Inteligencia

Inteligencia Artificial

Curso: https://ia-unison.github.io

¿Qué es la inteligencia artificial?

Sistemas que...
- Piensan como humanos
- Actúan como humanos
- Piensan racionalmente
- Actúan racionalmente

Matriz

para i de 1 a n — Si n=20, 400 operaciones
para j de 1 a n — Si n=100, 10,000 operaciones

nxn

Problema del viajero

- Permutaciones n!

5 ciudades, 120 operaciones

Ciencia – Explican los por qué
Ingeniería – Aplicación de conocimientos científicos en la resolución de problemas

Concepto importante

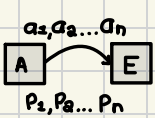
Racional
- Con los sensores disponibles
- Con los actuadores
- Con los modelos

Definición: Maximizar la esperanza de una utilidad futura

13/Enero/26

¿Qué es un agente?

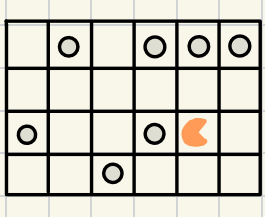
Se comunica con un entorno o ambiente



PEAS
- Sensores
- Actuadores
- Entorno
- Performance

Estado S

$S = (S_1, S_2, \dots, S_n) \in D_1 \times D_2 \times \dots \times D_n$



$S = (f, c, i_0, \dots i_{n \times m-1})$

i_n = Indicador si hay punto en la posición

$C = n // m$

$f = n \div m$ módulo

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

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

$i_n \in \{0, 1\} \quad \forall n = 0_1, \dots, m \times n-1$

¿Cuántos valores puede tener f? n

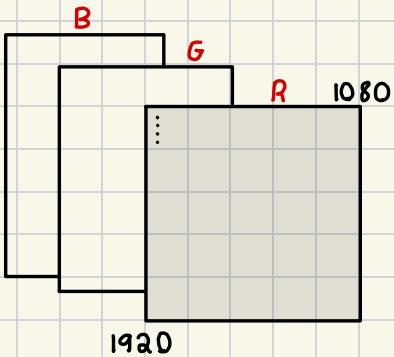
$|S| = n \times m \times 2^{n \times m} - (n \times m)$ 30mil millones - 30

$\hookrightarrow 2^{30}$

Problema: Imagen si es rostro o no

Estado: Imagen

Compuesta: Pixeles

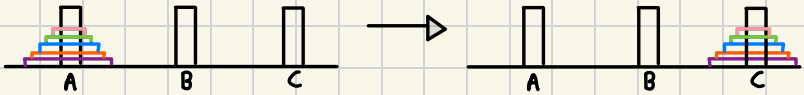


Pixeles: 8 bit

$2^{8 \times 3 \times 1080 \times 1920} \rightarrow 256 = 2^8$

Entrada: Cualquier imagen El juego tiene 30×10^{30}

Ejemplo: Torres de Janoi 5 DISCOS



$S = [S_1, \ldots, S_5]$
$S_i \in \{A, B, C\}$
$|S| = 3^5 = 243$

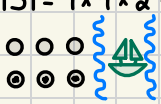
Ejemplo Misioneros y caníbales

↳Barco: 1 o 2 personas

Necesitamos transportar a 6 personas y 2 lados

Estados:
$S = [E, O, L]$
$E = \{0, 1, 2, 3\}$
$O = \{0, 1, 2, 3\}$
$L = \{D, I\}$

$|S| = 4 \times 4 \times 2 = 32$



No puede ser 1 y 1 porque uno morirá

Ejemplo: Helicóptero

$x = \{h, \cancel{\theta_F}, \cancel{\theta_L}, lat, long, \cancel{\dot{\theta}_F}, \cancel{\dot{\theta}_L}, \dot{lat}, \dot{long}, \dot{h}\}$
$x \in \mathbb{R}^{10}$
Cardinalidad: Infinita

Modelos
- Estáticos: Se resuelve directo (Rec. facial)
- Dinámicos: Depende del estado anterior

- Modelos
  - Discretos
  - Continuos
  - Observable
  - No observable
  - Determinista → Se que pasará
  - Estocástico → Incertidumbre (Dados)
  - Episódicos → Hay forma de haber aprendizaje
  - Continuos

ENTORNO

14/Enero/26

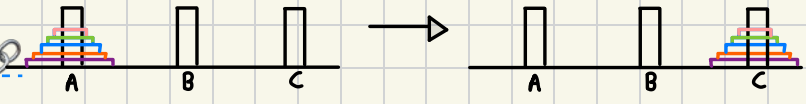
Determinista (Calcular valor exacto)
Estático
Observable
Discreto (S ∈ a un conjunto finito de estados)

$S = f(a)$ Tengo una acción y devuelve un estado

S → Estado
a → Acción

$S = (S_1, \dots, S_n) \in D_1 \times \dots \times D_n$
$a \in A$ $a \in A(s)$ acciones legales de S

Ejemplo: Torres de Janoi



$a \in \{AB, AC, BA, BC, CA, CB\}$

Det, Dinámico, Obs, Discreto
$S_{n+1} = f(S_n, a_n)$ Misioneros y canibales

Det, Obs Torres de Janoi

__, __, Parc. Obs, __

$S_{n+1} = f(S_n, a_n)$

$P_n = g(S_n)$

Continuo

$\dot{X} = f(X_t, a_t)$

$P_t = g(X_t)$

Discreto, Estático, Estocástico

$S = \{S^1, \dots, S^{(m)}\}$ $M = \text{card}(S)$

$$P(S|a) = \begin{bmatrix} Pr[S = s^{(1)} | a] \\ Pr[S = s^{(2)} | a] \\ \vdots \\ Pr[S = s^{(m)} | a] \end{bmatrix}$$

Estocástico, Dinámico

$Pr[S_{n+1} | S_n, a_n]$

Estocástico, Dinámico, Discreto, Observable → Conozco el estado

$$Pr[S_{n+1} | S_n, a_n] = \begin{bmatrix} Pr[S_{n+1} = S^{(1)} | S_n, a_n] \\ Pr[S_{n+1} = S^{(2)} | S_n, a_n] \\ \\ Pr[S_{n+1} = S^{(n)} | S_n, a_n] \end{bmatrix}$$

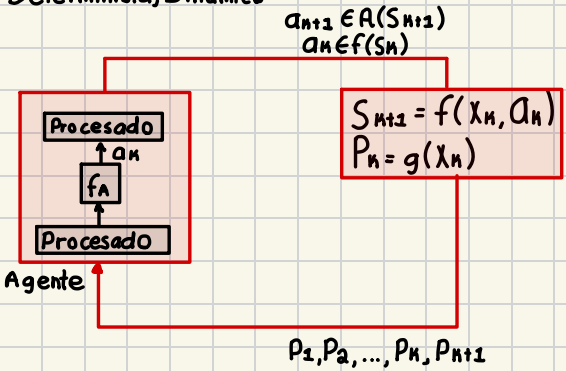
PEAS
- Sensores
- Actuadores
- Entorno → Discreto, Continuo, Determinista...
- Performance

$-\min(-a, -b) = \max(a, b)$

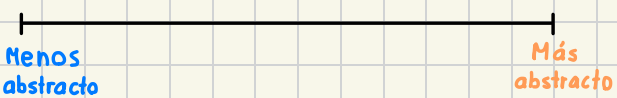
$\arg\max_x f(x) = \arg\min_x -f(x)$

} Recordatorio

Determinista, Dinámico



$f_A: P^* \rightarrow A$

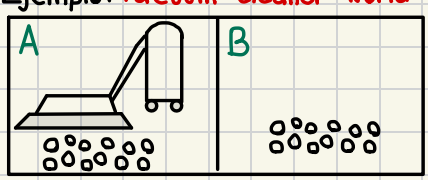


Agentes inteligentes

15/Enero/26

Agente: Caja cerrada de software

Ejemplo: Vacuum-cleaner world Parc. obs. Dinámico. Discreto. Determinista



Cuartos, Robot

$S = (A, B, R)$

$A \in \{L, S\}$

$B \in \{L, S\}$

Limpio, Sucio

8 POSIBILIDADES 2^3

$R \in \{A, B\}$

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

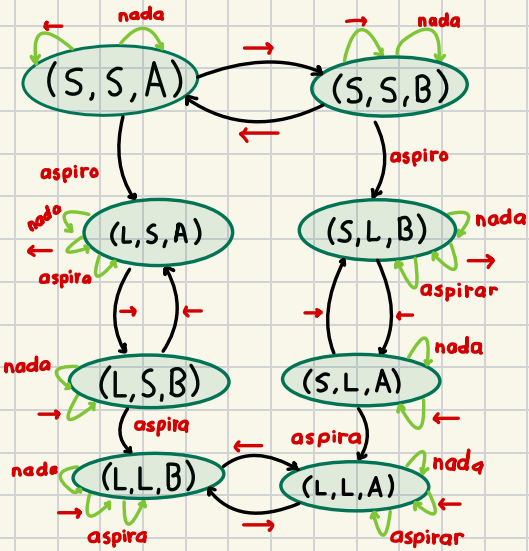
$p = (R, s[R])$

¿Qué lugar?

¿Limpio o sucio?

Medida de desempeño ¿✓ o X?

Algoritmo



Un agente racional no es perfecto

Agente reflejo

$a_k = f(P_t)$

$a_t = f(P_t, X_t)$

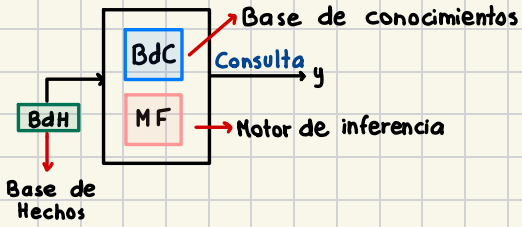
Inicializo $X_t = (S,S,A)$

```
def f(Pt,Xt)
# Actualizar Xt     Pt=(R,s[val[R]])
Xt[z]=Pt[0]
if Pt[0]='A':
   Xt[0]=Pt[0]
else
   Xt[1]=Pt[1]

if Xt[0]=Xt[1]=='L'
a=nada
elif Pt[1]=='S':
a='aspira'
elif Pt[0]=='A':
a='→'
```

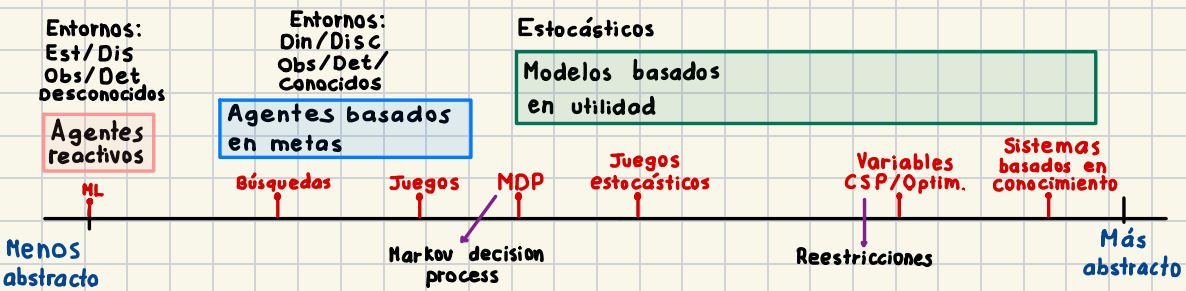
else
$a = '\leftarrow'$
$X_{t+1} \leftarrow X_t$
regreso q

En un modelo no det. → No puede haber un plan

16/Enero/25



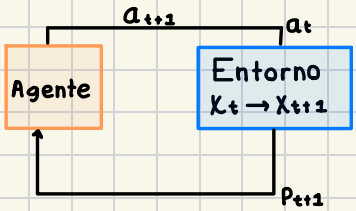
Libro: AIMA https://aima.cs.berkeley.edu/global-index.html



19/Enero/26

Entorno
S = Estado
$S = (S_1, \dots S_n) \in D_1 \times \dots D_n = S$
$A = \{a_1, \dots a_m\}$
$a_t \in A(S_t)$ acciones legales
$P_t \in \mathcal{P}$ espacio de percepciones
P_t = percepción (S_t)
Percepción: $S \rightarrow \mathcal{P}$

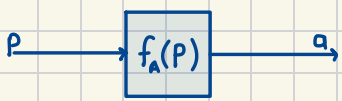
S_{t+1} = transición (S_t, a_t)
C = costo (S_t, a_t, S_{t+1})



Estados
Actuadores
Percepciones

Dos cuartos ciegos:
Ir a 'A'
Limpiar
Ir a 'B'
Limpiar
Nada

Tenemos: 20/Enero/26



f_A es desconocida
$f_A : X \rightarrow y$
↳ Salidas

Tipicamente $X = \mathbb{R}^n$
$x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$
Si $y = \mathbb{R}$ REGRESIÓN
Si $y = \{-1, 1\}$ CLASIFICACIÓN BINARIA
Si $y = \{C_1, C_2, \dots, C_n\}$ CLASIFICACIÓN EN VARIAS CLASES

$\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ una muestra de X
$D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$
donde $y^{(i)} = f_A(x^{(i)}) + \hat{e}$ V.A distribución desconocida

$f(x) = 3x$
$f(x) = -3x$
} Infinitas

El problema es encontrar una función $h^* \; x \rightarrow y$
tal que $h^* \approx f_A$ $h^* \in \mathcal{H}$ hipótesis posibles

$\mathcal{H} = \{h \mid h : \mathbb{R} \rightarrow \mathbb{R} \text{ donde } h_\alpha(x) = \alpha x, \forall \alpha \in \mathbb{R}\}$
$x = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}]$
x_1 Ingreso mensual
x_2 Cuánto debe
x_3 Si pago a tiempo
x_4 Que otros creditos hay
x_5 Aval

$x^{(i)} \in \mathbb{R}^{1a}$

$$\underbrace{\begin{bmatrix} x_1^{(1)} \dots x_{1a}^{(1)} \\ x_1^{(2)} \dots x_{1a}^{(2)} \\ \vdots \\ x_1^{(m)} \dots x_{1a}^{(m)} \end{bmatrix}}_{X} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

CLASIFICACIÓN BINARIA

V o F

$h: X \; x \longmapsto y$

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

$h_\theta(x^h) = \hat{y}^{(i)}$

Si θ es un vector de parámetros fijo

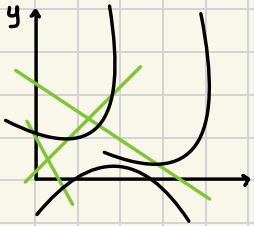
$h_\theta: X \to y$

$h(x): \sum_{j=1}^{T} w_j x^j + w_0$ $\quad x^J \in \mathbb{R}$

$h_w: \mathbb{R} \to \mathbb{R}$

$w = (w_0, w_1, \dots, w_T) \in \mathbb{R}^{T+1}$

$\hat{y} = w_1 x + w$, $\hat{y} = w_2 x^2 + w_1 + b$



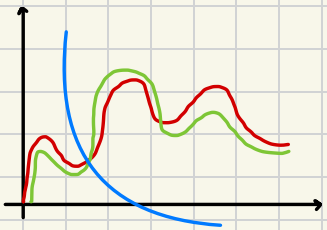
"Probablemente aproximadamente correcto" PAC-Learning

