CIS

Regresión Lineal

APRENDIZAJE DE MÁQUINA
Fernando Teshima

UNIVERSIDAD DE LIMA

Regresión Lineal y Gradient Descent

Breve Recordatorio

- n número de observaciones o registros
- p número de variables o features

Ejemplo:

n=5000 habitantes y p=12 características (features)

Breve Recordatorio

- Una variable del tipo $x_{ij}, x_{i,j}, x_j^{(i)}$ indicaría el valor del feature j para la observación i; es decir:

$$i = 1,2, . . , n$$
$$j = 1,2, \dots, p$$

En forma de matriz: $\boldsymbol{X}_{np}$

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}$$

Breve Recordatorio

	sqft	City	bedrooms	baths	price
x_i	3392	Dublin	3	2.1	339000
	4100	pleasanton	4	3	899900
	3200	Clayton	5	4	448641
	1436	Moraga	4	3	239999
	1944	Antioch	3	2	377500
	1500	Danville	3	2.5	299900
	1700	El Dorado Hills	4	3	265000
	2507	Shingle Springs	4	3	449000
	1580	McKinleyville	3	2	439950
	1500	Marina	4	2	699888
	2705	Roseville	3	2	1250000
	1715	Rocklin	4	3	439000

b)

$$x_i = \begin{matrix} 3392 \\ \text{Dublin} \\ 3 \\ 2.1 \end{matrix}$$

c)

$$\boldsymbol{x}_j = \begin{matrix} 3392 \\ 4100 \\ 3200 \\ 1436 \\ 1944 \\ 1500 \\ 1700 \\ 2507 \\ 1580 \\ 1500 \\ 2705 \\ 1715 \end{matrix}$$

d)

$$\boldsymbol{X} = \begin{array}{cccc} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_p \\ \text{sqft} & \text{City} & \text{bedrooms} & \text{baths} \\ \begin{bmatrix} 3392 \\ 4100 \\ 3200 \\ 1436 \\ 1944 \\ 1500 \end{bmatrix} & \begin{bmatrix} \text{Dublin} \\ \text{pleasanton} \\ \text{Clayton} \\ \text{Moraga} \\ \text{Antioch} \\ \text{Danville} \end{bmatrix} & \begin{matrix} 3 \\ 4 \\ 5 \\ 4 \\ 3 \\ 3 \end{matrix} & \begin{bmatrix} 2.1 \\ 3 \\ 4 \\ 3 \\ 2 \\ 2.5 \end{bmatrix} \end{array}$$

$$\boldsymbol{X} = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix} \quad \begin{matrix} 3392 & 4100 & 3200 & 1436 & \cdots \\ \text{Dublin} & \text{pleasanton} & \cdots & & \\ 2.1 & 3 & 4 & & \end{matrix}$$

Breve Recordatorio

- $\boldsymbol{x}^{(i)}$ o $\boldsymbol{x}_i$: Variable de entrada, input feature, donde i es el item
- $\boldsymbol{y}^{(i)}$ o $\boldsymbol{y}_i$: Variable de salida, target variable

- $x_{ij}, x_{i,j}, x_j^{(i)}$: Dato particular de la observación / item

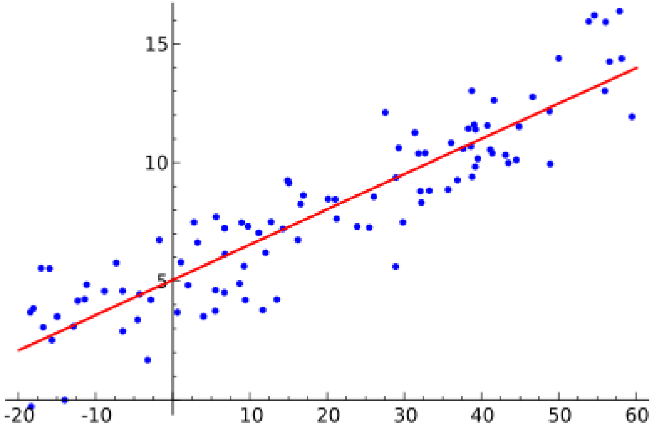
- Data de entrenamiento: $(\boldsymbol{x}^{(\mathrm{i})}, \boldsymbol{y}^{(\mathrm{i})})$

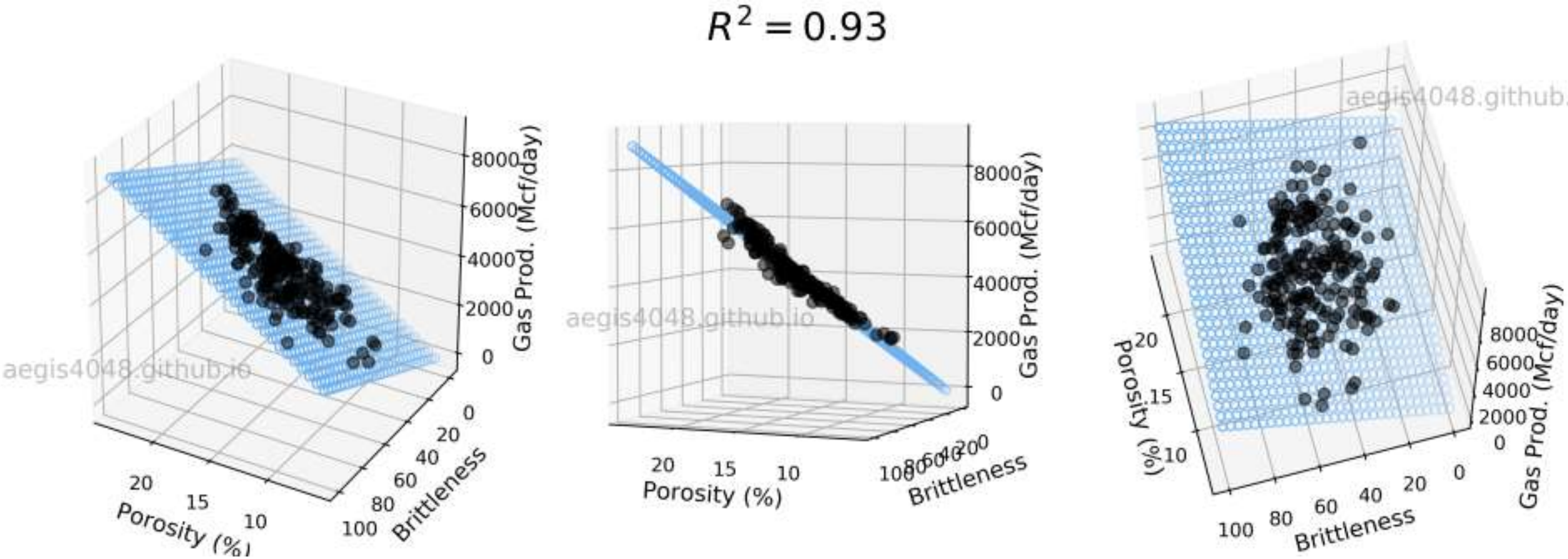
Regresión Lineal

- Es un método de aprendizaje supervisado
- Nos ayuda a predecir valores

Regresión Lineal

- Relación lineal de una variable independiente con una o más variables dependientes





Regresión Lineal

Una sola variable independiente: $y = ax + b$

Dos variables independiente: $z = ax + by + c$

Nos quedamos sin letras…. Y estamos cambiando letras a cada rato

Generalizar

$$f(x) = \ w_0 + w_1 x_1 + w_2 x_2 \dots w_n x_n$$

Regresión Lineal en ecuaciones

$$f(x) = w_0 + w_1x_1 + w_2x_2 \dots w_nx_n$$

$$f(x) = \sum_{i=0}^{n} w_ix_i$$

$$W = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} \quad ; X = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

Aplicando multiplicación de matrices y recordando el producto escalar

$$W = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} \quad ; X = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$f(x) = W^T X$$

Queremos encontrar los coeficientes

$$f(w, x) = W^T X$$

De tal forma que

$$f(w, x_i) \approx y_i$$

Ejemplo

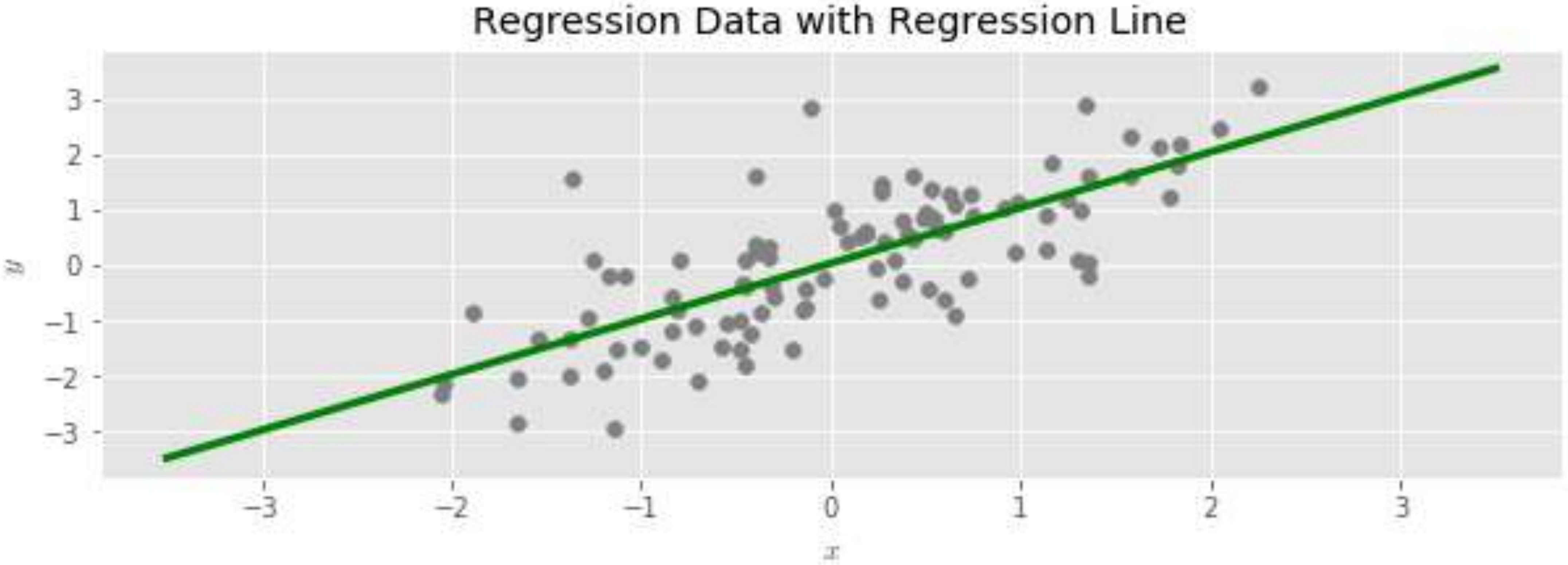
$$f(w,x) = W^T X$$

sqft	City	bedrooms	baths	price
3392	Dublin	3	2.1	339000
4100	pleasanton	4	3	899900
3200	Clayton	5	4	448641

