UNIVERSIDAD NACIONAL
DEL NORDESTE

CONICET

I M I T

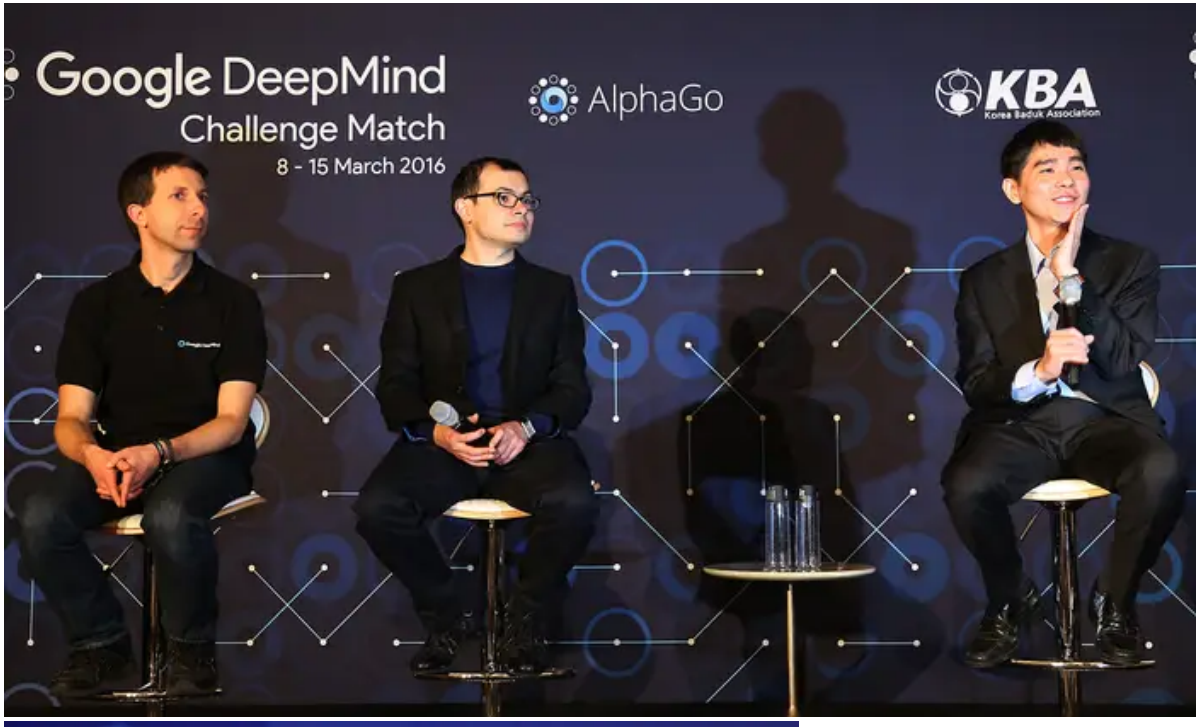
# Modulo 6:

# Aprendizaje por refuerzos

# Objetivos

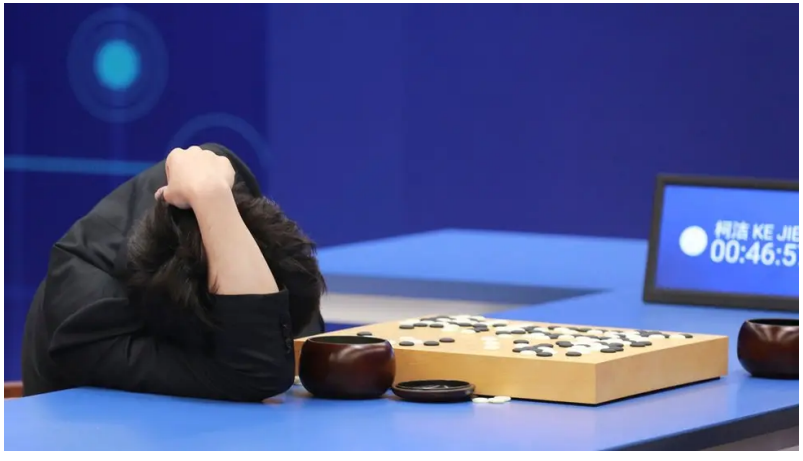
- Aprendizaje por refuerzo
- Agente y sistema de recompensas
- Ecuaciones de Bellman
- Programación dinámica
- Q-learning
- Gradientes de la política

# Reinforce learning



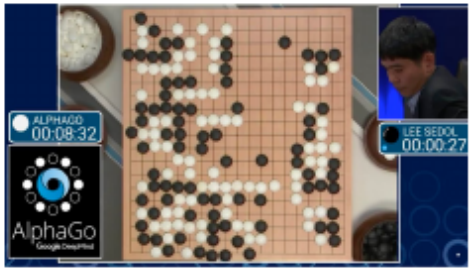

Alpha go

David Silver, Demis Hassabis, y Lee Sedol en 2016



Silver et al. 2017: Mastering the game of Go without human knowledge
Nature, 66, 354-359.

