Regresión lineal en línea

Raúl Martínez Noriega

Universidad Panamericana

Verano 2020

Aprendizaje supervisado

Los datos de entrenamiento consisten en pares de objetos: datos de entrada y su resultado deseado. El objetivo es encontrar una función que relacione la entrada con su resultado deseado, de tal forma que podamos hacer predicciones sobre el resultado de muestras futuras.

Básicamente, los problemas de aprendizaje supervisado están clasificados en:

- Regresión: tratamos de predecir resultados de una variable continua.
    - Regresión lineal
    - Redes neuronales
- Clasificación: predecir resultados de una variable discreta.
    - Regresión logística
    - Redes neuronales

Regresión lineal

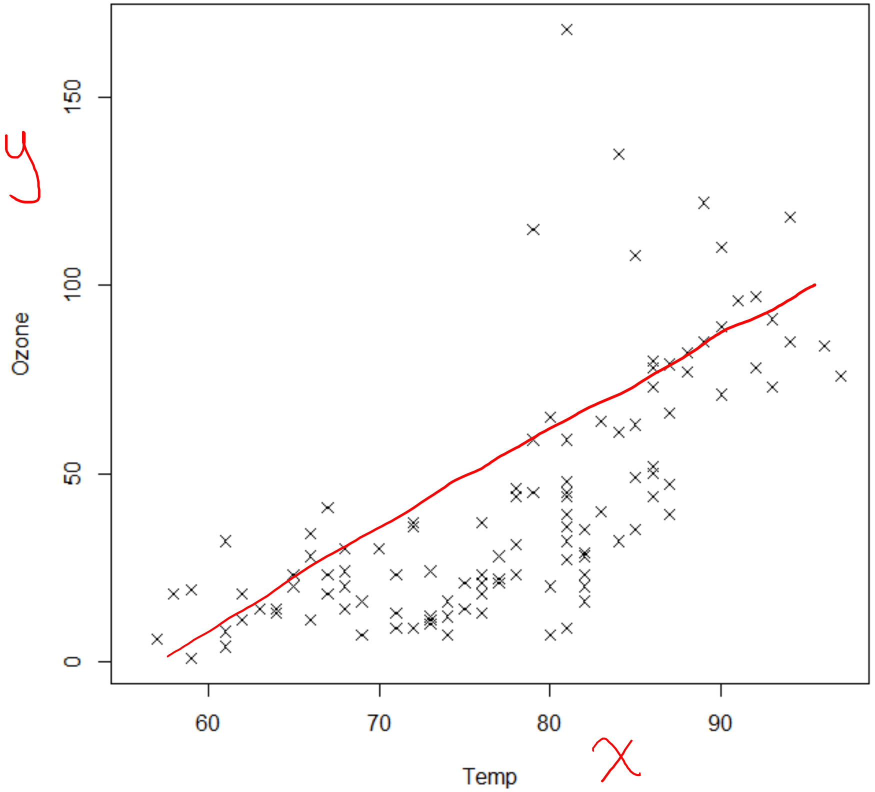
Ecuación normal

Gradient descent

Stochastic Gradient Descent (SGD)

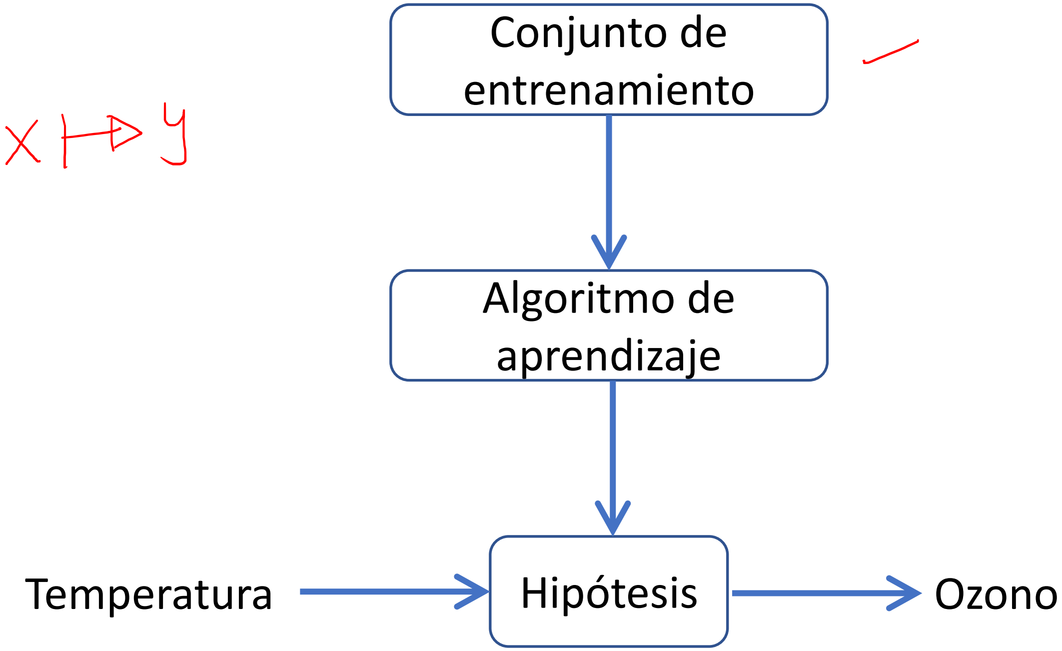
Regresión lineal

Daily air quality measurements in New York, May to September 1973.



