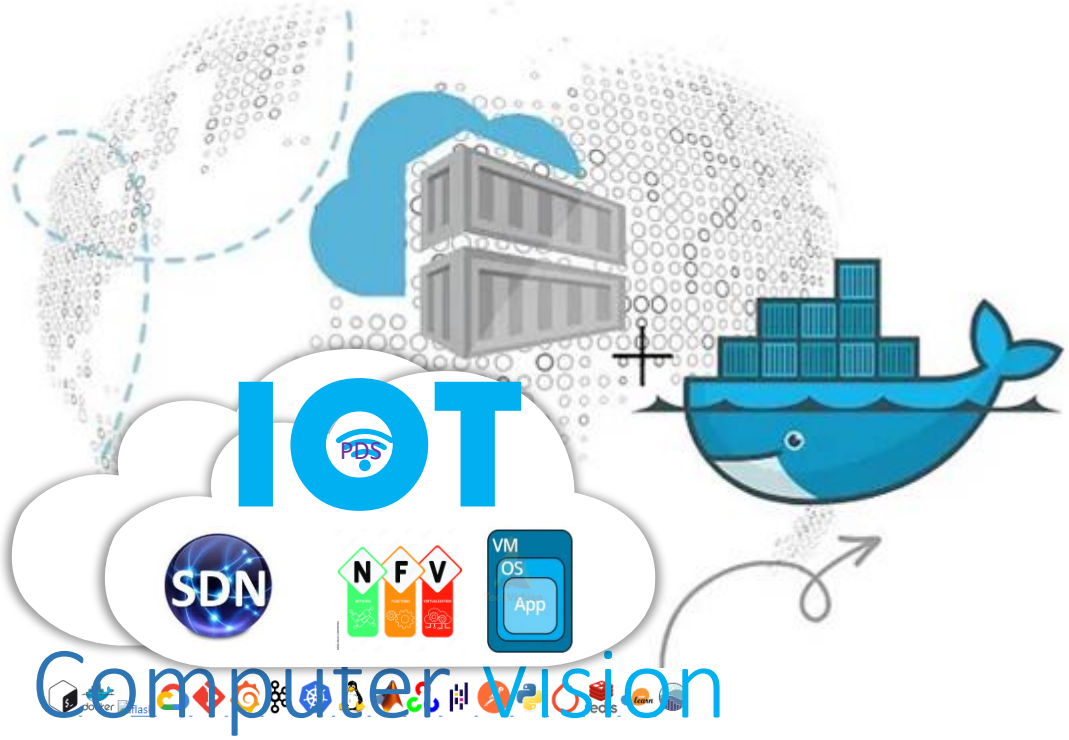


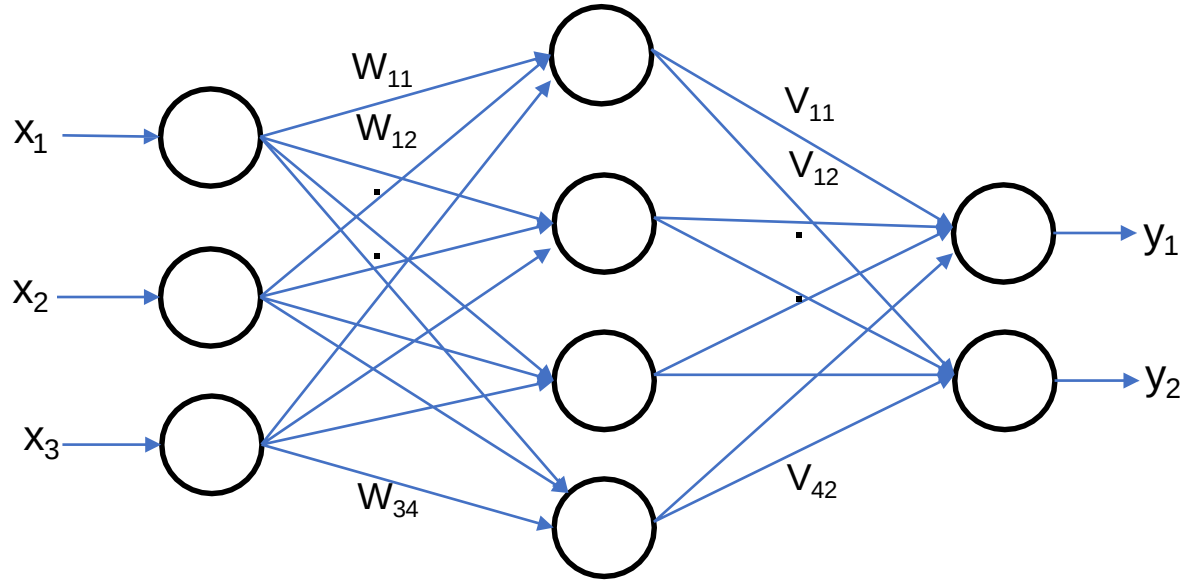
Chapter 5-3:

Multilayer Perceptron

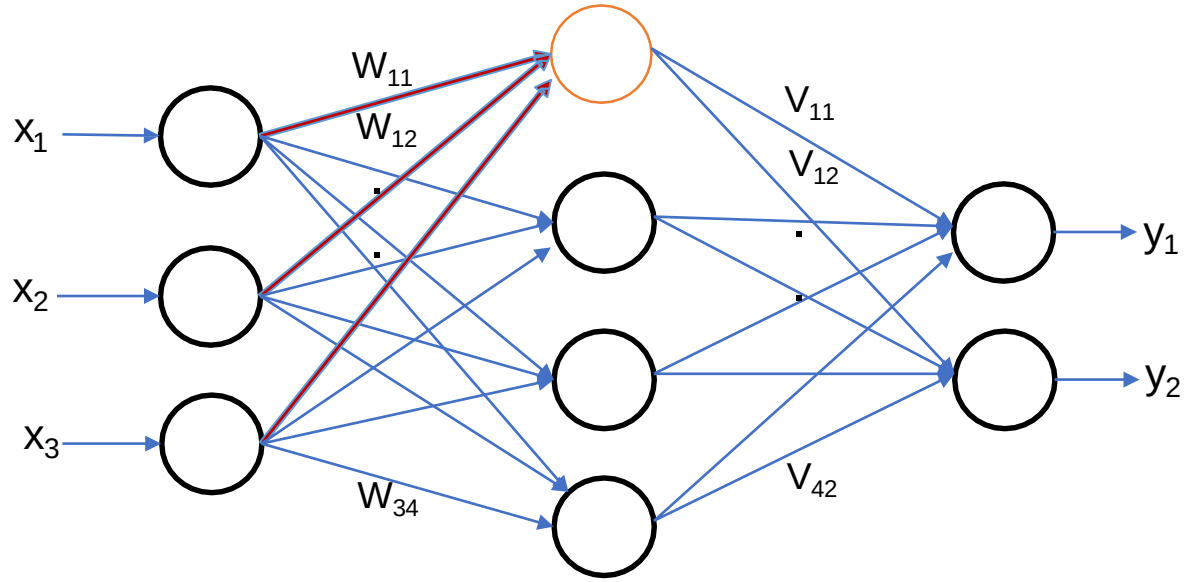
https://github.com/amelcharolinesgn2/IoT_simulator-mqtt-NodeRed

https://github.com/siagianp
https://www.youtube.com/@AmelOline/videos

What is Multilayer Perceptron NN?



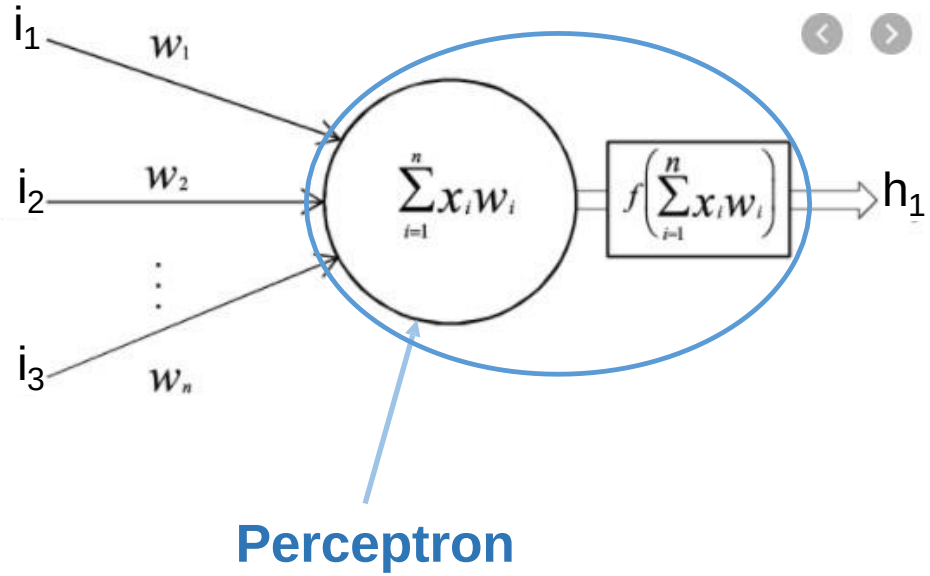
Multilayer Perceptron



Input Layer Layer

Hidden

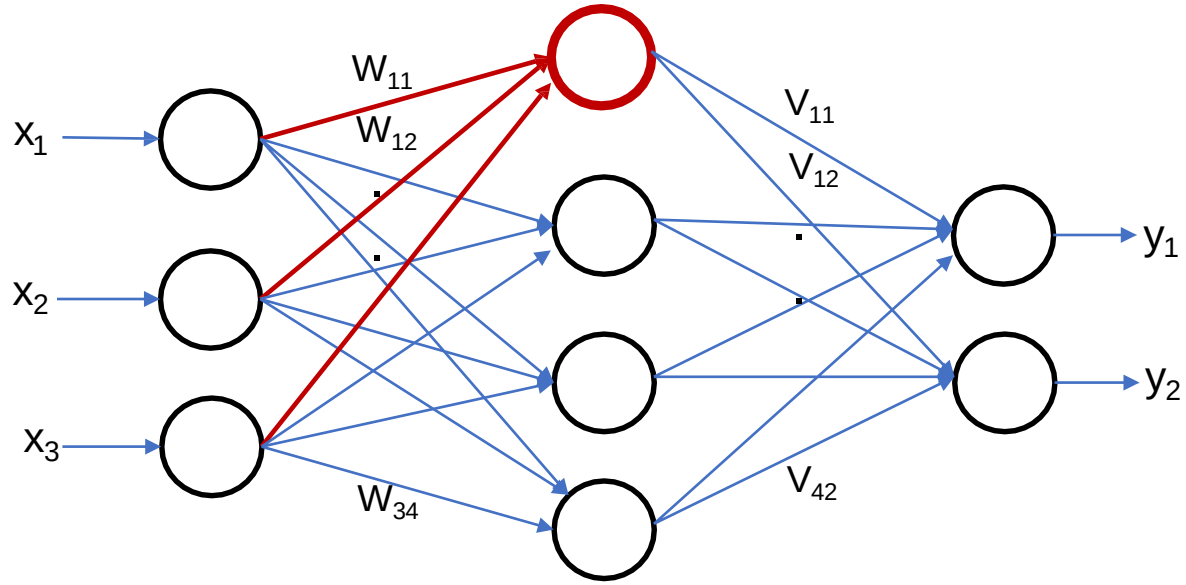
Output Layer



For a given node I, the perceptron is defined as weighted summation of the incoming data from the nodes of the previous layer.