# Aplicaciones de reinforce learning

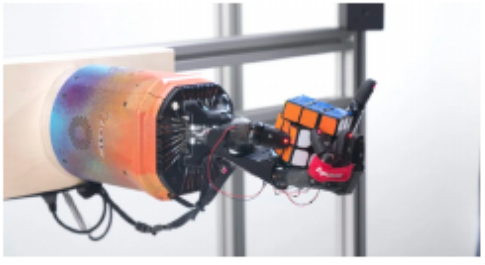



AlphaGo ('16)


ChatGPT ('22)

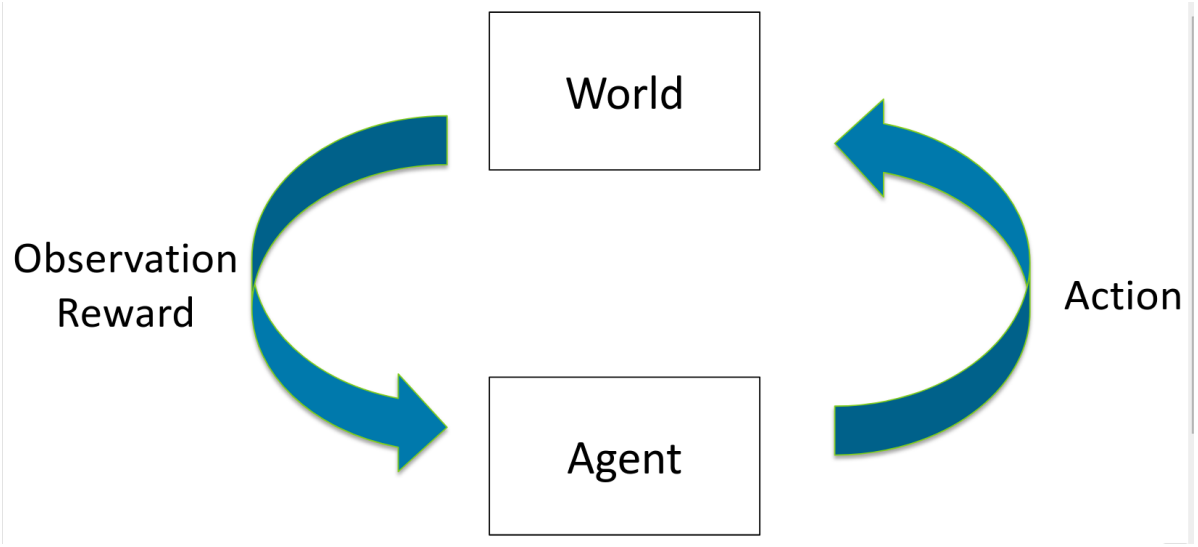
Stratospheric balloon navigation ('20)


Racing in Gran Turismo ('22)



Dexterous manipulation ('19)

# Reinforce learning




- Agentes aprenden a realizar acciones en un entorno
- interactuan con el entorno via prueba y error
- con el fin de maximizar los beneficios obtenidos en respuesta a las acciones conducidas.

Al final del día los agentes son como los humanos.

# ¿Son los LLMs inteligentes?

- Aprenden lo que los humanos conocen
- Entienden el lenguaje perfectamente.
- Se pueden entrenar para funciones específicas (fine tuning).
- Pueden conversar como si fuera un humano.

Eso es todo para ser inteligentes?

- Los LLMs no pueden descubrir nuevo conocimiento
- Los LLMs no se pueden autocorregir

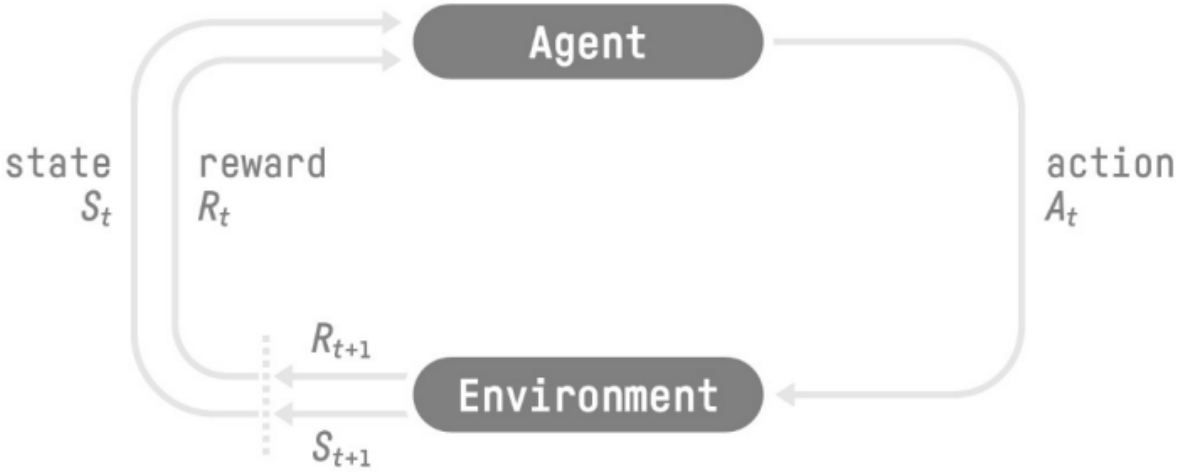
# Reinforce learning

Agentes aprenden a realizar acciones en un entorno con el fin de maximizar los beneficios obtenidos.

- El beneficio es a largo plazo.
- El entorno es estocástico.

Ejemplo. una partida de ajedrez. Se deben tomar decisiones en cada movida. El entorno sería el oponente. El retorno/beneficio es ganar la partida.
Se debe tener un balance entre exploración de nuevas opciones o usar las ya conocidas.
Se aprenden estrategias/políticas que maximizan el retorno en un proceso de decisión de Markov (MPD).

# Proceso secuencial




- Se recibe del entorno el estado $s_0$ o $x_0$.
- Basado en el estado $s_0$, el agente toma la acción $a_0$.
- El entorno evoluciona el estado a $s_1$ o $x_1$. Ej. movida del oponente en ajedrez.
- Y le da el reward/beneficio $r_1$ al agente.

Objetivo: Los agentes aprenden a realizar acciones a partir de interacciones con el mundo que maximizan el beneficio acumulativo esperado.

# Estado y observaciones

El estado es la descripción completa del “mundo” o entorno donde vive el agente.