Ozone: Mean ozone in parts per billion from 1300 to 1500 hours at Roosevelt Island
Temp: Maximum daily temperature in degrees Fahrenheit at La Guardia Airport.

$f(x) := x \mapsto y$



$\underline{f(x) = \beta_0 + \beta_1 x_1}$

reg. lineal

Función costo

Hipótesis:

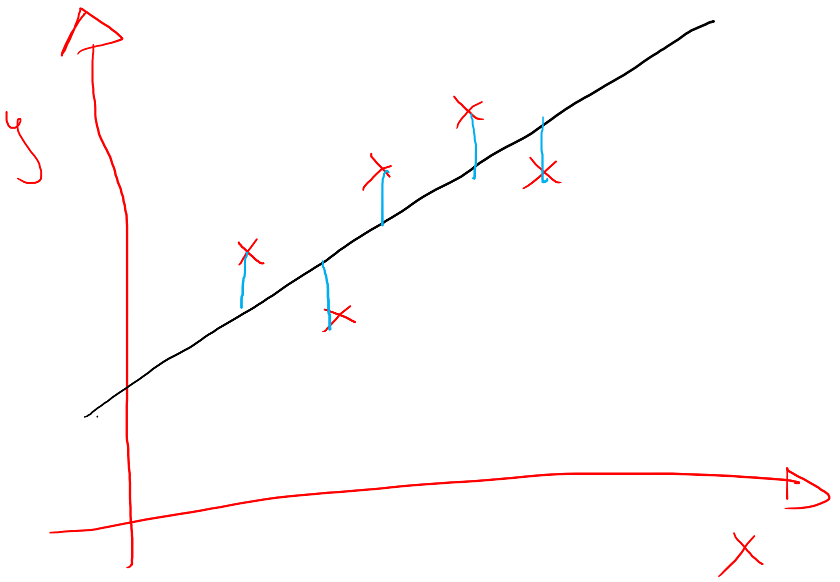
$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

$\hat{y} = \beta_0 + \beta_1 x_1$

Función costo:

$$J(\beta_0, \beta_1) = \sum_{i=1}^{n} (f(x_i) - y_i)^2$$

$$= \sum_{i=1}^{n} \left(\overbrace{\beta_0 + \sum_{j=1}^{p} X_{i,j}\beta_j}^{\hat{y}_i} - y_i \right)^2$$



minimizar $J(\beta_0, \beta_1)$

encontrar β_0 y β_1

¿Cuál es la "mejor" línea que ajusta los datos?

Ecuación normal

Hipótesis: $f(X) = \beta_0 + \sum_{j=1}^{p} X_j\beta_j$

Función costo:

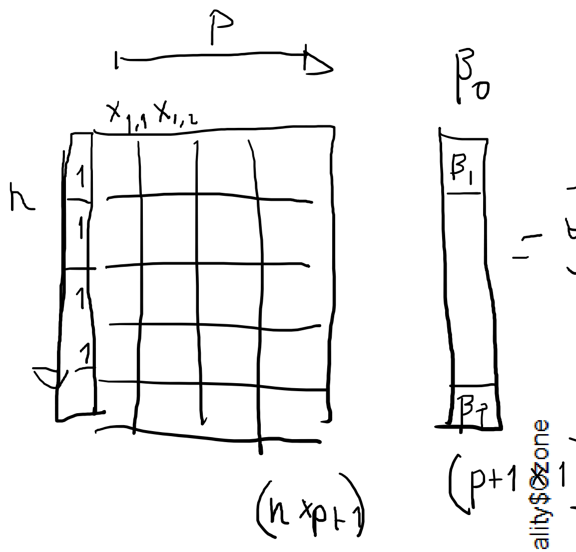
$$J(\beta_0,\beta_1) = \sum_{i=1}^{n}\Big(y_i - \overbrace{\beta_0 - \sum_{j=1}^{p} X_j\beta_j}^{\hat{y}}\Big)^2$$

$$J(\beta_0,\beta_1) = (\boldsymbol{y} - \boldsymbol{X\beta})^T(\boldsymbol{y} - \boldsymbol{X\beta})$$

