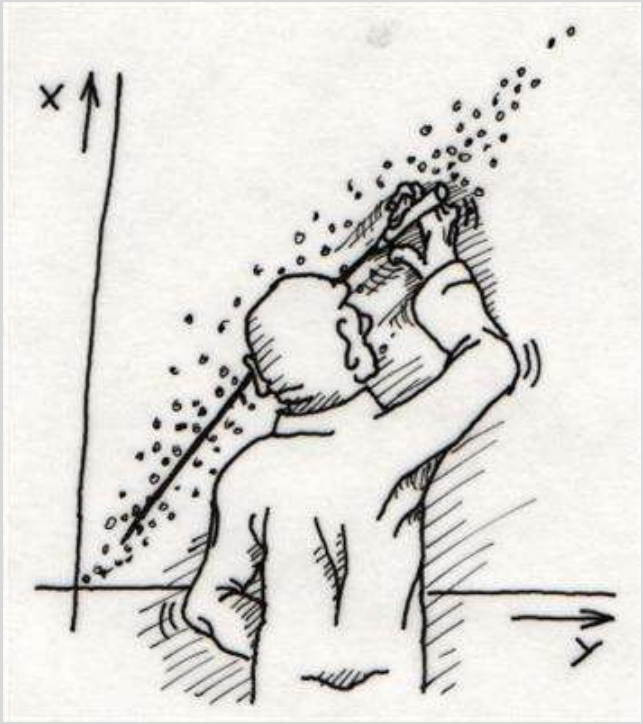



DigitalHouse > Coding School

# DATA SCIENCE

MÓDULO 3

Supuestos de la regresión lineal y análisis de residuos.

Descenso del gradiente

Normalización de variables.

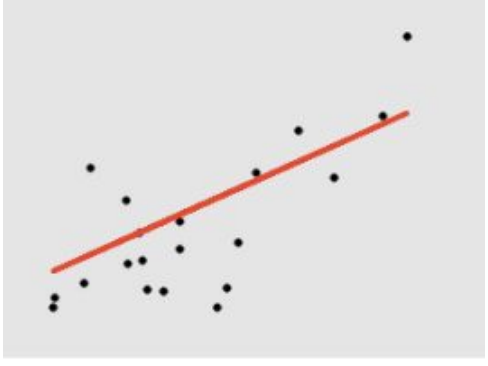
# OBJETIVOS DE LA CLASE

1. Exponer los supuestos de Gauss Markov del modelo lineal y técnicas de análisis de residuos

2. Introducir el algoritmo de descenso del gradiente

3. Entender el concepto y utilidad de la normalización de datos

4. Usar el módulo de preprocesamiento de scikit-learn para normalizar datos



# Regresión Lineal: supuestos de Gauss-Markov y análisis de residuos

La **regresión lineal simple** intenta predecir una respuesta cuantitativa **Y** en base a una única variable predictora **X.** Asume que hay aproximadamente una relación lineal entre X e Y.

$$Y \approx \beta_0 + \beta_1 X.$$

$\beta_0$ y $\beta_1$ son dos constantes que representan el intercepto y la pendiente en el modelo lineal. Juntos, $\beta_0$ y $\beta_1$ son conocidos como los **parámetros** del modelo.
Una vez que hemos usado nuestro set de entrenamiento para producir los estimadores $\hat{\beta}_0$ y $\hat{\beta}_1$ para los coeficientes del modelo, podemos predecir futuras valores de la variable **Y.**

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x,$$

donde $\hat{y}$ indica una **predicción de Y en base a X.** Aquí usamos un símbolo ^ para denotar el valor estimado para un parámetro o coeficiente desconocido, o para denotar el valor predicho de la respuesta.

En lugar de ajustar un modelo distinto de regresión simple para cada predictor, una mejor aproximación es extender el modelo de regresión simple para que puede incluir **múltiples predictores**:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$

Dados estimadores de los coeficientes de pendiente podemos pronosticar la variable de respuesta para una observación con valores dados de los predictores como:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p.$$

Elegimos los valores para los estimadores de los coeficientes que minimizan la suma de residuos al cuadrado:

$$\begin{aligned} \text{RSS} &= \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip})^2. \end{aligned}$$

Bajo ciertas condiciones, conocidas como los **supuestos de Gauss - Markov**, los coeficientes de la regresión son lineales, insesgados y tienen varianza mínima.