$$price = w_0 + w_1 sqft + w_2 bedroom + \cdots$$

Esta es nuestra situación ideal

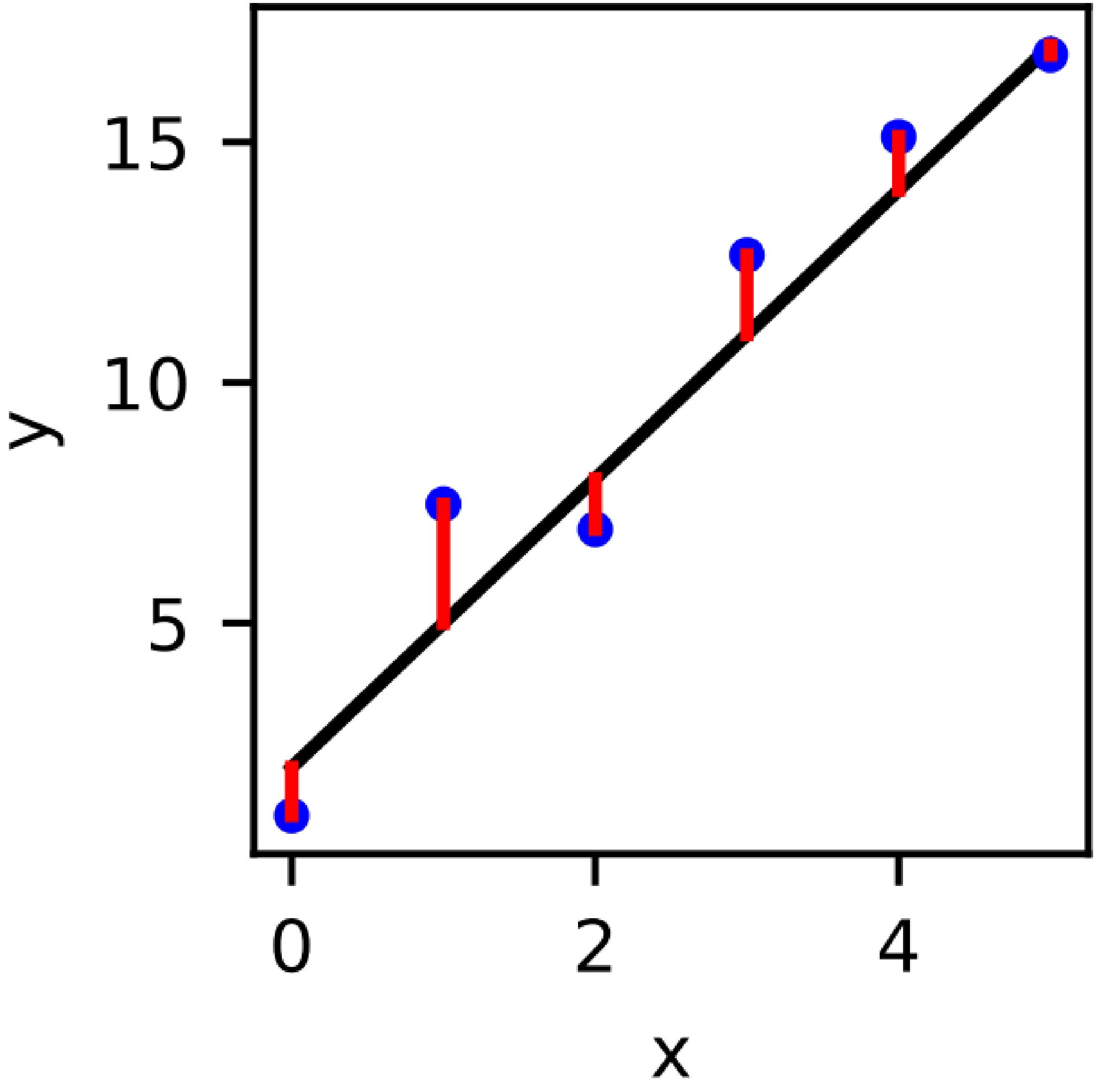
El mundo real es ruidoso



Residuales

- Los residuales, mostrado en rojo, son una métrica del fit de una función (línea negra) con respecto a un conjunto de data (azul)

$$r_i(w) = y_i - f(w, x_i)$$



Una forma simple es un problema de optimización. Definiendo una función de pérdida

$$\mathcal{L}(w)$$

Queremos encontrar los w que minimizen el “error”

$$\boldsymbol{w}^* = \min_{w} \mathcal{L}(w)$$

- Definiendo la función de pérdida en función a los Mínimos Cuadrados (Least-Squares Error) (minimizar RSS, SSR, SSE….)

$$\mathcal{L}_{LSE}(W) = \sum_{i=1}^{n} r_i^2 = r^T r$$

*Recordemos que todas nuestras variables son vectores

- Reemplazando términos

$$\mathcal{L}_{LSE}(W) = \sum_{i=1}^{n} (y_i - W^T X_i)^2$$

X = variable independiente (features)

y = variable dependiente (target)

W = coeficientes o pesos

Intuitivamente

Mientras más cercano nuestra función se asemeje a los datos, menor es el costo

Mientras menos el costo, menos desviación entre la línea (función) y nuestros datos

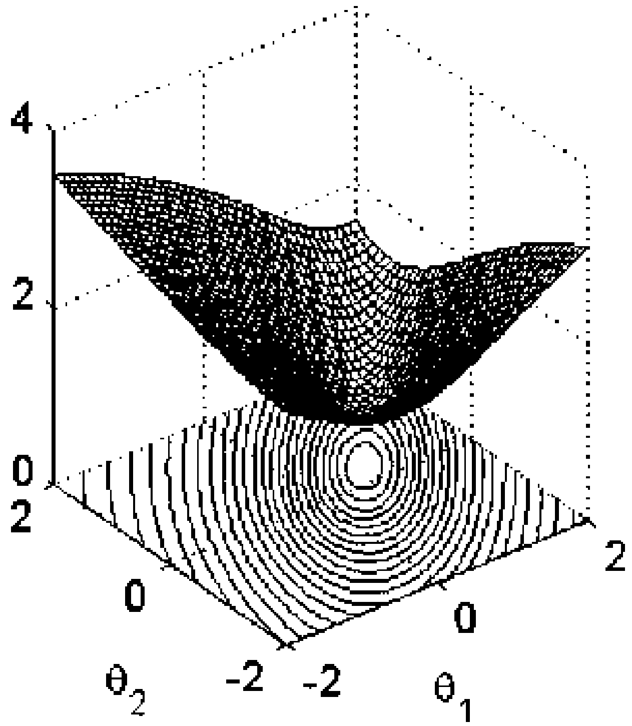
Mejor predicción!

El problema

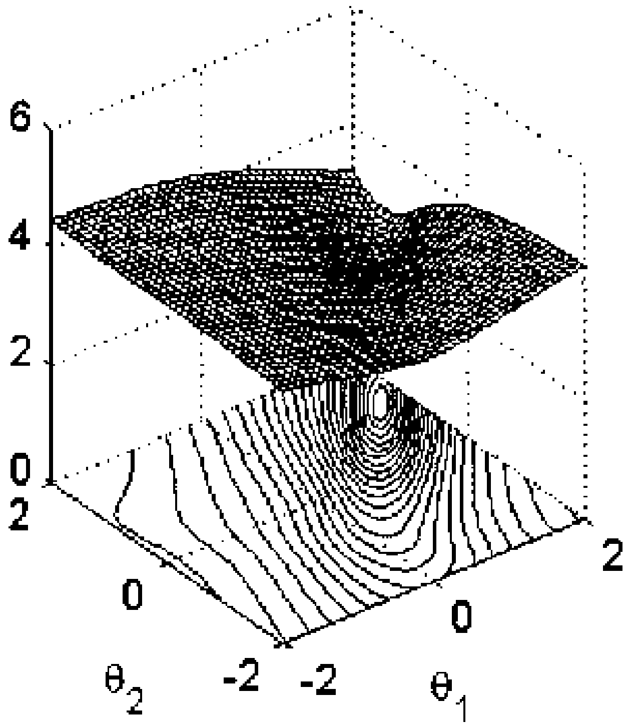
- ¿Cómo estimar los pesos W?
- De tal forma que $f(w, x_i)$ sea cercano a y_i
- ¿Cómo minimizar la función pérdida ajustando solo W?

Problema

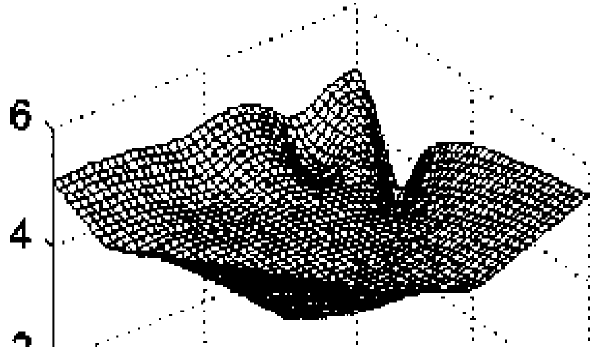
No sabemos que forma tiene la función objetivo