$$p_i = x_0 w_{0i} + x_2 w_{2i} + \ldots + x_n w_{ni} = \sum_{j=1}^{n} x_j w_j$$

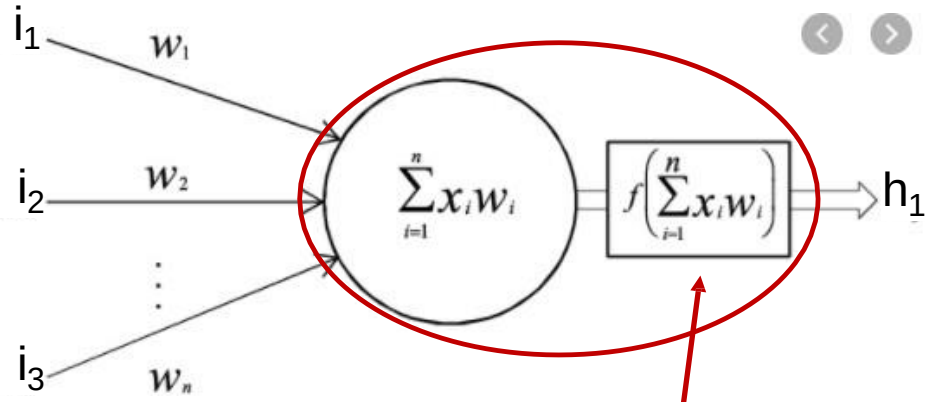
Multilayer Perceptron



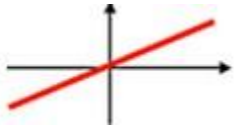
Input Layer

Hidden Layer

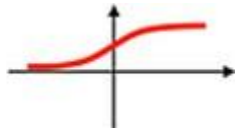
Output Layer



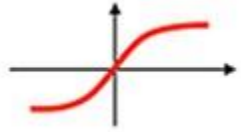
Linear $\phi(z) = z$



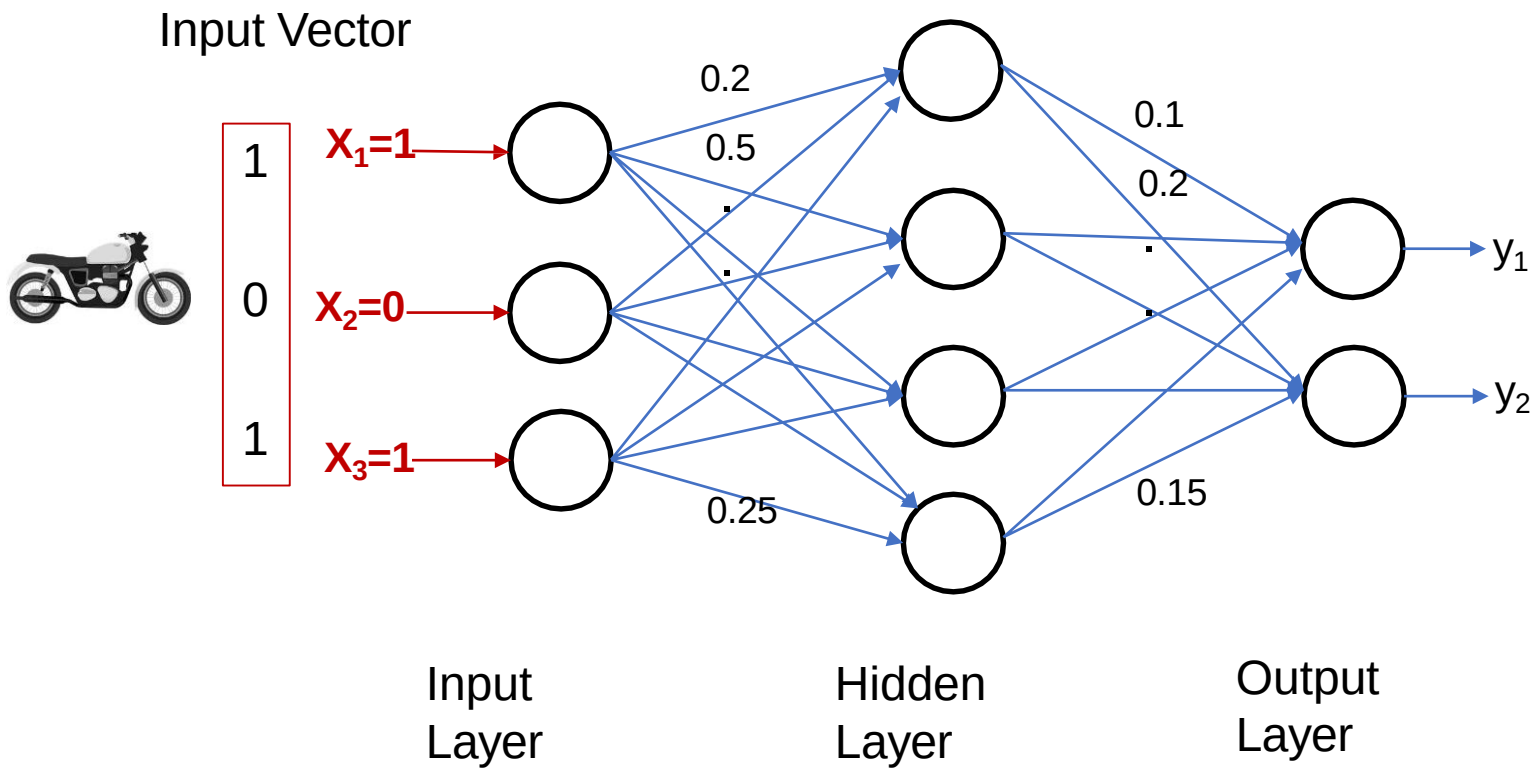
Logistic (sigmoid) $\phi(z) = \frac{1}{1 + e^{-z}}$



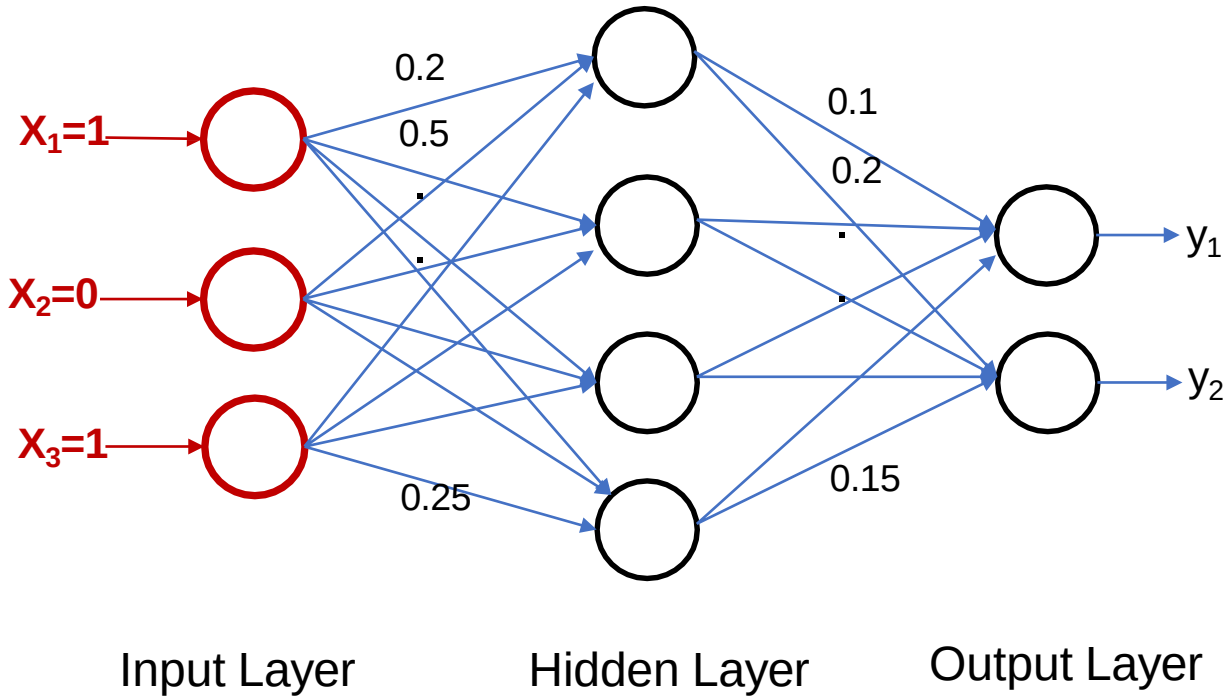
Hyperbolic tangent $\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$



