ARTIFICIAL NEURAL NETWORK

in Python LANGUAGE

Chapter 2.2: Activation functions

3. Activation functions

- **Role of activation functions:**
  - Activation functions are used in order to activate the signal at the output of a neuron.
  - 2 types of activation function: Activation functions at hidden layers and activations function for the output function.
  - When one have to map with non-linear functions, non-linear activation functions (such as sigmoid / softmax / ReLU can be used).

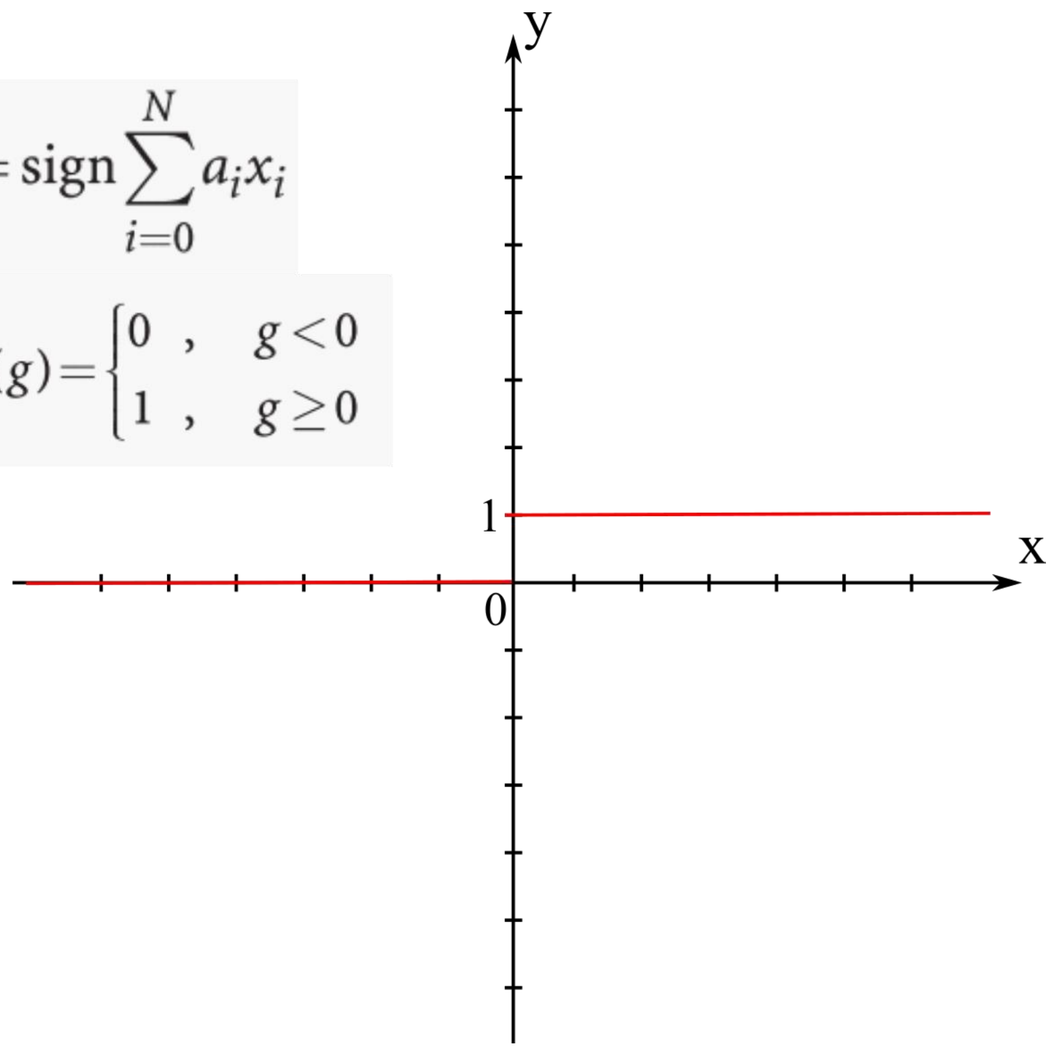
3. Activation functions

- **Step activation function:**

Simplest activation function. Not in used today anymore.