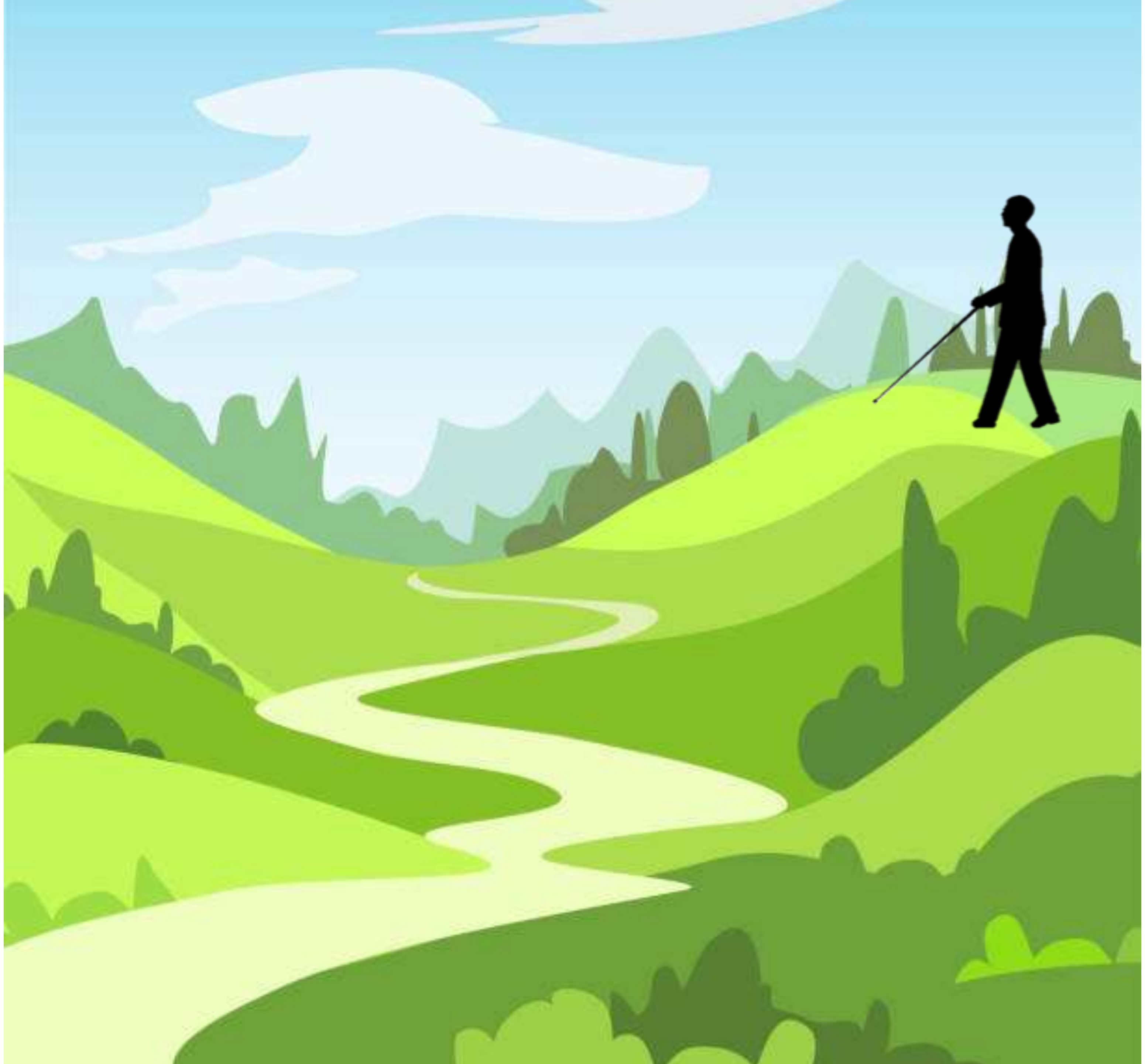


(a)



(b)





Preguntas a Resolver

Que tan empinado es donde estoy parado

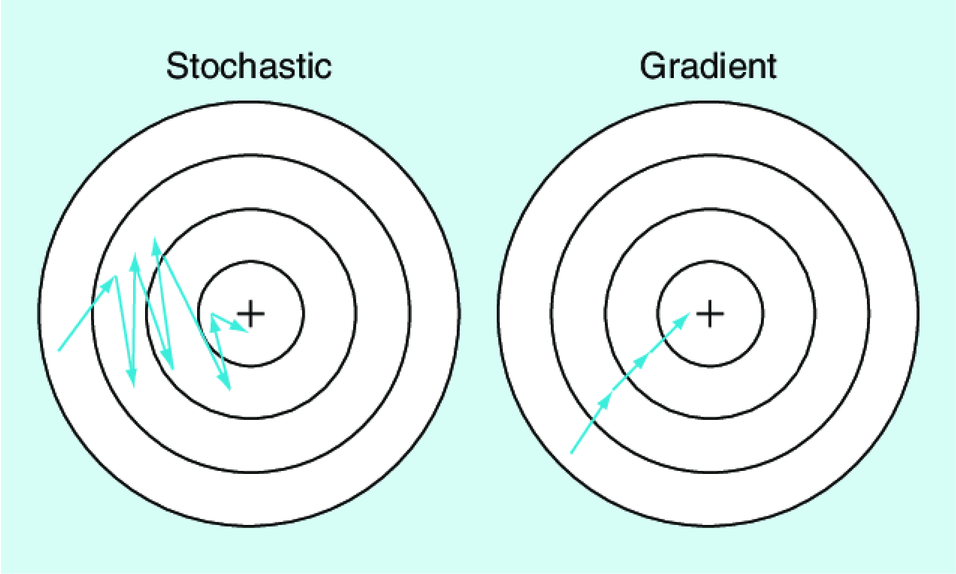
Hacia donde tengo que moverme

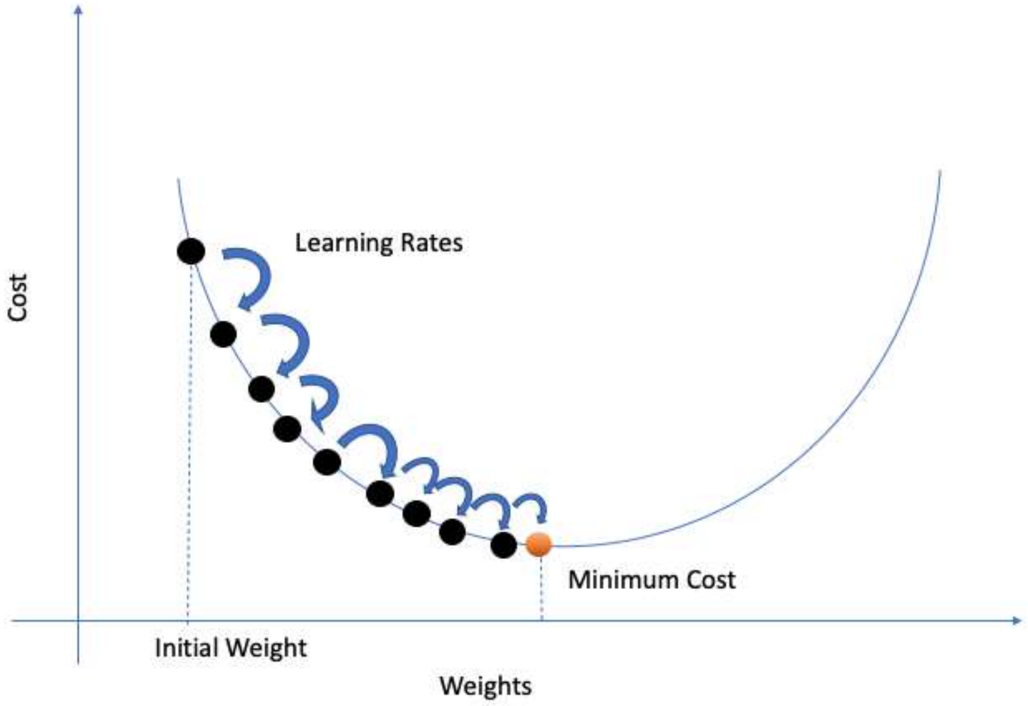
Puedo tomar
muchas medidas

Puedo tomar
una sola medida

Gradient Descent

- Gradient Descent
  - La ruta más directa
  - Usa todos los data points y luego actualiza los pesos
  - Mejor cuando son pocos
- Stochastic Gradient Decent (SGD)
  - Va actualizando pesos con cada data point

