22 DE FEBRERO DE 2024

DESCESO DEL GRADIENTE

IVAN YUTLANIH MORALES ZEPEDA
CRUZ GALLEGOS RAMSES AARÓN
SEM INTELIGENCIA ARTIFICIAL 2
MTRO: CAMPOS PEÑA DIEGO

INDICE

INTRODUCCIÓN

El descenso de gradiente es un algoritmo fundamental en el campo del aprendizaje automático y la optimización. Su objetivo es encontrar el mínimo (o máximo) de una función. Consta de tres partes importante:

- **Función Objetivo**: Se define una función matemática que se busca minimizar o maximizar. En este caso, utilizamos la función: f(x1, x2) = 10 - e^(-(x1^2 + 3*x2^2)).
- **Gradiente**: Se calcula el gradiente de la función, que es un vector que indica la dirección y magnitud del cambio más rápido en la función. El gradiente es un guía hacia el mínimo local.
- **Actualización de Parámetros**: Iterativamente, se actualizan los valores de los parámetros (x1 y x2) utilizando la fórmula: x -= lr * grad, donde lr es el *learning rate* (tasa de aprendizaje).

DESARROLLO

1. Función Objetivo (f(x1, x2)):
    - La función f(x1, x2) es: f(x1, x2) = 10 - e^(-(x1^2 + 3*x2^2))
    - Esta función toma dos argumentos (x1 y x2) y devuelve un valor numérico.
2. Gradiente de la Función (gradient(x1, x2)):
    - El gradiente de la función se calcula con la función gradient(x1, x2).
    - El gradiente es un vector que indica la dirección y magnitud del cambio más rápido en la función.
    - En este caso, se derivan las componentes de f(x1, x2) con respecto a x1 y x2.
3. Descenso de Gradiente (gradient_descent(lr, num_iterations)):
    - El descenso de gradiente es un algoritmo de optimización que busca el mínimo de una función.
    - lr (learning rate) es un hiperparámetro que controla el tamaño de los pasos en cada iteración.
    - num_iterations es el número de iteraciones que se realizarán.
    - Se comienza con valores iniciales aleatorios para x1 y x2.
    - En cada iteración, se calcula el gradiente y se actualiza x1 y x2 usando la fórmula: x -= lr * grad.
4. Resultado Final:
    - El programa ejecuta el descenso de gradiente con los parámetros dados.
    - Imprime el punto óptimo (x1 y x2) y el valor mínimo de la función.