1. El modelo es **lineal en los parámetros**.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$

2. Los estimadores de los parámetros poblacionales se estiman a partir de una **muestra aleatoria**.

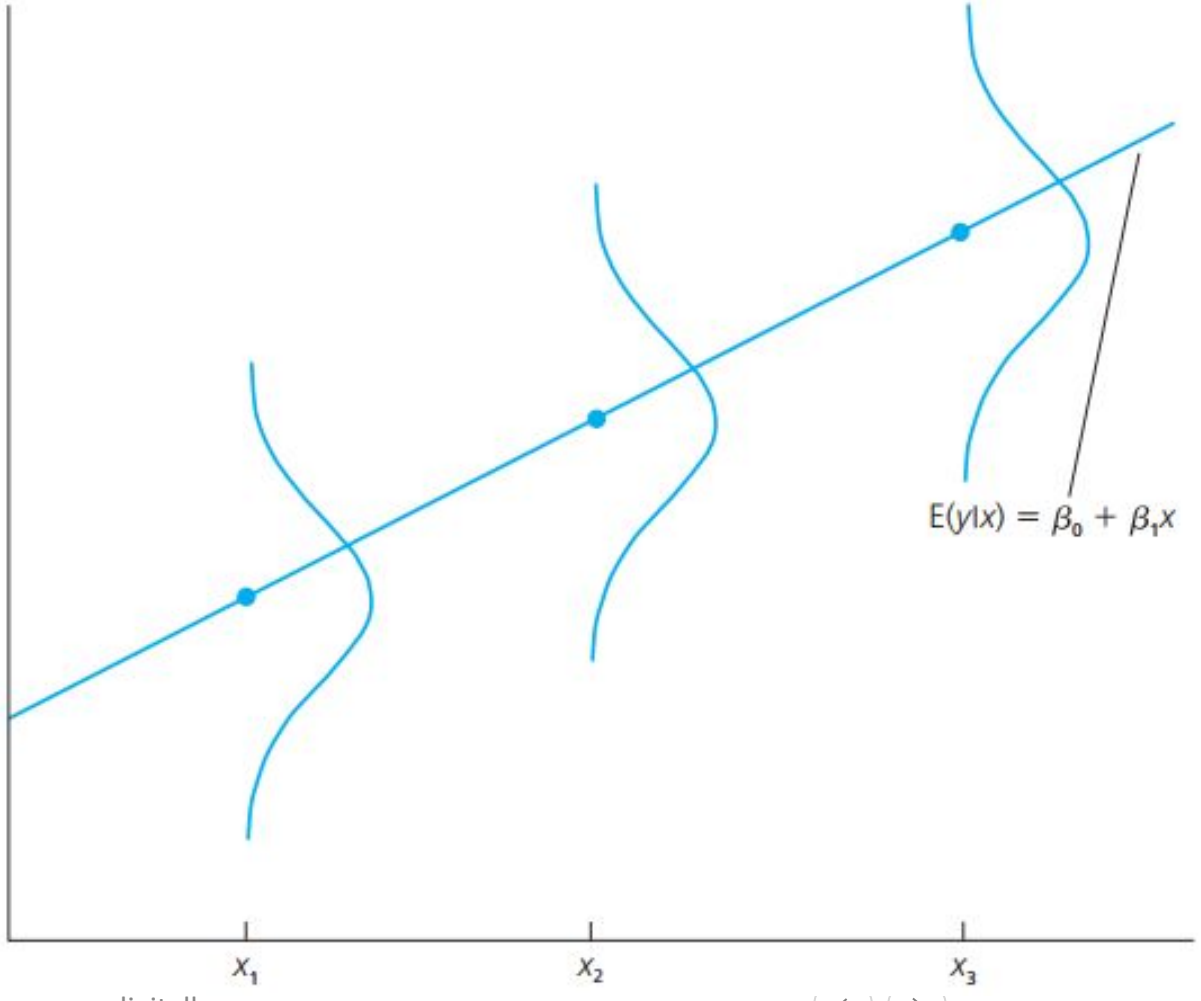
3. **No hay colinealidad perfecta** entre las variables explicativas.

