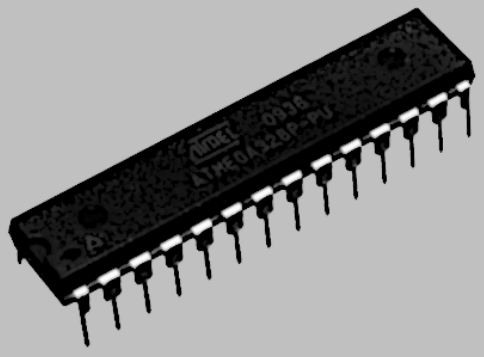

Inteligencia Artificial Embebida

Nombre: _____ Grupo: _____

Dr. Enrique García Trinidad
Tecnológico de Estudios Superiores de Huixquilucan
https://enriquegarcia.xyz
enrique.g.t@huixquilucan.tecnm.mx

Práctica 10: Implementación de Regresión Lineal desde Cero (Descenso de Gradiente)

1. Introducción Teórica

En la práctica anterior, utilizamos una herramienta de caja negra (`scikit-learn`) para resolver un problema de regresión. En esta práctica de expansión, construiremos el algoritmo de **Regresión Lineal desde cero**, utilizando únicamente operaciones algebraicas mediante la biblioteca `NumPy`.

Para encontrar los parámetros óptimos (pesos θ) que minimizan el error entre las predicciones y los datos reales, implementaremos el algoritmo de optimización conocido como **Descenso de Gradiente**.

La función de costo $J(\theta)$ (Error Cuadrático Medio) mide qué tan equivocado está el modelo. El descenso de gradiente actualiza iterativamente los parámetros θ moviéndose en la dirección opuesta al gradiente (la pendiente) de la función de costo:

$$\theta = \theta - \alpha \nabla J(\theta)$$

Donde:

- α es la **tasa de aprendizaje (learning rate)**, que determina el tamaño de los pasos en cada iteración.
- $\nabla J(\theta)$ es el gradiente de la función de costo, calculado de forma vectorizada como: $\frac{1}{m} X^T (X\theta - y)$, siendo m el número de muestras.

2. Desarrollo de la Práctica e Implementación

2.1. Paso 1: Importar dependencias

Solo requeriremos `NumPy` para el manejo de matrices y operaciones vectorizadas, y `Matplotlib` para visualizar el aprendizaje del modelo.

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

2.2. Paso 2: Función para calcular los gradientes

Esta función calcula el gradiente del error actual. Utilizar matrices (`.dot()`) en lugar de bucles `for` hace que el cálculo sea significativamente más rápido y eficiente, un concepto clave en Machine Learning conocido como vectorización.

```
def generate_gradient(X, theta, y):
    sample_count = X.shape[0]
    # Calcular el gradiente basado en algebra matricial
    return (1./sample_count)*X.T.dot(X.dot(theta)-y)
```

2.3. Paso 3 y 4: Lectura de datos e Inicialización

Se define la función para cargar el conjunto de datos de entrenamiento separando las características (X) de las etiquetas (y). Posteriormente, inicializamos los parámetros θ (pesos) con valores de 1.

```
def get_training_data(file_path):
    orig_data = np.loadtxt(file_path, skiprows=1)
    cols = orig_data.shape[1]
    return (orig_data, orig_data[:, :cols-1], orig_data[:, cols-1:])

# Inicializar el arreglo theta
def init_theta(feature_count):
    return np.ones(feature_count).reshape(feature_count, 1)
```

2.4. Paso 5: Implementación del Descenso de Gradiente

Este es el núcleo del algoritmo. Entramos en un bucle `while` que actualiza iterativamente el vector θ restándole el gradiente multiplicado por α. El ciclo se detiene cuando los gradientes son tan pequeños (menores a $1e-5$) que el modelo ha convergido a un mínimo. Además, guardamos el historial del costo $J(\theta)$ cada 10 pasos para analizar el aprendizaje.

```
def gradient_descending(X, y, theta, alpha):
    Jthetas = []
    Jtheta = (X.dot(theta)-y).T.dot(X.dot(theta)-y)
    index = 0
    gradient = generate_gradient(X, theta, y)

    while not np.all(np.absolute(gradient) <= 1e-5):
        theta = theta - alpha * gradient
        gradient = generate_gradient(X, theta, y)
        Jtheta = (X.dot(theta)-y).T.dot(X.dot(theta)-y)
        if (index+1) % 10 == 0:
            Jthetas.append((index, Jtheta[0]))
        index += 1
```

```
    return theta, Jthetas
```