Hipótesis:

22/Enero/26

1. $f: X \to y$ existe y es desconocida
2. Tengo $\{x^{(1)}, \dots, x^{(m)}\} \subseteq X$ una muestra de m datos con distribución reconocida
3. $D = \{(x^{(1)}, y^{(i)}), \dots, (x^{(m)}, y^{(m)})\}$, donde
   $y^{(i)} = f(x^{(i)}) + e^{(i)}$ donde $e^{(i)}$ son VA
4. Tenemos una función "parametrizada" $h: X \times \Theta \to y$, típicamente $\Theta = \mathbb{R}^p$
5. Para un valor específico de Θ, $h_\theta: X \to y$
6. $h_\theta \in \mathcal{H}$ conjunto de hipótesis
7. El aprendizaje supervisado consiste en encontrar $h^* \in \mathcal{H}$ tal que $h^* \approx f$

Θ vector de parámetros



$L: y \times y \to \mathbb{R}$ función de pérdida

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

$L(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2 \qquad y \in \mathbb{R}$

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

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

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

$$E_{out}(h^*) = \underset{x \in X}{E}\left[L(f(x), h^*(x))\right]$$

↓ Error fuera de muestra

$f \approx h^*$ si y solo si $E_{out}(h^*) \approx 0$

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

↓ Error de muestra

→ Pérdida

MSE
MAE
MAPE

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

$h_\theta(x) = w_1 x_1 + w_2 x_2 + w_{3x1x2} + b$

$\theta = (w_1, w_2, w_3, b) \in \mathbb{R}^4$

Desigualdad de Hoeffding

Si tengo una esp y el prom de una muestra:

$P_\sigma(|E_{out}(h^*) - E_{in}(h^*)| > \epsilon) \leq -2\, e^{2\epsilon^2 M}$

donde M es el tamaño de la muestra

La dimensión vc

$d_{vc}(\mathcal{H}) \approx$ #parámetros INDEPENDIENTES

El aprendizaje es posible si

$10 * d_{vc}(\mathcal{H}) << M \leftrightarrow E_{in}(h^*) \approx E_{out}(h^*)$

23/Enero/26

$$\begin{matrix} & a_1 & a_2 & \dots & a_n \\ \left[\begin{matrix} X^{(1)T} \\ X^{(2)T} \\ \vdots \\ X^{(m)T} \end{matrix}\right] \rightarrow & \left[\begin{matrix} (X_1^{(1)} & X_2^{(1)} & \dots & X^{(1)}) \\ (X_1^{(2)} & X_2^{(2)} & \dots & X^{(2)}) \\ \vdots & \vdots & \dots & \vdots \\ (X_1^{(m)} & X_2^{(m)} & \dots & X^{(m)}) \end{matrix}\right] & \left[\begin{matrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{matrix}\right] \\ & \underbrace{X}_{(m,n)} & \underbrace{Y}_{(m,1)} \end{matrix}$$

$x^{(i)} \in \mathbb{R}^n$

$y^{(i)} \in \mathbb{R}$

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

$h_\theta(x) = w_1 x_1 + w_2 x_2 + \dots + w_n x_n + b = \sum_{j=1}^{n} w_j x_j + b = w^T x + b = x^T w + b$

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

$w = (w_1, \dots, w_n) \in \mathbb{R}^n$

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

Aprendizaje:

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

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

$h^* = \arg\min_{h \in \mathcal{H}} E_{in}(h) \leftrightarrow \theta^* = w^*, b^* = \arg\min_{\substack{w \in \mathbb{R}^n \\ b \in \mathbb{R}}} E_{in}(h_{w,b})$

$w^*, b^* = \arg\min_{\substack{w \in \mathbb{R}^n \\ b \in \mathbb{R}}} \underbrace{1/m}_{\text{promedio}} \sum_{i=1}^{m} 1/2\, (y^{(i)} - h_{w,b}(x^{(i)}))^2$ MSE → Mean square error

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

↓ v. constantes

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

$$\begin{bmatrix} y^{(1)} - X^{(1)T} w - b \\ y^{(2)} - X^{(2)T} w - b \\ \vdots \\ y^{(m)} - X^{(m)T} w - b \end{bmatrix}$$

$$= \arg \min_{\substack{w \in \mathbb{R}^n \\ b \in \mathbb{R}}} \frac{1}{2M} \left[y - X w - \vec{1} b \right]^T \left[y - X w - \vec{1} b \right]$$

$$= \arg \min_{\theta \in \mathbb{R}^{n+1}} \frac{1}{2M} \left[y - \underbrace{[X, \vec{1}]}_{X_e} \theta \right]^T \left[y - \underbrace{[X, \vec{1}]}_{X_e} \theta \right]$$

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} - \begin{bmatrix} X^{(1)T} \\ X^{(2)T} \\ \vdots \\ X^{(m)T} \end{bmatrix} w - \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} b$$

$$\begin{bmatrix} x_1^{m} \dots x_n^{m} & 1 \\ \vdots & \vdots \\ x_1^{(m)} \dots x_n^{(m)} & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_n \\ b \end{bmatrix}$$

$f(x_1, \dots x_n)$

$f(x) \quad x \in \mathbb{R}^n, f: \mathbb{R}^n \to \mathbb{R}$

$$\nabla_x f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}$$

$$\frac{\delta v(x)^2}{\partial x} = 2 v(x) \frac{\partial v(x)}{\partial x}$$

$$\frac{1}{2M} \left[y - X_e \theta \right]^T \left[y - X_e \theta \right]$$

$$\underset{(n+1, m)}{X_e^T} \frac{\cancel{2}}{\cancel{2}M} \underset{m, 1}{\left[y - X_e \theta \right]}$$

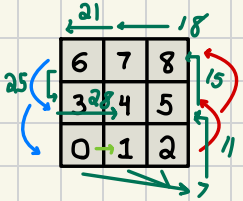
$$X_e^T [y - X_e \theta] = \vec{0}$$

$$X_e^T y - X_e^T X_e \theta = \vec{0}$$

$$X_e^T X_e \theta = X_e^T y$$

$$\theta = \underbrace{[X_e^T X_e]^{-1} X_e^T}_{\text{pinv}(X_e)} y$$

24/Enero/26



Elaboración de la lógica de la tarea

Izq i+1 Arriba +3
Der i-1 Abajo -3

COSTOS
Nada 0
Limpiar 1
→ ← 2
Subir y bajar 3

0 1 2 3 4 5 6 7 8 9
R O 1 2 3 4 5 6 7 8
['A', 'sucio', 'sucio']
['3', 'sucio', 'sucio', 'sucio', 'sucio', 'sucio', 'sucio', 'sucio', 'sucio']
2 0
limpio

Modelo lineal

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

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

$y \in \mathbb{R}$

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

$\mathcal{H}$ = Conjunto de hipótesis

26/Enero/26

$$\underset{(m,n)}{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & & \vdots \\ x_1^{(m)} & x_2^{(m)} & & x_n^{(m)} \end{bmatrix}$$

↘ Matriz de diseño

$$\underset{(m,1)}{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

↘ Vector de salidas

$$\theta = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{bmatrix}$$

$$\theta^* = \arg\min_{\theta \in \mathbb{R}^{n+1}} \frac{1}{2M} \sum_{i=1}^{m} \left(y^{(i)} - [X^{(i)}\ 1] \begin{bmatrix} w \\ b \end{bmatrix}\right)^2 \quad \text{MSE}$$

$$Xe = \begin{bmatrix} X_1^{(1)} & \dots & X_n^{(1)} & 1 \\ & & & \vdots \\ X_1^{(m)} & \dots & X_n^{(m)} & 1 \end{bmatrix}$$

$$\underset{(m,1)}{\hat{y}} = \underset{(m,n+1)}{Xe} \underset{(n+1,1)}{\theta} \quad (m,1)$$

$$\underset{(m,1)}{E} = \underset{(m,1)}{y} - \underset{(m,1)}{\hat{y}}$$

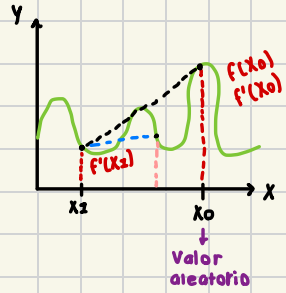
$$E_{in}(\theta) = \frac{1}{M} E^T E$$

Pseudoinversa

$$\theta^* = \underbrace{\underbrace{(Xe^T Xe)^{-1}}_{(n+1,n+1)} \underbrace{Xe^T}_{(n+1,m)}}_{(n+1,m)} \underbrace{y}_{(m,1)} \quad (n+1,1)$$

$f(x)$ $\quad f: \mathbb{R} \to \mathbb{R}$

Descenso de gradiente



$$f'(x_0) = \frac{df(x)}{dx}\bigg|_{x=x_0}$$

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

$$\theta_{k+1} \leftarrow \theta_k - \eta \underbrace{\nabla_\theta f(\theta)|_{\theta=\theta_k}}_{\nabla f(\theta_k)}$$

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

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

$$\theta_{k+1} \leftarrow \theta_k - \eta \nabla f(\theta_k)$$

27/Enero/26

Descenso de gradiente en una regresión lineal

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

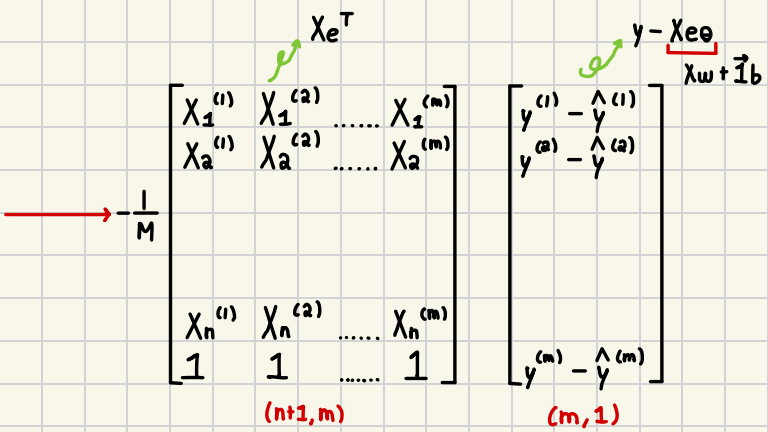
$$\theta^* = w^*, b^* = \arg \min_{\theta \in \mathbb{R}^{n+1}} \frac{1}{2M} \sum_{i=1}^{m} (y^{(i)} - [x^T - 1]\theta)^2$$

para w_j

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

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

$$\nabla E_{in}(\theta_k) = \begin{bmatrix} \frac{1}{M}\sum_{i=1}^{m} -(y^{(i)} - \hat{y}^{(i)}) x_1^{(i)} \\ \frac{1}{M}\sum_{i=1}^{m} -(y^{(i)} - \hat{y}^{(i)}) x_2^{(i)} \\ \vdots \\ \frac{1}{M}\sum_{i=1}^{m} -(y^{(i)} - \hat{y}^{(i)}) x_n^{(i)} \\ \frac{1}{M}\sum_{i=1}^{m} -(y^{(i)} - \hat{y}^{(i)}) \end{bmatrix} \longrightarrow \nabla J(\theta_k) - \frac{2}{M} = \begin{bmatrix} \sum_{i=1}^{m} -(y^{(i)} - \hat{y}^{(i)}) x_1^{(i)} \\ \sum_{i=1}^{m} -(y^{(i)} - \hat{y}^{(i)}) x_2^{(i)} \\ \vdots \\ \sum_{i=1}^{m} -(y^{(i)} - \hat{y}^{(i)}) x_n^{(i)} \\ \sum_{i=1}^{m} -(y^{(i)} - \hat{y}^{(i)}) \end{bmatrix}$$



$\theta \leftarrow \theta - \eta \nabla E_{in}(\theta)$

$\theta \leftarrow \theta + \frac{\eta}{M} X e^T (y - X e \theta)$

vec num num int float

```
def dg_lin(X,Y,wØ,bØ, lr ,max_epochs, etol
```

lr: Learning — max_epochs: Iteraciones — etol: Error de tolerancia

```
w = wØ.copy( )
b = bØ.copy( )
hist = [ ]
M = X.shape[0]
```

→ Descenso de grad.

```
for __ in range(max_epochs):
  y_est = X@ w + b
```

● Variable anonima

(Usando numpy)

```
  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
```

■ Gradiente respecto a w

■ Derivada de b

X, Y ¿Cómo mandar a llamar?

