Perceptron y red neuronal simple

Tinoco Sergio.
Instituto Politécnico Nacional, Escuela Superior de Cómputo.
Redes neuronales y aprendizaje profundo

22 de febrero de 2023

En el presente documento se van a presentar los resultados obtenidos en la primera práctica realizada en Google Colab.

Perceptrón.

Utilizando un perceptron con función de activación step se va a simular el comportamiento de una compuerta AND.

Para ello se asignaron los siguientes pesos:

```
weight1 = 0.8
weight2 = 0.8
bias = -1
```

Aplicando la suma de productos entre los pesos y los valores de entrada x que corresponden a todas las posibles entradas binarias (00, 01, 10, 11), se aplica la función escalón y se toman los que sean mayores o iguales a 0.

```
for test_input, correct_output in zip(test_inputs, correct_outputs):
    linear_combination = weight1 * test_input[0] + weight2 * test_input[1] + bias
    output = int(linear_combination >= 0)
    is_correct_string = 'Yes' if output == correct_output else 'No'
    outputs.append([test_input[0], test_input[1], linear_combination, output, is_correct_string])
```

Posteriormente se genera una tabla con los valores obtenidos y se verifica si es correcta.

Resultado:

```
Bien!  Acertaste todas.

 Input 1    Input 2    Linear Combination    Activation Output   Is Correct
       0          0                  -1.0                    0          Yes
       0          1                  -0.2                    0          Yes
       1          0                  -0.2                    0          Yes
       1          1                   0.6                    1          Yes
```

Red Neuronal Simple:

Para la red neuronal simple se van a implementar la función de activación sigmoide así como la simulación de la compuerta OR utilizando la función de activación escalón.

Primeramente se define la función ‘h’ que se encargará de realizar la suma de productos.

```
def function_h(X, W, b):
    suma = np.dot(X,W) + b
    return suma
```

Posteriormente se define la función de activación sigmoide.

```
def sigmoid(x):
    sg = 1/(1+(np.exp(-x)))
    return sg
```

Finalmente se definen los pesos y el sesgo para nuestra red neuronal simple.

```
inputs = np.array([0.7, -0.3])
weights = np.array([0.1, 0.8])
bias = -0.1

# Pasada frontal
h = function_h(inputs,weights,bias)
output = sigmoid(h)

print('Output:')
print(output)
```

Resultado:

```
Output:
0.4329070950345457
```

Posteriormente se define la función escalón para la compuerta OR.

```
def escalon(x):
  if(x>0):
    return 1
  else:
    return 0
```

Se definen los pesos y los valores de entrada por medio de 2 bucles for.

```
weights = np.array([1, 1])
bias = 0

for i in range(2):
  for j in range(2):
    inputs = np.array([i, j])

    # Pasada frontal
    h = function_h(inputs,weights,bias)
    output = escalon(h)

    print((i,j), "->",output)
```

Resultado

```
(0, 0) -> 0
(0, 1) -> 1
(1, 0) -> 1
(1, 1) -> 1
```