La observación es lo que puede “ver/detectar” el agente:

- Estados completamente observados. Ajedrez.
- Observaciones parciales. Juegos (Super Mario). Conducción de auto.

# Acciones o decisiones

Cual es el conjunto de acciones que podemos tomar en un dado $t$?

En el ajedrez? Todas las posibles movidas de piezas bajo los posibles movimientos.

En el auto sin chofer? Hay también un conjunto de acciones:

- Cambio de dirección (volante)
- Reducir la velocidad con el freno.
- Acelerar o desacelerar.
- Cambio de marcha.

Notar que hay algunas acciones continuas y otras discretas.

# Beneficio acumulativo

Trayectoria futura del agente: $\tau = t+1, t+2, t+3, t+4, \cdots = t+1 : \infty$

En principio el beneficio acumulativo lo podríamos definir como la suma de los beneficios:

$$R(\tau) = r_{t+1} + r_{t+2} + r_{t+3} + \cdots = \sum_{k=0}^{\infty} r_{t+k+1}$$

Pero en realidad el entorno es estocástico y los rewards futuros son mas inciertos que los mas actuales. Cuanto antes el beneficio mejor.

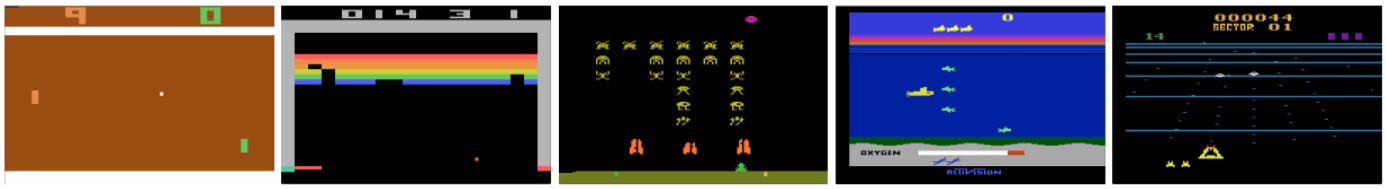
Ej. En el caso del ajedrez no podemos saber las movidas a largo plazo del oponente.

Tomamos una razón de descuento $\gamma < 1$ y lo vamos a usar de peso de los rewards:

$$R(\tau) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots$$

Retorno final: $R_t = R(\tau) = \sum_k \gamma^k r_{t+k+1}$

# Proceso de Markov con decisiones

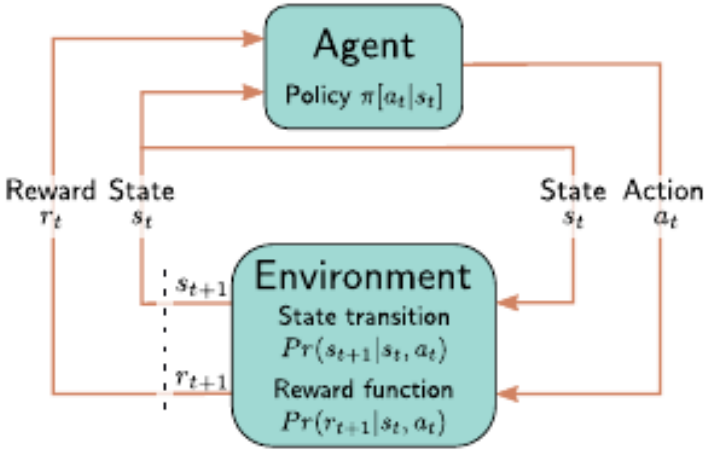


Pronóstico del estado y el beneficio en el próximo tiempo conociendo la situación actual

- $p(r_{k+1}|x_k)$ $x_k$ estado, $r_{k+1}$ beneficio. Predicción del beneficio.
- $p(x_{k+1}|x_k, a_k)$ El agente tomará decisiones que resultan en acciones $a_k$ a considerar para el estado siguiente.

  Asumimos un proceso de Markov, el beneficio solo depende del estado actual (y nos olvidamos del pasado). El próximo estado depende del actual y de la acción.
- $\pi(a_t|x_t)$ La política/estrategia del agente son las reglas que definen las acciones dado un estado.

# Ciclo de decisiones




- El agente recibe el estado $x_k$ y el beneficio $r_k$
- Luego modifica la política para definir la próxima acción: $\pi(a_k|x_k)$. La política son las decisiones del agente basados en el estado actual.
- El entorno asigna el próximo estado $p(x_{k+1}|x_k, a_k)$ y el beneficio $p(r_{k+1}|x_k, a_k)$.

Se asume que el agente observa/conoce todo el estado (completamente observado).

# Balance innovación - conocimiento previo

En el proceso de decisiones para definir la política, como en la vida real, tenemos que decidir

- Voy a lo conocido que me dio resultado (pero limitado)
- Exploro lo desconocido con la esperanza de maximizar mi reward.

¿Que regla pongo par definir entre las dos opciones?
En lo desconocido entra el azar, proceso estocastico para explorar.

# Frameworks/librerías para deep reinforce learning

- Gym. Open AI. Simula entornos de referencia: frozen lake, cart pole, mountain car, or Atari 2600 games.

