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 2.01: Descenso del gradiente en 2D en Python

Docente: Dr. Francisco Javier Luna Rosas

Alumno: Joel Alejandro Espinoza Sánchez (211800)

Fecha de Entrega: Aguascalientes, Ags., 15 de marzo del 2023.

Actividad 2.01: Descenso del gradiente en 2D en Python

El alumno deberá elaborar un documento `*.pdf` donde implemente el descenso del gradiente en 2D.

El documento debe contener lo siguiente:

- Análisis del descenso del gradiente en 2D.
- Implementación del descenso del gradiente en 2D en Python o en el lenguaje de programación de preferencia.
- Evaluación del descenso del gradiente en 2D 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 [1]:
```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

Se usarán las siguientes funciones como representación de sus contrapartes matemáticas:

$$f(x) = x^2 + 4x$$
$$g(x) = f'(x) = 2x + 4$$

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

def g(x):
    return (2*x)+(4)
```

Se definen las variables iniciales del código:

In [3]:
```
# Algoritmo
n = 0.07
x = [1]
y = [f(x[0])]
```

```
pend = [g(x[0])]
i = 0
tolerance = 0.000000001

# Graficación
figsize = (6,4)
dpi = 150
```

De esta forma ahora sí se aplica el algoritmo del descenso del gradiente:

In [6]:
```
while(g(x[i]) > tolerance):
    x.append(x[i] - (n*g(x[i])))
    y.append(f(x[i]) - (n*g(x[i])))
    pend.append(g(x[i]))
    i = i + 1

dt = pd.DataFrame({"X": x, "Y": y, "Pendiente": pend})
print(dt)
```

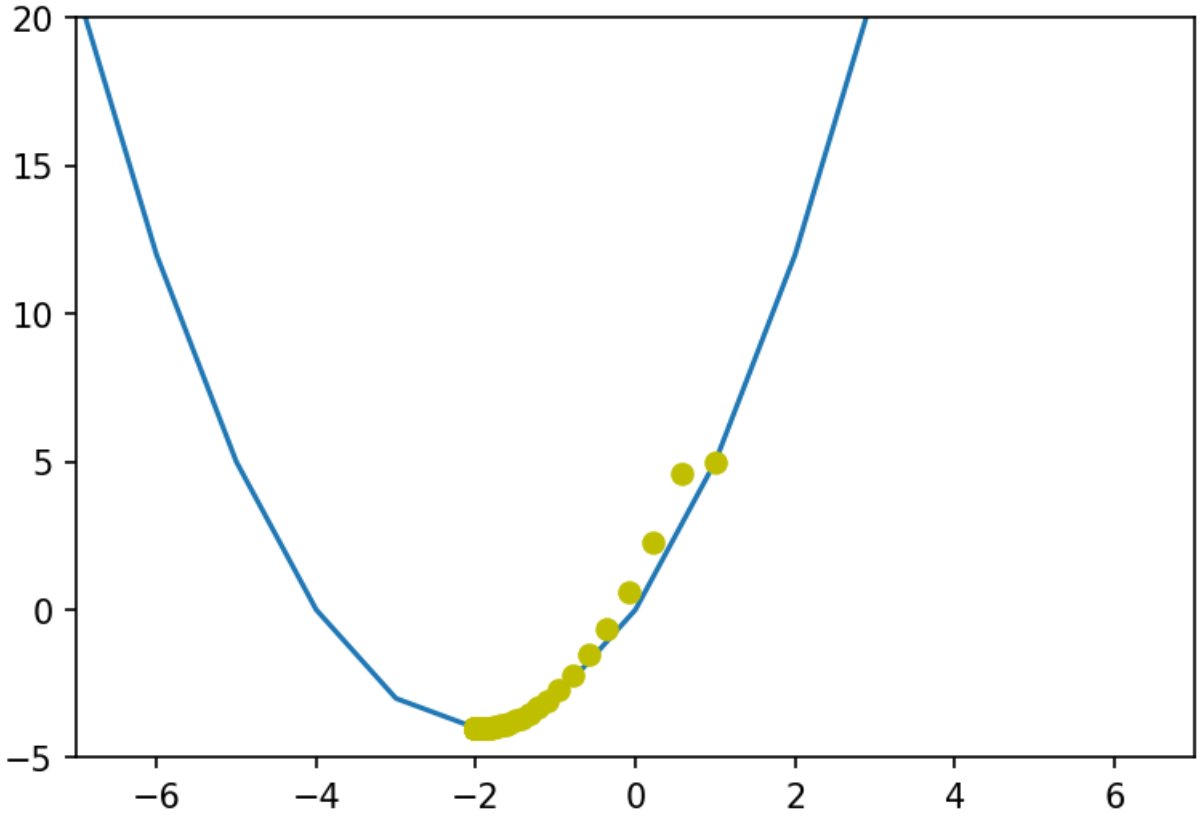
```
            X         Y     Pendiente
0    1.000000  5.000000  6.000000e+00
1    0.580000  4.580000  6.000000e+00
2    0.218800  2.295200  5.160000e+00
3   -0.091832  0.612441  4.437600e+00
4   -0.358976 -0.626038  3.816336e+00
..        ...       ...           ...
146 -2.000000 -4.000000  1.907334e-09
147 -2.000000 -4.000000  1.640307e-09
148 -2.000000 -4.000000  1.410664e-09
149 -2.000000 -4.000000  1.213171e-09
150 -2.000000 -4.000000  1.043327e-09

[151 rows x 3 columns]
```

Esto puede graficarse de la siguiente forma:

In [7]:
```
fig, ax = plt.subplots(1,1, figsize = figsize, dpi = dpi)
t1 = np.arange(-7,7,1)
ax.set_xlim(-7,7)
ax.set_ylim(-5, 20)
ax.plot(t1, f(t1))
ax.plot(x,y,'yo')
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0x1e002c18d08>]
```



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°.

In [ ]: