



UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

CENTRO UNIVERSITARIO UAEM ZUMPANGO

INGENIERÍA EN COMPUTACIÓN



REDES NEURONALES

Feed Forward

Alumno: Martín Márquez Cervantes

Profesor: Dr. Asdrúbal López Chau

Fecha de entrega: 12 de Noviembre 2019

Introducción

Para construir una red neuronal multicapa es necesario conocer la infraestructura que hay detrás y para ello en este laboratorio generamos las matrices necesarias para construir la arquitectura indicada para cada red neuronal multicapa con diferentes entradas, capas ocultas y capas de salida.

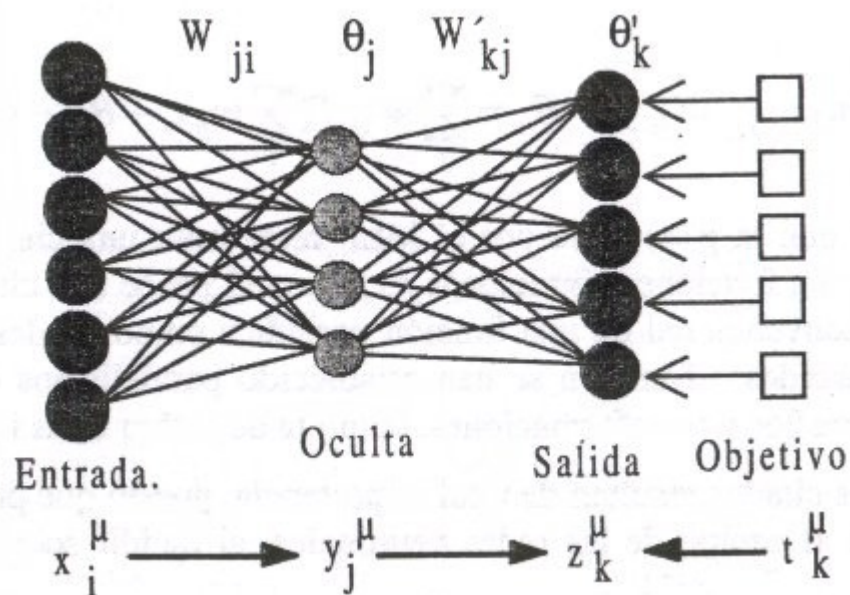


Figura 1 . Representación red neuronal multicapa

Desarrollo.

Para esta ocasión utilizaremos el código de la clase pasada el ejemplo_11.py para construir las matrices.

Importando numpy, creando la clase MyForwardNN con las funciones de activación y el método que se requiere, initNN.

```
18 #importamos numpy hasta el momento
19 import numpy as np
20 #crear clase MyForwardNN
21 class MyForwardNN:
22     #Funciones de activación
23     def heaviside(self,x):
24         if x>=0:
25             return 1
26         else:
27             return -1
28     def tanh(self,x):
29         return np.sinh(x)/np.cosh(x)
30     def sigmoid(self,x):
31         return 1/(1+np.exp(-x))
32     #def __init__():
33     #recibe numero de entradas,capas ocultas,numero neuronas en cada capa
34     #numero de salidas y funcion de activacion
35     """
36     b) Dentro del método, se deben de generar TODAS la
37     inicializándolas con valores pseudoaleatorios entre -1
38     distribución de probabilidad normal.
39     """
40     #metodo init que construye matrices
41     def initNN(self,inputs,HiddenLayer,neurons,outputs,funcActivation):
42         #Entradas
43         #xInput = np.random.randn(inputs,1)
```

Figura 2 Código fuente, clase y métodos..

Después nos creamos la instancia de la clase con `clf`, y definimos los parámetros como entradas, salidas, capas ocultas y neuronas de cada capa, este último en forma de lista para poder especificar las neuronas en cada capa en este ejemplo hace referencia a la primera capa oculta (2) la segunda capa oculta (4) y a la capa de salida (3), que se ocuparán para generar las matrices con el método `initNN`.

```
65 clf = MyForwardNN()
66 #inputs,hiddenlayers,neurons,outputs,funcActivacion
67 inputs = 2 #Numero de entradas
68 HiddenLayer = 2 #numero de capas ocultas
69 neurons = [2,4,3]
70 outputs = 3
71 funcActivacion = "Heaviside"
72 #llamamos al metodo initNN
73 clf.initNN(inputs,HiddenLayer,neurons,outputs,funcActivacion)
74
```

Figura 3 Instancia de la clase y parámetros.

