Redes Neuronales (2025)

Ajuste de datos por mínimos cuadrados: El Método del Descenso por el Gradiente
(Trabajo en progreso)*

Índice

*Reportar errores a: tristan.osan@unc.edu.ar

1. Introducción: ¿Qué es el ajuste de datos?

En ciencia e ingeniería, a menudo recolectamos datos experimentales que relacionan dos o más variables. Por ejemplo, podríamos medir la elongación de un resorte (x) a medida que aplicamos diferentes fuerzas (F). Generalmente, esperamos que estos datos sigan una ley física o un modelo matemático (En este caso, $|F| = kx$ (Ley de Hooke), donde k representa la constante del resorte).

El **ajuste de datos** (o regresión) es el proceso de encontrar la curva o función matemática que mejor se aproxima a una serie de puntos de datos. El **método de mínimos cuadrados** es la técnica más común para lograr este objetivo.

Nuestro objetivo es simple: dados n datos de la forma $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ (pares ordenados), queremos encontrar la línea recta

$$y = mx + b$$

que pase "lo más cerca posible" de todos los puntos a la vez. Las incógnitas que debemos encontrar son la pendiente m y la ordenada al origen b.

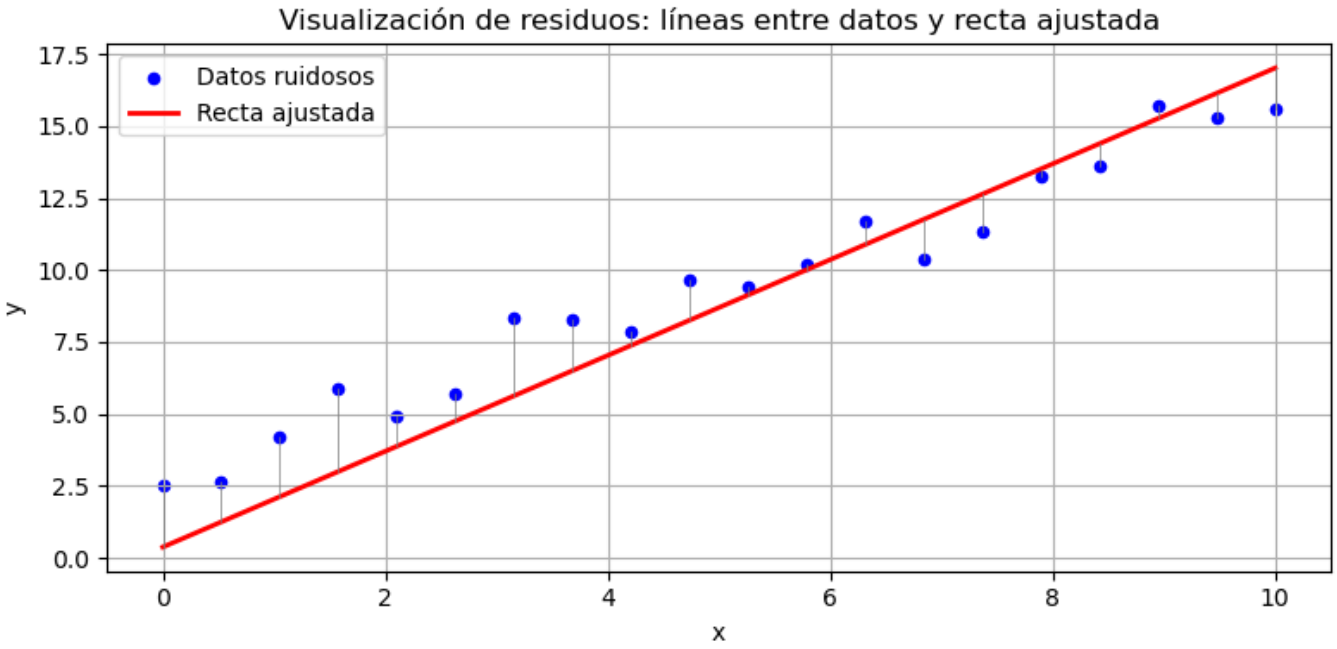

Figura 1: Visualización de los residuos de un ajuste lineal. Las líneas verticales grises conectan cada punto observado con su correspondiente valor ajustado sobre la recta, representando gráficamente el error individual $e_i = (y_i - \hat{y}_i)$.

2. El Criterio de "Mejor Ajuste"

¿Cómo definimos matemáticamente "lo más cerca posible"? Para cada punto de dato (x_i, y_i), la recta propuesta nos da un valor predicho $\hat{y}_i = mx_i + b$. La diferencia vertical entre el valor real y_i y el valor predicho $\hat{y}_i$ se llama **residuo** o error, e_i.

$$e_i = (y_i - \hat{y}_i) = (y_i - (mx_i + b)).$$

Podríamos intentar minimizar la suma de estos residuos, $\sum e_i$. Sin embargo, algunos residuos serán positivos (puntos por encima de la recta) y otros negativos (puntos por debajo), y podrían cancelarse entre sí, dándonos un mal ajuste.

Para evitar esto, minimizamos la **suma de los cuadrados de los residuos**, J. Esta es la idea central del método.

$$J = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - (mx_i + b))^2$$

La función J depende de los parámetros que queremos encontrar, m y b. Es decir, $J = J(m, b)$. Nuestro trabajo es encontrar los valores de m y b que hacen que J sea lo más pequeña posible.

La función J, en general, recibe nombres tales como: "Error Cuadrático Medio (ECM)", "Función Costo" o "Pérdida" ("loss", en inglés).

