Universidad Autónoma del Estado de Mexíco

Ingenieria en Computación

Materia: Redes Neuronales

Axel Valenzuela Juárez
12 de Octubre del 2019

1. Perceptrón Simple

- Vectores de entrada

$$X = \{x_i \in R^d, i = 1...N\}$$

- Vector de etiquetas

$$Y = \{y_i \in \{1, 0\}, i = 1...N\}$$

Figura 1: Entradas del Perceptrón.

Se necesita comprender el funcionamiento del perceptrón simple para empezar a programar, así que el primer paso fue revisar las presentaciones que nos proporcionó el profesor. Un perceptrón simple recibe 2 entradas, un vector de entrada y una vector de salida. Fig: 1

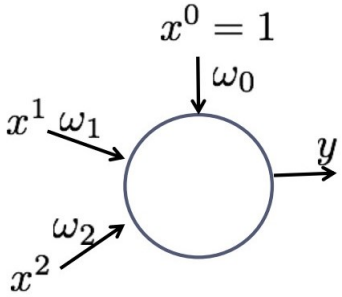

Figura 2: Representación de un perceptrón.

Para aplicar el perceptrón es necesario determinar los vectores de W, un perceptrón recibe dos entradas y un vector w para así obtener una salida, se puede ver la representación de un perceptrón a continuación. Fig: 2

$$y = \begin{cases} 1 \text{ si } \omega^T x_i \geq 0 \\ 0 \text{ otro caso} \end{cases}$$

Figura 3: Pormula para predecir el resultado de un perceptrón.

Para predecir la respuesta del perceptrón se ocupa la siguiente formula. Fig: 3

El algoritmo que ocupa el perceptrón es muy simple ya que se compone de dos for y un if y su respectivo else, se multiplica el vector W por el vector X, para poder multiplicarlos se necesita sacar la transpuesta de W, y después simplemente se compara el resultado, si el resultado es mayor que 0 se le asigna el valor de 1 a "y" y si es menor se le asigna un 0.

Lo interesante del algoritmo es la última parte ya que en esta es donde se decide la frontera de decisión más optima a base de los resultados de Y, el algoritmo está hecho para que la frontera de decisión se mueva según más convenga hasta obtener un resultado óptimo. Fig: 4

Input: ω inicial, X, Y, N: Número de iteraciones
Output: Neurona entrenada

```
for j=1:N do
    foreach x_i ∈ X do
        if ω^T x_i ≥ 0 then
            y=1
        else
            y=0
        ω = ω + (y_i − y)x_i
```

Figura 4: Algoritmo para implementar el Perceptrón Simple.

2. Codificación

El Primer paso en mi caso fue leer los datos de prueba, sacar los datos de la clase para tenerlos disponibles ya que se ocupan más adelante en el algoritmo. También separe los datos que en este caso serian X (x1,x2) para posteriormente agregarle una columna con números 1.

```
# -*- coding: utf-8 -*-
"""
Created on Sun Oct 13 20:45:35 2019

@author: Valenzuela Juarez Axel
"""

import numpy as np
import pandas as pd
class MyNeuron:
    def training(self):
        datos=pd.read_csv('training.csv')
        c = datos.iloc[:,2]#metemos la ultima columna
        x=datos.iloc[:,range(0,2)]#metemos todos los datos en X
        d=datos.iloc[0,range(0,2)]#metemos la primer fila en d
        k=int(np.round(len(d)))#numero de columna
        W=np.array(np.random.random((k+1,1)))#creacion de w con numero de columna
        Wt=np.transpose(W)
        b = np.ones((23,1))
```

Figura 5: Primer fragmento de codigo.

Como siguiente paso conté las columnas que existían esto para que el programa tenga la opción de funcionar independientemente de la cantidad de datos que se metan, el siguiente paso que decidí hacer fue crear N números aleatorios para el vector de W, en este caso fueron 3, ya que es el número de columnas más 1 e inmediatamente saque la transpuesta de W ya que así lo dicta el algoritmo. Fig: 5

Cree una matriz de números 1, para agregarse a los datos X y la agregue con la instrucción append. Después de realizar esto empecé a crear el algoritmo como lo muestra la Fig: 4