Forward Pass



Forward Pass



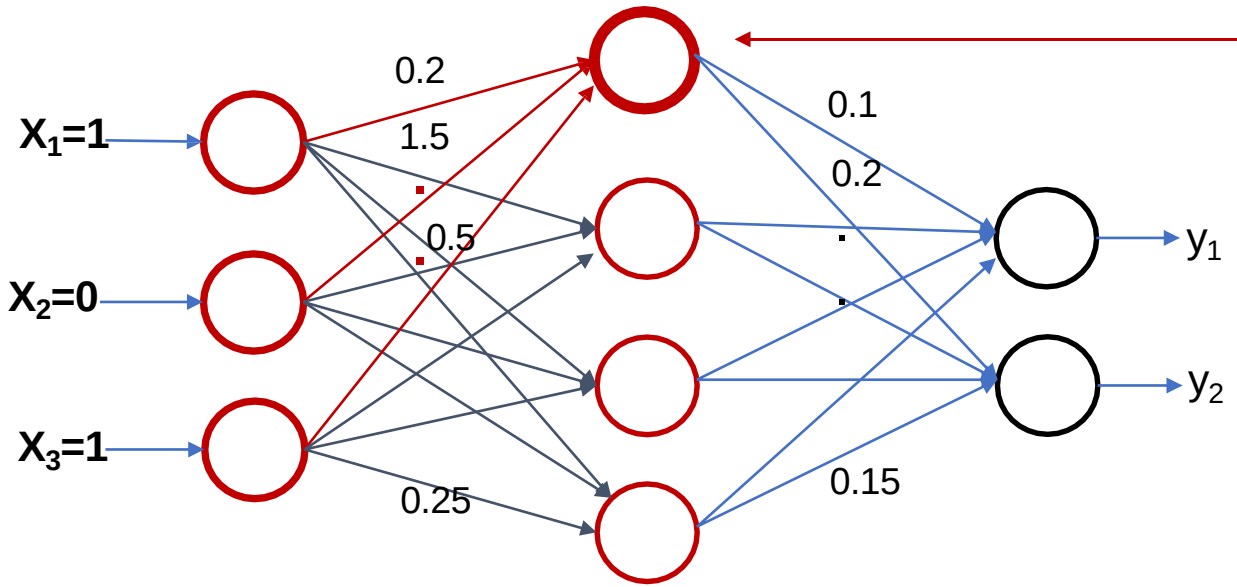
1
0
1

x=Perceptron(x)

Linear Activation function

x=f(x)

Forward Pass



Input Layer

Hidden Layer

Output Layer

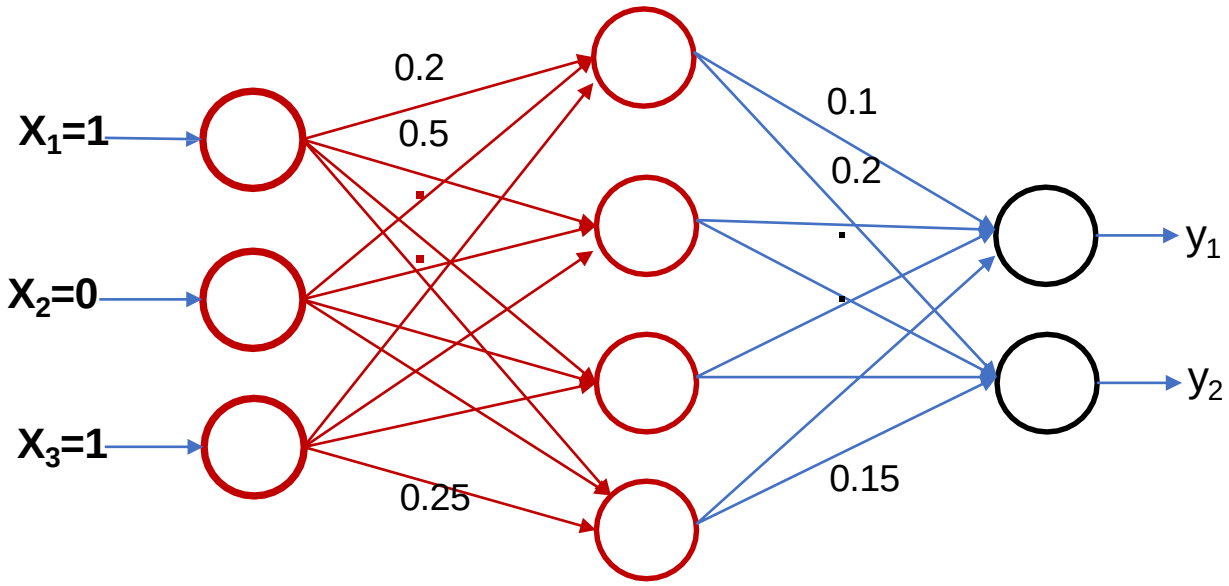
1
0
1

1x0.2 + 0x1.5 + 1x0.5= 0.7

Sigmoid Activation function

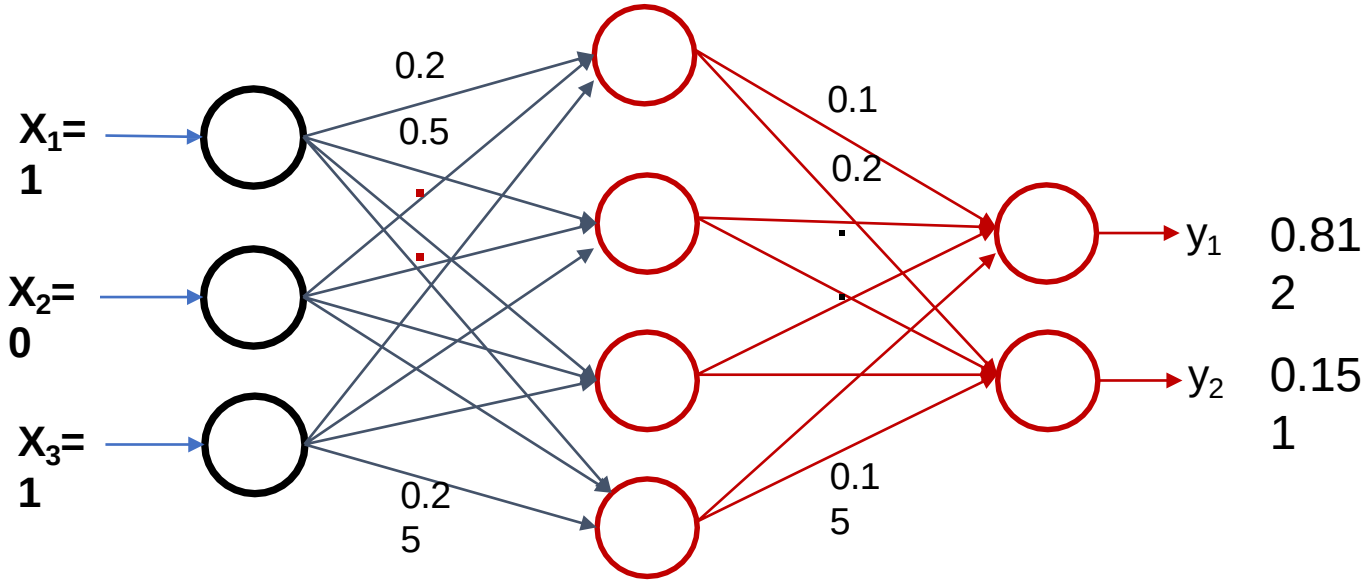
f(0.7)=0.668

Forward Pass



Input Layer	Hidden Layer	Output Layer
1	0.668	
0	0.912	
1	0.102	
	0.471	

Forward Pass



Input Layer

1
0
1

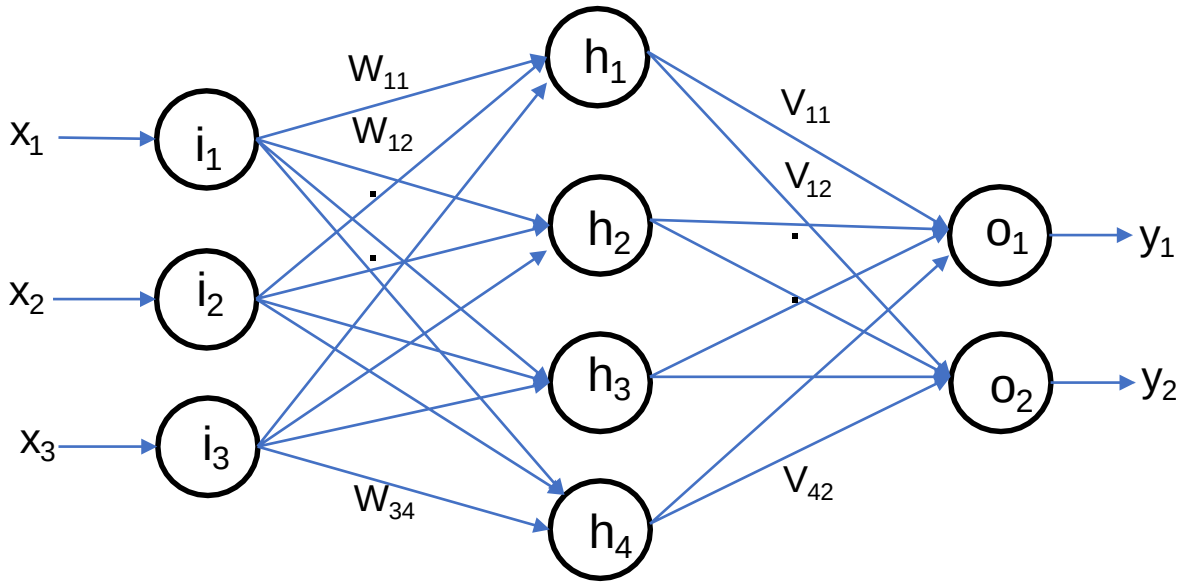
Hidden Layer

0.668
0.912
0.102
0.471

Output Layer

0.812
0.151

Weight Matrix



	h_1	h_2	h_3	h_4
i_1	W_{11}	W_{12}	W_{13}	W_{14}
i_2	W_{21}	W_{22}	W_{23}	W_{24}
i_3	W_{31}	W_{32}	W_{33}	W_{34}