$$y = \text{sign} \sum_{i=0}^{N} a_i x_i$$

$$\text{sign}(g) = \begin{cases} 0 , & g < 0 \\ 1 , & g \geq 0 \end{cases}$$



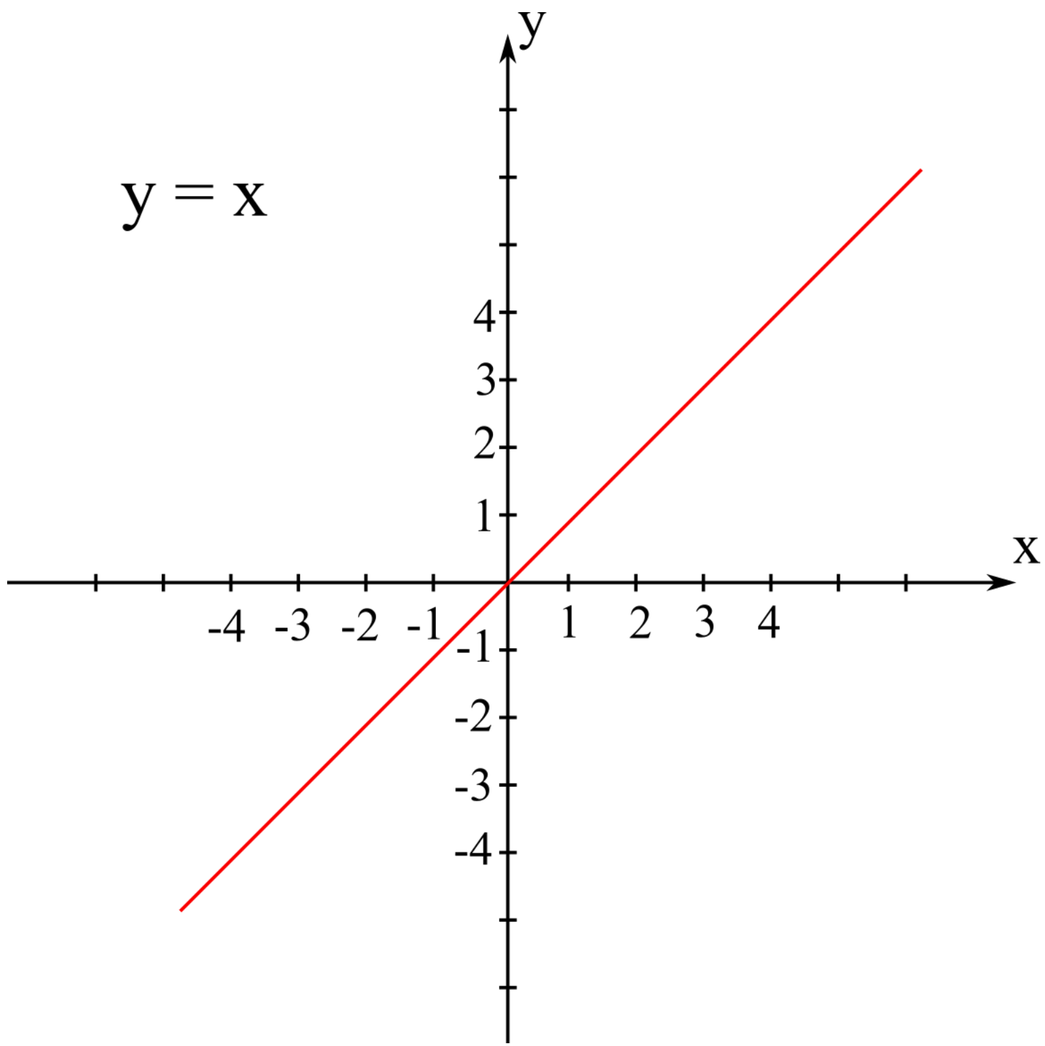
3. Activation functions

- **Linear activation function:**

The linear activation is generally used at the output layer for regression applications.

```
#Activation Linear function

class Activation_Linear:
    # Forward Pass
    def forward(self,inputs):
        #Calculate output values from input
        self.output = inputs
```



