ANN Part-2 Activation Functions and Simple Network

Friday, May 5, 2023 12:37 AM

Activation function

↳ Generally Non-Linear.

There are two computations that happen in a neuron:

(i) Weighted Sum → $z = \sum_{i=1}^{n} w_i x_i + w_0$

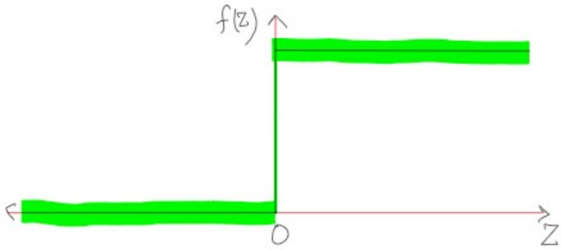
(ii) Non-Linear Activation → $f(z)$

↳ Activation function

Various Activation functions

1.) Thresholding function

$$f(z) = \begin{cases} 1 \; ; & z \geq 0 \\ 0 \; ; & z < 0 \end{cases}$$

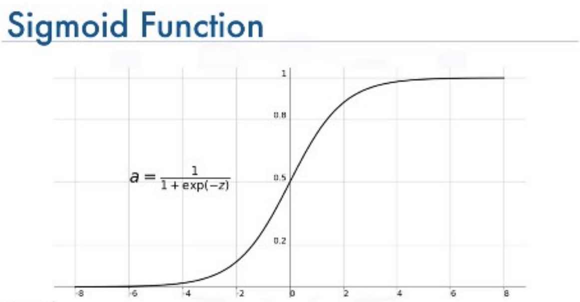


* Not used in practice that much

2.) Sigmoid Function (aka Logistic Function)

$$\sigma(z) = f(z) = \frac{1}{1+e^{-z}}$$

$$\sigma(0) = \frac{1}{1+e^{0}} = \frac{1}{1+1} = \frac{1}{2} = 0.5$$

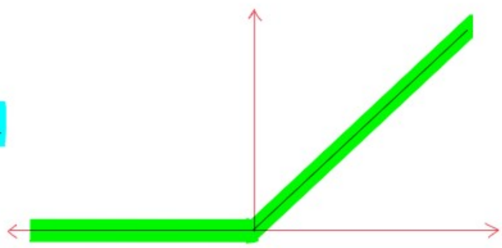


* One of the most popular activation functions till 2015-16.

3.) ReLu (Rectified Linear Unit)

$$f(z) = \max(0, z)$$

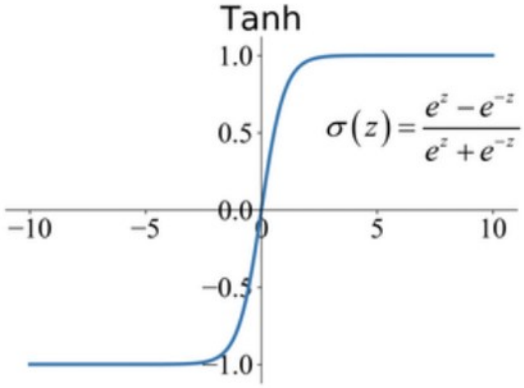
* Overall the ReLu function is non-linear but is piecewise-linear



4.) tanh

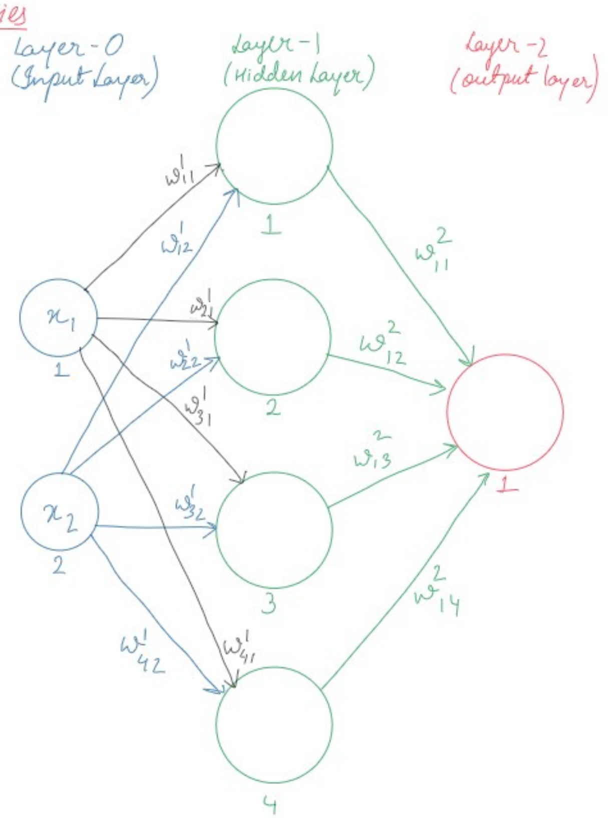
$$f(z) = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

Read as tan hyperbolic function



* The implementation of the activation function can be found in Keras. https://keras.io/api/layers/activations/

Terminologies



$w_{ab}^c \rightarrow$ a → Neuron of next layer
b → Neuron of current layer
c → Layer number of 'a' neuron

* The incoming weights to a neuron can also be combined into a vector

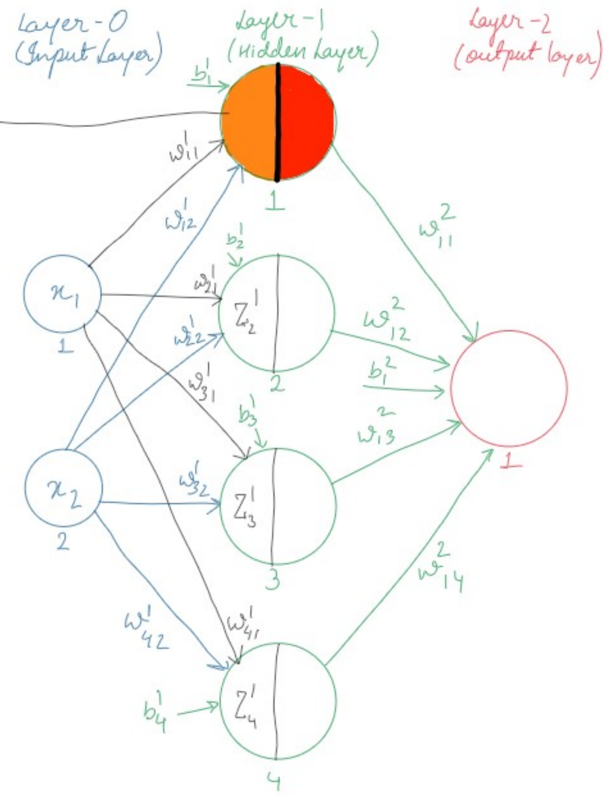
Eg for layer-1, neuron 1

$$w_1^1 = \begin{bmatrix} w_{11}^1 \\ w_{12}^1 \end{bmatrix}$$

It represents all the incoming weights to Neuron 1 in Layer 1

$$\rightarrow w_4^1 = \begin{bmatrix} w_{41}^1 \\ w_{42}^1 \end{bmatrix}$$ ⇒ All the incoming weights to Neuron 4 in Layer 1.

$$\rightarrow w_1^2 = \begin{bmatrix} w_{11}^2 \\ w_{12}^2 \\ w_{13}^2 \\ w_{14}^2 \end{bmatrix}$$



$$\underbrace{w_{11}^1 x_1 + w_{12}^1 x_2}_{(w_1^1)^T \cdot x} + b_1^1 = Z_1^1$$

$$(w_1^1)^T \cdot x + b_1^1$$

$$\Rightarrow w_1^1 = \begin{bmatrix} w_{11}^1 \\ w_{12}^1 \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$w_1^1 \cdot x = w_{11}^1 x_1 + w_{12}^1 x_2$$

So,

$$Z_1^1 = (w_1^1)^T \cdot x + b_1^1$$

$$Z_2^1 = (w_2^1)^T \cdot x + b_2^1$$

$$Z_3^1 = (w_3^1)^T \cdot x + b_3^1$$

$$Z_4^1 = (w_4^1)^T \cdot x + b_4^1$$

Combining them

Combining them

$$Z^1 = \begin{bmatrix} Z_1^1 \\ Z_2^1 \\ Z_3^1 \\ Z_4^1 \end{bmatrix} = \underbrace{\begin{bmatrix} (w_1^1)^T \\ (w_2^1)^T \\ (w_3^1)^T \\ (w_4^1)^T \end{bmatrix}}_{w^1} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} b_1^1 \\ b_2^1 \\ b_3^1 \\ b_4^1 \end{bmatrix}}_{b^1}$$

$\Rightarrow \boxed{Z^1 = w^1 x + b^1}$ Linear Computation part of the Neural Network

Now, passing it to activation function (Non-Linear)

$$\begin{bmatrix} a_1^1 \\ a_2^1 \\ a_3^1 \\ a_4^1 \end{bmatrix} = \begin{bmatrix} f(Z_1^1) \\ f(Z_2^1) \\ f(Z_3^1) \\ f(Z_4^1) \end{bmatrix}$$

$$a^1 = f(Z^1)$$

Now if there are 'l' hidden layers

$$\boxed{\begin{aligned} &\Rightarrow Z^l = w^l a^{(l-1)} + b^l \\ &\Rightarrow a^l = f(Z^l) \end{aligned}}$$

$\rightarrow a^0 = x$

$\rightarrow$ for $l = 1, 2, ---, L-1$

$L \rightarrow$ Number of Layers.

☆ $a^{(l-1)} \rightarrow$ output from the previous layer $(l-1)$ as input for the current layer (l)

$\hookrightarrow$ General formula of computations performed by a neural network