W

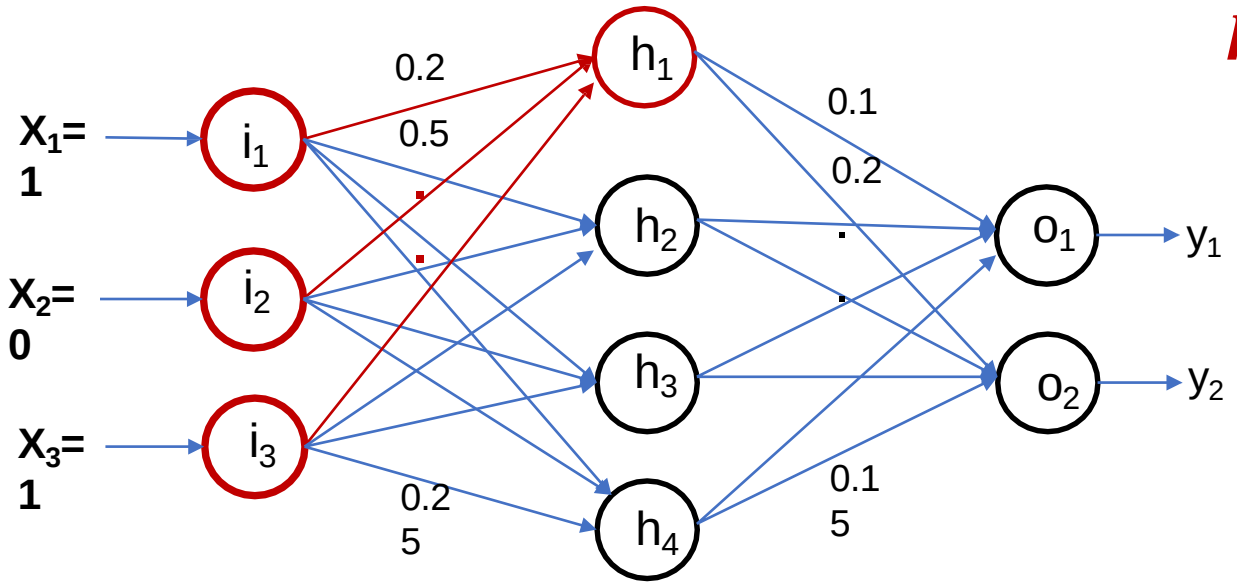
	o_1	o_2
h_1		
h_2		
h_3		
h_4		

V

Forward Pass

$$p_1 = x_1 W_{11} + x_2 W_{21} + x_3 W_{31}$$

$$h_1 = Sigmoid(p_1)$$

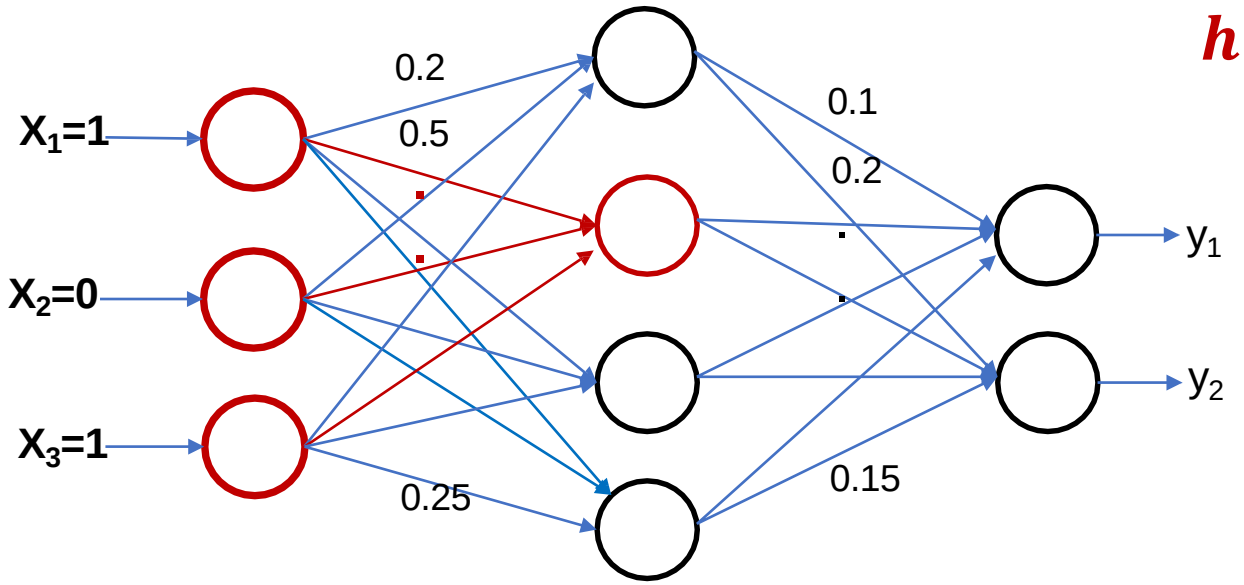


1 0 1	X	h_1 h_2 h_3 h_4 i_1 i_2 i_3 W	=	h_1 h_2 h_3 h_4 **0.668**

Forward Pass

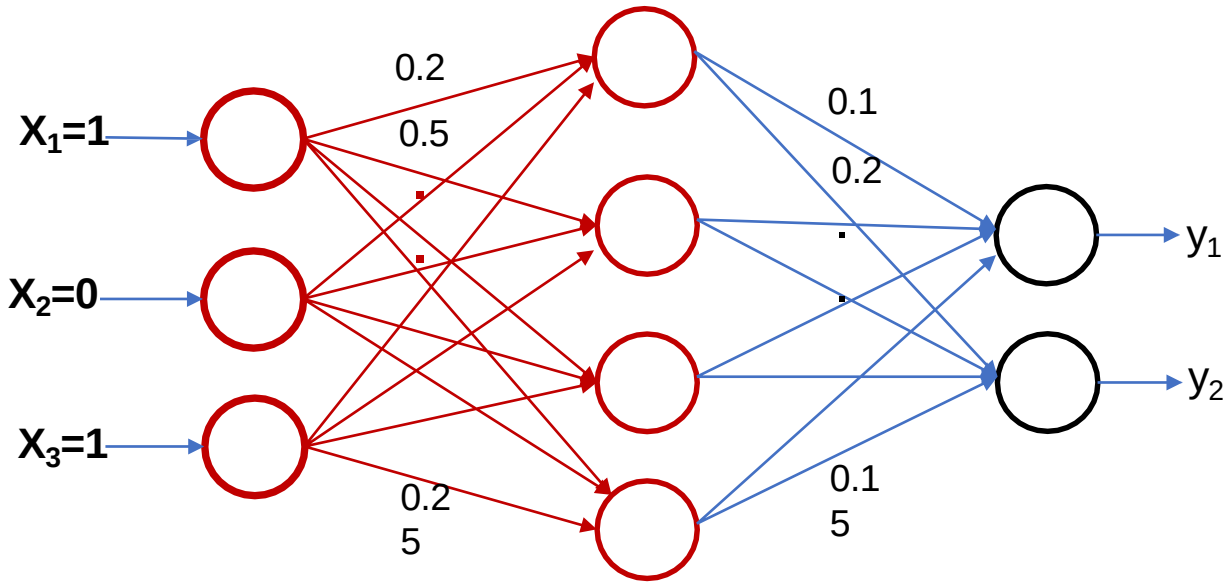
$$p_2 = x_1 W_{12} + x_2 W_{22} + x_3 W_{32}$$

$$h_2 = Sigmoid(p_2)$$



1 0 1	X	h_1 h_2 h_3 h_4 i_1 i_2 i_3 W	=	h_1 h_2 h_3 h_4 0.668 **0.912**

Forward Pass

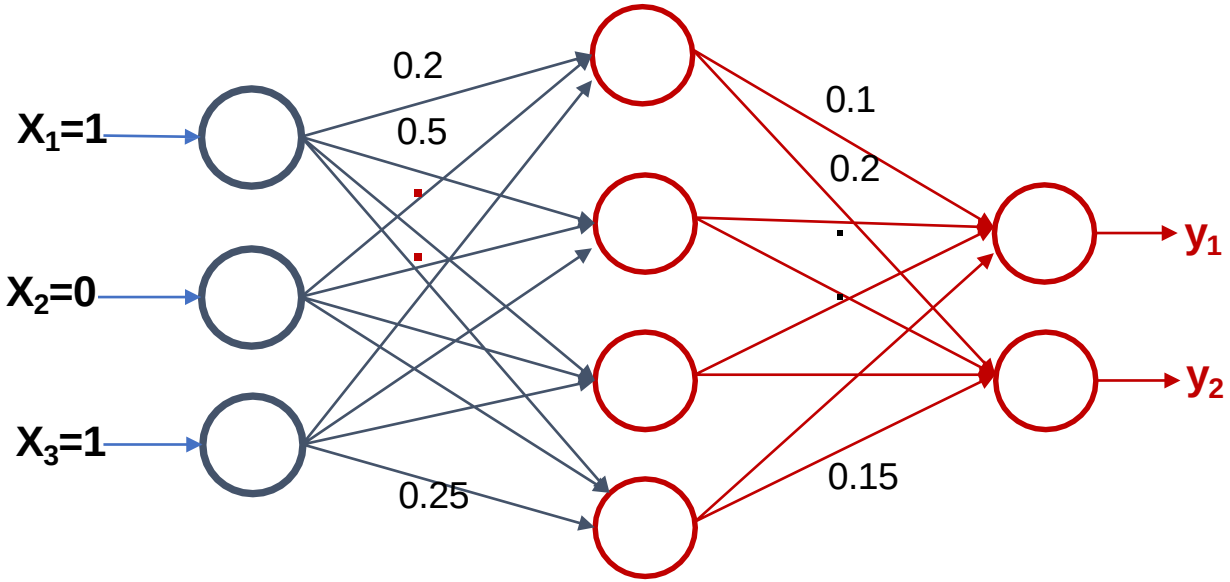


$$\bar{p}^T = \bar{x}^T . W$$

$$\bar{h}^T = Sigmoid(\bar{p}^T)$$

$\bar{x}^T$		W		$\bar{h}^T$
1 0 1	X	i_1	=	h_1 h_2 h_3 h_4
		i_2		0.668 0.912 0.102 0.471
		i_3		
		h_1 h_2 h_3 h_4		