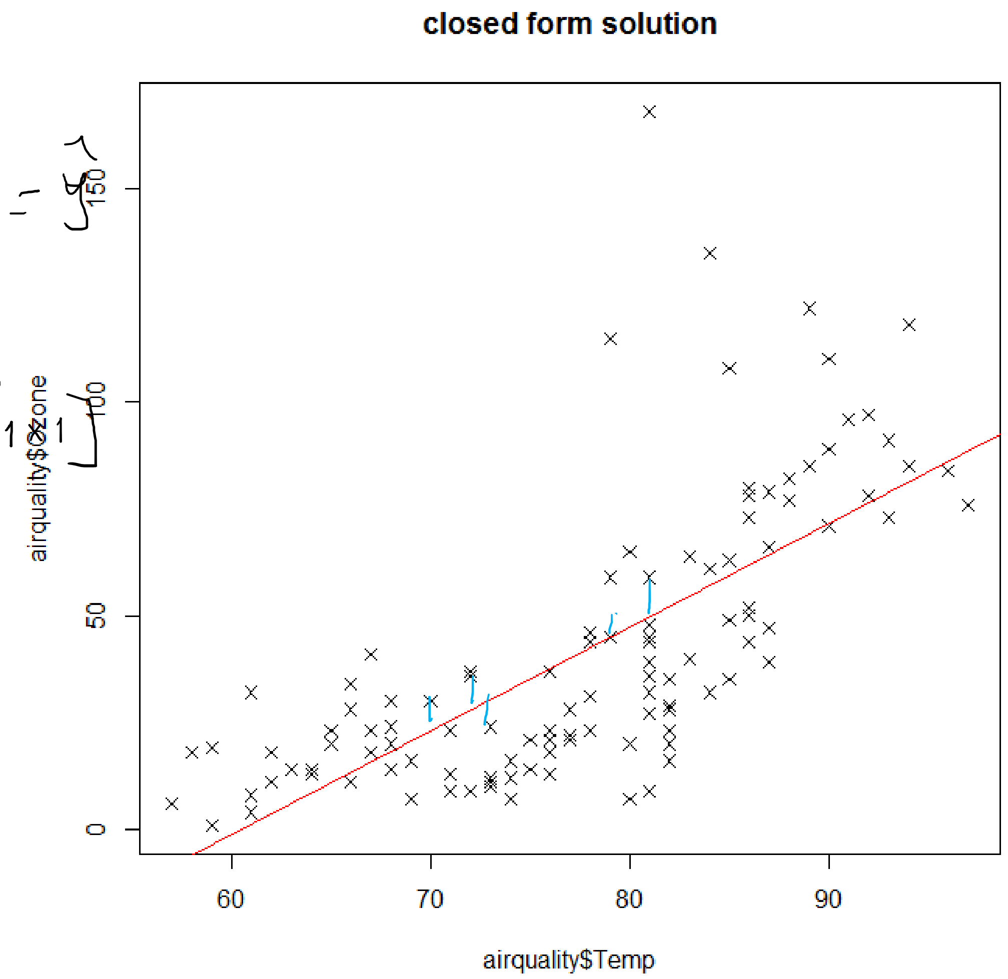
$$\frac{\partial J(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -2\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X\beta})$$

$$\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X\beta}) = \boldsymbol{0}$$

$$\hat{\beta} = \underbrace{\boxed{(\boldsymbol{X}^T\boldsymbol{X})^{-1}}}_{O(n^3)}\boldsymbol{X}^T\boldsymbol{y}$$

