Construcción de una Red Neuronal en Python

Robles Martínez Karina

Universidad Nacional Autónoma de México
Facultad de Ciencias
Algoritmos Computacionales

karinarobles@ciencias.unam.mx

29 de Mayo de 2019

Índice

Introducción

¿Qué es una red neuronal?

La conexión entre neuronas tiene lugar gracias a la transmisión de impulsos nerviosos lo cual se llama sinapsis. La sinapsis se establece normalmente entre la parte terminalde un axón y el cuerpo o las dendritas de otra neurona.

La estructura sináptica esta formada por la membrana presináptica,la hedidura sináptica y la membrana postsináptica. [1]

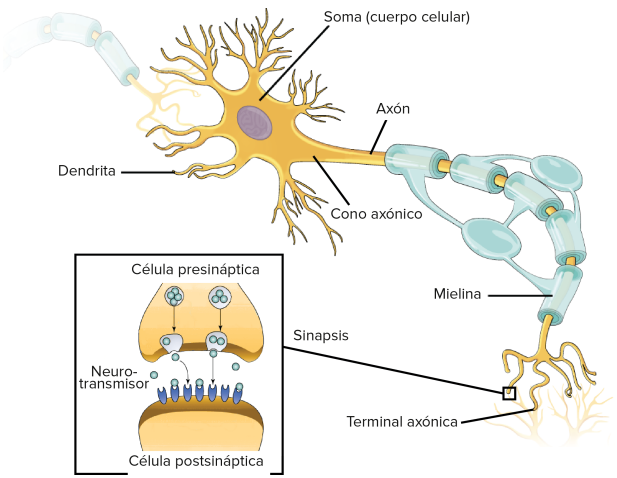


Figura: "Neurona"

Cada botón sináptico tiene depositos de neurotransmisoresque que son los encargados de pasar información de una neurona a otra, algunos de ellos son acetilcolina, dopamina, glutamato y glicina.

Una red neuronal biológica esta conformada por una serie de neuronas conectadas realizando el proceso de sinapsis nerviosa, pero si alguna de estas neuronas no se encuentra conectada, entonces la transmisión de información por medio del impuso nervioso no se puede llevar a cabo.

Introducción

¿Qué es una red neuronal?

Una red neuronal artificial (ANN) es un algoritmo que imita el funcionamiento de una red neuronal biológica en donde se requiere tener una señal de entrada y transformarla en una salida y así introducir una respuesta.

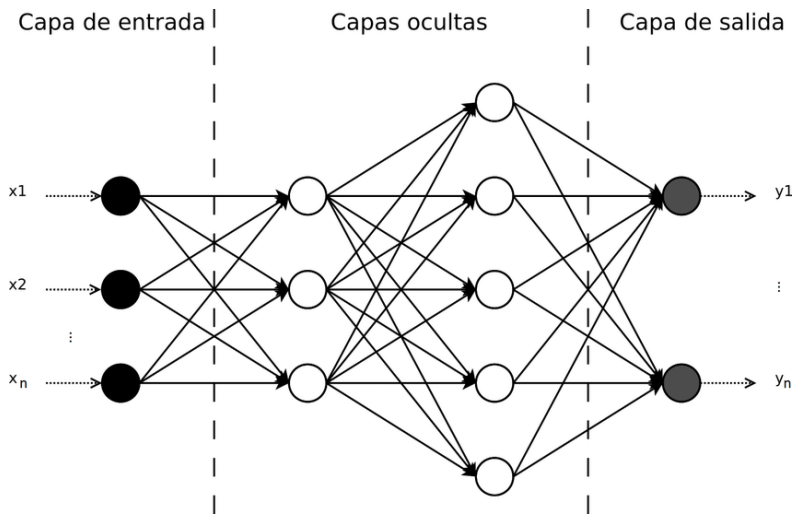
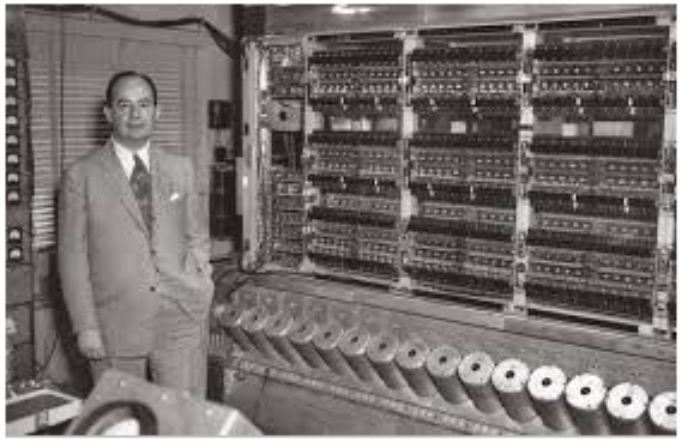


Figura: Red neuronal

Introducción

Breve historia de redes neuronales



1936 - Alan Turing.

