

12/01/2026

¿Qué es la inteligencia artificial?

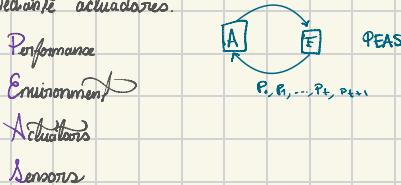
- Sistemas que piensan como humanos.
- Sistemas que actúan como humanos.
- Sistemas que piensan racionalmente.
- Sistemas que actúan **racionalmente**.

Se define como el desarrollo de agentes racionales que interactúan con su entorno para maximizar la esperanza de una utilidad futura.

13/01/2026

El enfoque unificado: El agente racional

¿Qué es un agente? Cualquier cosa que perciba su ambiente a través de sensores y actúe sobre él mediante actuadores.



Un agente racional:

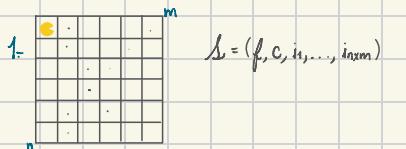
- Maximiza la utilidad esperada.
- Usa la inf. disponible.
- Considera las acciones posibles.
- Aprende y explora si es necesario.

Estado:

Un estado se representa como un vector de variables. Cada variable tiene su dominio.

$$S = (s_1, s_2, \dots, s_n) \in D_1 \times D_2 \times \dots \times D_n$$

Representación de estados:



$$S = (f, c, i_1, \dots, i_m)$$

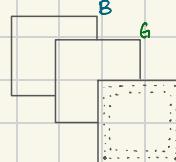
$$f \in \{0, 1, \dots, n-1\}$$

$$c \in \{0, 1, \dots, m-1\}$$

$$i_k \in \{0, 1\} \quad k=0, \dots, m-1$$

2- El estado es la imagen

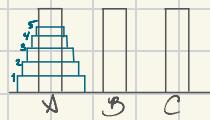
(cada estado es un pixel)



$$|S| = n * m * 2^{n*m} = (n * m)$$

$$8 \times 3 \times 1080 \times 1920$$

3- Juego: Torre de Hanói

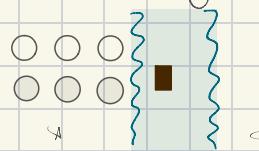


$$S = [s_1, \dots, s_6]$$

$$s_i \in \{A, B, C\}$$

$$|S| = 3^3 = 243$$

4- Juego: Monos y caníbales

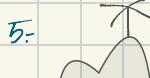


$$S = [E, O, L]$$

$$E = \{0, 1, 2, 3\}$$

$$O = \{0, 1, 2, 3\}$$

$$L = \{D, I\}$$



$$X = (h, \theta_f, \phi_f, \text{lat}, \text{long}, \theta_f, \phi_f, \text{lat}, \text{long}, h)$$

$$X \in \mathbb{R}^{10}$$

Porque hay un no finito de estados.

Cómo se modela un entorno?

Implícita definición:

- Variables de estado
- Estado actual
- Percepciones
- Acciones
- Acciones legales
- Probabilidades (si es estocástico)

Típos de entornos:

Totalmente observable vs parcialmente obs. Un entorno es totalmente obs. cuando los sensores del agente le permiten acceder al estado completo del entorno en cada momento. Si el agente no tiene acceso a toda la inf. se considera parcialmente obs.

Determinista vs. Estocástico: En un entorno determinista, el sig. estado está determinado por el estado actual y la acción ejecutada por el agente. Si existe incertidumbre el entorno es estocástico.

Estático vs. dinámico: Un entorno es estático si no cambia mientras el agente decide. Si el entorno cambia por sí solo o mientras el agente "piensa" → es dinámico.

Discreto vs. continuo: Un entorno discreto tiene un número limitado y definido de estados y acciones.

14/01/2026

Entorno:

Determinista, discreto, estático, obs.

fijo y conocido

$S = f(a)$ → acción → AEA, $a \in A(a)$ → acciones legales

$$a = (a_1, \dots, a_n) \in D_1 \times \dots \times D_n$$

Determinista, dinámico, obs, discreto

$$S_{k+1} = f(S_k, a_k)$$

—, —, parcialmente obs., —

$$S_{k+1} = f(S_k, a_k)$$

Continuo:

$$P_k = g(S_k)$$

$$\dot{x}_i = f(x_i, a_i)$$

$$P_i = g(x_i)$$

Estático, discreto, estático:

$$S = \{S^{(0)}, \dots, S^{(M)}\}, M = \text{card}(S)$$

$$P_1[S^{(0)} | a]$$

$$P_1[S^{(1)} | a]$$

$$P[S/a] = \begin{matrix} \vdots \\ P_1[S^{(M)} | a] \end{matrix}$$

$$P_1[S^{(0)} | a]$$

Qué pasa si se tiene

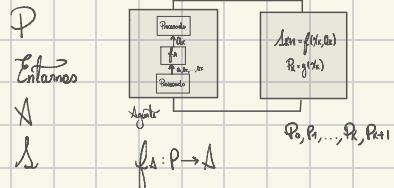
un sistema estocástico, dinámico → se puede conocer la discreta y completamente obs? Por de S_{k+1}

$$P_1[S_{k+1} = S^{(0)} | S_k, a_k]$$

$$P_1[S_{k+1} = S^{(1)} | S_k, a_k] = P_1[S_{k+1} = S^{(2)} | S_k, a_k]$$

$$P_1[S_{k+1} = S^{(m)} | S_k, a_k]$$

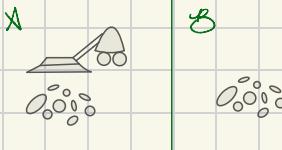
$$\text{descripción } a_k \text{ de } f(a_k)$$



