Universidad de Guadalajara.

Centro Universitario de Ciencias Exactas e Ingenierías.

Ingeniería en computación.

Sección: D05.

Seminario de solución de problemas de inteligencia artificial II.

Pre. 1.3.

Mtro. Diego Campos Peña

Montoya Vargas Roberto.

José Ángel Sevilla Varela.

Crea un software que por medio del descenso del gradiente sea capaz de optimizar la función adjunta en la imagen en los limites -1 a 1. Tiene que ser capaz de cambiar el lr (learning rate). Los valores iniciales tienen que ser de forma aleatoria.

$$f(x_1, x_2) = 10 - e^{-(x_1^2 + 3x_2^2)}$$

Introducción:

El gradiente descendente es como un valioso buscador de caminos en un paisaje de montañas y valles. En el campo del aprendizaje automático y la optimización numérica, se utiliza para encontrar los puntos más bajos (o altos) de una función matemática, lo que es esencial para resolver problemas como la optimización de modelos, regresión, clasificación, y más. Su idea principal es ajustar continuamente los parámetros de un modelo o función de manera que se minimice el error o la pérdida. En resumen, el gradiente descendente es como un guía confiable que nos lleva a través de un terreno complicado para encontrar los puntos más bajos (o altos) que estamos buscando.

Desarrollo:

El gradiente descendente es como seguir el rastro de migas de pan en un bosque oscuro. Imagina que estás en un bosque y quieres llegar al punto más bajo del terreno. Para hacerlo, sueltas migas de pan mientras caminas y sientes la inclinación del terreno. Luego, das pasos en la dirección opuesta a la pendiente, asegurándote de no dar pasos demasiado grandes o pequeños, controlados por tu ritmo. Repites este proceso, siguiendo las migas de pan y ajustando tu ritmo según la inclinación del terreno, hasta que llegas al punto más bajo. De manera similar, el gradiente descendente calcula la pendiente de una función matemática y ajusta los parámetros del modelo en la dirección opuesta a esta pendiente, controlando el tamaño de los pasos con una tasa de aprendizaje. Se repite este proceso hasta que se llega al mínimo de la función objetivo. Para visualizar este algoritmo, lo aplicaremos a una función específica.

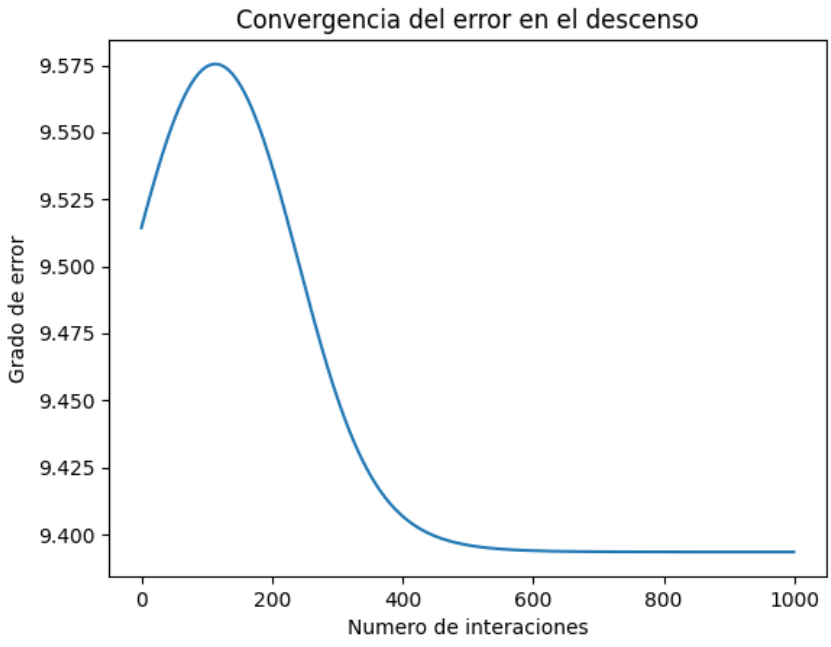
Código:

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

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

def G(x, y):
    return np.array([(1 - 2*(x**2))*np.exp(-x**2-y**2),
                     -2*x*y*np.exp(-x**2-y**2)])

def gradient_descent(lr, max_iterations):
    x1 = np.random.uniform(-1, 1)
    x2 = np.random.uniform(-1, 1)
    errors = []
    for i in range(max_iterations):
        grad = G(x1, x2)
        x1 -= lr * grad[0]
        x2 -= lr * grad[1]
        x1 = max(-1, min(x1, 1))
        x2 = max(-1, min(x2, 1))
        error = F(x1, x2)
        errors.append(error)
        if i % 100 == 0:
            print(f"Numero de iteracion: {i}: F({x1}, {x2}) = {error}")
    return x1, x2, errors

lr = 0.01
max_iterations = 1000
opt_x1, opt_x2, errors = gradient_descent(lr, max_iterations)

plt.plot(range(len(errors)), errors)
plt.xlabel('Numero de interaciones')
plt.ylabel('Grado de error')
plt.title('Convergencia del error en el descenso')
plt.show()

print("\n\n\n")
print(f"Evaluacion optimizada para: X1 = {opt_x1}, x2 = {opt_x2}")
print(f"Optimizacion para la funcion F(x1, x2) = {F(opt_x1, opt_x2)}")
```

Evaluación de iteraciones:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
[Running] python -u "g:\Otros ordenadores\Mi PC\Universidad\INCO\Semestre 8 INCO\Seminario de I
Numero de iteracion: 0: F(0.430118034151786, -0.7330648558710127) = 9.514407559266134
Numero de iteracion: 100: F(0.05108044049731945, -0.9230191777632771) = 9.574535435420255
Numero de iteracion: 200: F(-0.3511116507721209, -0.8048204705373754) = 9.537456827807015
Numero de iteracion: 300: F(-0.6026099529576111, -0.4863586515058462) = 9.45101463385319
Numero de iteracion: 400: F(-0.6855368741984104, -0.22862306571496194) = 9.406803347622377
Numero de iteracion: 500: F(-0.703160850838407, -0.09876225951294261) = 9.396004883325437
Numero de iteracion: 600: F(-0.7064033980824166, -0.04190151914996053) = 9.39393103643751
Numero de iteracion: 700: F(-0.7069820019086437, -0.01771814260371666) = 9.393552722574043
Numero de iteracion: 800: F(-0.7070846647153254, -0.007487656741923933) = 9.393484374830978
Numero de iteracion: 900: F(-0.7071028617685754, -0.003163931410726194) = 9.393472050007547
```

Grafica



Conclusión:

En esta práctica, observamos cómo funciona el algoritmo de descenso del gradiente utilizando una función matemática para ver cómo optimiza la búsqueda de mínimos o máximos en esa función. Al entrenar este algoritmo con funciones aplicadas, como en el aprendizaje automático, podemos ver cómo estas funciones mejoran su rendimiento a medida que se entrenan, lo que nos lleva a una búsqueda más eficiente de soluciones.

Bibliografía:

Frederickson, B. (s. f.). *An Interactive Tutorial on Numerical Optimization*. https://www.benfrederickson.com/numerical-optimization/

Studocu. (s. f.). *Redes neuronales - Tema 8. Redes Neuronales Pedro Larra˜naga, I˜naki Inza, Abdelmalik Moujahid - Studocu*. https://www.studocu.com/es-ar/document/universidad-tecnologica-nacional/inteligencia-artificial/redes-neuronales/79957842