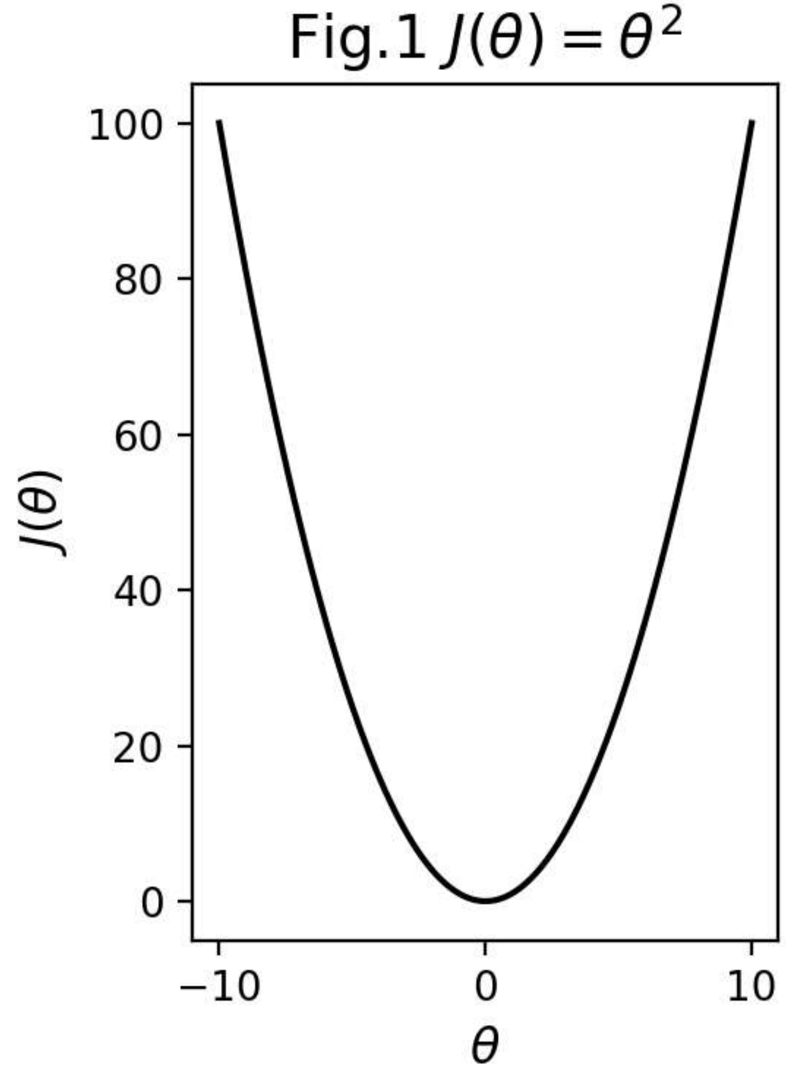




Derivadas

Ejemplo Simple

$$J(\theta) = \theta^2$$

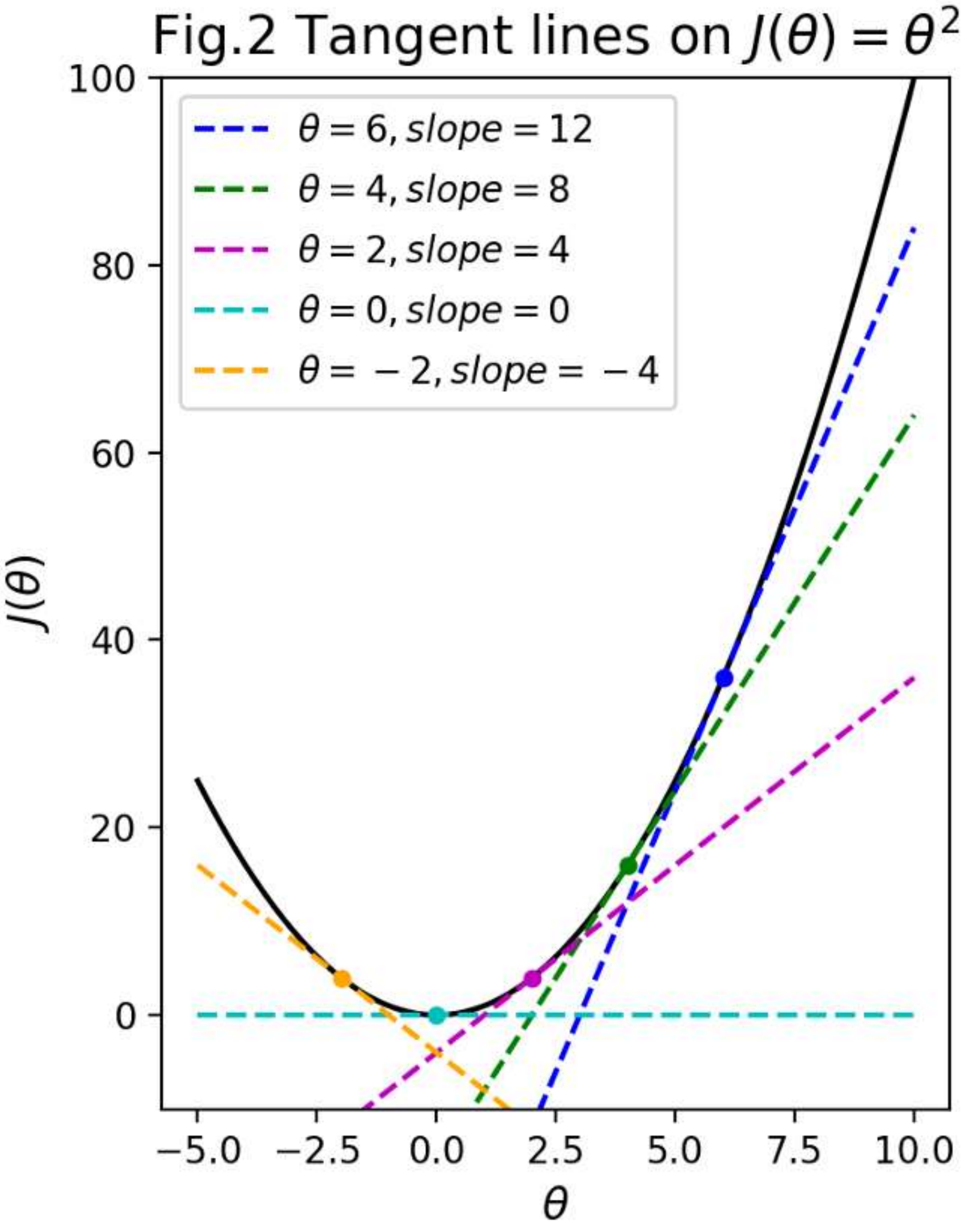


$$\min_{\theta} J(\theta)$$

1) ¿Cómo sabemos que $J(\theta)$ ha llegado al mínimo?
2) ¿Si $J(\theta)$ no está a su mínimo, como sabemos que valor de θ probar?

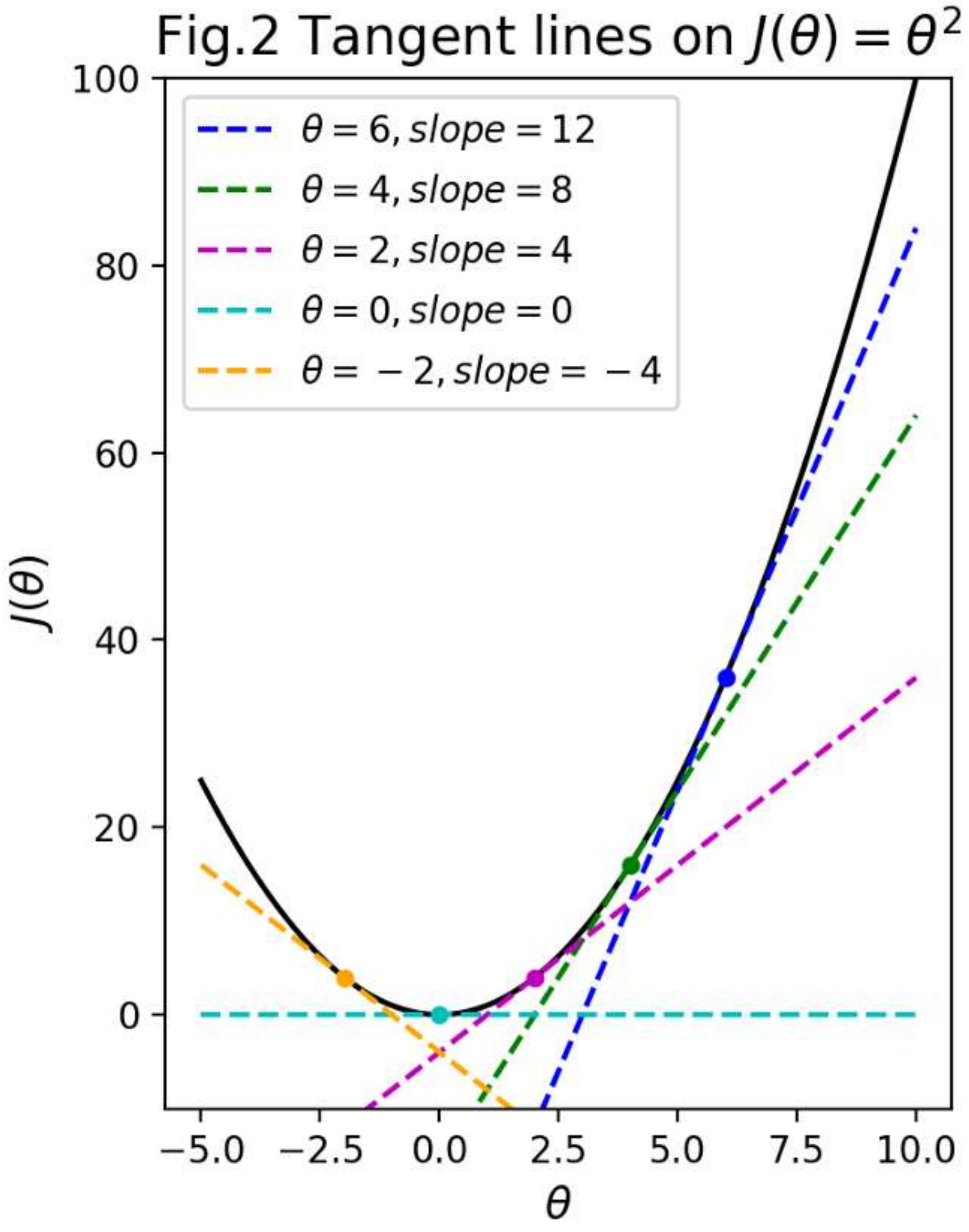
Derivadas!

- Si la pendiente es positiva, θ tiene que disminuir
- Si la pendiente es negativa, θ tiene que aumentar
- Si la pendiente es 0, felicidades! Has encontrado el mínimo (o máximo 🙈)



El Mínimo

- Si la pendiente es 0, o cambia muy poco (0.001), $J(\theta)$ ha convergido



¿Como cambiar θ ?

Regla de Actualización (Update Rule)

$$\theta = \theta - \alpha \frac{dJ}{d\theta}$$

α es conocido como "learning rate"

$\frac{\partial J}{\partial \theta}$ es la derivada de $J(\theta)$ con respecto a θ

Derivando

$$J(\theta) = \theta^2$$

$$\frac{dJ}{d\theta} = 2\theta$$

$$\theta = \theta - \alpha 2\theta$$

$$\theta = (1 - 2\alpha)\theta$$

Learning Rate

- Muy grande: El algoritmo no converge
- Muy pequeño: Demora mucho
- Es un hiperparametro
- Tradicionalmente 0.1 a 0.01
- Podría ser variable

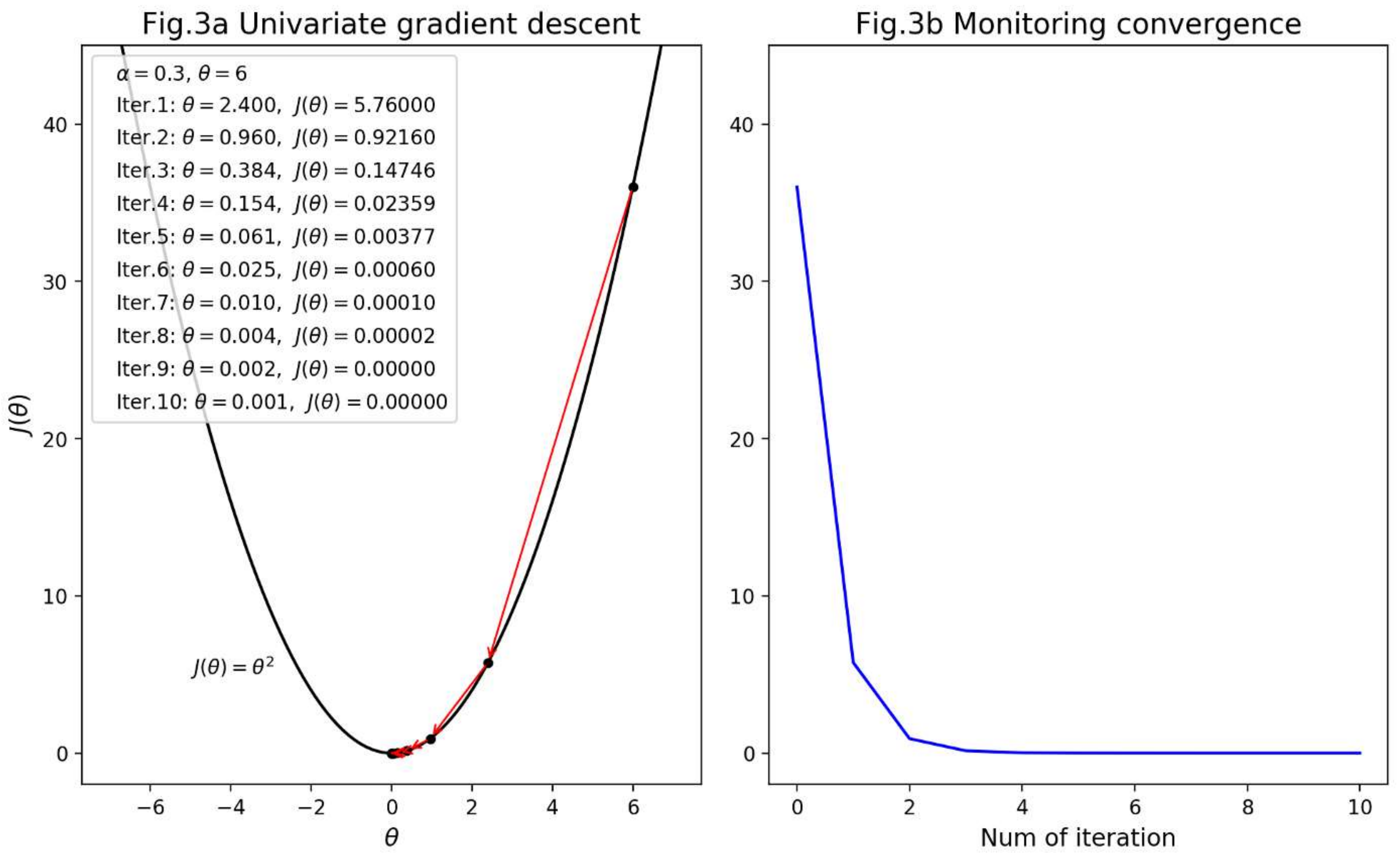


Fig.3a Univariate gradient descent
$\alpha = 0.3$, $\theta = 6$
Iter.1: $\theta = 2.400$, $J(\theta) = 5.76000$
Iter.2: $\theta = 0.960$, $J(\theta) = 0.92160$
Iter.3: $\theta = 0.384$, $J(\theta) = 0.14746$
Iter.4: $\theta = 0.154$, $J(\theta) = 0.02359$
Iter.5: $\theta = 0.061$, $J(\theta) = 0.00377$
Iter.6: $\theta = 0.025$, $J(\theta) = 0.00060$
Iter.7: $\theta = 0.010$, $J(\theta) = 0.00010$
Iter.8: $\theta = 0.004$, $J(\theta) = 0.00002$
Iter.9: $\theta = 0.002$, $J(\theta) = 0.00000$
Iter.10: $\theta = 0.001$, $J(\theta) = 0.00000$
$J(\theta) = \theta^2$
$J(\theta)$
θ
Fig.3b Monitoring convergence
Num of iteration

