Practica 1

Neurona perceptrón (sin entrenamiento)

Seminario de solución de problemas de inteligencia artificial 2

Clave del curso: I7041

NRC: 124882

Calendario: 2023-A

Sección: D01

Martínez Sepúlveda Alan Jahir

216569127

INCO

12/02/2023

Introducción:

Para la realización de esta práctica se pide la simulación de una neurona perceptrón sin entrenamiento, en la cual se le den los pesos de w1 y w2, así como si bias correspondiente.

Desarrollo:

Como punto principal de partida, se desarrolla la practica en el lenguaje de programación de Python, el cual ofrece todas las características para las características de la neurona, así como la graficacion correspondiente.

```
class neurona:
    def __init__(self, w1, w2, b, dim):
        self.n = dim
        self.w = np.array([w1, w2]).reshape(dim, 1)
        self.b = b
```

(Modelo neurona)

Los parámetros para la neurona son los pesos ya mencionados en w1 y w2, así como el bias representado con una b, las dimensiones que se le darán. Los pesos en w se leeran de un archivo txt así como el bias.

Se utiliza la función reshape, la cual es una función que permite cambiar la forma de una matriz (array) en Numpy. Esta función toma como argumentos una matriz y las dimensiones deseadas para la nueva forma de la matriz, y devuelve una nueva matriz con las dimensiones especificadas.

A continuación se toman los pesos del archivo txt, y se leen de una forma que se puedan manipular para pasárselos como argumentos a nuestra neurona:

```
file = open("weights.txt")
lines = file.readlines()
file.close()

w1, w2, b = [float(value) for value in lines[0].strip().split(',')]

net = neurona(w1, w2, b, 2)
```

(Apertura archivo y lectura de datos)

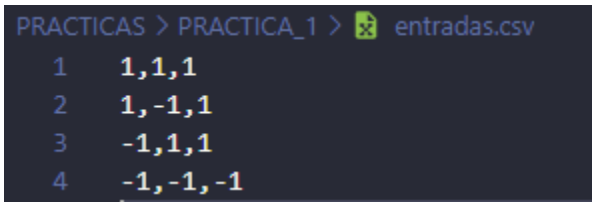
(Datos ingresados en archivo weights.txt)

Después de la lectura de los pesos y el bias, es necesario darle a la neurona las entradas, para sus correspondientes pesos en x así como en y, estas se introducen en un archivo .csv para su fácil iteración según su fila y columna.

```
for i in range(2):
    x = np.array(np.loadtxt("entradas.csv", delimiter=',', usecols=i))
    patterns.append(x)
X = np.array(patterns)

y = np.array(np.loadtxt("entradas.csv", delimiter=',', usecols=2))
```

(Iteración de entradas)

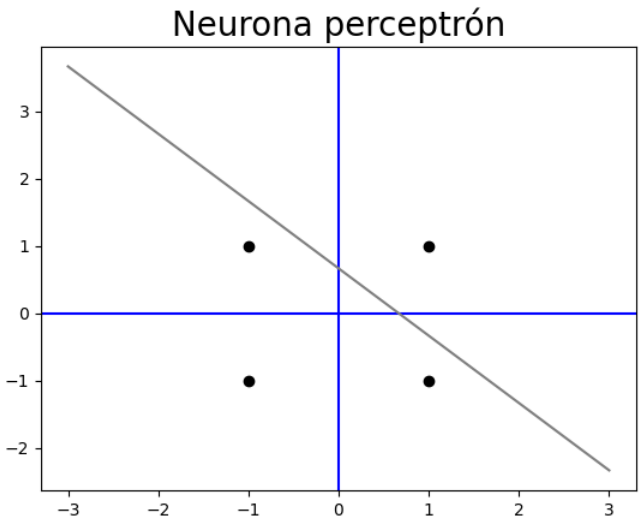


(Datos de entradas en .csv)

Como ultimo paso se grafican los resultados obtenidos, pasando los datos obtenidos de los pesos en w, así como su bias sacados en el archivo anteriormente iterado:

```
plt.clf()
plt.title("Neurona perceptrón", fontsize=20)
plt.scatter(X[0], X[1], color="black")
plt.axhline(color="blue")
plt.axvline(color="blue")
x_values = [-3,3]
y_values = [-(net.w[0][0]/net.w[1][0])*(-3) - (net.b / net.w[1][0]), -(net.w[0][0]/net.w[1][0])*(3) - (net.b / net.w[1][0])]
plt.plot(x_values, y_values, color="gray")
plt.savefig('grafica')
```

Finalmente se guarda la imagen como un png con el nombre de ‘grafica’ la cual corresponde a la siguiente:



Conclusión:

El código anterior es un ejemplo de un perceptrón. Se utiliza para resolver problemas de clasificación binaria, donde se espera que la salida sea 1 o -1. El código define una clase "neurona" que contiene los pesos y el bias inicializados desde un archivo de texto. Además, también se muestra cómo se cargan las etiquetas desde un archivo csv.

En si la estructura de la neurona es muy simple, ya que clasifica los valores según las entradas dadas, pero sin duda es una forma de ver como funciona desde la forma más básica el perceptrón, y que según vaya avanzando, la utilidad que se le

puede dar para ciertas situaciones puede ser muy satisfactoria, sobre todo si esta bien implementada ya que es capaz de clasificar los datos de una forma muy rápida.

Código:

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

class neurona:
    def __init__(self, w1, w2, b, dim):
        self.n = dim
        self.w = np.array([w1, w2]).reshape(dim, 1)
        self.b = b

file = open("weights.txt")
lines = file.readlines()
file.close()

def grafica(w_values):
    plt.clf()
    plt.scatter(X[0], X[1], color="black")
    plt.axhline(color="blue")
    plt.axvline(color="blue")
    x_values = [-3,3]
    y_values = [-(percep.w[0][0]/percep.w[1][0])*(-3) - (percep.b /
percep.w[1][0]),
                -(percep.w[0][0]/percep.w[1][0])*(3) - (percep.b /
percep.w[1][0])]
    plt.plot(x_values, y_values, color="gray")

w1, w2, b = [float(value) for value in lines[0].strip().split(',')]

percep = neurona(w1, w2, b, 2)

arreglo = []

for i in range(2):
    x = np.array(np.loadtxt("entradas.csv", delimiter=',', usecols=i))
    arreglo.append(x)
X = np.array(arreglo)

y = np.array(np.loadtxt("entradas.csv", delimiter=',', usecols=2))
```

```
plt.clf()
plt.title("Neurona perceptrón", fontsize=20)
plt.scatter(X[0], X[1], color="black")
plt.axhline(color="blue")
plt.axvline(color="blue")
x_values = [-3,3]
y_values = [-(percep.w[0][0]/percep.w[1][0])*(-3) - (percep.b /
percep.w[1][0]), -(percep.w[0][0]/percep.w[1][0])*(3) - (percep.b /
percep.w[1][0])]
plt.plot(x_values, y_values, color="gray")
plt.savefig('grafica')
```