Universidad Autónoma de Aguascalientes

Centro de Ciencias Básicas

Departamento de Ciencias de la Computación

Optativa Profesionalizante II: Machine Learning y Deep Learning

10° "A"

Actividad 3: Descenso del gradiente en 3D en Python

Docente: Dr. Francisco Javier Luna Rosas

Alumno: Joel Alejandro Espinoza Sánchez (211800)

Fecha de Entrega: Aguascalientes, Ags., 16 de marzo del 2023.

Actividad 3: Descenso del gradiente en 3D en Python

La Regla de Aprendizaje del Perceptrón

El alumno deberá elaborar un documento `*.pdf` donde implemente el descenso del gradiente en 3D.

El documento debe contener lo siguiente:

- Análisis del descenso del gradiente en 3D.
- Implementación del descenso del gradiente en 3D en Python o en el lenguaje de programación de preferencia.
- Evaluación del descenso del gradiente en 3D en Python o en el lenguaje de programación de preferencia.

El alumno deberá subir a la plataforma el archivo (`*.pdf`) y un documento auto-reproducible (`*.html`) que deberá contener:

- Portada.
- Evidencias de la actividad.
- Conclusiones.
- Referencias (formato APA).

El algoritmo de gradiente descendente es un mecanismo de entrenamiento de sistemas de aprendizaje automático como los basados en redes neuronales. Es el más extendido y utilizado para el aprendizaje o entrenamiento de redes neuronales debido a su sencillez y facilidad de implementación.

Una neurona artificial está formada por los siguientes componentes:

- Entradas: canales o interfaces por donde reciben la información de entrada del exterior del sistema o de otras neuronas.
- Pesos sinápticos: nivel, grado o importancia de la información o comunicación que recibe de otra neurona.
- Regla de propagación: nivel, grado o importancia de la información de salida que proporcionará a una red neuronal en función de la información de entrada que recibe y los pesos sinápticos.
- Función de activación: estado de activación de cada neurona.
- Función de salida: información de salida que proporciona la neurona a otras o al exterior del sistema.

Primero se toma de manera aleatoria el valor de x, después en cada iteración se actualiza el valor de x con la siguiente fórmula:

$$x_1 = x_0 - n(f'(x_0))$$

Donde n es la tasa de aprendizaje y $f'(x_0)$ es el gradiente.

El programa termina hasta que se hayan completado el número de iteraciones preestablecidas.

La implementación de este código requiere de las siguientes librerías:

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

Después se definen las funciones en tres dimensiones con las que se trabajarán

In [4]:
```
def f(x,y):
    return x**2 + y**3

def g(x,y):
    return np.array([2*x, 3*y*y])
```

Posteriormente se definen los parámetros que se utilizarán en el procedimiento:

In [6]:
```
alpha = 0.1  # Tasa de aprendizaje
num_iters = 100  # Número de iteraciones
x0 = np.array([2,2])  # Punto inicial
```

A continuación se define una función donde se aplica el descenso del gradiente en 3D y después se ejecuta:

In [7]:
```
# Descenso del gradiente
def grad_desc_3d(f, g, alpha, num_iters, x0):
    x = np.zeros((num_iters+1,2))
    x[0,:] = x0
    for i in range(num_iters):
        x[i+1,:] = x[i,:] - alpha*g(x[i,0], x[i,1])
    return x

# Ejecución del descenso del gradiente
coords = grad_desc_3d(f, g, alpha, num_iters, x0)
```

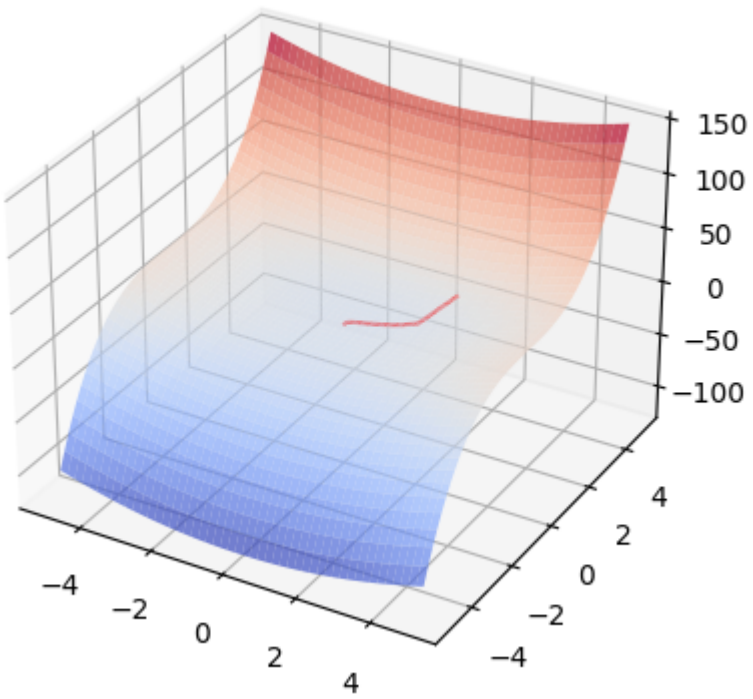
Para visualizar los resultados podemos observar en un espacio tridimensional lo realizado:

In [8]:
```
x = np.linspace(-5, 5, 50)
y = np.linspace(-5, 5, 50)
X, Y = np.meshgrid(x, y)
Z = f(X,Y)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.plot(coords[:,0], coords[:,1], f(coords[:,0], coords[:,1]), color='r')
ax.plot_surface(X, Y, Z, cmap='coolwarm', alpha=0.7)

plt.show()
```



Conclusiones

Es interesante e importante poder implementar las bases de estos temas para entenderlos en un futuro, pues, posteriormente no basta con sólo importar librerías que realicen el trabajo pesado, ya que, implementar manualmente estos algoritmos nos enseña a qué hay detrás del algoritmo, cómo funciona y poder comprender realmente qué está ocurriendo como la base de una red neuronal y la forma en la que ésta aprende realizando el descenso del gradiente. Es muy útil la implementación de estos algoritmos en estas tareas para las futuras tareas de la materia y aplicaciones de Machine Learning en la vida personal.

Referencias

- Anónimo (s.f.) “Red neuronal artificial”. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Red_neuronal_artificial.
- Data Scientest (2021) “Perceptrón. ¿Qué es y para qué sirve?”. Obtenido de Data Scientest: https://datascientest.com/es/perceptron-que-es-y-para-que-sirve.
- Luna, F. (2023) “El Modelo de McCulloch – Pitts”. Apuntes de ICI 10°.