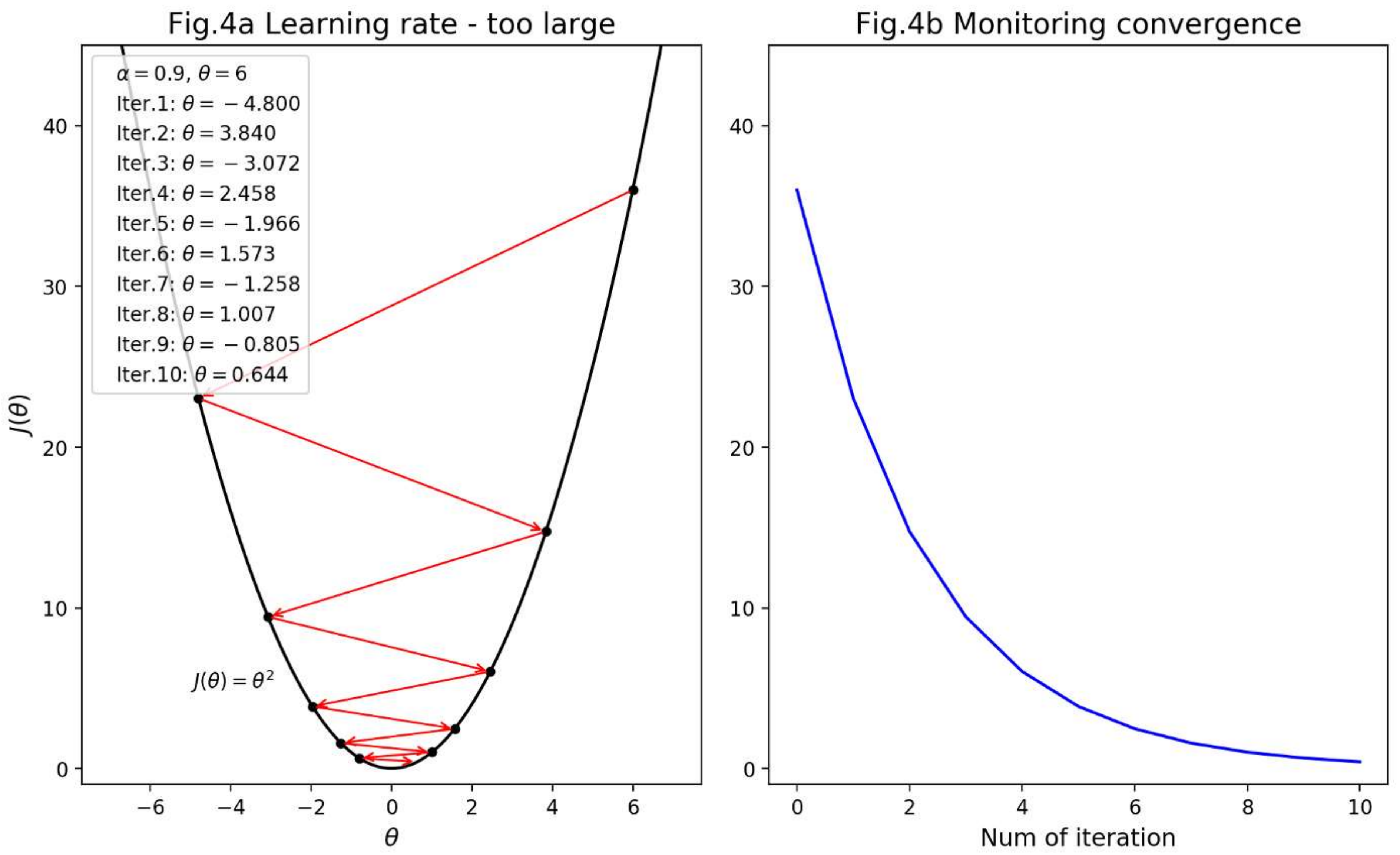


Fig.4a Learning rate - too large
$\alpha = 0.9, \theta = 6$
Iter.1: $\theta = -4.800$
Iter.2: $\theta = 3.840$
Iter.3: $\theta = -3.072$
Iter.4: $\theta = 2.458$
Iter.5: $\theta = -1.966$
Iter.6: $\theta = 1.573$
Iter.7: $\theta = -1.258$
Iter.8: $\theta = 1.007$
Iter.9: $\theta = -0.805$
Iter.10: $\theta = 0.644$
$J(\theta) = \theta^2$
$J(\theta)$
θ
Fig.4b Monitoring convergence
Num of iteration

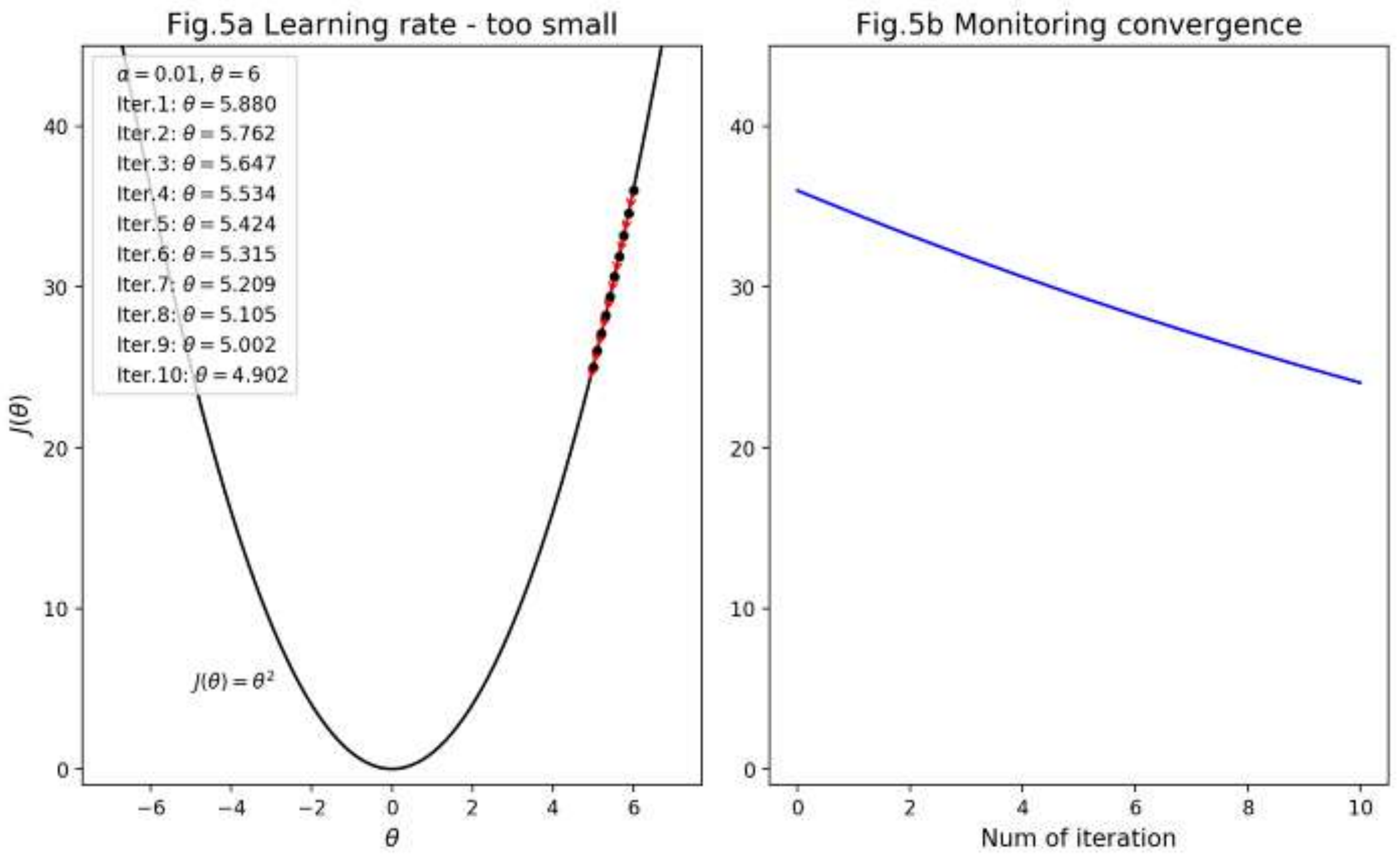
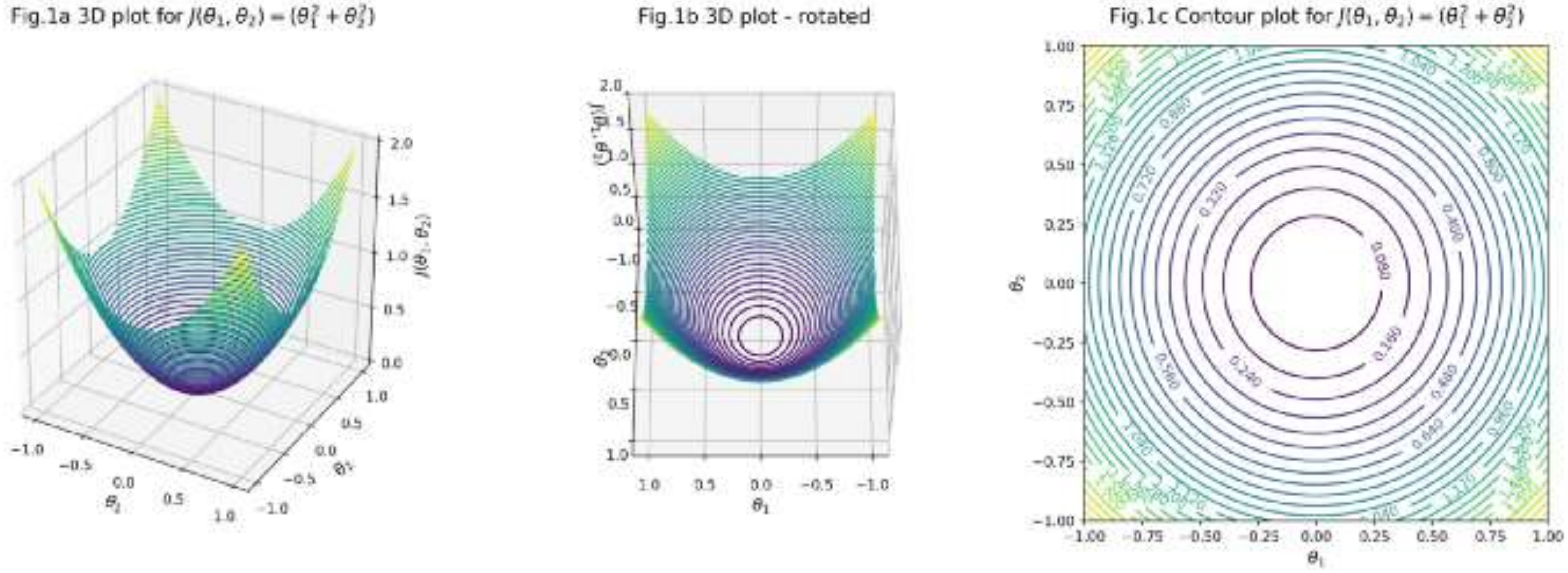