3. Deducción matemática del método para el caso lineal

Para encontrar el mínimo de una función de varias variables como $J(m, b)$, utilizamos el cálculo. El mínimo se encontrará en el punto donde las derivadas parciales de J con respecto a cada una de sus variables (m y b) son iguales a cero.

$$\frac{\partial J}{\partial m} = 0 \quad \text{y} \quad \frac{\partial J}{\partial b} = 0$$

3.1. Derivada con respecto a b

Calculemos la primera derivada parcial. Usamos la regla de la cadena, tratando a m como una constante.

$$\begin{aligned}
\frac{\partial J}{\partial b} &= \frac{\partial}{\partial b} \sum_{i=1}^{n} (y_i - mx_i - b)^2 \\
&= \sum_{i=1}^{n} 2(y_i - mx_i - b) \cdot \frac{\partial}{\partial b}(y_i - mx_i - b) \\
&= \sum_{i=1}^{n} 2(y_i - mx_i - b) \cdot (-1) \\
&= -2 \sum_{i=1}^{n} (y_i - mx_i - b)
\end{aligned}$$

Igualando a cero:

$$-2 \sum_{i=1}^{n} (y_i - mx_i - b) = 0$$

$$\sum y_i - m \sum x_i - \sum b = 0$$

Como la suma de b, n veces, es nb:

$$\sum y_i - m \sum x_i - nb = 0$$

Reordenando, obtenemos nuestra primera **ecuación normal**:

$$\left(\sum_{i=1}^{n} x_i \right) m + nb = \sum_{i=1}^{n} y_i \tag{1}$$

3.2. Derivada con respecto a m

Ahora derivamos con respecto a m, tratando a b como una constante.

$$\begin{aligned}\frac{\partial J}{\partial m} &= \frac{\partial}{\partial m}\sum_{i=1}^{n}(y_i - mx_i - b)^2 \\ &= \sum_{i=1}^{n} 2(y_i - mx_i - b)\cdot\frac{\partial}{\partial m}(y_i - mx_i - b) \\ &= \sum_{i=1}^{n} 2(y_i - mx_i - b)\cdot(-x_i) \\ &= -2\sum_{i=1}^{n} x_i(y_i - mx_i - b)\end{aligned}$$

Igualando a cero:

$$-2\sum_{i=1}^{n}(x_i y_i - mx_i^2 - bx_i) = 0$$

$$\sum x_i y_i - m\sum x_i^2 - b\sum x_i = 0$$

Reordenando, obtenemos nuestra segunda **ecuación normal**:

$$\left(\sum_{i=1}^{n} x_i^2\right) m + \left(\sum_{i=1}^{n} x_i\right) b = \sum_{i=1}^{n} x_i y_i \tag{2}$$

3.3. Solución del Sistema de Ecuaciones

Ahora tenemos un sistema de dos ecuaciones lineales (1) y (2) con dos incógnitas, m y b. Los coeficientes de este sistema ($\sum x_i$, $\sum y_i$, etc.) se pueden calcular directamente a partir de nuestros datos.

Resolviendo este sistema (por ejemplo, usando sustitución o la regla de Cramer), llegamos a las siguientes fórmulas para m y b:

$$m = \frac{n\left(\sum x_i y_i\right) - \left(\sum x_i\right)\left(\sum y_i\right)}{n\left(\sum x_i^2\right) - \left(\sum x_i\right)^2}$$

Y una vez que tenemos m, es más fácil calcular b a partir de la Ecuación (1) despejada, usando los promedios $\bar{x} = \frac{\sum x_i}{n}$ y $\bar{y} = \frac{\sum y_i}{n}$:

$$b = \bar{y} - m\bar{x}$$

Estas resultan ser las fórmulas que nos dan la pendiente y la ordenada al origen de la recta que mejor ajusta el conjunto de datos.

3.4. Ejemplo Práctico

Supongamos que tenemos los siguientes 4 puntos de datos ($n = 4$): **(1, 3), (2, 4), (3, 6), (4, 8)**

Paso 1: Calcular las sumatorias necesarias. Organizamos los cálculos en una tabla para mayor claridad.

x_i	y_i	x_i^2	$x_i y_i$
1	3	1	3
2	4	4	8
3	6	9	18
4	8	16	32
$\sum x_i = 10$	$\sum y_i = 21$	$\sum x_i^2 = 30$	$\sum x_i y_i = 61$

Paso 2: Calcular la pendiente m. Usamos la fórmula con $n = 4$ y las sumas de la tabla.

$$m = \frac{4(61) - (10)(21)}{4(30) - (10)^2} = \frac{244 - 210}{120 - 100} = \frac{34}{20} = 1{,}7$$

Paso 3: Calcular la ordenada al origen b. Primero, calculamos los promedios.

$$\bar{x} = \frac{10}{4} = 2{,}5 \qquad \bar{y} = \frac{21}{4} = 5{,}25$$

Ahora usamos la fórmula para b.

$$b = 5{,}25 - (1{,}7)(2{,}5) = 5{,}25 - 4{,}25 = 1{,}0$$

Paso 4: Escribir la ecuación de la recta. La recta de mejor ajuste por mínimos cuadrados es:

$$\boxed{y = 1{,}7x + 1{,}0}$$

Esta es la recta que minimiza la suma de las distancias verticales, elevadas al cuadrado, para el conjunto de datos proporcionado, es decir, **minimiza** el ECM dado por la función $J(m, b)$.

