Centro Universitario de Ciencias Exactas e Ingenierías

Ingeniería en Computación
Pre 1.3 - Desceso del gradiente

PRESENTA:

Ramirez Gutierrez Hugo Vladimir
Código: 220287144
Materia:

Seminario de Solución de problemas de Inteligencia Artificial 2

Docente: Diego Campos Pena

Índice:

Introducción

El descenso de gradiente es un algoritmo de optimización utilizado para minimizar una función objetivo. Funciona iterativamente moviéndose en la dirección opuesta al gradiente de la función en el punto actual.

El gradiente de una función es un vector que indica la dirección y la tasa de cambio más pronunciada de la función en un punto dado. En el contexto de optimización, el gradiente apunta en la dirección de máximo crecimiento de la función. Por lo tanto, moverse en la dirección opuesta al gradiente nos lleva hacia donde la función disminuye más rápidamente.

El descenso de gradiente comienza con un punto inicial y calcula el gradiente de la función en ese punto. Luego, se mueve una pequeña distancia (determinada por la tasa de aprendizaje) en la dirección opuesta al gradiente. Este proceso se repite iterativamente hasta que se alcanza un punto donde el gradiente es cercano a cero o hasta que se alcanza un número predeterminado de iteraciones.

El descenso de gradiente es ampliamente utilizado en el campo del aprendizaje automático y la optimización numérica para entrenar modelos y encontrar soluciones óptimas en problemas de optimización.

Desarrollo:

Se utiliza la biblioteca NumPy para cálculos numéricos y Matplotlib para visualización.

1. Función Objetivo (f)
   - Define una función objetivo que toma dos variables (x1 y x2) y devuelve un valor basado en una fórmula.

2. Gradiente de la Función (gradient)
   - Calcula el gradiente de la función objetivo en un punto dado (x1, x2)

3. Descenso de Gradiente (gradient_descent)
   - Implementa el algoritmo de descenso de gradiente.
   - Comienza con un punto aleatorio.
   - En cada iteración, actualiza el punto moviéndose en la dirección opuesta al gradiente multiplicado por una tasa de aprendizaje (learning_rate).
   - Registra el historial de puntos en cada iteración.

4. Parámetros:
   - learning_rate: Tasa de aprendizaje para el descenso de gradiente.

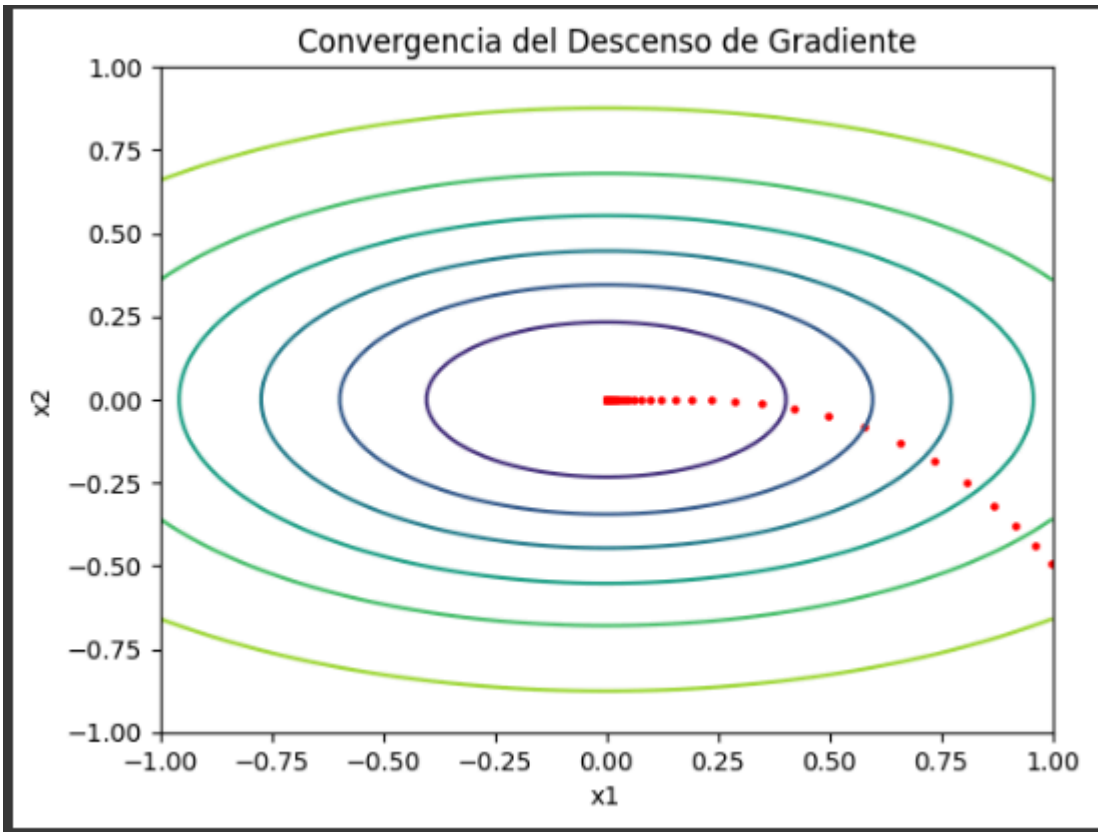
- iterations: Número de iteraciones del descenso de gradiente.

5. Visualización:
   - Genera una cuadrícula de puntos en el rango [-1, 1] para x1 y x2.
   - Calcula los valores de la función objetivo en esta cuadrícula.
   - Traza las curvas de nivel de la función objetivo.
   - Ejecuta el descenso de gradiente y traza los puntos en cada iteración.
   - Muestra el punto óptimo encontrado y el valor mínimo de la función.

Código:

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

def f(x1, x2):
    return 10 - np.exp(-(x1**2 + 3*x2**2))

def gradient(x1, x2):
    return np.array([2 * x1 * np.exp(-(x1**2 + 3*x2**2)), 6 * x2 *
np.exp(-(x1**2 + 3*x2**2))])

def gradient_descent(lr, num_iterations):
    x = np.random.uniform(-1, 1, size=2)
    points_history = [x.copy()]
    for _ in range(num_iterations):
        x -= lr * gradient(*x)
        points_history.append(x.copy())
    return points_history

learning_rate = 0.1
iterations = 100

x_vals = np.linspace(-1, 1, 100)
X1, X2 = np.meshgrid(x_vals, x_vals)
Z = f(X1, X2)

plt.figure()
plt.contour(X1, X2, Z, cmap='viridis')
plt.xlabel('x1')
plt.ylabel('x2')
plt.title('Convergencia del Descenso de Gradiente')

points_history = gradient_descent(learning_rate, iterations)
for point in points_history:
    plt.scatter(point[0], point[1], color='red', s=5)

optimal_point = points_history[-1]
min_value = f(*optimal_point)
print("\nPunto óptimo:", optimal_point)
print(f'Valor mínimo de la función: {min_value}\n')

plt.show()
```

Resultado:



Conclusión:

En resumen, el descenso de gradiente es un algoritmo fundamental en el campo de la optimización, utilizado para encontrar el mínimo de una función objetivo. Se basa en el concepto de moverse en la dirección opuesta al gradiente de la función en cada iteración, lo que nos lleva hacia el punto donde la función alcanza su mínimo local o global. Este método es eficaz y ampliamente aplicable en diversas áreas, desde el entrenamiento de modelos de aprendizaje automático hasta la optimización de problemas numéricos. Su comprensión y aplicación son esenciales para resolver una amplia gama de problemas de optimización en la ciencia, la ingeniería y otros campos.