Reinforcement Learning
(I)

Javier Martí

- Senior Data Scientist en Aquiles Solutions
- Máster en Inteligencia Artificial

1 Supervised vs Unsupervised vs Reinforcement Learning

2 Elementos de Reinforcement Learning

3 Aplicaciones de Reinforcement Learning

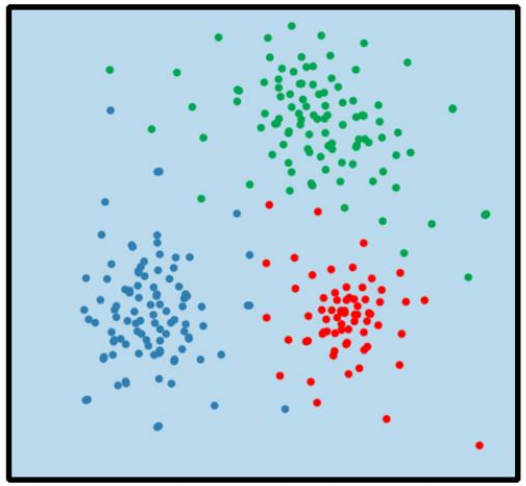
4 Historia del Reinforcement Learning

5 K-armed bandit problem

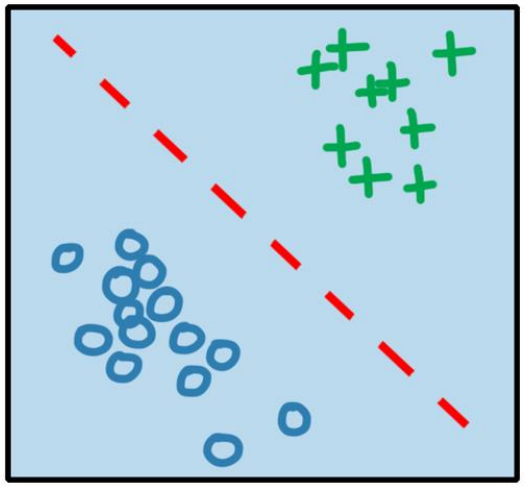
1. Supervised vs Unsupervised vs Reinforcement Learning

machine learning

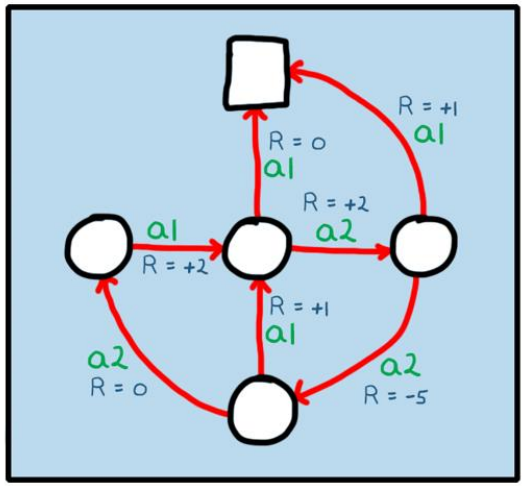
unsupervised learning



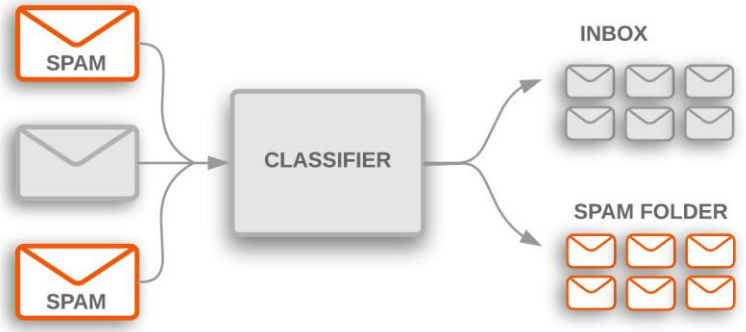
supervised learning



reinforcement learning



Supervised Learning (aprendizaje supervisado)



Datos de entrada: (x,y)
x es el dato, y es la etiqueta

Objetivo: aprender la función para mapear x → y
Aprender de un conjunto de datos etiquetados (1), para poder aplicar el resultado a un conjunto de datos no etiquetados (2).

Ejemplos: clasificación, regresión, detección de objetos, etiquetaje de imágenes, …

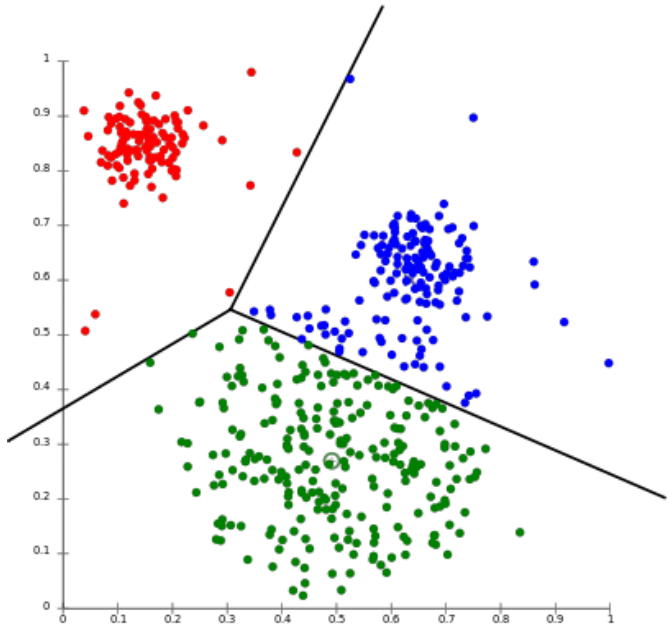
1

x1	x2	x3	x4	x5	x6	y
0	1	0	0	1	0	SPAM
1	0	0	0	0	1	NO SPAM
0	1	1	0	0	1	SPAM

2

x1	x2	x3	x4	x5	x6	y
0	1	0	0	1	0	??
1	0	0	0	0	1	??
0	1	1	0	0	1	??