Agentes inteligentes

15/01/2026

Un agente inteligente es una entidad que percibe su entorno mediante sensores y actúa sobre él mediante actuadores, eligiendo acciones racionales para maximizar una medida de desempeño (utilidad).



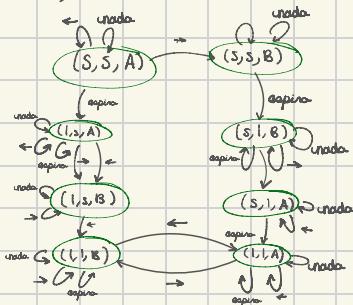
descripción observable, determinista

$$\text{Representación de forma simbólica} \quad \begin{aligned} \mathcal{A} &= \{A, B, R\} \\ A &= (X, B, R) \end{aligned}$$

actuadores

$$A = \{\leftarrow, \rightarrow, \text{aspira}, \text{nada}\}$$

$$P = (R, A|R)$$



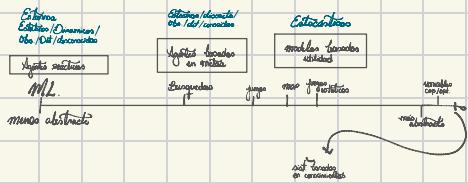
16/01/2026

Agentes de reflejo simple: Actúan basándose únicamente en la percepción actual, ignorando el resto del historial de percepciones.

Agentes de reflejo con estado: Estos agentes mantienen un estado interno y usan un modelo del mundo.

Agentes basados en metas: Saben a dónde querer llegar, tienen un objetivo claro y conocen el estado actual.

Agentes basados en utilidad: Eligen la mejor opción, pueden manejar metas en conflicto e incoherentes (intentan hacerlo de la manera más eficiente).



Entorno:

19/01/2026

$S = \text{Estado}$

$$A = (A_1, \dots, A_n) \in D_1 \times \dots \times D_n = A$$

$$A_i \in A = \{a_1, \dots, a_n\}$$

$$a_i \in A(S_t) \text{ acciones legales}$$

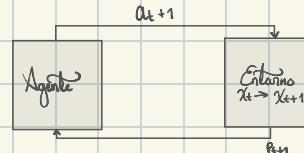
$P_t = S$ espacio de percepciones

$P_t = \text{percepción}(S_t)$

percepción: $S \rightarrow P$

$$A_{t+1} = \text{Acción}(a_t, A_t)$$

$$C = \text{costo}(S_t, a_t, S_{t+1})$$



class Entorno:

def acción_legal(self, s, a):

return True

def transición(self, s, a):

pass

def percepción(self, s):

return s

class Agente(object):

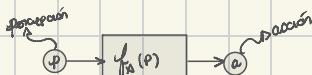
def programa(self, p):

pass

def simulador(entorno, agente, s, T=10, c=0):

Agentes reactivos

20/01/2026



f_θ es desconocida

$$f: x \rightarrow y$$

Tipicamente $x \in \mathbb{R}^n$

$$x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$$

Si $y \in \mathbb{R}$ regresión

Si $y \in \mathbb{R}^k$ clasificación binaria

Si $y \in \{C_1, C_2, \dots, C_k\}$ clasificación en varias clases

$\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ una muestra de X .

$$\mathcal{D} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$$

donde $y^{(i)} = f_\theta(x^{(i)}) + \epsilon$ u.a. dist. desconocida

El problema es encontrar una función $h: x \rightarrow y$

tal que $h \approx f_\theta$ $h \in \mathcal{H}$

$$\mathcal{H} = \{h: h: \mathbb{R} \rightarrow \mathbb{R} \text{ donde } h(x) = \alpha x, \forall \alpha \in \mathbb{R}\}$$

Ejemp.

$$X = [X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}, X_{11}, X_{12}]$$

$$X^{(i)} \in \mathbb{R}^2$$

$$\begin{bmatrix} X_1^{(1)} & \dots & X_{12}^{(1)} \\ X_1^{(2)} & \dots & X_{12}^{(2)} \\ \vdots & & \vdots \\ X_1^{(m)} & \dots & X_{12}^{(m)} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

x y

matriz de diseño

$$h: x \times \Theta \rightarrow y, \Theta = \mathbb{R}$$

$$h(x^{(i)}, \Theta) = \hat{y}^{(i)}$$

$$h_\Theta(x^{(i)}, \Theta) = \hat{y}^{(i)}$$

Si Θ es un vector de parámetros f_θ .

$$h_\Theta: x \rightarrow y$$

Ejemplo:

polinomio:

$$h_\Theta(x) = \sum_{j=1}^J \omega_j x^j + \omega_0, x^j \in \mathbb{R}$$

$$h_\Theta: \mathbb{R} \rightarrow \mathbb{R}$$

$$\omega = (\omega_0, \omega_1, \dots, \omega_J) \in \mathbb{R}^{J+1}$$

$$\hat{y} = \omega_0 + \omega_1 x + \omega_2 x^2 + \dots + \omega_J x^J$$



22/01/2026

Aprendizaje supervisado

Se presenta como un problema de optimización cuyo objetivo es encontrar un vector de parámetros

(Θ) para un modelo o hipótesis (h_Θ) de tal manera que, ante una entrada (x), la salida sea lo más cercana posible a un valor esperado (y).

