

Neural Network

Artificial Neural Network | Jaringan Saraf Tiruan

Nama Pengajar:

Dian Ade Kurnia, M.Kom
Rusnanda Farhan
Rusnandi Fikri
Rika Sahriana

Agenda

01

Introduction
Neural Network

02

Activation
Function

03

Feedforward and
Backpropagation

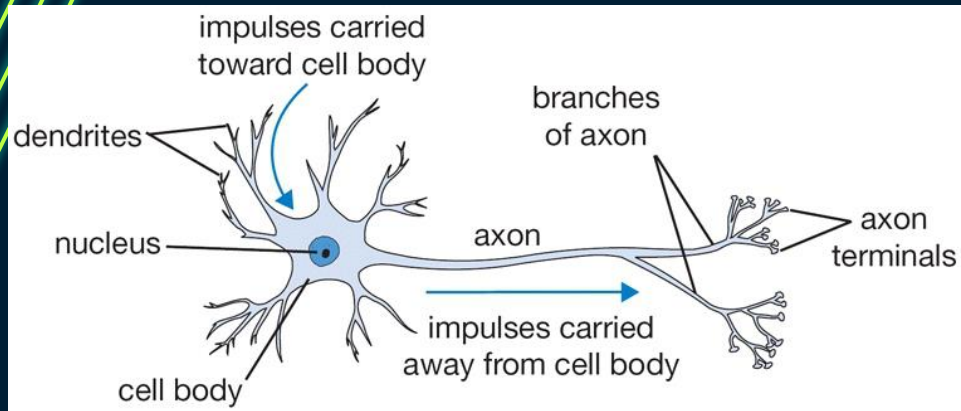
04

Conclusion

A decorative graphic on the left side of the slide consisting of multiple concentric hexagons. The hexagons are outlined in a light green color and are arranged in a way that they appear to recede into the distance, creating a tunnel-like effect. The number '01' is centered within the innermost hexagon.