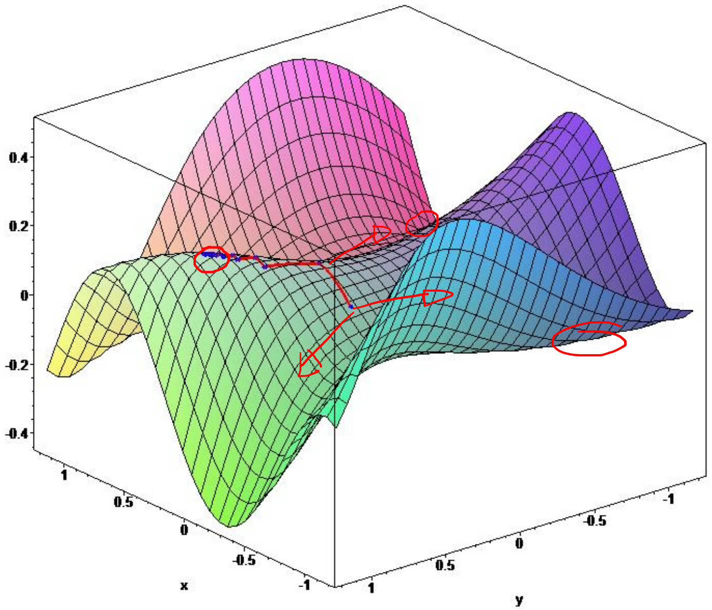


$X\beta = \hat{y}$



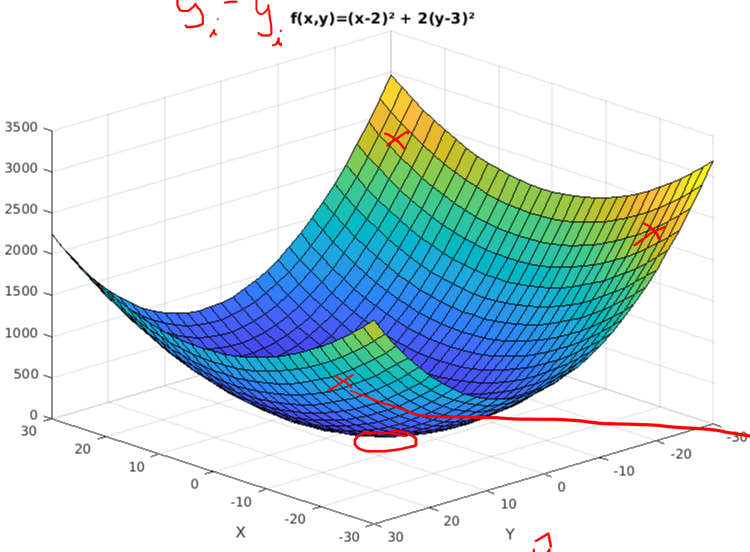
Gradient descent - I

Es un algoritmo de optimización iterativo que nos ayuda a encontrar un mínimo local de una función diferenciable.

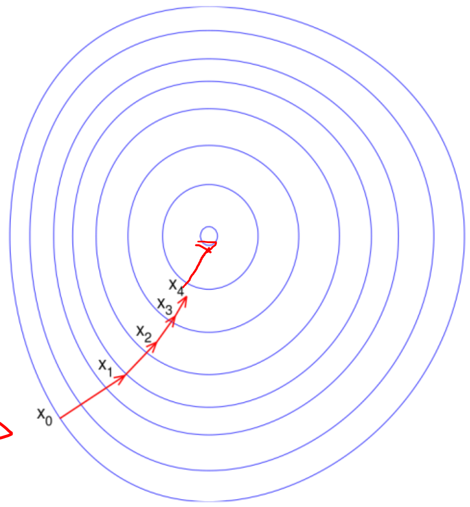


error = $(y_i - \hat{y}_i)$

$\hat{y}_i - y_i$



β_0



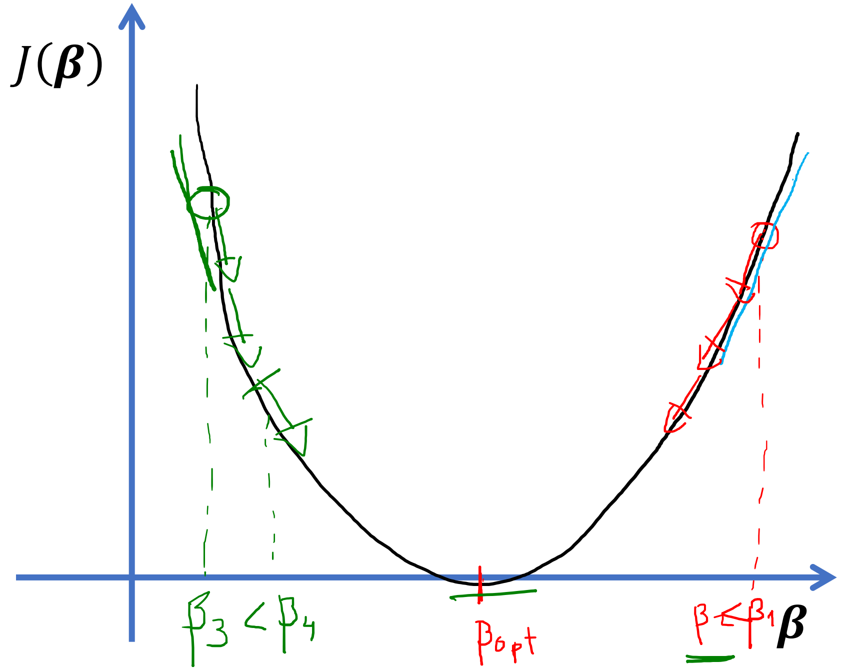
$$J(\beta_0, \beta_1) = \sum_{i=1}^{n} \left(y_i - \underbrace{\beta_0 - \sum_{j=1}^{p} X_j \beta_j}_{} \right)^2$$

$\hat{y}$

Gradient descent - II

Función costo:

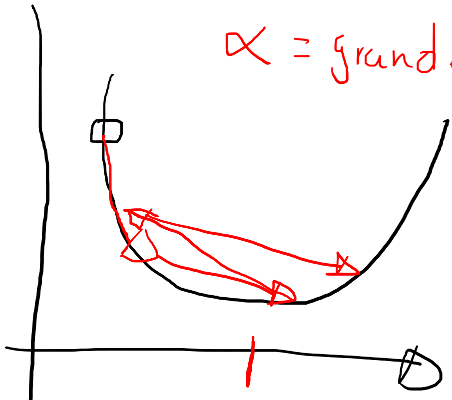
$$J(\beta_0,\beta_1) = \frac{1}{2n}\sum_{i=1}^{n}\left(\overbrace{\beta_0 + \sum_{j=1}^{p} X_{i,j}\beta_j}^{\hat{y}_i} - y_i\right)^2$$