Hipótesis:

1. $f: x \rightarrow y$ existe y es conocida

2. Tengo $\{x^{(1)}, \dots, x^{(m)}\} \subseteq X$ una muestra de M de datos con distribución desconocida.

3. $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$, donde

$$y^{(i)} = f(x^{(i)}) + \epsilon^{(i)}$$
 donde $\epsilon^{(i)}$ son u.a.

4. Tenemos una función "parametrizada" $h: x \times \Theta \rightarrow y$, típicamente $\Theta = \mathbb{R}^p$

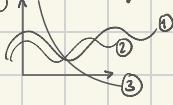
5. Para un valor específico de Θ , $h: x \rightarrow y$

6. $h_\Theta \in \mathcal{H}$ conj. de hipótesis.

7. El aprendizaje supervisado consiste en encontrar $\hat{\Theta} \in \mathcal{H}$.

Aprendizaje no optimización: La diferencia fundamental es la generalización. Mientras que la optimización busca minimizar el error dentro de los datos conocidos (Edu) el aprendizaje supervisado busca que el modelo funcione correctamente con datos nuevos, es decir,

minimizar el error fuera de la muestra (Eout).



la distancia debe ser pequeña para decir que h se parece a f.

función de error:

$L: y, \hat{y} \rightarrow \mathbb{R}$ función de pérdida.

$$L(y, \hat{y}) = L(f(x), \hat{f}^*(x))$$

$$L(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2 \quad (\text{la distancia})$$

$$= |y - \hat{y}|$$

$$L(y, \hat{y}) = \begin{cases} 0 & \text{si } y = \hat{y} \\ 1 & \text{si } y \neq \hat{y} \end{cases} \quad y \in \{-1, 1\}$$

son alternativas posibles.

$$\text{si } y \in \{-1, 1\}$$

$$E_{int}(h^*) = E_{int}[\lambda_1(f(x)), \lambda_1^*(x)] \quad \text{Error fuera de muestra}$$

$$f \approx h^* \text{ si } y \text{ solo es } \pm 1 \Rightarrow E_{int}(h^*) \approx 0$$

$$E_{in}(h^*) = \frac{1}{m} \sum_{i=1}^m L(y^{(i)}, h^*(x^{(i)}))$$

ya sea alternativa o modelo

$$f \approx h^* \text{ si } \begin{cases} E_{in}(h^*) = 0 \\ E_{in}(h^*) \approx E_{out}(h^*) \end{cases}$$

Desigualdad de Hoeffding

$$P(|E_{out}(h^*) - E_{in}(h^*)| > \epsilon) \leq e^{-2EM}$$

donde M es el tamaño de la muestra

dice $H \approx \#$ parámetros independientes

Ej) aprendizaje es posible si:

$$10 * \text{diseño}(H) \ll M \iff E_{in}(h^*) \approx E_{out}(h^*)$$

22/01/2024

Modelo lineal

$$\begin{aligned} X^{(1)^T} &\rightarrow \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \end{bmatrix} & y^{(1)} & \quad X^{(1)} \in \mathbb{R}^n \\ X^{(2)^T} &\rightarrow \begin{bmatrix} x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} \end{bmatrix} & y^{(2)} & \\ \vdots & \vdots & \vdots & \\ X^{(m)^T} &\rightarrow \begin{bmatrix} x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} \end{bmatrix} & y^{(m)} & \end{aligned}$$

$X \in \mathbb{R}^{m \times n}$

matriz de diseño

$$\mathcal{D} = \{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$$

Ejemplo sencillo:

$$h_0(x) = w_0 x_0 + w_1 x_1 + \dots + w_n x_n + b = \sum_{j=0}^n w_j x_j + b = \vec{w}^T \vec{x} + b$$

si $x = (x_1, \dots, x_n) \in \mathbb{R}^n$

$$\vec{w} = (w_0, \dots, w_n) \in \mathbb{R}^n$$

23/01/2024

$$\Theta = (w_0, \dots, w_n, b) \in \mathbb{R}^{n+1}$$

Aprendizaje (tener suficientes datos)

$$\check{E}_{in}(h^*) \approx E_{out}(h^*)$$

$$E_{in}(h^*) \approx 0$$

$$h^* = \arg \min_{h \in \mathcal{H}} E_{in}(h) \iff \Theta^* = \vec{w}^*, b^* = \arg \min_{\substack{\vec{w} \in \mathbb{R}^n \\ b \in \mathbb{R}}} E_{in}(\vec{w}, b)$$

