

Aplicando el gradiente y utilizando las fórmulas:

$$\nabla_{\theta} \left(\cancel{y^T y} - y^T X \theta - \theta^T \cancel{X^T y} + \theta^T X^T X \theta \right) = 0$$

Entonces:

$$- X^T y - X^T y + 2 X^T X \theta = 0$$

Luego:

$$\cancel{2} X^T X \theta = \cancel{2} X^T y$$

$$(X^T X) \theta = X^T y \quad | (X^T X)^{-1}$$

$$\theta = (X^T X)^{-1} X^T y \quad \square$$

Clase 3 : Antes de comenzar... resumen clase anterior!

La clase anterior hallamos la solución al problema de minimización :

$$\underset{\theta \in \mathbb{R}^d}{\operatorname{argmin}} \quad \frac{1}{2m} \sum_{i=1}^m (\hat{y}_i - y_i)^2$$

Que en su forma matricial es equivalente a :

$$\min_{\theta} \| X\theta - y \|^2_2 = \min_{\theta} \| \varepsilon \|^2_2$$

Donde X es la matriz asociada al "input"
o medición, θ el vector de parámetros
que queremos minimizar e y el vector de
"outputs" o "targets" que supervisa el
entrenamiento.

La solución al problema vía álgebra lineal + cálculo es:

$$\Theta = (X^T X)^{-1} X^T y \quad (\text{Ecuaciones Normales})$$

En nuestro caso tendremos más mediciones que parámetros es decir, X es una matriz rectangular.

Por ejemplo:

x_i	y_i
1	2
2	$1/2$
3	-1
4	3
5	1

$\underbrace{\hspace{10em}}$ Datos

Plantearmos
el
problema

$$\begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_0 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \end{bmatrix} = \begin{bmatrix} 2 \\ 1/2 \\ -1 \\ 3 \\ 1 \end{bmatrix}$$

Matriz
rectangular

Queremos
minimizar
el tamaño
del error!

$$f_{\theta}(x) = \theta_1 x + \theta_0$$

$\underbrace{\hspace{10em}}$ Modelo

Podemos inferir directamente, que dado
 $g_0(x) = \Theta_1 x + \Theta_0$ (Una recta), será
imposible "clarar" el error, pero queremos
encontrar la mejor combinación de (Θ_1, Θ_0)
para que sea:

$$\left\| \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \end{bmatrix} \right\|_2^2 = \sqrt{\varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 + \varepsilon_4^2 + \varepsilon_5^2} = \sqrt{\sum_{i=1}^5 \varepsilon_i^2}$$


Esto sea lo más pequeño posible!

Inicio clase 03

- Sin embargo, en la vida real la inversión de matrices es costosa computacionalmente.

Por ejemplo, calculemos el costo del producto interno, contando el número de operaciones :

- En 2D :

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = x_1 u_1 + x_2 u_2$$

$\rightarrow 2 \text{ multiplicaciones}$ } 3 op.
 $\rightarrow 1 \text{ Suma}$

- En 3D :

$$\begin{bmatrix} x_1, x_2, x_3 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = x_1 u_1 + x_2 u_2 + x_3 u_3$$

$\rightarrow 3 \text{ multiplicaciones}$
 $\rightarrow 2 \text{ Sumas}$

- En N-dimensiones :

$$\begin{bmatrix} x_1, x_2, \dots, x_N \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} = x_1 u_1 + x_2 u_2 + \dots + x_N u_N$$

$\rightarrow N \text{ multiplicaciones}$
 $\rightarrow N-1 \text{ sumas}$

Total : $2N - 1$ operaciones

Generalmente, se piensa el número de operaciones para $N \rightarrow \infty$, en tal caso) Si el "orden" o costo del producto interno es $2N-1$
Sólo el primer término es el que sera más y suele reducirse la expresión a:

$$\lim_{N \rightarrow \infty} 2N - 1 \equiv O(N) \text{ "orden } N"$$

Es decir, el costo dependerá del tamaño de los vectores y el tiempo que tome cada operación.

