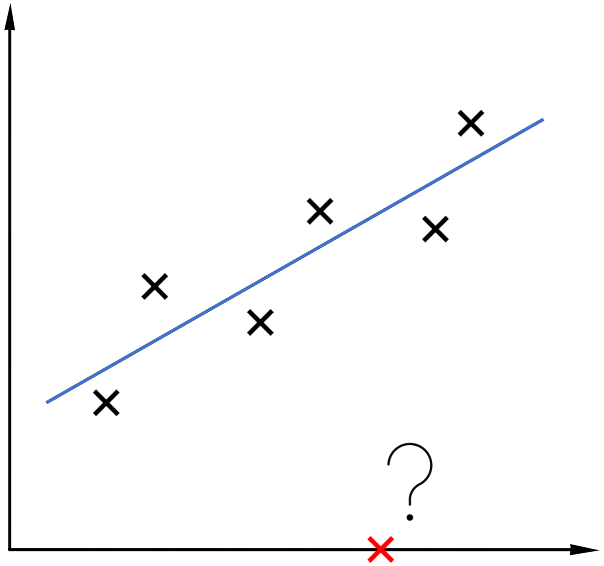
HANDS ON 04 – EXTREMOS

COMPLEMENTOS DE MATEMÁTICAS

Gonzalo Rubio
g.rubio@upm.es

Predicción basada en datos



- ¿Cómo obtengo esa recta?
  - Camino 1: Álgebra lineal
  - Camino 2: Cálculo (de varias variables)
  - Camino 3: Machine learning

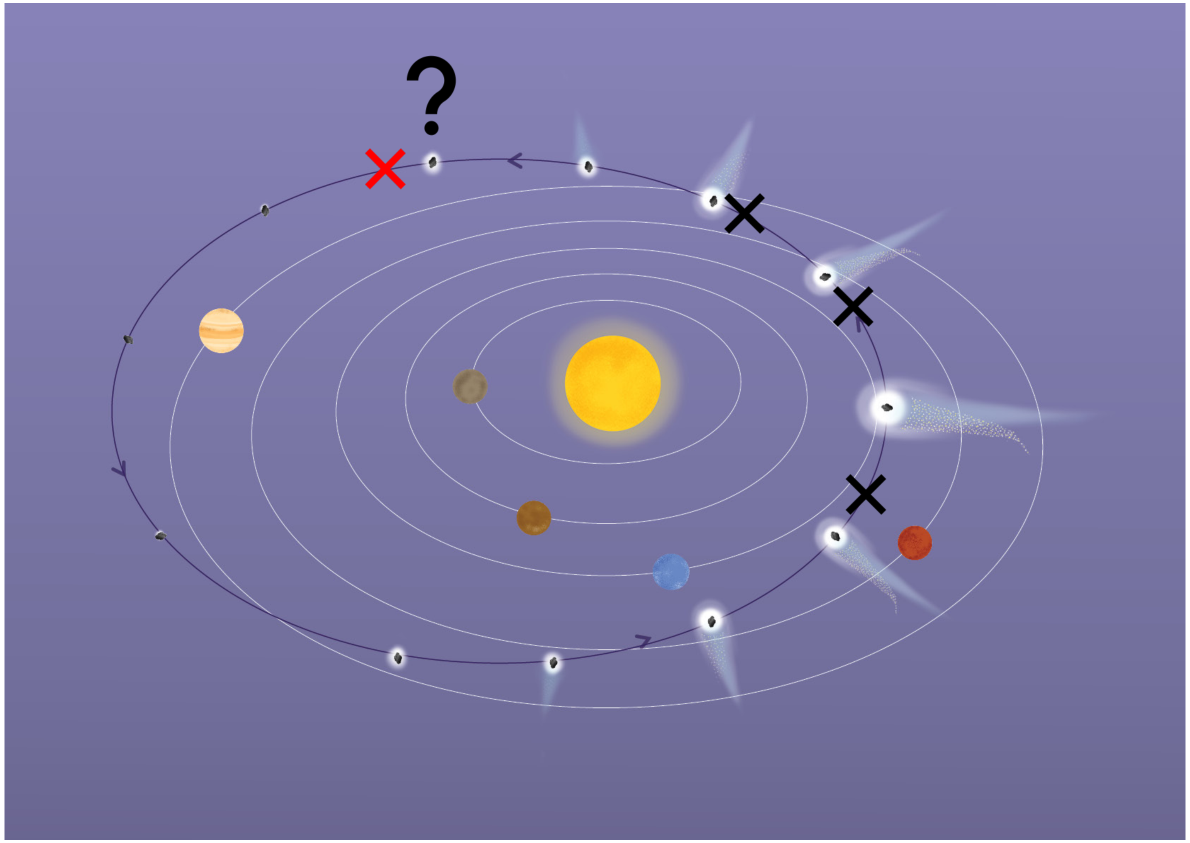
Mínimos cuadrados. Historia.

- Principios del siglo XIX
  - Adrien-Marie Legendre introduce el método de mínimos cuadrados
  - Legendre trabajó en la predicción de la trayectoria de proyectiles, midiendo la forma de la tierra y en astronomía.
  - Problema recurrente ajuste de datos contaminados por error.

Retrato usado durante 200 años para representar a Adrien-Marie Legendre

En 2005 se descubrió que que este es en realidad un político llamado Louis Legendre

Mínimos cuadrados. Historia.



Solución: tener los errores en cuenta.

Usar la ecuación orbital que mejor ajustara los datos, teniendo en cuenta que contenían un error.

Método fue publicado en 1805 y fue todo un éxito. En solo 10 años se había convertido en una herramienta estandar en Francia, Italia y Prusia.

¿Cómo obtengo esa recta? Álgebra lineal

$(1,1)\,,(2,2)\,,(3,2)$

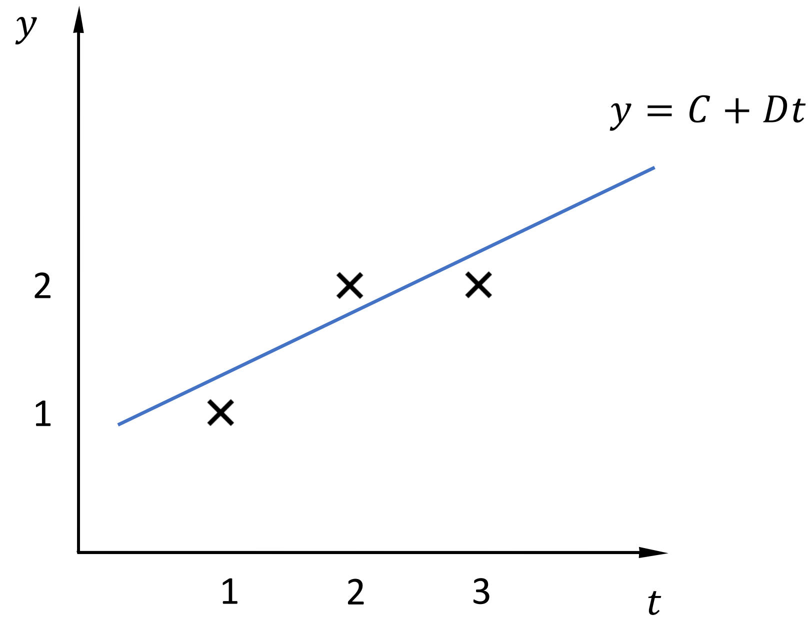
$$1 = C + D$$
$$2 = C + 2D$$
$$2 = C + 3D$$

No parece que este sistema tenga solución

$$\boldsymbol{Ax} = \boldsymbol{b}$$

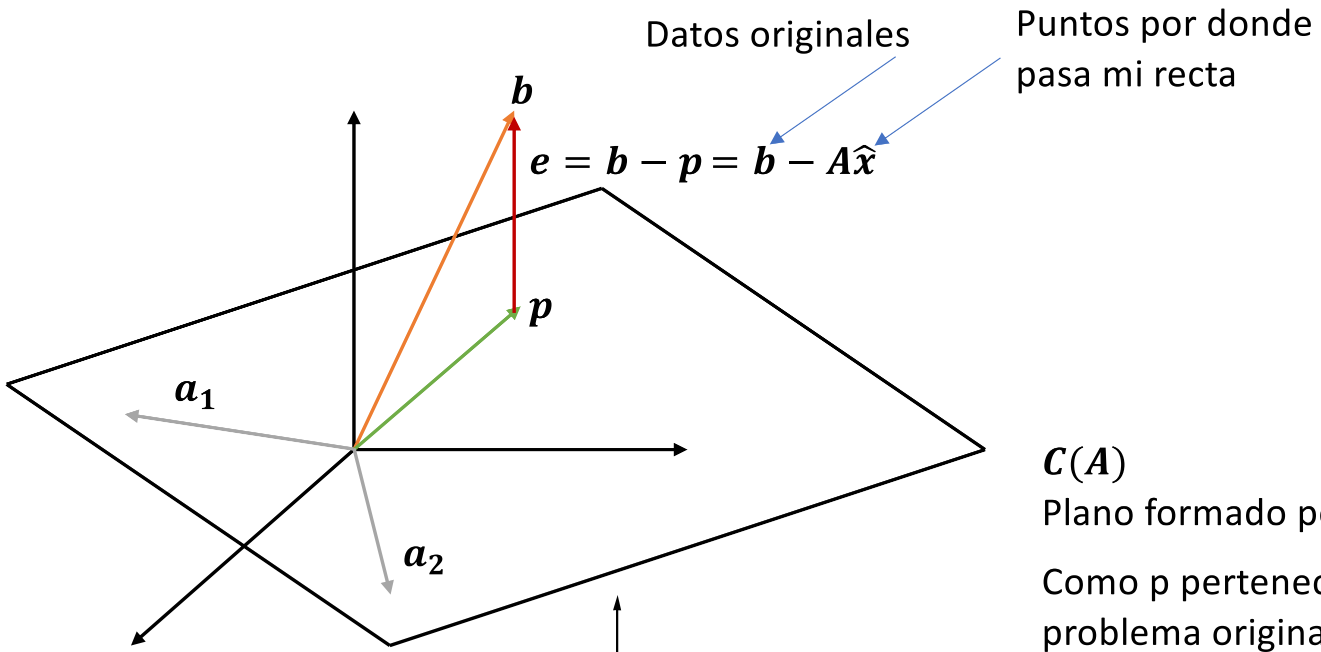
$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

Efectivamente, $\boldsymbol{b}$ no pertenece a $\boldsymbol{C}(\boldsymbol{A})$, luego el sistema no tiene solución



Hagamos que $\boldsymbol{b}$ pertenezca a $\boldsymbol{C}(\boldsymbol{A})$

Recordatorio: proyección ortogonal



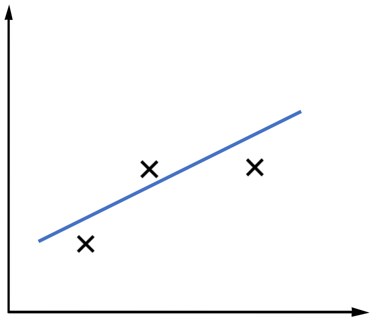
$$\boldsymbol{Ax} = \boldsymbol{b}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

$$\boldsymbol{A} = \begin{bmatrix} \vdots & \vdots \\ \boldsymbol{a_1} & \boldsymbol{a_2} \\ \vdots & \vdots \end{bmatrix}$$

$\boldsymbol{C}(\boldsymbol{A})$
Plano formado por los vectores columna de $\boldsymbol{A}$

Como p pertenece a $\boldsymbol{C}(\boldsymbol{A})$, puedo usarlo de RHS del problema original $\boldsymbol{Ax} = \boldsymbol{b}$ -> $\boldsymbol{A}\hat{\boldsymbol{x}} = \boldsymbol{p}$ y obtener solución



No puedo pasar por los puntos $\begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$.

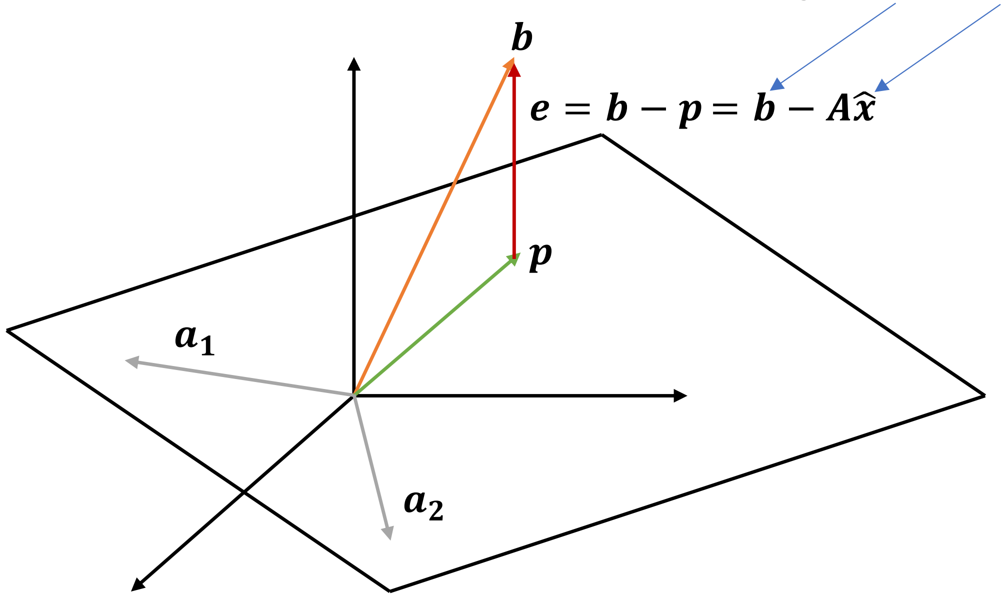
Intento pasar por unos alineados y que estén **lo más cerca posible** de los originales.