Fig.5a Learning rate - too small
$\alpha = 0.01$, $\theta = 6$
Iter.1: $\theta = 5.880$
Iter.2: $\theta = 5.762$
Iter.3: $\theta = 5.647$
Iter.4: $\theta = 5.534$
Iter.5: $\theta = 5.424$
Iter.6: $\theta = 5.315$
Iter.7: $\theta = 5.209$
Iter.8: $\theta = 5.105$
Iter.9: $\theta = 5.002$
Iter.10: $\theta = 4.902$
$J(\theta) = \theta^2$
$J(\theta)$
θ
Fig.5b Monitoring convergence
Num of iteration

Agreguemos un poco de complejidad

Dos variables independientes

$$J(\theta_0, \theta_1) = \theta_0^2 + \theta_1^2$$



Update Rule

- Tenemos que derivar a cada una de las dos variables

$$\theta_0 = \theta_0 + \alpha \frac{\partial J}{\partial \theta_0}$$

$$\theta_1 = \theta_1 + \alpha \frac{\partial J}{\partial \theta_1}$$

Razón: No podemos calcularlo en simultáneo, por eso se hace uno por uno

Derivando

$$J(\theta_0, \theta_1) = \theta_0^2 + \theta_1^2$$

$$\frac{\partial J}{\partial \theta_0} = 2\theta_0$$

$$\theta_0 = \theta_0 + \alpha \frac{\partial J}{\partial \theta_0}$$

$$\theta_0 = (1 - 2\alpha)\theta_0$$

$$\frac{\partial J}{\partial \theta_1} = 2\theta_1$$

$$\theta_1 = \theta_1 + \alpha \frac{\partial J}{\partial \theta_1}$$

$$\theta_1 = (1 - 2\alpha)\theta_1$$

Fig.3a Multivariate gradient descent

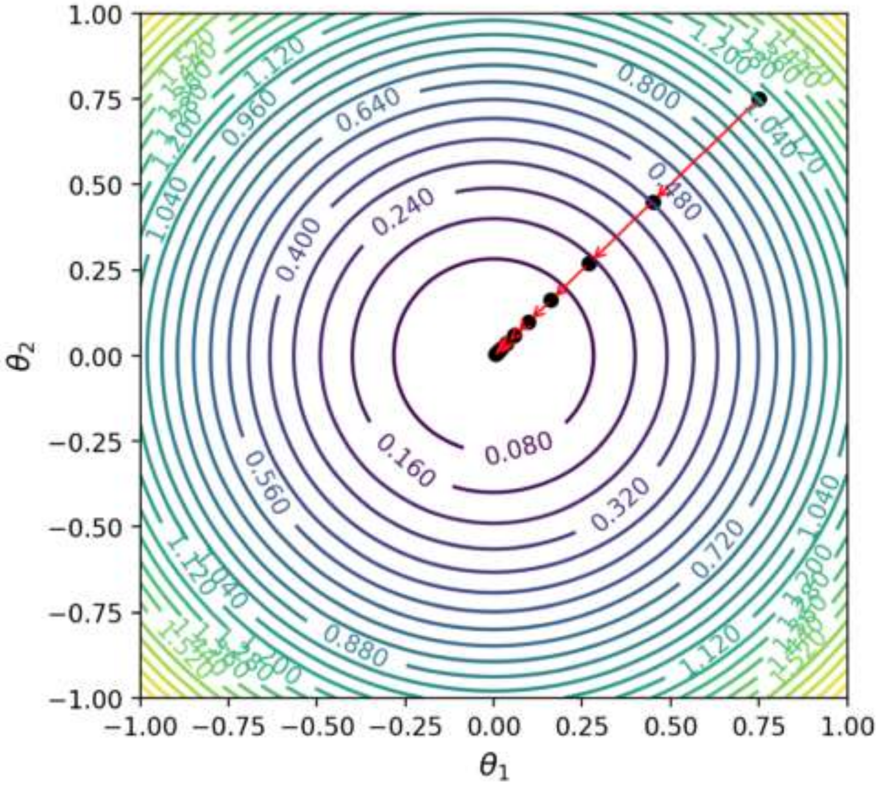
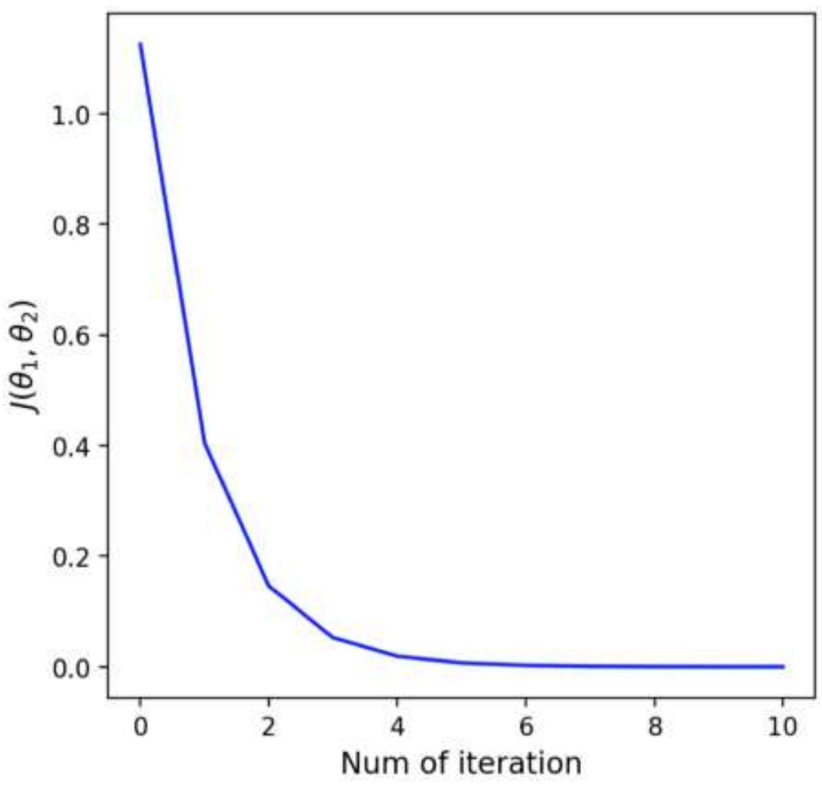


Fig.3b Monitoring convergence



Iter.1: $[\theta_1, \theta_2] = [0.450, 0.450]$, $J(\theta) = 0.40500$

Iter.2: $[\theta_1, \theta_2] = [0.270, 0.270]$, $J(\theta) = 0.14580$

Iter.3: $[\theta_1, \theta_2] = [0.162, 0.162]$, $J(\theta) = 0.05249$

Iter.4: $[\theta_1, \theta_2] = [0.097, 0.097]$, $J(\theta) = 0.01890$

Iter.5: $[\theta_1, \theta_2] = [0.058, 0.058]$, $J(\theta) = 0.00680$

Iter.6: $[\theta_1, \theta_2] = [0.035, 0.035]$, $J(\theta) = 0.00245$

Iter.7: $[\theta_1, \theta_2] = [0.021, 0.021]$, $J(\theta) = 0.00088$

Iter.8: $[\theta_1, \theta_2] = [0.013, 0.013]$, $J(\theta) = 0.00032$

Iter.9: $[\theta_1, \theta_2] = [0.008, 0.008]$, $J(\theta) = 0.00011$

Iter.10: $[\theta_1, \theta_2] = [0.005, 0.005]$, $J(\theta) = 0.00004$

Aún más complejidad!!

Loss function is back!

$$\mathcal{L}_{LSE}(W) = \sum_{i=0}^{n} (y_i - W^T X_i)^2$$

$$(w_0, w_1 \ldots \boldsymbol{w_d}) \qquad \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ \boldsymbol{y_n} \end{bmatrix} \qquad \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ \boldsymbol{w_d} \end{bmatrix} \qquad \begin{bmatrix} x_{0,0} & x_{0,1} & \cdots & x_{0,d} \\ x_{1,0} & x_{1,1} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,0} & x_{n,1} & \cdots & \boldsymbol{x_{n,d}} \end{bmatrix}$$

La Derivada

$$\frac{\partial \mathcal{L}}{\partial w_j} = \sum_{i=1}^{n} x_j^{(i)} (w_j x_j^{(i)} - y^{(i)})$$

Lo igualamos a 0, y se pone feo rápido…

Si quieren> OLS, Ecuación Normal (one-step learning).

O hacemos gradient descent...

El Gradient Descent

$$w_j \leftarrow w_j - \alpha \sum_{i=1}^{n} x_j^{(i)}(w_j x_j^{(i)} - y^{(i)})$$

Lo podemos poner más bonito y sin tantos índices.