Forward Pass

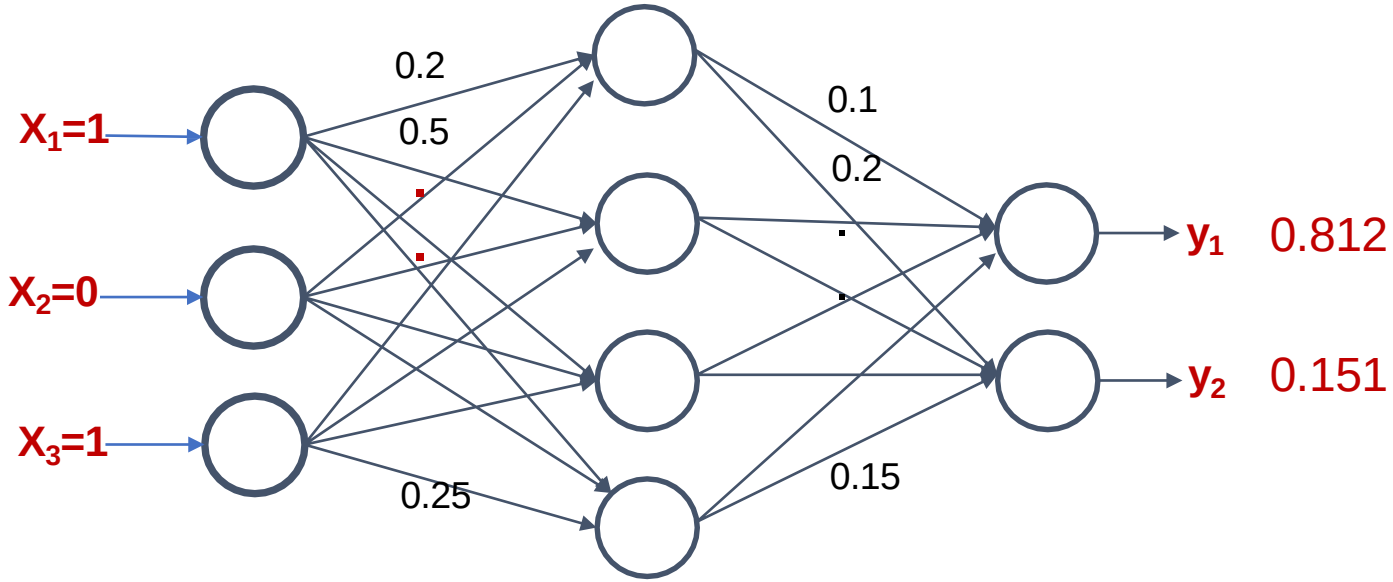


$$\bar{p}^T = \bar{h}^T . V$$
$$\bar{y}^T = Sigmoid(\bar{p}^T)$$

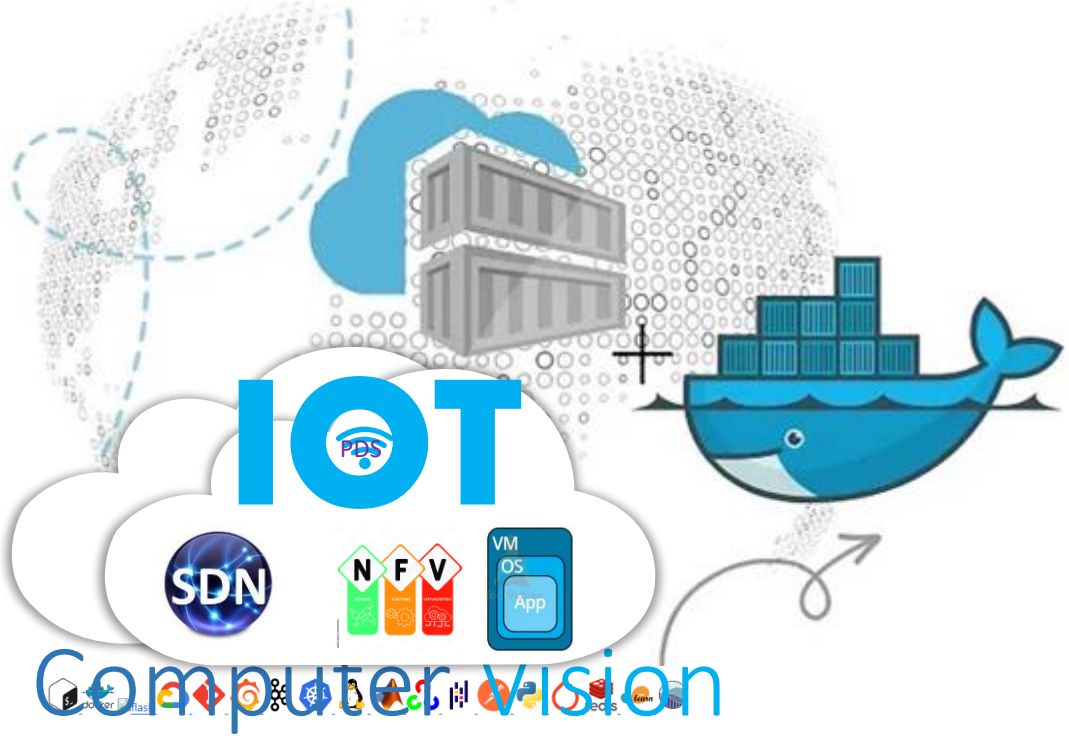
$\bar{h}^T$	X	h_1	h_2	h_3	h_4	=	$\bar{y}^T$
0.668 0.912 0.102 0.471		o_1	o_2				0.812 0.151

V

Forward Pass



$$\bar{y}^T = Sigmoid(\, Sigmoid(\bar{x}^T W) \vec{.} V \,)$$



Chapter 5-4:

Learning the parameters :

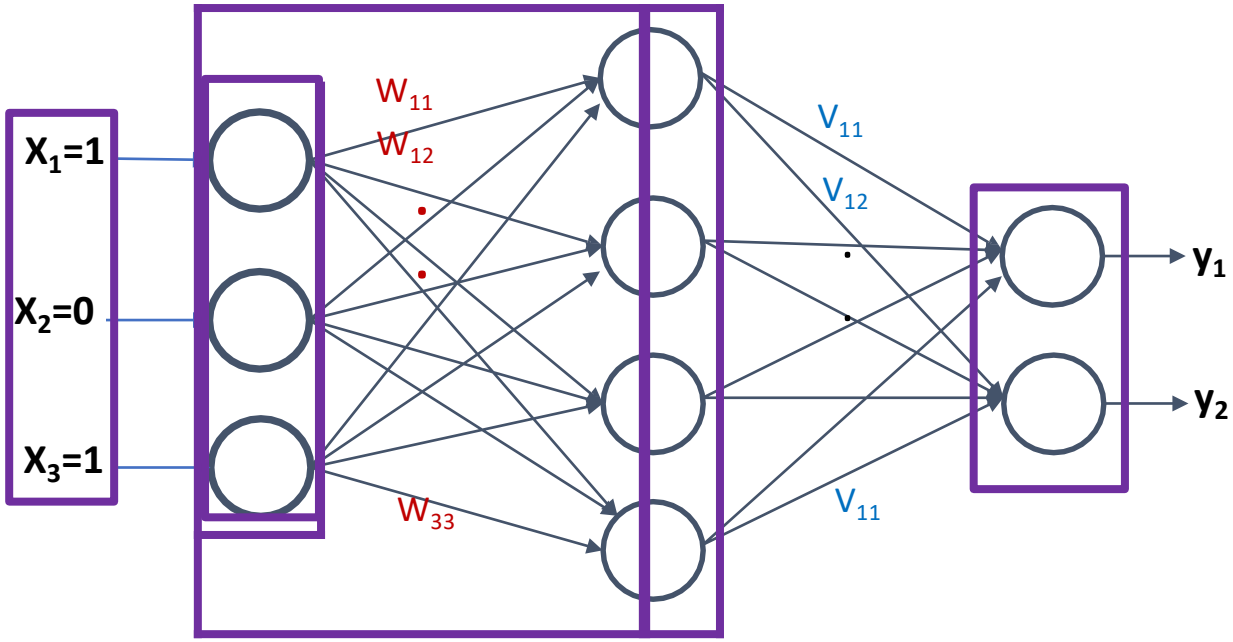
Backpropagation through time

https://github.com/amelcharolinesgn2/IoT_simulator-mqtt-NodeRed

https://github.com/siagianp
https://www.youtube.com/@AmelOline/videos

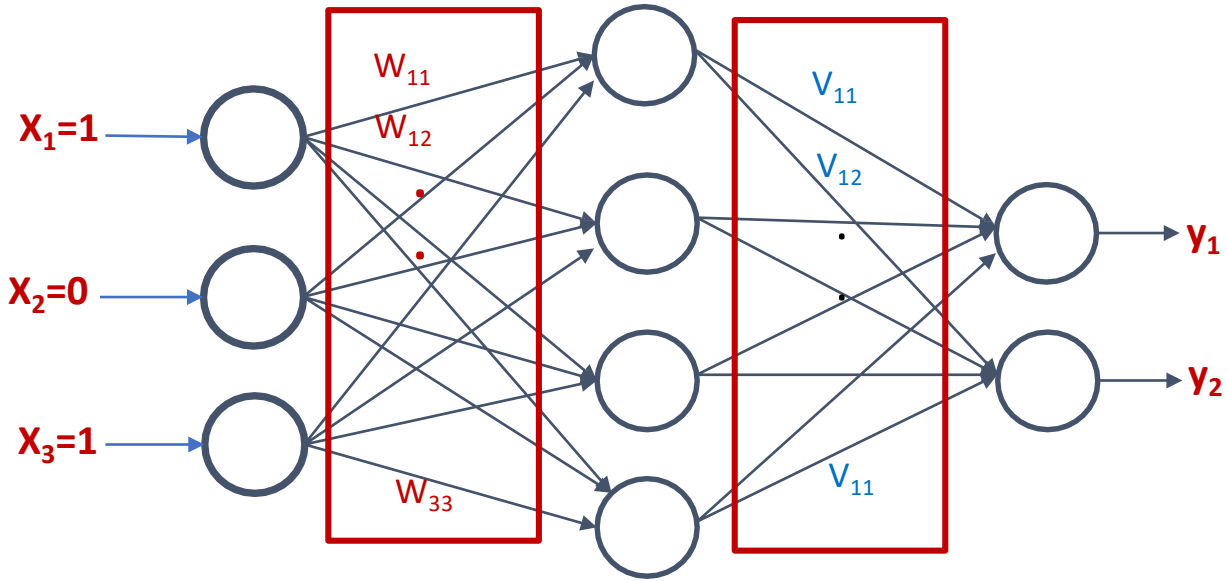
- What is loss function?
- What are the parameters of a multilayer perceptron neural network?
- How to estimate parameters using backpropagation through time?