2.5. Paso 6 y 7: Funciones de Visualización

Creamos dos funciones auxiliares: `showJTheta` para graficar la curva de la función de pérdida a lo largo de las iteraciones (curva de aprendizaje), y `showlinecurve` para visualizar la recta generada frente a los datos reales.

```
# Graficar la curva de cambio de la funcion de perdida
def showJTheta(diff_value):
    p_x, p_y = [], []
    for (index, sum) in diff_value:
        p_x.append(index)
        p_y.append(sum)
    plt.plot(p_x, p_y, color="b")
    plt.xlabel("Iteraciones (steps)")
    plt.ylabel("Funcion de Perdida")
    plt.title("Curva de Iteracion vs Perdida")
    plt.show()

# Graficar los puntos de datos y la curva ajustada
def showlinecurve(theta, sample_training_set):
    x, y = sample_training_set[:,1], sample_training_set[:,2]
    z = theta[0] + theta[1]*x
    plt.scatter(x, y, color="b", marker="x", label="Datos de muestra
        ")
    plt.plot(x, z, color="r", label="Curva de regresion")
    plt.legend()
    plt.show()
```

2.6. Paso 8: Ejecución y Resultados Finales

Se cargan los datos, se define una tasa de aprendizaje $\alpha = 0{,}01$, y se ejecuta el descenso de gradiente.

```
# Leer el conjunto de datos (Asegurese de tener el archivo correcto)
# training_data_include_y, training_x, y = get_training_data("./ML
    /02/lr2_data.txt")
# sample_count, feature_count = training_x.shape

# Para este ejemplo teorico asumimos feature_count = 2
alpha = 0.01
# theta = init_theta(feature_count)
# result_theta, Jthetas = gradient_descending(training_x, y, theta,
    alpha)
```

```
# Mostrar el parametro
# print(f"w: {result_theta[0][0]}, b: {result_theta[1][0]}")
# showJTheta(Jthetas)
# showlinecurve(result_theta, training_data_include_y)
```

3. Conclusión

Al ejecutar el modelo matemático estructurado desde cero, el algoritmo logra converger encontrando una pendiente (w) de aproximadamente **3.007** y un sesgo (b) de **1.668**. La curva de la función de pérdida demuestra cómo el error disminuye drásticamente en las primeras iteraciones hasta estabilizarse (converger).

4. Evaluación de Comprensión

Instrucciones: Selecciona la respuesta correcta para cada una de las siguientes preguntas.

1. ¿Cuál es el propósito principal del algoritmo de Descenso de Gradiente en esta práctica?
    A. Aumentar el número de características de los datos de entrada.
    B. Clasificar los datos en diferentes grupos de forma no supervisada.
    C. Encontrar los parámetros (pesos y sesgo) que minimicen la función de pérdida o error del modelo.
    D. Graficar automáticamente la línea de regresión sobre los puntos de dispersión.
2. En la regla de actualización del descenso de gradiente ($\theta = \theta - \alpha \nabla J$), ¿qué representa el parámetro α (alpha)?
    A. El error cuadrático medio actual.
    B. La tasa de aprendizaje (learning rate), que controla el tamaño de los pasos para alcanzar el mínimo.
    C. La cantidad de muestras totales en el conjunto de entrenamiento.
    D. El valor de la intersección con el eje Y.
3. ¿Por qué es una buena práctica graficar la curva de la función de pérdida vs las iteraciones (como lo hace la función `showJTheta`)?
    A. Para diagnosticar si el algoritmo está aprendiendo correctamente y si ha convergido, observando si el error disminuye con el tiempo.
    B. Para mostrar la relación lineal exacta entre el área de una casa y su precio.
    C. Para cambiar los datos atípicos de forma manual antes de entrenar el modelo.

D. Para determinar en qué lenguaje de programación fue escrito el algoritmo.

4. ¿Cuál de las siguientes ventajas se obtiene al calcular el gradiente utilizando operaciones matriciales como `X.T.dot(X.dot(theta)-y)` en lugar de bucles iterativos?

   A. Hace que el código sea más largo y fácil de depurar línea por línea.

   B. Evita el problema del sobreajuste (overfitting) en los datos de entrenamiento.

   C. Permite aprovechar la vectorización optimizada del CPU/GPU, haciendo los cálculos drásticamente más rápidos.

   D. Fuerza al modelo a usar una tasa de aprendizaje constante sin importar los datos.

5. ¿Qué sucedería muy probablemente si el valor de α (tasa de aprendizaje) se configura a un valor excesivamente alto?

   A. El modelo convergería al mínimo global en una sola iteración de forma perfecta.

   B. El algoritmo podría divergir; es decir, la función de pérdida aumentaría en cada iteración en lugar de disminuir, rebotando en el valle del gradiente.

   C. El algoritmo se transformaría automáticamente en un árbol de decisión.

   D. La recta de regresión tendría una pendiente de cero de inmediato.