# Assignment 5

Elena Pérez-Ródenas Martínez
REDID: 827222533

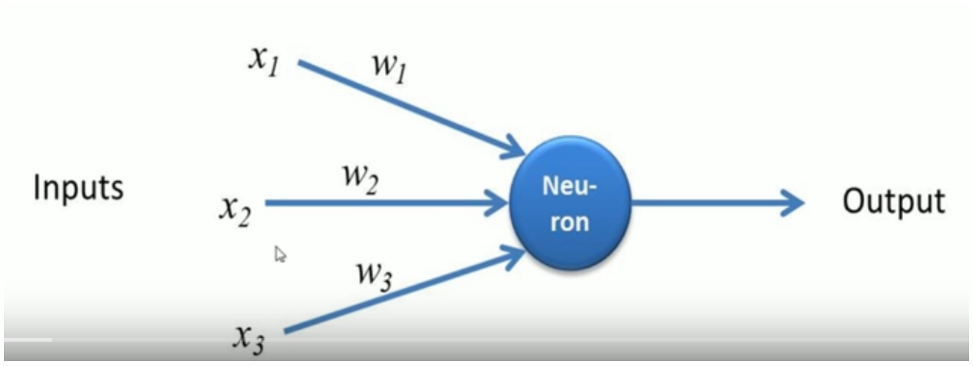
November 2021

*I, Elena Pérez-Ródenas Martínez, guarantee that this homework is my independent work and I have never copied any part from other resources. Also, I acknowledge and agree with the plagiarism penalty specified in the course syllabus.*

## 1. Exercise

### 1.1.

The perceptron is the basic unit of a neural network. It's a network that takes a number of inputs, carries out some processing on those inputs and produces an output. It has a single node called neuron. A neural network is a network that mimics the behavior of the network in the human brain.



$x_1$, $x_2$ and $x_3$ are the inputs and $w_1$, $w_2$ and $w_3$ refer to the width of the edges (weights).

The formula of the perceptron is the following one (being $y$ the output):

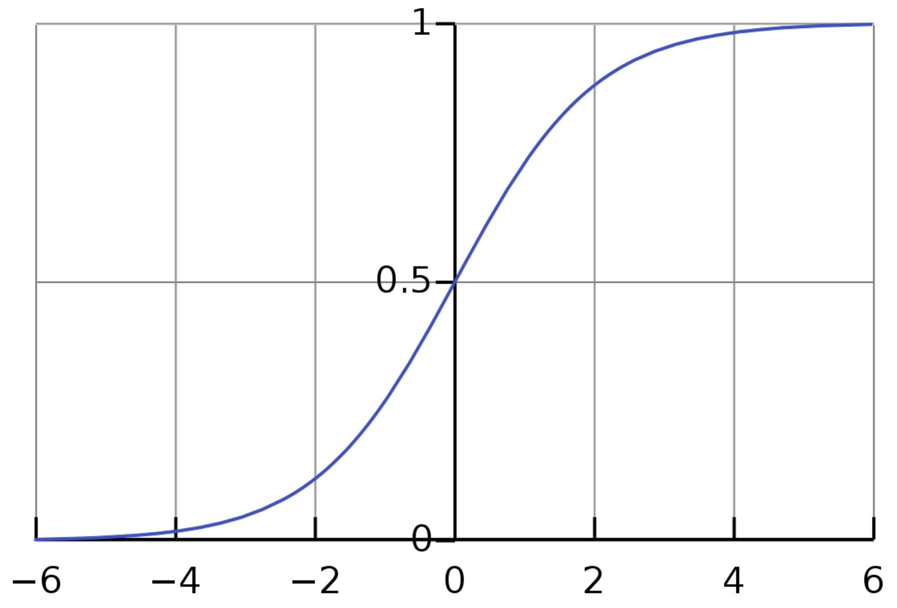
$y = x_1w_1 + x_2w_2 + x_3w_3$

But we may want our range to be between 0 and 1 or between -1 to $+1$. So we need an activation function to achieve this.