X.shape = [m,n] , Y.shape = [m,]

w = np.zeros(X.shape[-1])

b = 0

w_n, b_n, hist = dg_lin(X,Y,w,b,0.1,50,-1e-4)

Regresión lineal con descenso de gradiente

28/Enero/26

$\hat{y} = Xw + \vec{1}b$

$Err = y - \hat{y}$

$E_{in} = \frac{1}{M} Err^T Err$

$\nabla E_{in} = -\frac{1}{M} X^T Err$

$\frac{dE_{in}}{db} = -\frac{1}{M} \sum_{i=1}^{m} Err^{(i)}$

$w \leftarrow w - lr \nabla_w E_{in}$

$b \leftarrow b - lr \frac{\partial E_{in}}{\partial b}$

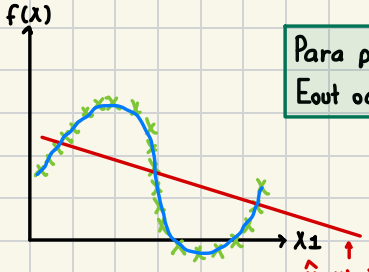
$\Theta^* = \arg\min_{\Theta} E_{in}(\Theta)$

$f \approx h^*$ → $E_{in} \approx 0$

$f \approx h^*$ → $E_{in} \approx E_{out}$

$E_{out} \approx 0$

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



Para pasar de Ein a Eout ocupo muchos datos

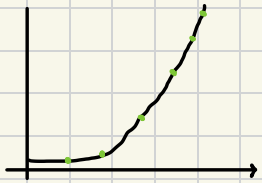
$\hat{y} = w_1 x_1 + w_2 x_i^a + w_3 x_i^3 + b$

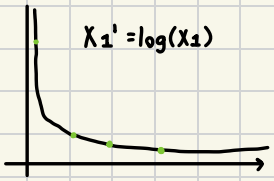
x_1	x_2	x_3	y
$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	$y^{(1)}$
⋮	⋮	⋮	⋮
$x_n^{(m)}$	$x_2^{(m)}$	$x_n^{(3)}$	$y^{(m)}$

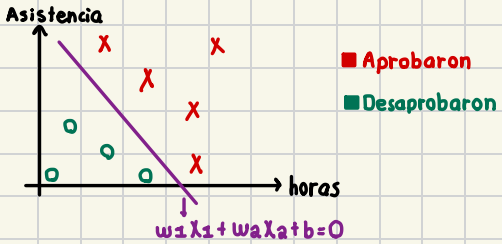
x_1

$x_2 = x_1^a$

$x_3 = x_1^3$







$h_\theta = \text{sign}(w_1 x_1 + w_2 x_2 + b)$

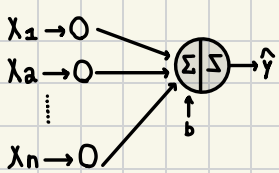
$X \in \mathbb{R}^n$

$y \in \{-1, 1\}$, por ende, y est = $\{-1, 1\}$

$\mathcal{H} = \{h_\theta | h_\theta = \text{sign}(w^T x + b), w \in \mathbb{R}^n, b \in \mathbb{R}, \Theta = (w, b)\}$

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

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



PLA

Libro: Perceptrons

29/Enero/26

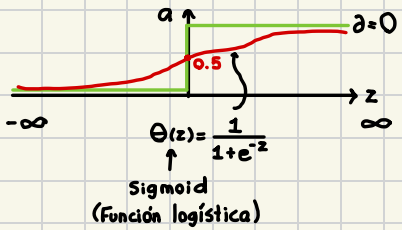
Problema de clasificación a un problema de regresión

$\hat{a} = Pr(Y=1 | X=x, \Theta) = \sigma(w^T x + b)$ → Regresión logística

$$\hat{y} = \begin{cases} 1 & \text{si } \hat{a} > h \\ -1 & \text{en o c} \end{cases} \quad h=0.5 \quad \text{MAP}$$

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

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



Para aprender la regresion logística

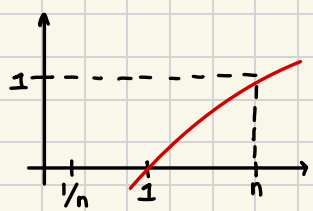
$$\underbrace{\begin{bmatrix} x_1^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & \dots & x_n^{(2)} \\ & & \\ x_1^{(m)} & & x_n^{(m)} \end{bmatrix}}_{X} \quad \underbrace{\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}}_{Y} \quad \underbrace{\begin{bmatrix} a^{(1)} \\ a^{(2)} \\ \vdots \\ a^{(m)} \end{bmatrix}}_{A}$$

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

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

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

La probabilidad de pertenecer al otro equipo es 1



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

$$w \leftarrow w - lr\ \Delta w E_{in}(w,b)$$
$$b \leftarrow b - lr\ \frac{\partial}{\partial b} E_{in}(w,b)$$

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

donde $\hat{a}^{(i)} = \frac{1}{1+e^{-z^{(i)}}}$, y $z^{(i)} = w_1 x_1^{(i)} + \dots + w_n x_n^{(i)} + b$

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

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

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

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

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

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

$$= \left(\hat{a}^{(i)} - \hat{a}^{(i)2} \right) x_j^{(i)}$$

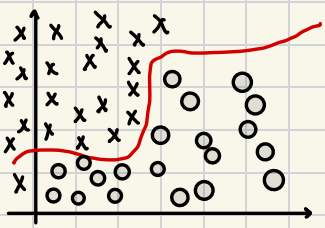
$$= \hat{a}^{(i)} (1-\hat{a}^{(i)}) x_j^{(i)}$$

30/Enero/26

Función:
$\phi: \mathbb{R}^n \to \mathbb{R}^{n'}$
$x \in \mathbb{R}$

$x' = \phi(x) \equiv (x, x^2, x^3, x^4)$
$x' = \phi(x) \equiv (\phi_1(x), \phi_2(x), \ldots, \phi_n(x))$

Tenemos:



$\mathbb{R}^n \to \mathbb{R}^{n'}$

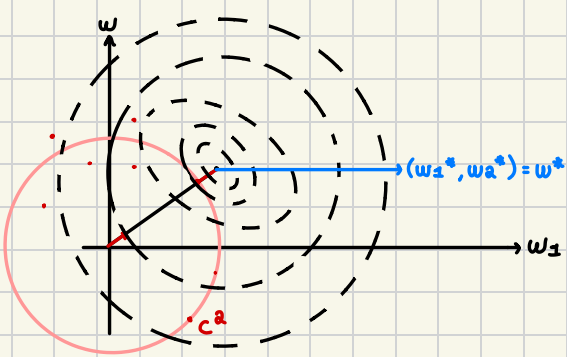
Expansión polinomial

$X = (x_1, x_2)$
$d(x) = (x_1, x_2, x_1^2, x_1x_2, x_2^2)$
$d(x) = (x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3)$
$X = (x_1, x_2, x_3, x_4)$
$\phi(x) = (x_1, x_2, x_3, x_4, x_1^2, x_1x_2, x_1x_3, x_1x_4, x_2^2, x_2x_3, x_2x_4, x_3^2, x_3, x_4, x_4^2)$

$H = \{h\}$ t.q.
$h(x) = w_1x_1 + w_2x_1^2 + w_3x_1^3, w_4x_1^4 + w_5x_1^5 + w_6x_1^6 + w_7x_1^7 + w_8x_1^8 + b$ $Vc = 9$ (Parámetros)

$h(x) = w_1x_1 + w_2x_1^2 + w_3x_1^3 + w_4x_1^4 + w_5x_1^6 + w_7x_1^7 + w_8x_1^8 + b$
bajo $w_j = 0$ si $j \geq 3 \to Vc = 3$
$h(x) = W^T\phi(x) + b$ $w, \phi(x) \in \mathbb{R}^{n'}, b \in \mathbb{R} \to Vc = n' + 1$

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



$\|w\|_2 = \sqrt{w^Tw}$
$\|w\|_1 = \sum_{j=1}^{n'} |w_j|$

3/Feb/26

$$\frac{\partial E_{in}(w,b)}{\partial w_j} = -\frac{1}{M}\sum_{i=1}^{M}\left(\frac{-a^{(i)}}{\hat{a}^{(i)}} + \frac{1-a^{(i)}}{1-\hat{a}^{(i)}}\right)\hat{a}^{(i)}(1-\hat{a}^{(i)})\,x_j^{(i)}$$

$$\frac{\partial E_{in}(w,b)}{\partial w_j} = \frac{\partial}{\partial w_j}\frac{1}{M}\sum_{i=1}^{M}\left[-a^{(i)}\ln(\hat{a}^{(i)}) - (1-a^{(i)})\ln(1-\hat{a}^{(i)})\right]$$

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

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

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

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

$$z^{(i)} = \sum_{i=1}^{M} w_j x_j^{(i)} + b$$

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

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

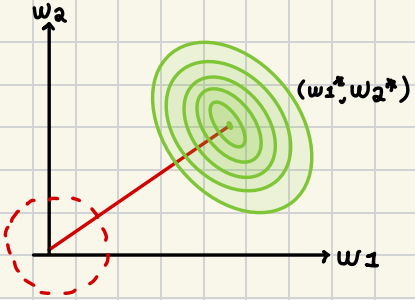
$$\frac{\partial}{\partial w_j}E(w,b) = -\frac{1}{M}\sum_{i=1}^{M}(a^{(i)} - \hat{a}^{(i)})\,x_j^{(i)}$$

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

$$b \leftarrow b - \frac{lr}{M}(a^{(i)} - \hat{a}^{(i)})$$

$$w^*, b^* = \arg\min E_{in}(w,b)$$

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



$w^*, \alpha^* = \arg\min E_{in}(w,b)$

$w_1^2 + w_2^2 = C^2$

$\lambda = 1/c$ — Normalizar datos

$$w^*_{reg}, b^*_{reg} = \arg\min \left[E_{in}(w,b) + \frac{\lambda}{M} \sum_{i=1}^{n} w_j^2 \right]$$

Multiplicador de Lagrange

$$\frac{\partial}{\partial w_j} \left[E_{in}(w,b) + \frac{\lambda}{M} \sum_{i=1}^{M} \right] = -\frac{1}{M} \sum_{i=1}^{n} (a^{(i)} - \hat{a}^{(i)}) x_j^{(i)} + \frac{\partial \lambda}{M} w_j$$

$$w^*_{reg}, b_{reg}^* = \arg\min \left[E_{in}(w,b) + \frac{\lambda}{M} \text{regu}(w) \right]$$

$$\text{regu}(w) = \begin{cases} \sum_{i=1}^{n} w_i^2 = \|w\| & l_2 \\ \sum_{i=1}^{n} |w_i| = & l_1 \end{cases}$$

reg. lineal + $\begin{cases} l_2 - \text{Ridge} \\ \text{LASSO} \\ l_1 + l_2 \end{cases}$

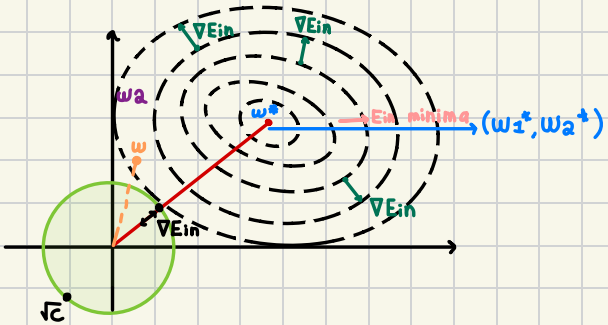
$w^*, b^* = \arg \min_{w,b} E_{in}(w,b)$

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

$w^T w \leq C$

$\|w\|_2^2 \leq C$

$w_1^2 + w_2^2 \leq \sqrt{C}^2$



$w_1^2 + w_2^2 = C$

$w_{reg} = -K \nabla_w E_{in}(w,b)$

$2\lambda w_{reg} + \nabla E_{in}(w_{res}, b) = 0$

$\nabla_w (E_{in}(w,b) + \lambda w^T w)$

$\nabla_w E_{in}(w,b) + 2\lambda w$

Sesgo cognitivo

- La escencia del aprendizaje automático
  - Existe un patrón
  - No es posible establecerlo de forma analitica
  - Tenemos datos o los podemos generar

$f: X \to y$ 5/Feb/26

y quiero ajustar

$h_\theta: \lambda \to y$ a partir de

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

$$\begin{array}{c} \\ x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(N)} \end{array} \begin{bmatrix} f_1 & f_a & \cdots & f_n \\ x_1^{(1)} & x_a^{(1)} & \cdots & x_n^{(1)} \\ x_1^{(2)} & x_a^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & & \vdots \\ x_1^{(m)} & x_a^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \quad \begin{bmatrix} y \\ y^{(1)} \\ y^{(a)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

def genera_arbol(features, X, y, nodo):

Si todos los datos misma clase o no hay datos o no hay features

nodo.terminal = True

regresa nada

var = escoge feature(feature, X, y)

Quitar var de features

por valor en valores(var):

X_h, y_h = separa_datos(X, Y, var, valor)

n_h = crea_hijo(n, y_h)

n_h = genera_arbol(features, X_h, y_h, n_h)

n variables

cada variable 2 valores

a^{2^n}

Canal de comunicación

0 0 0 0 1 0 0 1 0 0 0

Hay un dato perdido (le asignas lo que mas se repite)

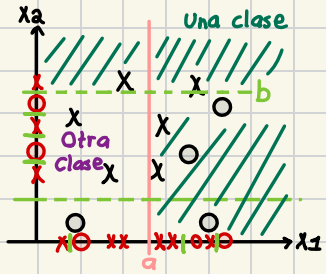
1 1 1 1 1 1 1 0 0 1 1

Calcular entropía

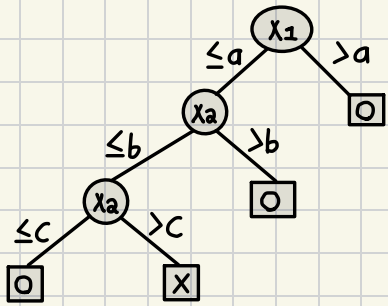
$$\frac{4}{6}\left[-1\log_a 1 - 0\log_a 0\right] + \frac{2}{6}\left[-\frac{1}{2}\log_a\frac{1}{2} - \frac{1}{2}\log_a\frac{1}{2}\right]$$

Si no hay ganancia los datos quedan igual
Aunque la ganancia sea 0, al menos debemos dar un paso más.

6/Feb/26



- Mejor ganancia de información (partición)
- Conviene hacer un árbol

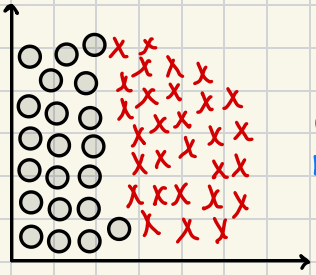


Probables particiones
- Donde haya cambio de clase

Ejemplo:

$$H(y) = \frac{5}{11}\log_2\left(\frac{5}{11}\right) - \frac{6}{11}\log_2\left(\frac{6}{11}\right)$$

$$H(y|x_a, a) = \frac{2}{11}\left(-1\log(1) - 0\log_2(0)\right) + \frac{9}{11}\left[\frac{-6}{9}\log_2\frac{6}{9} - \frac{3}{9}\log_2\frac{3}{9}\right]$$



- No conviene hacer un árbol

Métodos de ensamble

¿Cómo puedo juntar clasificaciones sencillas?

Bagging ¿Qué es?

Método de aprendizaje por conjuntos que se emplea comúnmente para reducir la varianza dentro de un conjunto de datos ruidoso.