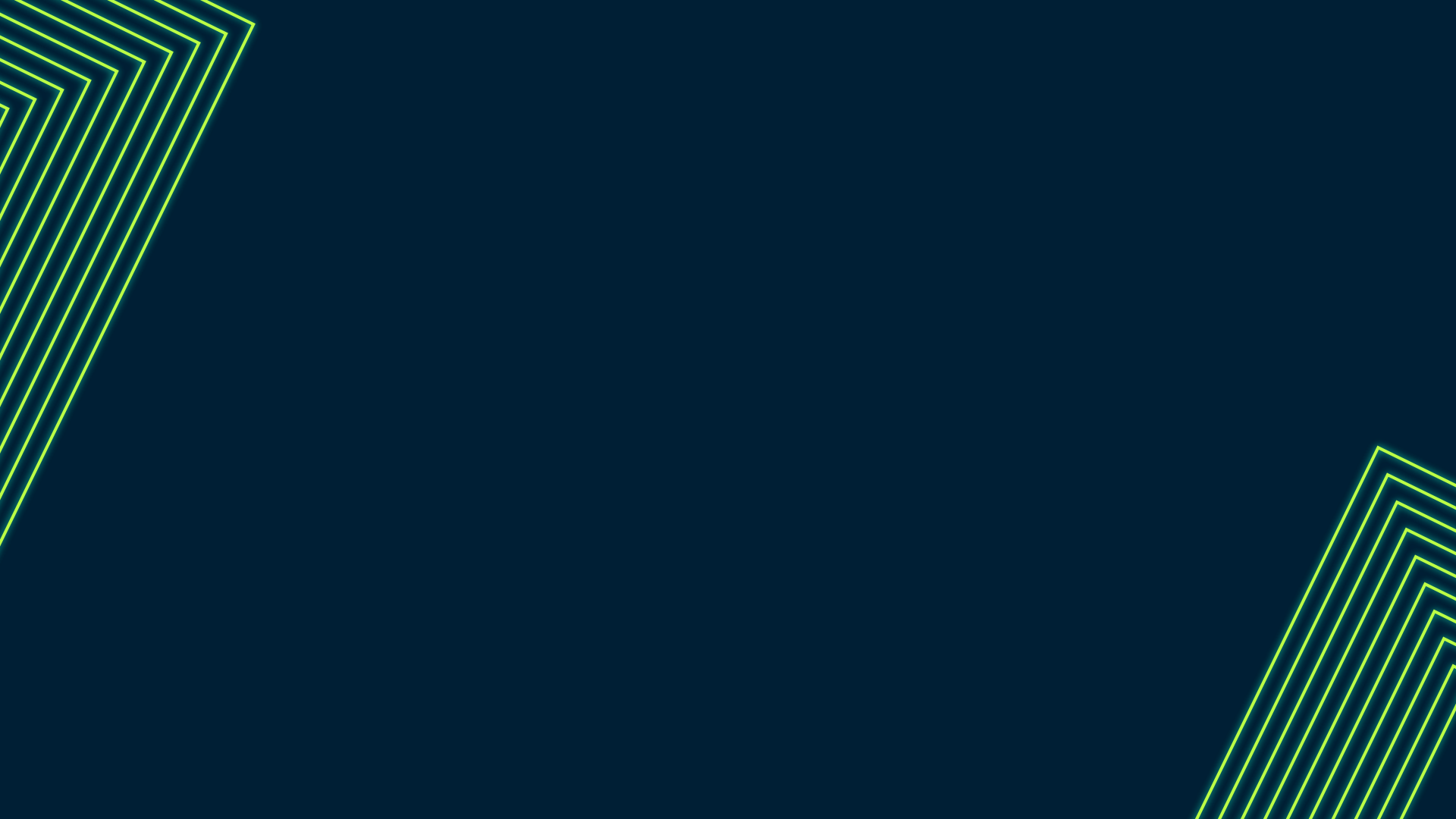
01

Introduction Neural Network



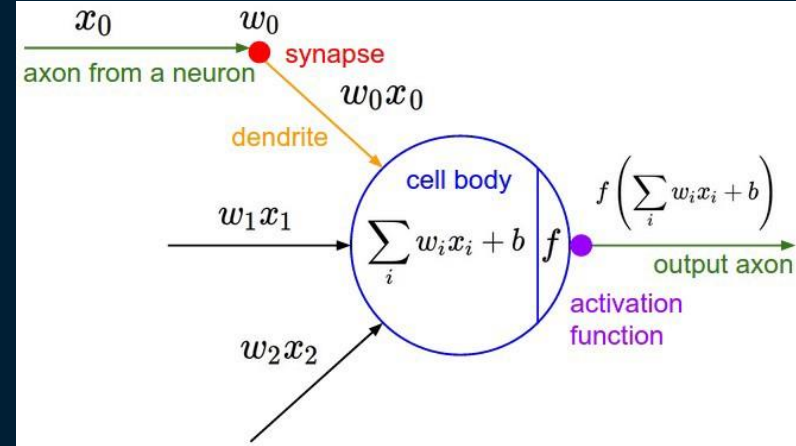
Main Idea of Neural Network

Artificial Neural Networks is a computational model inspired by the way biological neural networks in the human brain process information.



Building Blocks: Neurons

The basic unit of computation in a neural network is the neuron, often called a node or unit. It receives input from some other nodes, or from an external source and computes an output. Each input has an associated weight (w), which is assigned on the basis of its relative importance to other inputs. The node applies a function to the weighted sum of its inputs. Bias (b) is also an additional input that is included in the weighted sum calculation. Weight (w) and bias (b) are usually random values.



A decorative graphic on the left side of the slide consisting of multiple concentric hexagonal outlines in a light green color, creating a tunnel-like effect that draws the eye towards the center.