$$\vec{w}^*, b^* = \arg \min_{\substack{\vec{w} \in \mathbb{R}^n \\ b \in \mathbb{R}}} \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (y^{(i)} - \vec{w}^T \vec{x}^{(i)} - b)^2 \quad \text{MSE (mean squared error)}$$

$$\vec{w}^*, b^* = \arg \min_{\substack{\vec{w} \in \mathbb{R}^n \\ b \in \mathbb{R}}} \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (y^{(i)} - \vec{w}^T \vec{x}^{(i)} - b)^2$$

$$= \arg \min_{\substack{\vec{w} \in \mathbb{R}^n \\ b \in \mathbb{R}}} \frac{1}{2m} \left[(y^{(1)} - \vec{w}^T \vec{x}^{(1)} - b)^2 + (y^{(2)} - \vec{w}^T \vec{x}^{(2)} - b)^2 + \dots + (y^{(m)} - \vec{w}^T \vec{x}^{(m)} - b)^2 \right]$$

$$= \arg \min_{\substack{\vec{w} \in \mathbb{R}^n \\ b \in \mathbb{R}}} \frac{1}{2m} \left[\underbrace{y^{(1)} - \vec{w}^T \vec{x}^{(1)} - b}_{\text{error}} \right]^T \left[\underbrace{y^{(2)} - \vec{w}^T \vec{x}^{(2)} - b}_{\text{error}} \right]$$

$$\text{Calcular el gradiente: } \frac{\partial J(\vec{w}, b)}{\partial \vec{w}} = 2 \vec{v}(\vec{w}) = 2 \vec{v}(\vec{w}) = \frac{\partial J(\vec{w}, b)}{\partial \vec{w}}$$

$$\frac{1}{2m} \left[\underbrace{y^{(1)} - \vec{w}^T \vec{x}^{(1)} - b}_{\text{error}} \right]^T \left[\underbrace{y^{(2)} - \vec{w}^T \vec{x}^{(2)} - b}_{\text{error}} \right]$$

$$\frac{1}{m} \vec{X}^T \vec{X} \left[\underbrace{y^{(1)} - \vec{w}^T \vec{x}^{(1)} - b}_{\text{error}} \right] = \vec{0}$$

$$= \vec{X}^T \left[\underbrace{y^{(1)} - \vec{w}^T \vec{x}^{(1)} - b}_{\text{error}} \right] = \vec{0}$$

$$= \vec{X}^T \vec{y} - \vec{X}^T \vec{w} - \vec{X}^T b = \vec{0}$$

$$\vec{X}^T \vec{w} = \vec{X}^T \vec{y} - \vec{X}^T b$$

$$\vec{w} = [\vec{X}^T \vec{X}]^{-1} \vec{X}^T \vec{y}$$

Punto (1)

Tarea de agente

tray 2 cuartos: A y B {limpio, sucio}

El robot puede moverse entre habitantes, limpiar/nada.

Estado:

(robot, A, B) El robot está en el cuarto A, A está (limpio/sucio)

El entorno solo cambia cuando el agente hace una acción.

Percepción:

El agente no ve todo el estado, solo ve: (robot, estado del cuarto actual)

El agente no sabe cómo está el otro cuarto.

Acciones:

ir A	mueve al robot al cuarto A
ir B	mueve el robot al cuarto B
limpiar	limpia el cuarto donde está
nada	nada

Limpio/sucio
 legal

Transiciones

Dado un estado actual y una acción

El entorno decide el nuevo estado y el costo de la acción.

Cada acción cuarto 1, excepto cuando todo está limpio y se elige nada.

Simulador

Toma el estado actual

Pide la percepción del entorno

Se lo pasa al agente

El agente elige una acción

El entorno calcula el nuevo estado

Se actualiza el costo

Repita

Acciones

ir derecho	mueve al robot al cuarto A
ir izq	mueve al robot al cuarto B
subir	legal en los primeros dos pisos (cuarto de la derecha)
bajar	legal en los primeros dos pisos (cuarto de la izquierda)
limpiar	limpia el cuarto donde está
nada	nada

Las acciones de subir y bajar son más costosas que las acciones: ir-derecho, ir-izq.

El costo de limpiar < al costo de cualquier acción

Agente aleatorio:

Pierde mucho tiempo repitiendo acciones útiles (como nada o limpiar un cuarto ya limpio). Requiere de más pasos y más costo, porque no tiene estrategia en memoria.

Agente racional modul:

A diferencia del agente aleatorio este agente si usa memoria para recordar qué cuartos ya vio y en qué estados estaban.

Requiere de menos costo y termina antes que el aleatorio.

Nuevos Cuartos Ciegos:

El agente solo sabe en qué cuarto está, no sabe si está limpio o sucio.