4. El valor **esperado del error es 0** para cualquier valor de la variable explicativa.
5. Para cualquier valor de la variable explicativa, el error tienen la misma varianza (**homocedasticidad**).
6. El error es independiente de las variables explicativas y se distribuye normalmente.

$$\epsilon \sim \mathcal{N}(0, \sigma_\epsilon)$$

7. No existe autocorrelación entre los errores de dos observaciones diferentes condicionadas a X.

$$Cov(\varepsilon_i, \varepsilon_h | X) = 0$$




**Homocedasticidad + Media Condicional igual a 0.**

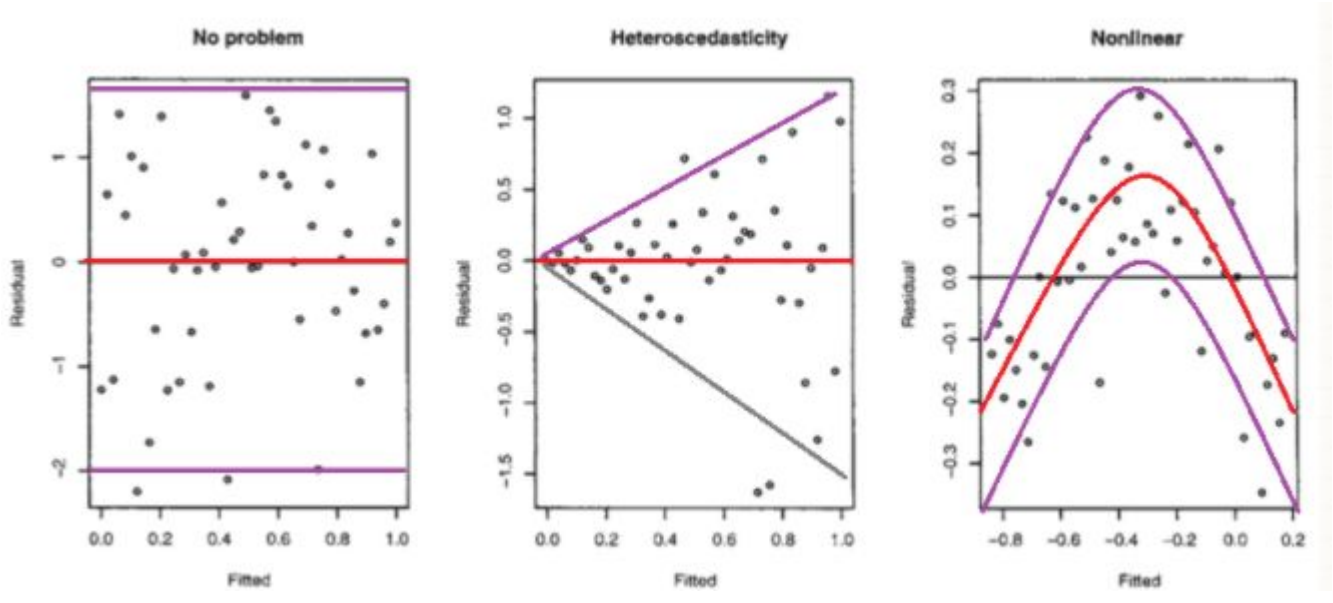
**La consecuencia de estos dos supuestos, es que para cada valor de X, los errores se distribuyen con media cero y desvío estándar constante (σ)**

$$E(y|x) = \beta_0 + \beta_1 x.$$

$$Var(y|x) = \sigma^2.$$

Una buena forma de evaluar el cumplimiento de los supuestos sobre los residuos es graficar los mismos contra los valores predichos.

- En el primer caso los residuos tienen la misma media y varianza para todos los valores de ŷ.
- En el segundo caso, la varianza de los residuos aumenta con ŷ
- En el último caso la varianza parece constante, pero los residuos tienden a ser negativos para valores muy bajos o muy altos de ŷ y positivos para valores intermedios.



# Entrenamiento de modelos y descenso del gradiente

Vamos a ver ahora cómo se entrena efectivamente un modelo de regresión lineal.

Expresemos nuevamente la predicción de un modelo de regresión lineal múltiple:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

donde $\hat{y}$ indica una **predicción de Y en base a X.**

Hands-On Machine Learning with Scikit-Learn and TensorFlow by Aurélien Géron

Si expresamos la predicción del modelo en forma vectorizada,

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \cdot \mathbf{x}$$

el error que vamos a querer minimizar, es decir, nuestra función de costo, es la siguiente:

$$\text{MSE}\,(\mathbf{X}, h_{\boldsymbol{\theta}}) = \frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)})^2$$

Hands-On Machine Learning with Scikit-Learn and TensorFlow by Aurélien Géron

La regresión lineal tiene una solución analítica, que se obtiene minimizando MSE en los parámetros. El resultado es el siguiente:

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\ \mathbf{X}^T\ \mathbf{y}$$

El sistema se llama *sistema de ecuaciones normales* y, si existe solución, se llama ***estimador de mínimos cuadrados ordinarios***.