02

Activation Function

Activation Function

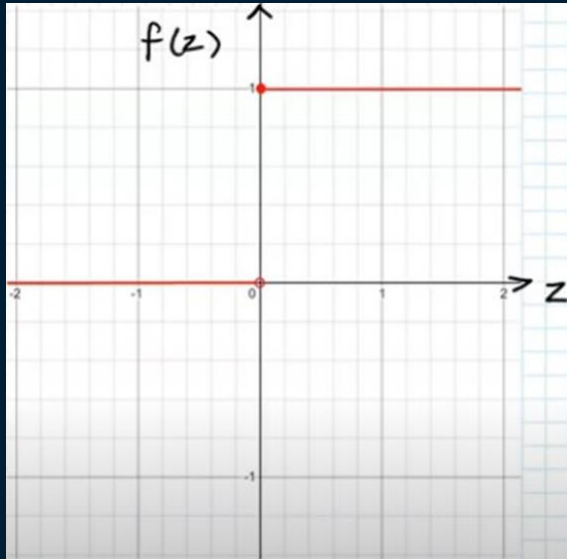
- a) Step Function $\varphi(v) = \begin{cases} 0, v < 0 \\ 1, v \geq 0 \end{cases}$, range $\{0,1\}$
- b) Linear Function $\varphi(v) = v$ or $F(z) = z$, range $\{-\infty, \infty\}$
- c) ReLU Function $\varphi(v) = \begin{cases} 0, v < 0 \\ v, v \geq 0 \end{cases}$, range $\{0, \infty\}$
- d) Leaky ReLU Function* $\varphi(v) = \begin{cases} av, v < 0 \\ v, v \geq 0 \end{cases}$, a= small pos. (+) number, range $\{-\infty, \infty\}$
- e) Sigmoid Function $\varphi(v) = \frac{e^v}{1+e^v}$ or $\frac{1}{1+e^{-v}}$, range $\{0,1\}$
- f) Softmax Function** class = $\hat{y}_1, \hat{y}_2, \dots, \hat{y}_i \mid F(\hat{y}_n) = \frac{e^{\hat{y}_n}}{\sum_{n=1}^i e^{\hat{y}_n}}$

* = only for hidden nodes

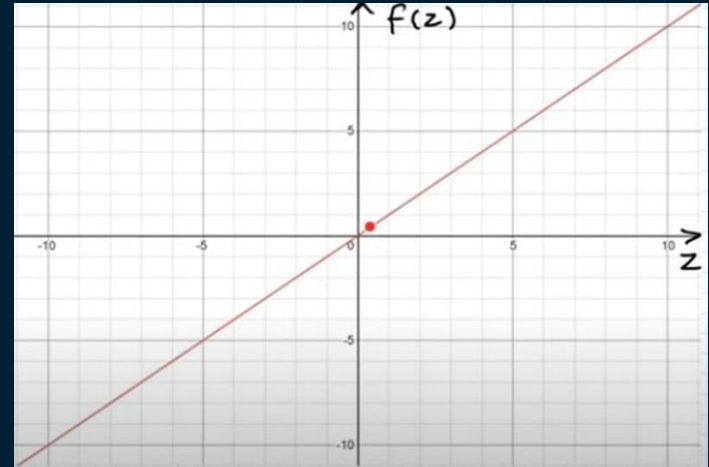
** = only for output nodes

Activation Function

a.)