Unsupervised Learning (aprendizaje no supervisado)



Datos de entrada: x
Sólo datos. NO hay etiquetas!

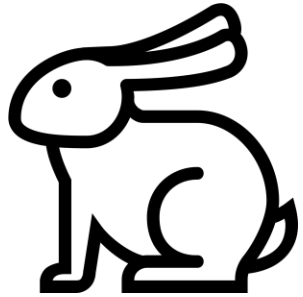
Objetivo: aprender alguna estructura oculta subyacente en los datos no etiquetados.

Ejemplos: clusterización, reducción de dimensionalidad, aprendizaje de características, estimación de densidad, ...

x1	x2	x3	x4	x5	x6	Cluster
0	1	0	0	1	0	??
1	0	0	0	0	1	??
0	1	1	0	0	1	??

Reinforcement Learning (aprendizaje por refuerzo)

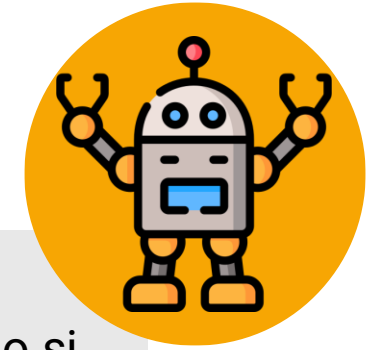
Un agente inteligente tiene que interactuar con un entorno, escogiendo una de las acciones que el entorno ofrece



Objetivo: aprender a realizar acciones para maximizar la recompensa del agente

Aprendizaje por refuerzo: ejemplos

Jugar al ajedrez. Para realizar un movimiento, el jugador tiene en cuenta tanto la planificación (anticipando el movimiento del otro jugador) como la conveniencia inmediata de determinadas posiciones y movimientos.

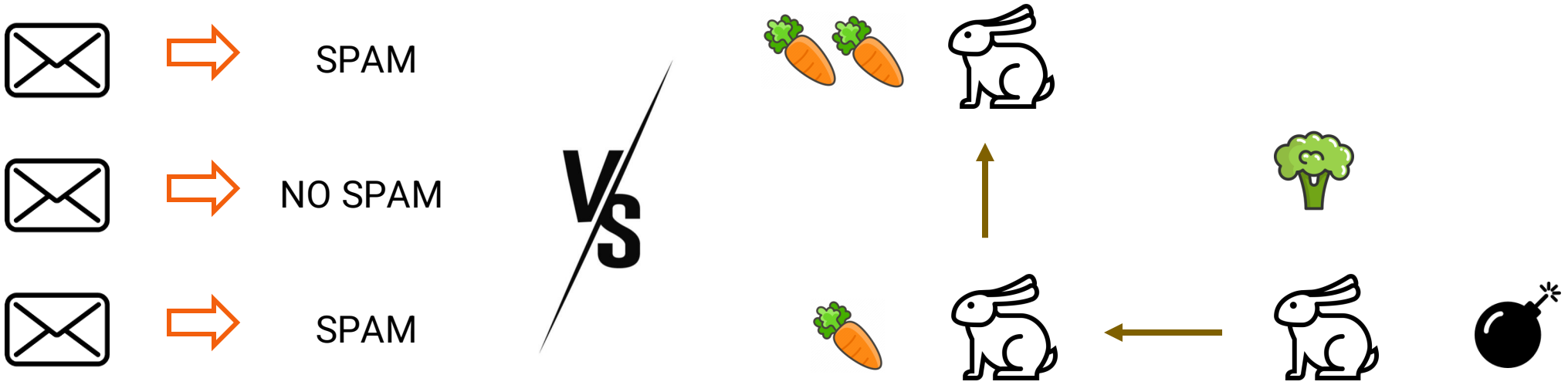


Un robot que recoge basura. El robot decide si entra a una nueva habitación en busca de más basura para recolectar, o si empezar su camino de vuelta a la estación de recarga de baterías. Toma su decisión en función del nivel de carga actual de su batería y la rapidez y facilidad con que ha podido encontrar el cargador en el pasado.

Diferencias entre aprendizaje supervisado, no supervisado y por refuerzo

1. Estático vs dinámico

- El objetivo del aprendizaje supervisado y no supervisado es buscar y aprender sobre patrones en los datos de entrenamiento, lo cual es bastante **estático**.
- RL trata de desarrollar una política que le diga a un agente qué acción elegir en cada paso, haciéndolo más **dinámico**.