Algoritmos de decisión

9/Feb/26

Problema de búsqueda

- Espacio de estados

$S=(s_1,\dots,s_n)$ estado $s \in S$ → Espacio de estado

$\text{card}|S|$ finita

$A=\{a_1,\dots,a_m\}$ acciones

$A: S \to P(A)$

donde

$A(s) \subseteq A$ → Acciones legales en el estado s

$succ: S \times A \to S$

$costo\_local: S \times A \to \mathbb{R}^+$

$costo\_local(s,a)$ es el costo de aplicar la acción a en el estado S

y un estado inicial $S_\emptyset \in S$

y un conjunto de estados finales $S_f \subseteq S$

Problema:

Encontrar un plan

$[(s_0,q_0,c_0),(s_1,q_1,c_1),\dots,(s_T,q_T,c_T)]$

donde (Acción, Estado)

$S_{i+1} = succ(s_i,a_i)$

$succ(s_T,a_T) \in S_f$

$C_i = C_{i-1} + costo\_local(s_i,a_i)$

tal que

C_T sea mínimo

Barquito

$S=\{FCGW \mid F,C,G,W \in \{0,1\}\}$

$F=f$ si ← $\overline{f}$ →

$C=c$ si ← $\overline{c}$ →

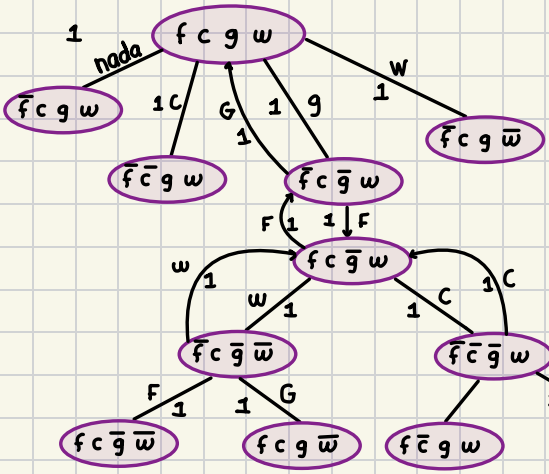
$G=g$ si ← $\overline{g}$ →

$W=w$ si ← $\overline{w}$ →

$2^4=16$ estados

$A=\{C,G,W,\text{nada}\}$

Problema:

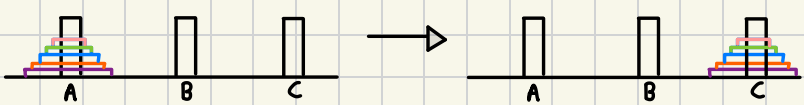


Problema de búsqueda

10/Feb/26

```
class SearchPb(Object):
def __init__(self, s_ini):
self.s = s_ini
def acciones(self, s):
raise NotImplementedError('A desarrollar')
def sucesor(self, s, a):
raise Not...
def terminal (self,s):
raise Not...
```

Ejemplo: Torres de Hanoi



$S = (S_1, \ldots, S_5)$

$S_i \in \{'A', 'B', 'C'\}$

```
class TorresHanoi(SearchPb)
def __init__(self, sø=5*['A'], sf=5*['C']):
self.sø = Sø
```

```
self.sf = sf  → self.acciones = {'AB','AC','BA','BC','CA','CB'}

def acciones(self,s):
  if any (Si not in ['A','B','C'] for si in S):
    raise ValueError(f'Estado {s} imposible')

acciones = {'AB','AC','BA','BC','CA','CB'}
acciones.legales = [ ]
for a in self.acciones:
  if a[0] in S:
      if a[1] not in S or s.index(a[0]) < S index(a[1]):
       acciones.legales.append(a)
```

Uno más bonito

```
class TorresHanoi(SearchPb)
def __init__(self, sø=5*['A'], sf=5*['C']):
self.sø = Sø
self.sf = sf  → self.acciones = {'AB','AC','BA','BC','CA','CB'}

def acciones(self,s):
  if any (Si not in ['A','B','C'] for si in S):
    raise ValueError(f'Estado {s} imposible')
return   [a in self.acciones if a[0] in S and a[1] not in S or S.index(a[0]) <
          s.index(a[1]))]

def succesor(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
```

```
import random...
S0 = 5*['A'] | costo=0
Sf = 5*['C'] |
torres = Torres_Hanoi(S0, Sf)
historial = [ ]
S = S0[i]
for _ in range(5000):
    acciones = torres.acciones(S)
    a = random.choice(acciones)
    s_n, c_l = torres.succesor(s,a)
    S = s_n[]
```

Juego "15" 11/Feb/26

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

$S = (S_0, \ldots, S_{15})$ 16

$S_i = \{0, \ldots, 15\}$

$Card(S) = \frac{16!}{2}$

$A = \{\uparrow, \downarrow, \leftarrow, \rightarrow\}$

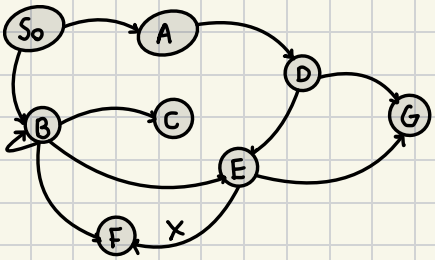
```
class NPuzzle(SearchPb)
  def __init__(self, n, Sf = None):
     self.n = n
     if Sf != None and len(Sf) != n**2:
        raise ValueError('Sf y n no coinciden')
     if Sf == None
        self.s_f = [range (n**2)]
     elif sorted(Sf) != [range (n**2)]
        raise ValueError('Sf no es una permutación')
     else
        self.s_f = Sf[i]
```

```
def acciones(self,S):          ← Diccionario
  if not exist(self.a_legales):
    self.a_legales={ }
    for i in range(self.n**2)
      temp=[ ]
      if i//self.n > 0
         temp.append('↑')
      if i//self.n < self.n-1:
         temp.append('↓')
      if i%.self.n >0:
         temp.append('→')
      if i%.self.n<self.n-1:
         temp.append('←')
        self a_legales[i]=temp[i]
 return  self a_legales[s.index[0]]

def succesor(self,s,a):
  costo_local=1
 if a not in self acciones(?):
   raise ValueError('—')
 S_n=S[i]
 i_vacio=S_n.index(0)
 i_otro=(i_vacio+1  if a=='→' else
         i_vacio-1  if a=='←' else
         i_vacio+self.n if a=='↓' else
         i_vacio-self.n)
 S_n[i_vacio],S_n[i_otro]=S_n[i_otro],S_n[i_vacio]
return  S_n, costo_local

def terminal(self,s):
   return  s==self.s_f
plan=busqueda(SearchPb,Sø)
plan=[(Sø,aø,Cø),(S1,a1,C1),(S2,a2,C2),...
```

Costo acumulado

$(S_{T-1}, a_{T-1}, C_{T-1}), (S_T, None, None)]$

donde $S_{Ti}, \_ = succesor(S_{T-1}, a_{T-1})$ es un estado terminal

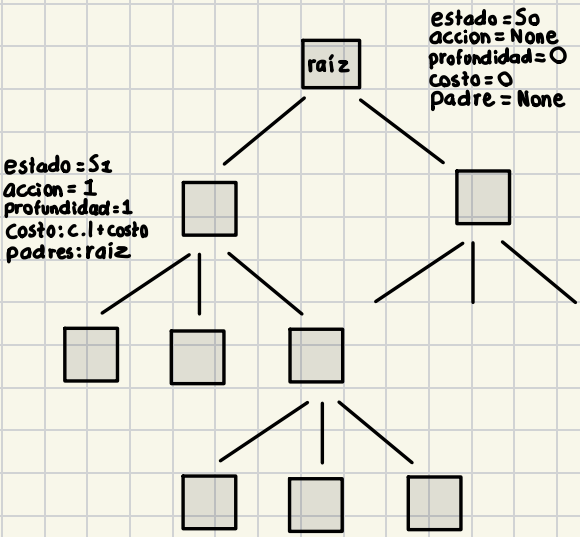


So

A(s) = acciones(S)

S', c_{local} = succesor(s,a)

terminal(S)




```
class NodoSearch(object):
    def __init__(self, S, a=None, padre=None, costo_l=None)
        self.s=s
        self.a=a
        self.padre=padre
        self.d= 0  if  padre == None  else  self.padre.d + 1
        self.costo = 0  if  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. succesor(s,a)
```

yield NodoSearch(s_n, a, self, costo_l)

$[(S_0, a_0, C_0), (S_1, a_1, C_1), \ldots, (S_{T-1}, a_{T-1}, C_{T-1}), (S_T, None, None)]$

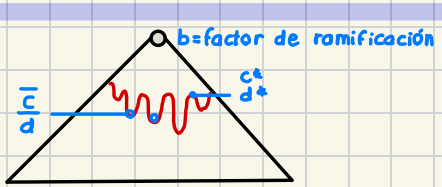
```
def plan(self):
  return [(self.s, None, None)] if self.padre = None else
   self.padre.plan()[:-1]+[(self.padre.s, self.a, self.costo),(self.s, None, None)]
```

$[(S_0, a_0, C_0), (S_1, a_1, C_1), \ldots, (S_{T-1}, None, None)]$

```
def busqueda_generica(S, pb):
  frontera=[NodoSearch(S)]
```

Posibles planes a realizar

```
  while frontera:
    plan = saca_nodo(frontera)
    if pb.terminal(plan.s):
        return plan
    for plan_hijo in plan.expande(pb):
      agrega_a_frontera(frontera, plan_hijo)
return None
```

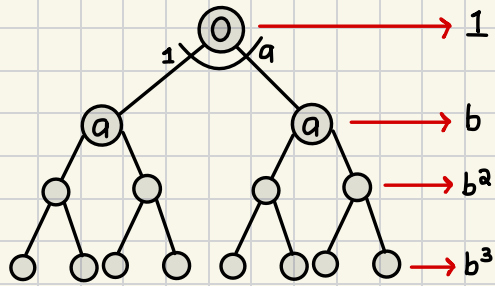


13/Feb/26

Algoritmo
- Admisibilidad
- Optimalidad
- Complejidad malerial
- Complejidad temporal

```
for i in range(n)
 for j in range(n)
Proposiciones atómicas
q1...qn  p(q1...qn)
```

O(n)
$O(n^2)$
$O(2^n)$ P.C.



Primero a lo ancho (BFS)

[ø]
[] → [1,...,a]
[2,...,a] → [2,...,a,11,.... 1a]
[3,...a,11,...,1a] → [3,....a,11,...,1a,21,...2a]

Si el plan admisible de menor profundidad tiene profundidad $\bar{d}$ entonces vamos a revisar $1+b+b^2+....+b^{\bar{d}}$ que es $O(b^{\bar{d}})$

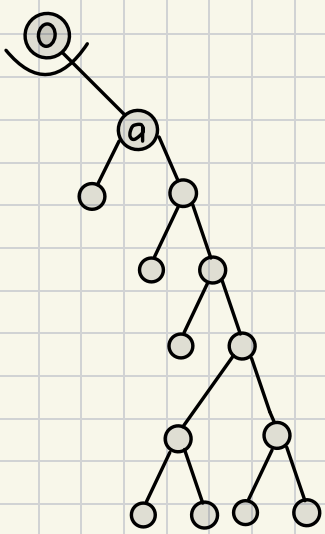
¿Admisible? Sí porque si existe al menos un plan habra un d^* de profundidad más pequeño
¿Óptimo? Solo si $C\ell=1$

¿El mejor plan se hace en menor número de pasos? No necesariamente

Complejidad temporal $O(b^{\bar{d}})$

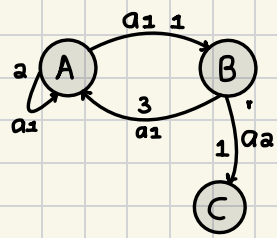
Complejidad malerial $O(b^{I})$

Probar pila

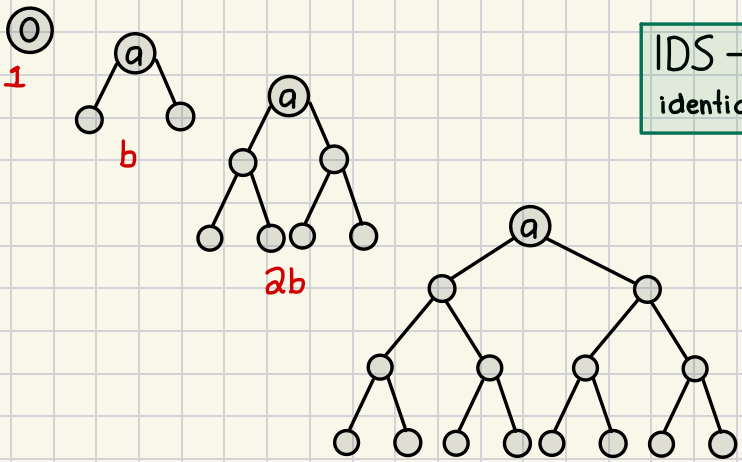


[ø]
[] → [1,...,a]
[1,...,a-1] → [1,...,a-1, a_1,...,aa]

Cualquier grafo con ciclos tiene profundidad infinita



¿Admisible? Si dmax es ∞ NO
¿Óptimo?
¿Temporal? $O(b^{dmax})$
¿Material? $O(b*dmax)$



IDS → Búsqueda por identidad iterativa

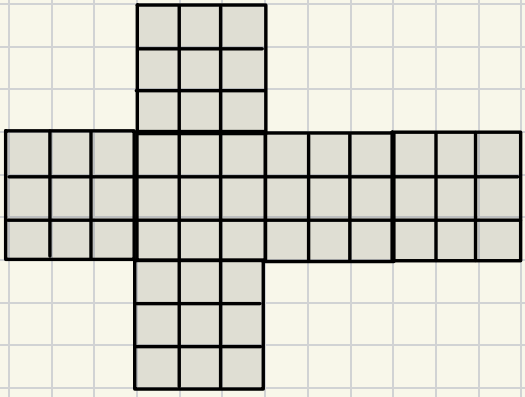
Problema de búsqueda

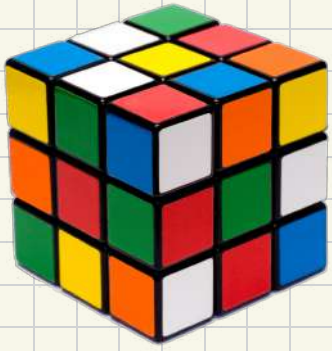
16/Feb/26

- Estado inicial
- Acciones posibles
- Costo de la acción
- Sucesor de un nuevo estado
- Estado terminal

} Problema

Cubo Rubik





Transportation problem

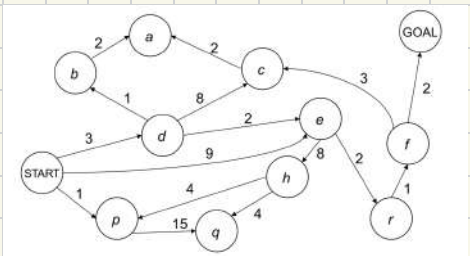
Example: transportation

Street with blocks numbered 1 to n.
Walking from s to $s+1$ takes 1 minute.
Taking a magic tram from s to $2s$ takes 2 minutes.
How to travel from 1 to n in the least time?