El problema de estimar nuestros parámetros de esta manera es el costo computacional que implica invertir la matríz $\mathbf{X}^T\mathbf{X}$, del orden de $O(n^3)$

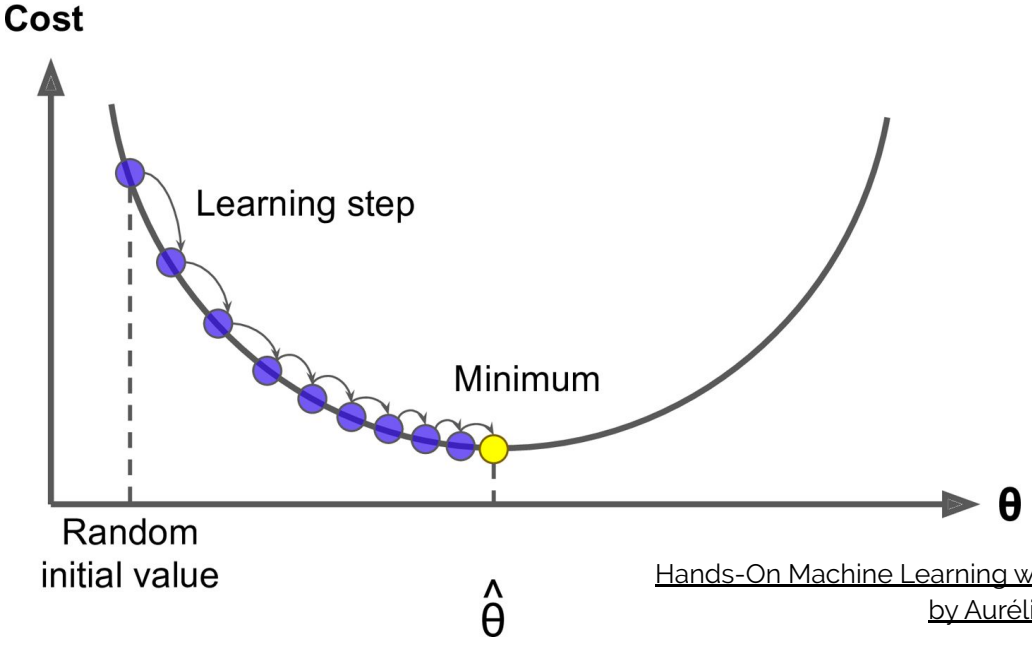
Hands-On Machine Learning with Scikit-Learn and TensorFlow by Aurélien Géron

La clase **`LinearRegression`** de Scikit-Learn no usa el sistema de ecuaciones normales.

En cambio, estima los parámetros usando una ***pseudoinversa*** que se obtiene a partir del método de descomposición de matrices ***Singular Value Decomposition*** (similar al que vimos para PCA, pero más general, no requiere que la matriz sea simétrica).

El costo computacional de este método es del orden de $O(n^2)$

Cuando nuestro modelo tiene muchas variables, muchas observaciones o cuando su función de costo no puede resolverse de forma analítica (como es el caso de la regresión logística y las redes neuronales), podemos entrenar nuestro modelo usando el algoritmo del **descenso del gradiente**.




Hands-On Machine Learning with Scikit-Learn and TensorFlow by Aurélien Géron

**Cost Function**

$$J\left(\Theta_0, \Theta_1\right) = \frac{1}{2m} \sum_{i=1}^{m} [h_\Theta(x_i) - y_i]^2$$

Predicted Value

True Value

**Gradient Descent**

$$\Theta_j = \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J\left(\Theta_0, \Theta_1\right)$$

Learning Rate

Now,

$$\frac{\partial}{\partial \Theta} J_\Theta = \frac{\partial}{\partial \Theta} \frac{1}{2m} \sum_{i=1}^{m} [h_\Theta(x_i) - y]^2$$