Recordatorio: proyección ortogonal

Datos originales

Puntos por donde pasa mi recta

$$\boldsymbol{e} = \boldsymbol{b} - \boldsymbol{p} = \boldsymbol{b} - \boldsymbol{A}\widehat{\boldsymbol{x}}$$



$$\boldsymbol{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \qquad \|\boldsymbol{e}\|^2 = e_1^2 + e_2^2 + e_3^2 \qquad \text{Mínimo}$$

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

$$\boldsymbol{A} = \begin{bmatrix} \vdots & \vdots \\ \boldsymbol{a_1} & \boldsymbol{a_2} \\ \vdots & \vdots \end{bmatrix}$$

$$\boldsymbol{p} = \hat{x}_1\boldsymbol{a_1} + \hat{x}_2\boldsymbol{a_2} = \boldsymbol{A}\widehat{\boldsymbol{x}}$$

Si quiero conocer $\boldsymbol{p}$, mi problema es encontrar $\widehat{\boldsymbol{x}}$

CLAVE: $\boldsymbol{e} = \boldsymbol{b} - \boldsymbol{p} = \boldsymbol{b} - \boldsymbol{A}\widehat{\boldsymbol{x}}$ es perpendicular al plano (**para que $\boldsymbol{b}$ y $\boldsymbol{p}$ estén lo más cerca posible**)

Luego debe ser perpendicular a $\boldsymbol{a_1}$ y también a $\boldsymbol{a_2}$

Recordatorio: proyección ortogonal

$$a_1^T(b - A\hat{x}) = 0$$
$$a_2^T(b - A\hat{x}) = 0$$
$$\longrightarrow \quad A^T(b - A\hat{x}) = \mathbf{0}$$

Reordenando términos:

$$A^T A\hat{x} = A^T b$$

Luego $\hat{x}$

$$\hat{x} = (A^T A)^{-1} A^T b$$

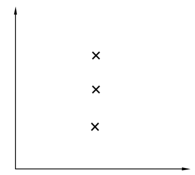
Y p

$$p = A(A^T A)^{-1} A^T b$$

Y la ecuación a resolver es:

$$A\hat{x} = A(A^T A)^{-1} A^T b$$

$(A^T A)$ es invertible si A tiene columnas independientes



$$Ax = b$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

$$A = \begin{bmatrix} \vdots & \vdots \\ a_1 & a_2 \\ \vdots & \vdots \end{bmatrix}$$

$$p = \hat{x}_1 a_1 + \hat{x}_2 a_2 = A\hat{x}$$

Si quiero conocer p, mi problema es encontrar $\hat{x}$

CLAVE: $e = b - p = b - A\hat{x}$ es perpendicular al plano (**para que b y p estén lo más cerca posible**)

Luego debe ser perpendicular a a_1 y también a a_2

PEQUEÑO INCISO

$(\boldsymbol{A^T A})$ es invertible si $\boldsymbol{A}$ tiene columnas independientes

Demostración:

$\boldsymbol{A^T A}$ es invertible implica que $\boldsymbol{A^T A x = 0}$ si y solo si $\boldsymbol{x = 0}$

IDEA FELIZ

$$\boldsymbol{x^T A^T A x = 0}$$

$$\boldsymbol{(Ax)^T A x = 0}$$

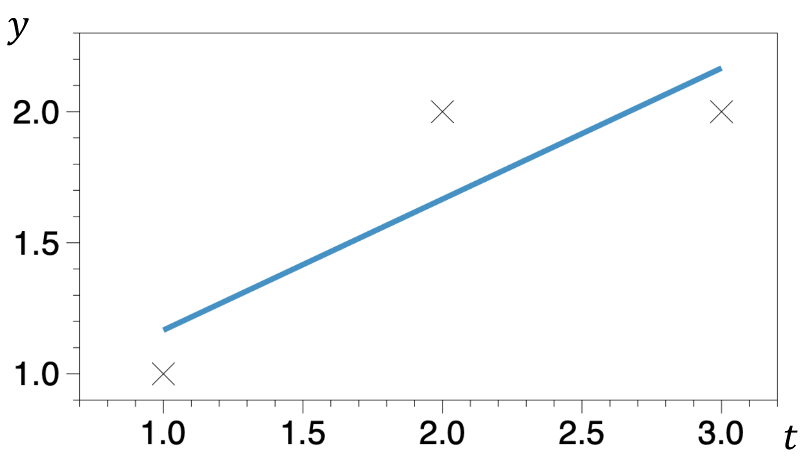
$$||\boldsymbol{Ax}||^2 = \boldsymbol{0}$$

Las columnas de $\boldsymbol{A}$ son independientes luego esto solo se cumple si $\boldsymbol{x = 0}$

Recta de regresión lineal

$$\boldsymbol{Ax} = \boldsymbol{b} \qquad \boldsymbol{A^T A\hat{x}} = \boldsymbol{A^T b}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} \hat{C} \\ \widehat{D} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$



$$\begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix} \begin{bmatrix} \hat{C} \\ \widehat{D} \end{bmatrix} = \begin{bmatrix} 5 \\ 11 \end{bmatrix}$$

Ecuaciones normales

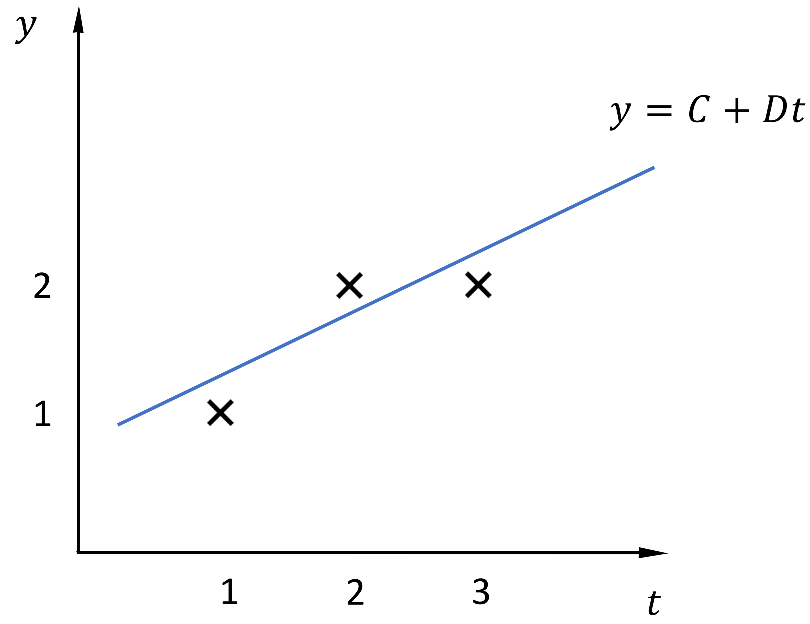
$$y = 2/3 + 1/2t$$

Recta de regresión lineal. Un poco de Código.

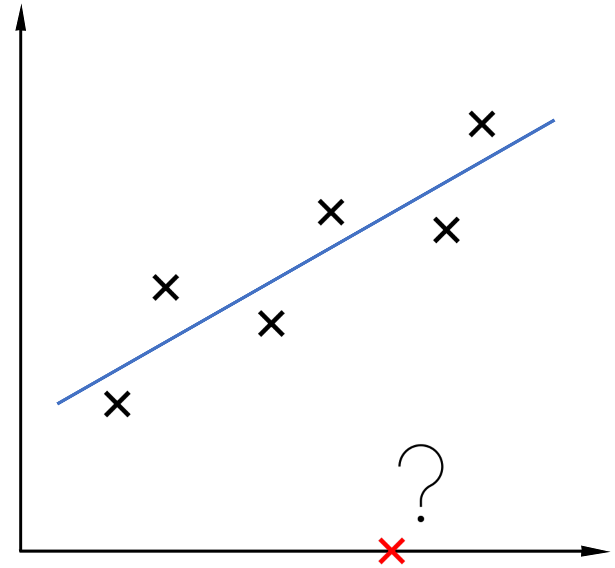
$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix} \qquad \begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ 1 & t_3 \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

```python
import numpy as np


A = np.matrix([[1, 1], [1, 2], [1, 3]])
b = np.matrix([1,2,2])
b = b.transpose()

AT = A.transpose()

Ahat = np.matmul(AT,A)
bhat = np.matmul(AT,b)

x = np.linalg.solve(Ahat, bhat)
```

$$\boldsymbol{A^T A \hat{x} = A^T b}$$



Predicción basada en datos

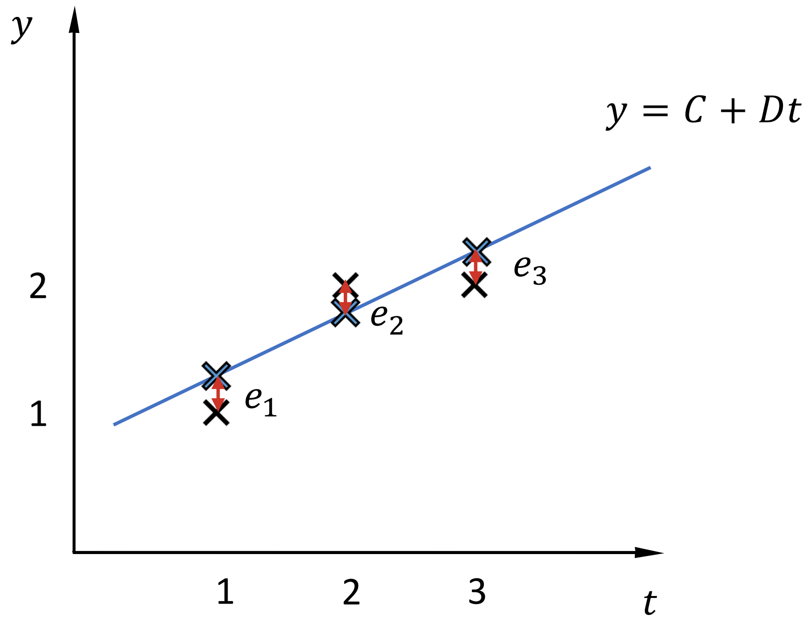


- ¿Cómo obtengo esa recta?
  - Camino 1: Álgebra lineal
  - Camino 2: Cálculo (de varias variables)
  - Camino 3: Machine learning

¿Cómo obtengo esa recta? Cálculo

$e_1 = C + D - 1$
$e_2 = C + 2D - 2$
$e_3 = C + 3D - 2$

$$\boldsymbol{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix}$$



Para que la recta sea "buena" $\|\boldsymbol{e}\|$ debe ser pequeño

Es útil trabajar con $\|\boldsymbol{e}\|^2$. Buscamos su mínimo

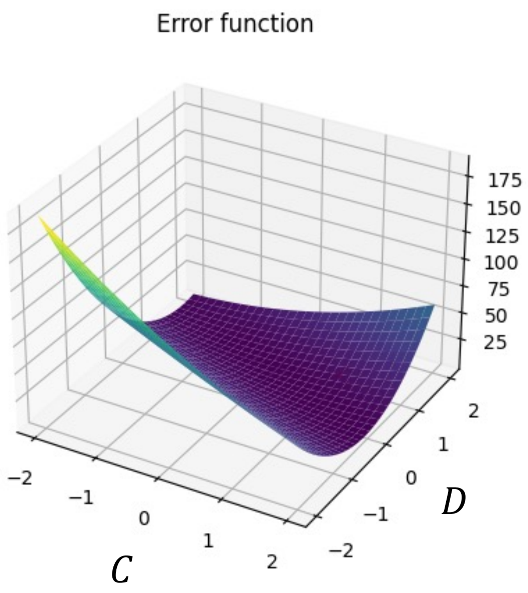
$$\|\boldsymbol{e}\|^2 = (C + D - 1)^2 + (C + 2D - 2)^2 + (C + 3D - 2)^2$$

Esto es una función real de varias variables $\|\boldsymbol{e}\|^2 = f(C, D)$

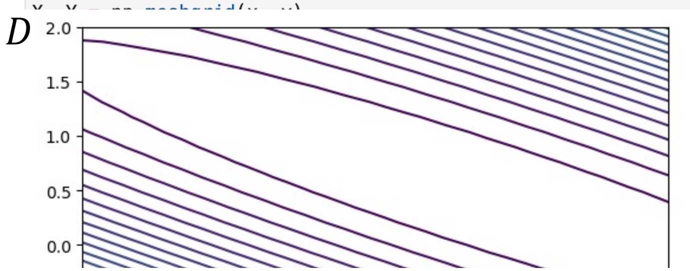
El error es una función de varias variables

$$\|\boldsymbol{e}\|^2 = (C + D - 1)^2 + (C + 2D - 2)^2 + (C + 3D - 2)^2$$

Esto es una función real de varias variables $\|\boldsymbol{e}\|^2 = f(C, D)$




```
def f(C, D):
    return np.square(C+D-1)+np.square(C+2*D-2)+np.square(C+3*D-2)

x = np.linspace(-2, 2, 30)
y = np.linspace(-2, 2, 30)
```



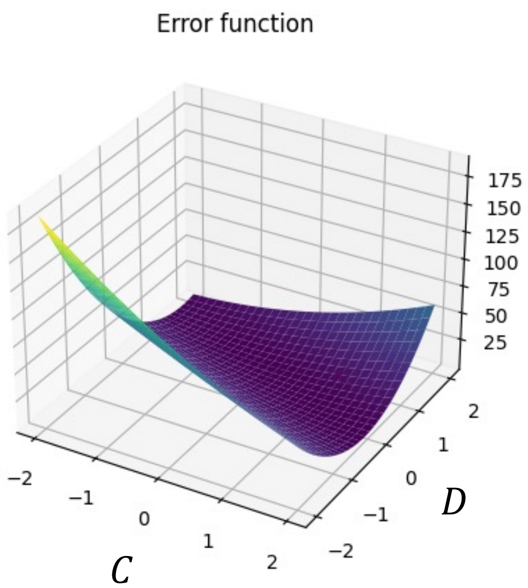

```
color="red", markerfacecolor="red")
```