Se multiplica el vector W por el vector X con el resultado se compara en un if si es mayor a 0 se le asignara el valor de 1 a Y, si es menor a 0 se le asignara un 0 a Y. Esto se realiza

por cada fila que contenga X, al final el algoritmo corrige a W aumentándole o restándole valores.

```
            else
                y=0
            W=w+(y1-y)*x

        """

    #construct or
    def __init__(self,funcActivation):
        self.funcAct=funcActivation
    #Predice si aprueba o no
    def func(self):

        if self.funcAct=="sigmoid":
            return self.training()

    def datos(self,wf):

        datos=pd.read_csv('test.csv')
        x=datos.iloc[:,range(0,2)]#metemos todos los datos en X

        b = np.ones((35,1))
        l=np.append(b,x,axis = 1)#se agrego columna de 1 al principio en b
        l[1,range(0,3)]#metemos la primer fila en d

        for j in range(0,35):
            if(np.dot(wf,l[j,range(0,3)])>=0):
                y=1
                print ("Salida fila",j+1," : ",y)
            else:
                y=0
                print ("Salida fila",j+1," : ",y)#salidas finales

clf=MyNeuron("sigmoid")
#clf.func()

w=clf.func()
print("resfin: ",w)
clf.datos(w)
```

Figura 6: Uso de W para los datos de testing.

Una vez que obtuve el mejor valor de W lo mande a otra función para aplicarlo a los datos de test, para hacerlo utilice lo que ya había hecho antes, leer el archivo, obtener el total de filas y aumentar una columna de unos. Multiplique a W por el vector de X de los datos de testing, y simplemente imprimí a Y como lo muestra la Fig: 6

```
Archivo  Edición  Formato  Ver  Ayuda
X1,X2,Class
-0.59217,0.82799,1
-0.73041,0.61808,1
-0.43088,0.62391,1
-0.62442,0.37318,1
-0.41244,0.92711,1
-0.89171,0.8105,1
-0.39862,0.55977,1
-0.70276,0.47813,1
-0.78111,0.046647,1
-0.42627,0.20991,1
0.73502,-0.4898,0
0.67051,-0.29738,0
0.47696,-0.39067,0
0.31106,-0.49563,0
0.44931,-0.69388,0
0.62442,-0.5481,0
-0.052995,-0.69971,0
0.20968,-0.25656,0
0.58295,0.023324,0
0.11751,0.11662,0
-0.025346,-0.46064,0
0.19585,-0.77551,0
```

```
Terminal de IPython
Terminal 1/A
In [21]: runfile('C:/Users/Valenzuela/Documents/redes neuronales/perceptronaxel.py',
wdir='C:/Users/Valenzuela/Documents/redes neuronales')
resfin:  [[-0.51221808 -0.7257622   1.24200015]]
Salida fila 1  :  1
Salida fila 2  :  1
Salida fila 3  :  1
Salida fila 4  :  1
Salida fila 5  :  1
Salida fila 6  :  1
Salida fila 7  :  1
Salida fila 8  :  1
Salida fila 9  :  1
Salida fila 10  :  1
Salida fila 11  :  1
Salida fila 12  :  1
Salida fila 13  :  1
Salida fila 14  :  1
Salida fila 15  :  1
Salida fila 16  :  1
Salida fila 17  :  1
Salida fila 18  :  1
Salida fila 19  :  1
Salida fila 20  :  0
Salida fila 21  :  0
Salida fila 22  :  0
Salida fila 23  :  0
Salida fila 24  :  0
```

Figura 7: Comparación del Resultado de la Ejecución del Programa.

Para finalizar ejecute el programa y compare los resultados, los cuales se pueden observar en la Fig: 7

3. Conclusion

El perceptrón simple nos muestra como algo tan simple logra un gran resultado, específicamente en lograr obtener la mejor frontera de decisión para un caso en particular, esto

permite tener más autonomía ya que no se adivina ni se calcula por otra persona. En conclusión el perceptrón simple es un gran comienzo a la hora de adentrarnos a las redes neuronales, es simple, efectivo y demuestra que a veces menos es más.

4. Referencias

Kumar, N. (14 de Octubre de 2019). hackernoon.com. Obtenido de $https://hackernoon.com/implementing-the-perceptron-algorithm-from-scratch-in-python-48be2d07b1c0$

CHAU, L . A.2019.El perceptrón. Zumpango: INGENIERÍA EN COMPUTACIÓN.