Fue el primero en estudiar el cerebro como una forma de ver el mundo de la computación. Sin embargo, los primeros teóricos que concibieron los fundamentos de la computación neuronal fueron **Warren McCulloch, un neurofisiólogo, y Walter Pitts, un matemático**, quienes, en 1943, lanzaron una teoría acerca de la forma de trabajar de las neuronas (Un Cálculo Lógico de la Inminente Idea de la Actividad Nerviosa - Boletín de Matemática Biofísica 5: 115-133). Ellos modelaron una red neuronal simple mediante circuitos eléctricos.

1949 - Donald Hebb.

Construcción de la red neuronal

Arquitectura

```
import numpy as np

import matplotlib.pyplot as plt

from scipy import optimize
```

```
# X = (horas dormidas, horas estudiadas), y = calificacion
X = np.array(([2,5], [9,1], [6,10]), dtype=float)
Y = np.array(([70], [50], [90]), dtype=float)
```

```
X
```

```
array([[ 2.,  5.],
       [ 9.,  1.],
       [ 6., 10.]])
```

```
Y
```

```
array([[70.],
       [50.],
       [90.]])
```

Construcción de la red neuronal

Arquitectura

```
class Neural_Network(object):
    def __init__(self):
        self.NumNeuronasInput = 2 #numero de neuronas de la capa de entrada
        self.NumNeuronasOutput = 1#numero de neuronas de la capa de salida
        self.NumNeuronasHidden = 3  #numero de neuronas de la capa oculta

#PESOS

#Vamos a definir los pesos de manera aleatoria  con random  y con rand:nos genera ur
#pesos W1 sera los que estan entre la capa de entrada y la capa oculta

        self.W1 = np.random.rand(self.NumNeuronasInput, self.NumNeuronasHidden)

#pesos W2 seran los que estan entre la capa hidden y la de salida

        self.W2 = np.random.rand(self.NumNeuronasHidden, self.NumNeuronasOutput)

#X es la matriz de entrada
#W es la matriz de pesos
#Z sera el resultado de actividad neuronal
```

Figura: Código

Construcción de la red neuronal

Arquitectura

```
def avanzar (self,X):

    self.z2 = np.dot(X, self.W1)  #suma ponderada de XW1

    self.a2 = self.sigmoid(self.z2) #activacion de las neuronas de la capa 2

    self.z3 = np.dot(self.a2, self.W2) #resultado del todo este algoritmo va a
    yS = self.sigmoid(self.z3)

    return yS

def sigmoid(self,z):
#aplicar la activacion de la funcion sigmoide a un vector, escalar o matriz
    return 1/(1+np.exp(-z))
```

