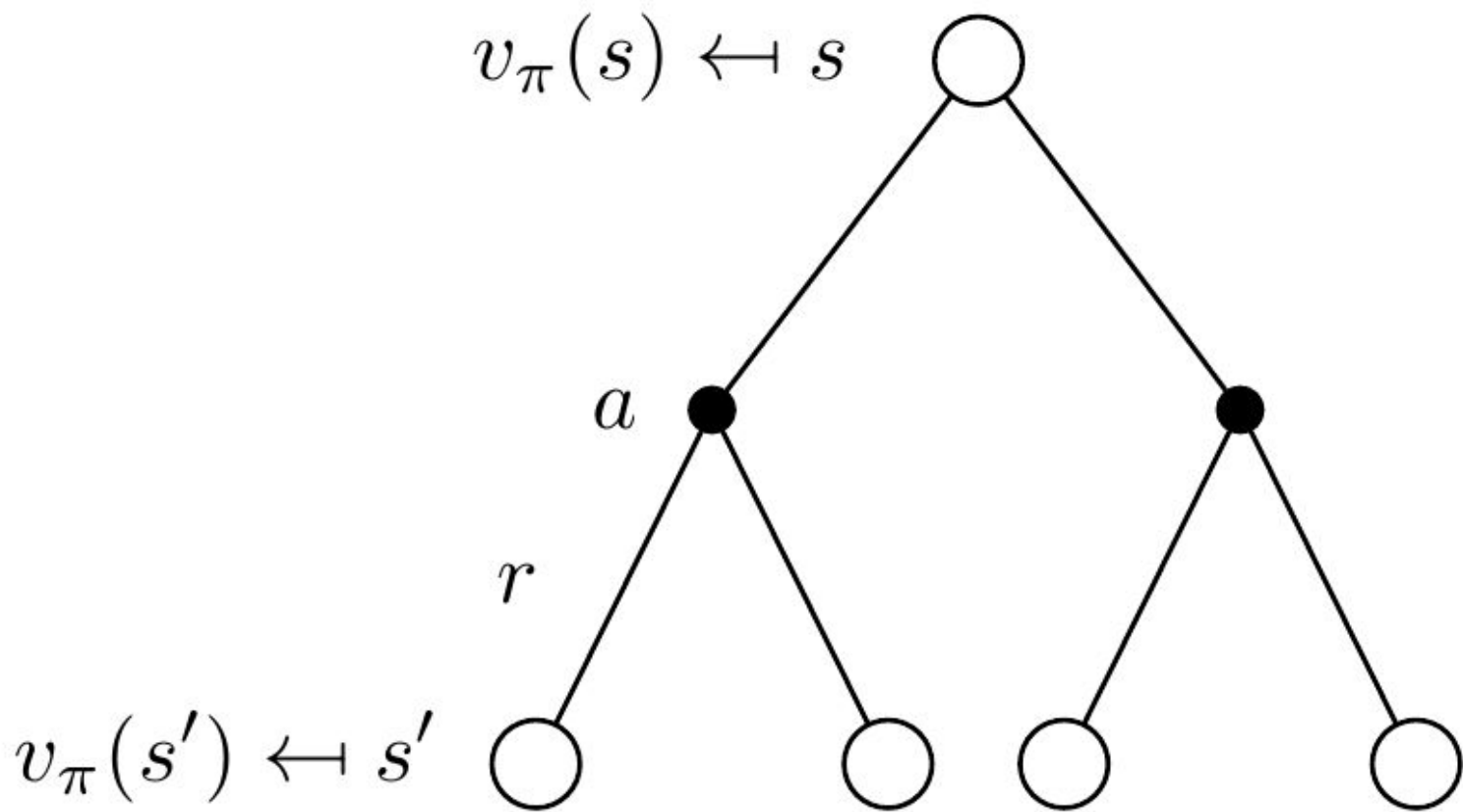


## Ecuación de Bellman (bis)




$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a)$$

$$q_\pi(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$



## Evaluación de Política


$$\begin{aligned} v_{\pi}(s) &= \sum_a [\mathcal{R}_s^a + \gamma \sum_{s'} v_{\pi}(s') p_{s,s'}^a] \pi(a|s) \\ &= \mathcal{R}_s^{\pi} + \gamma \sum_{s'} v_{\pi}(s') p_{s,s'}^{\pi} \end{aligned}$$

Método Iterativo

$$v_{\pi}^{k+1}(s) = \mathcal{R}_s^{\pi} + \gamma \sum_{s'} v_{\pi}^k(s') p_{s,s'}^{\pi}$$

# Función de Valor Óptima



$$v_*(s) = \max_{\pi} v_{\pi}(s)$$

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a)$$

## Optimalidad de MDP

$\exists \pi_* / \pi_* \geq \pi \forall \pi$  such that

$$v_*(s) = v_{\pi_*}(s), \quad q_*(s, a) = q_{\pi_*}(s, a) \quad \forall s, a$$

$$\pi_*(s) = \arg \max_a q_*(s, a)$$

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

## Ecuación de optimalidad de Bellman

$$v_*(s) = \max_a [\mathcal{R}_s^a + \gamma \sum_{s'} p_{s,s'}^a v_*(s')]$$

Son ecuaciones NO lineales!

¿Cómo obtener  $v_*$  y  $\pi_*$ ?

Si conocemos  $p_{s,s'}$  con programación dinámica podemos usar métodos iterativos.

## Predicción libre de modelo



$$v_{\pi}(s) = \mathcal{R}_s^{\pi} + \sum_{s'} v_{\pi}(s') p_{s,s'}^{\pi}$$

En general NO CONOCEMOS  $p_{s,s'}^{\pi}$

¿Podemos hacer evaluación, aprendiendo tan sólo de la experiencia?

Lo vamos a ver la próxima clase.

# Software utilizado

- ¿Cómo instalarlo? ubuntu 16.04, Python, jupyter, open ai gym
  - Linux:
    - `sudo apt-get update`
    - `sudo apt-get install python3 python3-pip ipython3 python3-fontconfig`
    - `sudo apt-get install libglu1-mesa-dev freeglut3-dev mesa-common-dev python-opengl`
    - `pip3 install numpy pandas matplotlib jupyter gym`
  - Windows:
    - Virtual Box:
      - instalar ubuntu 16.04 y usar el instructivo de la parte de Linux
    - WSL:(inspirado en <https://github.com/openai/gym/issues/11#issuecomment-242950165>)
      - instalar Windows Subsystem for Linux (WSL): <https://docs.microsoft.com/en-us/windows/wsl/install-win10>
      - instalar ubuntu 16.04 LTS para WSL yendo a Microsoft Store (barra de búsqueda de Windows) y buscando Ubuntu 16.04
      - correr una consola WSL (buscar Ubuntu en la barra de búsqueda de Windows)
      - realizar los mismos pasos que en el instructivo de linux en esa consola
      - instalar vcXsrv/xming;
      - correr vcXsrv (elegir one large window); tipear en la consola de comandos de WSL: `export DISPLAY=:0`
  - Correr jupyter: `jupyter notebook --no-browser`





## Lectura recomendada:

- AlphaGo paper: <https://ai.google/research/pubs/pub44806>
- Brief Survey of Deep RL: <https://arxiv.org/pdf/1708.05866.pdf>
- Sutton capítulo 1 para una introducción, capítulo 16 para aplicaciones, 14 y 15 para relación con psicología y neurociencia.