Facultad: Ingeniería
Escuela: Computación
Asignatura: Sistemas Expertos e Inteligencia Artificial

Redes Neuronales Artificiales – El Perceptrón

Contenido

En la presente guía se introduce al estudiante al contexto de las Redes Neuronales Artificiales, elemento que es de suma utilidad para que los sistemas que lo contienen consigan un aprendizaje que puede ser supervisado o no supervisado. El aprendizaje se llevará a cabo a través de aplicación de los algoritmos propios de las RNA, ajustes y respectivas actualizaciones en los valores (pesos) que permitirán obtener las salidas deseadas.

Objetivos Específicos

- Conocer la definición de Rede Neuronal.
- Identificar los elementos que componen una neurona.
- Implementar la Red Neuronal "Perceptrón".

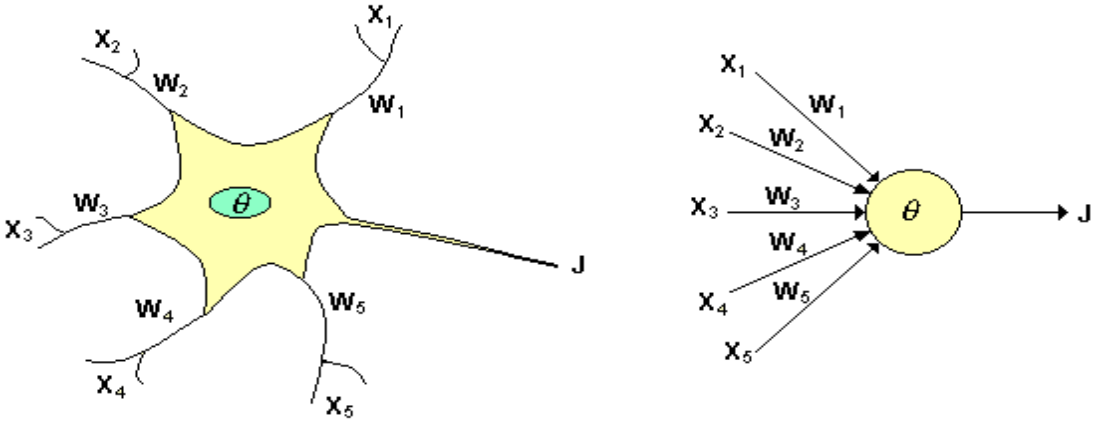
Material y Equipo

- Guía de laboratorio N° 9.
- Computadora con Python 3.6, PyCharm o navegador web.
- Dispositivo de almacenamiento.