$f(x) = x\beta$

$$\beta_2 = \beta_1 - \alpha \frac{\overset{(-)}{\partial J(\beta)}}{\beta}$$

α = grande



Gradient descent:

Repetir hasta converger{

$$\beta_j = \beta_j - \alpha\frac{\partial}{\partial\beta_j}J(\boldsymbol{\beta})$$

paso

}

Por simplicidad, asumamos una regresión lineal univariable:

$$\frac{\partial}{\partial\beta_j}J(\boldsymbol{\beta}) = \frac{\partial}{\partial\beta_j}\frac{1}{2n}\sum_{i=1}^{n}(\beta_0 + x_i\beta_1 - y_i)^2$$

$$= \frac{\cancel{2}}{\cancel{2}n}\sum_{i=1}^{n}(\overbrace{\beta_0 + x_i\beta_1}^{\hat{y}} - y_i)\frac{\partial}{\partial\beta_j}(\beta_0 + x_i\beta_1 - y_i)$$

$\beta_0 = 1$

$\beta_1 = x_i$

Por tanto las derivadas parciales son:

$$\frac{\partial}{\partial\beta_0}J(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i) \quad 1$$

$$\frac{\partial}{\partial\beta_1}J(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}x_i * (\hat{y}_i - y_i)$$

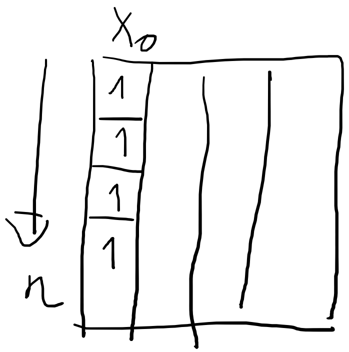
Gradient descent - III

Linear regression

Update rules:

$$\beta_0 = \beta_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)$$

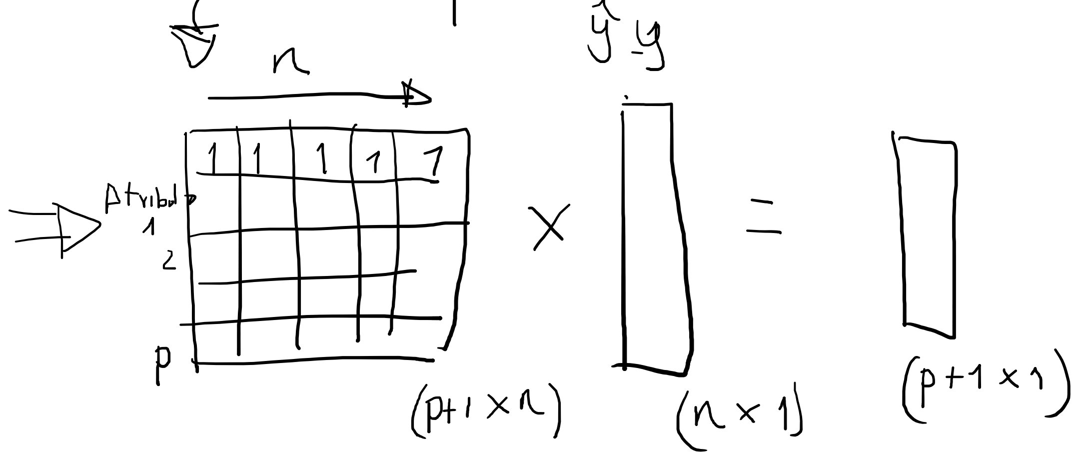
$$\beta_1 = \beta_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} x_i (\hat{y}_i - y_i)$$



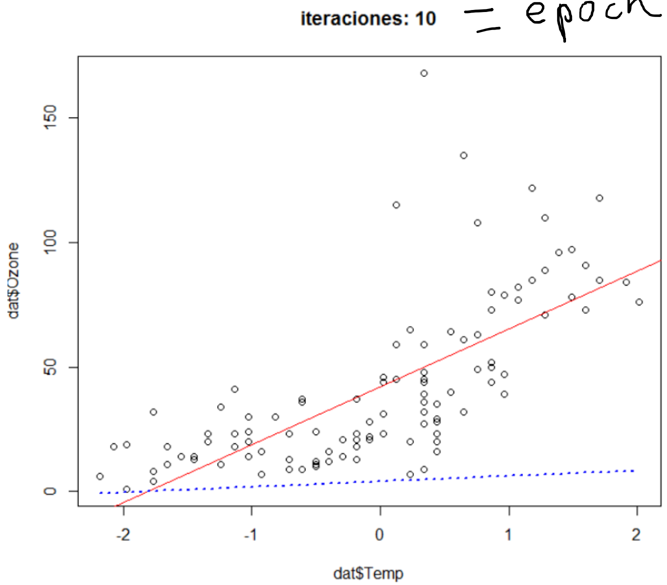
Multivariate linear regression

Update rule:

$$\boldsymbol{\beta} = \boldsymbol{\beta} - \alpha \frac{1}{n} \boldsymbol{X}^T \, (\hat{\boldsymbol{y}} - \boldsymbol{y})$$