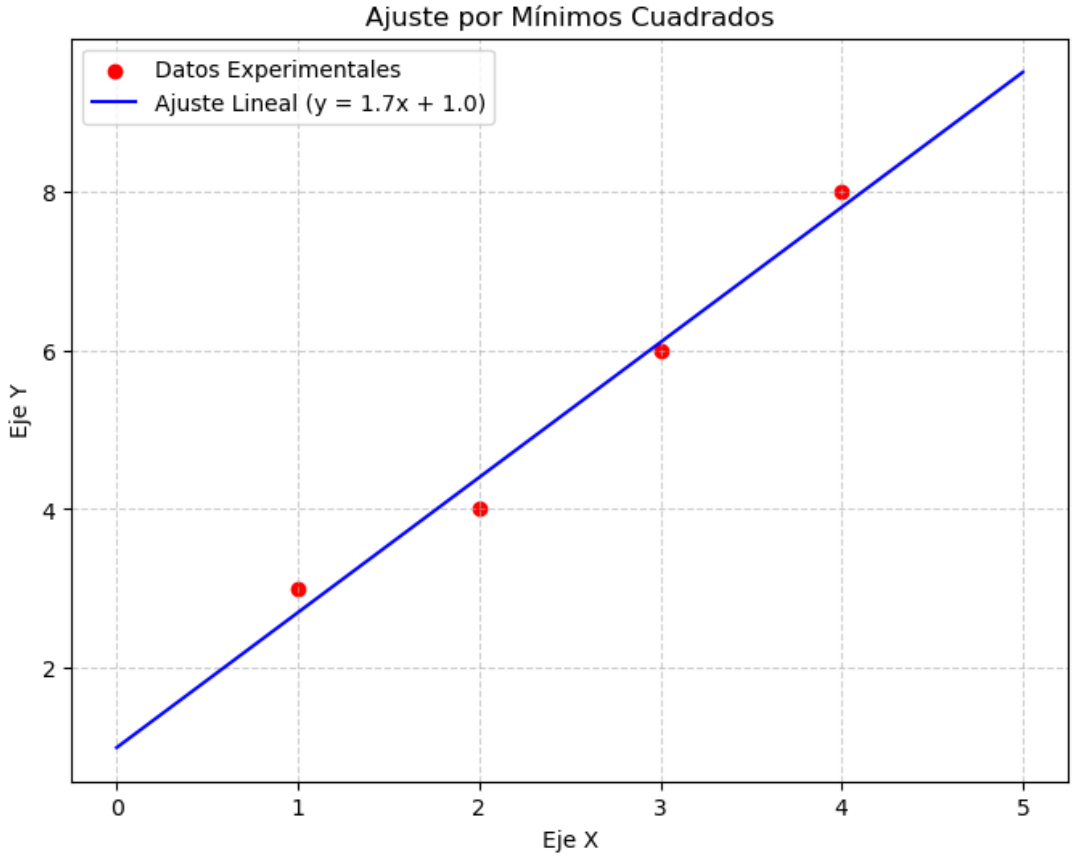

Figura 2: Visualización del ajuste por mínimos cuadrados para el caso lineal.

3.5. Observaciones

El método de **mínimos cuadrados** nos proporciona una forma robusta y determinista de encontrar la función lineal que mejor se ajusta a un conjunto de datos. La deducción se basa en un principio simple: **minimizar la suma de los errores al cuadrado**, lo que nos lleva a un sistema de ecuaciones lineales cuyas soluciones son los parámetros del modelo. Este mismo principio puede extenderse a ajustes más complejos, como polinomios (regresión polinómica) o funciones no lineales.

4. Interpretación geométrica del gradiente

Dada una función escalar $f : \mathbb{R}^n \to \mathbb{R}$, el **gradiente** $\nabla f(\mathbf{x})$ en un punto $\mathbf{x} \in \mathbb{R}^n$ es un vector que apunta en la dirección de *máximo crecimiento de la función*.

Por ejemplo, el **gradiente** $\nabla f(\mathbf{x})$ de una función $f : \mathbb{R}^2 \to \mathbb{R}$, en un punto $\mathbf{x} = (x, y)^\top \in \mathbb{R}^2$, está dado por:

$$\mathbf{x} = (x, y)^\top$$

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right)^\top$$

Geométricamente, este vector es ortogonal a las *superficies de nivel* de f, es decir, a los conjuntos donde $f(\mathbf{x}) = \textit{constante}$. Además, la magnitud $\|\nabla f(\mathbf{x})\|$ indica la *tasa de cambio más grande* de f en ese punto. En el contexto del *Método de Descenso por el Gradiente*, moverse en el *sentido opuesto* al gradiente permite reducir el valor de la función de costo J de manera eficiente, siguiendo el camino más empinado hacia un mínimo local.

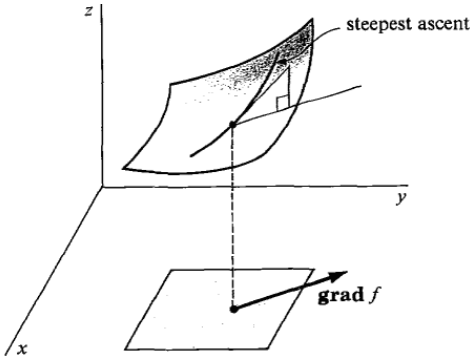

Figura 3: Visualización geométrica del gradiente de una función f en un punto dado. Se observa que el gradiente apunta en la *dirección de máximo crecimiento* de la función $f(x, y)$.