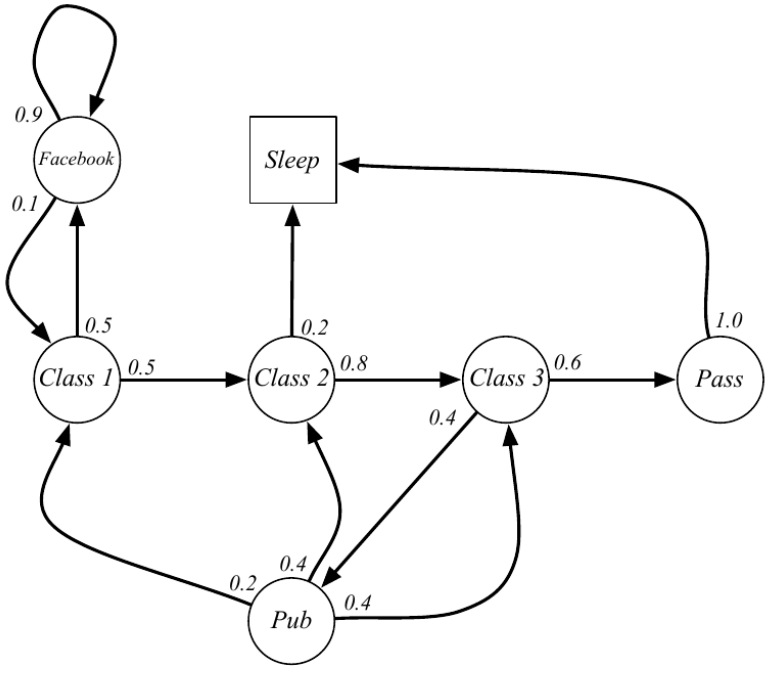
Brockman, G., 2016. OpenAI Gym. arXiv preprint arXiv:1606.01540.

- Stable-Base lines Implementación de algoritmos de reinforcement learning en pytorch.

Raffin, A., et al, 2021. Stable-baselines3: Reliable reinforcement learning implementations. JMLR, 22, 1-8.

- PyTorch RL

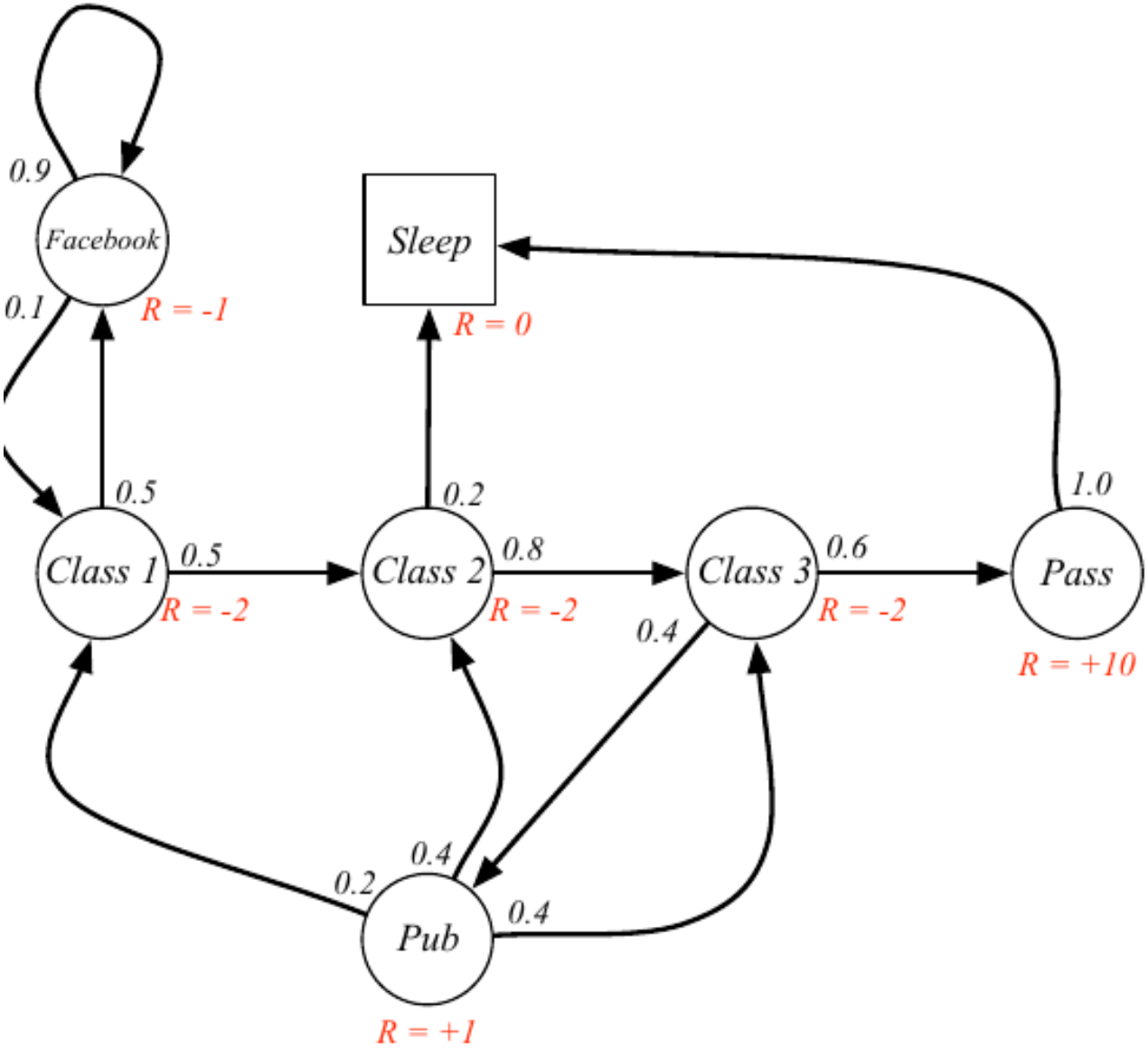
# Matriz de transición de un estudiante universitario




$$\mathcal{P} = \begin{array}{c} \\ C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ FB \\ Sleep \end{array} \begin{array}{c} \begin{array}{ccccccc} C1 & C2 & C3 & Pass & Pub & FB & Sleep \end{array} \\ \begin{bmatrix} & 0.5 & & & & 0.5 & \\ & & 0.8 & & & & 0.2 \\ & & & 0.6 & 0.4 & & \\ & & & & & & 1.0 \\ 0.2 & 0.4 & 0.4 & & & & \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1 \end{bmatrix} \end{array}$$

Ejemplo tomado de David Silver

# Beneficios de un estudiante universitario




¿Como encontramos una trayectoria que maximice los beneficios?