- La inversión de matrices tiene orden:
 $O(m^4 \log^2(m))$ o incluso $O(m^3)$

Info Extra: Una manera de invertir matrices podría ser vía factorización QR. Donde Q es una matriz con columnas orthonormales es decir

$Q^T = Q^{-1}$, y R es una matriz triangular superior. Esta factorización es dada por:

$$A = Q R$$

$m \times m \quad m \times m \quad m \times m$

y tiene un costo de $\mathcal{O}(m^3)$. Luego, supongamos estamos ante el siguiente problema:

$$\underbrace{(X^T X)}_A \theta = X^T y$$

Debemos invertir A , entonces hacemos factorización QR :

$$QR\theta = X^T y \quad (\text{Aqui usamos } \mathcal{O}(m^3) \text{ operaciones})$$

Luego, multiplicaremos por θ^T por la izquierda:

$$R\theta = Q^T X^T y$$

(Aqui usamos $3 \cdot 2$
 $(2N-1) \cdot N \cdot N = 2N - N$
operaciones, es decir,
 N^3)

finalmente, como R es triangular superior :

$$\begin{bmatrix} r_{11} & 0 & 0 & \cdots & 0 \\ r_{21} & r_{22} & 0 & \cdots & 0 \\ r_{31} & r_{32} & r_{33} & 0 & \cdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mm} & \end{bmatrix} \begin{bmatrix} \theta_m \\ \theta_{m-1} \\ \vdots \\ \theta_0 \end{bmatrix} = \begin{bmatrix} \alpha_m \\ \alpha_{m-1} \\ \vdots \\ \alpha_0 \end{bmatrix}$$

Número de operaciones

$$\theta_m = \alpha_m / r_{11} \rightarrow 1 \text{ div}$$

$$\theta_{m-1} = \frac{\alpha_{m-1} - r_{21} \cdot \theta_m}{r_{22}} \rightarrow \begin{array}{l} 1 \text{ mult} \\ 1 +/- \\ 1 \text{ div} \end{array}$$

$$\theta_{m-2} = \frac{\alpha_{m-2} - r_{31} \theta_m - r_{32} \theta_{m-1}}{r_{33}} \rightarrow \begin{array}{l} 2 \text{ mult} \\ 2 +/- \\ 1 \text{ div} \end{array}$$

\vdots

O Secu : $1 + 2 + \cdots + N$ (Multiplicación)

$1 + 2 + \cdots + N$ (Sumar/restar)
 N (divisiones)

$$\text{Entonces} : 1 + 2 + \cdots + N = \frac{N(N-1)}{2}$$

de este orden:

$$\therefore N^2 - N + N = N^2 \quad (=) \quad O(n^2)$$

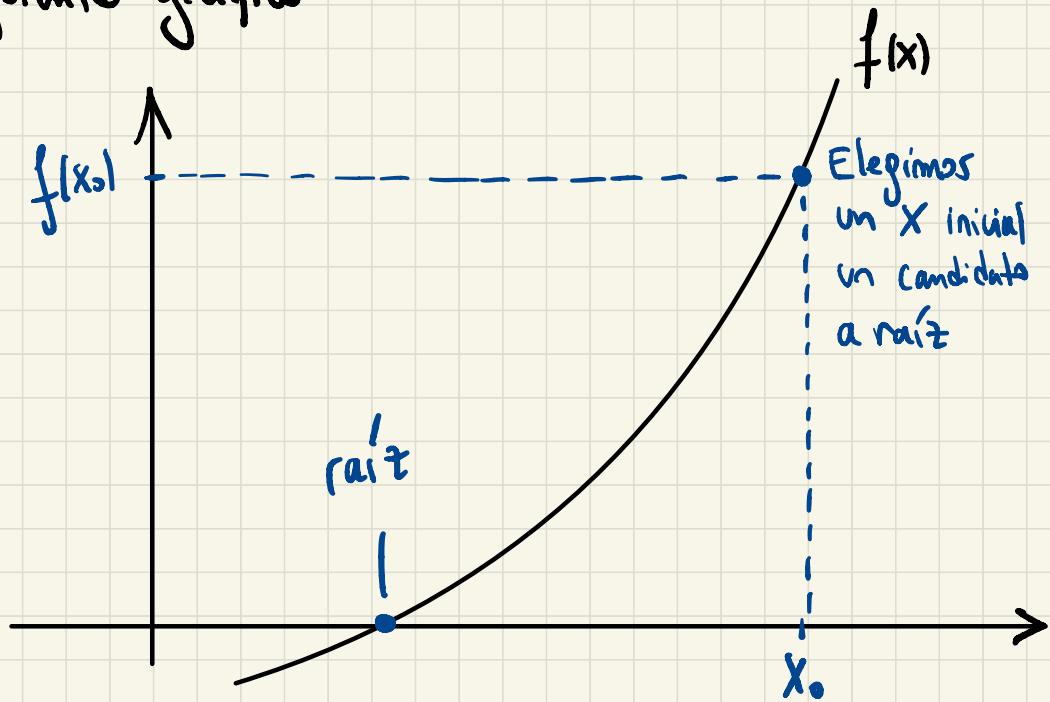
↓

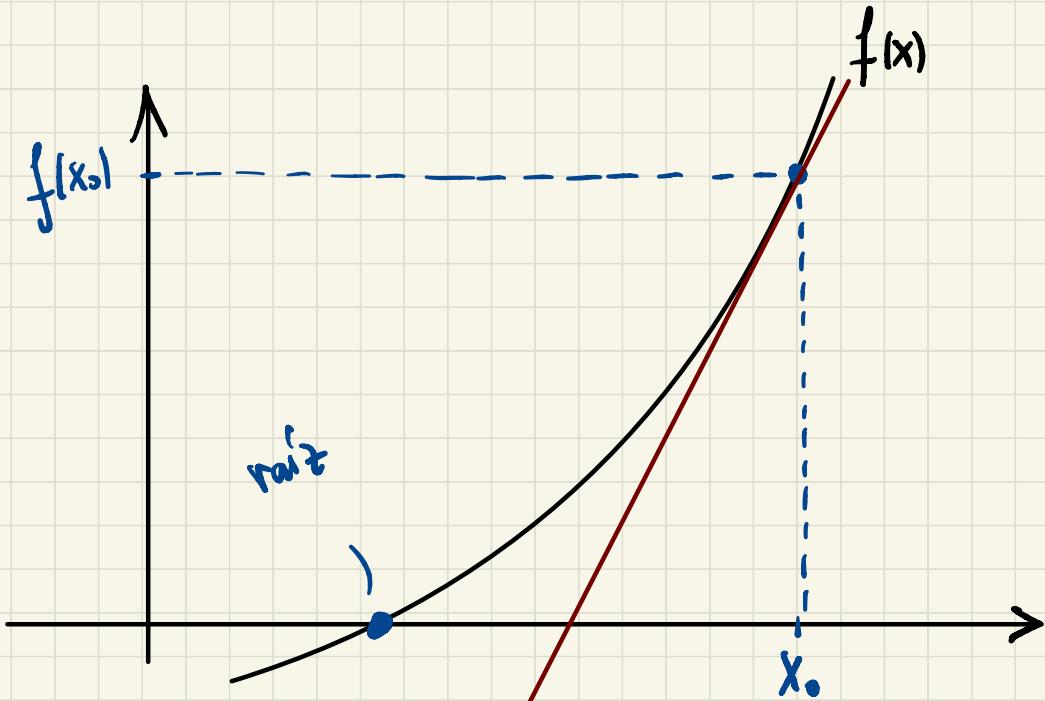
∴ Invertir los ceros' : $O(m^3) + O(m^2) \equiv O(m^3)$

- Pues, volviendo al inicio, invertir es costoso.
Necesitaremos entonces un método iterativo,
que nos permita poco a poco informar al
modelo sobre los datos.