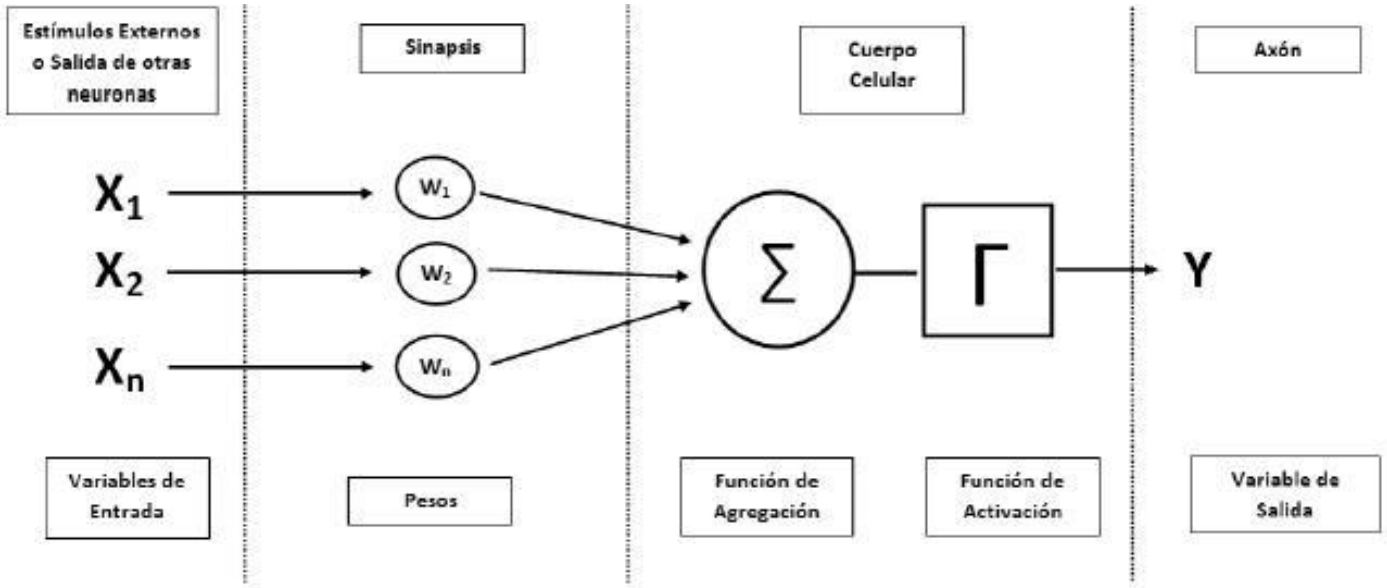
Introducción Teórica

Una red neuronal se compone de unidades llamadas neuronas. Cada neurona recibe una serie de entradas a través de interconexiones y emiten una salida.
Se caracterizan por ser sistemas desordenados capaces de guardar información.

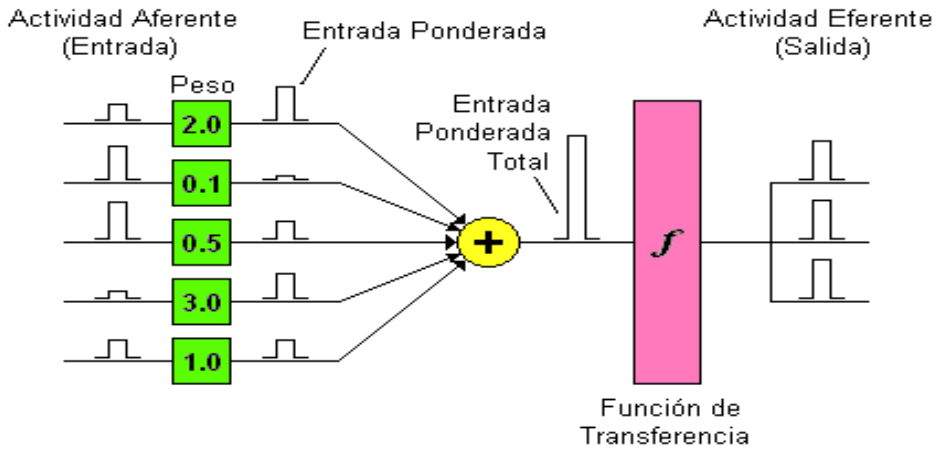
Neurona Biológica Vs Neurona Artificial



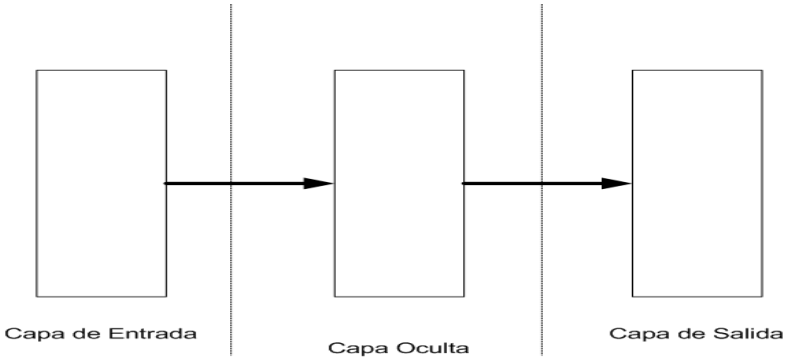
Partes de la Red Neuronal



- **Entradas:** señales que ingresan a la red neuronal, su valor puede variar dependiendo de la aplicación en la que se trabaje.
- **Salida:** indicador de salida de la red neuronal, el cual indica si la neurona está activa o no, por lo general esta salida proviene de una función de transferencia que limita la salida a un rango.
- **Pesos:** intensidad que conecta a dos neuronas. Memoria de aprendizaje de una neurona.