Programar este y el cubo de Rubik

- Las acciones son diferentes
- Los costos cambian

Dijkstra no puede usarse en grafos grandes como el pacman (120 mil millones de estados) porque no habría memoria



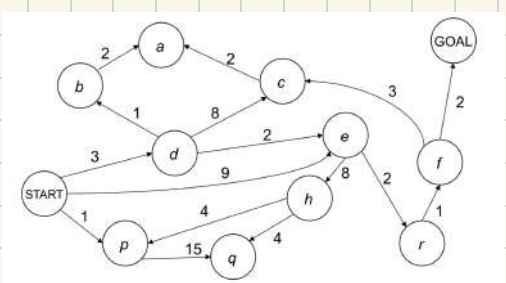
start $\xrightarrow{9}$ e $\xrightarrow{11}$ r $\xrightarrow{12}$ f $\xrightarrow{14}$ Goal Mejor profundidad 4 pasos

start $\xrightarrow{3}$ d $\xrightarrow{5}$ e $\xrightarrow{7}$ r $\xrightarrow{8}$ f $\xrightarrow{10}$ Goal Menor costo 5 pasos

(n costo ... n)
Lista → sort (lento)
Cola tipo heap

[(1,p),(3,d),(9,e)]
[(3,d),(9,e),(15,q)]
[(4,b),(5,c),(9,e),(11,c),(15,q)]
[(5,e),(6,a),(9,e),(11,c),(15,q)

Retomando... (n. costo, n) heap 17/Feb/26



[(0, START]
[(1,p),(3,d),(9,e)]
[(3,d),(9,e),(15,q)]

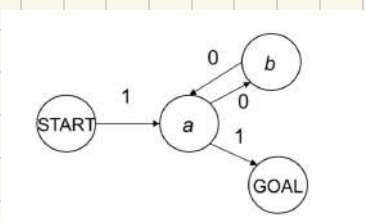
[(4,b),(5,e),(9,e),(11,c),(15,q)]
[(5,e),(6,a),(9,e),(11,c),(15,q)]
[(6,a),(7,r),(9,e),(11,c),(13,h),(15,q)]
[(7,r),(9,e),(11,c),(13,h),(15,q)]
[(8,f),(9,e),(11,c),(13,h),(15,q)]
[(9,e),(10,G),(11,c),(13,h),(15,q)]
[(10,G),(11,c),(13,h),(15,q)]

g^* un plan óptimo terminal(g^*.estado) = True
g^*.costo $\leq \bar{g}$.costo
para todo $\bar{g}$ tal que
terminal($\bar{g}$.estado = True

Cuando sacamos g^* de Frontera hemos revisado TODOS los planes p tal que p.costo < g^*.costo y algunos que p.costo = g^*.costo pero NINGUNO que p.costo > g^*.costo

¿Óptimo? ✓
¿Admisible? ✓
} Si es óptimo es admisible
¿Complejidad? $O(b^{[\frac{g^*.costo}{\epsilon}]})$

Costos > 0 No negativos



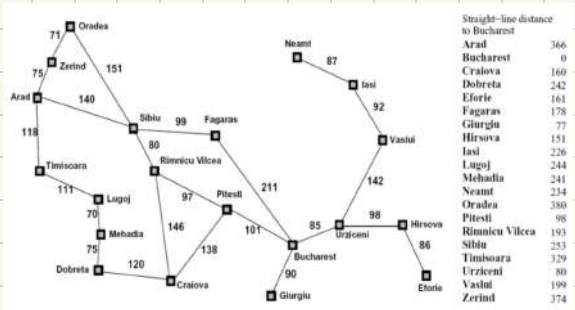
Si no, se va a ciclar

Pacman modelo relajado → Laberinto No
→ Cualquier ángulo
→ Costo menor que la solución real

Heuristica
h = heurística (plan)
h es un costo estimado de un problema relajado
h ≤ Costo de un plan completo desde plan.estado

$h \leq g^*$ iniciando en plan.estado

si esto ocurre entonces $\forall s \in S$ se dice que la heuristica es ADMISIBLE



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

[Arad]
[Sibiu, Timisoara, Zerind]
[Faragas, Rinmineu, Timisoara, Zerind, Oradea]
[Bucarest, Rinmineu, Timisoara, Zerind, Oradea]

Arad → Sibiu → Faragas → Bucarest Costo: 450
No encontramos la óptima

UCS → (n.costo, n)
Greedy → (heurística(n), n) Mucho menos búsqueda pero no garantiza ser el más óptimo
A^* → (n.costo + heuristica(n), n)

$\bar{g}$ es un plan completo que pasa por n donde n es un plan intermedio
$\bar{g}$.costo $\geq$ n.costo + heurística(n)
heurística($\bar{g}$) = 0 → Admisible
Si g^* es un plan óptimo
heurística(g^*) = 0
$\bar{g}$.costo $\leq g^*$.costo
$\bar{g}$.costo $\leq g^*$.costo + heurística(g^*)

Heurísticas

18/Feb/26

Ejemplo: 8 Puzzle
$S = (S_0, \ldots, S_8)$
$S_i \in \{0, \ldots, 8\}$
$card|S| = \frac{9!}{2}$

acciones = {←, →, ↑, ↓}

Costo local = 1

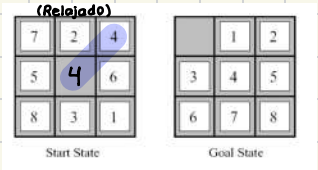
h_1 = # piezas mal colocadas

El espacio en blanco no cuenta como ficha

h_a = dist. manhattan de cada pieza a su posición

dom.

(Relajado)

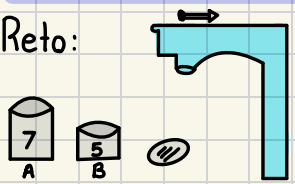


Se resuelve en el número de fichas mal ordenados

deque = Lista que accede por posición, hace popleft y mete al principio

heap = Saca siempre la mayor

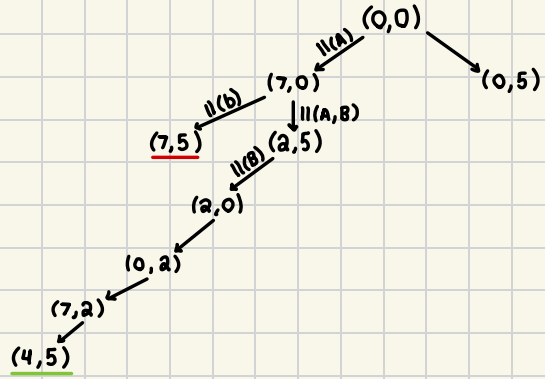
Reto:



Como llenar 4 litros de agua en A o B y solo puedo trasladar o vaciar

19/Feb/26

A lo profundo



El que más gasta es con 20 y 19 lts

Heurística trivial

$h(n) = 0 \quad \forall n$

La mejor: h(n) + n.costo es el costo exacto de la solución óptima que n sea un plan intermedio

Si $h_1(n) \leq h_a(n)$

$\forall n$ entonces

h_a domina a h_1

Si h_a y h_1 son admisibles h_a me dará un mejor valor

La dominante siempre será la mayor

Cuestionario en Teams

20/Feb/26

h heurística

h es admisible si

$h(n) \leq g^*(n)$

donde $g^*(n)$ es el costo del plan óptimo iniciando con n.estado

$h(n)=0$ si n.estado es terminal → El plan óptimo es no hacer nada

Yo en A^* ordeno los nodos por n.costo + h(n) si n.estado es terminal n.costo+h(n)=n.costo

Carretera: GPS tienes info de tu ubicación, se puede medir en linea recta → Desde donde estoy hasta donde quiero llegar. Tenemos una heurística e información extra para crear la misma.

Al tener información extra el algoritmo óptimo es el A^*

Un puzzle un poco diferente

1. Establezca una manera de representar el estado del problema.

$S=(S_0,....S_{16})$

Cardinalidad:

$card(S)=16!$

2. Establezca cuales serían las acciones legales en un estado dado.

16 acciones

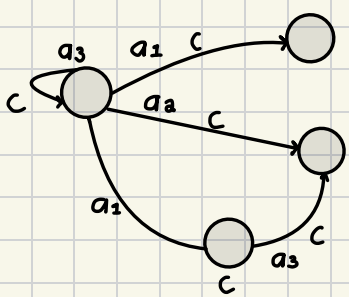
3. Establezca el costo local dependiendo del estado y la acción.

1 para que sea más barato

Búsqueda

23/Feb/26

Discreto, dinámico, determinista, conocido. → Grafo con diferentes estados:



sigue

Discreto
Dinámico
Estocástico
Conocido

sigue

Discreto
Dinámico
Estocástico
Desconocido

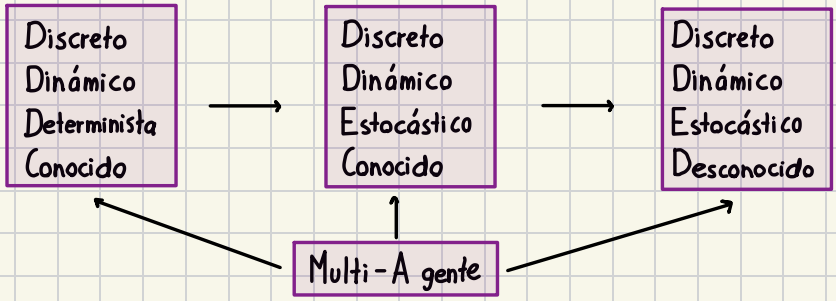
Plan admisible → Óptimo: El de menor costo

Búsqueda por profundidad iterativa

Es un algoritmo de búsqueda no informada que combina la eficiencia de memoria de la búsqueda en profundidad (DFS) con la completitud y optimalidad de la búsqueda en amplitud

Planeación: Ir de un estado inicial a un estado terminal, con el mejor costo

MULTI-AGENTE



Todos los agentes trabajan

Introducción a los juegos con adversarios

Tipos de juegos

- Game = task environment with > 1 agent
- Axes:
  - Deterministic or stochastic?
  - Perfect information (fully observable)?
  - Two, three, or more players?
  - Teams or individuals?
  - Turn-taking or simultaneous?
  - Zero sum?

- Want algorithms for calculating a **strategy** (policy) which recommends a move from every possible state

Juegos deterministas

- Many possible formalizations, one is:
  - States: S (start at s_0)
  - Players: P={1...N} (usually take turns)
  - Actions: A (may depend on player/state)
  - Transition function: S x A → S
  - Terminal test: S → {true, false}
  - Terminal utilities: S x P → R
- Solution for a player is a policy: S → A

24/Feb/26

```
class Juego2TZD(object):
    def acciones_legales(self,estado,jugador):
        return conjunto acciones

    def sucesor(self,estado,accion,jugador):
        return estado_nuevo

    def terminal(self,estado):
        return bool

    def ganancia(self,estado):
        return numero
```

Juego del gato

$S=(S_0,\dots,S_8)$

$S_i \in \{1, 0, -1\}$

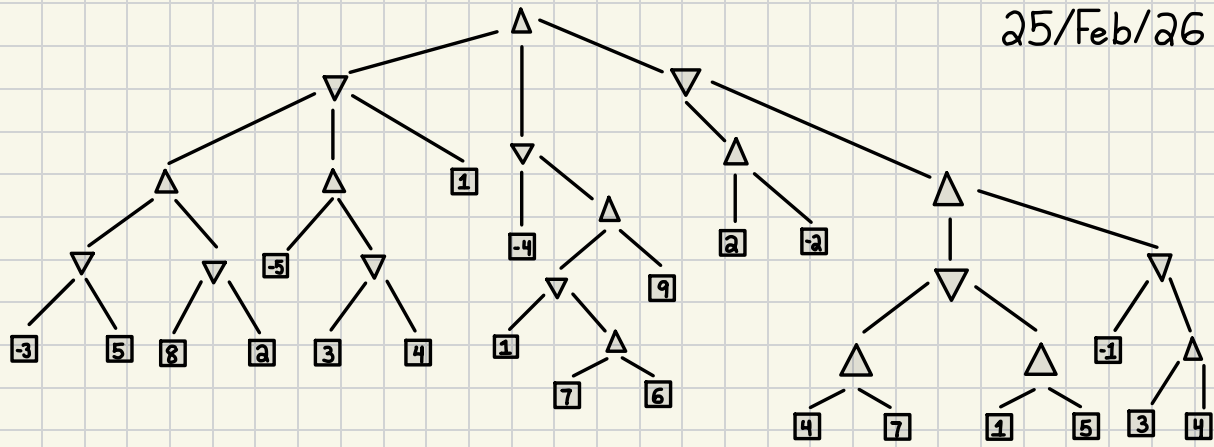
$a \in \{0,\dots,8\}$

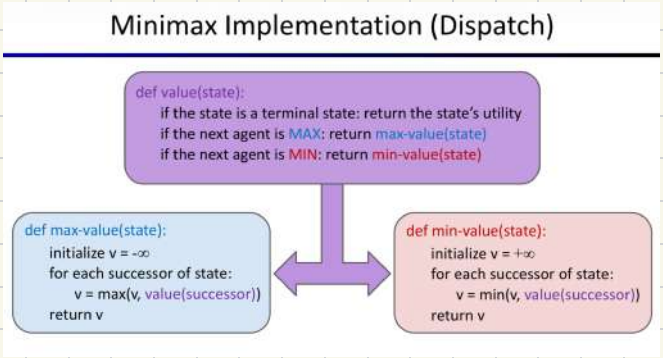
```
class Gato(Juego2TZD):
    def acciones_legales(self,estado,jugador):
        return[i for (i,s) in enumerate (estado) if s==0]
    def sucesor(self,estado,accion,jugador):
        s_n=estado[:]
        s_n[accion]=jugador
        return s_n
```

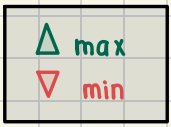
```
    def terminal(self,estado):
        is 0 not in estado:
            return True
        if ((s[0]==S[4]==S[8]) or
            s[2]==S[4]==S[6])) and S[4]!=0:
            return True
        for i in range(3):
            if((s[i]==S[i+3]==S[i+6]) or
              (S[3i]==S[3i+1]==S[3i+2])) and s[1] !=0
            return S[3i]
        return True

    def ganancia(self,estado):
        is 0 not in estado:
            return Ø
        if ((s[0]==S[4]==S[8]) or
            s[2]==S[4]==S[6])) and S[4]!=0:
            return S[4]
        for i in range(3):
            if((s[i]==S[i+3]==S[i+6]) or
              (S[3i]==S[3i+1]==S[3i+2])) and s[3i] !=0
            return S[3i]
        return Ø

gato = Gato( )
humano = 1 #1 ó -1
S=[0]*9
j=1
actualiza_tablero(S)
for _ in range(9):
    if S==humano:
        acciones=gato.acciones_legales(s,j)
        a=escoge_accion(acciones)
    else:
        a=agente(gato,S,j)
S=gato.sucesor(s,a,j)
```

```
actualiza_tablero(S)
j = -j
if gato.terminal(S):
    break
actualiza_score(gato.ganancia(S))
```

25/Feb/26







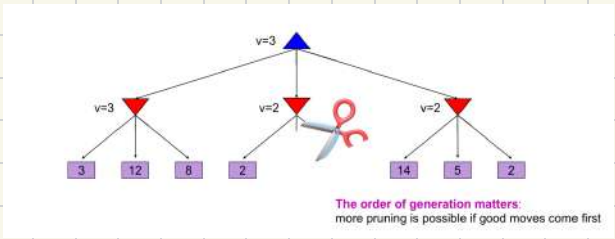

```
a = max(a for a in juego.acciones_legales(s,j),
        key = value(juego.sucesor(s,a,j), -1) * j)
```

$\max(\min(3,10), \min(1,x,x), \min(-3,x,x,x))$
$\leq 1 \qquad \leq -3$

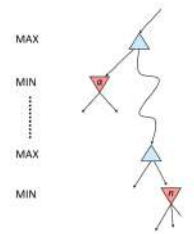
Poda α-β
Costo: $O(b^{dmax})$

Propiedades minimax

26/Feb/26



- General case (pruning children of MIN node)
  - We're computing the MIN-VALUE at some node *n*
  - We're looping over *n*'s children
  - *n*'s estimate of the childrens' min is dropping
  - Who cares about *n*'s value? MAX
  - Let **α** be the best value that MAX can get so far at any choice point along the current path from the root
  - If *n* becomes worse than **α**, MAX will avoid it, so we can prune *n*'s other children (it's already bad enough that it won't be played)
- Pruning children of MAX node is symmetric
  - Let **β** be the best value that MIN can get so far at any choice point along the current path from the root