CÓDIGO

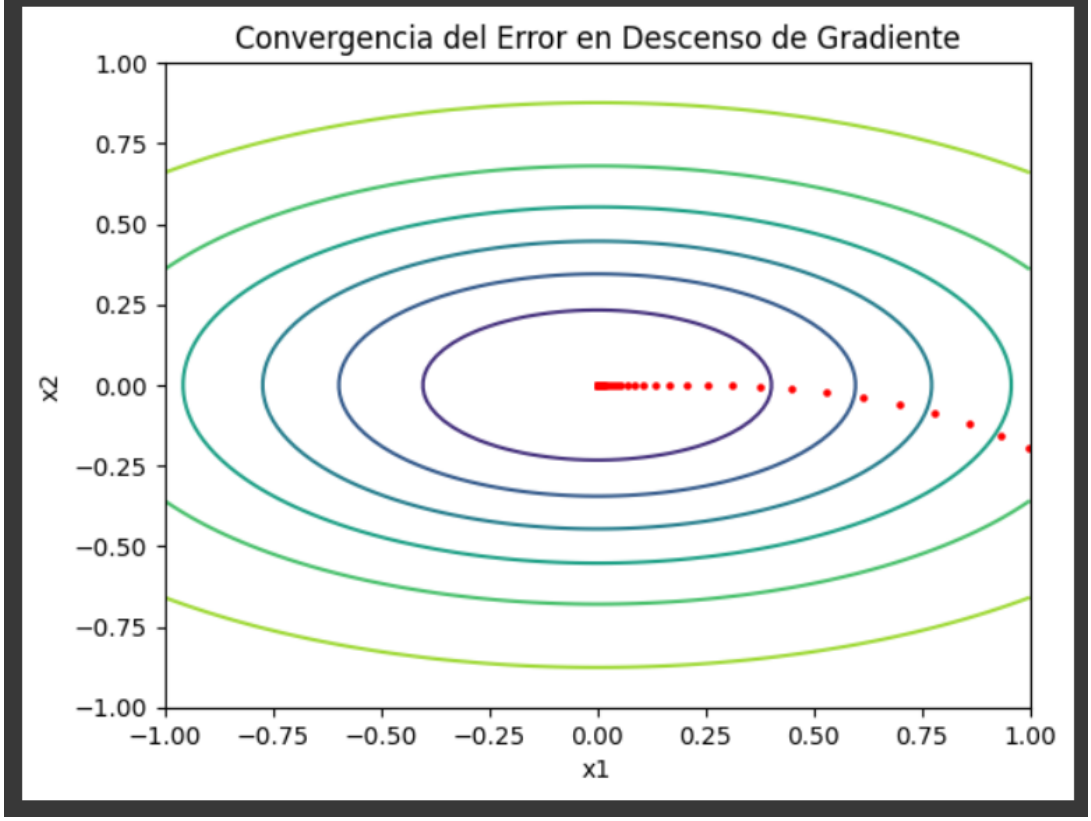
```
import numpy as np
import matplotlib.pyplot as plt

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

# Gradiente de la función
def gradient(x1, x2):
    df_dx1 = 2 * x1 * np.exp(-(x1**2 + 3*x2**2))
    df_dx2 = 6 * x2 * np.exp(-(x1**2 + 3*x2**2))
    return np.array([df_dx1, df_dx2])

# Descenso de gradiente
def gradient_descent(lr, num_iterations):
    x = np.random.uniform(-1, 1, size=2)
    points_history = [x.copy()]  # Inicializar lista para historial de
puntos
    for _ in range(num_iterations):
        grad = gradient(x[0], x[1])
        x -= lr * grad
        points_history.append(x.copy())  # Agregar el nuevo punto al
historial
    return points_history

# Parámetros
learning_rate = 0.1
iterations = 100

# Creación de la gráfica
x1_vals = np.linspace(-1, 1, 100)
x2_vals = np.linspace(-1, 1, 100)
X1, X2 = np.meshgrid(x1_vals, x2_vals)
Z = f(X1, X2)

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

# Ejecución 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)  # Graficar cada
punto del historial

# Obtener y mostrar el punto óptimo y el valor mínimo de la función
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()
```

RESULTADOS

- Podemos observar cómo los valores de (x_1) y (x_2) evolucionan a lo largo de las iteraciones hasta alcanzar un mínimo local de la función.
- La función (f) disminuye a medida que avanzan en las iteraciones. El algoritmo busca los valores de (x_1) y (x_2) que minimizan (f).
- Los resultados muestran convergencia (es decir, la función se estabiliza en un valor mínimo), significa que el descenso de gradiente ha encontrado una solución.

```
Punto óptimo: [ 5.04851329e-10 -2.67425015e-39]
Valor mínimo de la función:, 9.0
```



CONCLUSIONES

Este algoritmo nos permite encontrar soluciones óptimas en problemas de optimización, ya que el descenso gradiente es altamente utilizado en el aprendizaje automático y la ciencia de datos.

Los resultados del algoritmo fueron satisfactorios, logrando optimizar la función objetivo. El desarrollo del código no representó un problema mayor y con base a los resultados interpretamos que la implementación del código es correcta porque observamos como los valores de X1 y X2 minimizaron la función objetivo hasta encontrar el valor mínimo.

GITHUB

I. SEM-IA-2/R_Pre_1.3_DescesoDelGradiente at main · CRUZITO4O4/SEM-IA-2 (github.com)
II. https://github.com/IvanYMz/SSPIA2-Tareas