Modelo lineal

+

$$\hat{y} = h_{\theta}(x) = w_1 x_1 + \dots + w_n x_n + b$$

$$\Theta^* = \arg \min_{\Theta} \frac{1}{2M} \sum_{i=1}^M (\hat{y}^{(i)} - [x^{(i)}]^\top [\begin{bmatrix} w \\ b \end{bmatrix}])^2$$

MAE

$$X_e = \begin{bmatrix} x_1^{(1)} & \dots & x_n^{(1)} & 1 \\ \vdots & & \vdots & \vdots \\ x_1^{(m)} & \dots & x_n^{(m)} & 1 \end{bmatrix}$$

$$y \in \mathbb{R}$$

$$\Theta = (w_1, \dots, w_n, b) \in \mathbb{R}^{n+1}$$

$$\hat{y} = X_e \Theta$$

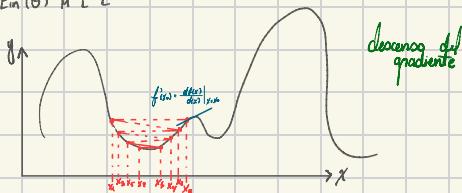
$$(w_{j,1}) \quad \dots \quad (w_{j,n+1}) \quad (w_{j,1}, 1)$$

$$(w_{m,1}) \quad \dots \quad (w_{m,n+1})$$

$$E = \hat{y} - y$$

$$(w_{j,1}) \quad (w_{j,1}) \quad (w_{m,1})$$

$$E_{in}(\Theta) = \frac{1}{M} E^T E$$



$$x_1 \leftarrow x_1 - \eta f'(x_0) \\ x_2 \leftarrow x_2 - \eta f'(x_1) \Rightarrow x_{k+1} \leftarrow x_k - \eta f'(x_k)$$

nº 7: Fase de aprendizaje

$$\Theta_{k+1} \leftarrow \Theta_k - \eta \nabla_{\Theta} f(\Theta)|_{\Theta=\Theta_k}$$

$$x_{k+1} \leftarrow x_k - \eta \frac{\partial E_{in}(w_k, b_k)}{\partial b}$$

$$w_{j,k+1} \leftarrow w_{j,k} - \eta \frac{\partial E_{in}(w_k, b_k)}{\partial w_j}$$

Ecuación del descenso del gradiente

$$\Theta_{k+1} \leftarrow \Theta_k - \eta \nabla_{\Theta} f(\Theta_k)$$

$$h_{\theta}(x) = w_1 x_1 + w_2 x_2 + \dots + w_n x_n + b = w^T x + b = [x^T, 1] [\begin{bmatrix} w \\ b \end{bmatrix}]$$

$$= [x^T, 1] \Theta$$

$$\Theta^* = w, b = \arg \min_{\Theta \in \mathbb{R}^{n+1}} \frac{1}{2M} \sum_{i=1}^M (\hat{y}^{(i)} - [x^{(i)}]^\top \Theta)^2$$

para w_j

$$\frac{\partial E_{in}(\Theta)}{\partial w_j} = \frac{1}{2M} \sum_{i=1}^M -2(y^{(i)} - \hat{y}^{(i)}) x_j^{(i)}$$

$$\frac{\partial E_{in}(\Theta)}{\partial b} = \frac{1}{2M} \sum_{i=1}^M -2(y^{(i)} - \hat{y}^{(i)})$$

Algoritmo de aprendizaje:

#on numpy

```
def dgf_lm(X, y, w0, b0, lr, max_epochs, e_tol):
    M = X.shape[1]
```

```
w = w0.copy()
```

```
b = b0.copy()
```

aprendizaje

```
hist = []
```

```
for _ in range(max_epochs):
```

```
y_est = X @ w + b
```

```
Err = y - y_est
```

```
hist.append(np.square(Err).mean())
```

```
grad_w = -(1/M) X.T @ Err
```

```
d_b = Err.mean()
```

```
w = lr * grad_w
```

```
b = lr * d_b
```

```
if np.abs(grad_w).max() < e_tol:
```

```
break
```

```
return w, b, hist
```

aprende cuando: $E_n \approx 0$

$f \approx h^*$

\downarrow

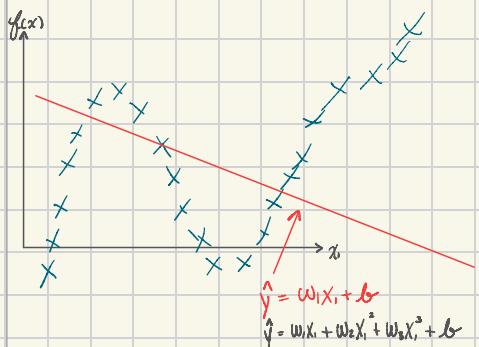
$E_{out} \approx E_{in}$

$E_{out} \approx 0$

Para que E_{out} se acerque a E_{in} debe de haber muchos datos

Probablemente Aproximadamente Correcto

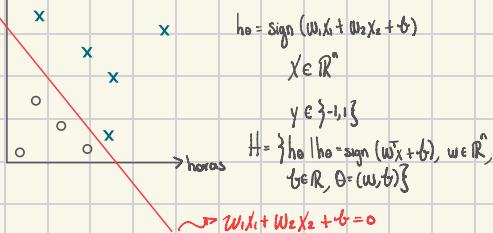
$P(|E_{out} - E_{in}| \geq \epsilon) \leq \delta$



$$\begin{array}{l}
 x_1 \\
 x_2 = x_1^2 \\
 x_3 = x_1^3
 \end{array}
 \quad
 \begin{array}{c}
 x_1 \quad x_2 \quad x_3 \quad y \\
 x_1^{(1)} \quad x_2^{(1)} \quad x_3^{(1)} \quad y^{(1)} \\
 \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\
 x_1^{(m)} \quad x_2^{(m)} \quad x_3^{(m)} \quad y^{(m)}
 \end{array}$$