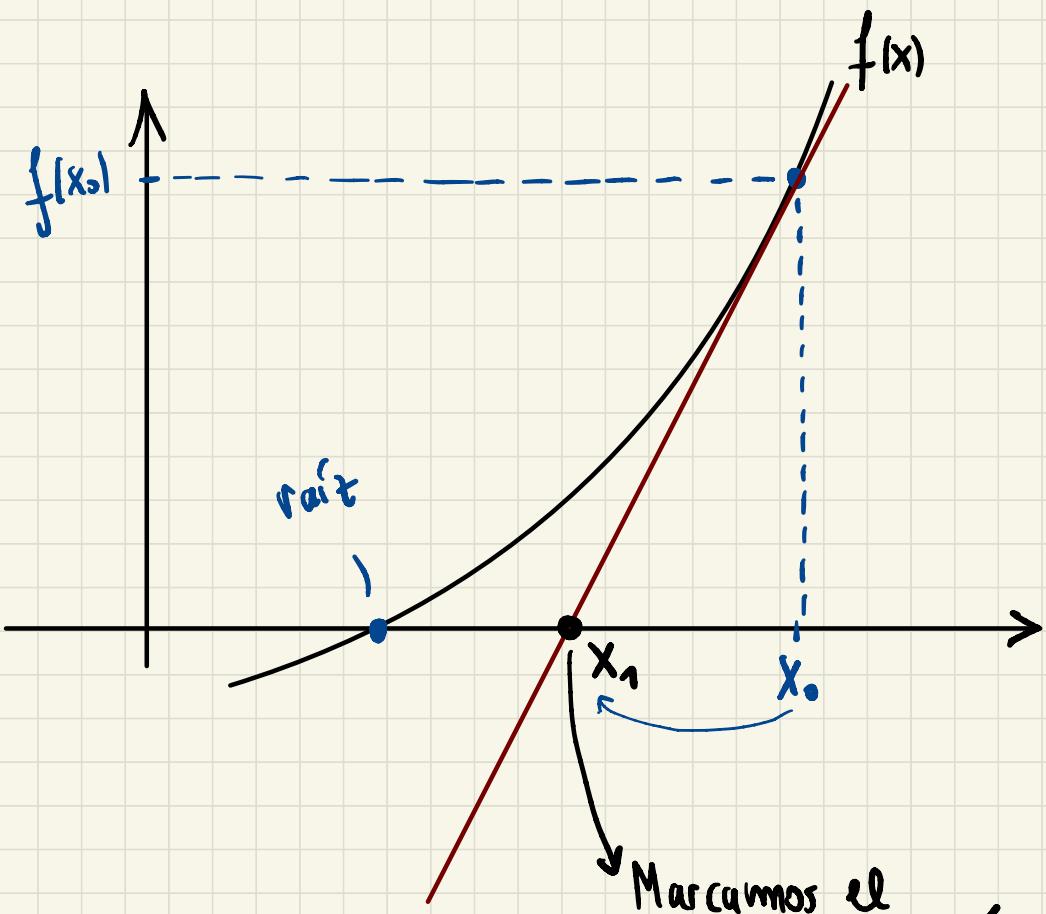
- Ejemplo : Método de Newton Raphson

Un famoso método iterativo es el método de Newton para encontrar raíces de funciones o los "ceros" de una función. Miraremos el siguiente gráfico :