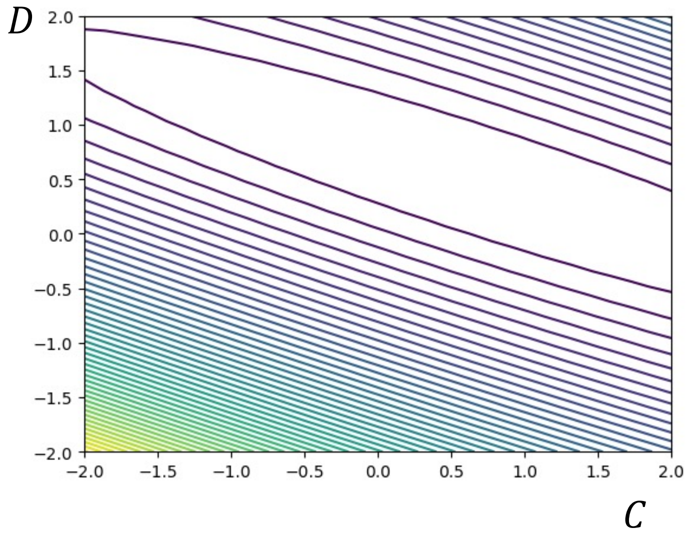
```
def f(C, D):
    return np.square(C+D-1)+np.square(C+2*D-2)+np.square(C+3*D-2)

x = np.linspace(-2, 2, 30)
y = np.linspace(-2, 2, 30)

X, Y = np.meshgrid(x, y)
Z = f(X, Y)

fig = plt.contour(X, Y, Z, 50, cmap='viridis')
#plt.plot(-1,-1, marker="x", markersize=10, markeredgecolor="red", markerfacecolor="red")
```

Relación entre función de error y recta

$\|\boldsymbol{e}\|^2 = (C + D - 1)^2 + (C + 2D - 2)^2 + (C + 3D - 2)^2$

Esto es una función real de varias variables $\|\boldsymbol{e}\|^2 = f(C, D)$





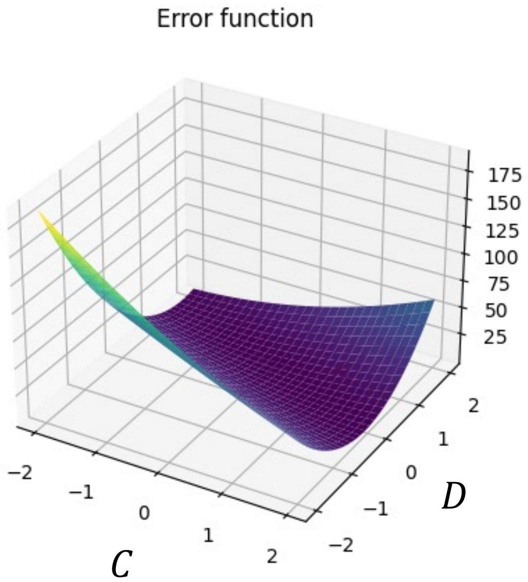
Relación entre función de error y recta

$\|\boldsymbol{e}\|^2 = (C + D - 1)^2 + (C + 2D - 2)^2$ -

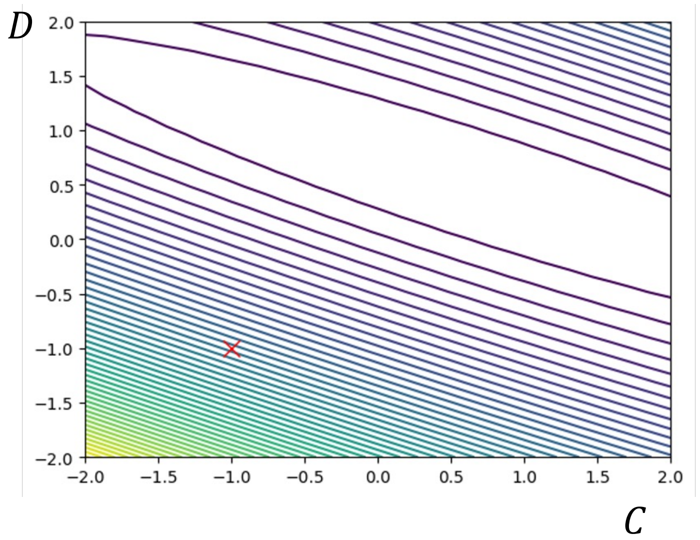
Esto es una función real de varias variable

```python
x = np.linspace(0, 3, 100)
y = -1-1*x

plt.plot(x, y)
plt.plot(1,1,marker="x", markersize=10, markeredgecolor="red", markerfacecolor="red")
plt.plot(2,2,marker="x", markersize=10, markeredgecolor="red", markerfacecolor="red")
plt.plot(3,2,marker="x", markersize=10, markeredgecolor="red", markerfacecolor="red")
plt.show()
```

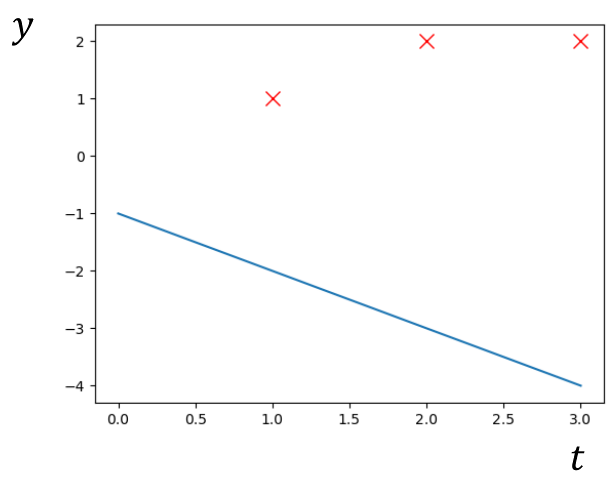
$y = C + Dt$



$f(-1,-1) = 70$



$y = -1 - t$

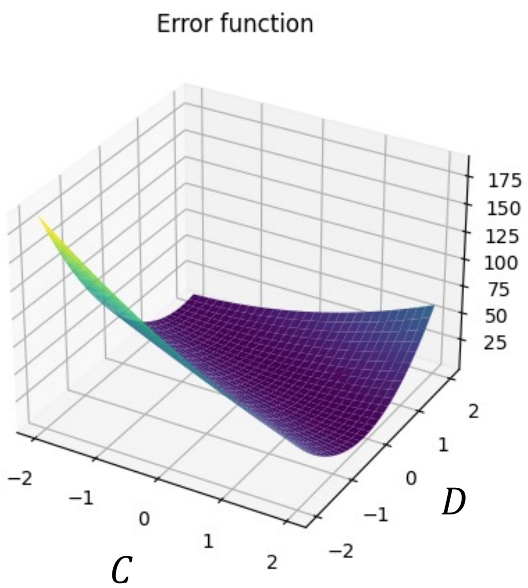


Relación entre función de error y recta

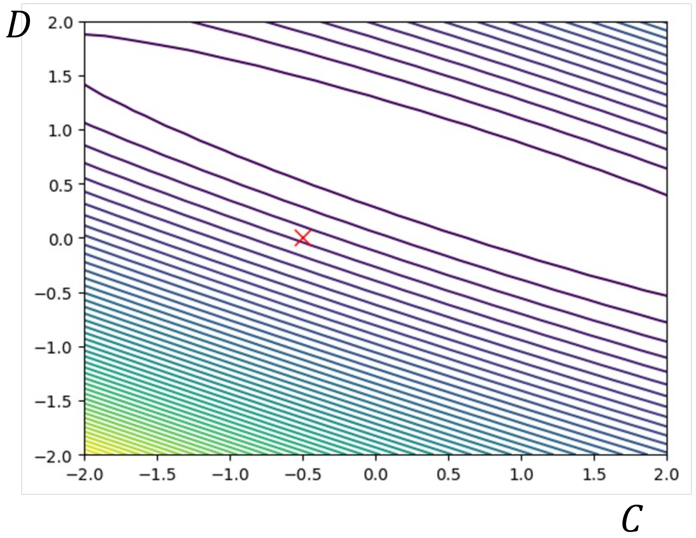
$\|\boldsymbol{e}\|^2 = (C + D - 1)^2 + (C + 2D - 2)^2 + (C + 3D - 2)^2$

Esto es una función real de varias variables $\|\boldsymbol{e}\|^2 = f(C, D)$

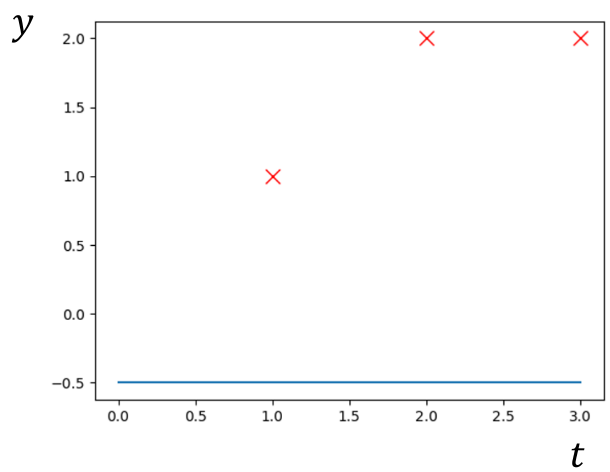
$y = C + Dt$



$f(-0.5,0) = 14.75$



$y = -0.5$

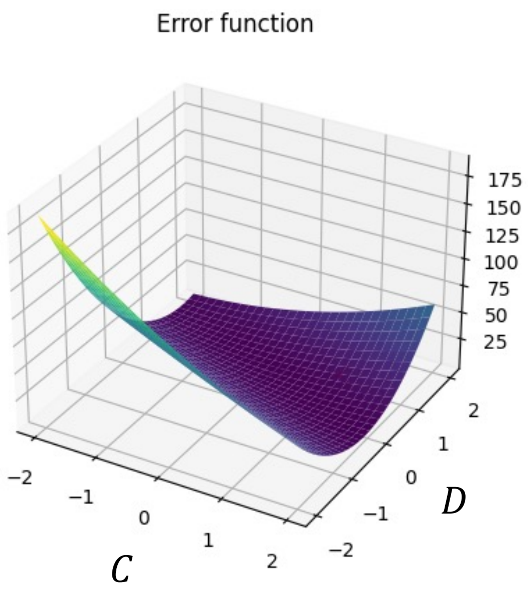


Relación entre función de error y recta

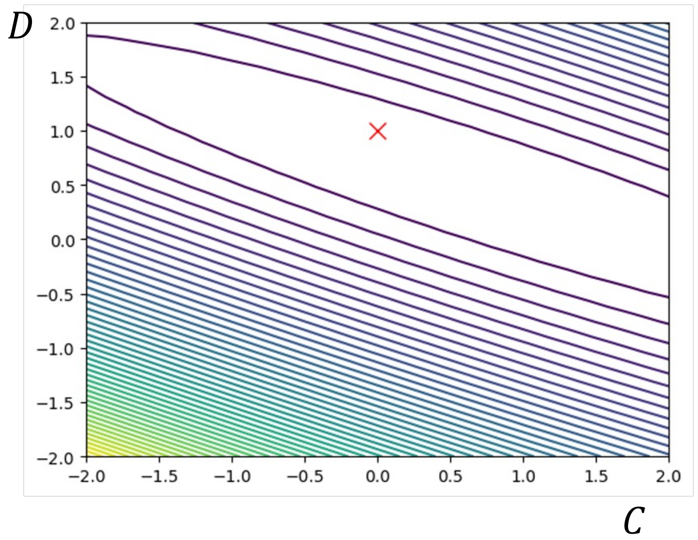
$\|\boldsymbol{e}\|^2 = (C + D - 1)^2 + (C + 2D - 2)^2 + (C + 3D - 2)^2$

$y = C + Dt$

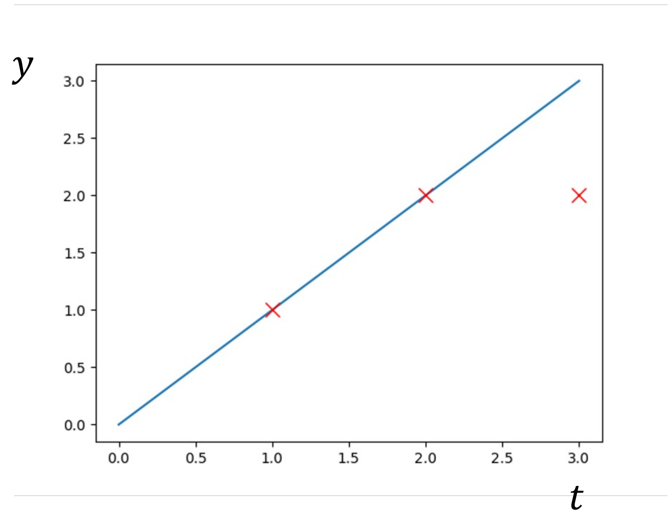
Esto es una función real de varias variables $\|\boldsymbol{e}\|^2 = f(C, D)$



$f(0,1) = 1$



$y = t$



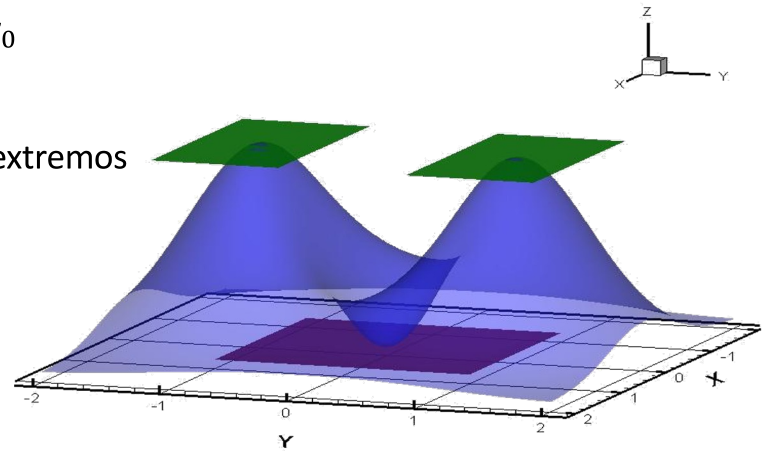
Recordatorio: cálculo de extremos

Buscamos candidatos a extremos en funciones de varias variables.