Clasificación lineal

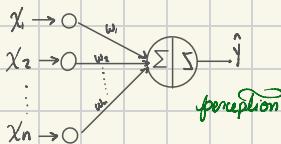
Asistente



función de pérdida:

$$\text{loss}(y, \hat{y}) = \begin{cases} 1 & \text{si } y \neq \hat{y} \\ 0 & \text{en otros casos} \end{cases} = \max(-y, \hat{y}, 0)$$

$$E_{\text{in}}(w, b) = \frac{1}{M} \sum_{i=1}^M \text{loss}(y^{(i)}, \text{sign}(w^T x^{(i)} + b))$$



Regresión logística

09/01/26

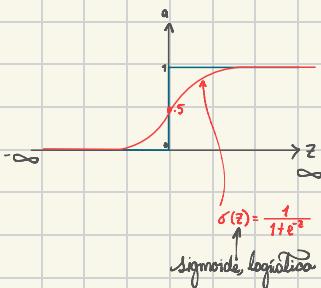
$$\hat{y} = \Pr(y=1 | x; \theta) = \sigma(w^T x + b)$$

$$\begin{cases} 1 & \text{si } \theta > h \\ 0 & \text{en otros casos} \end{cases}$$

clase

$$a = f(w_1 x_1 + w_2 x_2 + \dots + w_n x_n + b)$$

$$Z = w_1 x_1 + w_2 x_2 + \dots + w_n x_n + b = X^T w + b = X e^T \theta$$



$$\begin{array}{ccc}
 x_1^{(1)} & \dots & x_n^{(1)} & | & y^{(1)} & | & a^{(1)} \\
 x_1^{(2)} & \dots & x_n^{(2)} & | & y^{(2)} & | & a^{(2)} \\
 \vdots & & \vdots & | & \vdots & | & \vdots \\
 x_1^{(m)} & \dots & x_n^{(m)} & | & y^{(m)} & | & a^{(m)}
 \end{array}$$

x y A

$$y^{(i)} \in \{-1, 1\}$$

$$E_{\text{in}}(w, b) = \frac{1}{M} \sum_{i=1}^M \text{loss}(a^{(i)}, \hat{a}^{(i)})$$

$$\text{loss}(a^{(i)}, \hat{a}^{(i)}) = \begin{cases} -\log(\hat{a}^{(i)}) & \text{si } a^{(i)} = 1 \\ -\log(1-\hat{a}^{(i)}) & \text{si } a^{(i)} = 0 \end{cases}$$

fución de pérdida

$$\text{loss}(a^{(i)}, \hat{a}^{(i)}) = -a^{(i)} \log(\hat{a}^{(i)}) - (1-a^{(i)}) \log(1-\hat{a}^{(i)})$$

$$W \leftarrow W - \eta_r \nabla E_{\text{in}}(w, b)$$

$$b \leftarrow b - \eta_r \frac{\partial}{\partial b} E_{\text{in}}(w, b)$$

$$\frac{\partial}{\partial w_j} E_{\text{in}}(w, b) = \frac{\partial}{\partial w_j} \frac{1}{M} \sum_{i=1}^M -a^{(i)} \log(\hat{a}^{(i)}) - (1-a^{(i)}) \log(1-\hat{a}^{(i)})$$

$$\begin{aligned}
 \text{donde } \hat{a}^{(i)} &= \frac{1}{1+e^{-z^{(i)}}} \quad z^{(i)} = w_1 x_1^{(i)} + \dots + w_n x_n^{(i)} + b \\
 -\frac{1}{M} \sum_{i=1}^M \frac{\partial a^{(i)}}{\partial w_j} &= \frac{\partial a^{(i)}}{\partial w_j} + \frac{1-a^{(i)}}{1-\hat{a}^{(i)}} \frac{\partial \hat{a}^{(i)}}{\partial w_j} \\
 &\downarrow
 \end{aligned}$$

$$= \frac{1}{M} \sum_{i=1}^M [-a^{(i)}(1-\hat{a}^{(i)}) + (1-a^{(i)})\hat{a}^{(i)}] x_j^{(i)}$$

$$= \frac{1}{M} \sum_{i=1}^M [-a^{(i)} + \hat{a}^{(i)} - a^{(i)}\hat{a}^{(i)} - \hat{a}^{(i)}\hat{a}^{(i)}] x_j^{(i)}$$

$$\frac{\partial \sigma(z^{(i)})}{\partial w_j} = \frac{\partial \sigma(z^{(i)})}{\partial z^{(i)}} \frac{\partial z^{(i)}}{\partial w_j}$$

$$W \leftarrow W - \frac{\eta_r}{M} X^T (A - \hat{A})$$

$$b \leftarrow b - \frac{\eta_r}{M} \sum_{i=1}^M (a^{(i)} - \hat{a}^{(i)})$$

$$\frac{\partial \hat{a}^{(i)}}{\partial w_j} = \frac{\partial}{\partial w_j} \frac{1}{1+e^{-z^{(i)}}}$$

$$= \frac{\partial}{\partial z^{(i)}} (1+e^{-z^{(i)}}) \frac{\partial z^{(i)}}{\partial w_j}$$