What are the parameters?



$$\bar{y}^T = f_{hidden}\left(f_{output}\left(\bar{x}^T \boldsymbol{W}\right)^T . \boldsymbol{V}\right)$$

What are the parameters?



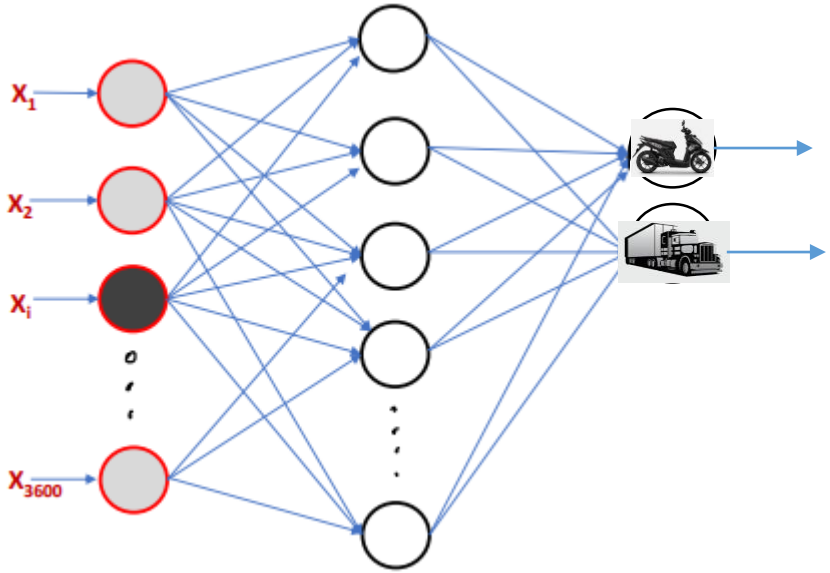
	h_1	h_2	h_3	h_4
i_1	W_{11}	W_{12}	W_{13}	W_{14}
i_2	W_{21}	W_{22}	W_{23}	W_{24}
i_3	W_{31}	W_{32}	W_{33}	W_{34}

W

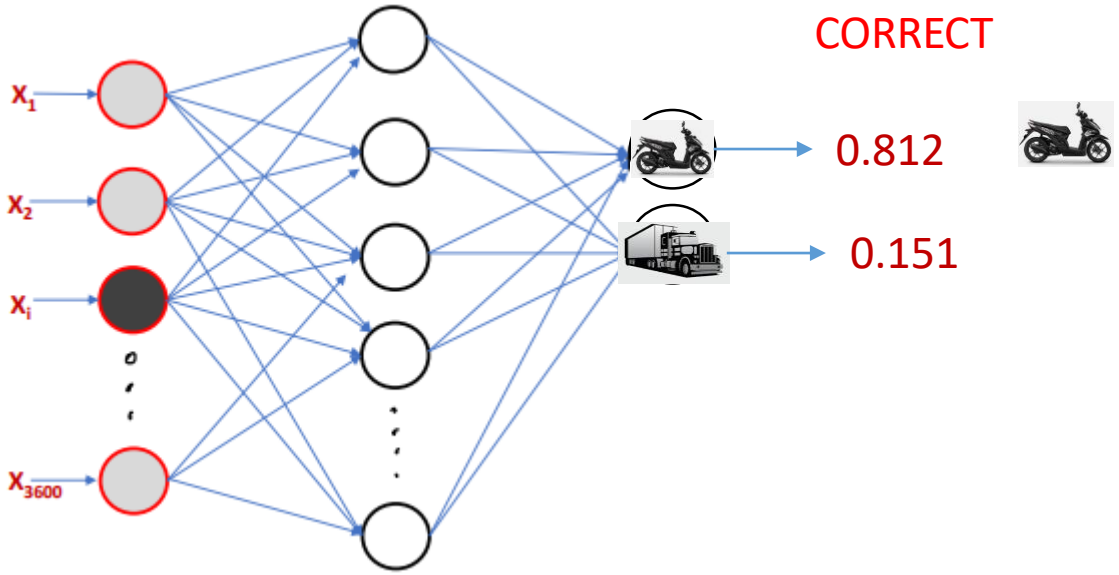
	o_1	o_2
h_1	V_{11}	V_{12}
h_2	V_{21}	V_{22}
h_3	V_{31}	V_{32}
h_4	V_{41}	V_{42}

V

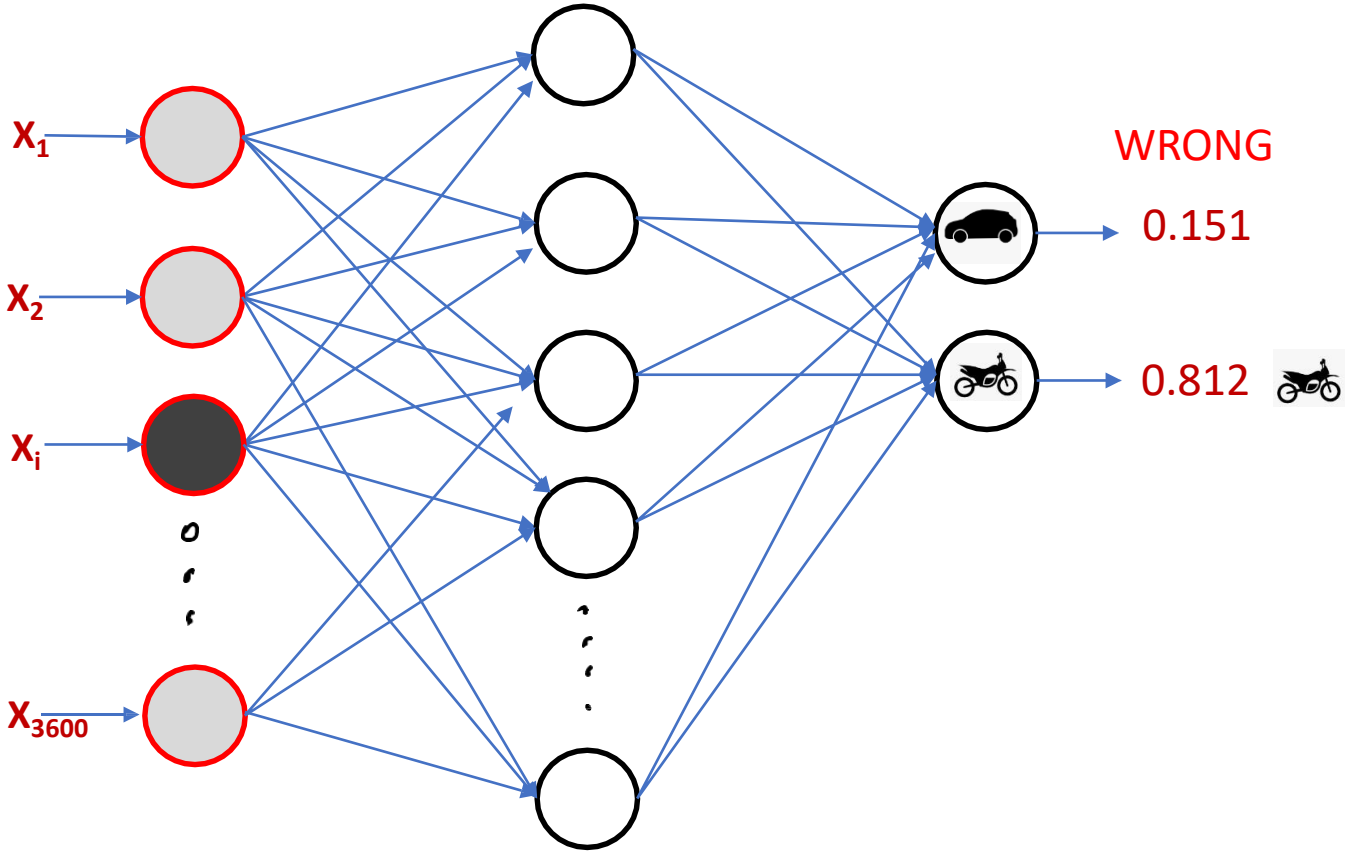
60x60=3600



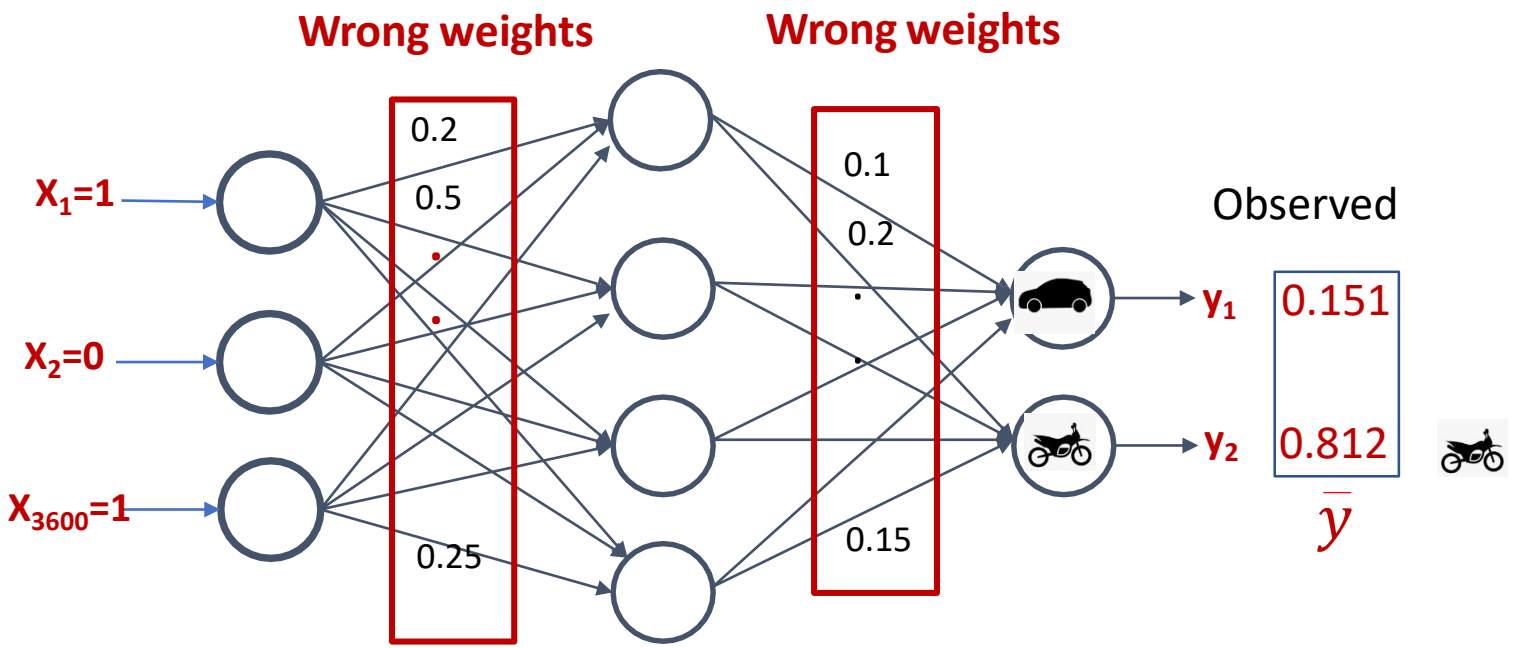
60x60=3600



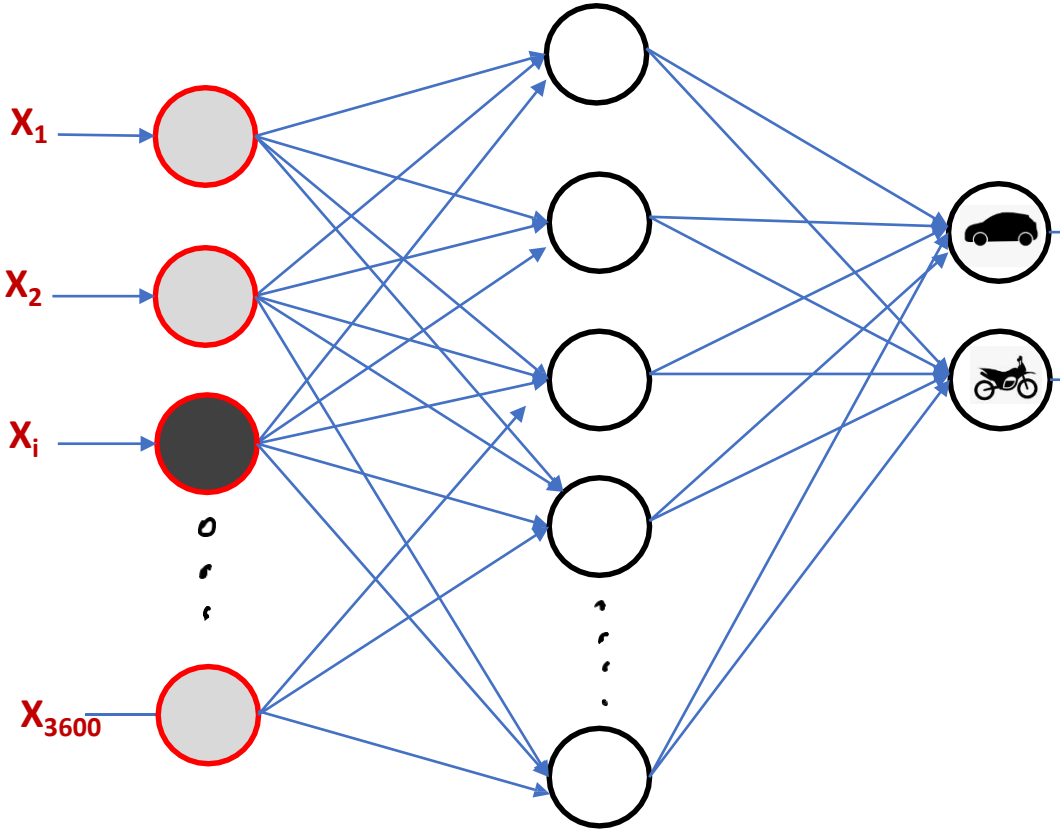
60x60=3600



Why there is an error in prediction?



