



# Universidad Autónoma del Estado de México

## Ingeniería 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.

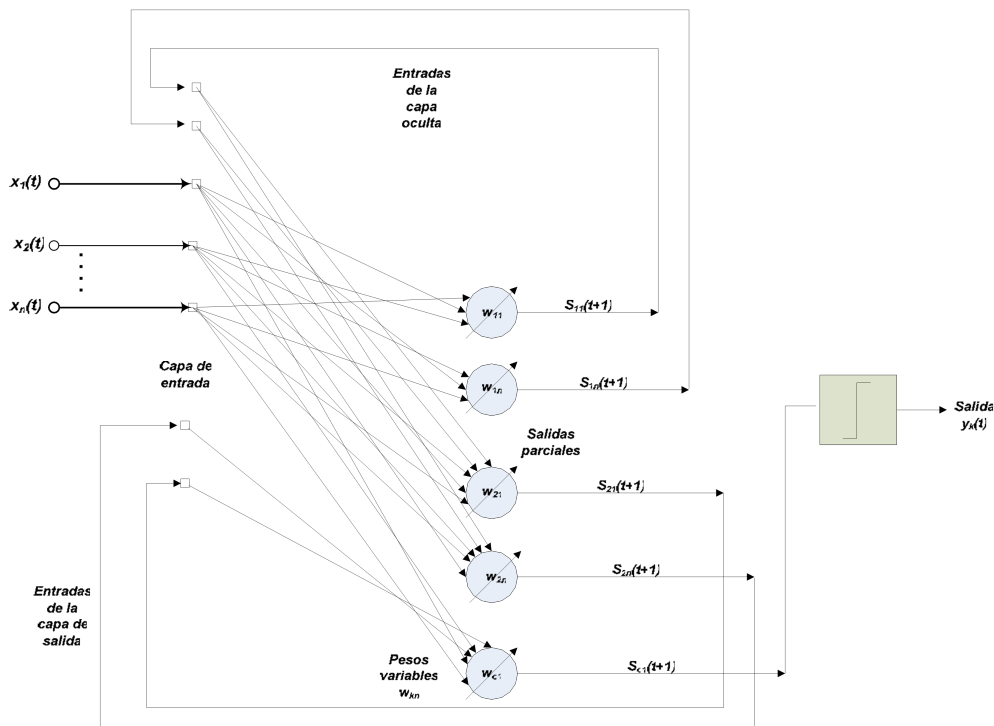


Figura 1: Red Feedforward.

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 desempeño 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 código 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 Código.

Se creara la matriz w en la linea 28, despues se calculara la matriz transpuesta entre w y los datos de entrada así 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))#calcula de transpuesta
39         fq=np.append(1,f)

```

Figura 3: Código correspondiente a capa oculta.

El algoritmo pasara al caso de las capas ocultas, en esta parte del código que comienza desde la línea 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

```

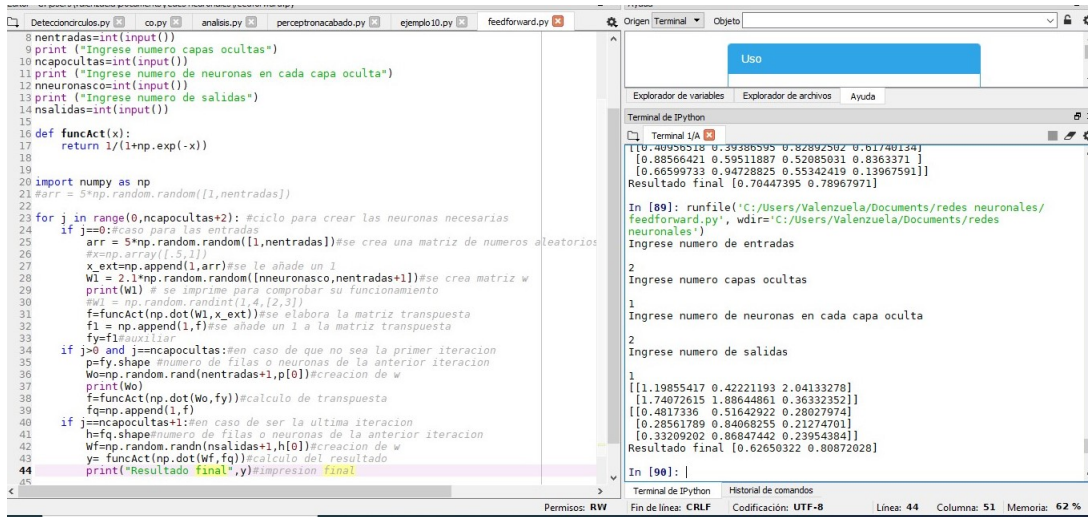
8 nentradas=int(input())
9 print ("Ingrese numero capas ocultas")
10 ncapocultas=int(input())
11 print ("Ingrese numero de neuronas en cada capa oculta")
12 nneuronasco=int(input())
13 print ("Ingrese numero de salidas")
14 nsalidas=int(input())
15
16 def funcAct(x):
17     return 1/(1+np.exp(-x))
18
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 aleatori
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
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)
40     if j==ncapocultas+1:#en caso de ser la ultima iteracion
41         h=fq.shape#numero de filas o neuronas de la anterior iteracion
42         Wf=np.random.randn(nsalidas+1,h[0])#creacion de w
43         y= funcAct(np.dot(Wf,fq))#calculo del resultado
44     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

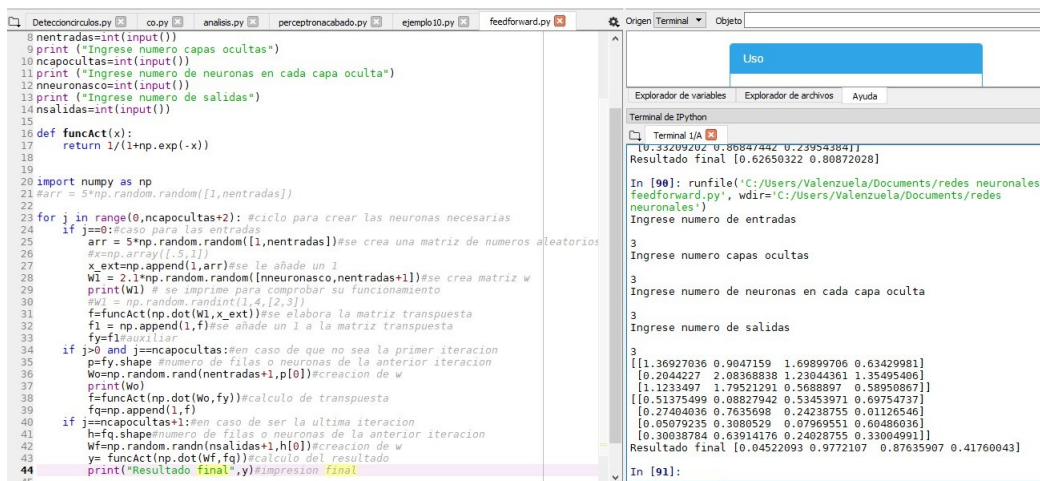


```
8 nentradas=int(input())
9 print ("Ingrese numero capas ocultas")
10 ncapocultas=int(input())
11 print ("Ingrese numero de neuronas en cada capa oculta")
12 nneuronasco=int(input())
13 print ("Ingrese numero de salidas")
14 nsalidas=int(input())
15
16 def funcAct(x):
17     return 1/(1+np.exp(-x))
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 aleatorios
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
33         #fy=f1#salidas
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))#calcula de transpuesta
39             fq=np.append(1,f)
40         if j==ncapocultas+1:#en caso de ser la ultima iteracion
41             h=fq.shape#numero de filas o neuronas de la anterior iteracion
42             Wf=np.random.randn(nsalidas+1,h[0])#creacion de w
43             y= funcAct(np.dot(Wf,fq))#calcula del resultado
44             print("Resultado final",y)#impresion final
45
```

Terminal de Python

```
In [89]: runfile('C:/Users/Valenzuela/Documents/redes neuronales/
feedforward.py', wdir='C:/Users/Valenzuela/Documents/redes
neuronales')
Ingreso numero de entradas
2
Ingreso numero capas ocultas
1
Ingreso numero de neuronas en cada capa oculta
2
Ingreso numero de salidas
1
[[[1.19855417 0.42221193 2.04133278]
[1.74072615 1.88644861 0.36323522]
[[0.4817336 0.51642922 0.28027974]
[0.28561789 0.84068255 0.21274701]
[0.33209202 0.80647442 0.23954384]]
Resultado final [0.62650322 0.80872028]
```

Figura 5: Prueba 1.



```
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 aleatorios
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
33         #fy=f1#salidas
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))#calcula de transpuesta
39             fq=np.append(1,f)
40         if j==ncapocultas+1:#en caso de ser la ultima iteracion
41             h=fq.shape#numero de filas o neuronas de la anterior iteracion
42             Wf=np.random.randn(nsalidas+1,h[0])#creacion de w
43             y= funcAct(np.dot(Wf,fq))#calcula del resultado
44             print("Resultado final",y)#impresion final
45
```

Terminal de Python

```
In [90]: runfile('C:/Users/Valenzuela/Documents/redes neuronales/
feedforward.py', wdir='C:/Users/Valenzuela/Documents/redes
neuronales')
Ingreso numero de entradas
3
Ingreso numero capas ocultas
3
Ingreso numero de neuronas en cada capa oculta
3
Ingreso numero de salidas
3
[[[1.36927036 0.9047159 1.69899706 0.63429981]
[0.2044227 2.08368838 1.23044361 1.35495406]
[[1.1233497 1.79521201 0.5688897 0.58950867]]
[[0.51375499 0.08827942 0.53453971 0.69754737]
[0.27404036 0.7635698 0.24238755 0.01126546]
[0.05079235 0.3080529 0.07969551 0.60486036]
[0.30038784 0.63914176 0.24028755 0.33004991]]
Resultado final [0.04522093 0.9772107 0.87635907 0.41760043]
```

Figura 6: Prueba 2.

```

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 aleatorios
26         #np.savetxt('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
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)) #calcula de transpuesta
39         fq=np.append(1,f)
40     if j==ncapocultas+1: #en caso de ser la ultima iteracion
41         h=fq.shape #numero de filas o neuronas de la anterior iteracion
42         wf=np.random.rand(nsalidas+1,h[0]) #creacion de w
43         y= funcAct(np.dot(wf,fq)) #calcula del resultado
44     print("Resultado final",y) #impresion final
45

```

Terminal de Python

```

In [91]: runfile('C:/Users/Valenzuela/Documents/redes neuronales/
feedforward.py', wdir='C:/Users/Valenzuela/Documents/redes
neuronales')
Ingreso numero de entradas
1
Ingreso numero capas ocultas
1
Ingreso numero de neuronas en cada capa oculta
1
Ingreso numero de salidas
1
[[1.18040512 1.01449906]]
[[0.5716524 0.61065887]
 [0.04321829 0.1909209 ]]
Resultado final [0.37672391 0.12468371]
In [92]:

```

Figura 7: Prueba 3.

```

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 aleatorios
26         #np.savetxt('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
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)) #calcula de transpuesta
39         fq=np.append(1,f)
40     if j==ncapocultas+1: #en caso de ser la ultima iteracion
41         h=fq.shape #numero de filas o neuronas de la anterior iteracion
42         wf=np.random.rand(nsalidas+1,h[0]) #creacion de w
43         y= funcAct(np.dot(wf,fq)) #calcula del resultado
44     print("Resultado final",y) #impresion final
45

```

Terminal de Python

```

In [92]: runfile('C:/Users/Valenzuela/Documents/redes neuronales/
feedforward.py', wdir='C:/Users/Valenzuela/Documents/redes
neuronales')
Ingreso numero de entradas
1
Ingreso numero capas ocultas
2
Ingreso numero de neuronas en cada capa oculta
3
Ingreso numero de salidas
4
[[0.70811412 2.04375907]
 [1.34936296 0.68850284]
 [1.46428708 0.92463163]]
[[0.62270202 0.62461296 0.36873843 0.51594669]
 [0.68208319 0.65128652 0.79746935 0.34088244]]
Resultado final [0.24379402 0.59292307 0.62856703 0.53118412
0.89026907]
In [93]:

```

Figura 8: Prueba 4.



```

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 aleatorios
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
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         W=np.random.randn(nentradas+1,p[0])#creacion de w
37         print(Wo)
38         f=funcAct(np.dot(Wo,fy))#calcula de transpuesta
39         fq=np.append(1,f)
40     if j==ncapocultas+1:#en caso de ser la ultima iteracion
41         h=fy.shape #numero de filas o neuronas de la anterior iteracion
42         wf=np.random.randn(nsalidas+1,h[0])#creacion de w
43         y= funcAct(np.dot(wf,fq))#calcula del resultado
44         print("Resultado final",y)#impresion final

```

Terminal de Python

```

In [93]: runfile('C:/Users/Valenzuela/Documents/redes neuronales/
feedforward.py', wdir='C:/Users/Valenzuela/Documents/redes
neuronales')
Ingreso numero de entradas
2
Ingreso numero capas ocultas
2
Ingreso numero de neuronas en cada capa oculta
2
Ingreso numero de salidas
2
[[[0.01708299 2.00703966 1.0059718 ]
[0.40345164 1.4501299 1.42864659]]
[[0.31064151 0.75348519 0.24339955]
[0.18423029 0.1041616 0.7341848]
[0.25095269 0.04417568 0.95739242]]
Resultado final [0.18476094 0.36416083 0.29811573]]
In [94]:

```

Figura 9: Prueba 5.

```

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 aleatorios
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
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         W=np.random.randn(nentradas+1,p[0])#creacion de w
37         print(Wo)
38         f=funcAct(np.dot(Wo,fy))#calcula de transpuesta
39         fq=np.append(1,f)
40     if j==ncapocultas+1:#en caso de ser la ultima iteracion
41         h=fy.shape #numero de filas o neuronas de la anterior iteracion
42         wf=np.random.randn(nsalidas+1,h[0])#creacion de w
43         y= funcAct(np.dot(wf,fq))#calcula del resultado
44         print("Resultado final",y)#impresion final

```

Terminal de Python

```

In [96]:
Ingreso numero capas ocultas
2
Ingreso numero de neuronas en cada capa oculta
2
Ingreso numero de salidas
2
[[[1.72050708 1.43538776 0.28876814 1.20383971 0.35855957 1.43157886
0.16597978 0.39259849 1.77194047 0.41336088 0.22369583]
[0.990021 0.99482926 0.08034658 1.50750338 1.24284829 0.43409695
1.43230426 1.15150467 0.54085575 1.28819277 1.2745769 ]]
[[0.81518514 0.5974828 0.2826428 ]
[0.75718638 0.80105715 0.41942752]
[0.36882875 0.53638446 0.80309487]
[0.78796415 0.37878234 0.94450711]
[0.7396 0.5295844 0.22074447]
[0.84292034 0.39585298 0.99935184]
[0.69073351 0.174321 0.88243227]
[0.88938499 0.01190198 0.68122188]
[0.86094404 0.88552237 0.23570929]]
Resultado final [0.33482163 0.7239746 0.33566834]
In [96]:

```

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 problemas, 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.