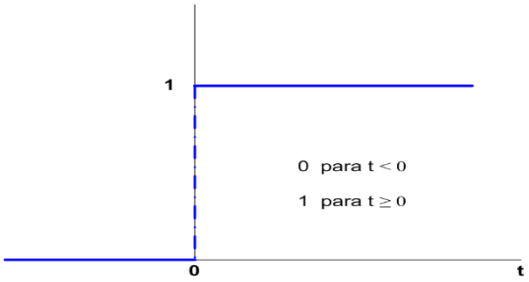


- **Capas:** representan las diferentes partes de la red neuronal multinivel.

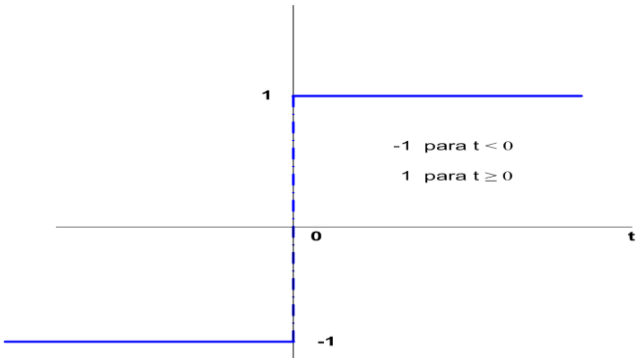


Funciones de Transferencia más usadas

Escalón unitario:



Escalón unitario bipolar:



Características de las Redes Neuronales Artificiales

- Robusto y tolerante a fallas.
- Es flexible, se ajusta a casos no aprendidos.
- Puede manejar información con ruido.
- Alta capacidad de procesamiento en paralelo.

Entrenamiento

El entrenamiento de una RNA (Red Neuronal Artificial) hace referencia a la búsqueda de los pesos, que, multiplicados con los valores de las entradas, proporcionan una salida deseada (patrón).

Caben mencionar que existen dos tipos de salida: salida deseada y salida esperada, en lo cual nuestro objetivo a la hora de entrenar una RNA es llegar a una salida deseada.

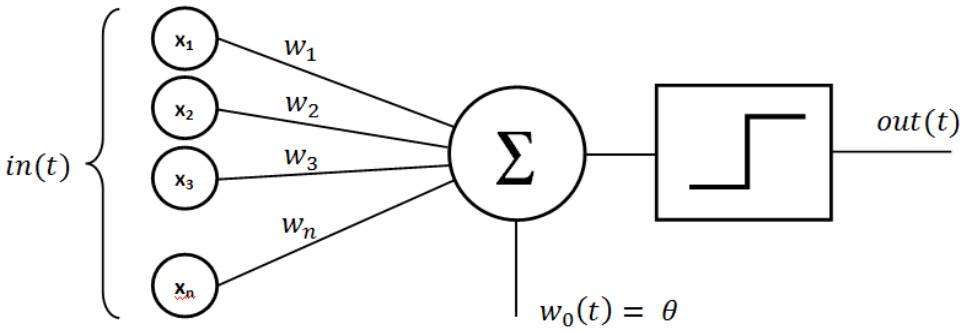
Para este procedimiento, se usan algunas fórmulas para calcular el valor de la salida, que luego es sometida a una función de activación que compara si el valor obtenido es igual al deseado. Además, utilizaremos una fórmula para realizar la actualización de los pesos, así como también del umbral.

