Universidad Autónoma del Estado de Mexíco

Ingenieria en Computación

Materia: Redes Neuronales
Red Neuronal Feed Forward

Axel Valenzuela Juárez
Profesor: Dr. Asdrúbal López Chau
12 de Noviembre del 2019

1. Introducción a Red Neuronal Feed Forward

1.1. Red Neuronal Feed Forward

Pueden clasificarse en distintas categorías.

En las redes feedforward se empieza con un vector de entradas el cual es equivalente en magnitud al número de neuronas de la primera capa de la red, las cuales procesan dicho vector elemento por elemento en paralelo.

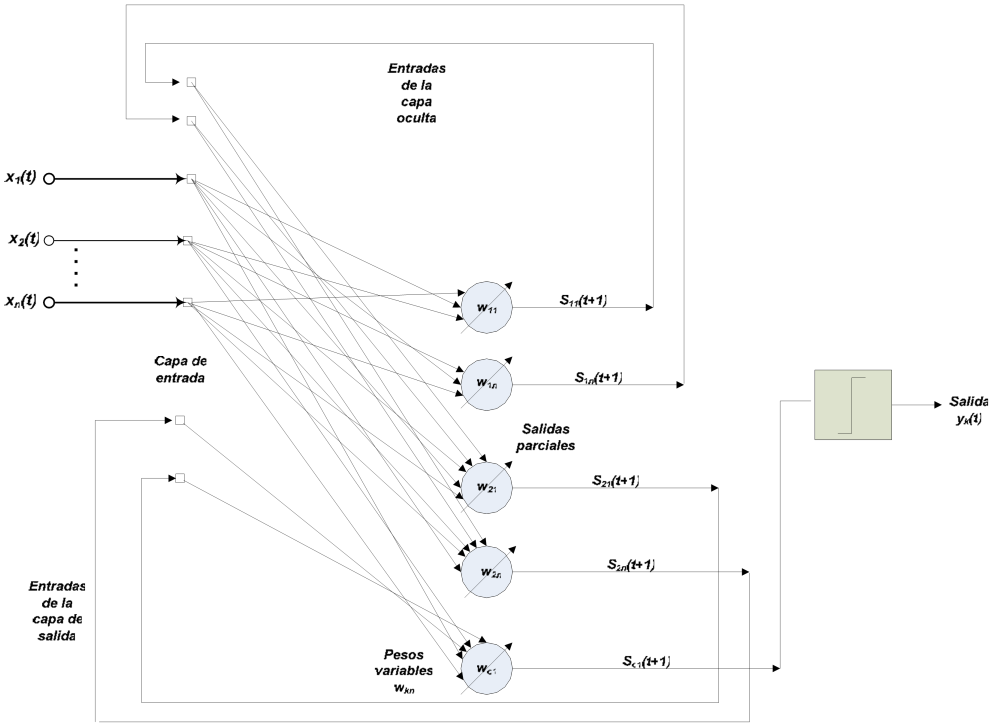

Figura 1: Red Feedforward.

La información, modificada por los factores multiplicativos de los pesos en cada neurona, es transmitida hacia delante por la red pasando por las capas ocultas para finalmente ser procesada por la capa de salida. Es por eso que este tipo de redes reciben su nombre. feed-forward son las más sencillas en cuanto a implementación y simulación, pero su desempenio es bueno para aplicaciones en los que no se requiera que la red retenga información de eventos pasados como ayuda para evaluar eventos futuros. en la Fig:1

1.2. Desarrollo

El desarrollo puede ser fácil de entender, es una red multicapa la cual queremos ocupar en múltiples casos, por lo cual se tiene que comprender el funcionamiento de una neurona anteriormente visto en otras prácticas. Se debe identificar la estructura de una red multicapa la cual está compuesta por una capa de entrada, una o múltiples capas ocultas y una capa de salida. En cada una de ellas se deberá sacar el resultado de la red con la siguiente ecuación: $Y = f(w^t(x))$

Se puede comenzar partiendo del codigo visto en clase de como hacer una red neuronal fija.

En este caso la primer modificación sera pedir al usuario los requisitos de la practica, siendo ,Número de entradas (valor entero) ,Número de capas ocultas (valor entero) ,Número de neuronas en cada capa oculta (lista de valores enteros) ,Número de salidas (valor entero).

