Proyecto Final: Optimización de Regresión Lineal con Descenso del Gradiente

Matro Entrenamiento de Inteligencia Artificial (MeIA) 2025

Objetivo

Implementar el método del **descenso del gradiente** para resolver una tarea de **regresión lineal** sobre el conjunto de datos **Iris**.

1. Introducción

El **descenso del gradiente** es un algoritmo de optimización iterativo usado para encontrar el mínimo de una función objetivo. En el caso de **regresión lineal**, se utiliza para ajustar los parámetros del modelo que minimizan el error cuadrático medio entre las predicciones y los valores reales.

La forma funcional del modelo de regresión lineal simple es:

$$\hat{y} = \sum_{j=1}^{m} w_j x_j + b$$

donde:

- m: indica el numero de caracteristicas todal de la base de datos.
- y: la variable dependiente que se desea predecir,
- x_j: la variable independiente j que influyen en la variable, dependiente (caracteristica i: por ejemplo, velocidad)
- b: término independiente (bias o sesgo),
- w_j: coeficientes del modelo de regresión lineal.

2. Función de Costo

Usamos el **error cuadrático medio (MSE)** como función de costo a minimizar:

$$f(W, b) = \frac{1}{2N} \sum_{i=1}^{N} \left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

$$= \frac{1}{2N} \sum_{i=1}^{N} \left[\left(\sum_{i=1}^{m} w_j x_j^{(i)} + b \right) - y^{(i)} \right]^2$$

donde N es el número de muestras (registros de la base de datos), $y^{(i)}$ es el valor real del registro i en la base de datos, $x_{i,j}$ es el valo real i de la caracteristica j en la base de datos y $W = (w_1, w_2, ..., w_m)$ los parametros a estimar (al igual que la b)

3. Gradientes Parciales

Para minimizar $f(W, b)$, derivamos con respecto a los parámetros:

$$\partial_b f = \frac{\partial f}{\partial b} = \frac{1}{N} \sum_{i=1}^{N} \left[\left(\sum_{i=1}^{m} w_j x_j^{(i)} + b \right) - y^{(i)} \right],$$

$$\partial_{w_j} f = \frac{\partial f}{\partial w_j} = \frac{1}{N} \sum_{i=1}^{N} \left[\left(\sum_{i=1}^{m} w_j x_j^{(i)} + b \right) - y^{(i)} \right] x_j^{(i)}$$

4. Algoritmo del Descenso del Gradiente

1. Sean b^0 y $w_j^0 \quad \forall j = 1, ..., m$ los valores iniciales. Estos pueden ser todos ceros (o valores aleatorios).

2. Repetir para $k = 0, ...K$:

   - Calcular gradientes

   $$\partial_{b^k} f = \frac{1}{N} \sum_{i=1}^{N} \left[\left(\sum_{i=1}^{m} w_j^k x_j^{(i)} + b^k \right) - y^{(i)} \right],$$

   $$\partial_{w_j^k} f = \frac{1}{N} \sum_{i=1}^{N} \left[\left(\sum_{i=1}^{m} w_j^k x_j^{(i)} + b \right) - y^{(i)} \right] x_j^{(i)}$$

   - Actualizar parámetros:

   $$b^{k+1} := b^k - \alpha \partial_{b^k} f$$
   $$w_j^{k+1} := w_j^{k+1} - \alpha \partial_{w_j^k} f$$

   donde α es la **tasa de aprendizaje** tipicamente igual a 0.1.

3. verificamos convergencia

   - si se cumple convergencia terminamos

5. Actividades

a) **Cargar el dataset Iris:** Utiliza `scikit-learn` o `pandas` para obtener las columnas:

- `sepal width (cm)` como variable independiente x_1
- `petal width (cm)` como variable independiente x_2
- `sepal length (cm)` como variable independiente x_3
- `petal length (cm)` como variable dependiente y

b) **Implementar desde cero el algoritmo de descenso del gradiente:** Evita usar `LinearRegression` de sklearn u otra herramienta donde ya este implementado. Usa `numpy` para todas las operaciones vectorizadas.

c) **Mostrar:**

- Evolución de la función de costo $f(W, b)$ a través de las iteraciones.
- Gráfica de los puntos reales y la línea ajustada.
- Comparar tus resultados con los obtenidos usando `LinearRegression` de `sklearn`.

6. Entregables

- Código documentado en Python (puede ser `.ipynb` o `.py`).
- Archivo PDF con:
  - Gráficas solicitadas.
  - Explicación paso a paso del algoritmo.
  - Interpretación de resultados.

7. Recursos Sugeridos

- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.* O'Reilly.
- `https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html`
- `https://numpy.org/doc/`