Introducción al Gradiente Descendente

January 28, 2025

1 Introducción

El gradiente descendente es un algoritmo clave en aprendizaje automático utilizado para minimizar funciones de pérdida ajustando los parámetros de un modelo. En este documento, lo explicaremos utilizando el modelo de regresión lineal como ejemplo base.

2 Regresión Lineal

La regresión lineal busca encontrar la mejor línea que ajuste un conjunto de datos, definida como:

$$y = mx + b$$

Donde:

- y: variable dependiente (lo que queremos predecir).
- x: variable independiente (el dato para predecir).
- m: pendiente de la línea.
- b: intersección con el eje y.

3 Función de Pérdida

Para medir qué tan buena es una línea, utilizamos la función de pérdida del error cuadrático medio:

$$L(m, b) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

El objetivo es minimizar esta función ajustando los valores de m y b.

4 El Gradiente Descendente

El gradiente descendente es un método iterativo que ajusta los parámetros para minimizar la función de pérdida. Los parámetros se actualizan usando:

$$\text{nuevo valor} = \text{valor actual} - \alpha \cdot \frac{\partial L}{\partial \text{parámetro}}$$

Donde:

- α: tasa de aprendizaje (controla el tamaño del paso).
- $\frac{\partial L}{\partial \text{parámetro}}$: gradiente de la función de pérdida.

5 Derivadas Parciales

En la regresión lineal, las derivadas parciales de la función de pérdida son:

$$\frac{\partial L}{\partial m} = -\frac{2}{n} \sum_{i=1}^{n} x_i \cdot (y_i - \hat{y}_i)$$

$$\frac{\partial L}{\partial b} = -\frac{2}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)$$

6 Código de Ejemplo

A continuación, se presenta un ejemplo en Python para implementar gradiente descendente con regresión lineal:

```
import numpy as np

# Datos de ejemplo
x = np.array([1, 2, 3, 4, 5])
y = np.array([2.2, 2.8, 3.6, 4.5, 5.1])

# Parámetros iniciales
m, b = 0, 0
alpha = 0.01  # Tasa de aprendizaje
n = len(x)

# Gradiente descendente
for _ in range(1000):
    y_pred = m * x + b
    dm = (-2 / n) * np.sum(x * (y - y_pred))
    db = (-2 / n) * np.sum(y - y_pred)
```

```
    m -= alpha * dm
    b -= alpha * db

print(f"Pendiente (m): {m}, Intersección (b): {b}")
```