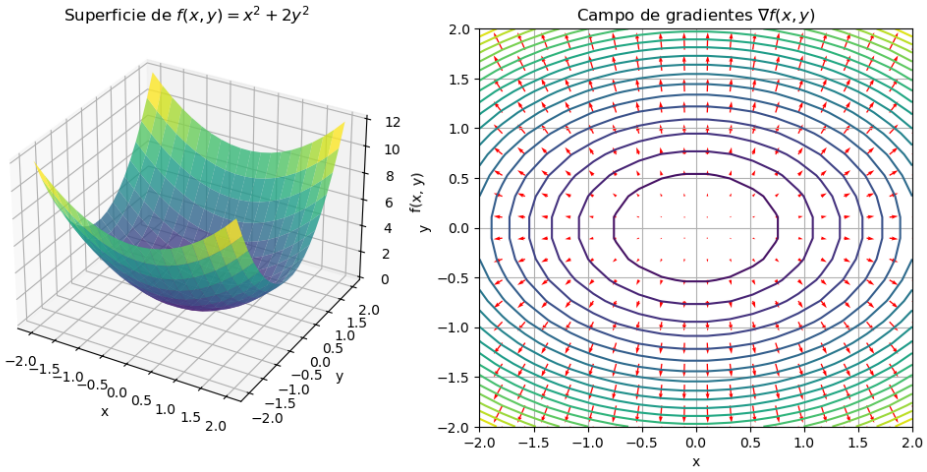

Figura 4: Visualización geométrica del gradiente en el paraboloide $f(x, y) = x^2 + 2y^2$. **Izquierda:** superficie 3D de la función. **Derecha:** campo de gradientes sobre el plano xy, donde los vectores apuntan en la *dirección de máximo crecimiento* de la función $f(x, y)$. Nótese que la curvatura en y es mayor, lo que se refleja en la magnitud de los vectores.

5. Método del descenso por el gradiente: Modelo lineal con 2 parámetros ajustables

5.1. Planteamiento del problema

Dado un conjunto de datos $\{(x_i, y_i)\}_{i=1}^{n}$, queremos encontrar una función $f(x;\boldsymbol{\theta})$ parametrizada por el vector $\boldsymbol{\theta}$ que se ajuste lo mejor posible a los datos. En el caso más simple, consideramos un modelo lineal $\boldsymbol{\theta} = [\theta_0, \theta_1]^\top$:

$$f(x;\boldsymbol{\theta}) = \theta_0 + \theta_1 x = \hat{y}$$

El objetivo es minimizar el valor de la función $J(\boldsymbol{\theta})$, definida como:

$$J(\boldsymbol{\theta}) = J(\theta_0, \theta_1) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{3}$$

$$J(\boldsymbol{\theta}) = J(\theta_0, \theta_1) = \frac{1}{2n}\sum_{i=1}^{n}((\theta_0 + \theta_1 x_i) - y_i)^2 \tag{4}$$

- El factor $\frac{1}{2n}$ facilita las derivadas y normaliza por el número de muestras.
- Como ya mencionamos anteriormente, a la función $J(\boldsymbol{\theta})$ se la denomina “Error Cuadrático Medio (ECM)”, “Función Costo” o “Pérdida” (“loss”, en inglés).

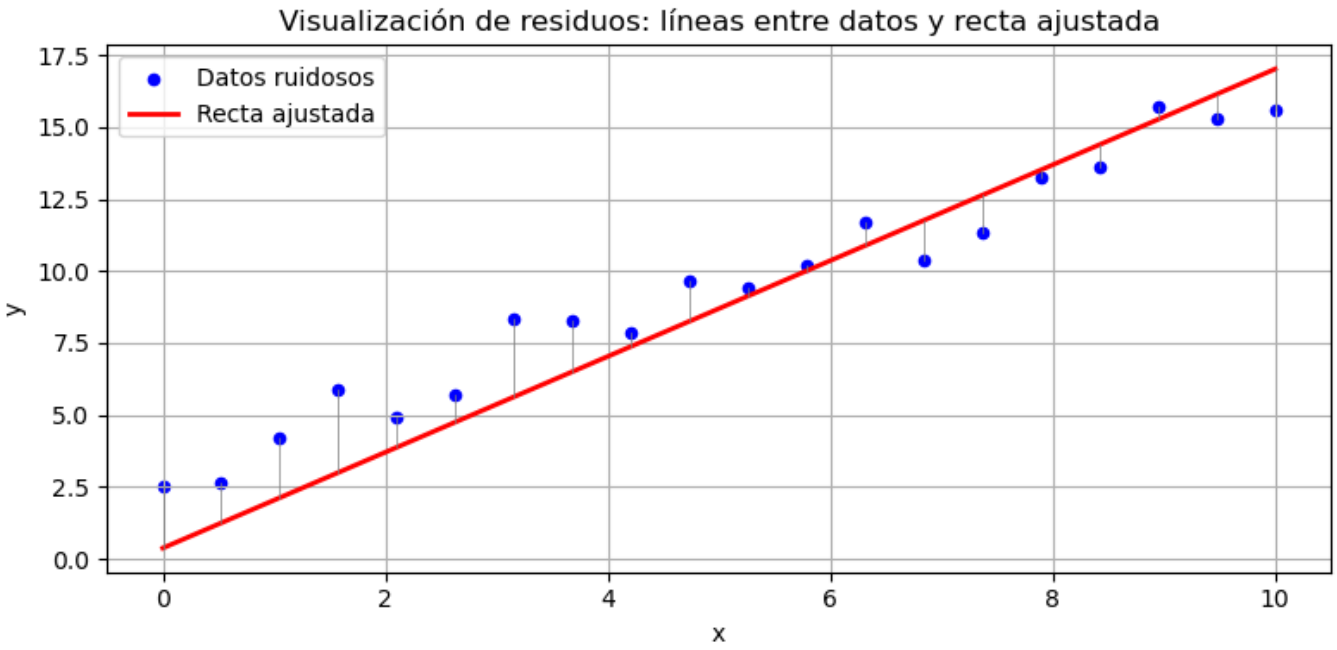

Figura 5: Visualización de los residuos de un ajuste lineal. Las líneas verticales grises conectan cada punto observado con su correspondiente valor ajustado sobre la recta, representando gráficamente el error individual $e_i = (y_i - \hat{y}_i)$.

5.2. Significado del gradiente

El vector gradiente $\nabla J(\boldsymbol{\theta}) = [\partial J/\partial\theta_0, \partial J/\partial\theta_1]^\top$ indica la dirección de *máximo crecimiento* de la función $J(\boldsymbol{\theta})$ correspondiente a ese conjunto particular de valores de los parámetros ($\boldsymbol{\theta} = [\theta_0, \theta_1]^\top$). Para *disminuir* el valor de $J(\boldsymbol{\theta})$, avanzamos en el *sentido contrario* al gradiente de $J(\boldsymbol{\theta})$ correspondiente a ese conjunto particular de valores de los parámetros $\boldsymbol{\theta}$.