# Optimización en RL

Valor del estado o función valor

$$v(x_t|\pi) = \mathbb{E}\left(R_t|x_t, \pi\right)$$

Estamos en $x_t$ y seguimos la política $\pi$. Nos da la idea de cuanto sera el retorno a largo plazo dada la ubicación actual y una política.

Valor de la acción-estado o función calidad

$$Q(x_t, a_t|\pi) = \mathbb{E}\left(R_t|x_t, a_t, \pi\right)$$

Condiciono también con la acción.

La política optima maximiza a $v^*$ y $q^*$:

$$v^*(x_t) = \max_{\pi} \mathbb{E}\left(R_t|x_t, \pi\right)$$

$$Q^*(x_t, a_t) = \max_{\pi} \mathbb{E}\left(R_t|x_t, a_t, \pi\right)$$

De esta manera determinamos la política óptima sería la que optimiza el valor del estado-acción $Q$ dadas todas las posibles acciones para un $x_t$ conocido.

# Ecuaciones de Bellman

La función valor puede ser descompuesta en dos términos:

- Beneficio inmediato $r_{t+1}$
- La función valor del estado sucesor con el descuento $\gamma v(x_{t+1})$

$$\begin{aligned} v(x_t) &= \mathcal{E}[R_t|x_t] \\ &= \mathcal{E}[r_{t+1} + \gamma R_{t+1}|x_t] \\ &= \mathcal{E}[r_{t+1} + \gamma v_{t+1}|x_t] \end{aligned}$$

Si consideramos las políticas me queda una suma de todas las acciones:

$$v(x_t) = \sum_{a_t} \pi(a_t|x_t) \left[ r(x_{t+1}, a_{t+1}) + \gamma \sum_{x'} p(x'|x_t, a_t) v(x') \right]$$

# Solución de las ecuaciones de Bellman

Puedo ponerlas en formato matricial:

$$\mathbf{v} = \mathcal{R} + \gamma \mathcal{P} \mathbf{v}$$

donde $\mathbf{v}$ tiene las dimensiones de todos los estados posibles, $\mathcal{R}$ son los beneficios correspondientes y $\mathcal{P}$ es el modelo o la probabilidad de las transiciones.

Puedo encontrar la solución por

$$\mathbf{v} = (\mathbf{I} - \gamma \mathcal{P} \mathbf{v})^{-1} \mathcal{R}$$

Solo puede tener solución para estados muy pequeños.

Alternativas, los métodos iterativos:

- Programación dinámica.
- Monte Carlo.
- Aprendiendo con diferencias temporales

# Resolución de la ecuación de optimalidad de Bellman

La ecuación de optimalidad de Bellman es no lineal y no tiene forma cerrada.

Existen métodos iterativos para encontrar la solución:

- Iteración del valor (PD)
- Iteración de la política (PD)
- Sarsa (MC - On-policy)
- Aprendizaje Q (MC - Off-policy)

# Principio de programación dinámica (PD): Ecuaciones de Bellman

Def: PD son algoritmos para obtener políticas óptimas dado un modelo perfecto del mundo como un proceso de decisión de Markov.

- Es el marco teórico para desarrollar metodologías.
- En la práctica las metodologías asumen aproximación/simplificaciones a la PD.

# PD: Ecuaciones de Bellman en forma iterativa

Se inicializa los valores de estado $v(x_0)$ y la política $\pi(a_0|x_0)$.

**Evaluación de políticas:** Impacto de las políticas en el valor del estado

$$v(x_t) = \sum_{a_t} \pi(a_t|x_t) \left[ r(x_t, a_t) + \gamma \sum_{x_{t+1}} p(x_{t+1}|x_t, a_t) v(x_{t+1}) \right]$$

**Mejora de políticas** Se determina la política que maximiza el valor del estado:

$$\pi(a_t|x_t) = \text{argmax}_{a_t} \left[ r(x_t, a_t) + \gamma \sum_{x_{t+1}} p(x_{t+1}|x_t, a_t) v(x_{t+1}) \right]$$

Hay distintas variantes paso a paso o iteración de política o iteración de valores.

# Métodos de Monte Carlo

Asumimos que no conocemos ni las probabilidades de transición (model-free) ni los beneficios.

Lo que vamos a hacer es ir probando distintas trayectorias (muestreando las distribuciones) y vamos a ir observando los beneficios en las secuencias.

En este caso tenemos dos pasos también:

- Se determinan los valores de la acción $Q(x_t, a_t)$ basado en la experiencia (una serie de episodios).
- Se mejora la política basado en los valores de acción.

$$\pi(a_t|x_t) = \text{argmax}_{a_t} Q(x_t, a_t)$$

# Método de diferencia temporal: SARSA

SARSA = State-Action-Reward-State-Action

El algoritmo SARSA es on-policy, lo que se hace es:

$$Q(x_t, a_t) = Q(x_t, a_t) + \eta \left[ Y - Q(x_t, a_t) \right]$$

donde $\eta$ es la learning rate.