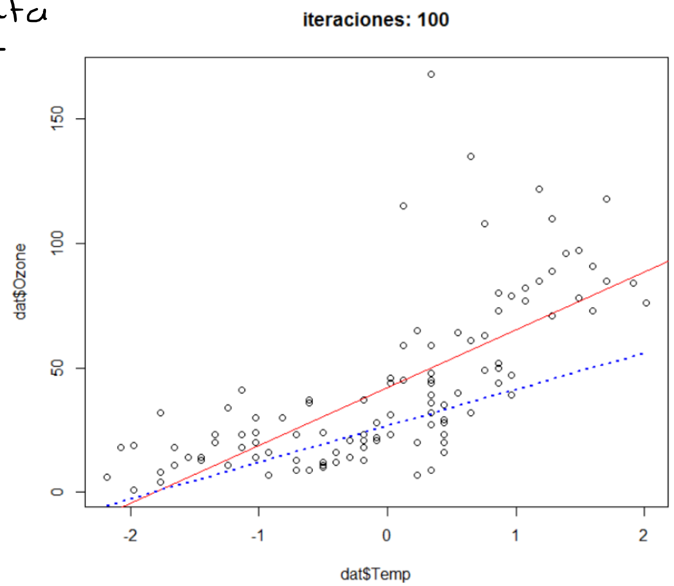


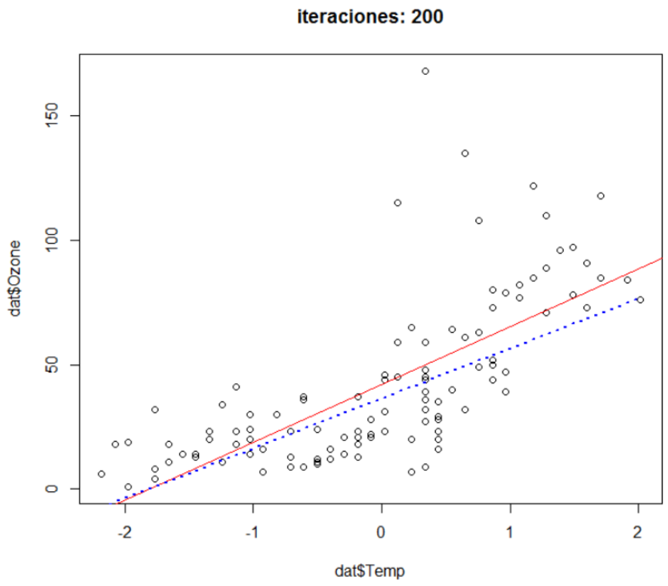
Ejemplo



= epoch

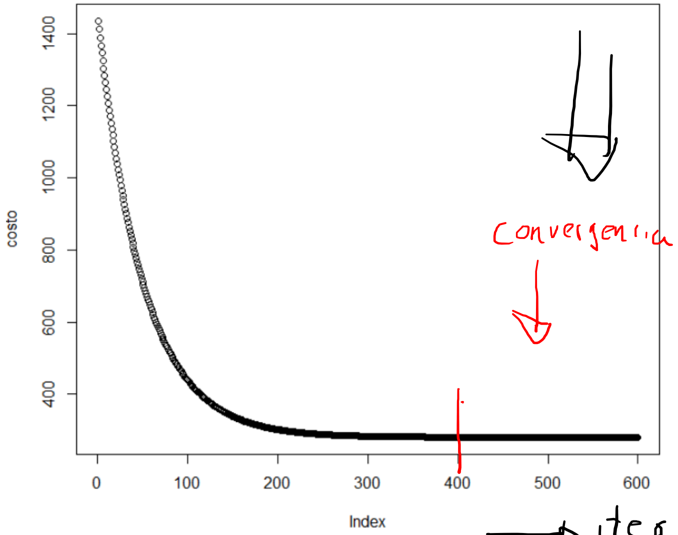
todo el data set





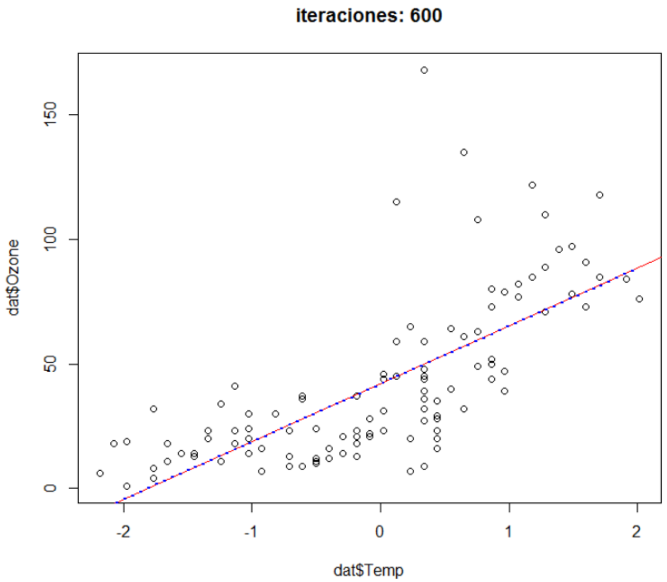
$J(\beta_1, \beta_2)$

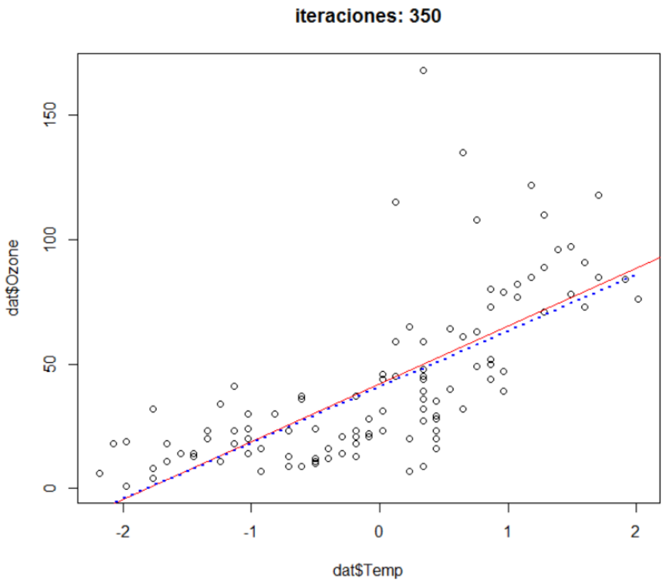
$J_i - J_{i+1} < 10^{-3}$



convergencia

iter





(SGD)

Stochastic Gradient descent

Gradient descent:

Repetir hasta converger{

$$\boldsymbol{\beta} = \boldsymbol{\beta} - \alpha \frac{1}{n} \boldsymbol{X}^T (\hat{\boldsymbol{y}} - \boldsymbol{y})$$

}

Stochastic Gradient descent:

1 muestra

Repetir hasta converger{

$$\boldsymbol{\beta} = \boldsymbol{\beta} - \alpha x_i (\hat{y}_i - y_i)$$

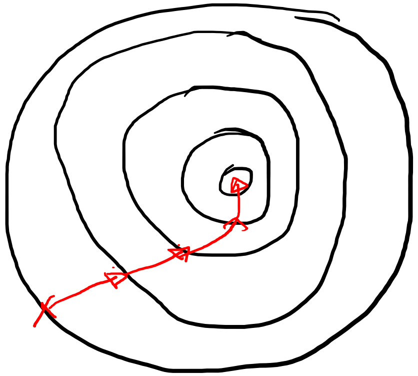
}

fila

rápido

K muestras

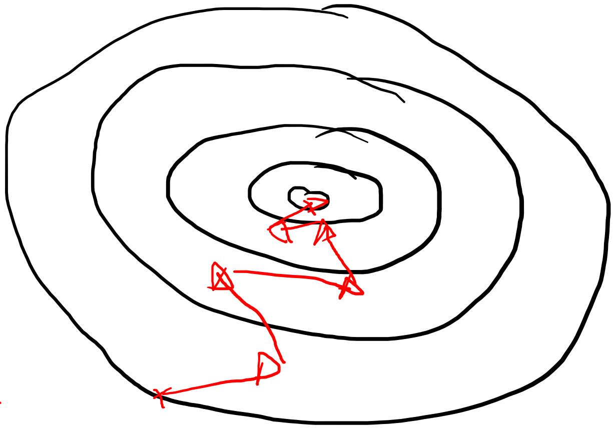
Batch - SGD

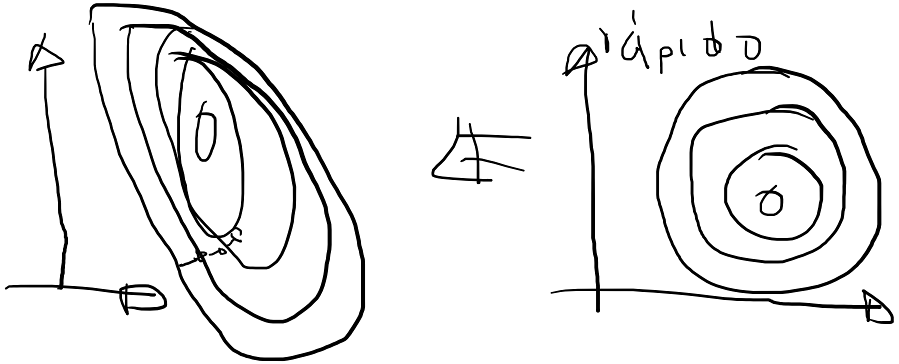


$$\beta = \beta - \frac{\alpha}{K} X^{*T} (\hat{y}_i - y_i)$$

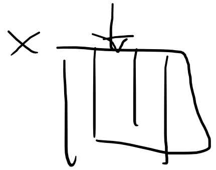
X^* -> batch

dim => $K \times (p+1)$





Consejos prácticos - I



- Revisar que las variables tengan una escala similar
  - Usualmente entre [-1, 1], e.g. normalizar