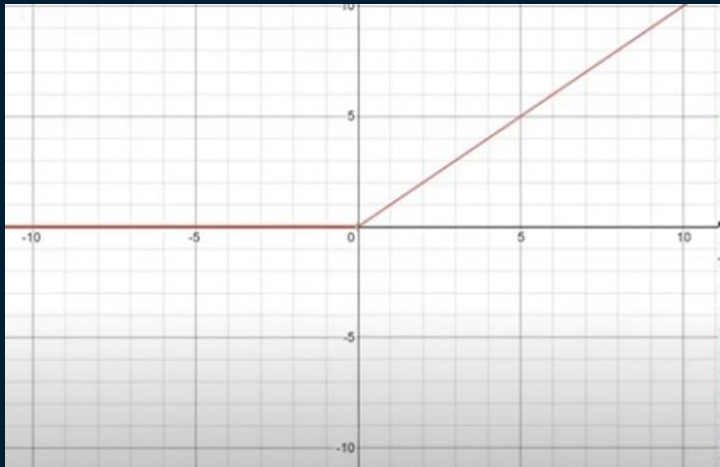


b.)



Activation Function

c.)

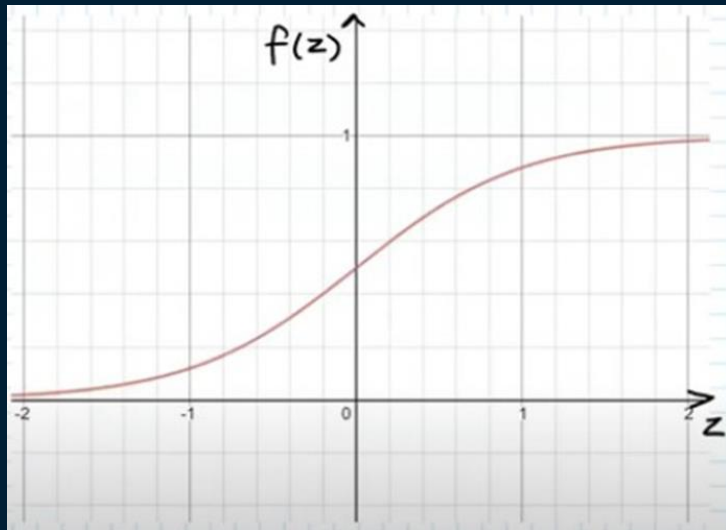


d.)

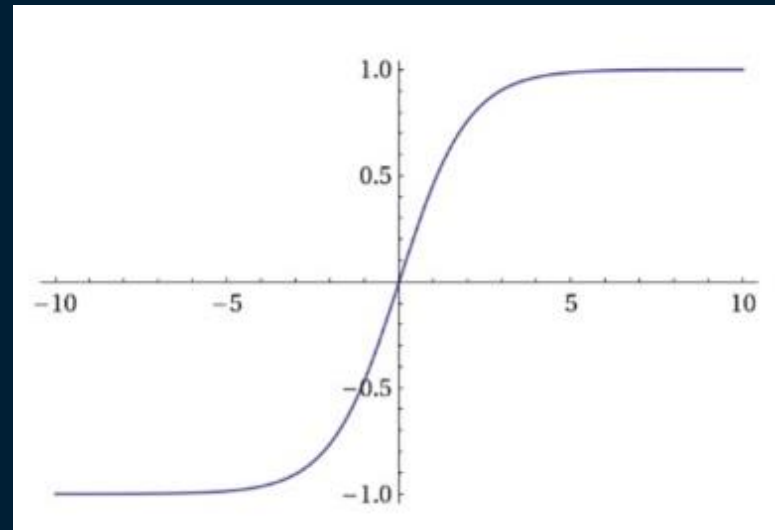


Activation Function

e.)



f.)



A decorative graphic on the left side of the slide consisting of multiple concentric hexagonal outlines. The hexagons are drawn with thin, light green lines and are centered towards the left edge of the frame. The number '03' is placed in the center of the innermost hexagon.

03

Feedforward & Backprop

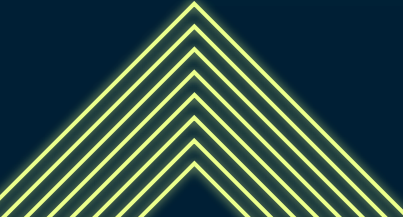


Feedforward

feedforward neural network is an artificial neural network where connections moving the information in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes.