Se crea un ciclo for el cual ayudara a crear las multiples capas que sean necesarias, se le agrego un mas 2 por la capa de entrada y salida, se usaron 3 if para determinar en que capa se encontrara el algoritmo, en la linea 25 se puede observar que se crea una matriz con números aleatorios para las entradas, posteriormente se le agregara un 1 a la matriz.

```
 6 """
 7 print ("Ingrese numero de entradas")
 8 nentradas=int(input())
 9 print ("Ingrese numero capas ocultas")
10 ncapocultas=int(input())
11 print ("Ingrese numero de neuronas en cada capa oculta")
12 nneuronasco=int(input())
13 print ("Ingrese numero de salidas")
14 nsalidas=int(input())
15
16 def funcAct(x):
17     return 1/(1+np.exp(-x))
18
19
20 import numpy as np
21 #arr = 5*np.random.random([1,nentradas])
22
23 for j in range(0,ncapocultas+2): #ciclo para crear las neuronas necesarias
24     if j==0:#caso para las entradas
25         arr = 5*np.random.random([1,nentradas])#se crea una matriz de numeros
26         #x=np.array([.5,1])
27         x_ext=np.append(1,arr)#se le añade un 1
28         W1 = 2.1*np.random.random([nneuronasco,nentradas+1])#se crea matriz w
29         print(W1) # se imprime para comprobar su funcionamiento
30         #W1 = np.random.randint(1,4,[2,3])
31         f=funcAct(np.dot(W1,x_ext))#se elabora la matriz transpuesta
32         f1 = np.append(1,f)#se añade un 1 a la matriz transpuesta
```

Figura 2: Previsualización de Codigo.

Se creara la matriz w en la linea 28, despues se calculara la matriz transpuesta entre w y los datos de entrada asi como se le pasara los datos para ser calculados en la funcion funcAct, como se puede observar en la Fig: 2

Obtendremos el resultado y se agregara un 1 a la matriz para ser ocupado posteriormente.

```
31         f=funcAct(np.dot(W1,x_ext))#se elabora la matriz transpuesta
32         f1 = np.append(1,f)#se añade un 1 a la matriz transpuesta
33         fy=f1#auxiliar
34     if j>0 and j==ncapocultas:#en caso de que no sea la primer iteracion
35         p=fy.shape #numero de filas o neuronas de la anterior iteracion
36         Wo=np.random.rand(nentradas+1,p[0])#creacion de w
37         print(Wo)
38         f=funcAct(np.dot(Wo,fy))#calculo de transpuesta
39         fq=np.append(1,f)
```

Figura 3: Codigo correspondiente a capa oculta.

El algoritmo pasara al caso de las capas ocultas, en esta parte del código que comienza desde la lÃnea 34 se realizaran los cálculos dependiendo el número de capas ocultas dado.

El algoritmo funciona muy parecido a la primer parte , solo que en este caso se tiene que tener presente los resultados de las neuronas anteriores para ello me apoye de la variable fy obteniendo el número de filas o neuronas, a partir de ese punto el procedimiento se repite, se creara una matriz w con números aleatorios y después se calculara la matriz transpuesta para ser pasada por la función funcAct, como se puede observar en la Fig: 3

```
nentradas=int(input())
print ("Ingrese numero capas ocultas")
ncapocultas=int(input())
print ("Ingrese numero de neuronas en cada capa oculta")
nneuronasco=int(input())
print ("Ingrese numero de salidas")
nsalidas=int(input())

def funcAct(x):
    return 1/(1+np.exp(-x))


import numpy as np
#arr = 5*np.random.random([1,nentradas])

for j in range(0,ncapocultas+2): #ciclo para crear las neuronas necesarias
    if j==0:#caso para las entradas
        arr = 5*np.random.random([1,nentradas])#se crea una matriz de numeros aleato
        #x=np.array([.5,1])
        x_ext=np.append(1,arr)#se le añade un 1
        W1 = 2.1*np.random.random([nneuronasco,nentradas+1])#se crea matriz w
        print(W1) # se imprime para comprobar su funcionamiento
        #W1 = np.random.randint(1,4,[2,3])
        f=funcAct(np.dot(W1,x_ext))#se elabora la matriz transpuesta
        f1 = np.append(1,f)#se añade un 1 a la matriz transpuesta
        fy=f1#auxiliar
    if j>0 and j==ncapocultas:#en caso de que no sea la primer iteracion
        p=fy.shape #numero de filas o neuronas de la anterior iteracion
        Wo=np.random.rand(nentradas+1,p[0])#creacion de w
        print(Wo)
        f=funcAct(np.dot(Wo,fy))#calculo de transpuesta
        fq=np.append(1,f)
    if j==ncapocultas+1:#en caso de ser la ultima iteracion
        h=fq.shape#numero de filas o neuronas de la anterior iteracion
        Wf=np.random.randn(nsalidas+1,h[0])#creacion de w
        y= funcAct(np.dot(Wf,fq))#calculo del resultado
        print("Resultado final",y)#impresion final
```