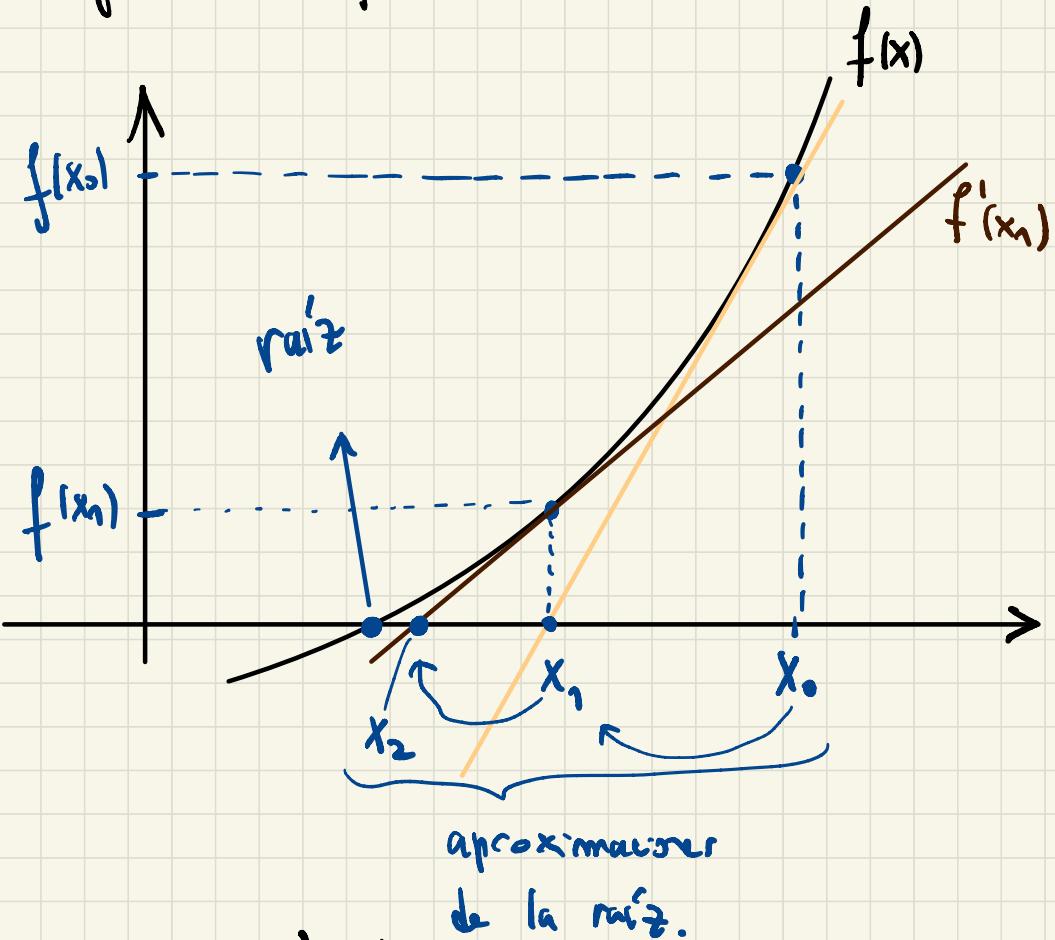


Calculamos su derivada
 $f'(x_0)$ y la recta con
pendiente $m = f'(x_0)$
que pasa por $(x_0, f'(x_0))$
la dibujamos!



Marcamos el punto de intersección con el eje x , este será nuestro nuevo candidato a raíz de $f(x)$, lo llamaremos x_1

Repetimos los pasos anteriores :



Fin del ejemplo!. La fórmula de este
método es dada por :

"Nueva aproximación"

"aproximación anterior"

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

En conclusión, vernos de un valor anterior, nos permite calcular un valor nuevo!

- Este mismo horizonte utilizaremos para hallar f.

Gradiente Descendiente

De forma general queremos:

- i) Comenzar con valores de Θ aleatorios, por ejemplo $\Theta_0 = 0$, $\Theta_1 = 0$
- ii) Modificar esos valores dados nuestros datos (X_i, Y_i) .
- iii) Detenernos hasta hallar un mínimo o algo cercano.

El algoritmo es dado por:

$$\Theta_{k+1} = \Theta_k - \alpha \frac{\partial f(\Theta_k, (X_i, Y_i))}{\partial \Theta_k}$$

- Donde el símbolo " $=$ " es una asignación, es decir, reemplazamos Θ_{k+1} por la expresión de la derecha.
- Θ_{k+1} : "Nuevo valor de Θ "
- Θ_k : Valor antiguo de Θ
- α : Tasa de aprendizaje, es un hiperparámetro del algoritmo.

Por ejemplo, para el caso de Θ_0 , Θ_1 , en nuestro problema donde $g_\Theta(x) = \Theta_1 x + \Theta_0$, el algoritmo sería:

$$\Theta_1^{k+1} \xrightarrow{\text{Como supríndice para no confundir con el "1" que es índice de la variable.}} \Theta = [\Theta_0^k, \Theta_1^k]$$

$$\Theta_1^{k+1} = \Theta_1^k - \alpha \frac{\partial \mathcal{L}(\Theta, (x, y))}{\partial \Theta_1}$$