5.3. Interpretación geométrica del Método de Descenso por el Gradiente

El método de mínimos cuadrados busca la recta que minimiza la suma de los cuadrados de las distancias verticales entre los puntos observados y la recta ajustada. Este criterio penaliza fuertemente los errores grandes y tiene una interpretación probabilística si se asume ruido gaussiano.

5.4. Método del Descenso por el Gradiente

Para minimizar $J(\boldsymbol{\theta})$, usamos el método de descenso por el gradiente. La idea es actualizar los parámetros en la dirección opuesta al gradiente:

$$\theta_j^{(k+1)} = \theta_j^{(k)} - \eta \frac{\partial J}{\partial \theta_j}$$

donde η es la tasa de aprendizaje y k representa el número de iteración del algoritmo.

5.4.1. Derivación de los gradientes

Definamos el error $e_i = \hat{y}_i - y_i = \theta_0 + \theta_1 x_i - y_i$. Entonces

$$J(\theta_0, \theta_1) = \frac{1}{2n} \sum_{i=1}^{n} e_i^2.$$

Usando regla de la cadena y linealidad:

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{n} \sum_{i=1}^{n} e_i \cdot \frac{\partial e_i}{\partial \theta_0} = \frac{1}{n} \sum_{i=1}^{n} e_i \cdot 1 = \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + \theta_1 x_i - y_i).$$

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{n} \sum_{i=1}^{n} e_i \cdot \frac{\partial e_i}{\partial \theta_1} = \frac{1}{n} \sum_{i=1}^{n} e_i \cdot x_i = \frac{1}{n} \sum_{i=1}^{n} x_i (\theta_0 + \theta_1 x_i - y_i).$$

Estas expresiones son las que se emplean para el descenso por gradiente.

5.4.2. Descenso por el gradiente

Dada una tasa de aprendizaje $\eta > 0$), la regla de actualización iterativa es

$$\theta_0^{(k+1)} = \theta_0^{(k)} - \eta \left. \frac{\partial J}{\partial \theta_0} \right|_{\theta^{(k)}}, \qquad \theta_1^{(k+1)} = \theta_1^{(k)} - \eta \left. \frac{\partial J}{\partial \theta_1} \right|_{\theta^{(k)}}.$$

5.4.3. Algoritmo conceptual

1. Inicializar θ_0 y θ_1 (por ejemplo, $\theta_0 = 0{,}5$ y $\theta_1 = 1{,}2$, estimados del gráfico de los datos a ser ajustados).
2. Calcular la función costo $J(\boldsymbol{\theta})$ con el valor actual de todos sus parámetros $\boldsymbol{\theta} = [\theta_0, \theta_1]^\top$.
3. Repetir hasta que se cumpla el *criterio de parada* del algoritmo:
   - Calcular el gradiente $\nabla J(\boldsymbol{\theta})$.
   - Actualizar todos los parámetros de $J(\boldsymbol{\theta})$ utilizando la "receta": $\theta \leftarrow \theta - \eta \nabla J$.
4. Devolver los parámetros ajustados.
5. Repetir los pasos 2-4 hasta que se cumpla *criterio de parada* del algoritmo (por ejemplo, se ejecutó el algoritmo el numéro máximo de iteraciones especificado).

5.4.4. Criterios de parada habituales:

- **Iteraciones máximas:** detener tras $N_{\text{máx.}}$ iteraciones.
- **Cambio en la función costo:** parar si

$$\left| \frac{J(\boldsymbol{\theta}^{(k+1)}) - J(\boldsymbol{\theta}^{(k)})}{J(\boldsymbol{\theta}^{(k)})} \right| < \varepsilon.$$

- **Norma del gradiente:** parar si $\|\nabla J(\boldsymbol{\theta}^{(k)})\| < \delta$.
- **Cambio en parámetros:** parar si $\frac{\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}\|}{\|\boldsymbol{\theta}^{(k)}\|} < \alpha$.

5.4.5. Consideraciones prácticas

- La tasa de aprendizaje η debe ser elegida cuidadosamente: si es muy grande, el algoritmo puede divergir; si es muy pequeña, puede tardar mucho en converger.
- El descenso por el gradiente puede extenderse a modelos no lineales y a múltiples variables.
- En la práctica, se usa *descenso por el gradiente estocástico* (SGD) para grandes conjuntos de datos.

5.5. Observaciones

El ajuste por mínimos cuadrados mediante descenso por el gradiente es una técnica fundamental en estadística, física computacional y aprendizaje automático. Su comprensión permite abordar problemas de regresión, clasificación y optimización en contextos científicos y tecnológicos.

5.6. Explicación del programa en Python

El programa simula datos ruidosos de una recta, optimiza θ_0, θ_1 por medio del algoritmo de descenso por el gradiente, y genera los gráficos de la evolución de la función costo $J(\theta_0, \theta_1)$ y la recta de ajuste final junto con los datos.

5.6.1. Generación de datos

- **Dominio:** se crea un vector x uniforme.
- **Recta verdadera:** $y^\star = \theta_0^\star + \theta_1^\star x$.
- **Ruido:** se añade ruido gaussiano para simular mediciones.

5.6.2. Bucle de optimización

- **Predicción:** $\hat{y} = \theta_0 + \theta_1 x$.
- **Costo:** $J = \frac{1}{2n} \sum (\hat{y} - y)^2$, almacenado en cada iteración.
- **Gradientes:** se calculan con las fórmulas obtenidas anteriormente.
- **Actualización:** se actualiza cada uno de los parámetros por medio de la receta de descenso por el gradiente.