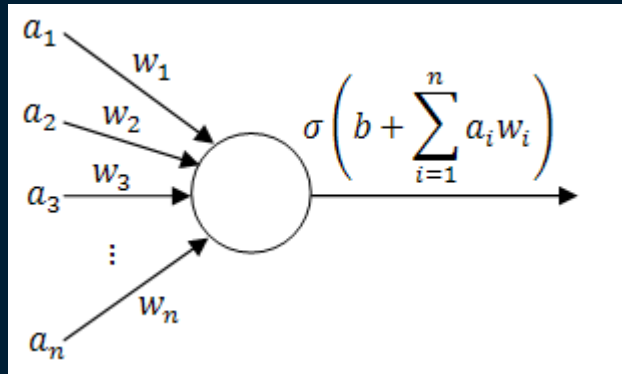
Backpropagation

Backpropagation is a process to update all of the weights in neural network architecture after calculating the feedforward process



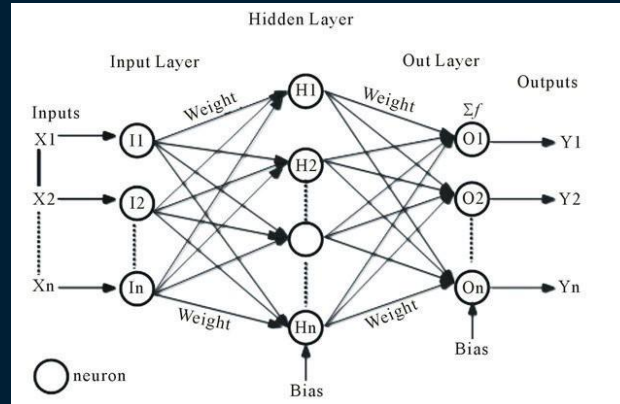
Single Preceptron

This is the simplest feedforward neural Network and does not contain any hidden layer, Which means it only consists of a single layer of output nodes. This is said to be single because when we count the layers we do not include the input layer, the reason for that is because at the input layer no computations is done, the inputs are fed directly to the outputs via a series of weights.



Multi Layer Preceptron

This class of networks consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons of the subsequent layer. MLP are very more useful and one good reason is that, they are able to learn non-linear representations (most of the cases the data presented to us is not linearly separable), Learning processing with at least 3 layer called as Deep Learning



Loss

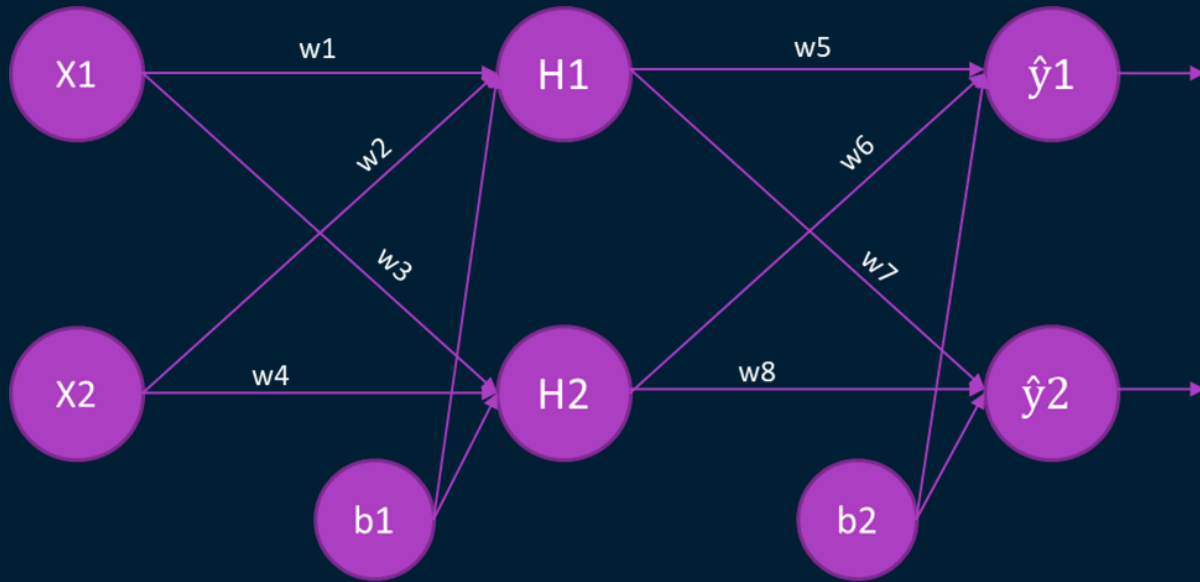
Before we train our network, we first need a way to quantify how “good” it’s doing so that it can try to do “better”. That’s what the loss is.