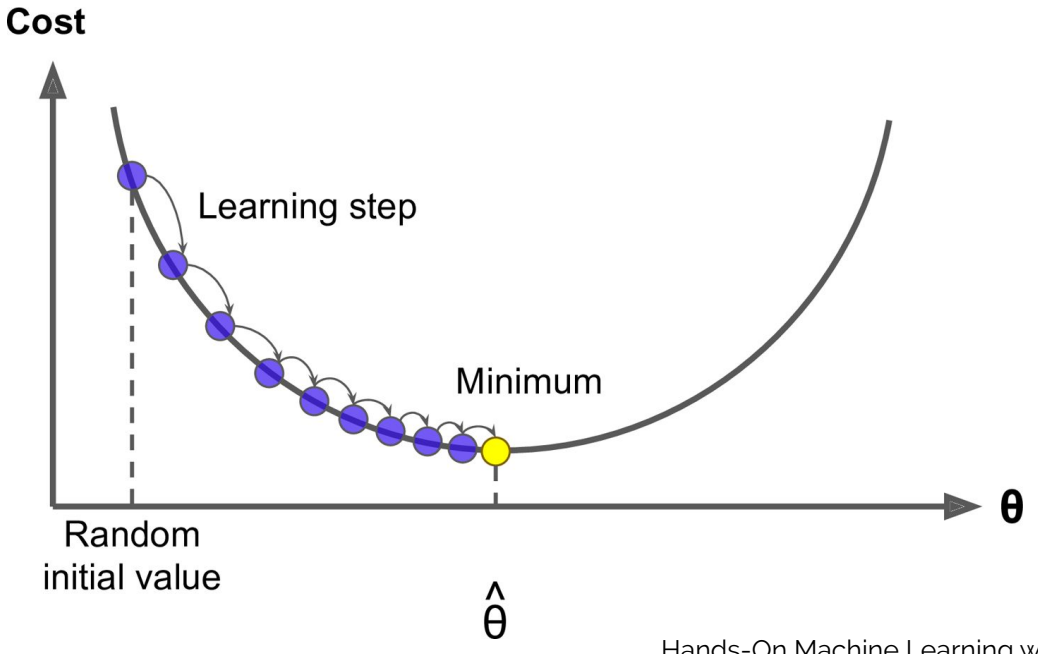
$$= \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x_i) - y) \frac{\partial}{\partial \Theta_j} (\Theta x_i - y)$$

$$= \frac{1}{m} (h_\Theta(x_i) - y) x_i$$

Therefore,

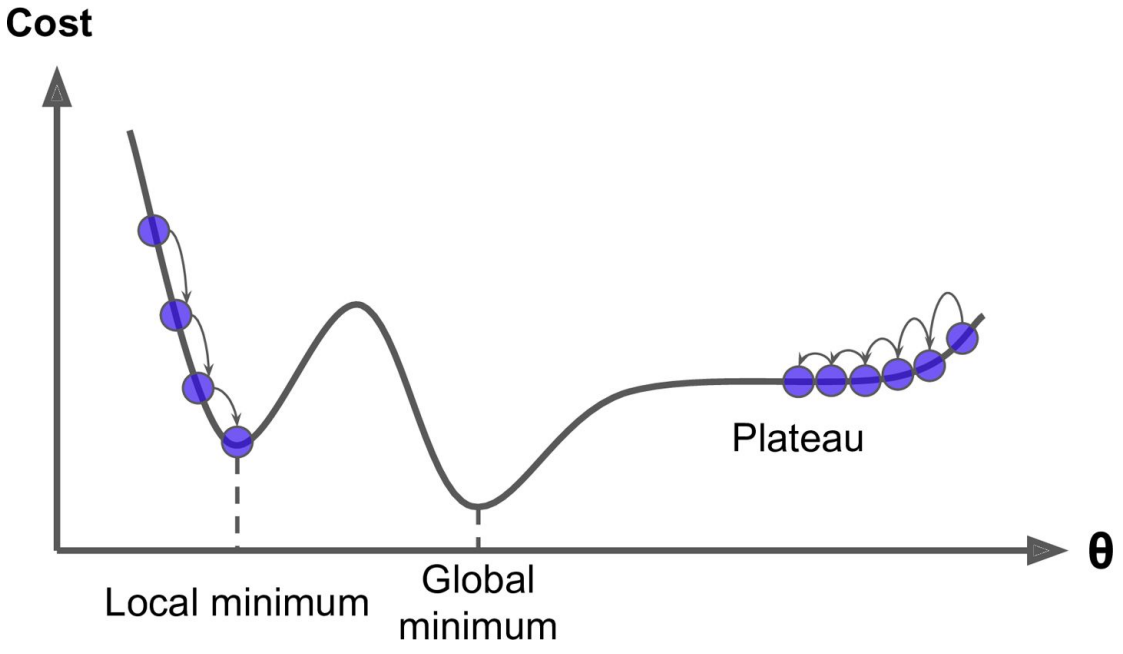
$$\Theta_j := \Theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} [(h_\Theta(x_i) - y) x_i]$$

La función de costo de la regresión lineal es una ***función convexa***, lo que nos asegura que el descenso del gradiente va a converger al **mínimo global**.