5.6.3. Visualizaciones

- **Costo vs. iteración:** permite verificar la disminución y detectar problemas de η.
- **Datos y ajuste final:** muestra la nube de puntos y la recta de ajuste final con los parámetros obtenidos por medio del algoritmo de descenso por el gradiente.

5.7. Código Python completo

```
import numpy as np
import matplotlib.pyplot as plt

# ----------------------------
# 1. Datos simulados (recta con ruido)
# ----------------------------
np.random.seed(42)
n = 20
x = np.linspace(0, 10, n)

# Parámetros verdaderos
true_theta0 = 1.0   # ordenada al origen
true_theta1 = 1.5   # pendiente

# Ruido gaussiano
noise = np.random.normal(0, 1.0, size=n)
y = true_theta0 + true_theta1 * x + noise

# ----------------------------
# 2. Inicialización y hiperparámetros
# ----------------------------
theta0, theta1 = 0.5, 1.2
alpha = 0.01
max_iter = 50

cost_history = []

# ----------------------------
# 3. Descenso por gradiente
# ----------------------------
for k in range(max_iter):
    y_pred = theta0 + theta1 * x
    error = y_pred - y
    cost = (1/(2*n)) * np.sum(error**2)
    cost_history.append(cost)

    # Gradientes (derivados de J)
    dtheta0 = (1/n) * np.sum(error)
    dtheta1 = (1/n) * np.sum(error * x)

    # Actualización
    theta0 -= alpha * dtheta0
    theta1 -= alpha * dtheta1

print(f"Parámetros finales: theta0 = {theta0:.3f}, theta1 = {theta1:.3f}")
```

```
# ------------------------------
# 4. Gráfico de la función de costo
# ------------------------------
plt.figure(figsize=(8,4))
plt.plot(cost_history, marker='o', linestyle='-.', color='red')
plt.xlabel("Iteraciones")
plt.ylabel("Costo J")
plt.title("Evolución de la función de costo (ajuste lineal)")
plt.grid(True)
plt.tight_layout()
plt.show()

# ------------------------------
# 5. Gráfico de datos y ajuste final
# ------------------------------
plt.figure(figsize=(8,4))
plt.scatter(x, y, color='blue', s=20, label="Datos ruidosos")
plt.plot(x, theta0 + theta1 * x, color='red', linewidth=2, label="Recta
    ajustada")
plt.xlabel("x")
plt.ylabel("y")
plt.title("Ajuste lineal por mínimos cuadrados con descenso por gradiente"
    )
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# ------------------------------
# 6. Gráfico de residuos (líneas verticales)
# ------------------------------
y_fit = theta0 + theta1 * x

plt.figure(figsize=(8,4))
plt.scatter(x, y, color='blue', s=20, label="Datos ruidosos")
plt.plot(x, y_fit, color='red', linewidth=2, label="Recta ajustada")

# Dibujar líneas verticales desde cada punto al modelo ajustado
for xi, yi, yfi in zip(x, y, y_fit):
    plt.plot([xi, xi], [yi, yfi], color='gray', linewidth=0.5)

plt.xlabel("x")
plt.ylabel("y")
plt.title("Visualización de residuos: líneas entre datos y recta ajustada"