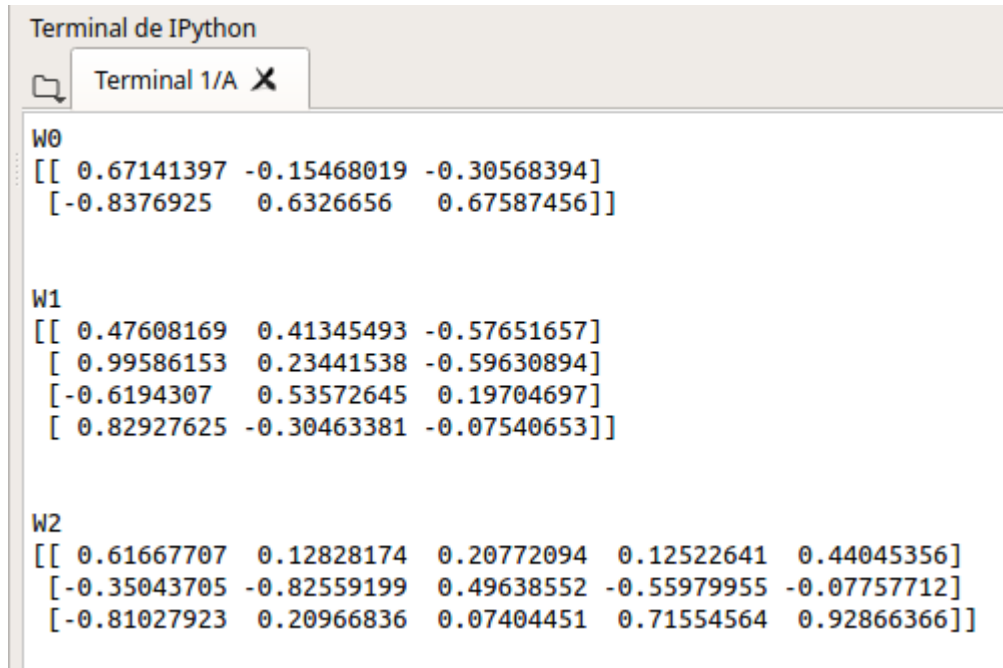
Después de mandar todos los parámetros al método, este los recibe

```
40 #metodo init que construye matrices
41 def initNN(self,inputs,HiddenLayer,neurons,outputs,funcActivacion):
42     #Entradas
43     #xInput = np.random.randn(inputs,1)
44     #x_Input = np.append(1,x)
45
46     #Crear matrices W
47     #lista W que contendrá matrices
48     W = []
49     #for para crear las matrices
50     for i in range(HiddenLayer+1):
51         #lista de arreglos
52         W.append( np.random.rand(neurons[i],inputs+1)*2.0-1.0 )
53         W[i].reshape((neurons[i],inputs+1))
54         #W[i] = np.append(1,W[i]) creo que debo añadir lista de ones
55         #W[i].reshape((neurons[i],inputs+1))
56         inputs = neurons[i]
57     #impresión de matrices
58     for i in range(HiddenLayer+1):
59         print("\n")
60         print("W"+str(i))
61         print(W[i])
62
```

Figura 4 Método `initNN`.

Implementando una lista y dentro las matrices, en el for generamos las matrices indicadas para cada indice, acomodamos y actualizamos el numero de entradas

Resultados y pruebas Desarrolladas.



```
Terminal de IPython
Terminal 1/A X

W0
[[ 0.67141397 -0.15468019 -0.30568394]
 [-0.8376925  0.6326656  0.67587456]]

W1
[[ 0.47608169  0.41345493 -0.57651657]
 [ 0.99586153  0.23441538 -0.59630894]
 [-0.6194307  0.53572645  0.19704697]
 [ 0.82927625 -0.30463381 -0.07540653]]

W2
[[ 0.61667707  0.12828174  0.20772094  0.12522641  0.44045356]
 [-0.35043705 -0.82559199  0.49638552 -0.55979955 -0.07757712]
 [-0.81027923  0.20966836  0.07404451  0.71554564  0.92866366]]
```

Figura 5 Matrices generadas.

Conclusiones.

Repase el tema que quedo un poco confuso y los reforcé aún más con la practica y descubrí nuevas formas de implementar un problema aun que sea básico.

Referencias.

https://www.ibiblio.org/pub/linux/docs/LuCaS/Presentaciones/200304curso-glisa/redes_neuronales/curso-glisa-redes_neuronales-html/x105.html

[decsai.ugr.es /pub /castro/Actividades/Redes-Neuronales/Apuntes](https://decsai.ugr.es/pub/castro/Actividades/Redes-Neuronales/Apuntes)

<https://disi.unal.edu.co/~lctorress/RedNeu/LiRna004.pdf>