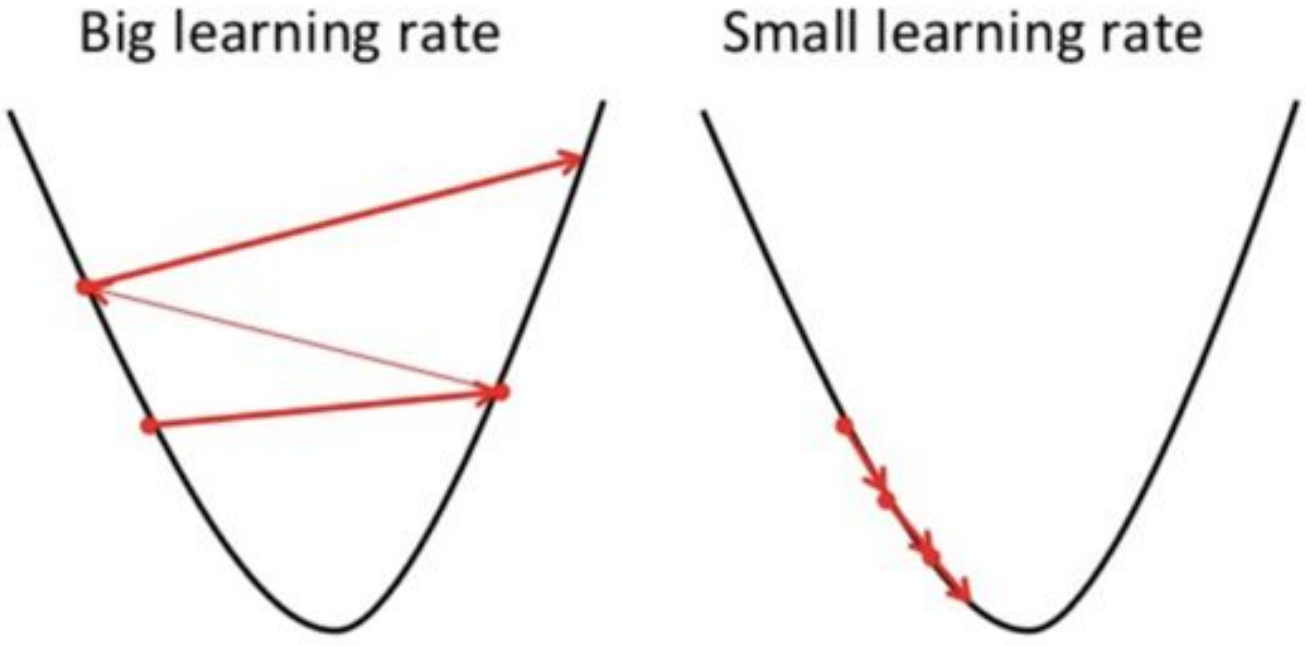

Hands-On Machine Learning with Scikit-Learn and TensorFlow
by Aurélien Géron

Sin embargo, en modelos cuya función de costo no es convexa, podríamos converger a un **mínimo local** o quedarnos trabados en una **meseta**.