### 1.2.

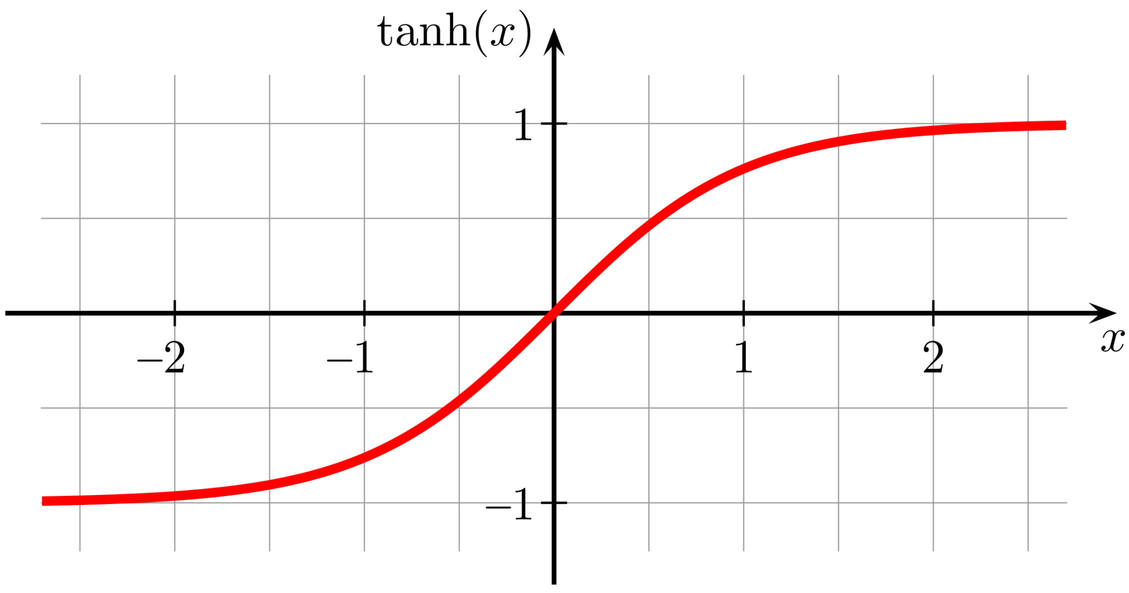
So the role of the activation functions in Neural Networks is to get an output in a specific range. The following ones are the most common:

**1. Logistic Function:** it produces outputs from 0 to 1. This is the graph for it and its formula:



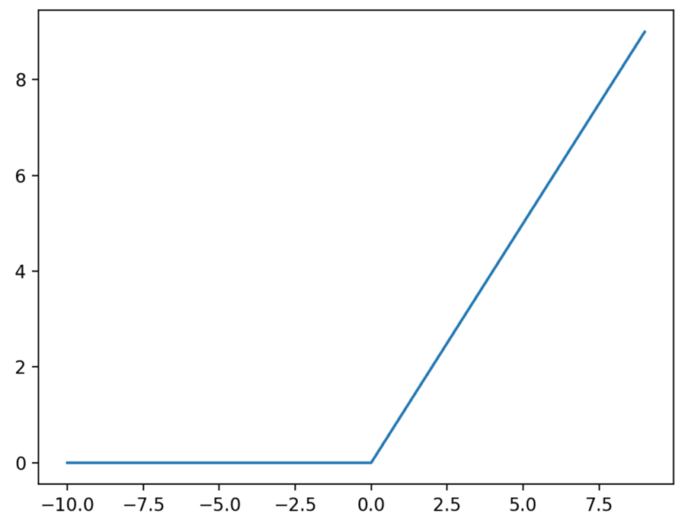
$f(x) = \frac{1}{1+e^{-\beta x}}$

**2. Hyperbolic Tangent Function:** it produces outputs from -1 to +1 and it has the following shape and formula:



$f(x) = tanh(x)$

**3. Rectified Linear Unit Function:** it produces outputs from 0 to infinity and has the following graph and formula:



$f(x) = max(0, x)$

So the modified formula for the perceptron will be:

$y = f(\sum_{i=1}^{n} w_i x_i)$ where f is the activation function. Another way to write this is:

$y = \phi(w_i x_i) = \phi(W^T X)$ where W=$(w_1, w_2, ..., w_n)^T$ and X=$(x_1, x_2, ..., x_n)$

And with the bias it would be $\phi(W^T X + b)$ [1]

## 1.3.

In forward propagation we have layers with the parameters $X^{l-1}$ for the input, $W$ and $b$ for the cache $Z$ and $X^l$ for the output. So in order to get the output we just have to apply the activation function g to WX+b as follows:

$Z^l = W^l X^{l-1} + B^l$

$X^l = g^l(Z^l)$

If we apply this formula to every layer in the neural network from the beggining we will be using forward propagation to get to the final output.