    )
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

5.8. Resultados esperados

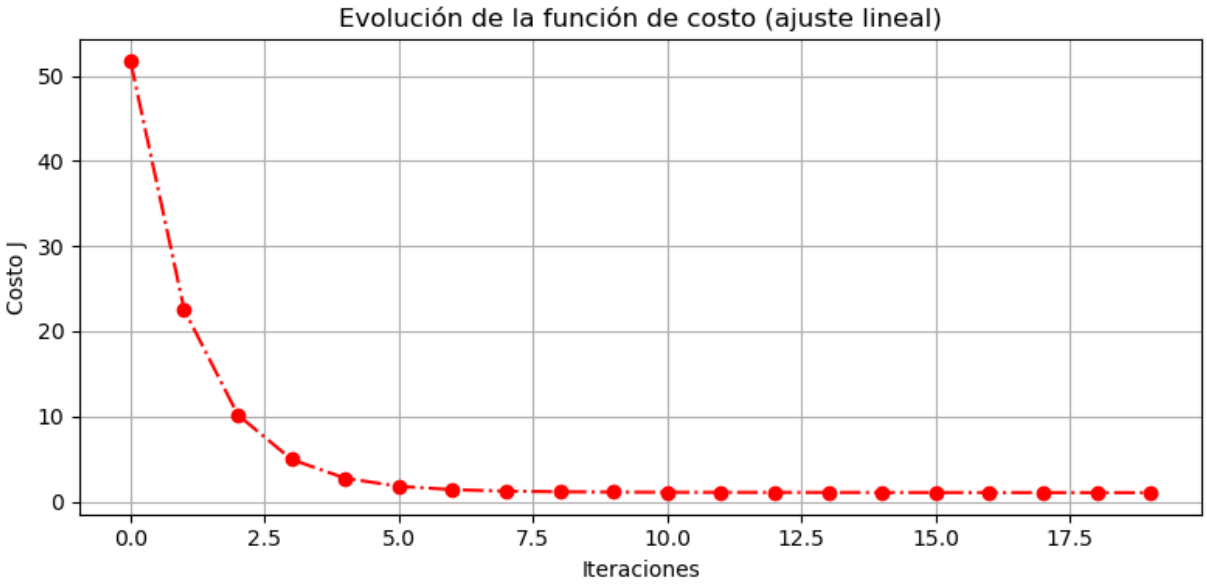

Figura 6: Evolución de la función de costo $J(\theta_0, \theta_1)$ durante el ajuste lineal por descenso por gradiente. Se observa cómo el error disminuye progresivamente a medida que los parámetros se actualizan en cada iteración.

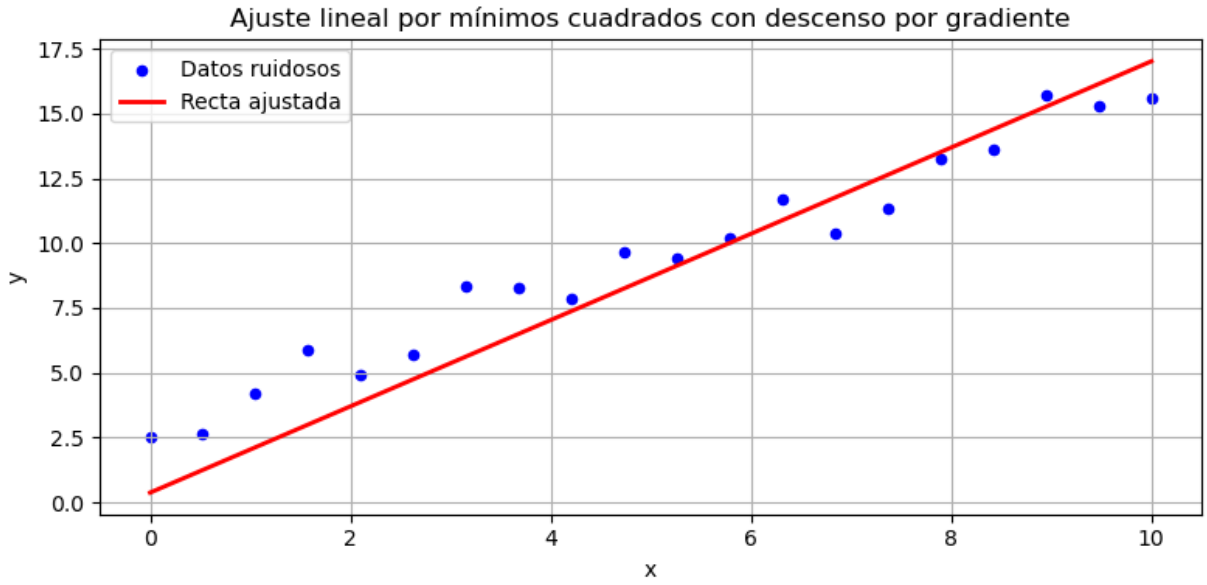

Figura 7: Datos ruidosos simulados (en azul) y recta ajustada final (en rojo) obtenida por descenso por gradiente. La recta representa el modelo $\hat{y} = \theta_0 + \theta_1 x$ con los parámetros optimizados.

6. Método de descenso por el gradiente: Modelo con una función senoidal

El objetivo es ajustar una función seno a un conjunto de datos observados "ruidosos" (es decir, con un cierto error en su determinación), estimando tres parámetros: amplitud A, frecuencia ν y fase ϕ. La estrategia consiste en formular una función de costo o pérdida. En este caso, utilizaremos el Error Cuadrático Medio (ECM), es decir, la suma del cuadrado de cada uno de los residuos, y buscaremos minimizarla con el algoritmo del descenso por gradiente. Este problema es *no convexo* en general, debido a la periodicidad y a la dependencia no lineal en ν y ϕ, por lo que la elección de los valores iniciales de los parámetros y de la tasa de aprendizaje η es relevante para la convergencia.

6.1. Formulación matemática

Dado un conjunto de datos $\{(x_i, y_i)\}_{i=1}^n$, se propone el modelo

$$\hat{y}_i = A\ \sin(\nu x_i + \phi).$$

6.2. Función de costo de mínimos cuadrados

La función de costo promedio (con factor conveniente $\frac{1}{2n}$) es

$$J(A, \nu, \phi) = \frac{1}{2n} \sum_{i=1}^{n} \left(A \sin(\nu x_i + \phi) - y_i\right)^2 .$$

Definiendo el **residuo** en el punto i como $e_i = A \sin(\nu x_i + \phi) - y_i$, se tiene

$$J(A, \nu, \phi) = \frac{1}{2n} \sum_{i=1}^{n} e_i^2.$$

6.3. Significado del gradiente

El vector gradiente $\nabla J(\boldsymbol{\theta}) = [\partial J/\partial A, \partial J/\partial \nu, \partial J/\partial \phi]^\top$ indica el *sentido de máximo crecimiento* de la función $J(\boldsymbol{\theta})$ correspondiente a ese conjunto particular de valores de los parámetros ($\boldsymbol{\theta} = [A, \nu, \phi]^\top$). Para *disminuir* el valor de $J(\boldsymbol{\theta})$, avanzamos en el *sentido opuesto* al gradiente de $J(\boldsymbol{\theta})$ correspondiente a ese conjunto particular de valores de los parámetros $\boldsymbol{\theta}$.

6.4. Gradientes respecto de los parámetros

Las derivadas parciales de $J(\boldsymbol{\theta})$ se obtienen aplicando la regla de la cadena y linealidad de la suma:

$$\frac{\partial J}{\partial A} = \frac{1}{n} \sum_{i=1}^{n} e_i\ \sin(\nu x_i + \phi),$$