We’ll use the mean squared error (MSE) loss:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_{\text{true}} - y_{\text{pred}})^2$$

lower loss indicates better prediction, so the aim of training process is to minimize the loss

Example



Input:

$X_1 = 0.05$

$X_2 = 0.10$

Target Values:

$\hat{y}_1 = 0.01$

$\hat{y}_2 = 0.99$

Biases:

$b_1 = 0.35$

$b_2 = 0.60$

Activation Func:

Sigmoid

Initial weights:

$W_1 = 0.15$

$W_2 = 0.20$

$W_3 = 0.25$

$W_4 = 0.30$

$W_5 = 0.40$

$W_6 = 0.45$

$W_7 = 0.50$

$W_8 = 0.55$

Feedforward

$$\begin{aligned}\text{In H1} &= (x1 * w1) + (x2 * w2) + b1 \\ &= (0.05 * 0.15) + (0.10 * 0.20) + 0.35 \\ &= 0.3775\end{aligned}$$

$$\begin{aligned}\text{Out H1} &= \frac{1}{1 + e^{-in H1}} \\ &= \frac{1}{1 + e^{-0.3775}} \\ &= 0.593269992\end{aligned}$$

$$\begin{aligned}\text{In H2} &= (x1 * w3) + (x2 * w4) + b1 \\ &= (0.05 * 0.25) + (0.10 * 0.30) + 0.35 \\ &= 0.3925\end{aligned}$$

$$\begin{aligned}\text{Out H2} &= \frac{1}{1 + e^{-in H2}} \\ &= \frac{1}{1 + e^{-0.3925}} \\ &= 0.5968843783\end{aligned}$$

Feedforward

In \hat{y}_1

$$= (\text{out H1} * w_5) + (\text{out H2} * w_6) + b_2$$

$$= (0.593269992 * 0.4) + (0.5968843783 * 0.45) + 0.6$$

$$= 1.105905967$$

$$\begin{aligned}\text{Out } \hat{y}_1 &= \frac{1}{1 + e^{-\text{In } \hat{y}_1}} \\ &= \frac{1}{1 + e^{-1.105905967}} \\ &= 0.75136507\end{aligned}$$

In \hat{y}_2

$$= (\text{out H1} * w_7) + (\text{out H2} * w_8) + b_2$$

$$= (0.593269992 * 0.5) + (0.5968843783 * 0.55) + 0.6$$

$$= 1.2249214041$$

$$\begin{aligned}\text{Out } \hat{y}_2 &= \frac{1}{1 + e^{-\text{In } \hat{y}_2}} \\ &= \frac{1}{1 + e^{-1.2249214041}} \\ &= 0.772928465\end{aligned}$$

Calculate Error

$$\text{Out } \hat{y}_1 = 0.75136507$$

$$\text{Out } \hat{y}_2 = 0.772928465$$

$$\begin{aligned} E_{\text{total}} &= 1/2 \sum (\text{target} - \text{output})^2 \\ &= 0.5(0.01 - 0.75136507)^2 + 0.5(0.99 - 0.772928465)^2 \\ &= 0.274811083 + 0.023560026 \\ &= 0.298371109 \end{aligned}$$