In backward propagation we have the input $dX^l$, and the outputs $dX^{l-1}, dW^l$ and $db^l$

$dZ^l = dX^l g^{l\prime}(Z^l)$

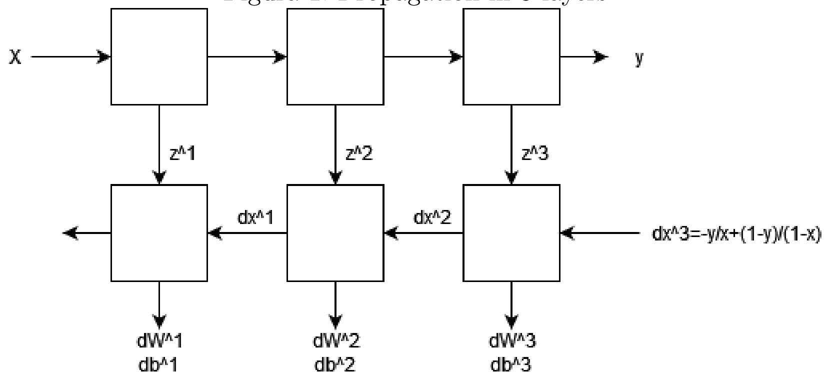
$dW^l = \frac{1}{m} dZ^l X^{l-1T}$

$db^l = \frac{1}{m} \sum(dZ^l)$

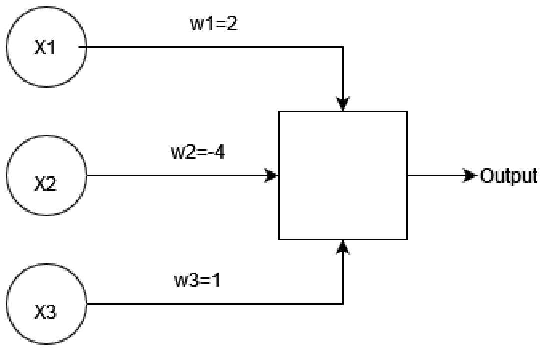
$dX^{l-1} = W^{lT} dZ^l$
[2]

To sum up, we may have an input and successive layers with different activation functions that end in a final output applying forward propagation. If we get that final output and apply backward propagation we will obtain the input we had in the first place.

Figura 1: Propagation in 3 layers



### 1.4.



P1: $v = X1w1 + X2w2 + X3w3 = 2$ so $\phi(v) = 1$

P2: $v = X1w1 + X2w2 + X3w3 = -4 + 1 = -3$ so $\phi(v) = 0$

P3: $v = X1w1 + X2w2 + X3w3 = 2 + 1 = 3$ so $\phi(v) = 1$

P4: $v = X1w1 + X2w2 + X3w3 = 2 - 4 + 1 = -1$ so $\phi(v) = 0$

### 1.5.

| $x_1$ | $x_2$ | sum | $h_\theta(x)$ |
|---|---|---|---|
| 0 | 0 | -30 | 0 |
| 0 | 1 | -10 | 0 |
| 1 | 0 | -10 | 0 |
| 1 | 1 | 10 | 1 |

So it's an AND gate.

