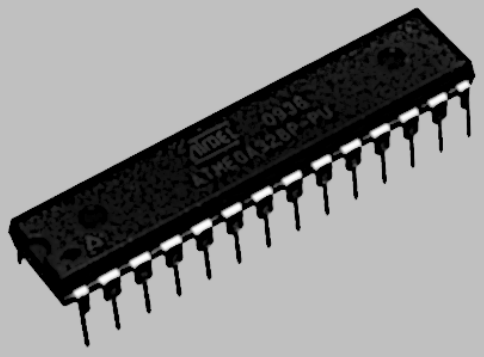

Inteligencia Artificial Embebida

Nombre: _____ Grupo: _____

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

Práctica 9: Fundamentos y Aplicación de la Regresión Lineal

1. Introducción Teórica

El **Aprendizaje Automático (Machine Learning)** se centra en el desarrollo de algoritmos que permiten a las computadoras aprender patrones a partir de datos empíricos. Dentro del aprendizaje supervisado, la **Regresión Lineal** es uno de los modelos fundamentales. Su objetivo es modelar la relación entre una variable dependiente (o etiqueta, y) y una o más variables independientes (o características, X).

En esta práctica, implementaremos un modelo de regresión lineal simple (una sola variable independiente) utilizando la biblioteca **scikit-learn**. Matemáticamente, el modelo asume que la relación se puede aproximar mediante la ecuación de una línea recta:

$$y = wx + b$$

Donde:

- w (weight o peso) representa la pendiente de la recta.
- b (bias o sesgo) es la intersección con el eje y.

El objetivo del algoritmo durante el entrenamiento es encontrar los valores óptimos de w y b que minimicen el error entre las predicciones del modelo y los valores reales, típicamente utilizando el método de **Mínimos Cuadrados Ordinarios (OLS)**.

2. Desarrollo de la Práctica e Implementación

2.1. Paso 1: Importar paquetes dependientes

Para construir el entorno de trabajo, importamos las herramientas matemáticas y de modelado necesarias. `numpy` nos permitirá manejar los datos como matrices y vectores (esencial para las operaciones algebraicas subyacentes), `matplotlib` facilitará el análisis visual, y `LinearRegression` de `sklearn` proporciona la implementación optimizada del algoritmo OLS.

```
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import numpy as np
```

2.2. Paso 2: Construir y visualizar el conjunto de datos

En el aprendizaje supervisado, necesitamos un conjunto de datos etiquetado. Aquí, x representa la característica de entrada (el área de la casa) y y representa la etiqueta objetivo (el precio). Es una buena práctica visualizar siempre los datos crudos en un diagrama de dispersión para confirmar empíricamente si existe una tendencia lineal antes de aplicar el modelo.

```
# x: area de la casa (matriz 2D), y: precio (vector)
x = np.array([121, 125, 131, 141, 152, 161]).reshape(-1,1)
y = np.array([300, 350, 425, 405, 496, 517])

plt.scatter(x, y)
plt.title("Distribucion de Datos: Area vs Precio")
plt.xlabel("Area")
plt.ylabel("Precio")
plt.show()
```

2.3. Paso 3: Entrenar el modelo

Al instanciar `LinearRegression()` y llamar al método `.fit(x, y)`, el algoritmo calcula internamente la función de costo y ajusta los parámetros w y b hasta encontrar la línea que mejor se ajusta a la distribución de los puntos proporcionados.

```
lr = LinearRegression() # Instanciar el modelo
lr.fit(x, y)            # Entrenar el modelo
```

2.4. Paso 4: Análisis y visualización del modelo entrenado

Una vez que el modelo ha convergido, podemos inspeccionar los parámetros aprendidos. El atributo `coef_` corresponde al peso w (cuánto aumenta el precio por cada unidad de área adicional) y `intercept_` corresponde al sesgo b.

```
w = lr.coef_[0]       # Pendiente del modelo (w)
b = lr.intercept_     # Interseccion del modelo (b)

print(f'Pendiente (w): {w:.4f}')
print(f'Interseccion (b): {b:.4f}')

# Graficar los datos reales junto con la linea de regresion
plt.scatter(x, y, label='Datos reales')
plt.plot([x[0], x[-1]], [x[0]*w + b, x[-1]*w + b], color='red',
   label='Modelo')
plt.xlabel("Area")
plt.ylabel("Precio")
plt.legend()
```

```
plt.title("Modelo Lineal Ajustado")
plt.show()
```

2.5. Paso 5: Realizar predicciones (Inferencia)

El propósito final del entrenamiento es la capacidad del modelo para predecir salidas precisas para datos nuevos o no vistos.

```
# Definir una nueva muestra (area = 130)
testX = np.array([[130]])
prediction = lr.predict(testX)

print(f'Precio inferido para un area de 130: {prediction[0]:.2f}')
```

3. Conclusión

El modelo logró extraer el patrón subyacente de los datos de entrenamiento, determinando una tasa de cambio (pendiente) de **4.98** y un sesgo de **-274.87**. Al aplicar este modelo a un dato no visto (área de 130), la red infiere un precio de **373.13**, demostrando la utilidad de la regresión lineal como herramienta predictiva.

4. Evaluación de Comprensión

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

1. ¿Cuál es el objetivo principal de un algoritmo de Regresión Lineal Simple?
   - A. Clasificar imágenes en diferentes categorías.
   - B. Encontrar la línea recta que mejor se ajuste y describa la relación entre una variable independiente y una dependiente.
   - C. Agrupar datos no etiquetados en clústeres.
   - D. Reducir la dimensionalidad de un conjunto de datos masivo.
2. En la ecuación del modelo lineal $y = wx + b$, ¿qué representa el parámetro w?
   - A. La variable independiente (característica).
   - B. El punto donde la recta cruza el eje y (sesgo o bias).
   - C. La pendiente de la recta, que indica el peso o la tasa de cambio.
   - D. El margen de error cuadrático medio.

3. En la biblioteca `scikit-learn`, ¿qué método se utiliza para que el modelo aprenda los parámetros óptimos a partir de los datos de entrenamiento?

    A. `.predict(x, y)`

    B. `.train(x, y)`

    C. `.fit(x, y)`

    D. `.transform(x, y)`

4. ¿Por qué es necesario redimensionar (usar `.reshape(-1,1)`) la variable de entrada x en este ejercicio antes de entrenar el modelo?

    A. Porque `scikit-learn` espera que las características de entrada sean una matriz bidimensional (2D), no un vector 1D.

    B. Para convertir los valores de coma flotante a enteros.

    C. Para eliminar los valores atípicos (outliers) del conjunto de datos.

    D. Porque el área de la casa siempre debe ser un número positivo.

5. Si tienes un conjunto de datos donde los puntos forman una curva en forma de U o parábola en un gráfico de dispersión, ¿sería adecuado usar este modelo de Regresión Lineal Simple?

    A. Sí, la regresión lineal puede ajustarse perfectamente a cualquier tipo de curva cambiando el sesgo (b).

    B. No, porque la regresión lineal simple asume una relación lineal (línea recta), y un modelo lineal tendría un alto error al intentar ajustar datos curvos.

    C. Sí, siempre y cuando se aumente el número de datos de entrenamiento.

    D. No, porque la biblioteca `scikit-learn` solo admite datos en línea recta creados artificialmente.