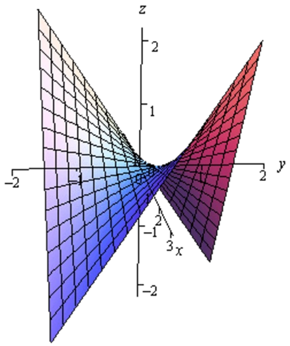
1. **Puntos críticos (o estacionarios):** Sea $f: D \subset \mathbb{R}^n \to \mathbb{R}$ y $x_0 \in \mathring{D}$ (interior de D)

   x_0 es punto crítico de f si $f: D \subset \mathbb{R}^n \to \mathbb{R}$ es diferenciable en x_0 y $\nabla f(x_0) = \mathbf{0} \Leftrightarrow df(x_0) = 0$

   En $\mathbb{R}^2$ esta condición es equivalente a que el plano tangente sea horizontal en x_0



Sin embargo la existencia de puntos críticos no garantiza la existencia de extremos



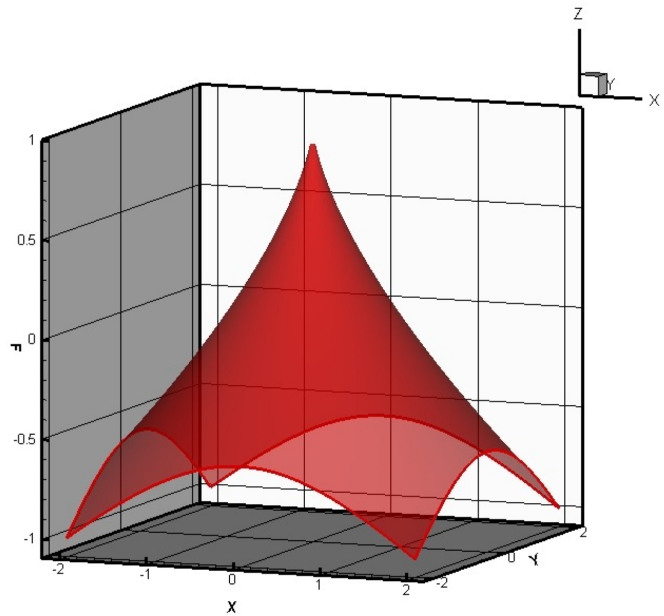
Ejemplo: la función $f(x, y) = xy$ tienen un punto crítico en $(x, y) = (0,0)$ pero no es un extremo local

Recordatorio: cálculo de extremos

Buscamos candidatos a extremos en funciones de varias variables.

2. Puntos singulares: Sea $f: D \subset \mathbb{R}^n \to \mathbb{R}$ y $x_0 \in \mathring{D}$ (interior de D)

x_0 es un punto singular de f si no es diferenciable en x_0



3. Puntos frontera del dominio.

Si D incluye al menos a parte de su frontera, puede ocurrir que existan extremos de la función en la frontera del dominio.

Condición de mínimo

$\|\boldsymbol{e}\|^2 = (C + D - 1)^2 + (C + 2D - 2)^2 + (C + 3D - 2)^2$

Esto es una función real de varias variables $f(C, D)$

Mínimos relativos cumplirán $\nabla f(C, D) = \mathbf{0}$ (Función de error $f(C, D)$ es siempre creciente)

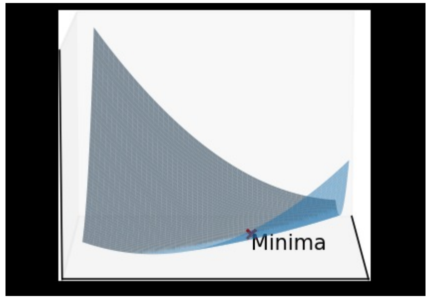
$$\frac{\partial f}{\partial C} = 0 \longrightarrow \frac{\partial f}{\partial C} = 2(C + D - 1) + 2(C + 2D - 2) + 2(C + 3D - 2) = 0 \longrightarrow 6C + 12D - 10 = 0$$

$$\frac{\partial f}{\partial D} = 0 \longrightarrow \frac{\partial f}{\partial D} = 2(C + D - 1) + 4(C + 2D - 2) + 6(C + 3D - 2) = 0 \longrightarrow 12C + 28D - 22 = 0$$

$$\begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} 5 \\ 11 \end{bmatrix}$$

¡Igual que por el camino anterior!

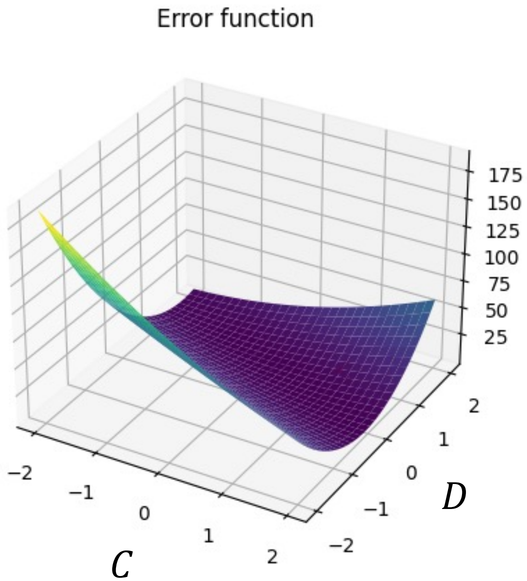
Recta de regresión. Mínimos cuadrados



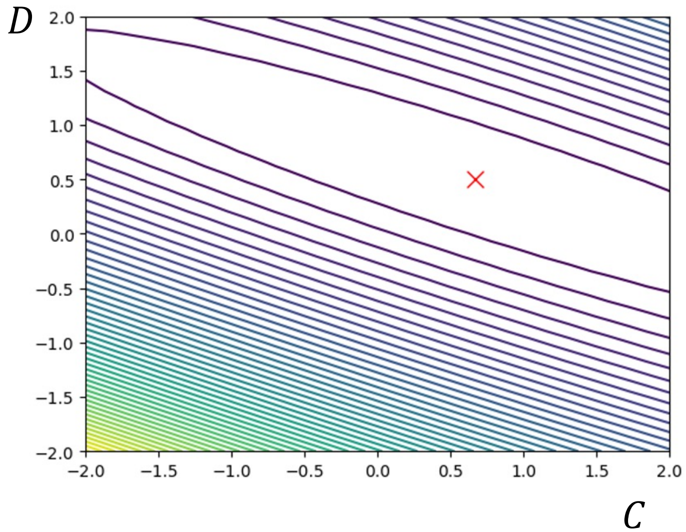
$y = C + Dt$

$$\|\boldsymbol{e}\|^2 = (C + D - 1)^2 + (C + 2D - 2)^2 + (C + 3D - 2)^2$$

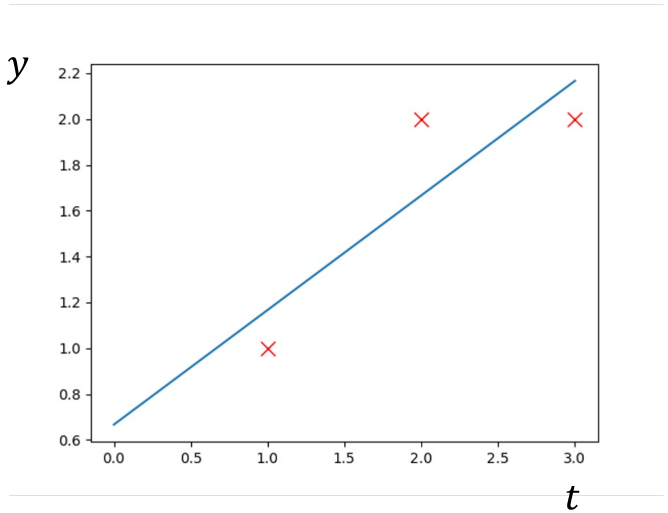
Esto es una función real de varias variables $f(C, D)$



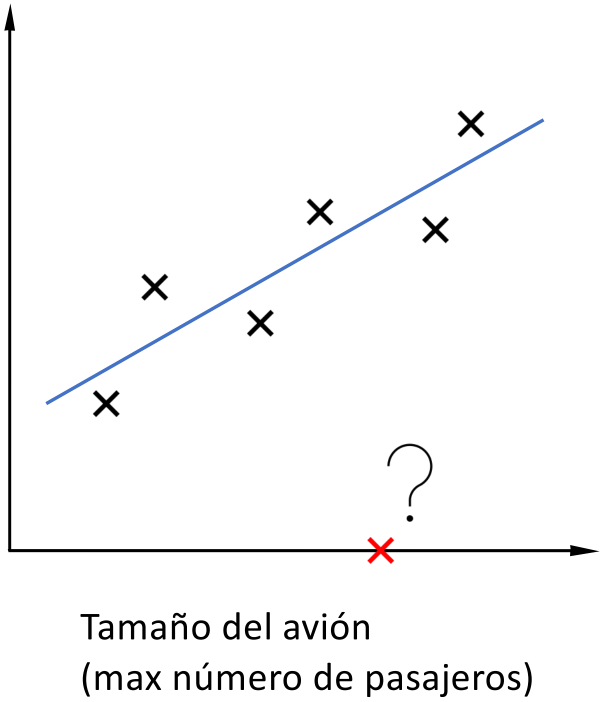
$f(0,1) = 0.167$



$y = 2/3 + 1/2t$



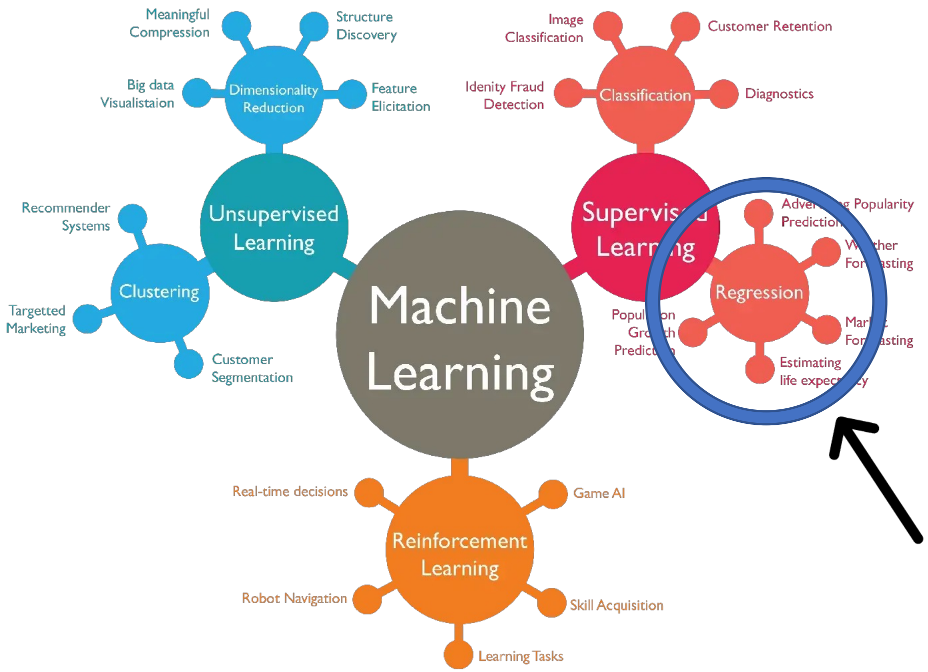
Predicción basada en datos

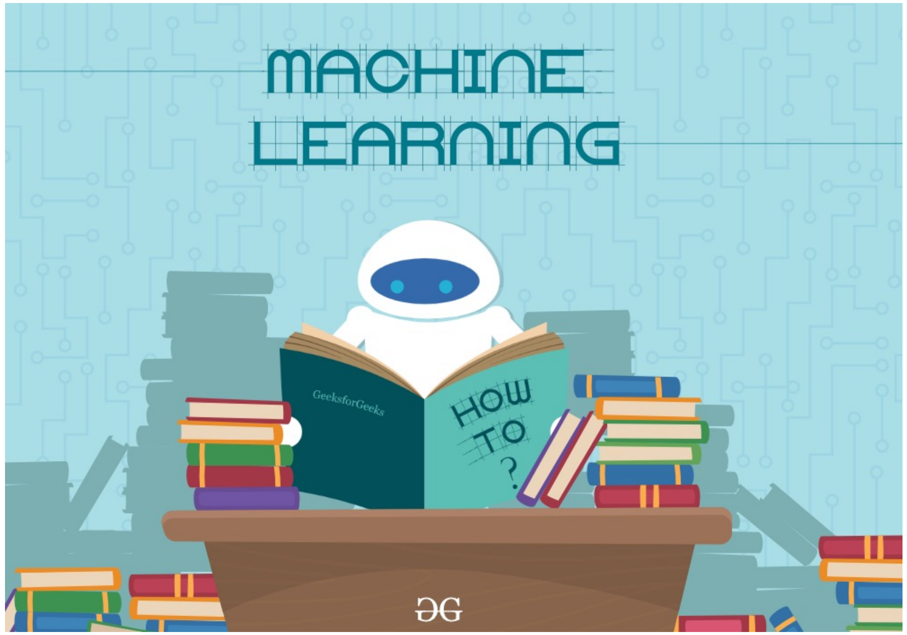


- ¿Cómo obtengo esa recta?
  - Camino 1: Álgebra lineal
  - Camino 2: Cálculo (de varias variables)
  - Camino 3: Machine learning

¿Cómo obtengo esa recta? Machine learning

Campo de estudio que permite que un ordenador "aprenda" sin necesidad de programarlo de forma explícita.





Mínimos cuadrados - Machine learning

Training set

↓

Learning algorithm

↓

$x \longrightarrow$ $\boxed{f}$ $\longrightarrow \hat{y}$

Features

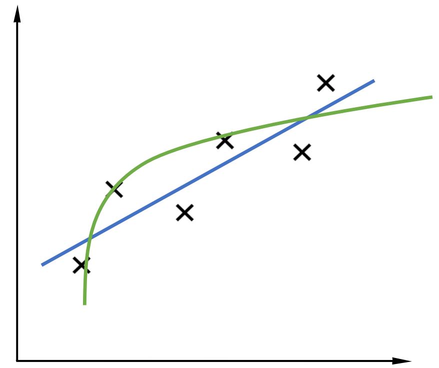
Targets

feature | model | prediction

(estimated y)

Tamaño avión $\longrightarrow f \longrightarrow$ precio (estimado)

¿Cómo representamos f?



$$f_{w,b}(x) = wx + b$$

Lineal

No lineal

Nuestro modelo es lineal. Sin embargo, se pueden utilizar modelos más complejos.