Implementación alpha beta

α: MAX's best option on path to root
β: MIN's best option on path to root

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v ≥ β return v
        α = max(α, v)
    return v
```

```
def min-value(state , α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v ≤ α return v
        β = min(β, v)
    return v
```

$[\alpha, \beta]$
↑ v

$\max(x,y) = \min(-x,-y)$

$j * \max(J * X, j * y)$

negamax

$\text{utilidad}(S): \sum_{j=1}^{n} w_i \, \phi_j(S)$

ls=['z','bc','aaa']
↑ max

2/Mar/26

Clasifica por orden lexicográfico

max(ls, key=lambda x len(x)) → Cadena con más caracteres

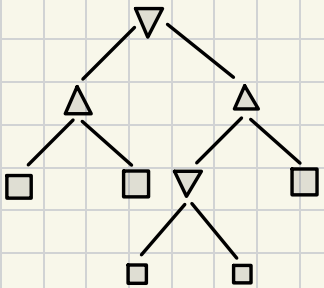
3/Mar/26

```
if juego.terminal( ):
 return j*juego.ganancia(s)
jugadas=ordena(list(___))
v=-∞
for a in jugadas
v=max(v, min.val(s,-j, α, β)

if juego.terminal( ):
 return -j*juego.ganancia(j)
jugadas=____
v=∞
for a in jugadas
v=min (v,max.val(s',-j, α, β)

 if juego.terminal( ):
  return j*juego.ganancia(s)*l
 jugadas=ordena(list(___))
 v=-∞
 for a in jugadas
 v=max(l*v, l*nega_val(s',-j,-β,-α,-l)

 if v>=β:
   return l*v
 α=max(α,v)

return l*α
```

TT → dict{
S:(d,v)
↓ Est ↓ Profundidad → Valor

```
if d==0
  j*utilidad(s)*l
if s in TT and TT[s][0]<=d:
  j*TT[s][1]*l
```

Analisis de código



Búsqueda Quiecente

4/Mar/26

$\min(x,y) = -\min(-x,-y)$

Va = negamax( )

Va = −negamax( ) valor 2

$V = -\infty$

para a en acciones_legales:

Dudas y análisis código conecta 4

Va = −negamax( )

$V = \max(V_1, Va)$

$V = \max(V_1, -\text{negamax}())$

$\alpha \leq \max(v,y) \leq \beta$

$-\beta \leq \max(-v,-y) \leq -\alpha$

$\min(x,y) = -\max(-x,-y)$

Cadenas de Markov

5/Mar/26

Deterministas → estocásticos

y → Variable estocástica

↘ salida

valor(Y) = $\{y_1, \ldots, y_n\}$ n valores posibles (estocástica)

y ~ distribución

$val(y) = \mathbb{R} \rightarrow y \sim N(\mu, \sigma^2) \leftrightarrow Pr(y=y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

$val(y) = \{0,1\} \rightarrow y \sim \text{Bernoulli}(\rho) \leftrightarrow Pr(y=y) = \rho^y (1-\rho)^{1-y}$

$val(y) = \{1, \ldots, K\} \rightarrow y \sim \text{Multinomial}(\emptyset) \leftrightarrow Pr(y=y) = \emptyset_1^{1\{y=1\}} \emptyset_2^{1\{y=2\}} \ldots \emptyset_K^{1\{y=K\}}$

$\emptyset_j > 0 \quad \forall j = 1, \ldots, K$

$\sum_{j=1}^{K-1} \emptyset_j \leq 1$

Variable aleatoria

6/Mar/26

- Función burrito (envuelve otra)
  - Azúcar sintáctica (Facilitas un problema)
    - Se llamará y

Cada posible conjunto de mi conjunto universo es un evento

$Pr: P(U) \to \mathbb{R}$

$Pr(\bar{A}) \geq 0$

$Pr(U) = 1$

$Pr(\emptyset) = 0$

$Pr(A \cup B) = P(A) + P(B) + P(A \cap B)$

$Pr(y_1) = Pr(Y = y_1) = Pr(\{w \in U \mid y(w) = y_1\})$

$A_1, \dots A_n$ es partición de U ssi $\bigcup_{i=1} A_i = U$ y $A_i \cap A_j = \emptyset \quad \forall ij, \ i \neq j$

$val(Y) = \{y_1, \dots y_N\}$

$Pr(y = y_1) = Pr(y_1)$

$y_0 = y_0, y_1 = y_1, y_2 = y_2, \dots, y_T = y_T$ → $y_{0:T}$, $y_{0:T}$

Serie de tiempo

$P(y_{0:T}) = P(y_0, y_1, y_2, \dots y_T)$ verosimilitud

$P(y_T \mid y_{0:T-1})$ → Estimación

$P(y_{T:T+h} \mid y_{0:T-1})$ → Pronóstico (forecasting)

$P(y_T \mid y_{0:t-1}, y_{t+1:T})$ → Suavizado (smoothing)

Si $Pr(y_T \mid y_{0T-1}) \equiv Pr(y_T \mid y_{T-1})$

entonces y es un proceso de Markov de primer orden

$Pr(y_T \mid y_{T-1}, y_{T-2})$

$z_T = \begin{bmatrix} y_T \\ y_{T-1} \end{bmatrix} \to Pr(z_T \mid z_{T-1})$

$\begin{bmatrix} y_T \\ y_{T-1} \end{bmatrix} \begin{bmatrix} y_{T-1} \\ y_{T-2} \end{bmatrix}$

$Pr(y_{0:T}) = Pr(y_T \mid y_{0:T-1}) Pr(y_{T-1} : y_{0:T-2}) \dots P(y_1 \mid y_0)$

$Pr(y_{0:T}) = Pr(y_0) \prod_{t=1}^{T} Pr(y_t | y_{t-1})$

$P(y_3, y_2, y_1 y_0)$

$P(x|y) = \frac{P(x,y)}{P(y)}$

$P(x|y)\ P(y) = P(x,y)$

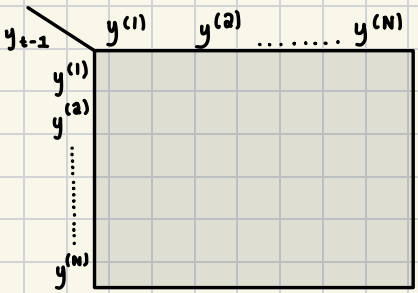
$val(Y) = \{G, P\}$

$Pr(y_T = G | y_{t-1} = G) = 0.6$

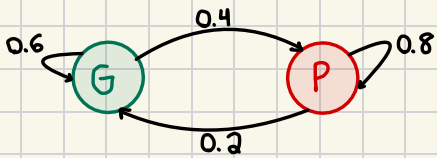
$Pr(y_T = G | y_{t-1} = P) = 0.2$

$Pr(y_T = P | y_{t-1} = G) = 0.4$

$Pr(y_T = P | y_{t-1} = P) = 0.8$



y_{T-1} \ y_T	P	G
P	0.8	0.4
G	0.2	0.6



$y_0 = \begin{bmatrix} P=1 \\ G=0 \end{bmatrix}$

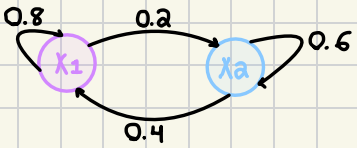
$y_1 = A y_0 \quad y_2 = A y_1 = A^2 y_0$

$\begin{bmatrix} P(P|G) & P(G|G) \\ P(G|G) & P(G|P) \end{bmatrix} \begin{bmatrix} P(G) \\ P(P) \end{bmatrix}$

$y_T = A^T y_0$

Continuación cadenas de Markov

9/Mar/26



$$\begin{array}{c} \\ X_1 \\ X_a \end{array}\begin{array}{c} \begin{array}{cc} X_1 & X_a \end{array} \\ \begin{bmatrix} 0.8 & 0.4 \\ 0.2 & 0.6 \end{bmatrix} \end{array} = \begin{bmatrix} P(X_t=X_1 \mid X_{t-1}=X_1) & P(X_t=X_1 \mid X_{t-1}=X_a) \\ P(X_t=X_a \mid X_{t-1}=X_1) & P(X_t=X_a \mid X_{t-1}=X_a) \end{bmatrix}$$

$$P(X_t=X_1) = P(X_{t-1}=X_1)P(X_t=X_1 \mid X_{t-1}=X_1) + P(X_{t-1}=X_a)P(X_t=X_1 \mid X_{t-1}=X_a)$$
$$P(X_t=X_a) = P(X_{t-1}=X_1)P(X_t=X_a \mid X_{t-1}=X_1) + P(X_{t-1}=X_a)P(X_t=X_a \mid X_{t-1}=X_a)$$

$$\begin{bmatrix} P(X_t=X_1) \\ P(X_t=X_a) \end{bmatrix} = \begin{bmatrix} P(X_t=X_1 \mid X_{t-1}=X_1) & P(X_t=X_1 \mid X_{t-1}=X_a) \\ P(X_t=X_a \mid X_{t-1}=X_1) & P(X_t=X_a \mid P(X_{t-1}=X_a) \end{bmatrix} \begin{bmatrix} P(X_{t-1}=X_1) \\ P(X_{t-1}=X_a) \end{bmatrix}$$

$$P(X_t) = A\,P(X_{t-1}) = A^t P(X_{t_0})$$
$$P(X_{t_1}) = A\,P(X_{t_0})$$
$$P(X_{t_2}) = A\,P(X_{t_1}) = AA P(X_{t_0})$$

$$P(X=X_1) = 0.8P(X=X_1) + 0.4P(X=X_a)$$
$$P(X=X_a) = 0.2P(X=X_1) + 0.6P(X=X_a)$$

$$P(X_T=X_1) = 2\,P(X_T=X_a) \rightarrow P(X_a) = \tfrac{1}{2}P(X_1)$$
$$P(X_1)+P(X_a)=1$$
$$P(X_1)+\tfrac{1}{2}P(X_1)=1 \rightarrow P(X_1)=\tfrac{2}{3}$$
$$P(X_a)=\tfrac{1}{3}$$

$$P(X_1) = 0.8\,\tfrac{2}{3} + 0.4\,\tfrac{1}{3} = \frac{8+2+4}{30} = \frac{20}{30} = \tfrac{2}{3}$$
$$P(X_a) = 0.2\,\tfrac{2}{3} + 0.6\,\tfrac{1}{3} = \frac{2+2+6}{30} = \frac{10}{30} = \tfrac{1}{3}$$

Modelo:

contiene...

$S \rightarrow \{S_1, \ldots, S_n\}$ estado $|S|$

$A = \{a_1, \ldots, a_m\}$

- acciones: $S \rightarrow P(A)$
- acciones(s) $\subseteq A$

Discreto Conocido Determinista Dinámico	→	Discreto Desconocido Dinámico Estocástico

- [sucesor: $S \times A \to S$; sucesor(s,a) $\in S$] — sucesor: $S \times A \times S \to \mathbb{R}$ — $\underbrace{\text{sucesor}(s,a,s')}_{Pr[S_{t+1}=s' \mid S_t=s, A_t=a]}$

- [costo_local: $S \times A \to \mathbb{R}^+$; costo_local(s,a) $\in \mathbb{R}^+$] — recompensa: $S \times A \times S \to \mathbb{R}$ recompensa(s,a,s')

- [terminal: $S \to \{F, V\}$; terminal(s)]

Juego de dado 10/Mar/26

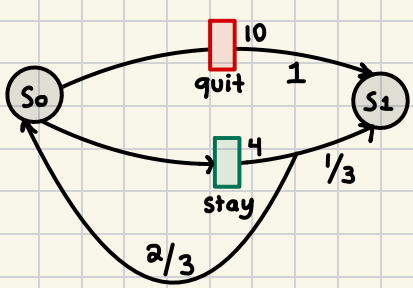
For each round r = 1,2,...
- You choose stay or quit
- If quit, you get $10 and we end the game
- If stay you get $4 and then roll a 6...

- If the dice = 1 or 2 we end game
- Otherwise continue to the next round

$\pi: S \to A$ Política determinista
$\pi(S_0) = quit$
$\pi_1(S_0) = stay$

$\pi: S \times A \to \mathbb{R}$ Política estocástica
$\pi(S_0, quit) = p$
$\pi_1(S_0, stay) = 1-p$

Representación:



Esperanzas

$E^{\pi_1}(R_t) = 10$

$E^{\pi_1}(R_t) = 1/3(4) + 2/3 \cdot 1/3(8) + 2/3 \cdot 2/3 \cdot 1/3(12) + \ldots = 12$

$E(x) = \sum x P_r(x)$

$E^{\pi}[R_t]$

$R_t = r_t + r_{t+1} + r_{t+2} \ldots$

$V^{\pi}(s) = E^{\pi}[R_T | S_T = s]$

$\pi_1 < \pi_2$

Sii

$V^{\pi 1}(s) \leq V^{\pi 2}(s) \quad \forall s \in S$

Factor de descuento

Caso felicidad a corto y largo plazo

Función de valor de estado

$R_T = \gamma^0 r_t + \gamma^1 r_t + \gamma^2 r_t \ldots$

π^* es una política optima ssi

$V^{\pi}(s) \geq V^{\pi *}(s) \quad \forall \pi$

Función de valor estado - acción

$Q^{\pi}(s,a) = E^{\pi}[R_t | S_t = s, a_t = a]$

MDP

11/Mar/26

- Markov
- Decision
- Process

> Mathematical model for sequential decision making when outcomes are uncertain

Analizaremos...

¿Que debemos tomar en cuenta?

$S = \{S_1, \ldots, S_n\}$

$A = \{a_1, \ldots, a_n\}$

$A_{Legal}: S \to P(A)$

$S_T \subseteq S$

$T = S \times A \times S \to \mathbb{R}$

$r = S \times A \times S \to \mathbb{R}$

$S_T \leq S$

$0 \leq \gamma \leq 1$

¿Qué quiero?

para cada $s \in S$

$E[R_T | S_T = s]$ son máxima

$R_T = r_T + \gamma r_{T+1} + \gamma^2 r_{T+2} + \gamma^3 r_{T+3} + \dots$

¿Cómo lo consigo?

$\pi: S \to A$

$\pi: S \to A \to \mathbb{R}$

¿y si tengo 2 políticas π_1 y π, cuál es mejor?

Calculamos la función de valor de estado de π

$$V^{\pi}(s) = E^{\pi}[R_T | S_T = s]$$

$$V^{\pi}(s) = \sum_{s' \in S} T(s, \pi(s), s') \left[r(s, \pi(s), s') + \gamma E^{\pi}[R_T | S_T = s] \right]$$

$$V^{\pi}(s) = \sum_{a \in A_L(s)} \cdot \pi(s,a) \sum_{s' \in S} T(s,a,s') \left[\gamma(s,a,s') + \gamma V^{\pi}(s') \right]$$

Si...

$V^{\pi_1}(s) \leq V^{\pi_2}(s) \quad \forall s \in S$

entonces $\pi_1 \leq \pi_2$

es mejor la política π_2

π_1 y π_2

y hago

$$\pi_3 \text{ t.q. } \pi_3(s) = \begin{cases} \pi_1(s) & \text{si } V^{\pi_1}(s) > V^{\pi_2}(s) \\ \pi_2(1) & \text{si } V^{\pi_2}(1) \geq V^{\pi_1}(1) \end{cases}$$

Función de valor de Estado_Acción

$Q^{\pi}(s,a) = E^{\pi}[R_t | S_t = s, a_t = a]$

$$Q^{\pi}(s,a) = \sum_{s' \in S} T(s,a,s') \left[\gamma(s,a,s') + \gamma \underbrace{V^{\pi}(s')}_{\sum_{a' \in A(s')} \pi(s',a') Q^{\pi}(s',a')} \right]$$

$$\begin{bmatrix} V^{\pi}(S_1) \\ V^{\pi}(S_2) \\ \vdots \\ V^{\pi}(S_n) \end{bmatrix} = \begin{matrix} \sum_{s' \in S} T(\ldots \text{ Ya vimos que poner aquí} \\ \sum_{s' \in S} T(\ldots \\ \\ \ldots \end{matrix}$$

$V^{\pi} = f(V^{\pi})$ ↰

Problema de punto fijo