$$\frac{\partial J}{\partial \nu} = \frac{1}{n} \sum_{i=1}^{n} e_i \Big(A x_i \cos(\nu x_i + \phi)\Big),$$

$$\frac{\partial J}{\partial \phi} = \frac{1}{n} \sum_{i=1}^{n} e_i \Big(A \cos(\nu x_i + \phi)\Big).$$

Estas expresiones surgen de que $\frac{\partial}{\partial A}(A \sin(\cdot)) = \sin(\cdot)$, $\frac{\partial}{\partial \nu} \sin(\nu x_i + \phi) = x_i \cos(\nu x_i + \phi)$ y $\frac{\partial}{\partial \phi} \sin(\nu x_i + \phi) = \cos(\nu x_i + \phi)$.

6.5. Regla de actualización del descenso por gradiente

Con una tasa de aprendizaje $\eta > 0$, las actualizaciones iterativas quedan

$$A^{(k+1)} = A^{(k)} - \eta \left.\frac{\partial J}{\partial A}\right|_{(A^{(k)},\nu^{(k)},\phi^{(k)})}, \quad \nu^{(k+1)} = \nu^{(k)} - \eta \left.\frac{\partial J}{\partial \nu}\right|_{(A^{(k)},\nu^{(k)},\phi^{(k)})},$$

$$\phi^{(k+1)} = \phi^{(k)} - \eta \left.\frac{\partial J}{\partial \phi}\right|_{(A^{(k)},\nu^{(k)},\phi^{(k)})}.$$

6.5.1. Algoritmo conceptual

1. Inicializar los parámetros A, ν y ϕ.
2. Calcular la función costo $J(\boldsymbol{\theta})$ con el valor actual de todos sus parámetros $\boldsymbol{\theta} = [A, \nu, \phi]^\top$.
3. Repetir hasta que se cumpla el *criterio de parada* del algoritmo:
   - Calcular el gradiente $\nabla J(\boldsymbol{\theta})$.
   - Actualizar todos los parámetros de $J(\boldsymbol{\theta})$ utilizando la "receta": $\theta \leftarrow \theta - \eta \nabla J$.
4. Devolver los parámetros ajustados.
5. Repetir los pasos 2-4 hasta que se cumpla el *criterio de parada* del algortimo (por ejemplo, se ejecutó el algoritmo el numéro máximo de iteraciones especificado).

6.5.2. Criterios de parada habituales:

- **Iteraciones máximas:** detener tras $N_{\text{máx.}}$ iteraciones.
- **Cambio en la función costo:** parar si

$$\left| \frac{J(\boldsymbol{\theta}^{(k+1)}) - J(\boldsymbol{\theta}^{(k)})}{J(\boldsymbol{\theta}^{(k)})} \right| < \varepsilon.$$

- **Norma del gradiente:** parar si $\|\nabla J(\boldsymbol{\theta}^{(k)})\| < \delta$.
- **Cambio en parámetros:** parar si $\frac{\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}\|}{\|\boldsymbol{\theta}^{(k)}\|} < \alpha$.

7. Resultados esperados

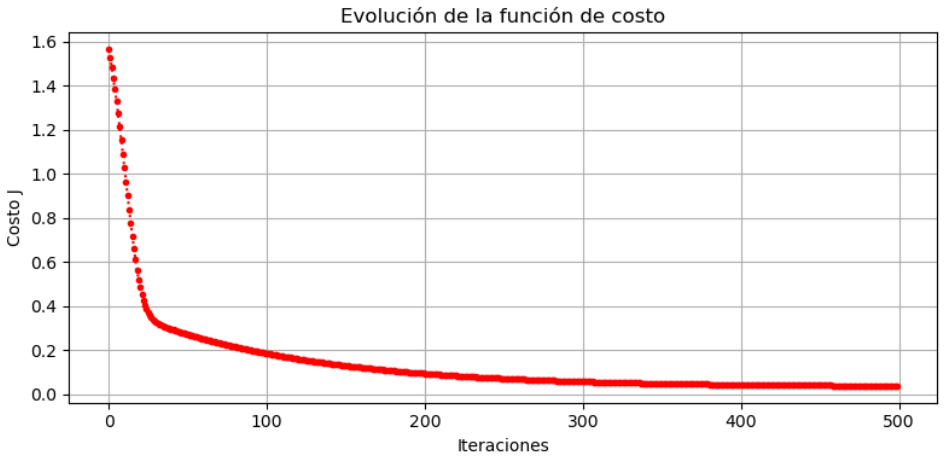

Figura 8: Evolución de la función de costo $J(A, \nu, \phi)$ durante el ajuste de la función $\hat{y} = A\sin(\nu x + \phi)$ mediante el método de descenso por el gradiente. Se observa cómo el error disminuye progresivamente a medida que los parámetros se optimizan.

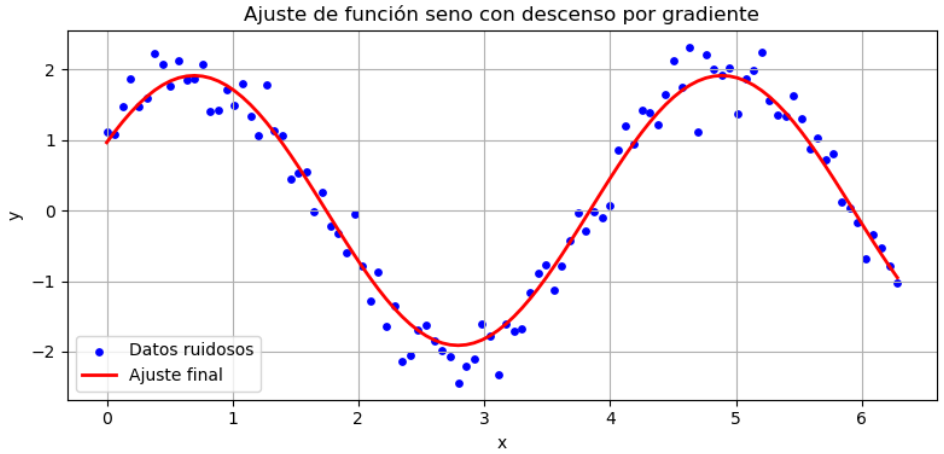

Figura 9: Datos ruidosos simulados (en azul) y curva senoidal $\hat{y} = A\sin(\nu x + \phi)$ final ajustada (en rojo), cuyos parámetros fueron obtenidos por medio del algoritmo de descenso por el gradiente.