Diferencias entre aprendizaje supervisado, no supervisado y por refuerzo

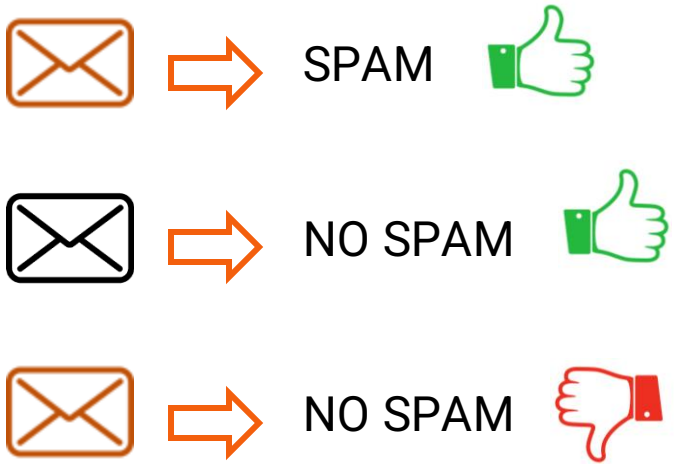
2. Específico vs global

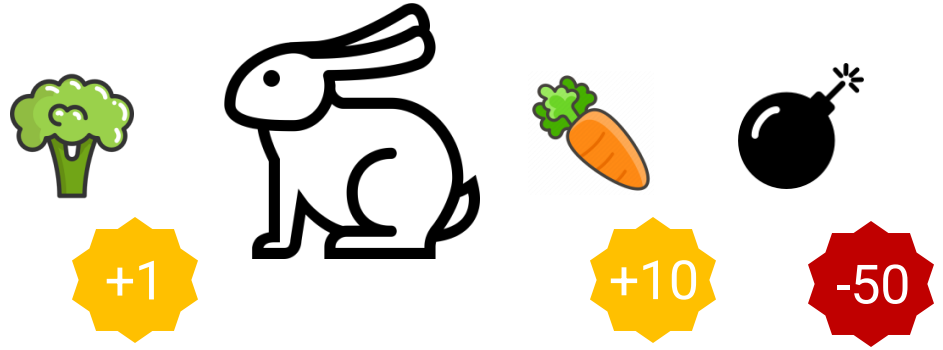
- Los problemas de ML supervisados y no supervisados son **específicos** de un caso de negocio en particular, sea de clasificación o predicción, y están muy delimitados, por ejemplo, clasificar “SPAM o NO SPAM“, o agrupar “k=3” clusters.
- En RL, como en el mundo real, contamos con múltiples variables que por lo general se interrelacionan y que dan lugar a escenarios **globales** más grandes en donde tomar decisiones.

Diferencias entre aprendizaje supervisado, no supervisado y por refuerzo

3. Sin respuesta correcta explícita

- En el aprendizaje supervisado, la respuesta correcta **se encuentra en los datos de entrenamiento**.
- En el aprendizaje por refuerzo, **la respuesta correcta NO se da explícitamente**: el agente necesita aprender por ensayo y error. La única referencia es la recompensa que recibe después de realizar una acción, que le dice al agente si está progresando o ha fallado (o podría actuar mejor).





Diferencias entre aprendizaje supervisado, no supervisado y por refuerzo

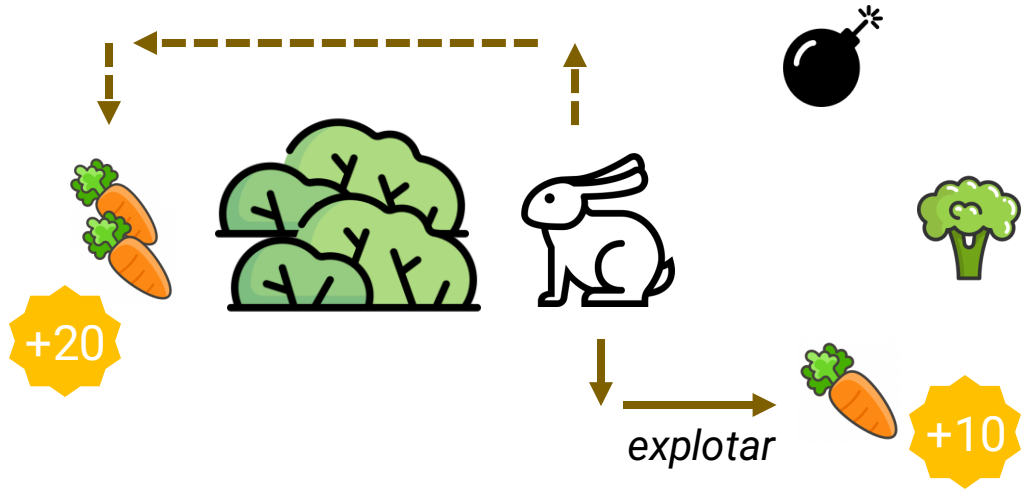
4. RL requiere exploración

- Los sistemas de aprendizaje supervisados y no supervisados toman la respuesta directamente de los datos de entrenamiento **sin tener que explorar** otras respuestas.
- Un agente de RL debe encontrar el equilibrio adecuado entre **explorar el entorno**, buscar nuevas formas de obtener recompensas y explotar las fuentes de recompensa que ya ha descubierto.

x1	x2	x3	x4	x5	x6	y
0	1	0	0	1	0	SPAM
1	0	0	0	0	1	NO SPAM
0	1	1	0	0	1	SPAM

explotar

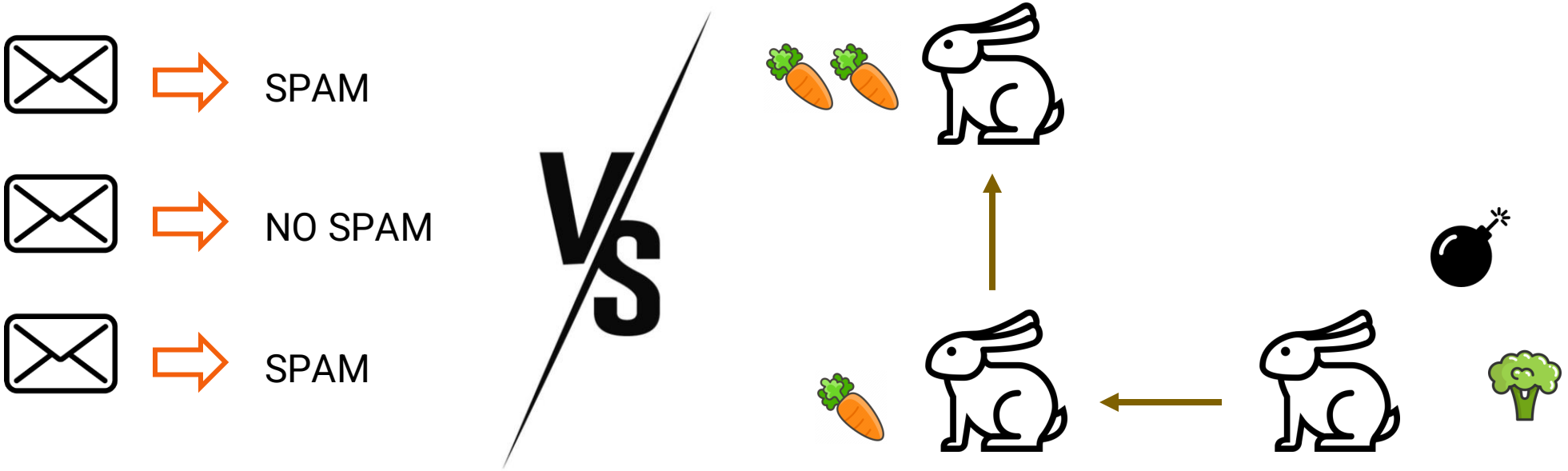
VS



Diferencias entre aprendizaje supervisado, no supervisado y por refuerzo

5. RL es un proceso de decisiones múltiples

- El aprendizaje supervisado es un proceso de **decisión única**: una instancia, una predicción.
- El aprendizaje por refuerzo es un proceso de **decisiones múltiples**: forma una secuencia de decisiones durante el tiempo necesario para terminar un trabajo específico.