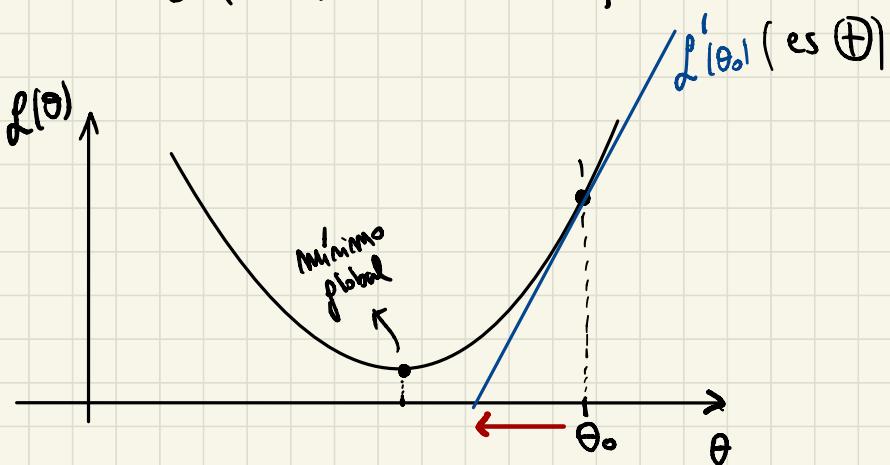
$$\Theta_0^{k+1} = \Theta_0^k - \alpha \frac{\partial \mathcal{L}(\Theta, (x, y))}{\partial \Theta_0}$$

Observaciones :

- ★ Los θ_i deben cambiar simultáneamente
- ★ $\alpha \in \mathbb{R}^+$ generalmente $0 < \alpha < 1$

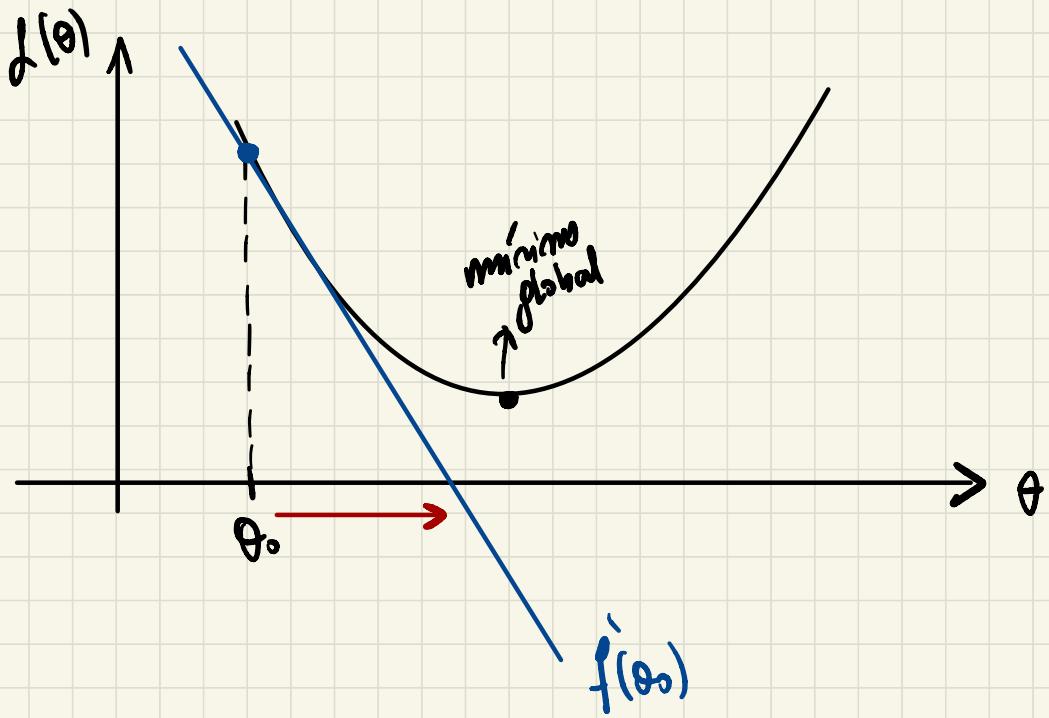
Intuición de Convergencia :

Como sabemos $L(\theta)$ es una función convexa y podemos representarla en su forma más (2D) como una parábola :



- Si θ inicial es $\theta = \theta_0$, la dirección de máximo crecimiento es hacia la derecha.
Si caminamos en la dirección contraria ($-\alpha \frac{\partial L}{\partial \theta}$) nos aproximaremos al mínimo.