Mínimos cuadrados - Machine learning

Training set

$\downarrow$

Learning algorithm

$\downarrow$

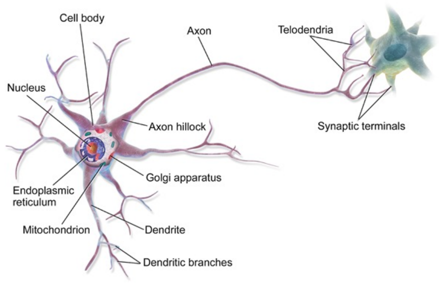
$x \longrightarrow \boxed{f} \longrightarrow \hat{y}$

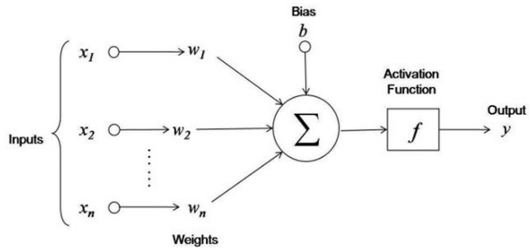
Features

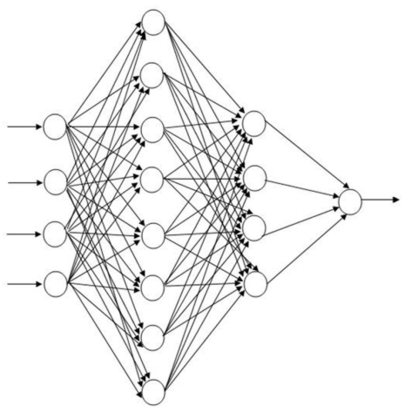
Targets

feature — model — prediction (estimated y)

Tamaño avión $\longrightarrow f \longrightarrow$ precio (estimado)





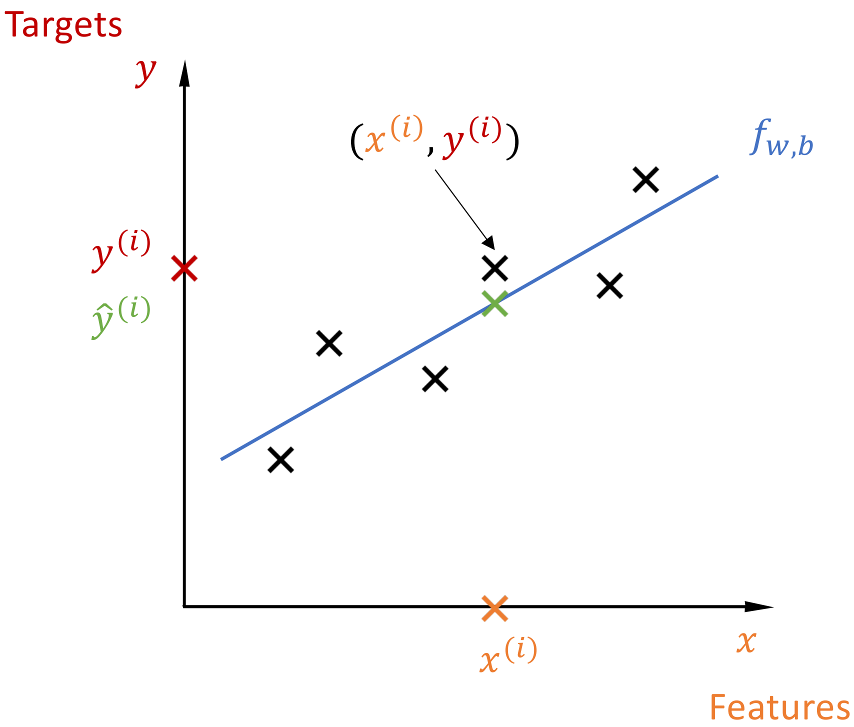


David E. Rumelhart

Geoffrey Hinton

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. ***Nature***, *323*(6088), 533-536.

Mínimos cuadrados - Función de coste



$$\hat{y}^{(i)} = f_{w,b}(x^{(i)})$$

$$f_{w,b}(x^{(i)}) = wx^{(i)} + b$$

Función de coste (Cost function): Error al cuadrado

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

m número de ejemplos en los datos de entrenamiento

Encontrar (w, b) :

$\hat{y}^{(i)}$ esté *cerca de* $y^{(i)}$ para todos los $(x^{(i)}, y^{(i)})$

Un poco de código

Training set

Features
Targets

Model

$$f_{w,b}(x) = wx + b$$

Lineal

Cost function

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

m número de ejemplos en los datos de entrenamiento

```python
# Training set
x = [1, 2, 3]
y = [1, 2, 2]

m = len(x)

# Model
def fwb(w,b,x):
    return w*x+b

# Cost function
def J(w,b,x,y):
    J = 0
    for i in range(m):
        J = J + (fwb(w,b,x[i])-y[i])**2
    J = J / (2*m)
    return J
```

¿Cómo encuentro el mínimo de $J(w, b)$?

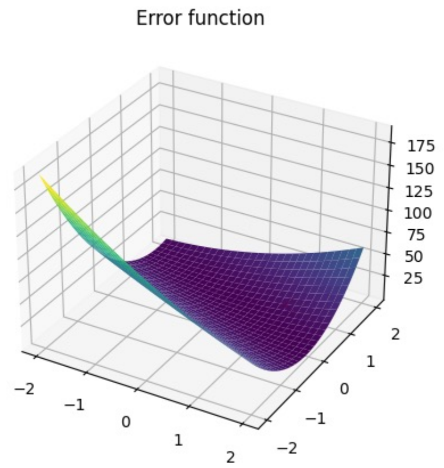
La calidad de la aproximación dependerá del valor alcanzado por la función de coste $J(w, b)$

$$\min_{w,b} J(w, b)$$

En el caso de derivación basada en cálculo, resolvimos este problema de forma analítica.

Plantear/Resolver un sistema lineal de ecuaciones puede no ser eficiente en un caso general/grande (dimensión).

En machine learning es típico utilizar el algoritmo Gradient descent (gradiente descendente).



Algoritmo Gradient descent:

Empezamos con un par de valores (w, b). Por ejemplo: $(w = 0, b = 0)$

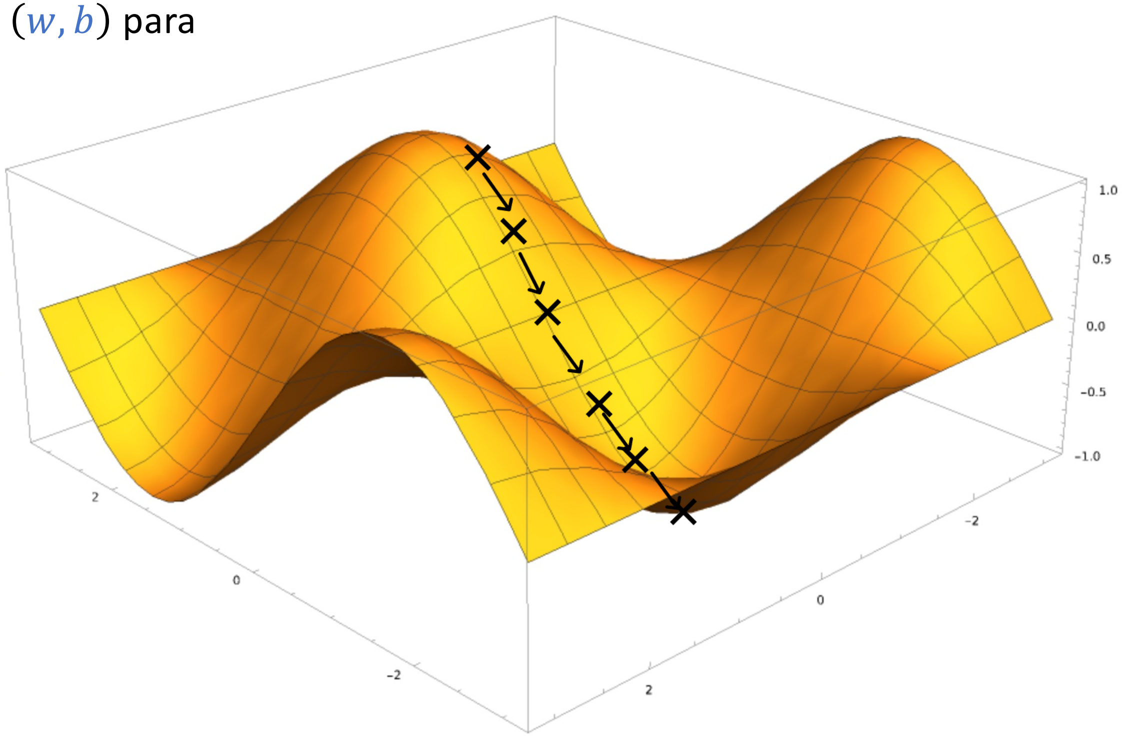
Cambiamos los valores de (w, b) para reducir $J(w, b)$

Gradient descent - intuición

Algoritmo Gradient descent:

Empezamos con un par de valores (w, b).

Cambiamos los valores de (w, b) para reducir $J(w, b)$



Gradient descent – dirección máximo crec.

Gradiente de $J(w, b)$ me da la dirección de máximo crecimiento/decrecimiento de la función

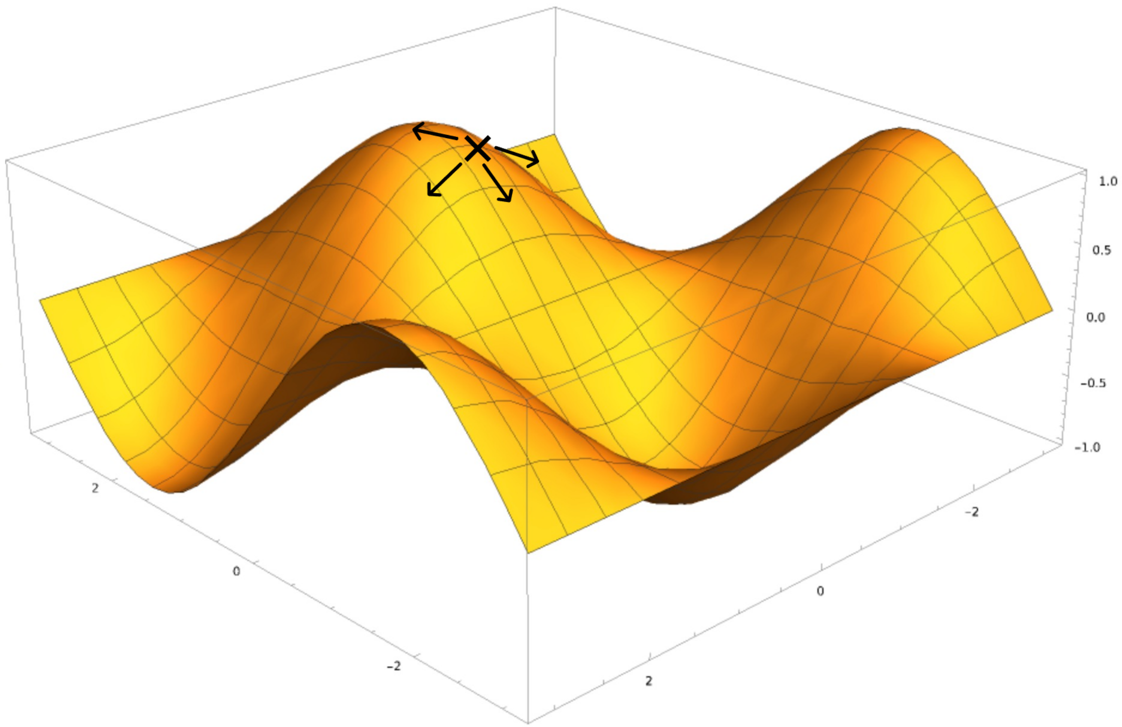
$$\nabla J(w, b) = \left(\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right)$$

Similitudes con el método de Newton (lo veremos más adelante)

Repetir hasta convergencia:

$$w = w - \alpha \frac{\partial}{\partial w} J(w, b)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w, b)$$



Gradient descent - problemas

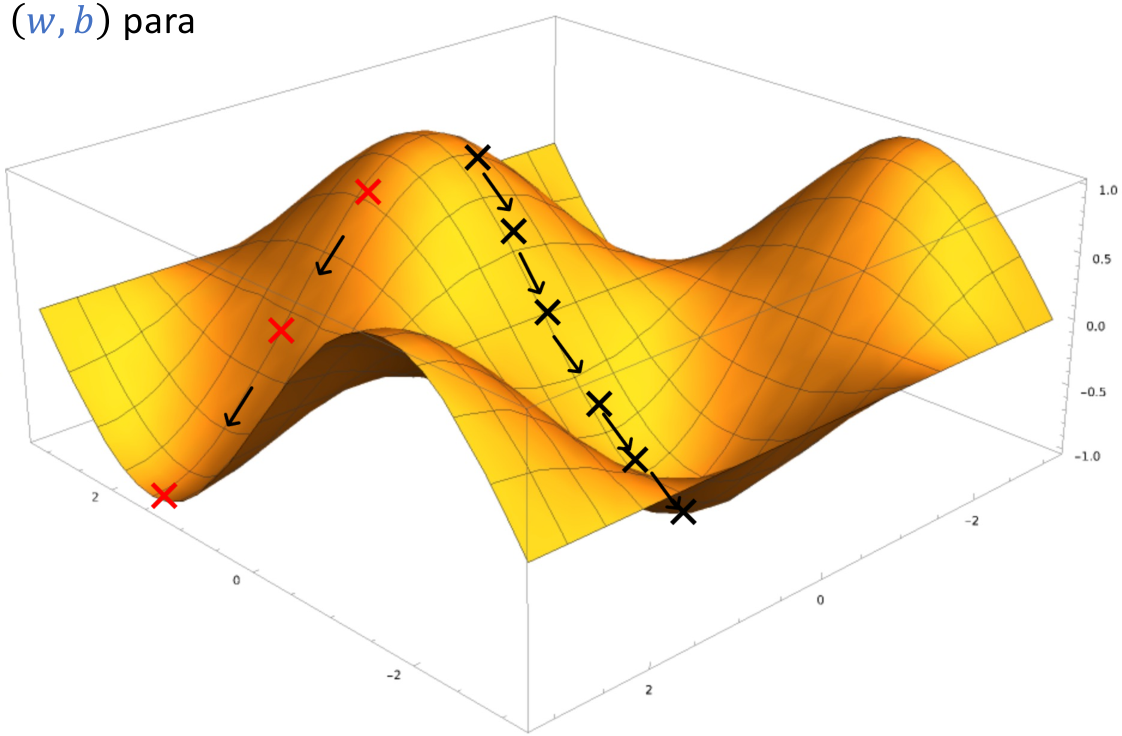
Algoritmo Gradient descent:

Empezamos con un par de valores (w, b).