Tres formas de crear una máquina inteligente

Michael Littman

Good Old-Fashioned AI: si queremos que una máquina sea inteligente, la programamos con el comportamiento que queremos que tenga.
Pero a medida que surgen nuevos problemas, la máquina no podrá adaptarse a las nuevas circunstancias. Requiere que siempre estemos ahí, con nuevos programas.

Supervised learning: si queremos que una máquina sea inteligente, le damos ejemplos para entrenar y la máquina escribe su propio programa para que coincida con esos ejemplos.
Pero las situaciones cambian: la mayoría de nosotros no tenemos la oportunidad de pescar nuestros alimentos todos los días.

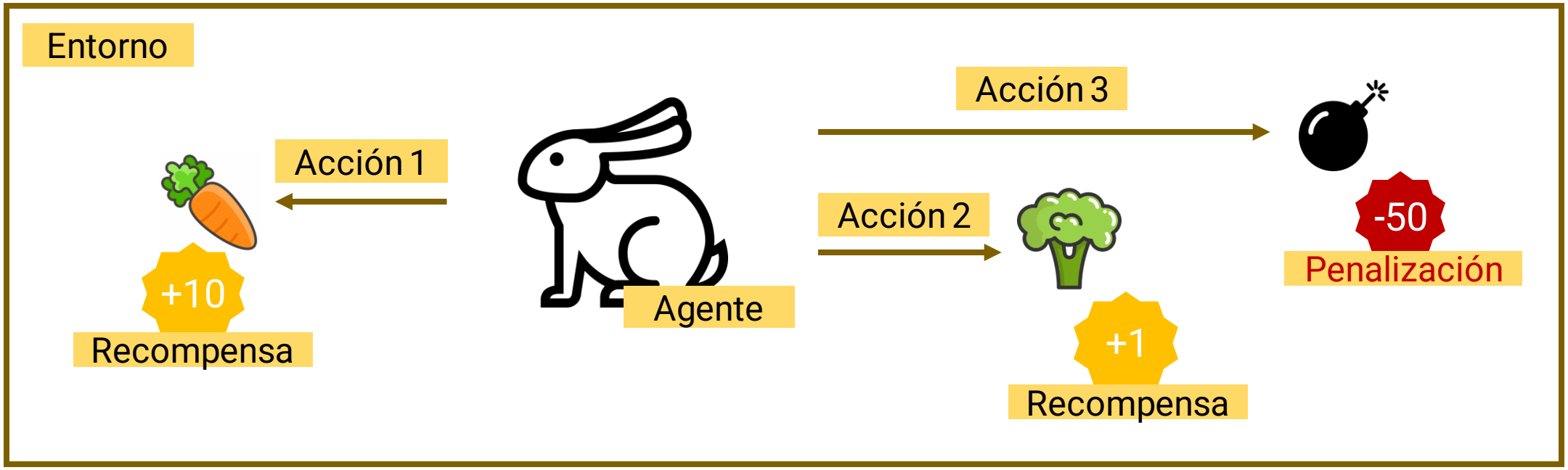
Reinforcement learning: equivale a darle al hombre el gusto por el pescado. Es la idea de que no tenemos que especificar el mecanismo para conseguir un objetivo. Simplemente podemos codificar este objetivo y la máquina diseñará su propia estrategia para lograrlo.

2. Elementos de Reinforcement Learning

Elementos de Reinforcement Learning:

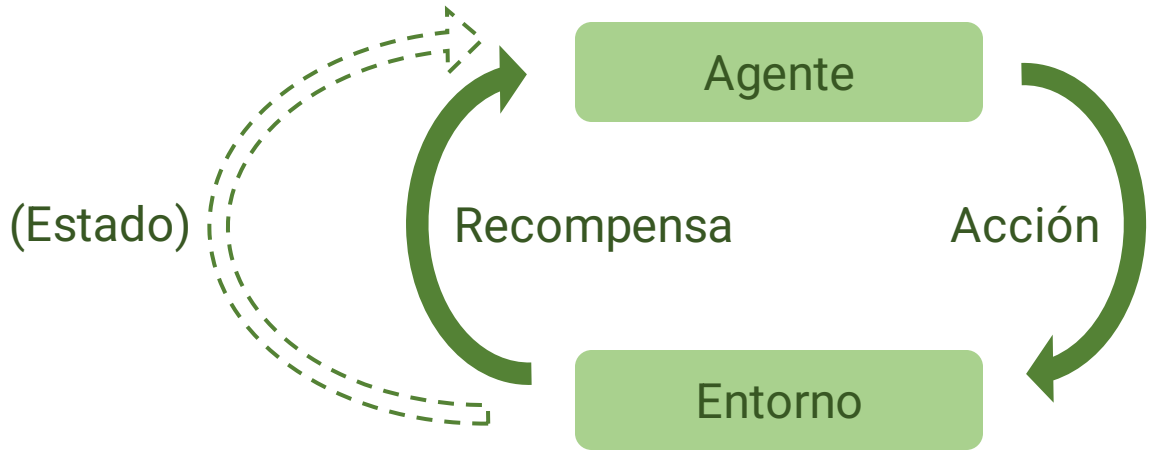
- Agente
- Entorno
- Acción
- Recompensa

El Reinforcement Learning es una técnica de Inteligencia Artificial en la que un **agente inteligente (agent)** tiene que interactuar con un **entorno (environment)**, escogiendo una de las **acciones (action)**, e intentar conseguir la mayor **recompensa (reward)** posible a través de esas acciones.