- En el caso contrario :



- La pendiente es negativa.

Si caminamos en la dirección

$$\begin{array}{c} -\alpha \\ \downarrow \\ \Theta \end{array} \quad \frac{\partial L}{\partial \Theta} \quad \downarrow \quad \begin{array}{c} + \\ \Theta \end{array} \equiv +$$

Nuevamente, no estaremos

aproximando al valor mínimo.

Observación : Es posible que encuentren esta información en forma matricial como :

$$\Theta^{k+1} = \Theta^k - \alpha \nabla_{\Theta} f(\Theta)$$

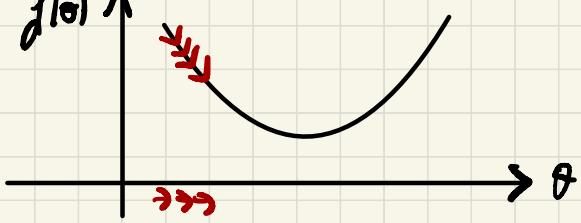
donde $\nabla_{\theta} = \begin{bmatrix} \frac{\partial}{\partial \theta_0} \\ \frac{\partial}{\partial \theta_1} \\ \vdots \\ \vdots \\ \frac{\partial}{\partial \theta_m} \end{bmatrix}$ es el vector gradiente

O sea, para nuestro ejemplo :

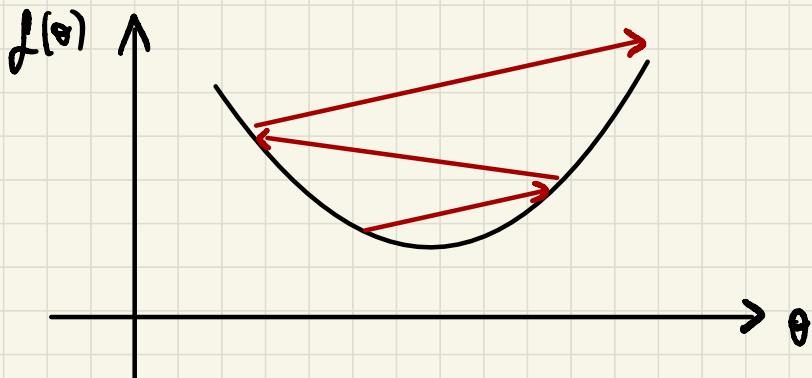
$$\begin{bmatrix} \theta_1 \\ \theta_0 \end{bmatrix}^{k+1} = \begin{bmatrix} \theta_1 \\ \theta_0 \end{bmatrix}^k - \alpha \begin{bmatrix} \frac{\partial}{\partial \theta_0} \left(\frac{1}{2n} \sum_{i=1}^n (\theta_1 x_i + \theta_0 - y_i)^2 \right) \\ \frac{\partial}{\partial \theta_1} \left(\frac{1}{2n} \sum_{i=1}^n (\theta_1 x_i + \theta_0 - y_i)^2 \right) \end{bmatrix}$$

Intuiciones : Tasa de Aprendizaje o "Learning rate".