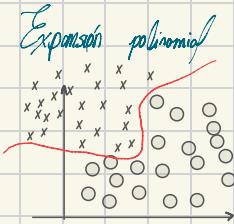
$$= (1+e^{-z^{(i)}})^{-2} e^{-z^{(i)}} x_j^{(i)}$$

$$= \frac{1+e^{-z^{(i)}}}{(1+e^{-z^{(i)}})^2} = \frac{1}{1+e^{-z^{(i)}}}$$

$$= \left(\frac{1}{1+e^{-z^{(i)}}} - \frac{1}{1+e^{-z^{(i)}}} \right) x_j^{(i)}$$

$$= (\hat{a}^{(i)} - \hat{a}^{(i)} \hat{a}^{(i)}) x_j^{(i)}$$

$$= \hat{a}^{(i)} (1-\hat{a}^{(i)}) x_j^{(i)}$$



$$\mathbb{R}^n \rightarrow \mathbb{R}^{n^2}$$

$$\phi(x) = (x, x^2)$$

$$X = (X_1, X_2)$$

$$d(X) = (X_1, X_2, X_1^2, X_1 X_2, X_2^2)$$

$$d(X) = (X_1, X_2, X_1^2, X_1 X_2, X_2^2, X_1^2 X_2, X_1 X_2^2, X_2^3)$$

$$X = (X_1, X_2, X_3, X_4)$$

$$\phi(X) = (X_1, X_2, X_3 X_4, X_1^2, X_1 X_2, X_1 X_3, X_1 X_4, X_2^2, X_2 X_3, X_2 X_4, X_3^2, X_3 X_4, X_4^2)$$

V_C = uno de parámetros (dimensiones)

$H = \{h_j\}$ daf que

$$h(x) = W_1 X_1 + W_2 X_1^2 + W_3 X_1^3 + W_4 X_1^4 + W_5 X_1^5 + W_6 X_1^6 + W_7 X_1^7 + W_8 X_1^8 + b$$

$\hookrightarrow V_C = 9$

Otro ejemplo:

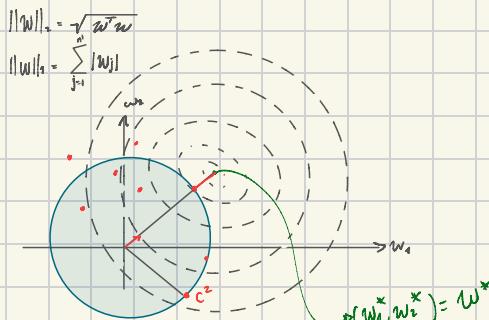
$$h(x) = W_1 X_1 + W_2 X_1^2 + W_3 X_1^3 + W_4 X_1^4 + W_5 X_1^5 + W_6 X_1^6 + W_7 X_1^7 + W_8 X_1^8 + b$$

(solo $w_j = 0$ si $j \geq 5 \rightsquigarrow V_C = 3$)

Otro ejemplo:

$$h(x) = W^T \phi(x) + b, \phi(x) \in \mathbb{R}^n, b \in \mathbb{R} \rightsquigarrow V_C = n+1$$

bajo $\sum_{j=1}^n w_j^2 \leq C$ Regularización

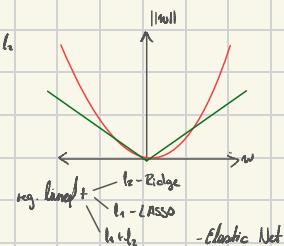


$$\text{Wres bres} = \arg \min [E_{\text{reg}}(w, b) + \frac{\lambda}{2} \sum_{j=1}^n w_j^2] \quad \text{donde } c > 0$$

$$\frac{\partial}{\partial w_j} [E_{\text{reg}}(w, b) + \frac{\lambda}{2} \sum_{j=1}^n w_j^2] = -\frac{1}{n} \sum_{i=1}^n (y_i - w^T x_i) x_i^{(j)} + \frac{\lambda}{2} j w_j$$

$$w_{\text{reg}}, b_{\text{reg}} = \arg \min [E_{\text{reg}}(w, b) + \frac{\lambda}{2} \text{reg}(w)]$$

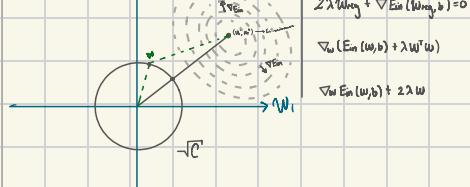
$$\begin{aligned} \text{reg}(w) = & \sum_{j=1}^n w_j^2 = \|w\|_2^2, \\ & \sum_{j=1}^n |w_j| \quad l_1 \\ & \text{reg. Lasso} + l_1 - \text{LASSO} \\ & l_1 + l_2 \end{aligned}$$



$$W_{\text{reg}} = -K \nabla_w E_{\text{reg}}(w, b) \quad \text{MN2}$$

$$W_{\text{reg}} + W_{\text{reg}}^T = 0$$

$$\nabla_w E_{\text{reg}}(w, b) + 2\lambda w = 0$$



¿Qué es el sesgo cognitivo?

Existe un patrón

No es posible establecerlo de forma analítica

Decision trees

Each internal node tests an attribute x_i .