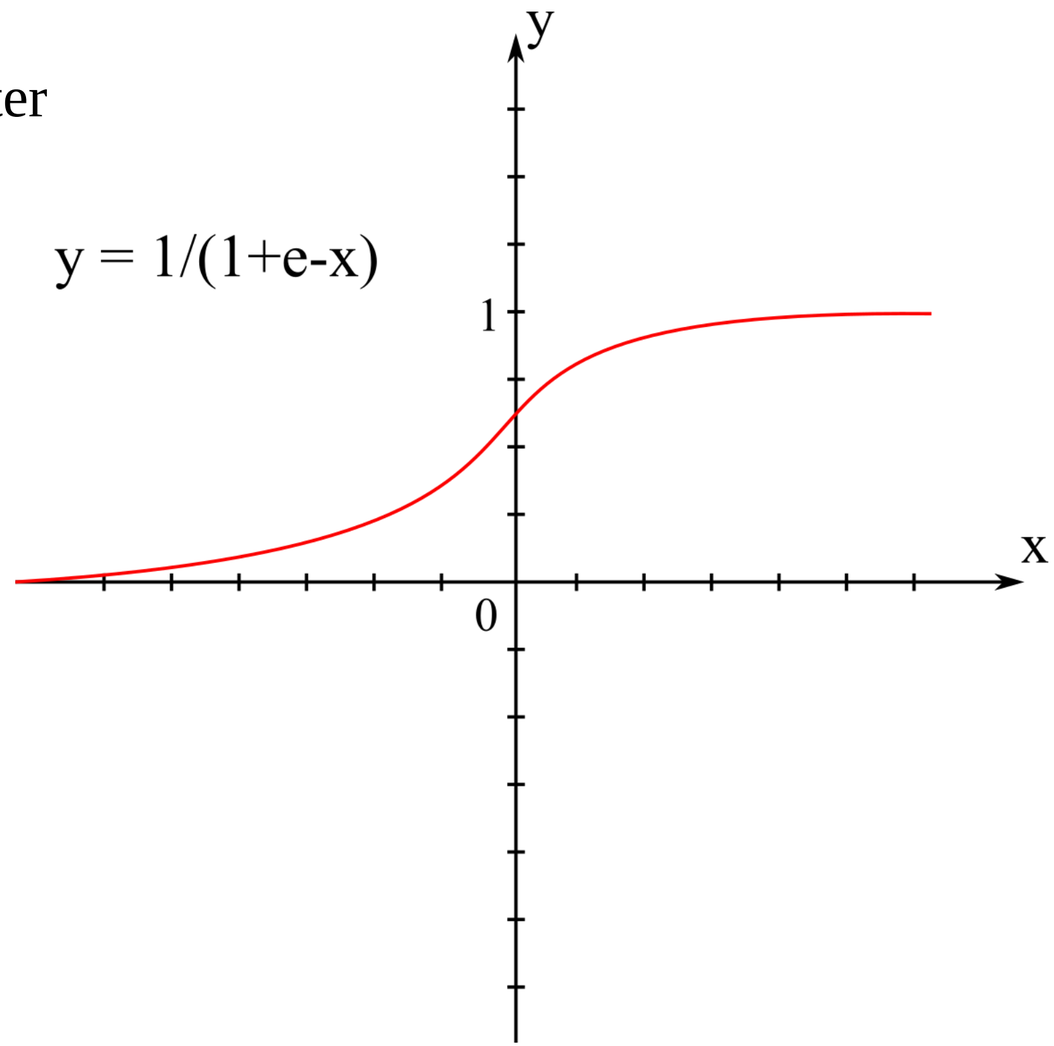
3. Activation functions

- **Sigmoid activation function:**

Usually used for hidden layer in order to allow a better optimization for weights and biases.

```
#Activation function Sigmoid

class Activation_Sigmoid:
    # Forward Pass
    def forward(self,inputs):
        #Calculate output values from input
        self.output = 1 / (1 + np.exp(-inputs))
```