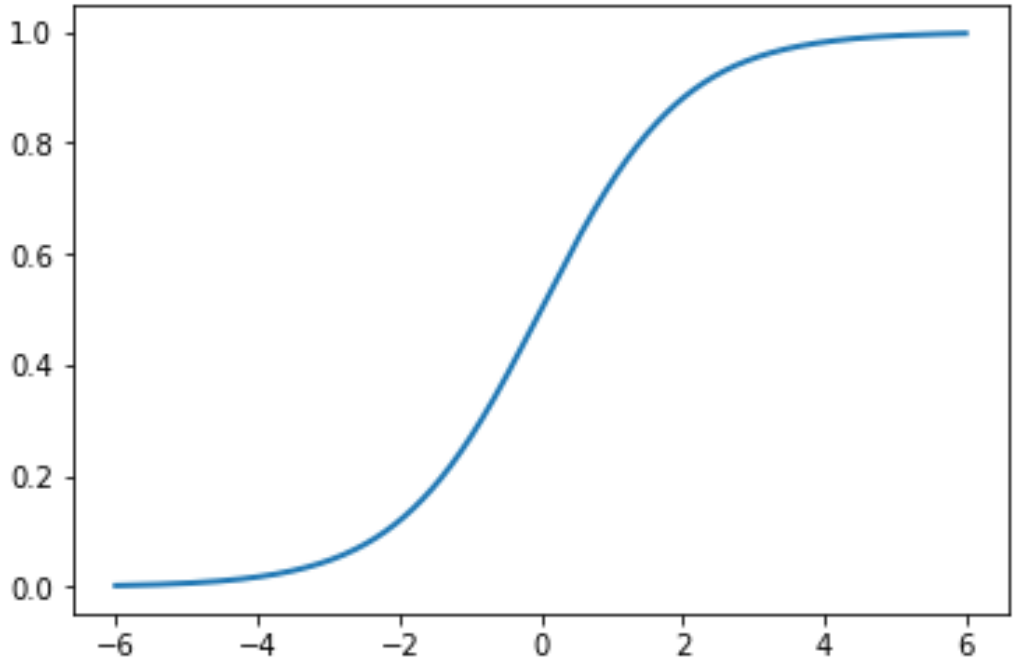
Figura: Código

Construcción de la red neuronal

Arquitectura

```
testInput = np.arange(-6,6,0.01)
plt.plot(testInput, sigmoid(testInput), linewidth= 2)
```

```
[<matplotlib.lines.Line2D at 0x7f4f51a00f28>]
```




```
sigmoid(6)
```

Construcción de la red neuronal

Gradientes de error

El algoritmo nos avienta resultados de manera aleatoria sin ningún proceso de aprendizaje, por lo que tenemos que investigar el ERROR que hay en este algoritmo en base a un resultado que nosotros ya tengamos (por ejemplo dormir 2 hrs, estudiar 5 hrs y sacar 7)

El calculo de este ERROR es el resultado deseado (Y) - el resultado obtenido del algoritmo (yS)

$$ERROR = ((Y - yS)^2) * \frac{1}{2}$$

Construcción de la red neuronal

Gradientes de error

COSTO = J, para no tener un problema multidimensional se minimizara este costo de manera tal que el algoritmo aprenda a modificarlo para que este sea mínimo.

Se utiliza la DERIVACIÓN PARCIAL para aquellas situaciones en las que sea más de un costo y la DERIVACIÓN NORMAL para aquellas situaciones en donde sea un solo costo.

Es necesario minimizar el costo del error provocado por los pesos,para esto es necesario corregirlos por el descenso del grandiente entonces es necesario derivar el costo con respecto al peso dC/dW

Construcción de la red neuronal

Gradientes de error

$\frac{\partial J}{\partial \dot{W}^2} = -(Y - yS)yS'$
$\frac{\partial J}{\partial \dot{W}^2} = -(Y - yS)F'(Z)Z'(W)$
$F'(Z) = \frac{e^{-z}}{(1+(e^{-z}))^2}$
$Z'(W) = A$; valores de activacion
$Z^3 = A^2W^2$

En el algoritmo queda :

$\frac{\partial J}{\partial \dot{W}^2} = (a^2)^T\delta^3$
$\delta^3 = -(Y - yS)F'(z^3)$

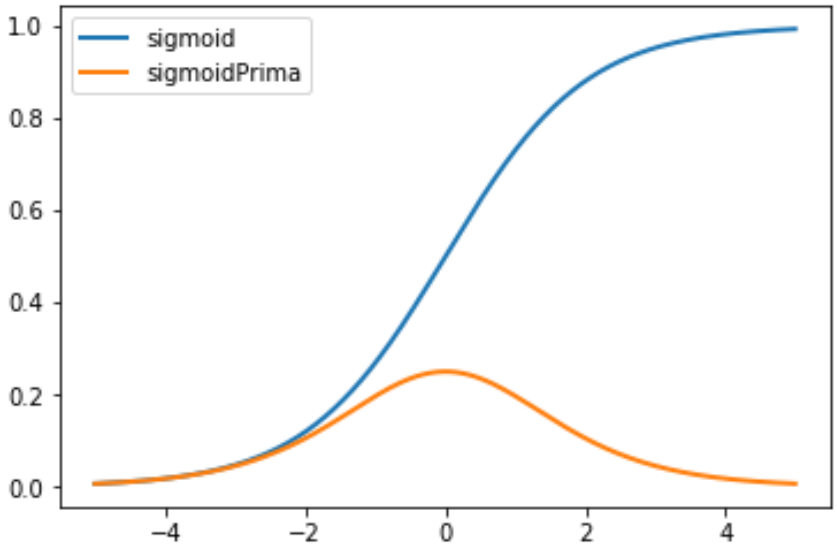
Corregimos de la capa oculta a la capa imput :