Figura 4: Codigo completo.

Para finalizar el código pasa a la sección de la capa de salida, para ello es nuevamente necesario apoyarse de una variable en este caso fq para saber la cantidad de neuronas en la anterior iteración, después de eso el algoritmo será el mismo , se calculara la matriz transpuesta y se pasara a la función funcAct para obtener el resultado final, el cual se imprimirá en la variable Y, como se puede observaren la Fig: 4

2. Pruebas Desarrolladas

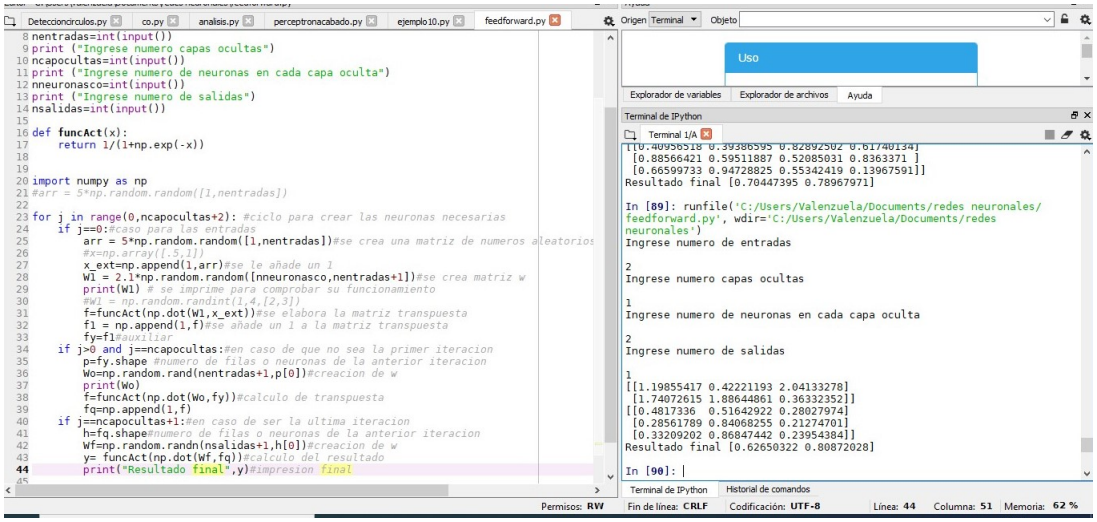

Figura 5: Prueba 1.

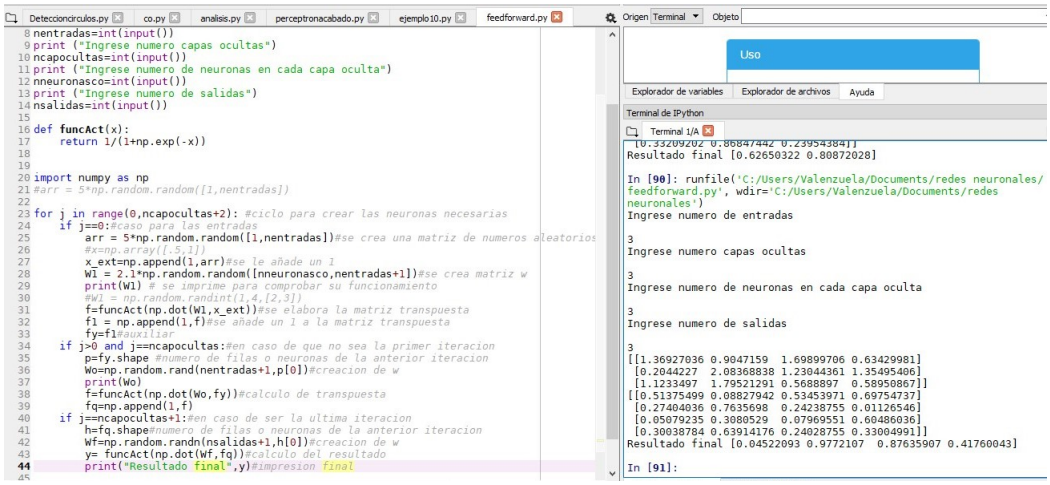

Figura 6: Prueba 2.

