Universidad Autónoma del Estado de Mexíco

Ingenieria en Computación

Materia: Redes Neuronales
Compuertas lógicas con perceptrón simple

Axel Valenzuela Juárez
Profesor: Dr. Asdrúbal López Chau
31 de Octubre del 2019

1. Introducción a redes multicapa

1.1. Compuertas Logicas

Los circuitos integrados digitales están formados por resistencias, diodos y transistores, montados en silicón u otro semiconductor conocido como sustrato. Cada uno de los integrados contiene varias compuertas lógicas. Las tres operaciones lógicas básicas son And Or y Not. Los símbolos que se utilizan para representar a cada una se pueden observar en la Fig:1

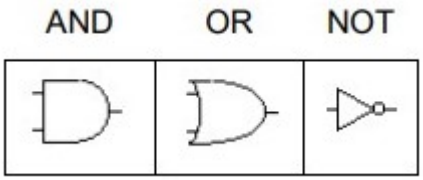

Figura 1: Compuertas And, Or ,Not.

ENTRADAS A	B	AND	NAND	OR	NOR	EXOR
0	0	0	1	0	1	0
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	0	1	0	0
funciones		A·B	$\overline{A \cdot B}$	A+B	$\overline{A+B}$	A⊕B

Figura 2: Tablas de verdad.

Cada compuerta tiene su tabla de verdad , la cual indica cuando habra voltaje y cuando no, siendo 1 voltaje y 0 sin voltaje, ver Fig:2

1.2. El perceptron

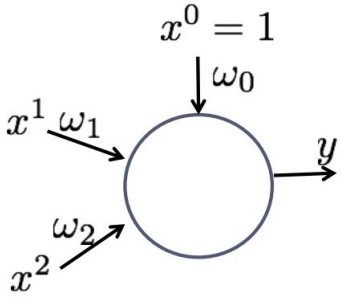

Figura 3: Representación de un perceptrón.

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: 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.

2. Desarrollo

```
import pandas as pd
#datos=pd.read_csv('training.csv')
#X=datos.iloc[:,[0,1]]
#Y=datos.iloc[:,2]
totalelementos=10
ceros=np.random.uniform(0,0.3,totalelementos)#vector de 10 ceros
unos= np.random.uniform(0.7,1.0,totalelementos)#vecto de 10 unos
numRen=ceros.shape[0]*4
#creo conjunto de datos
X=np.append(ceros,ceros)#se añaden elementos a la matriz
X=np.append(X,unos)
X=np.append(X,unos)
X=np.append(X,ceros)
X=np.append(X,unos)
X=np.append(X,ceros)
X=np.append(X,unos)
X.reshape(numRen,2,order=True)

Y=np.zeros((totalelementos*3,1))
Y=np.append(Y,np.ones((totalelementos,1)))
Y=Y.reshape(totalelementos*4,1)
clf.training(X,Y)
```

Figura 4: Creacion de datos de prueba.

Se uso el mismo algoritmo del perceptrón, solo se crearon un conjunto de datos por medio de np.ramdom.uniform, para ello se dio un rango de números a generar de 0 a 0.3 y de 0.7 a 1, los datos quedan ordenados de la manera que quedaban en el anterior ejercicio. Fig: 4

El algoritmo entrenara al perceptrón para que de como resultados sus tablas de verdad

Este mismo procedimiento se tiene que realizar para cada compuerta excepto en el caso de la compuerta not ya que en esa compuerta una entrada siempre será 0 para que simplemente se niegue.

3. Conclusion

En conclusión fue un gran avance lograr utilizar mÃºltiples neuronas para solucionar un problema como el de la xor, abriendo paso a un gran avance tecnológico, es necesario aprender a unir múltiples neuronas para comprender el funcionamiento de estas, hoy en día existen múltiples programas que pueden hacer esto de manera muy rápida pero siempre es importante comprender el funcionamiento de lo que hay detrás , para lograr innovar, al fin del día eso es lo que hace al ingeniero.

Tengo que destacar que este tema se me complico más de lo que pensaba por la falta de tiempo, y talvez comprensión a la hora de utilizar las compuertas , la manera en cómo funcionan las entradas de ellas y los datos a utilizar para el testing y entrenamiento.

4. Referencias

Fernández, J. L. (2003). Lógica Computacional. En Á. M. Riesco. UNED.

VEGA, J. L. (2012). Lógica Computacional. Estado de México: Red Tercer Milenio.