```
def_calcula_v(mdp, política, tolerancia=1e-6):
  inicializar(V(s) aleatorio ∀s ∈ S
  for _ in range (1 a 10):
        for s in S ← mdp.estados()
           vp = sum(mdp.sucesor(s, politica[s], s_p) * [mdp.recompensa(s, politica[s], s_p + mdp_descuento * V[s_p]
                                                  for s_p in mdp.estados())
           delta=max(delta, abs(vp, v[s])
           V[s]=vp
     if delta<tolerancia:
        break
return v
```

$$V^{\pi}(s) = \sum_{a \in A_{\pi}(S)} \pi(s,a) \sum_{s' \in S} T(s,a,s') \left[\gamma(s,a,s') + \gamma V^{\pi}(s') \right]$$

$V^{\pi}(s') \rightarrow E^{\pi}[R_{T+1} | S_{T+1} = s']$

$\gamma \leftarrow$ Factor de descuento

¿Cómo lo consigo? 12/Mar/26

Determine:

$\pi: S \rightarrow A$ // Estoy en un estado que acción tomo

$\pi: S \times A \rightarrow \mathbb{R}$ // Para un estado y acción, con que probabilidad realizo esa acción

¿Y si tengo 2 políticas π_1 y π, cuál es mejor?

Calculamos la función de valor de estado de π

V=Valor π=Política E=Esperanza

$$V^{\pi}(s) = E^{\pi}[R_T | S_T = s]$$

$$R_T = r_T + \gamma R_{T+1}$$

Función de valor de estado

$$V^{\pi}(s) = \sum_{s' \in S} T(s, \pi(s), s') \left[r(s, \pi(s), s') + \gamma \underbrace{E^{\pi}[R_T | S_T = s]}_{V^{\pi}(s')} \right]$$

$$V^{\pi}(s) = \sum_{a \in A_L(s)} \cdot \pi(s,a) \sum_{s' \in S} T(s,a,s') \left[r(s,a,s') + \gamma V^{\pi}(s') \right]$$

Si...

$V^{\pi_1}(s) \leq V^{\pi_2}(s) \quad \forall s \in S$

entonces $\pi_1 \leq \pi_2$

es mejor la política π_2

Tengo π_1 y π_2

y hago

$$\pi_3 \text{ t.q. } \pi_3(s) = \begin{cases} \pi_1(s) & \text{si } V^{\pi_1}(s) > V^{\pi_2}(s) \\ \pi_2(1) & \text{si } V^{\pi_2}(1) \geq V^{\pi_1}(1) \end{cases}$$

entonces $\pi_1 \leq \pi_3$ y $\pi_2 \leq \pi_3$

entonces $\exists \pi^*$ tal que

$V^{\pi^*}(s) \geq V^{\pi}(s)$

Función de valor de Estado_Acción

$$Q^{\pi}(s,a) = E^{\pi}[R_t | S_t = s, a_t = a]$$

$$Q^{\pi}(s,a) = \sum_{s' \in S} T(s,a,s') \left[r(s,a,s') + \gamma V^{\pi}(s') \right]$$

$$V^{\pi}(s) = \sum_{a \in A_L(s)} \pi(s,a) Q(s,a) \quad * \text{Punto fijo}$$

$$\begin{bmatrix} V^{\pi}(S_1) \\ V^{\pi}(S_2) \\ \vdots \\ V^{\pi}(S_n) \end{bmatrix} = \begin{matrix} \sum_{s' \in S} T(S_1, \pi(S_1), s' [r(S_1, \pi(S_1), s') + \gamma V^{\pi}(s')] \\ \sum_{s' \in S} T(S_2, \pi(S_2), s' [r(S_2, \pi(S_2), s') + \gamma V^{\pi}(s')] \\ \\ \sum_{s' \in S} T(S_n, \pi(S_n), s' [r(S_n, \pi(S_n), s') + \gamma V^{\pi}(s')] \end{matrix} \qquad V^{\pi} = f(V^{\pi})$$

13/Mar/26

$\pi_1(s,\rightarrow)=1 \quad \forall s$

$\pi_a(s,a)=1/4 \quad \forall s, \forall a$

$\top \gamma \gamma$

$$V^{\pi_1}(s_1)=p(s_1,\rightarrow,s_1)[-1+\gamma V^{\pi_1}(s_1)]+p(s_1,\rightarrow,s_2)[1]+$$
$$p(s_1,\rightarrow,s_3)[-1+\gamma V^{\pi_1}(s_3)]+p(s_1,\rightarrow,s_4)[-1+\gamma V^{\pi}(s_4)]$$

$$V^{\pi_1}(s_1)=0.05[-1+\gamma V^{\pi_1}(s_1)]+0.9+0.05[-1+\gamma V^{\pi_1}(s_3)]$$
$$V^{\pi_1}(s_3)=0.05[-1+\gamma V^{\pi_1}(s_1)]+0.9[-1+\gamma V^{\pi_1}(s_4)]+0.05[-1+\gamma V^{\pi_1}(s_3)]$$
$$V^{\pi_1}(s_4)=0.9[-1+\gamma V^{\pi_1}(s_4)]+0.05[-1+\gamma V^{\pi_1}(s_4)]+0.05[1]$$

Quiero establecer una política π

$\pi: S\times A\rightarrow \mathbb{R} \quad \pi(s,a)=Pr[A_T=a \mid S_T=s]$

¿Y cómo calculo que tan buena es una política?

- Función de valor de estado

$$V^{\pi}(s)=E^{\pi}[R_T \mid S_T=s]$$

$$R_T=r_T+\gamma r_{T+1}+\gamma^2 r_{T+2}+\gamma^3 r_{T+3}+\dots$$
$$=r_T+\gamma[r_{T+1}+\gamma r_{T+2}+\gamma^2 r_{T+3}+\dots]$$
$$=r_T+\gamma R_{T+1}$$

$$V^{\pi}(s)=\sum_{a\in A_\ell(s)}\pi(s,a)\sum_{s'\in S}T(s,a,s')[\gamma(s,a,s')+\gamma \underbrace{V^{\pi}(s')}_{E^{\pi}[R_{T+1}\mid S_{T+1}=s']}]$$

$\pi_1 \propto \pi_2$

$V^{\pi_1}(s)\leq V^{\pi_2}(s) \quad \forall s$

π_1 y π_2

π_3 tal que $\pi_3(s)=\begin{cases}\pi_1(s) \text{ si } V^{\pi_1}(s)\geq V^{\pi_2}(s)\\ \pi_2(s) \text{ en otro caso}\end{cases} \quad \forall s\in S$

$\pi_1\leq\pi_3$

$\pi_2 \propto \pi_2$

π^* es una política optima ssi

$V^{\pi^*}(s)\geq V^{\pi}(s) \quad \forall \pi$

Función de valor estado-acción

$$Q^{\pi}(s,a) = E^{\pi}[R_T \mid S_T = s, A_T = a]$$

$$Q^{\pi}(s,a) = \sum_{s' \in S} T(s,a,s')\left[r(s,a,s') + \gamma V^{\pi}(s')\right]$$

$$V^{\pi}(s) = \sum_{a \in A_L(s)} \pi(s,a)\, Q^{\pi}(s,a)$$

$$\pi^* \rightarrow V^* Q^*$$

$$\pi^*(s,a) = \begin{cases} 1 & \text{para } a^* \text{ tal que } a^* = \arg\max_a Q^*(s,a) \\ 0 & \text{en otro caso} \end{cases}$$

$$V^*(s) = \max_a Q^*(s,a)$$

$$V^*(s) = \sum_{s' \in S} T(s,a^*,s')\left[r(s,a^*,s') + \gamma \underbrace{\max_{a'} Q^*(s,a)}_{V^*(s')}\right]$$

Ecuación de optimalidad de Bellmann

$$V^*(s) = \max_a \left(\sum_{s' \in S} T(s,a,s')\left[r(s,a,s') + \gamma V^*(s')\right] \right)$$

Asueto — 16/Mar/26

Plática en el auditorio — 17/Mar/26

Evento: Smart City Hackathon — 18/Mar/26

Evento: Smart City Hackathon — 19/Mar/26

Markov desition process — 20/Mar/26

$s \in S \longrightarrow$ Estado

$a \in A$

$A_L : S \rightarrow \mathcal{P}(A)$

$T: S \times A \times S \to \mathbb{R}$ tal que $T(s,a,s') = Pr(S_{t+1} = s' | S_T = S, A_T = a)$ → Función de transición

$r: S \times A \times S \to \mathbb{R}$ tal que $r(s,a,s')$ es la recompensa de ir de s a S con la acción a

$S_T \subseteq S$ → Conjunto de estados terminales

$0 \leq r \leq 1$ Factor de descuento

Quiero encontrar una política

$\pi: S \times A \to \mathbb{R}$

$\pi(s,a) = Pr(A_t = a | S_T = s)$

tal que

$E^{\pi}[R_T]$ sea máxima

$R_T = r_t + r r_{T+1} + r^2 r_T + 2 + \ldots = r_T + r R_{T+1}$

Si $V^{\pi}(s) = E^{\pi}[R_T | S_T = S]$

$\pi_1 \leq \pi_2$ si

$V^{\pi_1}(s) \leq V^{\pi_2}(s) \quad \forall_s \in S$

$$V^{\pi}(S) = \sum_{a \in A_1(S)} \pi(s,a) \sum_{s' \in S} T(s,a,s') [r(s,a,s') + r V^{\pi}(s')]$$

```
def iter_politica(mdp, π, max_it=1e3, tol=1e-3)
  V = {s=0 for s in mdp.S}
  for _ in range(max_it):
    err=0
     for s in mdp.S
       vt = sum(π(s,a) * sum(mdp.T(s,a,s') * [mdp.r(s,a,s') + mdp.r * V[s']
                                for s' in mdp.S
                                )
                  for a in mdp.a.legales(s)
                )

err = max(err, abs(V[s] - Vt))
V[s] = Vt

if err < tol:
   break
return V
```

$$Q^{\pi}(s,a) = E^{\pi}[R_T \mid S_T = S, A_T = a]$$

$$Q^{\pi}(s,a) = \sum_{s' \in S} T(s,a,s') \left[r(s,a,s') + r \underbrace{\sum_{a' \in A_1(s')} \pi(s',a') Q^{\pi}(s',a')}_{V^{\pi}(s)} \right]$$

$$V^{\pi}(s) = \sum_{a \in A_1(a)} \pi(s,a) Q(s,a)$$

$\pi \leq \pi^* \quad \forall \pi \in \Pi$

π^* es la política óptima

V^* y Q^* son las funciones de valor óptimas

$V^*(s) = V^{\pi^*}(s)$, $Q^*(s,a) = Q^{\pi^*}(s,a)$

$$V^*(s) = \max_{a \in A_1(s)} Q^*(s,a)$$

$Q^*(s,a_1) = 5$

$Q^*(s,a_2) = 7$

$V^*(s) = \pi(s,a_1) * 5 + \pi(s,a_2) * 7$

$V^*(s) = 4 * 5 + .6 * .7$

$$\pi(s,a) = \begin{cases} 1 & \text{Si } a = \arg\max_{a \in A_1(s)} Q^*(s,a) \\ 0 & \text{o.c} \end{cases}$$

$$Q^*(s,a) = \sum_{s \in S} T(s,a,s') \left[r(s,a,s') + r \max_{a \in A_1(s)} Q^*(s',a') \right]$$

Eq. de Optimalidad de Bellman

Eq. de Optimalidad de Bellman 23/Mar/26

$$Q^*(s,a) = \sum_{s \in S} T(s,a,s') \left[r(s,a,s') + r \max_{a \in A_1(s)} Q^*(s',a') \right]$$

$$V^*(s) = \max_{a \in A_t(i)} Q^*(s,a)$$

```
def iter_valor(MDP, max_iter=1e5, tol=1e-6)
Q={(s,a):0 for s in MDP.estados() for a in MDP.acciones_legales(s)}
 for _ in range(max_iter)
  err=0
```

```
for (s,a) in Q.keys()
  q = sum(MDP.T(s,a,s')*(MDP.r(s,a,s')
   +MDP.r*max(Q[(s',a')] for a' in MDP.acciones_legales(s'))
   for s' in MDP.estados())
 err=max(err, abs(Q[(s,a)]-q))
 Q[(s,a)]=q
 if err<tol:
    break

return Q

def genera_política_greedy(MDP,Q):
 π={}
for s in MDP.estados():
π[s]:max(a for a in MDP.acciones_legales(s),
          key=lambda a:Q[(s,a)])

 return π
```

Examen intermedio del curso (Cuestionario en Teams) — 24/Mar/26

25/Mar/26

$$V^*[s] = \max_{a \in A_L(s)} \sum_{s' \in S} T(s,a,s')[r(s,a,s') + \gamma V^*(s')]$$

$$\pi(s) = \arg \max_{a \in A_L(s)} \underbrace{\sum_{s' \in S} T(s,a,s')[r(s,a,s') + \gamma V^*[s']]}_{Q^*(s,a)}$$

Analizamos código MDPs y RL

No hubo clase — 26/Mar/26

Blackjack

27/Mar/26

Estados:

$S=(n_1, n_2, a_1, a_2, d, n_3, a_3, d_1)$

- n_1: Número tuyo
- n_2: Numero de el
- a_1: As tuyos
- a_2: As oponente
- d: Doblar

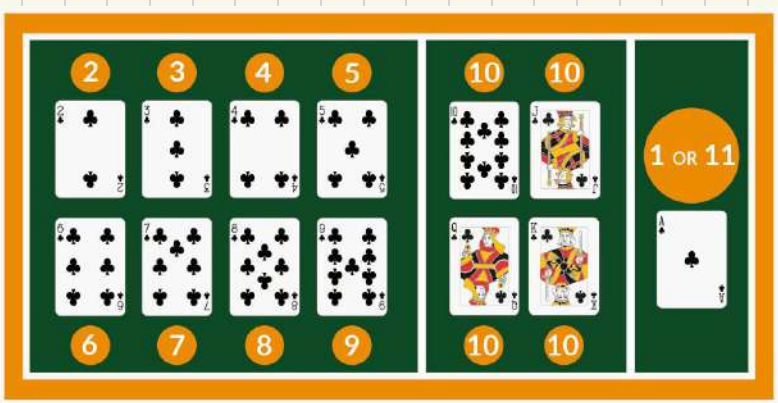
$As = 11$ o 1

Si llega a 21 pierde

≤ 17 se pide a la casa

A	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1/14	1/14									1/14	3/14			

Factor de descuento: 0 porque es continuo



MDP

6/Abr/26

$S = \{S_1, \ldots, S_n\}$

$A = \{a_1, \ldots, a_n\}$

$A_L: S \to P(A)$

$S_T \subseteq S$ estados terminales

$T = S \times A \times S \to \mathbb{R}$ $\quad T(s,a,s') = P_r(S_{t+1} = s' \mid S_t = s, A_t = a)$

$r = S \times A \times S \to \mathbb{R}$ $\quad r(s,a,s')$

γ: $\gamma \in [0,1]$ Factor de descuento

$$\sum_{s' \in S} T(s,a,s') = 1$$

$\pi : S \times A \to \mathbb{R} \quad \pi(s,a) = Pr(A_1 = a \mid S_t = s) \quad \sum_{a \in A_L(s)} \pi(s,a) = 1$

$\pi^* = \arg\max_{\pi} E^{\pi}[R_T] \quad \forall s \in S$

$E^{\pi}[r_T + \gamma r_{T+1} + \gamma^2 r_{T+2} + \dots] \quad \forall s \in S$

$E^{\pi}[r_T + \gamma R_{T+1}] \forall s \in S$

$V^{\pi}(s) = E^{\pi}[r_T + \gamma R_{T+1}] = \sum_{a \in A_L(s)} \pi(s,a) \sum_{s' \in S} T(s,a,s')[r(s,a,s') + \gamma V^{\pi}(s')]$

Ecuación de optimalidad de Bellmann

$V^*(s) = \max_a \left(\sum_{s' \in S} T(s,a,s')[r(s,a,s') + \gamma V^*(s')] \right)$

$\pi^*(s,a) = \begin{cases} 1 \text{ si } a = \arg\max_a \sum_{s' \in S} T(s,a,s')[r(s,a,s') + \gamma V^*(s')] \\ 0 \text{ en otro caso} \end{cases}$

MDPsim

$S = \{S_1, \dots, S_n\}$

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

$A_L : S \to P(A)$

$S_T \subseteq S$

$s' = \text{sucessor}(s,a)$

$r : S \times A \times S \to \mathbb{R} \quad r(s,a,s')$

$\gamma : \gamma \in [0,1]$

¿Cómo evaluar una política π con MDPsim?

$V^{\pi}(s) = \sum_{a \in A_L}(s,a) \sum_{s' \in S} T(s,a,s')[r(s,a,s') + \gamma V^{\pi}(s)]$

asumo que conozco $\hat{V}^{\pi}(s)$ y estoy en el estado S

escojo a a partir de π

aplico a a MDPsim y obtengo s' y r

$s' = \text{sucesor}(s,a)$

$r = r(s,a,s')$

entonces $\hat{V}^{\pi}_{t+1}(s) = r + \gamma V^{\pi}(s')$

$\hat{V}^{\pi}(s) \leftarrow \hat{V}^{\pi}(s) + \alpha [\hat{V}^{\pi}_{t+1}(s) - \hat{V}^{\pi}(s)]$

Diferencia temporal

π determinista

```
a = π(s)
```

π estocástico

```
umbral = random.ran()
nivel = 0
for a in A_e(S)
  nivel t= π(s,a)
if umbral <= nivel:
    return a
raise.ValueError(___)
```

No hubo clase 7/Abr/26

8/Abr/26

```
class MDPsim:
  def __init__(self, s, r):
    self s = s
    self r = r

def acciones_legales(self,s):
   return iterable con acciones legales en S

def sucesor(self,s,a):
   return r

def terminal(self,s):
  return True/False

class Política
 def __init__(self, p_dict):
   self.pi = p_dict
```

```
def __call__(self,s):
  return self.pi[s]
```

$$V^{\pi}(s) = \sum_{a \in A_L(s)} \pi(s,a) \sum_{s' \in S} T(s,a,s')\left[r(s,a,s') + \gamma V^{\pi}(s')\right]$$

```
def estado_inicial(self)
  return estado
```

ya tengo $\hat{V}^{\pi}(s)$ para un estado s

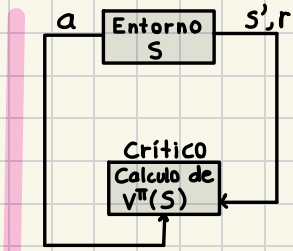
Aplico:

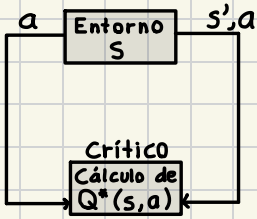
```
a = política(s)
s' = MDPsim.sucesor(s,a)
r = MDPsim.recompensa(s,a,s')
```

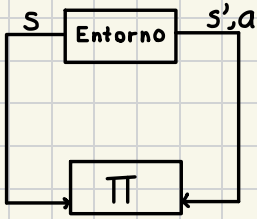
$V = r + MDP_{sim}.\gamma * V^{\pi}(s')$

$\hat{V}^{\pi}(s) = \hat{V}^{\pi}(s) + \alpha\left[V - \hat{V}^{\pi}(s)\right]$

```
def TD_0(mdl, π, α, max_episodios, max_it
  V = {0 for s in MDL.S}
  for _ in range(max_episodios)
   err = 0
   s = mdl.estado_inicial()
  for _ in range(max_it):
    a = π(s)
    s' = mdl.sucesor(s,a)
    r = mdl.sucesor(s,a,s')
    v = V[s] - d*(r - mdl.γ * V[s'] - V[s])
    err = max(err, abs(V - V[s]))
    V[s], s = V, s'
    if mdl.terminal(s)
        break

   if err < tol:
     break

  return V
```







$$Q^*(s,a) = \sum_{s \in S} T(s,a,s')\left[r(s,a,s') + \gamma \sum_{a' \in A_L(s')} \pi(s';a')Q(s';a')\right]$$

$$Q^*(s,a) = \sum_{s \in S} T(s,a,s')\left[r(s,a,s') + \gamma \max_{a' \in A_L(s)} Q^*(s';a')\right]$$

Tengo $Q(s,a)$

aplico:

$s' = mdl.sucesor(s,a)$

$r = mdl.recompensa(s,a,s')$

$$\begin{cases} a' = \arg\max_{a'} Q(s',a') \\ a' = politica(s') \end{cases}$$

$q = r + mdl.\gamma * Q(s',a')$

$Q(s,a) = Q(s,a) + \alpha(q - Q(s,a))$

Tengo $\hat{V}^{\pi}(S)$ para un estado S

9/Abr/26

Aplico:

$a = politica(S)$

$s' = MDP_{sim}.sucesor(s,a)$

$r = MDP_{sim}.recompensa(s,a,s')$

$v = r + MDP_{sim}.\gamma * V^{\pi}(s')$

$$\hat{V}^{\pi}(s) = \hat{V}^{\pi}(s) + \alpha[v - \hat{V}^{\pi}(s)]$$

$$\pi(s) = \arg\max_{a \in A_L(s)} Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s \in S} T(s,a,s')\left[r(s,a,s') + \gamma \sum_{a' \in A_L(s')} \pi(s';a')Q(s';a')\right]$$

$$Q^*(s,a) = \sum_{s \in S} T(s,a,s')\left[r(s,a,s') + \gamma \max_{a' \in A_L(s)} Q^*(s';a')\right]$$

SARSA: On-policy
Q-Learning: off-policy

```
class PoliticaEGreedy (Politica)
  def __init__(self, ε):
      self.ε = ε
  def __call__(s,Q):
      if random.random( ) < ε:
          return random.choice(A)
           return max(a in A, key=lambda a:Q[(s,a)])

def SARSA(mdl, num_ep, num_it, α, tol, ε)
    π= PoliticaEGreedy(ε)
    Q={(tuple(s), tuple(m)):0 for s in S for a in mdl.acciones_legales(s)}
    for _ in range(num_ep)
       err=0
       s=mdl.estado.inicial( )
       a=π(S, mdl.acciones_legales(s), Q)
        for _ in range(num_it)
          s'= mdl.sucesor(s,a)
          r= mdl.recompensa(s,a,s')
          a'= politica(s', mdl.acciones_legales(S'), Q)
          q= r+mdl.δ*Q[(s',a')]
          err=max(err, abs(Q[(s,a)]-q))
          s,a=s',a'
          if mdl.terminal(s)
            break
    if err<tol:
       break
return Q

def Qlearning(mdl, num_ep, num_it, α, tol, ε):
    π=PoliticaEGreedy(ε)
    Q={(s,a) for s in mdl.S for a in mdl.acciones.legales(s)}
    for _ in range(num_ep)
     err=0
```

```
    S = mdl.estado_inicial()
    for _ in range (num_it):
        a = π(S, mdl.acciones_legales(S), Q)
        s' = mdl.sucesor(s,a)
        r = mdl.recompensa(s,a,s')
        q = r + mdl.γ * max(Q[s',a'] for a' in mdl.acciones_legales(s'))
        err = max(err, abs(q, Q[(s,a)]))
        s = s'
        if mdl.terminal(s):
            break
    if err < tol:
        break
return Q
```

Analizar código MDPs y RL 10/Abr/26

Próximamente:
- Tarea programación dinámica
- Tarea aprendizaje por refuerzo
- Blackjack

Optimización 13/Abr/26

(Búsquedas locales)

$f: X \to \mathbb{R}$

y lo que quiero es encontrar una x^* tal que

$$f(x^*) = \max_{x \in X} f(x)$$

$$f(x^*) = \min_{x \in X} -f(x)$$

$$A = \begin{bmatrix} 0 & a_{12} & a_{1n} \\ a_{21} & \ddots & \ddots \\ \vdots & \ddots & 0 \\ a_{m1} & & \end{bmatrix}$$

lo que quiero es encontrar un circuito cerrado

X=permutacion de [1,...,n]

donde

$f(x) = \sum_{i=1}^{n-1} a_{x[i],x[i+1]} + a_{x[n],x[0]}$

[1,2,3,4,5,6,7,8,9]
[10,1,4,6,9,3,8,2,7]

y $x^* = \min_x f(x)$

$f: X \to \mathbb{R}$

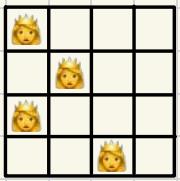
Un máximo global es:

$x^* = \arg\max_{x \in X} f(x)$

$x'_t = \lim_{\Delta \to 0} x_1 - \Delta$

$x'_z = \lim_{\Delta \to 0} x_1 + \Delta$

Tablero 4x4



(1,2,3,4)
(2,1,3,4)
(3,2,1,4)
(1,3,2,4)

} Permutaciones

Debemos de poner n reinas pero no pueden comerse entre sí. Ent. debe haber una en cada renglón

Si existe una topología en X

Un máximo local es:

$x^* = \arg\max_{x \in Vecinos(x)} f(x)$

Cuando el gradiente es cero es un máximo, mínimo o una "silla"

14/Abr/26

Inicializa $x \in X$ en forma aleatoria

e max ← f(x)

para max, epoch iteraciones

Lo implementaremos minimizando el costo

```
class PbOptim( ):
 def costo (self,x):
    if not self.es_estado(x):
       raise ValueError(" ")
     return f(x)
 def es_estado(self,x):
    return True/False
 def estado_aleatorio(self):
    return estado
 def vecinos(self,x):
       return iterable con los estados vecinos de x
 def vecino_aleatorio(self,x)
       return un vecino de x

def BusquedaAleatoria(pb,num_iter):
   x=pb.estado_aleatorio( )
   c=pb.costo(x)
   for _ in range(num_iter):
      otro=pb.estado_aleatorio( )
      c_otro=pb.costo(otro)
      if c_otro<C:
         x,c=otro,c,otro
```

Con esto modificamos la búsqueda aleatoria... para encontrar los mínimos locales

```
   def descenso_colinas(pb,num_iter):
      x=pb.estado_aleatorio( )
      c=pb.costo(x)
      for _ in range(num_iter):
         termina=True
         for vecino in pb.vecinos(x):
            c_vecino=pb.costo(vecino)
            if c_vecino<C:
               x,c=vecino,c_vecino
               terminal=False
         if terminal
            break
return x
```

```
def temple_simulado(pb, Tmax, calendarizador, T min):
    x = pb.estado_aleatorio()
    c = pb.costo
    T = Tmax
    while (T > Tmin)
        vecino = pb.vecino_aleatorio(x)
        c_vecino = pb.costo(vecino)
        inc = C - c_vecino
        if inc >= 0 or random.random() < math.exp(inc/T)
            x, c = vecino, c_vecino

        T = calendarizador(T, i)
        i += 1                                  Tmax / i+1
    return x, c

def calendarizador(Tmax, i):                    Tmax / ln(n+1) + 1
    return Tmax/(math.log(it1)+1)
```

15/Abr/26

población: $\{X_1, X_a, \ldots, X_p\}$ $X_i \in X$

caract(X_i): $[V_{i1}, V_{ia}, \ldots, V_{in}]$

adaptación: $X \to \mathbb{R}^+$ adaptación(X_i) (Inversamente proporcional al costo)

$$\text{adaptación}(x_1) = \frac{1}{\text{costo}(X_i)}$$

$$= \cdot K - \text{costo}(X_i)$$

$$= \max(0, K - \text{costo}(X_i))$$

selección: -ruleta
-torneo

cruza $[V_{i1}, V_{i-1}, \ldots, V_{in}]$
$[V_{j1}, V_{ja}, \ldots, V_{jn}]$

$[V_{h1}, V_{ha}, \ldots, V_{hn}]$

Mutación
Elitismo

Análisis de codigo: Búsquedas locales

16/Abr/26

Algoritmos genéticos:

-Le das un problema y tienes una población

[5 3 2 | 4 7 8 10 | 6 1 9]

[4 5 6 | 8 3 9 2 | 10 7 1]

[5 3 2 | 8 4 7 10 | 6 1 9]