$\epsilon_{TD} = Y - Q(x_t, a_t)$ es el error de diferencias temporales. El target $Y$ viene dado por

$$Y = r_t + \gamma \max_a Q(x_{t+1}, a)$$

La política se renueva tomando el maximo de los valores de acción para cada estado.

$$\pi(a_t | x_t) = \text{argmax}_{a_t} Q(x_t, a_t)$$

Optimización con dos pasos iterativos:

- Evaluación: mejora la estimación minimizando $\epsilon_{TD}$ de las trayectorias dada una política.
- Mejora: se mejora la política eligiendo las acciones que son basadas en la nueva función valor.

# RL con aprendizaje Q

Buscamos maximizar los beneficios usando los valores de la acción $Q$ para mejorar iterativamente el comportamiento del agente.

Expresamos a $Q$ por ecuaciones de Bellman en forma recursiva hacia atrás

$$Q(x_t, a_t) = r(x_t, a_t) + \gamma \sum_{x_{t+1}} p(x_{t+1}|x_t, a_t) \sum_{a_{t+1}} \pi(a_{t+1}|x_{t+1}) Q(x_{t+1}, a_{t+1})$$

$\Rightarrow$ Como no conocemos el modelo de transición usamos la suma sobre los estados que ya se han visitado (y conocemos la transición).

Usamos el valor actual de $Q$ para mejorar la estimación, i.e. bootstrapping.

# Exploración en Q-learning

Es fundamental conocer la política ya que de eso dependen los estados previos, que usamos para medir el $Q$.

- Pero entonces tenemos que tratar de visitar todos los posibles sectores del espacio acción-estado.
- Para eso se pueden usar políticas aleatorias que muestren todo el espacio de las acciones.
- Pero si es totalmente aleatorio nos vamos a pasar recorriendo sin maximizar los beneficios.

Entonces vamos a elegir entre un balance dado en exploración y explotación con parametro $\epsilon$:

$$\pi(a|x) = \begin{cases} \text{argmax}_{a'} \hat{Q}(x, a') & 1 - \epsilon \\ \text{uniform} \mathcal{A} & \epsilon \end{cases}$$

# Aprendizaje con deep Q-network

Deep Q-network. Los valores de la acción son inferidos a partir de una NN.

La idea sería que la red tenga las ultimas imágenes (propuesto par los juegos de ATARI) para definir el estado en que se encuentra $\mathbf{x}_t$ y la salida de la red sea el valor de la acción para ese estado.

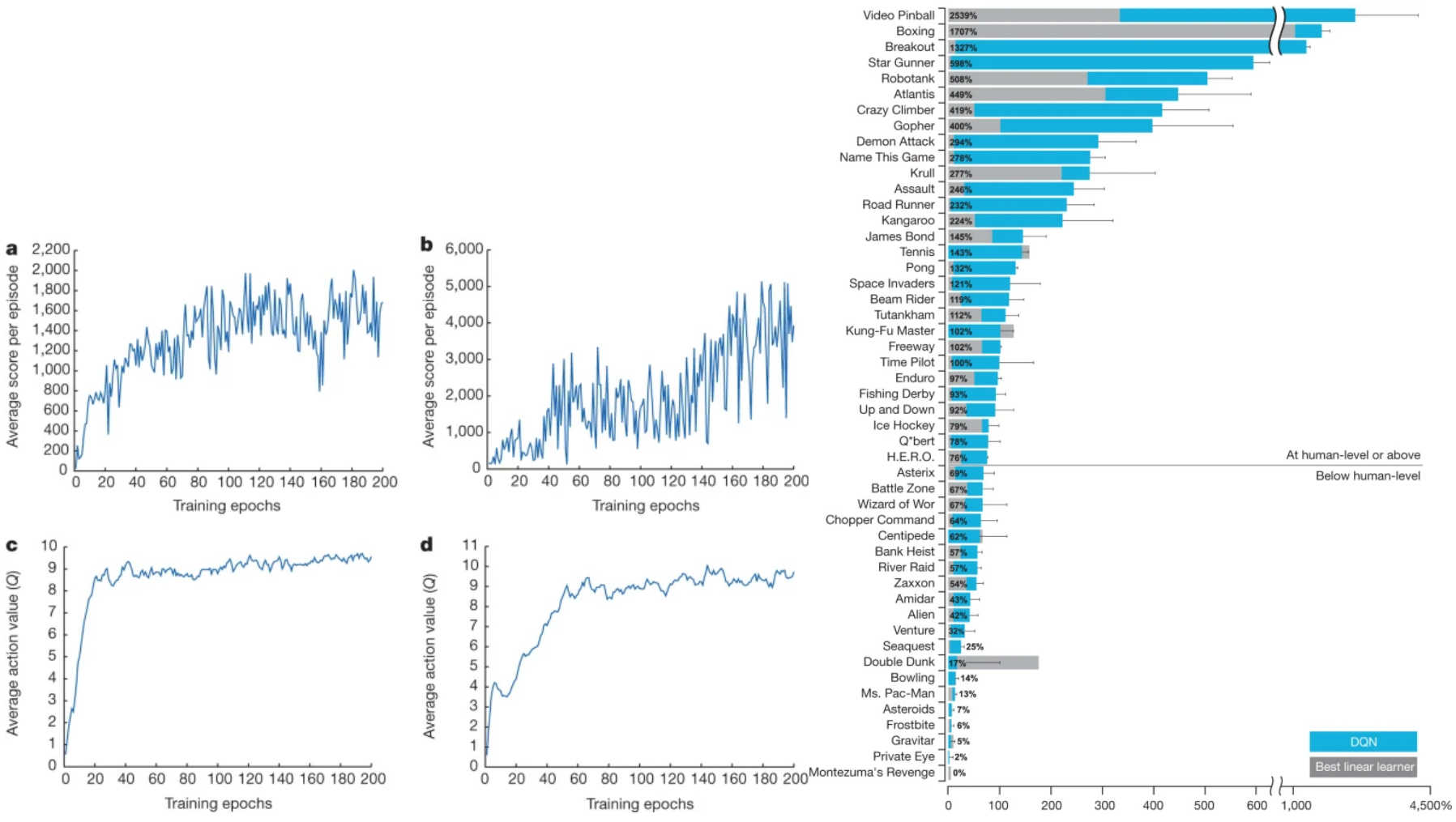
Para el aprendizaje de los parámetros de la red se usa

$$\theta_{i+1} = \theta_i + \alpha \left( r(x_t, a_t) + \gamma \max_a q(x_{t+1}, a_{t+1}, \theta^-) - q(x_t, a_t, \theta) \right) .$$

Se usan parámetros "estables" en los parámetros de salida, $\theta^-$ y solo se actualizan después de varias iteraciones.