One branch for each possible attribute value $x_i = v$.

Each leaf assigns a class y .

To classify input x :

Follow the tree from root to leaf
 $f: x \rightarrow y$

y que no agotar

$h = x \rightarrow y$ a partir de

$$\mathcal{D} = \{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$$

$x^{(1)}$	$x_1^{(1)}$	$x_2^{(1)}$	\dots	$x_n^{(1)}$	$y^{(1)}$
$x^{(2)}$	$x_1^{(2)}$	$x_2^{(2)}$	\dots	$x_n^{(2)}$	$y^{(2)}$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
$x^{(m)}$	$x_1^{(m)}$	$x_2^{(m)}$	\dots	$x_n^{(m)}$	$y^{(m)}$

```
class SearchPb(object)
```

```
def __init__(self, s_m):
    self.s = s_m
    self.acciones = []
    raise NotImplementedError('A desarrollar tbv')
```

```
def sucesor(self, s, a):
    raise NotImplemented...
```

```
def terminal(self, s):
    raise NotImplemented...
```

```
class TorreHanoi(SearchPb):
```

```
def __init__(self, s0 = 5*[['A'], 'C'], sf = 5*[['C']]):
    self.s0 = s0
    self.sf = sf
    self.acciones = ['AB', 'AC', 'BC', 'CA', 'CB']
```

```
def acciones(self, s):
    if any([si not in ['A', 'B', 'C'] for si in s]):
        raise ValueError('Estado %s impossible')
```

```
return [a in self.acciones if a[0] in s and a[1] in s or
       s.index(a[0]) < s.index(a[1])]
```

```
def sucesor(self, s, a):
```

```
costo_local = 1
```

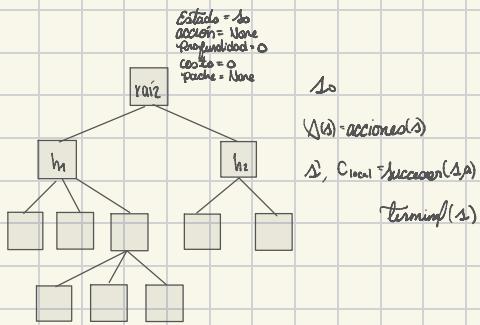
```
if a not in self.acciones(s):
    raise ValueError('...')
```

```
s.next = s[:]
```

```
s.next[s.index(a[0])] = a[1]
```

```
return s.next, costo_local
```

```
def terminal(self, s):
    return s == self.sf.
```



```
def __init__(self, s, a=None, padre=None, costo_L=None):
```

```
self.s = s
```

```
self.a = a
```

```
self.padre = padre
```

```
self.d = 0, f_padre == None else self.padre.d + 1
```

```
self.costo = 0, f_padre == None else costo_L + self.padre.costo
```

```
def expande(self, search_pb):
```

```
for a in search_pb.acciones(self.s):
```

```
s_n, costo_L = search_pb.sucesor(s,a)
```

```
yield NodoSearch(s_n, a, self, costo_L)
```

```
def plan(self)
```

```
return [[self], None, None] if self.padre == None else
```

```
[self.padre.plan()[-1] + [(self.padre.s, self.a, self.costo), (self, None, None)]]
```

```
def busquedageneralizada(s, pb):
```

```
frontera = [NodoSearch(s)]
```

```
while frontera:
```

```
plan = fronte.nodo(frontera)
```

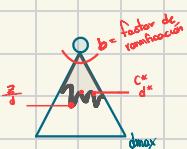
```
if pb.terminal(plan.s):
```

```
return plan
```

```
for plan.hijo in plan.expande(pb):
```

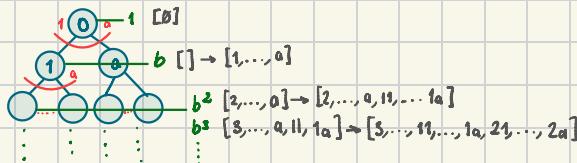
```
agrega_a_frontera(frontera, plan.hijo)
```

```
return None
```



Algoritmo:

- Admisibilidad
- Optimalidad
- Complejidad material (*Como hace las cosas los algoritmos*)
- Complejidad temporal



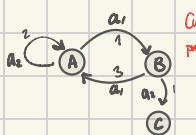
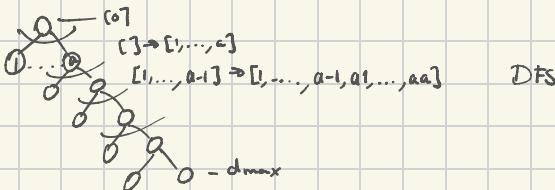
Si el plan admisible de menor profundidad tiene profundidad d entonces vamos a revisar:

$$1 + b + b^2 + \dots + b^d \text{ que es } O(b^d)$$

Primero a lo ancho:

BFS

Admisible? Si
Óptimo? Solo si $C=1$
Temporal: $O(b^d)$
Material: $O(b^{d-1})$



Cualquier grafo con ciclos, su profundidad es infinita.

¿Admisible? Si d_{\max} es ∞ no

¿Óptimo? No

¿Temporal? $O(b^{d_{\max}})$

¿Material? $O(b * d_{\max})$