$\frac{\partial J}{\partial \dot{W}^1} = X^T\delta^2$
$\delta^2 = \delta^3(W^2)^T F'(z^2)$

delta3 * costos obtenidos de LA CAPA 2 Y LA CAPA DE SALIDA * LA FUNCIÓN sigmoidePrima evaluada con los valores de la suma ponderada de la capa 2

Construcción de la red neuronal

Gradientes de error




```
NN = Neural_Network()
```

```
cost1 = NN.costFunction(X,Y)
```

```
dJdW1, dJdW2 = NN.costFunctionPrima(X,Y)
```

```
dJdW1
```

```
array([[-0.34248659, -1.76098767, -0.56053561],
       [-0.67325209, -3.00868847, -0.70030406]])
```

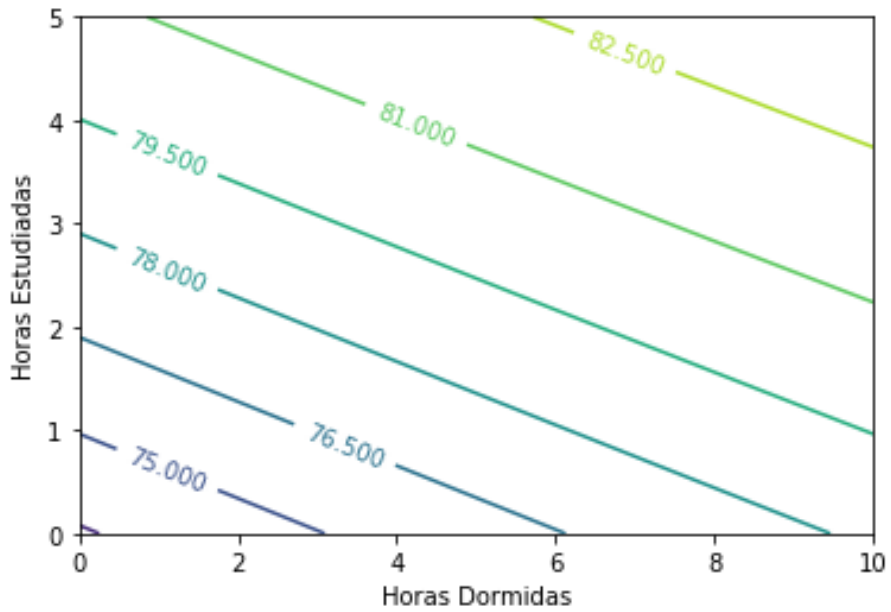
Grafica de horas dormidas, horas estudiadas y calificaciones

```
allOutputs = NN.avanzar(allInputs)
```

```
#GRAFICA DE contornos:
yy = np.dot(hrsEstudiadas.reshape(100,1), np.ones((1,100)))
xx = np.dot(hrsDormidas.reshape(100,1), np.ones((1,100))).T

CS = plt.contour(xx,yy,100*allOutputs.reshape(100, 100))
plt.clabel(CS, inline=1, fontsize=10)
plt.xlabel('Horas Dormidas')
plt.ylabel('Horas Estudiadas')
```

```
Text(0, 0.5, 'Horas Estudiadas')
```



Conclusión

Las redes de neuronas artificiales son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso.

Es un sistema de interconexión de neuronas que colaboran entre sí para producir un estímulo de salida.

References

[1]Saladin, Kenneth S. Anatom ´ıa y fisiolog ´ıa: la unidad entre forma yfunci ´on (6a. McGraw Hill M ´exico, 2013.

[2]Club de tecnología, Python: ¿Cómo construir una red neuronal simple?,en: http://www.clubdetecnologia.net/blog/2017/python-como-construir-una-red-neuronal-simple/.

[3] Robert Hecht Nielson, Neural Network Primer, Part I, Feb.1989

The End