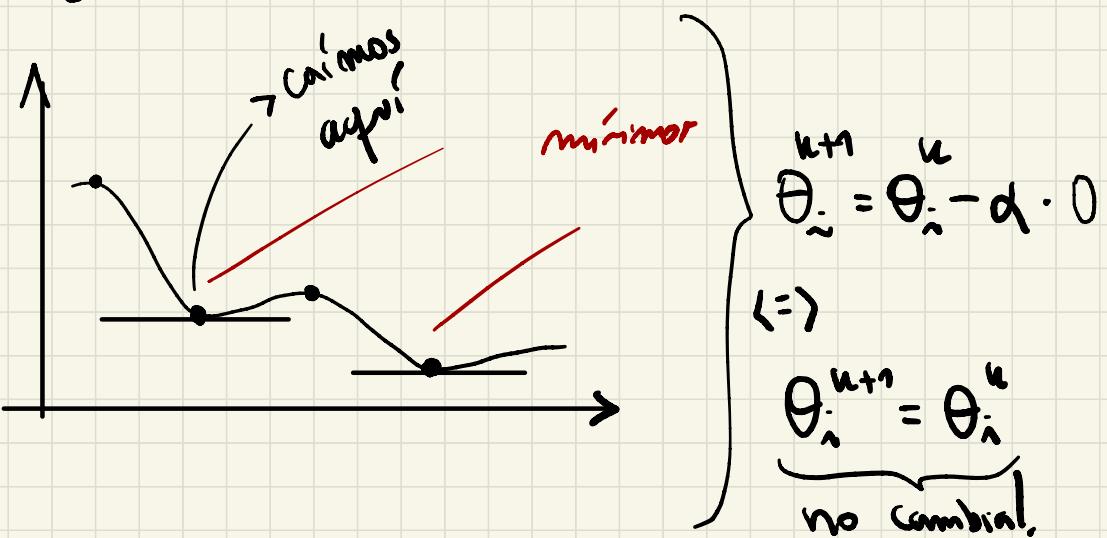
- Si tomamos $\alpha \ll 1$ (Muy Pequeño!)
 - El algoritmo da "pasos cortos"
 - Demora en converger $f(\alpha) \uparrow$
 - Gráficamente :



- Si α es muy grande
 - Podemos diverger



- ¿Qué ocurre si la función tiene más de un mínimo (Varios mínimos locales)
y caemos en uno de ellos?



- Es decir, quedamos atrapados en un mínimo local.

Construcción más formal (Justificación de por qué funciona).

Dada la función $f : \mathbb{R}^m \rightarrow \mathbb{R}$

Queremos representarla por su aproximación en series de Taylor en las cercanías de un punto \tilde{x} , es decir:

$$f(x) = f(\tilde{x}) + \nabla f(\tilde{x})^T (x - \tilde{x}) + \frac{1}{2} (x - \tilde{x})^T H (x - \tilde{x})$$

↓
Hessiano

Si approximamos $f(x)$ por:

$$f(x) \approx f(\tilde{x}) + \nabla f(\tilde{x})^T (x - \tilde{x})$$

Aproximación
de primer orden

$$\text{Si } x = \tilde{x} + u \Rightarrow f(\tilde{x} + u) \approx f(\tilde{x}) + \nabla f(\tilde{x})^T u$$

Nosotros queremos encontrar $f(\tilde{x} + u) < f(\tilde{x})$, es decir una dirección u donde vamos descendiendo!

$$\therefore f(\tilde{x} + u) - f(\tilde{x}) < 0 \quad (1)$$

} De la
desigualdad
anterior!

- Luego si: $f(\tilde{x} + u) \approx f(\tilde{x}) + \nabla f(\tilde{x})^T u$
entonces: $\nabla f(\tilde{x})^T u < 0$ por (1)
- Vamos a querer encontrar u , la dirección de
máximo descenso:

$$u^* = \min_u \nabla f(\tilde{x})^T \cdot u$$

\parallel (def de prod interno)

$-\nabla f(\tilde{x})$

} El vector paralelo
maximiza el
producto interno
y el signo menor
minimiza la expresión

$$\therefore f(\tilde{x} + u) = f(\tilde{x}) - \nabla f(\tilde{x})$$

Sin embargo, queremos tener control del paso
que damos entonces agregamos un hiperparámetro:

$$f(\tilde{x} + u) = f(\tilde{x}) - \alpha \nabla f(\tilde{x})$$

Observación :

* La serie de Taylor permite justificar porque funciona el método pero la diferencia radica en:

$$f(x^{n+1}) = f(x^n) - \alpha \nabla f(x^n) \quad \leftarrow \text{taylor}$$
$$\downarrow \qquad \downarrow$$
$$x^{n+1} = x^n - \alpha \nabla f(x^n) \quad \leftarrow \text{Gradiente Descendente}$$

La construcción completa es:

"An introduction to the Conjugate Gradient Method without the Agonizing Pain" Edition 1 1/4

Jonathan Richard Shewchuk

estamos
avanzando directo
en la dirección
de bisogada)
lo en la
función f.