Universidad Rafael Landívar
Inteligencia Artificial
Primer Semestre 2025

Laboratorio No. 6

Implementación de Regresión Lineal desde Cero

Objetivo

El objetivo de este laboratorio es que los estudiantes implementen regresión lineal desde cero, utilizando dos métodos fundamentales:

- Mínimos cuadrados ordinarios (MCO) (solución analítica).
- Gradiente descendiente (solución iterativa).

Los estudiantes deberán:

- Comprender la teoría detrás de cada método.
- Implementar funciones para calcular los coeficientes y hacer predicciones.
- Comparar los resultados obtenidos.
- Evaluar el rendimiento usando métricas básicas.
- Visualizar la evolución del ajuste del modelo con gradiente descendiente.

Instrucciones

1. Dataset

Usar el siguiente dataset sintético (generado en código) para regresión lineal:

```
import numpy as np

np.random.seed(42)  # Para reproducibilidad
X = 2 * np.random.rand(100, 1)  # 100 muestras, 1 característica
y = 4 + 3 * X + np.random.randn(100, 1)  # Relación lineal con ruido
```

Ecuación general:

$$y = \beta_0 + \beta_1 X + \epsilon$$

Donde:

- y: Variable dependiente (objetivo).
- X: Variable independiente (feature).
- β_0: Intercepto.
- β_1: Pendiente.
- ϵ: Error (ruido).

2. Implementación de Mínimos Cuadrados Ordinarios (MCO)

Fórmula teórica:

$$\beta = \left(X^T X\right)^{-1} X^T y$$

Requisitos:

- Crear una función `minimos_cuadrados(X, y)` que calcule los coeficientes β_0 (intercepto) y β_1 (pendiente).
- Usar operaciones matriciales con *numpy* (sin bibliotecas de ML).

3. Implementación de Gradiente Descendiente

Fórmula teórica:

El gradiente descendiente minimiza la función de costo (Error Cuadrático Medio, MSE):

$$J(\beta) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$$

Para actualizar los coeficientes:

$$\beta_j \leftarrow \beta_j - \alpha\frac{\partial}{\partial\beta_j}J(\beta)$$

donde J(β) es la función de costo (MSE).

Requisitos:

- Crear una función `gradiente_descendiente(X, y, alpha=0.01, iteraciones=1000)` que:
  - Inicialice β_0 y β_1 en cero.
  - Actualice los coeficientes iterativamente usando la regla del gradiente.
  - Guarde el historial del costo (MSE) en cada iteración.

4. Evaluación

- Calcular el Error Cuadrático Medio (MSE) para ambos métodos:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$$

- Graficar:
  1. La línea de regresión obtenida por cada método.
  2. La evolución del costo (MSE) en gradiente descendiente.
  3. La evolución de la línea de regresión a lo largo de varias iteraciones.

5. Comparación y Análisis

- ¿Qué método converge más rápido?
- ¿Cómo afecta la tasa de aprendizaje (*alpha*) al gradiente descendiente?
- ¿Los coeficientes obtenidos por ambos métodos son similares? Explica por qué.
- ¿Por qué MCO da una solución exacta mientras que el gradiente descendiente requiere múltiples iteraciones?
- ¿Cuándo conviene usar MCO en lugar de gradiente descendiente y viceversa?
- ¿Qué sucede si los datos tienen más de una variable independiente?

Entregables

- **Repositorio** https://classroom.github.com/a/2fDm2a5H
- **Archivo Tarea.py** con:
  - Implementación de MCO y gradiente descendiente en funciones.
  - Las funciones definidas sin ejecución de código ni gráficos.
- **Notebook Enunciado.ipynb** con:
  - Importación de las funciones desde Tarea.py.
  - Ejecución de las funciones con el dataset dado.
  - Explicación de las fórmulas usadas.
  - Comparación de los métodos.
  - Gráficos de los resultados obtenidos.
  - Análisis y respuestas a las preguntas planteadas.

Ejemplo de Salida Esperada

```
# Resultados de MCO
Beta MCO: [4.215] (intercepto), [2.977] (pendiente)
MSE MCO: 0.93

# Resultados de Gradiente Descendiente
Beta GD: [4.210] (intercepto), [2.975] (pendiente)
MSE GD: 0.93
```

Tips

- Usar numpy para operaciones matriciales (no está prohibido, pero no usar sklearn ni bibliotecas que hagan el proceso de MCO/GD).
- Vectorizar cálculos para mayor eficiencia.
- Para gradiente descendiente, probar con alpha=0.1, 0.01, 0.001 y observar/analizar/comentar diferencias

Referencias adicionales:

- https://www.ibm.com/es-es/think/topics/gradient-descent