Backpropagation

$$\text{Error at } w_5 = \frac{\partial E_{total}}{\partial w_5}$$

$$E_{total} = \frac{1}{2}(\text{target } \hat{y}_1 - \text{out } \hat{y}_1)^2 + \frac{1}{2}(\text{target } \hat{y}_2 - \text{out } \hat{y}_2)^2$$

$$\text{Out } \hat{y}_1 = \frac{1}{1 + e^{-in \hat{y}_1}}$$

$$In \hat{y}_1 = (\text{out } H_1 * w_5) + (\text{out } H_2 * w_6) + b_2$$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial \text{out } \hat{y}_1} * \frac{\partial \text{out } \hat{y}_1}{\partial in \hat{y}_1} * \frac{\partial in \hat{y}_1}{\partial w_5}$$

$$\begin{aligned} a) \quad \frac{\partial E_{total}}{\partial \text{out } \hat{y}_1} &= 2 * \frac{1}{2} (\text{target } \hat{y}_1 - \text{out } \hat{y}_1)^{2-1} * -1 + 0 \\ &= -(\text{target } \hat{y}_1 - \text{out } \hat{y}_1) \\ &= -(0.01 - 0.75136507) \\ &= 0.74136507 \end{aligned}$$

Backpropagation

$$b) \frac{\partial out \hat{y}_1}{\partial in \hat{y}_1} = \frac{1}{2(\cosh(in \hat{y}_1)+1)} = \frac{1}{2*(1.6764359889828+1)} = 0.186815602$$

$$\begin{aligned} c) \frac{\partial in \hat{y}_1}{\partial w_5} &= (out H1 * w_5) + (out H2 * w_6) + b_2 \\ &= 1 * out H1 * w_5^{1-1} + 0 + 0 \\ &= 0.593269992 \end{aligned}$$

$$\begin{aligned} \frac{\partial E_{total}}{\partial w_5} &= \frac{\partial E_{total}}{\partial out \hat{y}_1} * \frac{\partial out \hat{y}_1}{\partial in \hat{y}_1} * \frac{\partial in \hat{y}_1}{\partial w_5} \\ &= 0.74136507 * 0.186815602 * 0.593269992 \\ &= 0.082167041 \end{aligned}$$

Backpropagation

$$w5 = w5 - \mu * \frac{\partial E_{total}}{\partial w5}, \text{ Learning Rate } (\mu) = 0.5$$

$$\begin{aligned} w5 &= w5 - 0.5 * \frac{\partial E_{total}}{\partial w5} \\ &= 0.4 - 0.5 * 0.082167041 \\ &= 0.35891648 \end{aligned}$$

$$w6 = 0.408666186$$

$$w7 = 0.511301270$$

$$w8 = 0.561370121$$

Backpropagation

$$\text{Error at } w_1 = \frac{\partial E_{\text{total}}}{\partial w_1}$$

$$\frac{\partial E_{\text{total}}}{\partial w_1} = \underset{\text{A}}{\frac{\partial E_{\text{total}}}{\partial \text{out } H_1}} * \underset{\text{B}}{\frac{\partial \text{out } H_1}{\partial \text{in } H_1}} * \underset{\text{C}}{\frac{\partial \text{in } H_1}{\partial w_1}}$$

$$\begin{aligned} \text{A) } \frac{\partial E_{\text{total}}}{\partial \text{out } H_1} &= \frac{\partial E_1}{\partial \text{out } H_1} + \frac{\partial E_2}{\partial \text{out } H_2} \\ &= \left(\frac{\partial E_1}{\partial \text{in } \hat{y}_1} * \frac{\partial \text{in } \hat{y}_1}{\partial \text{out } H_1} \right) + \frac{\partial E_2}{\partial \text{out } H_2} \\ &= \left(\left(\frac{\partial E_1}{\partial \text{out } \hat{y}_1} * \frac{\partial \text{out } \hat{y}_1}{\partial \text{in } \hat{y}_1} \right) * \frac{\partial \text{in } \hat{y}_1}{\partial \text{out } H_1} \right) + \frac{\partial E_2}{\partial \text{out } H_2} \\ &= 0.74136607 * 0.186815602 * 0.4 + (-0.019049119) \\ &= 0.0363503805 \end{aligned}$$