Cambiamos los valores de (w, b) para reducir $J(w, b)$

Posibles problemas:

- Minimo relativo (solo uno)
- Influencia condición inicial



Gradient descent - algoritmo

$w = w_0$

$b = b_0$

Repetir hasta convergencia:

$$w = w - \alpha \frac{\partial}{\partial w} J(w, b)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w, b)$$

α learning rate
(tasa de aprendizaje)

Número positivo pequeño.
Controla el tamaño de los pasos (la distancia entre las cruces).

Convergencia: cuando w y b apenas cambian entre pasos.

Parámetros w y b se actualizan a la vez.
¡No actualizar $J(w, b)$ con nuevo valor de w al actualizar b!

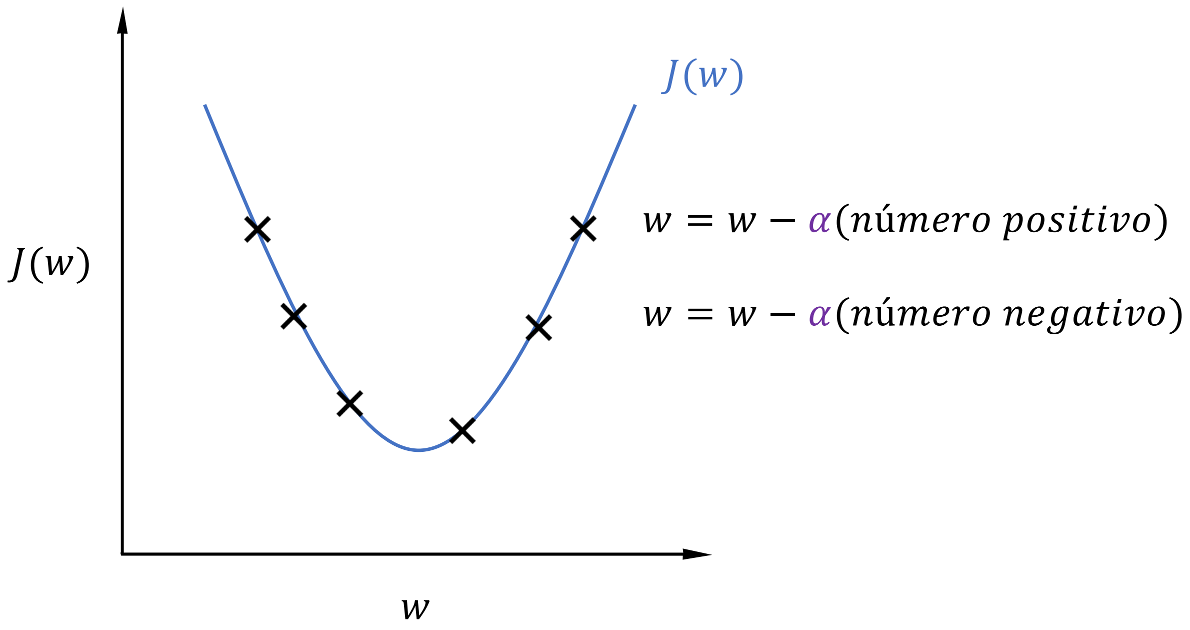
Gradient descent – una variable

$w = w_0$

Repetir hasta convergencia:

$$w = w - \alpha \frac{d}{dw} J(w)$$

α learning rate
(tasa de aprendizaje)

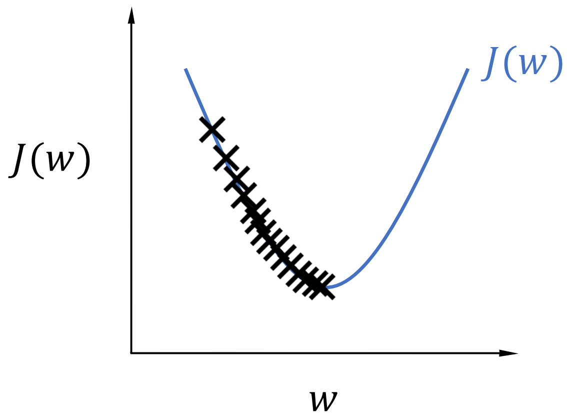


Gradient descent – learning rate

$w = w_0$

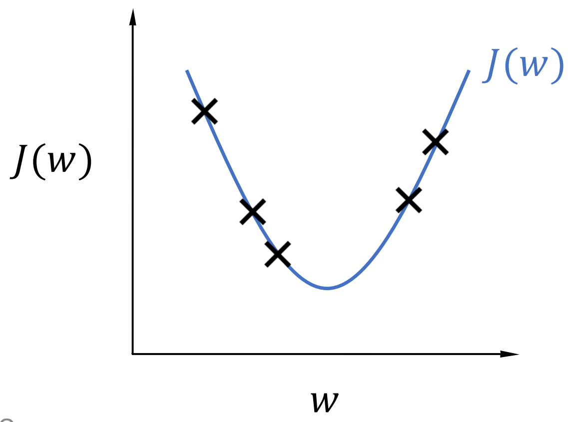
Repetir hasta convergencia:

$$w = w - \alpha \frac{d}{dw} J(w)$$



Si α es demasiado pequeño…
La convergencia será lenta

Si α es demasiado grande…
Fallo en la convergencia, divergencia



Gradient descent – mínimos cuadrados

Modelo lineal

$$f_{w,b}(x) = wx + b$$

Función de coste – error al cuadrado

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Algoritmo gradient descent

$$w = w_0$$

$$b = b_0$$

Repetir hasta convergencia:

$$w = w - \alpha\frac{\partial}{\partial w}J(w,b)$$

$$b = b - \alpha\frac{\partial}{\partial b}J(w,b)$$

$$\frac{\partial}{\partial w}J(w,b) = \frac{2}{2m}\sum_{i=1}^{m}\left(f_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)\ \cancel{x^{(i)}}\frac{\partial}{\partial w}f_{w,b}\left(x^{(i)}\right)$$

$$\frac{\partial}{\partial b}J(w,b) = \frac{2}{2m}\sum_{i=1}^{m}\left(f_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)\ \frac{\partial}{\partial b}f_{w,b}\left(x^{(i)}\right)$$

Gradient descent – mínimos cuadrados

Modelo lineal

$$f_{w,b}(x) = wx + b$$

Algoritmo gradient descent

$$w = w_0$$

$$b = b_0$$

Repetir hasta convergencia:

$$w = w - \alpha \frac{\partial}{\partial w} J(w,b)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$

```
# Gradient descent

alpha = 0.001

w = 0
b = 0

# Number of training steps
for i in range(1,100000):
    wnew = w - alpha * dJdw(w,b,x,y)
    bnew = b - alpha * dJdb(w,b,x,y)
    w = wnew
    b = bnew
```

¡Parámetros w y b se actualizan a la vez!

Gradient descent – mínimos cuadrados

Modelo lineal

$$f_{w,b}(x) = wx + b$$

```
# Model
def fwb(w,b,x):
    return w*x+b

# dJ/dw
def dJdw(w,b,x,y):
    dJdw = 0
    for i in range(m):
        dJdw = dJdw + (fwb(w,b,x[i])-y[i]) * x[i]
    dJdw = dJdw / (m)
    return dJdw

# dJ/db
def dJdb(w,b,x,y):
    dJdb = 0
    for i in range(m):
        dJdb = dJdb + (fwb(w,b,x[i])-y[i])
    dJdb = dJdb / (m)
    return dJdb
```

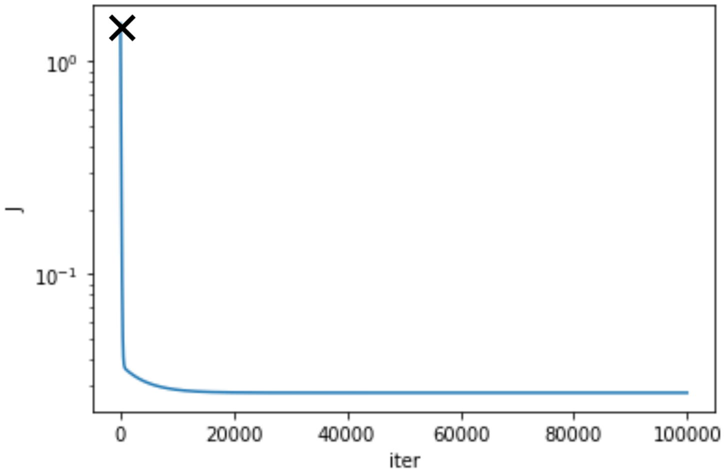
Función de coste – error al cuadrado

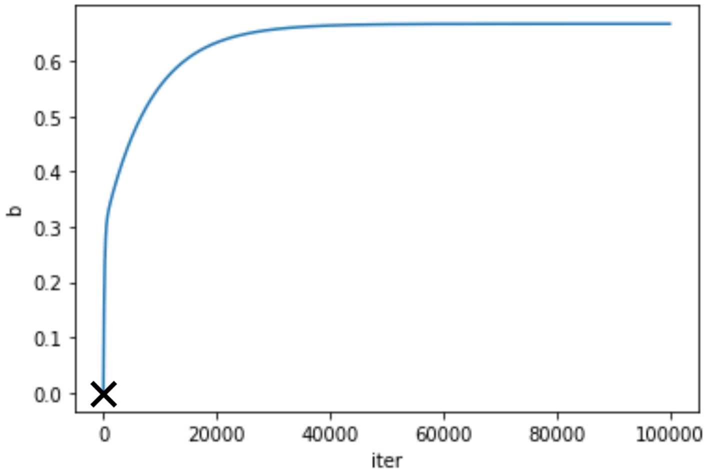
$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)^2$$

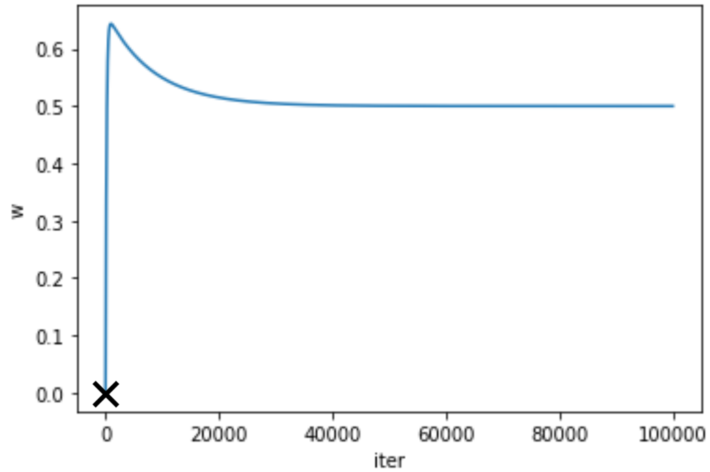
$$\frac{\partial}{\partial w}J(w,b) = \frac{2}{2m}\sum_{i=1}^{m}\left(f_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)\ x^{(i)}$$

$$\frac{\partial}{\partial b}J(w,b) = \frac{2}{2m}\sum_{i=1}^{m}\left(f_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)$$

Gradient descent – mínimos cuadrados

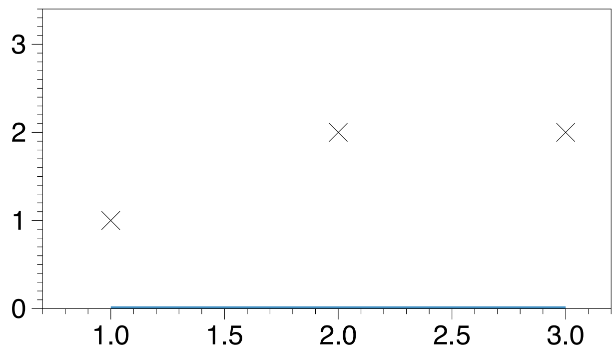




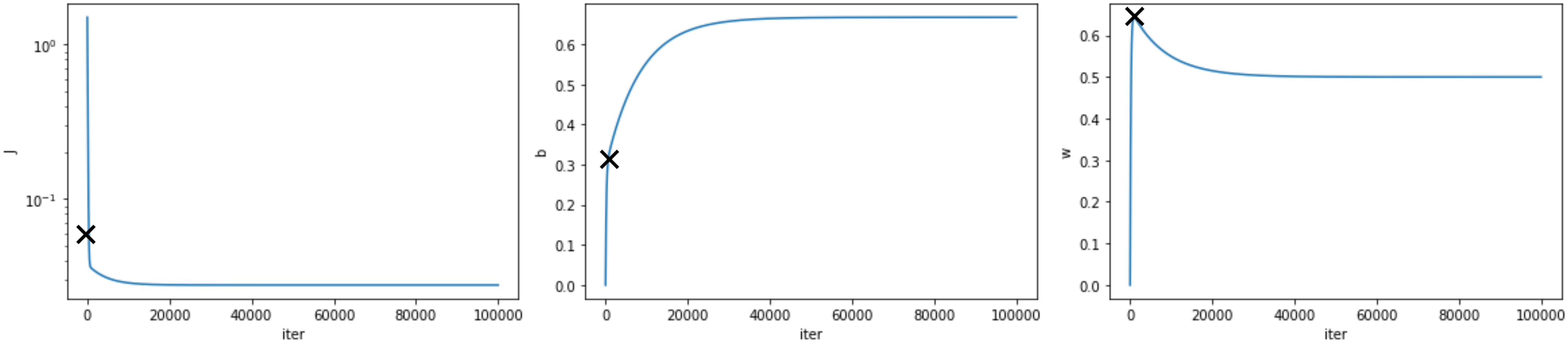


Inicio:

$$f_{w,b}(x) = wx + b = 0x + 0$$

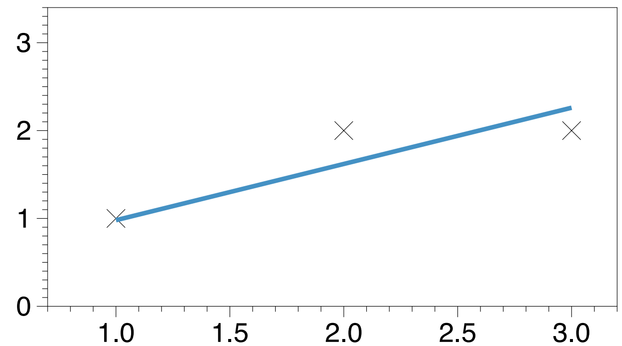


Gradient descent – mínimos cuadrados

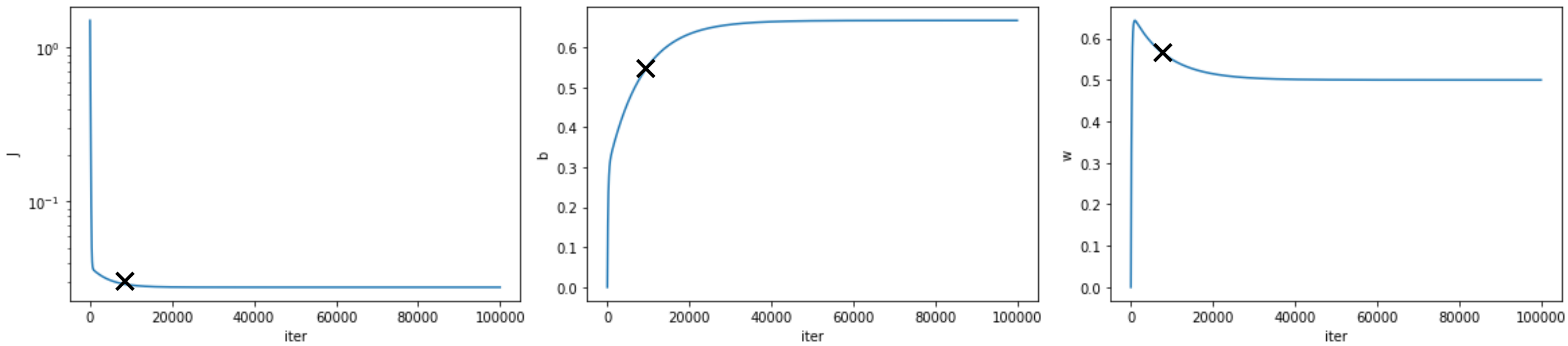


Tras 1000 iteraciones:

$$f_{w,b}(x) = wx + b = 0.64x + 0.34$$

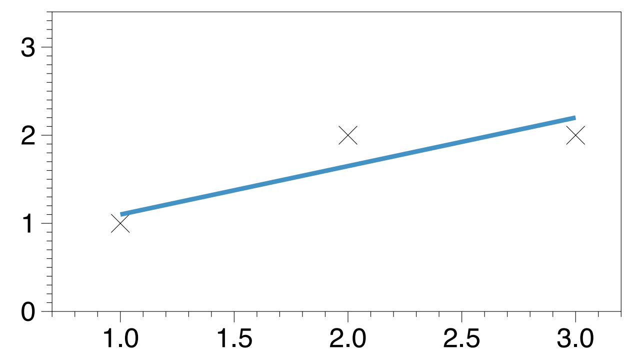


Gradient descent – mínimos cuadrados

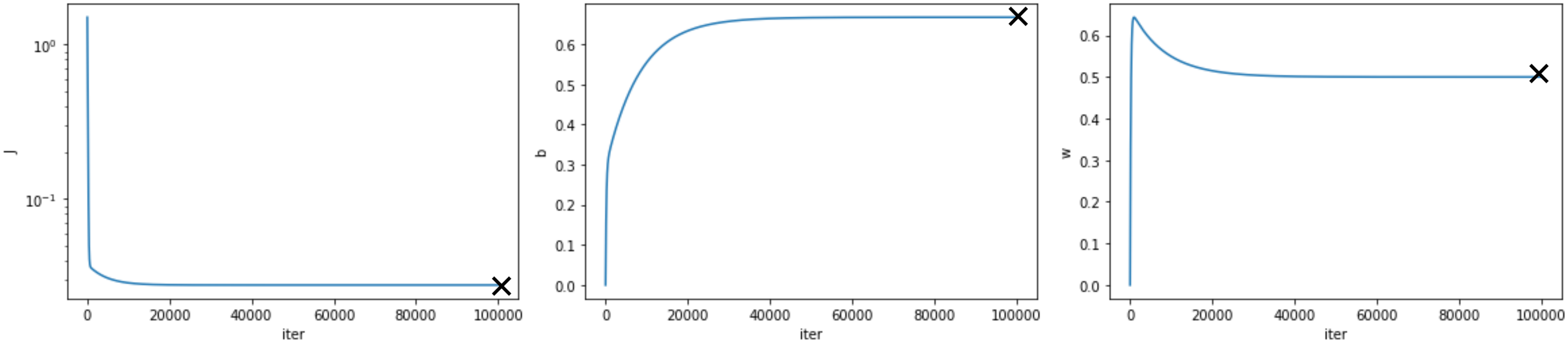


Tras 10000 iteraciones:

$$f_{w,b}(x) = wx + b = 0.55x + 0.55$$



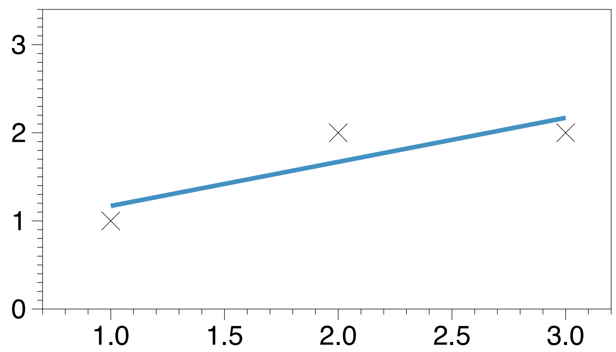
Gradient descent – mínimos cuadrados



Tras 100000 iteraciones:

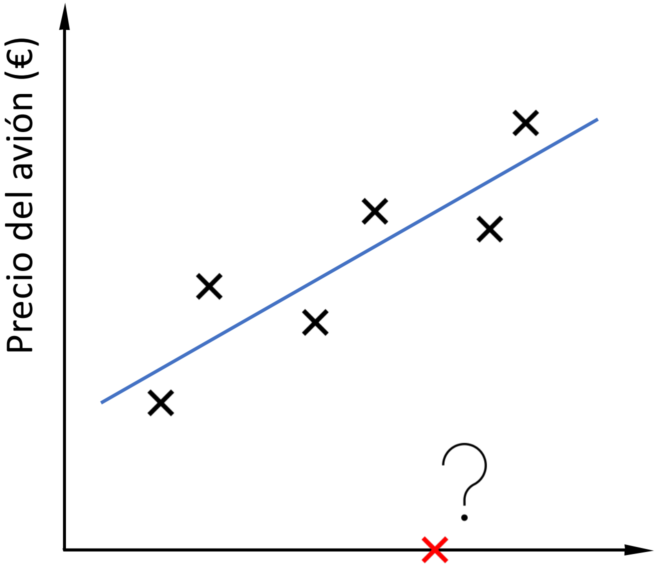
$$f_{w,b}(x) = wx + b = 0.5x + 0.67$$

¡Llego al mismo resultado!



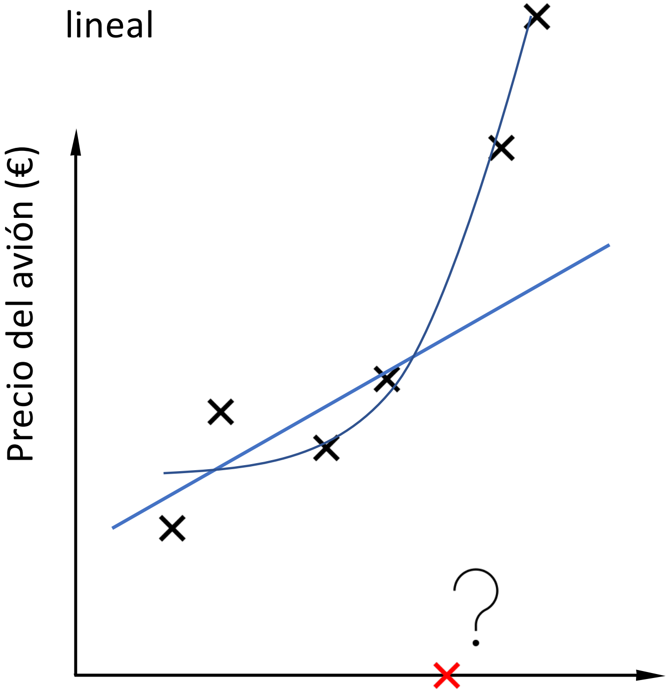
Conclusiones

La técnica de mínimos cuadrados permite realizar predicciones basadas en datos



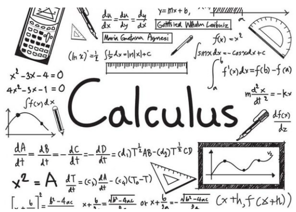
Tamaño del avión
(max número de pasajeros)

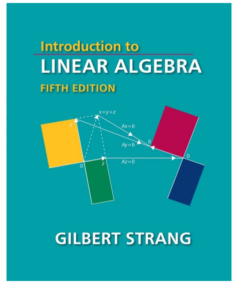
El modelo puede ser lineal o no lineal

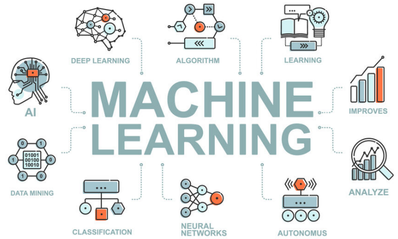


Tamaño del avión (max número de pasajeros)

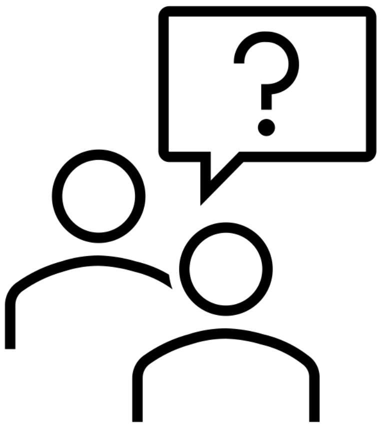
Puedo llegar al mismo resultado final por distintos caminos







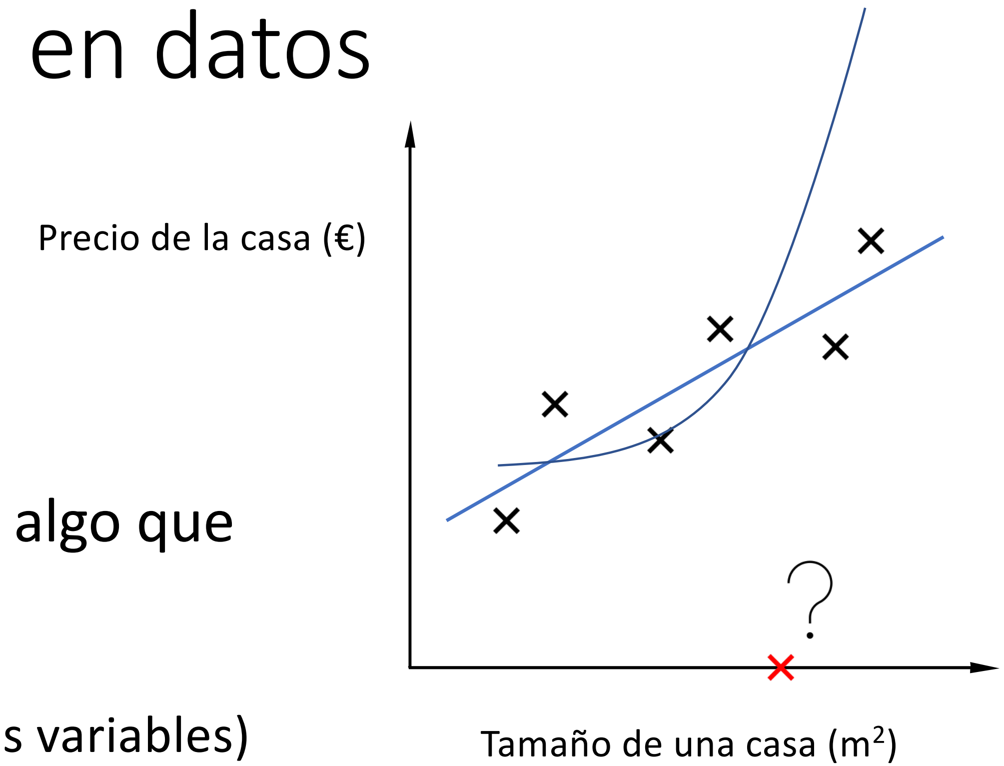
Gracias por su atención. ¿Preguntas?



Gonzalo Rubio Calzado g.rubio@upm.es

MATERIAL ADICIONAL

Predicción basada en datos



- ¿Y si quiero aproximar con algo que no sea una recta?
  - Camino 1: Álgebra lineal
  - Camino 2: Cálculo (de varias variables)

Más allá de la recta. Álgebra lineal

$(1{,}1)\ ,(2{,}2)\ ,(3{,}2)$

$$1 = C + D + E$$
$$\mathbf{2 = C + 2D} + 4E$$
$$\mathbf{2 = C + 3D} + 9E$$

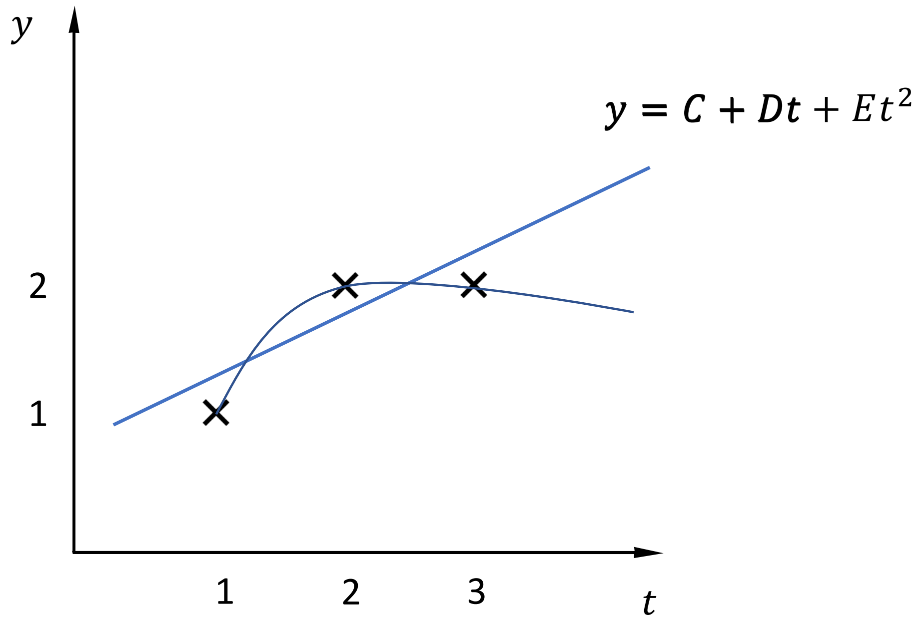
Este sistema tiene mejor pinta

$$\boldsymbol{Ax} = \boldsymbol{b}$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \begin{bmatrix} C \\ D \\ E \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

En este caso, la función puede pasar por todos los puntos (Sistema tiene solución)

$$\begin{bmatrix} C \\ D \\ E \end{bmatrix} = \begin{bmatrix} -1 \\ 5/2 \\ -1/2 \end{bmatrix}$$



Este caso particular se llama interpolación. Se verá en detalle más adelante.