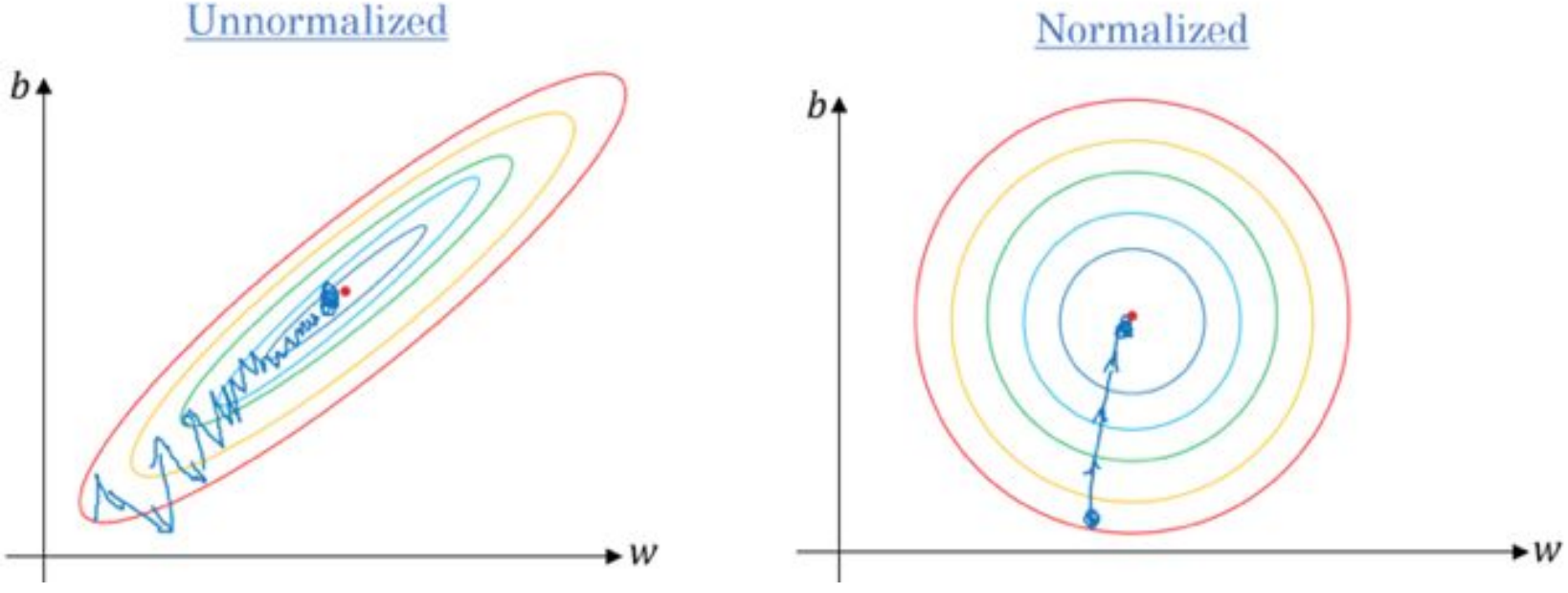



Hands-On Machine Learning with Scikit-Learn and TensorFlow by Aurélien Géron

El ***learning rate*** es un hiperparámetro muy importante del descenso del gradiente.



Para reducir el **tiempo de convergencia** del descenso del gradiente, es conveniente que las **variables estén normalizadas**.



# NORMALIZACIÓN

# ¿Por qué normalizar?

- Manejo de cantidades en **diferentes unidades o escalas**
- Muchos **algoritmos** de machine learning toman la normalización como **requerimiento**
- Existen distintas razones por las cuales un algoritmo de ML requiere estandarizar los datos.

Muchos algoritmos de ML se basan en el cálculo de medidas de distancia que se calculan entre todos los puntos en base a distintos features.

La medida de distancia que viene implementada por default, es la distancia euclídea que requiere matemáticamente que todos los features sean numéricos.

Además, para no favorecer a ningún feature en particular a la hora de explicar la distancia, tenemos que estandarizar y deshacernos de las unidades.

## ¿Cómo normalizar?

Existen algunas formas típicas de normalizar:

- La ***estandarización***: $x_{norm} = (x - \mu) / \sigma$
- La normalización ***min-max:*** $x_{norm} = (x - min) / (max - min)$

La elección entre min-max y estandarización depende del objetivo del método:

- **Min-max**: Tiene sentido en los casos donde importa que los features tengan las mismas unidades pero no necesariamente la misma varianza

- **Estandarización:** Tiene sentido donde se necesita que los features tengan las mismas unidades y también la misma varianza, como por ejemplo en componentes principales.

# Conclusión

**Usamos la normalización para:**

- Manejo de cantidades en **diferentes unidades o escalas**
- Muchos **algoritmos** de machine learning toman la normalización como **requerimiento**
- Puede **aumentar la velocidad de convergencia** usando el método de gradiente

**Existen diferentes métodos de normalización**, como la estandarización, min-max y L1 y L2.

# Práctica Guiada