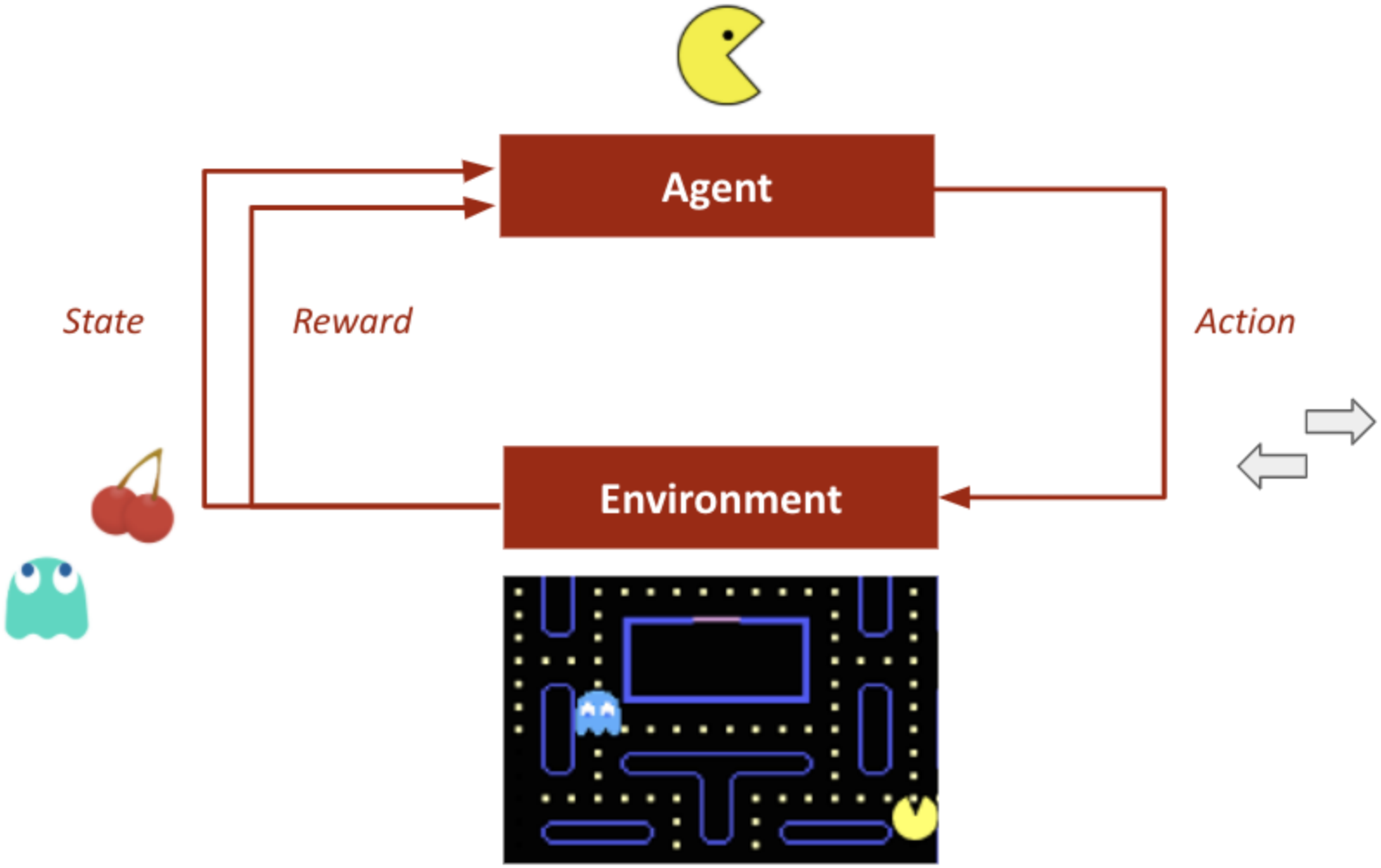


El aprendizaje por refuerzo tiene cuatro elementos esenciales:

1. **Agente**. El elemento inteligente que toma decisiones
2. **Entorno**. El mundo, real o virtual, en el que el agente realiza acciones
3. **Acción**. Un movimiento realizado por el agente
4. **Recompensa**. La valoración de una acción, que puede ser positiva o negativa



(5. El concepto de Estado lo veremos más adelante)


Agent
State
Reward
Action
Environment


State & Reward
Agent
Environment
Actions

3. Aplicaciones de Reinforcement Learning

¿Cómo modelar una tarea de Reinforcement Learning?

El primer paso es determinar cuáles son los 4 elementos:

1. Agente
2. Entorno
3. Acción
4. Recompensa

Consideramos 3 problemas y a continuación veremos cómo definir los distintos elementos:

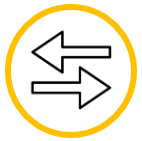
- Caso A: Enseñarle nuevos trucos a nuestro perro
- Caso B: Determinar el número de anuncios en una página web
- Caso C: Controlar un robot que recoge basura

¿Cómo modelar una tarea de Reinforcement Learning?

Caso A Enseñarle nuevos trucos a nuestro perro

Agente
Nuestro perro. No habla nuestro idioma, así que no podemos decirle directamente qué tiene que hacer. Tenemos que seguir una estrategia distinta

Entorno
Nuestra casa, dónde el perro se puede mover. Además, el entorno contiene todos los elementos relevantes para nuestra estrategia (ej. pantuflas)



Acción
Levantarse, sentarse, traer pantuflas

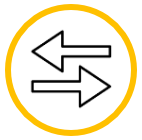
Recompensa
Si nuestro perro realiza la acción deseada, le damos un premio

¿Cómo modelar una tarea de Reinforcement Learning?