$$\frac{x-\mu}{\sigma}$$

atributos

X_1 [10,000 , -10,000]

X_2 [0.5 , 2]

X_3 [-3 , 10]

  - Convergencia más rápida

[-1, 1] ✓

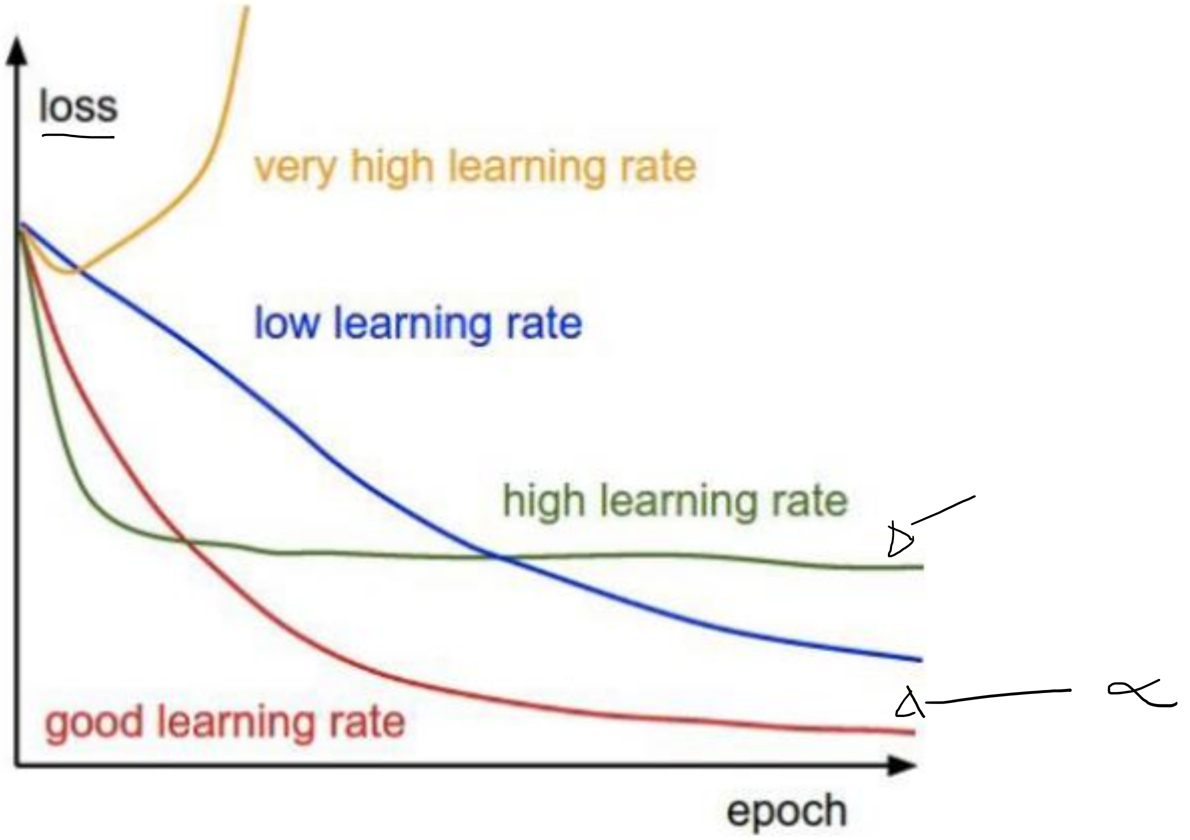
[0, 3] ✓

[-1, 2] ✓

- ¿Qué valor de α escoger?
  - .001, , 0.01, 0.03 , 0.1, , 1 …

Consejos prácticos - II

- ¿Cómo asegurarnos que gradient descent esta trabajando correctamente?



Podemos detener el algoritmo automáticamente:

Decidimos que el algoritmo ha convergido si de una iteración a otra $J(\boldsymbol{\beta})$ tiene un cambio menos a 10^{-3}.