## 2. Coding Project: Neural Network

What this program does is create a neural network, train it and test it with two test examples. Then it prints the predictions for both examples and plots each epoch vs the error plot.

```
NN = NeuralNetwork(inputs, outputs)
NN.train()

test_1 = np.array([[1, 1, 0]])
test_2 = np.array([[0, 1, 1]])

print(NN.predict(test_1), ' - Correct: ', test_1[0][0])
print(NN.predict(test_2), ' - Correct: ', test_2[0][0])

plt.figure(figsize=(15,5))
plt.plot(NN.epoch_list, NN.error_history)
plt.xlabel('Epoch')
plt.ylabel('Error')
plt.show()
```

The first task is to create the sigmoid activation function. The function receives x and a boolean that indicates if we have to return the derivative of the sigmoid function or the sigmoid function itself.

```
def sigmoid(self, x, deriv=False):
    der_sig = x * (1 - x)
    sig = 1 / (1 + np.exp(-x))
    if (deriv == True):
        return der_sig
    else:
        return sig
```

To allow the data to flow through the neural network we have to implement the function feed_forward. This function will set the result of the hidden layer by applying the sigmoid function to the inputs multiplied by the weights.

```
def feed_forward(self):
    inputs_dot_weights = np.dot(self.inputs, self.weights)
    self.hidden = self.sigmoid(inputs_dot_weights)
```

To go back through the layers of the neural network we have to complete the backpropagation function. It will determine the weights and error that contributed to the output and change them. First we calculate the error substracting the outputs minus the hidden layers. Then we calculate delta multiplying the error and the derivative of the hidden layer. Finally, we will add to the weights the inputs times delta.

```
def backpropagation(self):
    self.error  = self.outputs-self.hidden
    delta = self.error * self.sigmoid(self.hidden, deriv=True)
    self.weights += np.dot(self.inputs.T, delta)
```

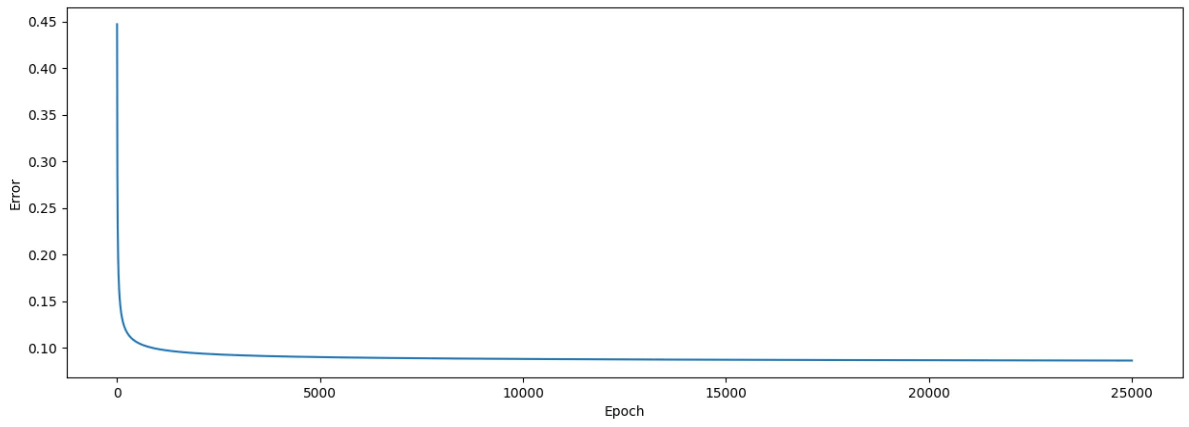
The train function trains the neural network for #epochs interactions. For each epoch we will flow forward the network to get the hidden layers and then go back again to make the corresponding corrections saving the different errors. We will also update the list of epochs and errors (average of the absolute error).

```
def train(self, epochs=25000):
    for epoch in range(epochs):
        self.feed_forward()
        self.backpropagation()
        self.error_history.append(np.average(np.abs(self.error)))
        self.epoch_list.append(epoch)
```

After completing all the tasks we obtain the following predictions and plot:

```
[[0.99089925]]  - Correct:  1
[[0.006409]]  - Correct:  0
```

We know that the first number in the input determines the output. The first test has a 1 in the first column, and therefore the output should be a 1. However, the second example has a 0 in the first column, and so the output is a 0.



The plot shows that there is a huge decrease in error during the earlier epochs, but that the error slightly plains after approximately 5000 iterations.

## 3. Referencias

[1] https://www.youtube.com/watch?v=RNYT9bECfOo.

[2] https://www.youtube.com/watch?v=qzPQ8cEsVK8.