Más allá de la recta. Álgebra lineal

$$(1{,}1)\ ,(2{,}2)\ ,(3{,}2)\ ,(4{,}2)$$

$$1 = C + D + E$$

$$2 = C + 2D + 4E$$

$$2 = C + 3D + 9E$$

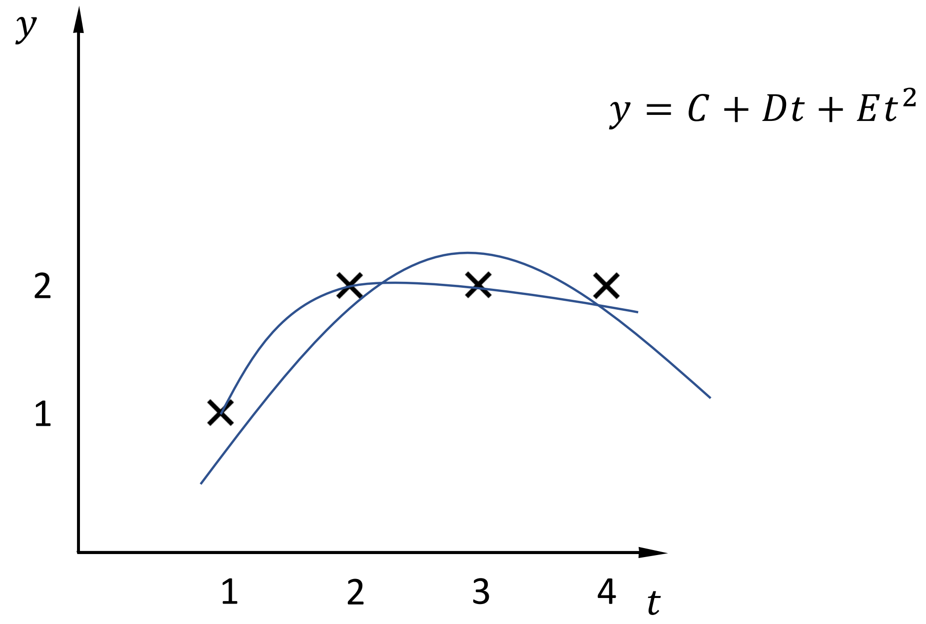
$$2 = C + 4D + 16E$$

Volvemos a tener problemas

$$\boldsymbol{Ax} = \boldsymbol{b}$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{bmatrix} \begin{bmatrix} C \\ D \\ E \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 2 \end{bmatrix}$$

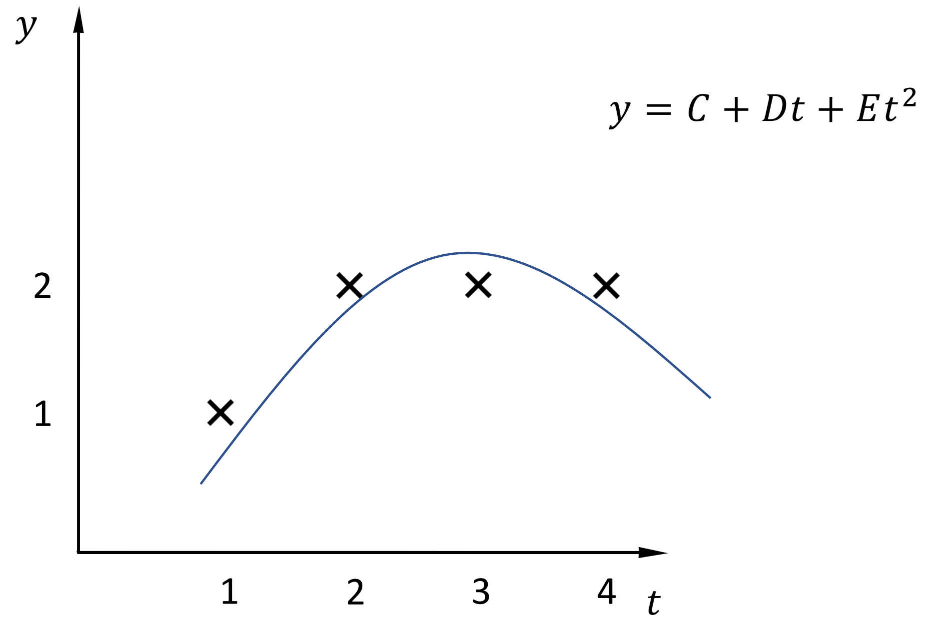
Y proponemos la misma solución $\boldsymbol{A^T A \hat{x}} = \boldsymbol{A^T b}$



Más allá de la recta. Álgebra lineal. Generalización

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{bmatrix} \begin{bmatrix} C \\ D \\ E \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ 1 & t_3 & t_3^2 \\ 1 & t_4 & t_4^2 \end{bmatrix} \begin{bmatrix} C \\ D \\ E \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

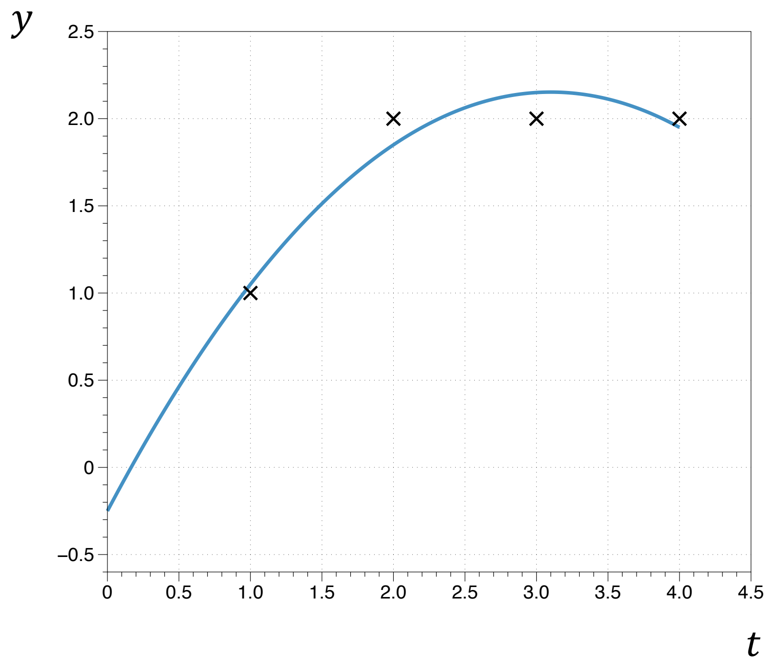


Más allá de la recta. Álgebra lineal

$$\boldsymbol{A^T A\hat{x} = A^T b}$$

```python
import numpy as np


A = np.matrix([[1, 1, 1], [1, 2, 4], [1, 3, 9], [1, 4, 16]])
b = np.matrix([1,2,2,2])
b = b.transpose()

AT = A.transpose()

Ahat = np.matmul(AT,A)
bhat = np.matmul(AT,b)

x = np.linalg.solve(Ahat, bhat)

```

$$\begin{bmatrix} C \\ D \\ E \end{bmatrix} = \begin{bmatrix} -0.25 \\ 1.55 \\ -0.25 \end{bmatrix}$$



Gradient descent – learning rate fijo

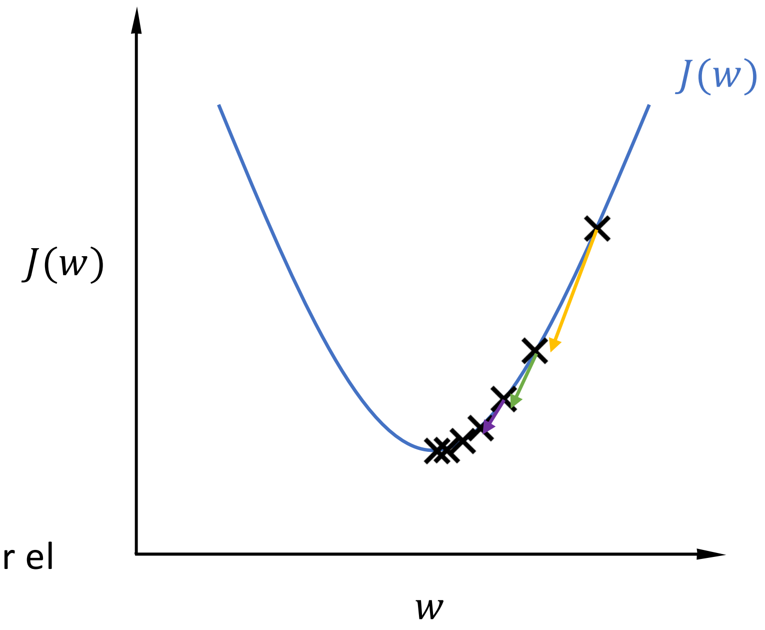
$w = w_0$

Repetir hasta convergencia:

$$w = w - \alpha \frac{d}{dw} J(w)$$

Cerca de un mínimo local:
- La derivada se hace más pequeña
- Los pasos se hacen más pequeños

Algoritmo alcanza el mínimo sin necesidad de variar el learning rate α



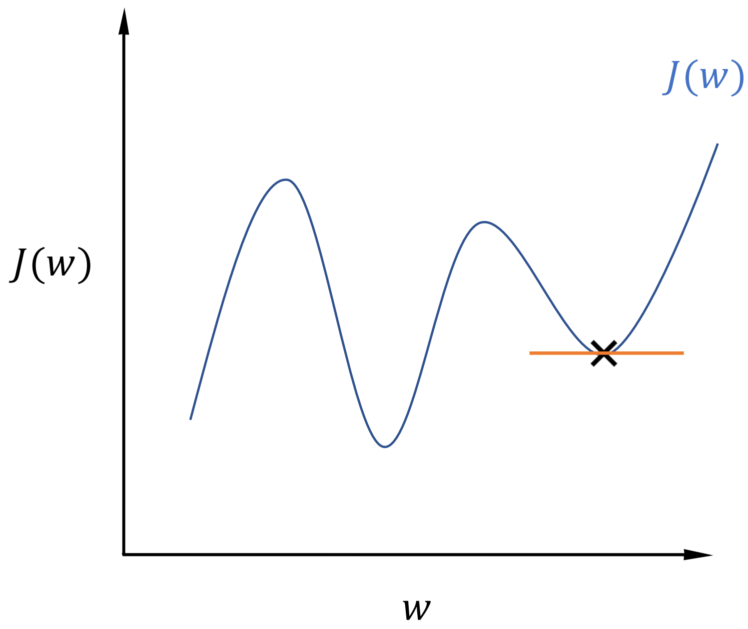
Gradient descent – learning rate

$w = w_0$

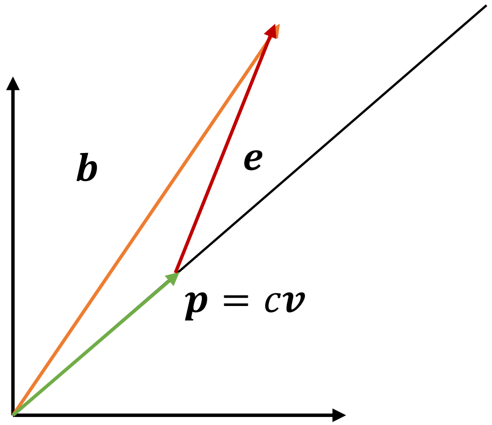
Repetir hasta convergencia:

$$w = w - \alpha \frac{d}{dw} J(w)$$

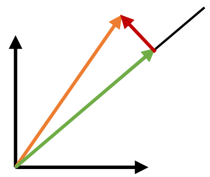
Si ya estoy en un mínimo local, el algoritmo deja de evolucionar



¿EQUIVALENCIA ALGEBRA Y CÁLCULO?



El objetivo en ambos casos es minimizar la distancia entre b y p:



Algebra:
La distancia será minima si e es ortogonal a p

$$(\boldsymbol{b} - \boldsymbol{p}) \cdot \boldsymbol{p} = 0$$

$$(\boldsymbol{b} - c\boldsymbol{v}) \cdot c\boldsymbol{v} = 0$$

$$c = \frac{\boldsymbol{b} \cdot \boldsymbol{v}}{\boldsymbol{v} \cdot \boldsymbol{v}}$$

Cálculo:
Minimizo la función distancia $\|\boldsymbol{b} - \boldsymbol{p}\|^2 = (\boldsymbol{b} - c\boldsymbol{v}) \cdot (\boldsymbol{b} - c\boldsymbol{v})$

$$\boldsymbol{b} \cdot \boldsymbol{b} + c^2\boldsymbol{v} \cdot \boldsymbol{v} - \boldsymbol{2}c\boldsymbol{b} \cdot \boldsymbol{v}$$

$$\frac{d\|\boldsymbol{b} - \boldsymbol{p}\|^2}{dc} = \boldsymbol{2}c\boldsymbol{v} \cdot \boldsymbol{v} - \boldsymbol{2b} \cdot \boldsymbol{v} = \boldsymbol{0} \qquad c = \frac{\boldsymbol{b} \cdot \boldsymbol{v}}{\boldsymbol{v} \cdot \boldsymbol{v}}$$