Mnih, et al. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529–533.

# Aprendizaje con deep Q-network




Los beneficios positivos se ponen todos en 1 y los negativos en -1 mientras si no hay son 0. Se normalizan para beneficiar el entrenamiento.

# RL con gradientes de la política

En lugar de los valores de acción, se puede parametrizar la política y mejorarla a través de los descensos de gradientes.
Pero para maximizar el retorno esperado debemos promediar sobre todas las posibles trayectorias de la política actual.

- Si conocemos la dinámica de transición y es determinística, los podemos calcular en forma directa.
- Sino tenemos que hacer una estimación de Monte Carlo, con valores random, del retorno esperado por lo que estamos con el mismo problema que VAE. No podemos propagar gradientes.

Con el truco de la verosimilitud los parámetros se actualizan con

$$\theta_{iter+1} = \theta_{iter} + \eta \frac{1}{N_I} \sum_{i=1}^{N_I} \sum_{t=1}^{N_t} \frac{\partial \log \pi[a_{it}|x_{it}, \theta]}{\partial \theta} \sum_{k=1}^{N_t} r_{ik}$$

donde $i$ corresponde a episodio, y $a_{it}$ es la acción del tiempo $t$ en el episodio $i$.

# Métodos de gradientes de la política

Algoritmos de gradientes de la política:

1. REINFORCE. Gradientes de política por Monte Carlo
2. Métodos actor-crítica.
3. Optimización por política de proximidad.

## REINFORCE

1. Dada la corriente politica realizar ensamble de trayectorias.
2. Calcular el retorno de cada trayectoria.
3. Actualizar los parámetros de la política para aumentar la probabilidad de acciones que lleven a retornos mas altos.

Se asume descuento $\gamma$. Con una red neuronal se modela $\pi(x|\theta)$ nos da una distribución sobre las acciones. Los parámetros se actualizan con

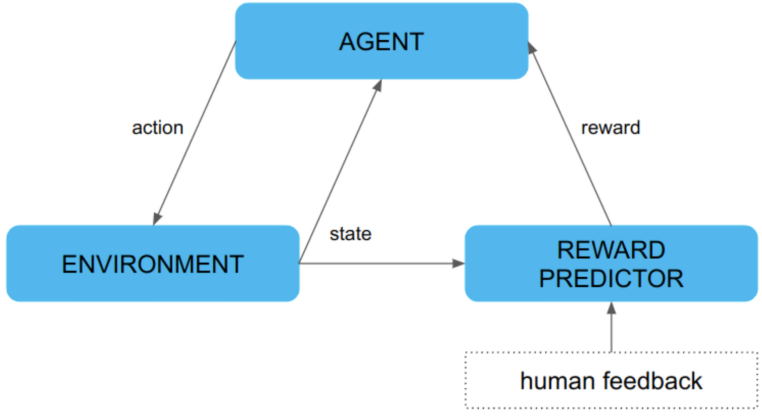
$$\theta_{iter+1} = \theta_{iter} + \eta\gamma^t \frac{\partial \log \pi[a_{it}|x_{it}, \theta]}{\partial \theta} r_{it}$$

para cada $it$.

# Instruct GPT

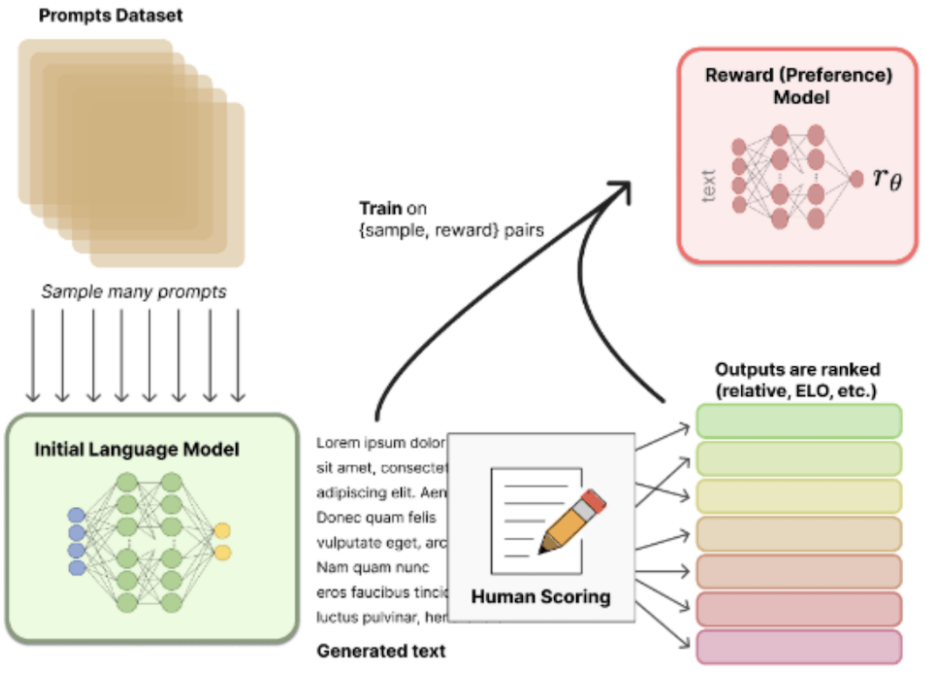
Aun los grandes modelos de lenguaje (LLMs) generan salidas que no son de utilidad para el usuario o son no-confiables o toxicas.