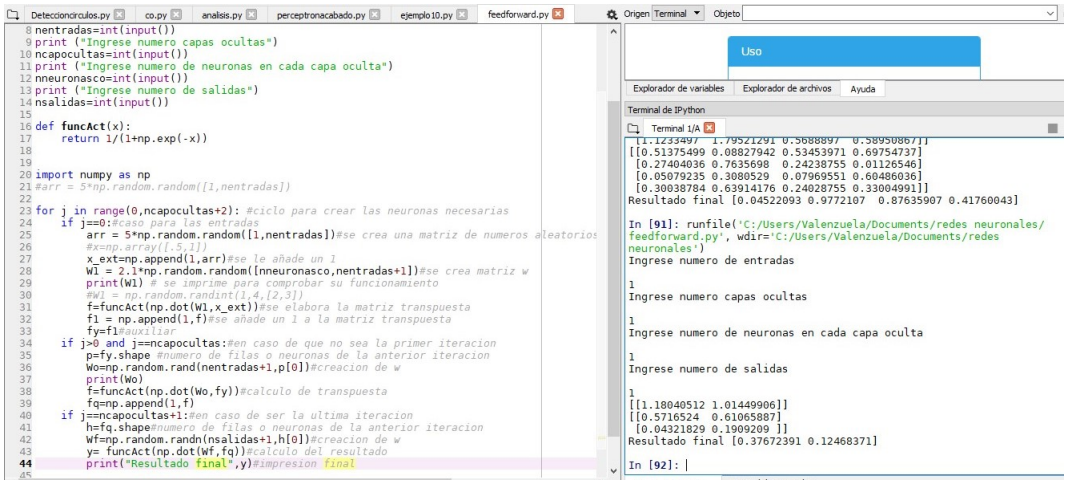

Figura 7: Prueba 3.

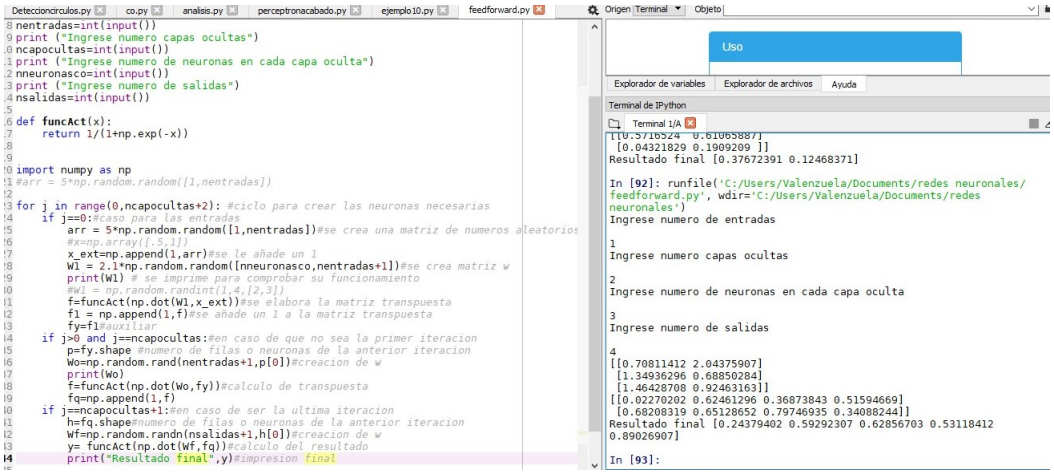

Figura 8: Prueba 4.