Vectorizar el problema

- X es una matriz de $N \times D$
  - N es el número de observaciones / ítems / data points
  - D es el número de features / características / dimensiones
- W es un vector columna de D pesos
- Y es un vector columna de N elementos target
- $\hat{Y}$ es el vector columna predicho de X con W

$$\hat{Y} = \mathrm{XW}$$

Vectorizando

La función de pérdida

$$\mathcal{L} = (\mathrm{Y} - \mathrm{XW})^2$$

El Gradient Descent update rule

$$W \ \leftarrow W - \alpha X^T(\mathrm{XW} - Y)$$

Y por si lo quieren resolver en un solo paso complicado:

Ecuación Normal

$$W = (X^T X)^{-1} X^T \hat{Y}$$

Recordando Gradient Descent

1) Calcular la gradiente (derivada) de la función de pérdida en el punto actual
2) Mover en la dirección opuesta (update rule)
3) Repetir hasta converger, o el cansancio, o el final de los tiempos…

Gradient Descent

$$W \leftarrow W - \alpha X^T (\mathrm{XW} - Y)$$

Ventajas 👍

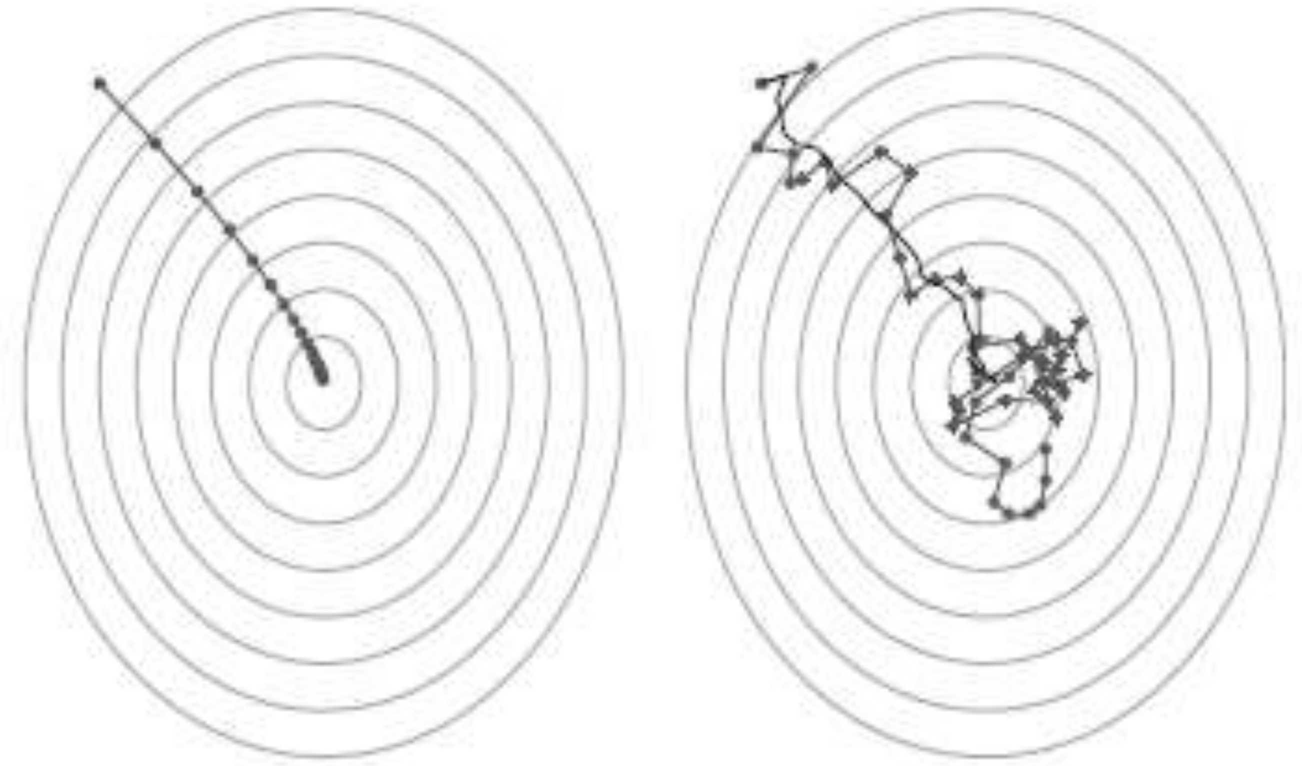
- Funciona para todas las funciones de optimización
- Eficiente en tiempos de computación

Desventaja 👎

- No llega al valor exacto, pero es suficiente
- En cada iteración tiene que usar todos los data points.

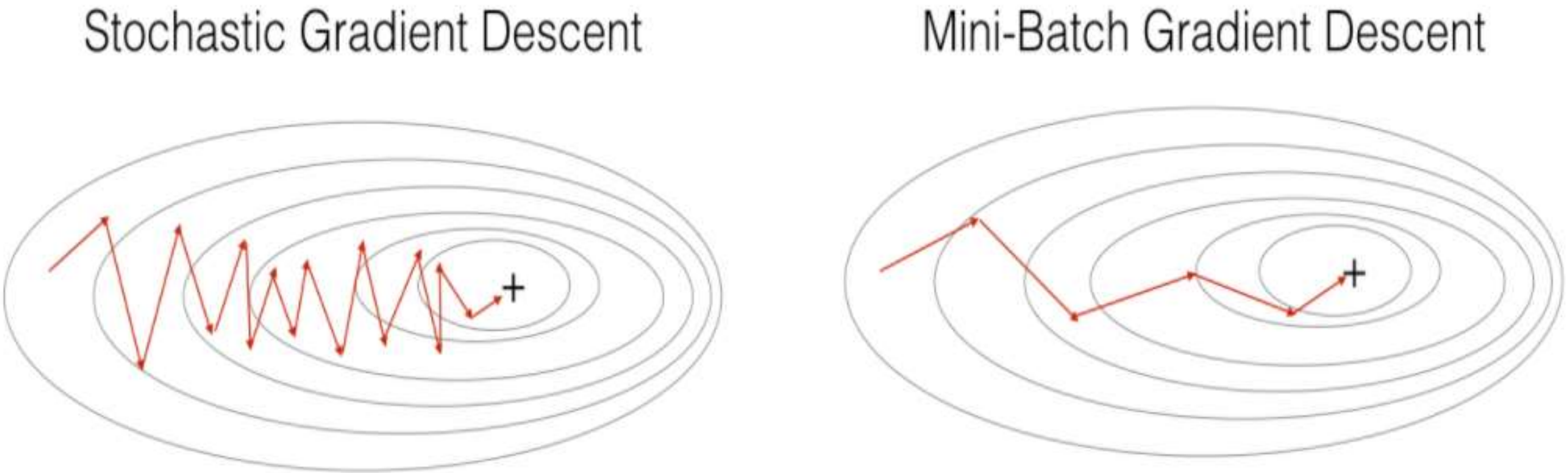
Stochastic Gradient Descent (SDG)

En lugar de tomar toda la matriz X, se toma aleatoriamente un solo ítem, se calcula la gradiente y se actualiza



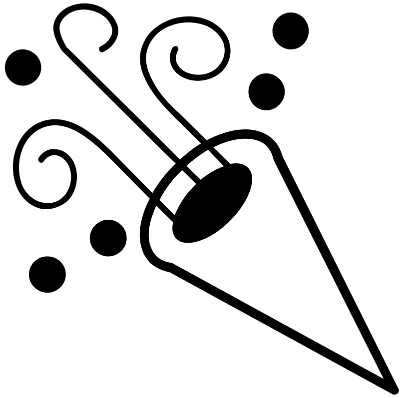
Mini-Batch Gradient Descent

En lugar de tomar toda la matriz X, se toma unos cuantos ítems en cada iteración, se calcula la gradiente y se actualiza



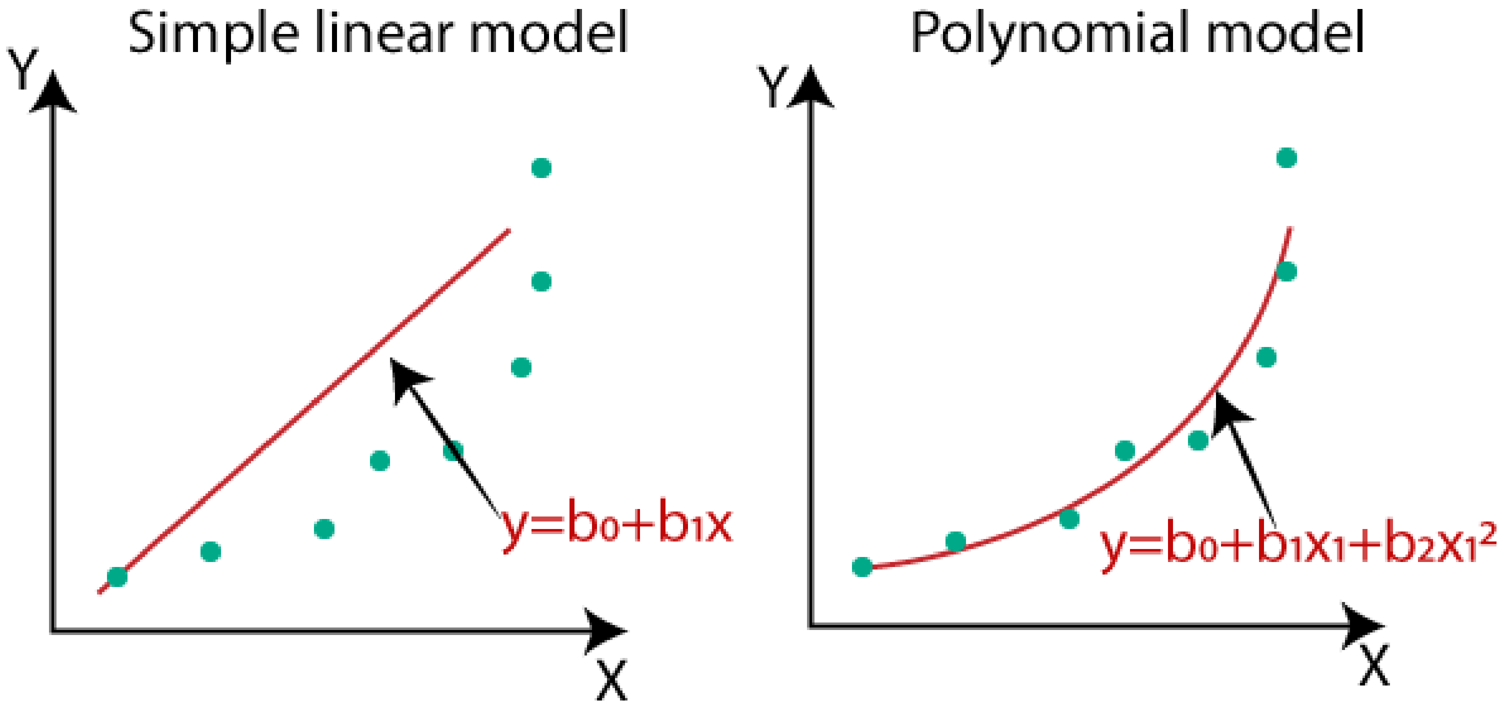
Finalizando

Cuando la gradiente converge en un valor cercano a 0, quiere decir que hemos llegado a un mínimo en la función pérdida, y que nuestros pesos se ajustan bien a la data.