Backpropagation

$$B) \frac{\partial out\ H1}{\partial in\ H1} = \frac{1}{2(\cosh(in\ H1)+1)} = 0.241800709$$

$$C) \frac{\partial in\ H1}{\partial w1} = 1 * x1 + 0 + 0 = 0.05$$

$$\begin{aligned} \frac{\partial E_{total}}{\partial w1} &= 0.0363503805 * 0.241800709 * 0.05 \\ &= 0.000438568 \end{aligned}$$

Backpropagation

$$w1 = w1 - \mu * \frac{\partial E_{total}}{\partial w1}, \text{ Learning Rate } (\mu) = 0.5$$

$$\begin{aligned} w1 &= w1 - 0.5 * \frac{\partial E_{total}}{\partial w1} \\ &= 0.15 - 0.5 * 0.000438568 \\ &= 0.149780716 \end{aligned}$$

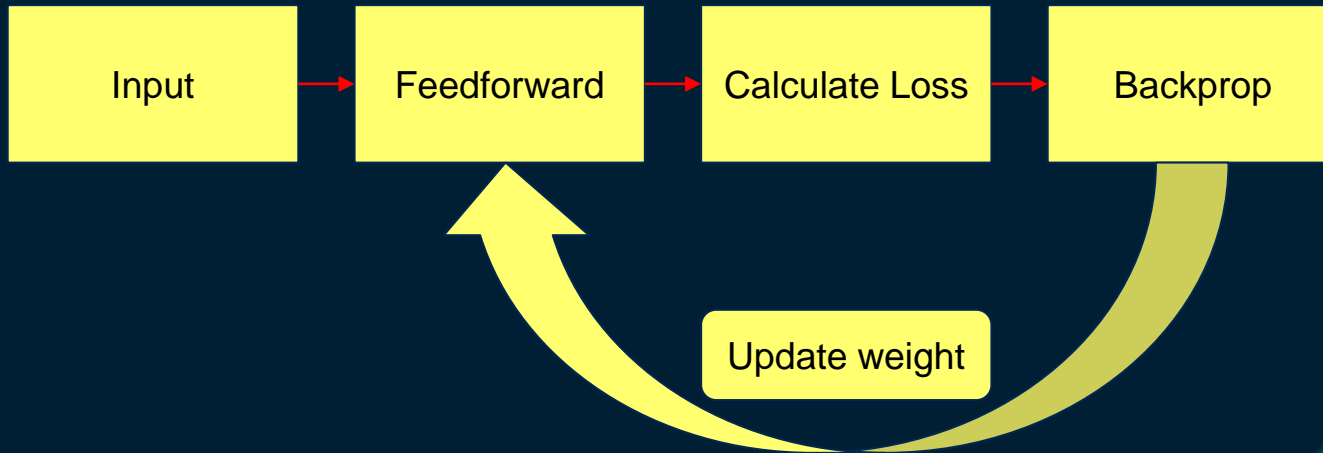
$$w2 = 0.19956143$$

$$w3 = 0.24975114$$

$$w4 = 0.29950229$$

Epoch

Epoch is the iteration when we do the pairwise feedforward and backprop process.





Conclusion

- Neural Network is a new form of machine learning method.
 - Neural Network is the beginning of deep learning concept development.
 - By using backprop, neural network tries to find minimum loss (error) to get best performance.
 - The deeper neural network architecture, the computational cost will be bigger.
- 