- Se requiere un alignment del modelo con los usuarios.
- OpenAI uso un conjunto de prompts con sus respuestas hechas por humanos para entrenar (bajo supervisión) al modelo
- Además hacen un ranking (con humanos) y puntuan la calidad de las respuestas




- Ouyang et al (2020) proponen con este datase usarlo para reinforcement learning de los LLMs o los instruct/chatLLMs (a partir de la retroalimentación humana).

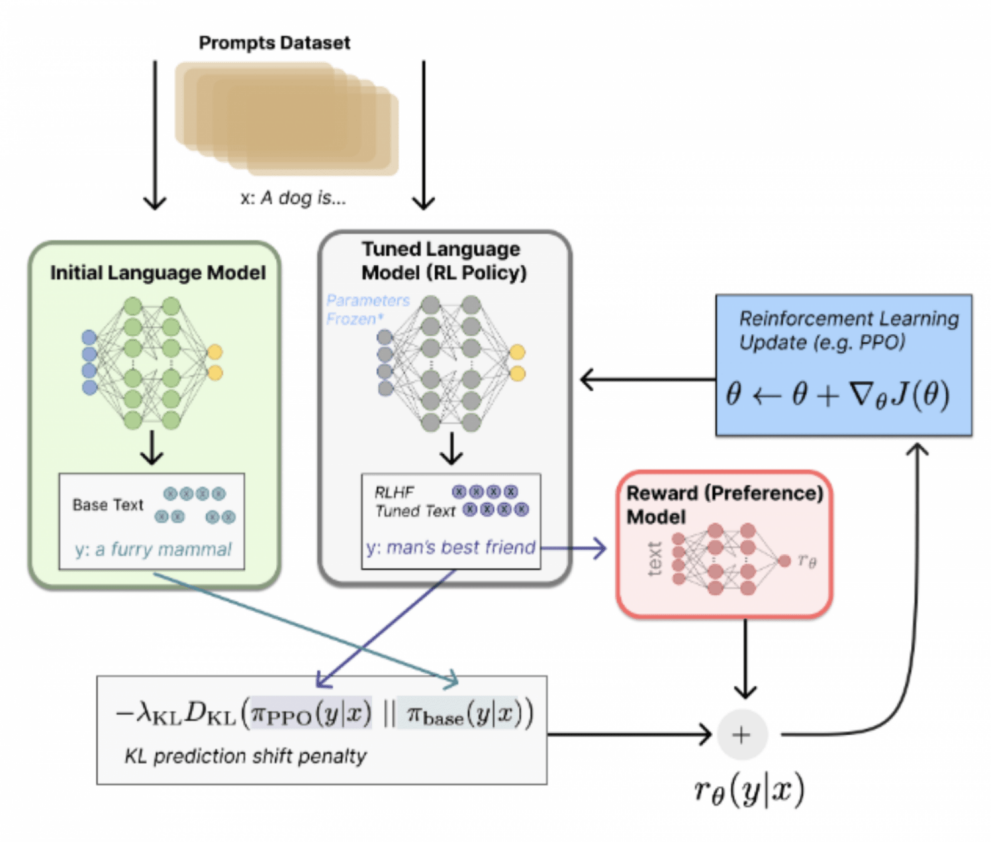
# Entrenamiento del modelo de beneficios para chatGPT




- Se realizan con el LLM de 4 a 9 respuestas al prompt.
- Esas respuestas son las que puntuan los humanos.
- Las puntuaciones se utilizan para entrentar un “reward model” que es a su vez una red neuronal.

El reward model aprende las preferencias humanas y basado en eso califica a las respuestas.

# Reinforcement learning para instruct/chatGPT




- Los agentes son instancias del modelo de lenguaje.
- El estado es la secuencia de tokens.
- El espacio de acciones posibles son todos los tokens correspondientes al vocabulario.

- Dado un estado (secuencia de tokens) el rewards es inferido por el reward model (que debe ser previamente entrenado).

# Reinforcement learning para instruct/chatGPT

Requiere la opiniones humanas de solo el 1% de las interacciones con el entorno.

En cada tiempo de la trayectoria tenemos la política $\pi$.
Pasos:

- La política $\pi$ interactua con el entorno y produce las trayectorias $\tau_{1:i}$. Los parámetros de $\pi$ se renuevan con RL para maximizar el $r_t(o_t, a_t)$.
- Se seleccionan pares de segmentos de las trayectorias $(\sigma_1, \sigma_2)$ y se las manda a un humano para evaluar.
- Los parámetros de $r(o_t, a_t)$ se optimizan con aprendizaje supervizado usando las medidas humanas.

Christiano, et al., 2017. Deep reinforcement learning from human preferences. NIPS, 30.

# Reinforcement learning para instruct/chatGPT

## Prompt

```
Explain the moon landing to a 6 year old in a few
sentences.
```

## Respuesta GPT3

```
Explain the theory of gravity to a 6 year old.
Explain the theory of relativity to a 6 year old in a
few sentences.
Explain the big bang theory to a 6 year old.
Explain evolution to a 6 year old.
```

## InstructGPT

```
People went to the moon, and they took pictures of what
they saw, and sent them back to the earth so we could
all see them.
```

Muchas gracias a Marina y Rodri.

Gracias a todos los docentes.

Gracias a uds.