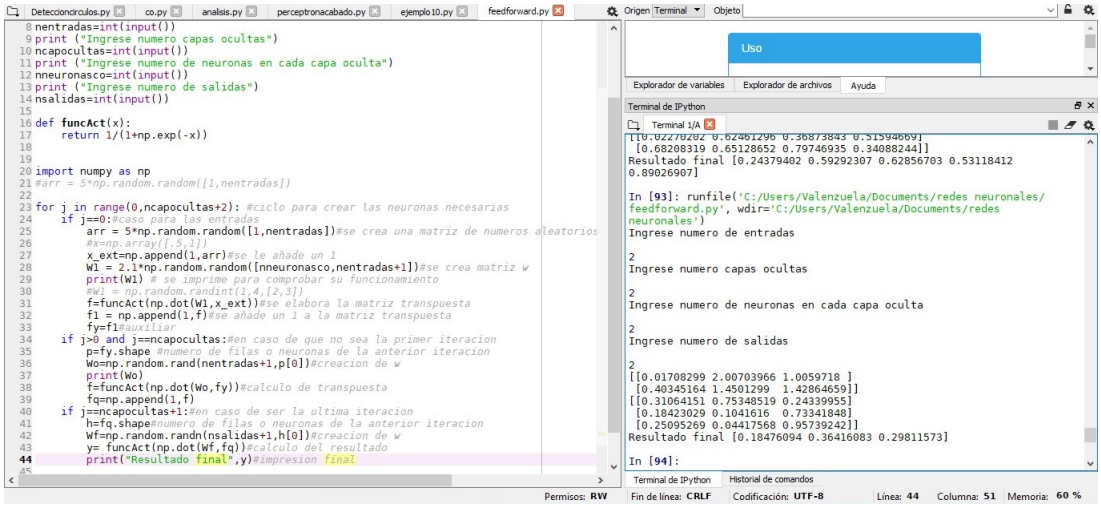

Figura 9: Prueba 5.

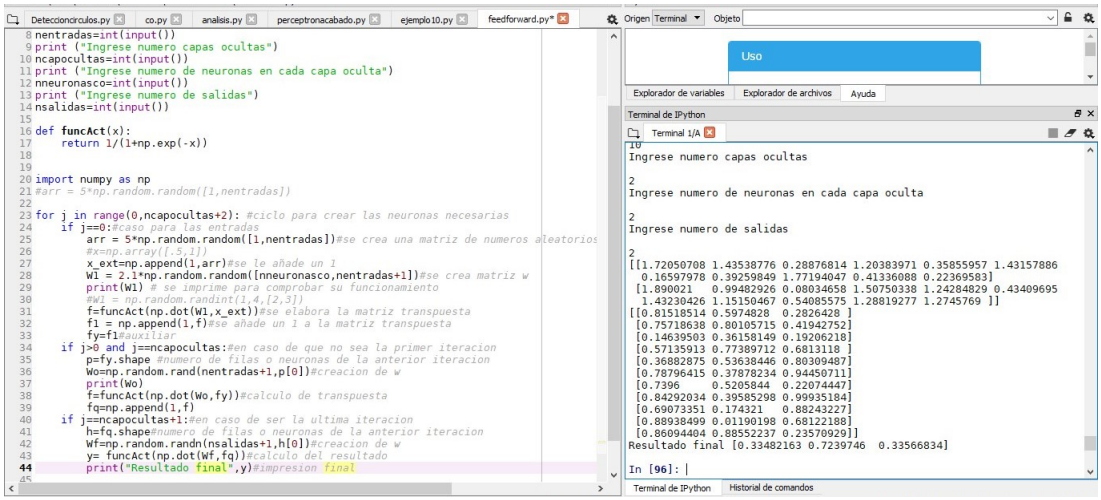

Figura 10: Prueba 6.

3. Conclusion

En conclusion el algoritmo de Feed Forward demuestra el potencial que tienen las redes multicapa, logrando analizar multiples clases y dando solucion a gran mayoria de ploblemas, Feed Forward es facil de programar e implementar.

Es de gran utilidad conocer el funcionamiento de este algoritmo ya que un ingeniero debe saber como funcionan las cosas para poder ser capaz de innovar y crear cosas nuevas.

4. Referencias

López, J. P. (2014). RED NEURONAL FEEDFORWARD. Sanfandila, Qro: Técnica No. 406.

Ortega, J. M. (2018). APLICACIÓN DE UNA RED NEURONAL . Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Internacional.