Caso B Determinar el número de anuncios en una página web

Agente
El programa que toma decisiones sobre cuántos anuncios son apropiados para una página

Entorno
La página web



Acción
Una de tres: (1) poner otro anuncio en la página; (2) eliminar un anuncio de la página; (3) ni añadir ni eliminar anuncios

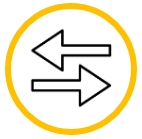
Recompensa
Positiva cuando aumentan los ingresos; negativa cuando los ingresos bajan

¿Cómo modelar una tarea de Reinforcement Learning?

Caso C Controlar un robot que recoge basura

Agente
El programa que controla el robot

Entorno
El mundo real



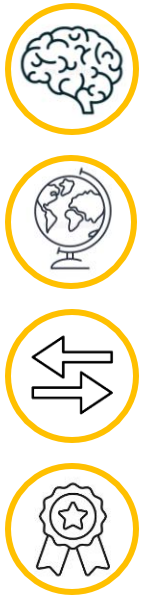
Acción
Una de cuatro: moverse (1) hacia adelante, (2) hacia atrás, (3) izquierda y (4) derecha

Recompensa
Positiva cuando recoge basura; negativa cuando pierde tiempo, se queda sin batería sin volver al punto de carga, va en la dirección equivocada o se cae

¿Cómo modelar una tarea de Reinforcement Learning?

Ejercicio: definir los 4 elementos (agente, entorno, acción, recompensa) para el caso de:

Una partida de ajedrez



Aplicaciones prácticas del aprendizaje por refuerzo:

- Robótica para automatización industrial
- Procesamiento de Lenguaje Natural (NLP)
- Sistemas educativos personalizados
- Control de aeronaves
- **Construcción de agentes inteligentes para juegos de ordenador**

4. Historia del Reinforcement Learning

Desarrollos iniciales

- **Aprendizaje por ensayo y error**
  Comenzó con la psicología del aprendizaje animal (1911)

- **Programación dinámica**
  Referido al problema del control óptimo y su solución mediante funciones de valor (1950s)

Desarrollos posteriores

- **Q-Learning**
  Combinación del aprendizaje por ensayo y error con la programación dinámica por Watkins (1989)

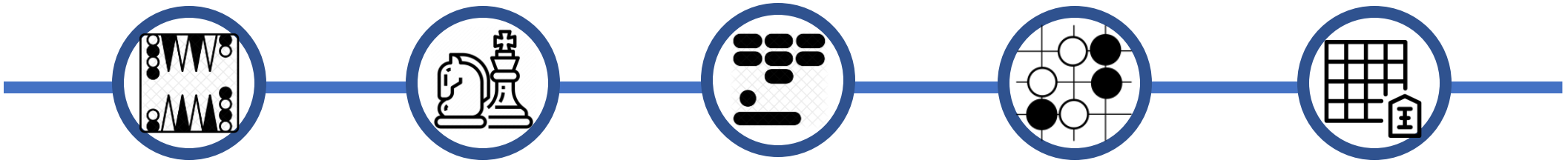
1992: TD-Gammon
Alcanza el "Nivel Maestro" en el juego de Backgammon

1997: Deep Blue de IBM
Gana contra el campeón mundial de ajedrez

2010s: DeepMind
Utiliza RL en juegos clásicos de Atari

2016: AlphaGo
Gana contra el campeón mundial de Go

2017: AlphaZero
Aniquila AlphaGo 100-0

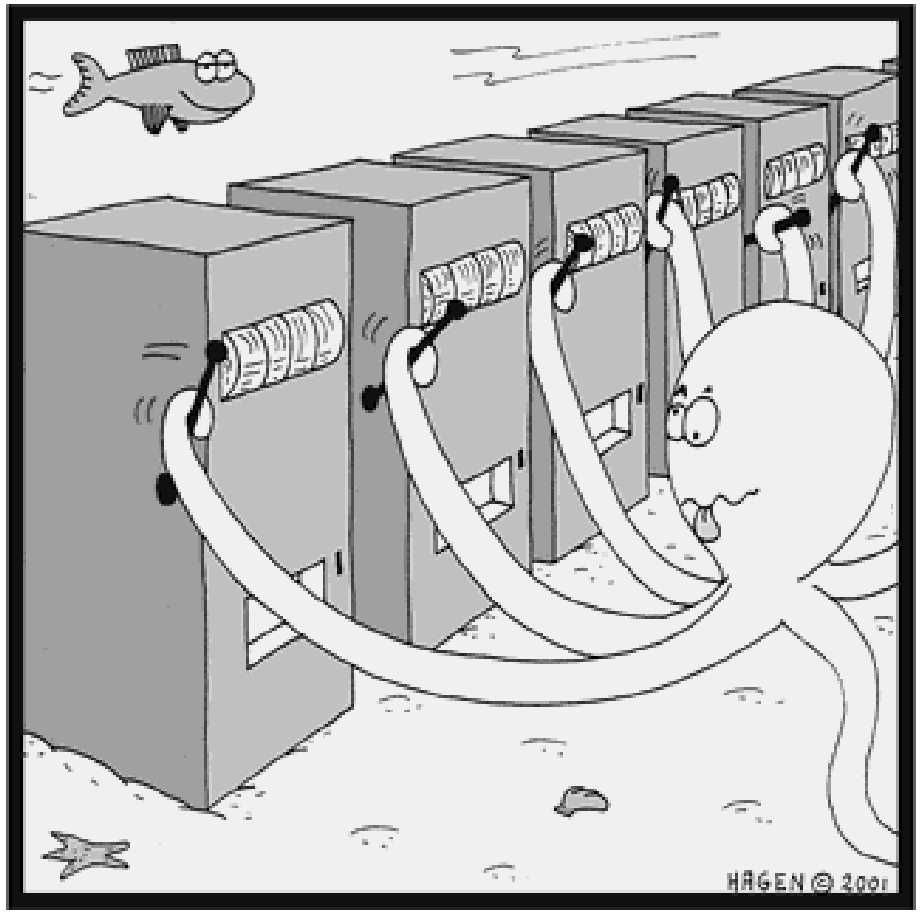


Documental de Netflix sobre **AlphaGo**
https://www.youtube.com/watch?v=WXuK6gekU1Y


021 3 1

5. K-armed bandit problem

K-armed bandit problem



One-armed bandit
máquinas tragaperras

K-armed bandit problem
Cuando hay más máquinas entre las cuales elegir

K-armed bandit problem

Consideraciones previas

- Jugar a estas máquinas tiene una **probabilidad de ganar**
- No conocemos la distribución de probabilidad y no podemos descubrirla con un número limitado de intentos
- Tenemos 3 máquinas tragaperras diferentes, cada una con una probabilidad de ganar
- Tenemos varias monedas para jugar

Problema

- **¿Qué máquina tragaperras elegimos**?