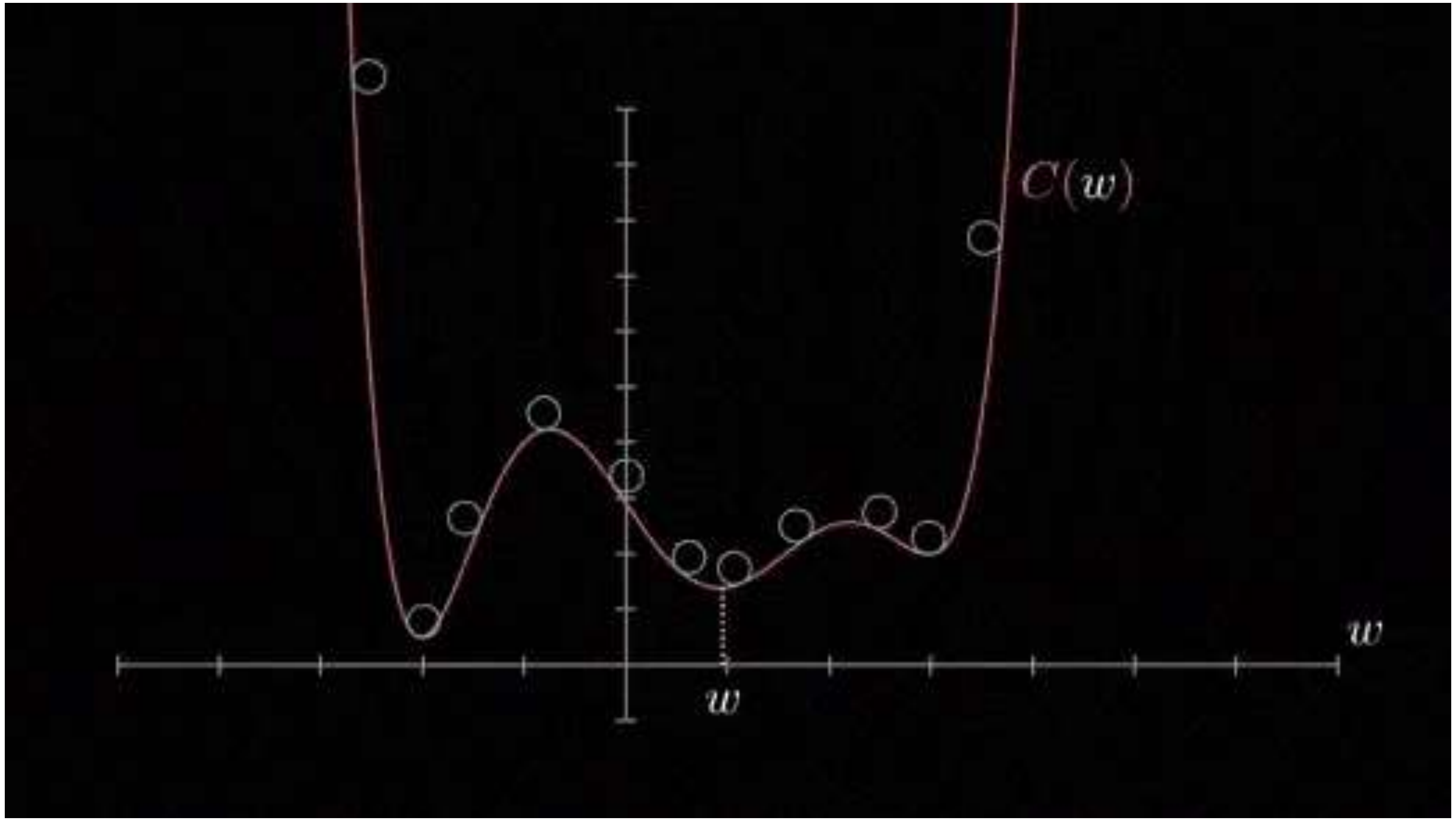



BUT WAIT!
THERE IS MORE!
imgflip.com

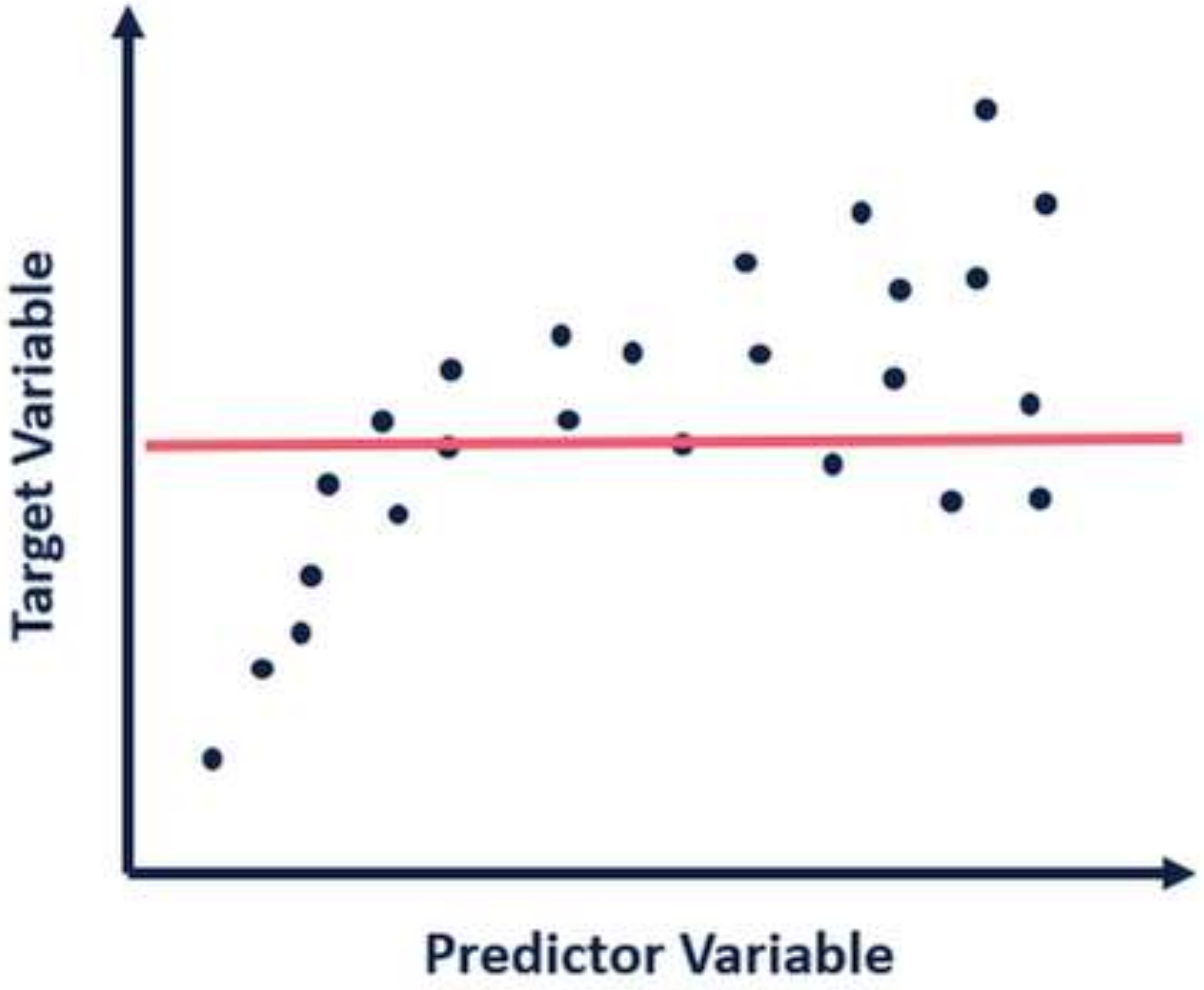
Regresión Polinomial



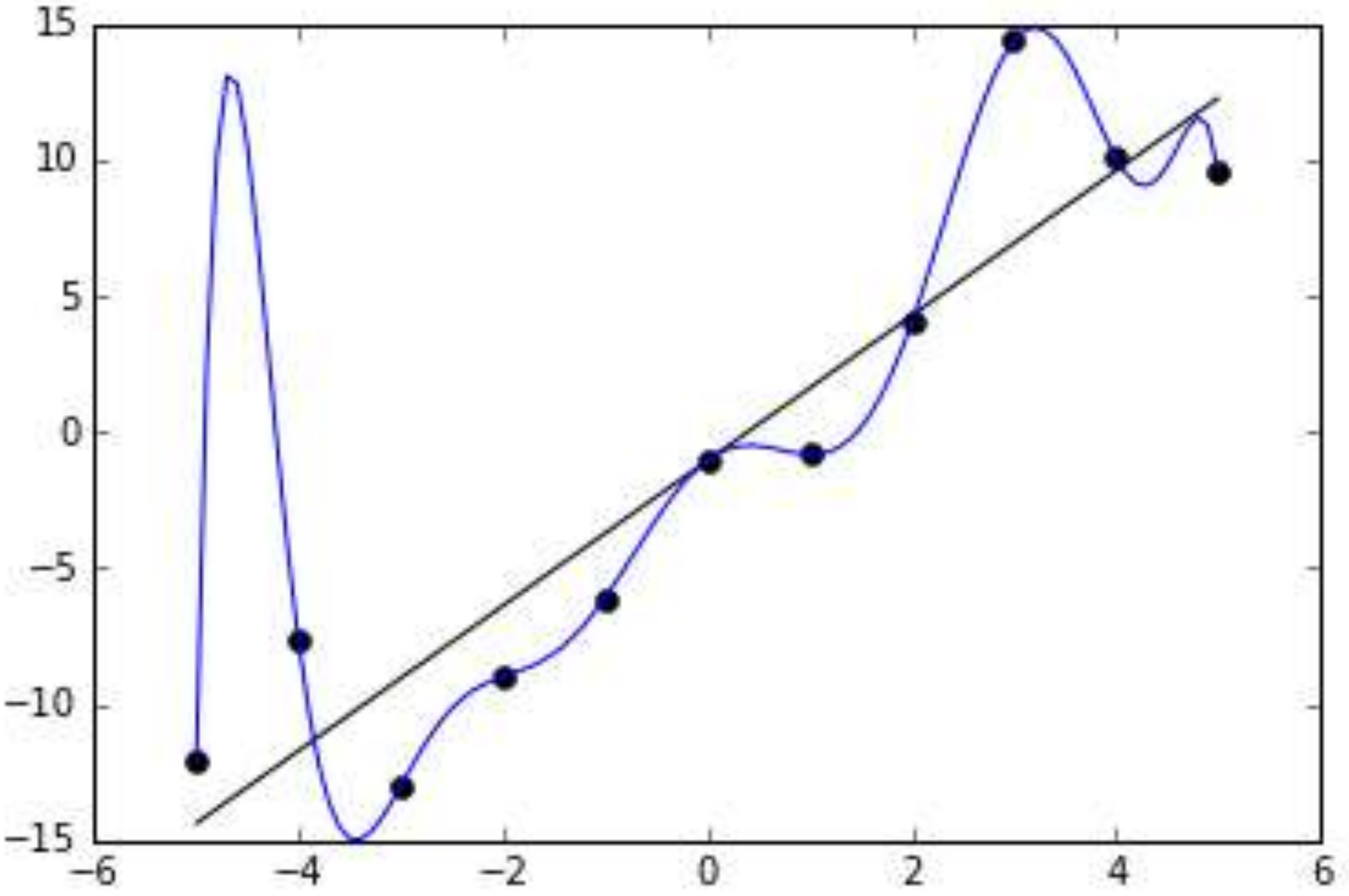
Mínimo Local y Mínimo Global



Underfitting



Overfitting



Regularización