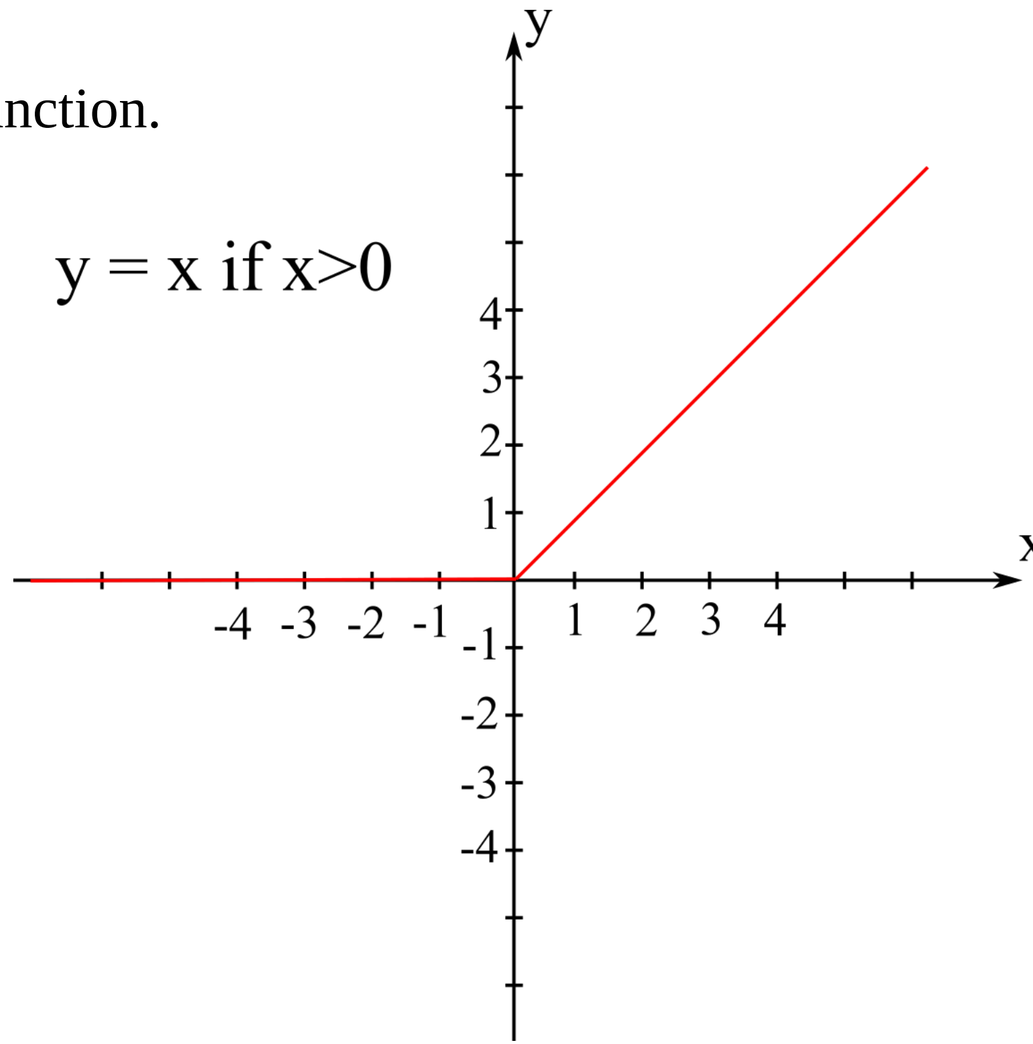


3. Activation functions

- **Rectified Linear activation function (ReLU):**

Rapid and efficient, it is today the most widely used activation function.

```
class Activation_ReLU:
    #Forward Pass
    def forward(self,inputs):
        #Calculate output values from input
        self.output = np.maximum(0,inputs)
```



3. Activation functions

- **Softmax activation function (ReLU):**

In order to design classifier using N, Softmax activation is a good choice at the output layer as it returns bounded and normalized output values, that represent the probabilities of the output neurons. The sum of the probabilities is equal to 1

$$S_{i,j} = \frac{e^{z_{i,j}}}{\sum_{l=1}^{L} e^{z_{i,l}}}$$

```
class Activation_SoftMax:

    #Forward pass:
    def forward(self,inputs):

        #Get unnormalized probabilities
        exp_values = np.exp(inputs - np.max(inputs, axis =
1,keepdims=True))
        self.exp_values = exp_values

        #Normalized them for each sample
        probabilities = exp_values / np.sum(exp_values, axis = 1,
keepdims=True)

        self.output = probabilities
```

Artificial Neural Network

END OF CHAPTER 2 – Part 2