Estrategia de resolución

- Aprenderemos como resolver este problema mediante **Reinforcement Learning**

K-armed bandit problem

probabilidad de ganar

? %

probabilidad de ganar

? %

probabilidad de ganar

? %

K-armed bandit problem: definir recompensa

probabilidad de ganar

60%

Recompensa 1 cuando ganamos (60%)

Recompensa 0 cuando perdemos (40%)

Recuerda!

El jugador no conoce a priori la probabilidad de ganar de la máquina

Clase 1: K-armed bandit problem

K-armed bandit problem: definir action-value

Expected Price Action (action-value) : probabilidad estimada de ganar

$$\text{exp\_price\_action} = \text{exp\_price\_action} + \frac{reward - \text{exp\_price\_action}}{iteration}$$

El jugador no conoce la probabilidad real de ganar de la máquina

Lo que puede hacer para intentar deducirla es jugar varias veces (**iteraciones**) y actualizar sus expectativas en función de los resultados obtenidos (**recompensas**)

Clase 1: K-armed bandit problema: Definir action-value

K-armed bandit problem: definir action-value

probabilidad estimada de ganar

100%

Inicializamos a 1 (100%)

$$\text{exp\_price\_action}_0 = 1$$

K-armed bandit problem: definir action-value

probabilidad estimada de ganar

100%

Iteración 1: GANAMOS

$$\text{exp\_price\_action}_1 = \text{exp\_price\_action}_0 + \frac{reward_1 - \text{exp\_price\_action}_0}{iteration} =$$

$$= 1 + \frac{1 - 1}{1} = 1$$

K-armed bandit problem: definir action-value

probabilidad estimada de ganar

100%

¿y si hubiéramos inicializado con otro valor?

Iteración 1: GANAMOS

$$\text{exp\_price\_action}_1 = \text{exp\_price\_action}_0 + \frac{reward_1 - \text{exp\_price\_action}_0}{iteration} =$$

$$= 0 + \frac{1 - 0}{1} = 1$$

$$= 10 + \frac{1 - 10}{1} = 1$$

$$= 42 + \frac{1 - 42}{1} = 1$$

K-armed bandit problem: definir action-value

probabilidad estimada de ganar

100%

Iteración 2: GANAMOS

¡ nueva iteración !

$$\text{exp\_price\_action}_2 = \text{exp\_price\_action}_1 + \frac{reward_2 - \text{exp\_price\_action}_1}{iteration} =$$

$$= 1 + \frac{1-1}{2} = 1$$

K-armed bandit problem: definir action-value

probabilidad estimada de ganar

100%

Iteración 3: GANAMOS

¡ nueva iteración !

$$\text{exp\_price\_action}_3 = \text{exp\_price\_action}_2 + \frac{reward_3 - \text{exp\_price\_action}_2}{iteration} =$$

$$= 1 + \frac{1-1}{3} = 1$$

K-armed bandit problem: definir action-value

probabilidad estimada de ganar

75%

Iteración 4: PERDEMOS

¡ reward = 0 !

$$\text{exp\_price\_action}_4 = \text{exp\_price\_action}_3 + \frac{reward_4 - \text{exp\_price\_action}_3}{iteration} =$$

$$= 1 + \frac{0 - 1}{4} = 0{,}75$$

K-armed bandit problem: definir action-value

probabilidad estimada de ganar

80%

Iteración 5: GANAMOS

¡ exp_price_action actualizada !

$$\text{exp\_price\_action}_5 = \text{exp\_price\_action}_4 + \frac{reward_5 - \text{exp\_price\_action}_4}{iteration} =$$

$$= 0{,}75 + \frac{1 - 0{,}75}{5} = 0{,}8$$

K-armed bandit problem: definir action-value

probabilidad estimada de ganar

83%

Iteración 6: GANAMOS

¡ exp_price_action actualizada !

$$\text{exp\_price\_action}_6 = \text{exp\_price\_action}_5 + \frac{reward_6 - \text{exp\_price\_action}_5}{iteration} =$$

$$= 0{,}8 + \frac{1 - 0{,}8}{6} = 0{,}83$$

K-armed bandit problem: definir action-value

probabilidad estimada de ganar

86%

Iteración 7: GANAMOS

$$\text{exp\_price\_action}_7 = \text{exp\_price\_action}_6 + \frac{reward_7 - \text{exp\_price\_action}_6}{iteration} =$$

$$= 0{,}83 + \frac{1 - 0{,}83}{7} = 0{,}86$$

K-armed bandit problem: definir action-value

probabilidad estimada de ganar

75%

Iteración 8: PERDEMOS

¡ reward = 0 !

$$\text{exp\_price\_action}_8 = \text{exp\_price\_action}_7 + \frac{reward_8 - \text{exp\_price\_action}_7}{iteration} =$$

$$= 0{,}86 + \frac{0 - 0{,}86}{8} = 0{,}75$$

K-armed bandit problem: definir action-value

probabilidad estimada de ganar

67%

Iteración 9: PERDEMOS

$$\text{exp\_price\_action}_9 = \text{exp\_price\_action}_8 + \frac{reward_9 - \text{exp\_price\_action}_8}{iteration} =$$

$$= 0{,}75 + \frac{0 - 0{,}75}{9} = 0{,}67$$

K-armed bandit problem: definir action-value

probabilidad estimada de ganar

70%

Iteración 10: GANAMOS

$$\text{exp\_price\_action}_{10} = \text{exp\_price\_action}_{9} + \frac{reward_{10} - \text{exp\_price\_action}_{9}}{iteration} =$$

$$= 0{,}67 + \frac{1 - 0{,}67}{10} = 0{,}7$$

K-armed bandit problem: definir action-value

Probabilidad estimada de ganar después de 10 iteraciones

70%

~

Probabilidad real de ganar (definida a priori)