Fórmula para calcular la salida	Fórmula para actualizar los pesos
$y = \begin{cases} 1, \text{ si } w_1x_1+\ldots+w_nx_n+\theta > 0 \\ -1, \text{ si } w_1x_1+\ldots+w_nx_n+\theta \leq 0 \end{cases}$	$\Delta W_{ij}(t) = W_{ij}(0) + 2\varepsilon(ti)x_j$

Para la fórmula de actualización de pesos:

- Wij es el peso actual.
- ε es el factor de aprendizaje.
- ti es la salida deseada.
- xi es la entrada actual.

Representación de gráfica de la RNA – Perceptrón



Algoritmo de aprendizaje del Perceptrón simple

1. Ajustar los pesos a cero y seleccionar un valor de ε (Factor de aprendizaje).
2. Se presenta un patrón de entrada.
3. Se determina la salida de la neurona.
4. Se pasa la salida por la función de transferencia.
5. Se compara la nueva salida con la salida deseada.
6. Si es igual se va al paso 8.
7. Si no se va al paso 10.
8. ¿Hay más patrones a aprender?, si la respuesta es sí, se va al paso 1.
9. Si la respuesta es no se termina el aprendizaje.
10. Se ajustan los pesos y se regresa al paso 1.

Aplicaciones más comunes del Perceptrón

- Procesamiento de imágenes y de voz.
- Reconocimiento de patrones.

- Planeamiento.
- Interfaces adaptivas para sistemas Hombre/máquina.
- Predicción.
- Control y optimización.
- Filtrado de señales.

Procedimiento

```
import random

#Función para actualizar los pesos en caso de no encontrar la salida deseada
def ActualizarPesos(PesoJ, Ti, Xi):
    #PesoActualizado = Peso_Actual+2*Salida_Deseada*Entrada_Actual
    PesoActualizado = PesoJ+2*0.5*Ti*Xi
    return PesoActualizado
#Iniciamos con una lista de valores para calcular los pesos aleatorios
#Pesos = [w1,w2,teta]
Pesos = [0,0,0]

#Obtenemos pesos aleatorios con la función ramdom()
for i in range(len(Pesos)):
    Pesos[i] = random.random()
    print('w',i+1,' = ',Pesos[i])

#Representación de la tabla de verdad de la compuerta lógica OR
Salidas_Esperadas = [1,1,1,-1]
Entradas = [[1,1,-1],[1,-1,-1],[-1,1,-1],[-1,-1,-1]]

Salida_Actual = 0
Contador = 0 #Para controlar el proceso
```

```
while i<len(Entradas):
    Salida_Actual = (Pesos[0]*Entradas[i][0]+Pesos[1]*Entradas[i][1]+Pesos[2]*Entradas[i][2])
    if Salida_Actual >= 0:
        Salida_Actual = 1
    else:
        Salida_Actual = -1
    if Salida_Actual == Salidas_Esperadas[i]:
        print('Salida Correcta',Salidas_Esperadas[i],' = ',Salida_Actual)
        print('-------------------------------------------------------')
    else:
        print('Salida Actual:',Salida_Actual, ' Salida Esperada: ', Salidas_Esperadas[i])
        print('Incorrecto, procedemos a la actualización de pesos...')
        for x in range(len(Pesos)):
            Pesos[x] = ActualizarPesos(Pesos[x],Salidas_Esperadas[i],Entradas[i][x])
            print(Pesos[x])
        print('-------------------------------------------------------')
        i=-1
    i=i+1
```

Verificar la salida.

Análisis de resultados

1. Modificar el ejemplo anterior, de manera que el Perceptrón sea capaz de aprender las salidas de una compuerta AND y XOR.

Investigación Complementaria

1. Desarrollar un programa en Python que permita seleccionar al usuario el tipo de red neuronal a entrenar. Se podrá seleccionar entre una compuerta AND, OR y XOR.

Bibliografía

- Inteligencia Artificial con Aplicaciones a la Ingeniería, Pedro Ponce Cruz.