60%

K-armed bandit problem: jugar con 3 máquinas

probabilidad de ganar

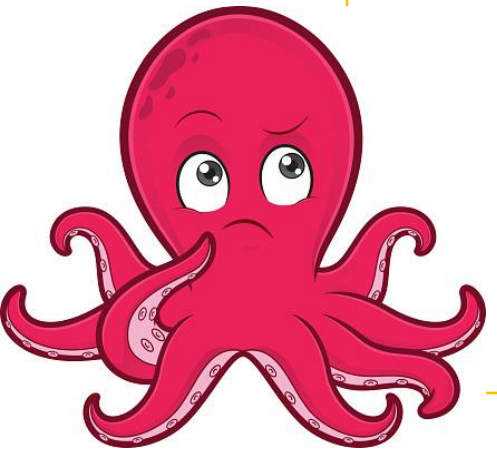
? %

probabilidad de ganar

? %

probabilidad de ganar

? %



Clase 1: K-armed bandit problema: Jugar con 3 máquinas

K-armed bandit problem

…después de 1000 iteraciones, eligiendo una máquina al azar

probabilidad estimada de ganar

92%

probabilidad real: 90%

probabilidad estimada de ganar

52%

probabilidad real: 50%

probabilidad estimada de ganar

11%

probabilidad real: 10%

K-armed bandit problem

…después de 1000 iteraciones, eligiendo una máquina al azar

probabilidad estimada de ganar: 92%

322

veces

probabilidad estimada de ganar: 52%

349

veces

probabilidad estimada de ganar: 11%

329

veces

K-armed bandit problem

...después de 1000 iteraciones, eligiendo una máquina al azar

probabilidad estimada de ganar: 92%

322

veces

probabilidad estimada de ganar: 52%

349

veces

probabilidad estimada de ganar: 11%

329

veces

Recompensa total: 513 puntos

K-armed bandit problem: definir greedy policy

¿Y si no quisiéramos elegir una máquina al azar,
sino la que nos da mayor recompensa?

```
def greedy(exp_price):
    index=np.argmax(exp_price)
    return index
```

Clase 1: K-armed bandit problema: Definir greedy policy

K-armed bandit problem: aplicar greedy policy (3 máquinas)

Después de 1000 iteraciones, eligiendo la máquina con mayor probabilidad esperada de ganar...

probabilidad estimada de ganar: 90%

997

veces

probabilidad estimada de ganar: 50%

2

veces

probabilidad estimada de ganar: 0%

1

vez

K-armed bandit problem: aplicar greedy policy (3 máquinas)

Después de 1000 iteraciones, eligiendo la máquina con mayor probabilidad esperada de ganar...

probabilidad estimada de ganar: 90%

997

veces

probabilidad estimada de ganar: 50%

2

veces

probabilidad estimada de ganar: 0%

1

vez

Recompensa total: 900 puntos

K-armed bandit problem: aplicar greedy policy (muchas máquinas)

probabilidad de ganar

80%

probabilidad de ganar

40%

probabilidad de ganar

60%

probabilidad de ganar

85%

probabilidad de ganar

20%

probabilidad de ganar

75%

probabilidad de ganar

90%

probabilidad de ganar

55%

probabilidad de ganar

80%

probabilidad de ganar

60%

Si tenemos muchas máquinas, corremos el riesgo de no explorar

Clase 1: K-armed bandit problema: Jugar con muchas máquinas

K-armed bandit problem: jugar con muchas máquinas

probabilidad real: 80%

probabilidad real: 40%

probabilidad real: 60%

probabilidad real: 85%

probabilidad real: 20%

probabilidad real: 75%

probabilidad real: 90%

probabilidad real: 55%

probabilidad real: 80%

probabilidad real: 60%

K-armed bandit problem: jugar con muchas máquinas

5

veces

probabilidad real: 80%

2

veces

probabilidad real: 40%

1

vez

probabilidad real: 60%

2

veces

probabilidad real: 85%

1

vez

probabilidad real: 20%

9

veces

probabilidad real: 75%

2

veces

probabilidad real: 90%

1

vez

probabilidad real: 55%

975

veces

probabilidad real: 80%

2

veces

probabilidad real: 60%

K-armed bandit problem: exploration vs explotation

Trade-off entre explorar y explotar

```
def greedy_trade_off(exp_price,epsilon):
    if(rd.random() < epsilon):
            index=rd.randint(0,len(exp_price)-1) #explorar
    else:
        index=np.argmax(exp_price) #explotar
    return index
```

Clase 1: K-armed bandit problema: Exploration vs Explotation

K-armed bandit problem: exploration vs explotation

Exploramos el 15% de las veces

probabilidad de ganar

80%

probabilidad de ganar

40%

probabilidad de ganar

60%

probabilidad de ganar

85%

probabilidad de ganar

20%

probabilidad de ganar

75%

probabilidad de ganar

90%

probabilidad de ganar

55%

probabilidad de ganar

80%

probabilidad de ganar

60%

K-armed bandit problem: exploration vs explotation

Exploramos el 15% de las veces

1

16

veces

probabilidad real: 80%

2

15

veces

probabilidad real: 40%

3

17

veces

probabilidad real: 60%

4

21

veces

probabilidad real: 85%

5

19

veces

probabilidad real: 20%

6

19

veces

probabilidad real: 75%

7

840

veces

probabilidad real: 90%

8

16

veces

probabilidad real: 55%

9

20

veces

probabilidad real: 80%

10

17

veces

probabilidad real: 60%