60x60=3600



Observed

0.151

0.812

$\overline{y}$

Ground Truth

1

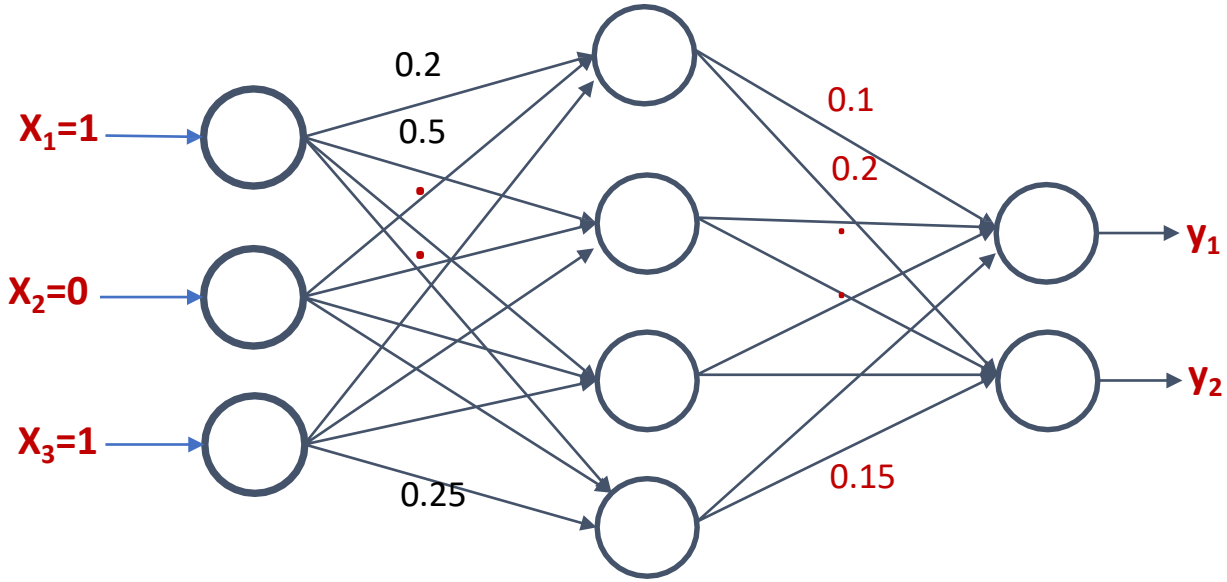
0

$\overline{\hat{y}}$

There is an error $E = \|\bar{y} - \hat{y}\|$

Also known as loss

Backpropagation Through Time

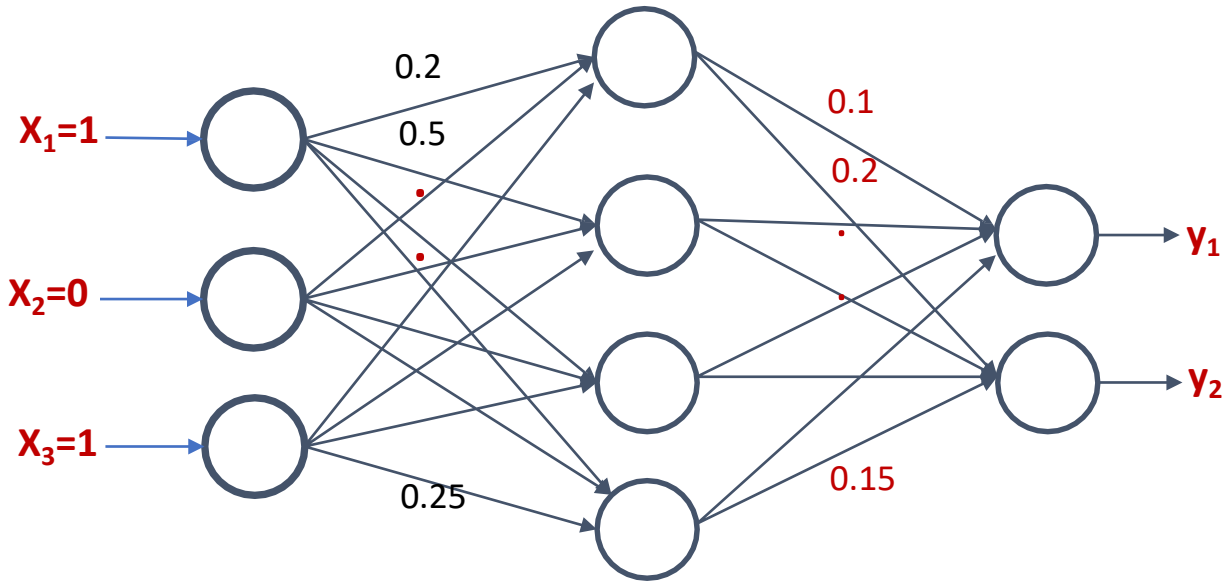


Minimize the Loss function

$$E = \|\bar{y} - \hat{y}\|$$

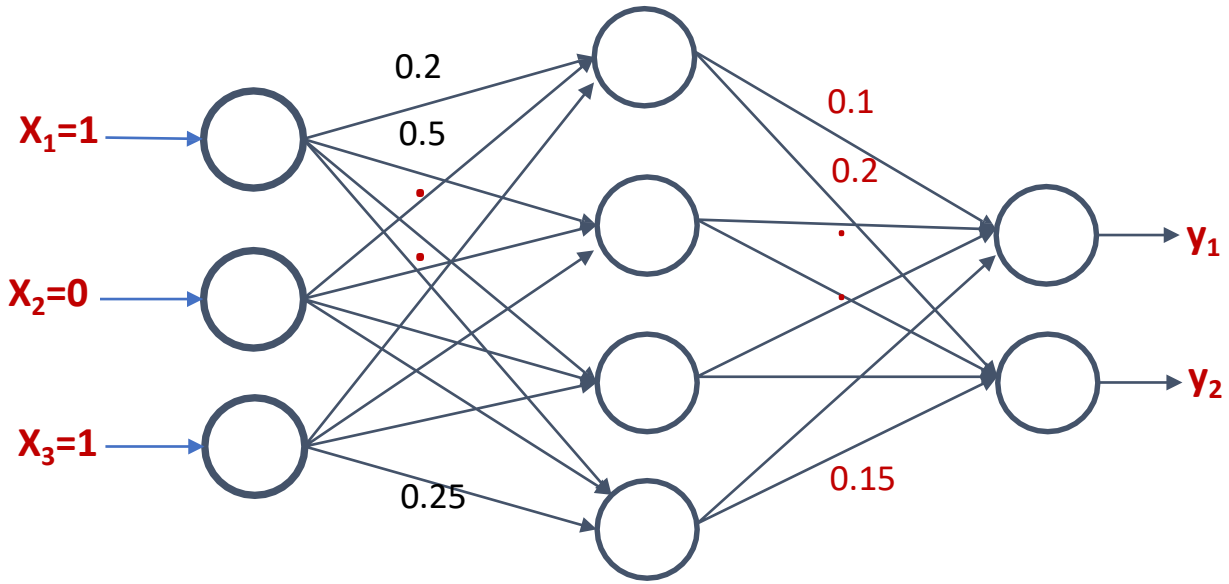
Backpropagation Through Time

Minimize the Loss function $E = \|\bar{y} - \hat{y}\|$



Backpropagation Through Time

Minimize the Loss function $E = \|\bar{y} - \hat{y}\|$

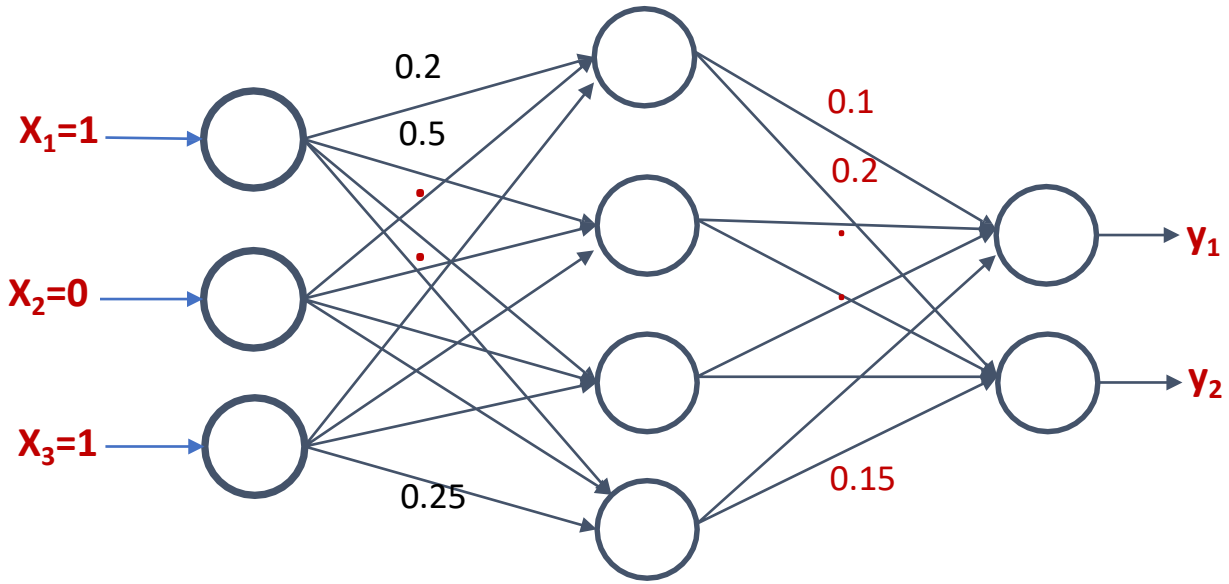


$$\nabla = \frac{\delta E}{\delta V} = 0$$

$$\nabla_{ij} = \frac{\delta E}{\delta V_{ij}} = \frac{\delta \|\bar{y} - \hat{y}\|}{\delta V_{ij}} = 0$$

Backpropagation Through Time

Minimize the Loss function $E = \|\bar{y} - \hat{y}\|$



$$\nabla = \frac{\delta E}{\delta V} = 0$$

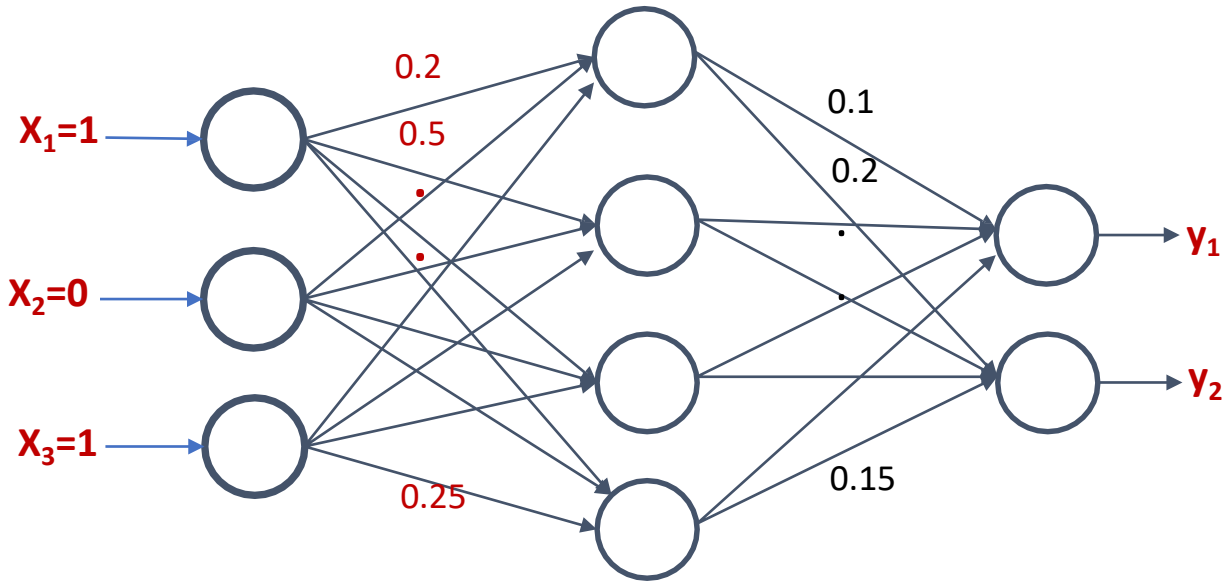
$$\nabla_{ij} = \frac{\delta E}{\delta V_{ij}} = \frac{\delta \|\bar{y} - \hat{y}\|}{\delta V_{ij}} = 0$$

$$\nabla = \frac{\delta E}{\delta V} = 0$$

$$V_{ij}^{t} = V_{ij}^{t-1} + \eta \nabla_{ij}^{t}$$

$$\eta = [0, 1]$$

Backpropagation Through Time

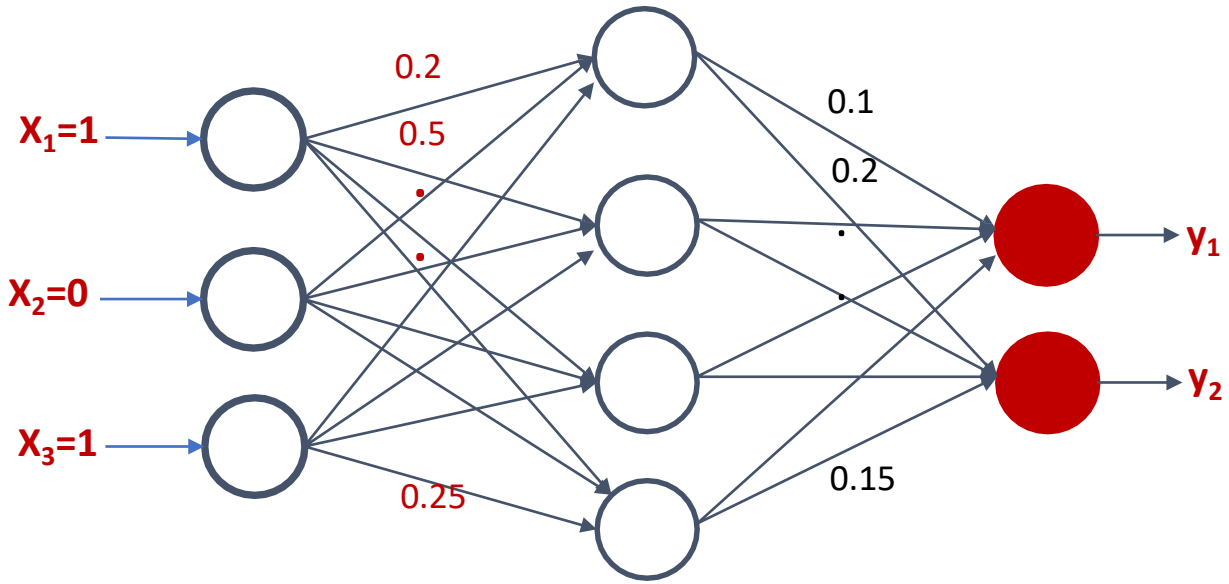


$$E = \|\bar{y} - \hat{y}\| = 0$$

or

$$E = \|\bar{y} - \hat{y}\| \leq \epsilon$$

Backpropagation Through Time

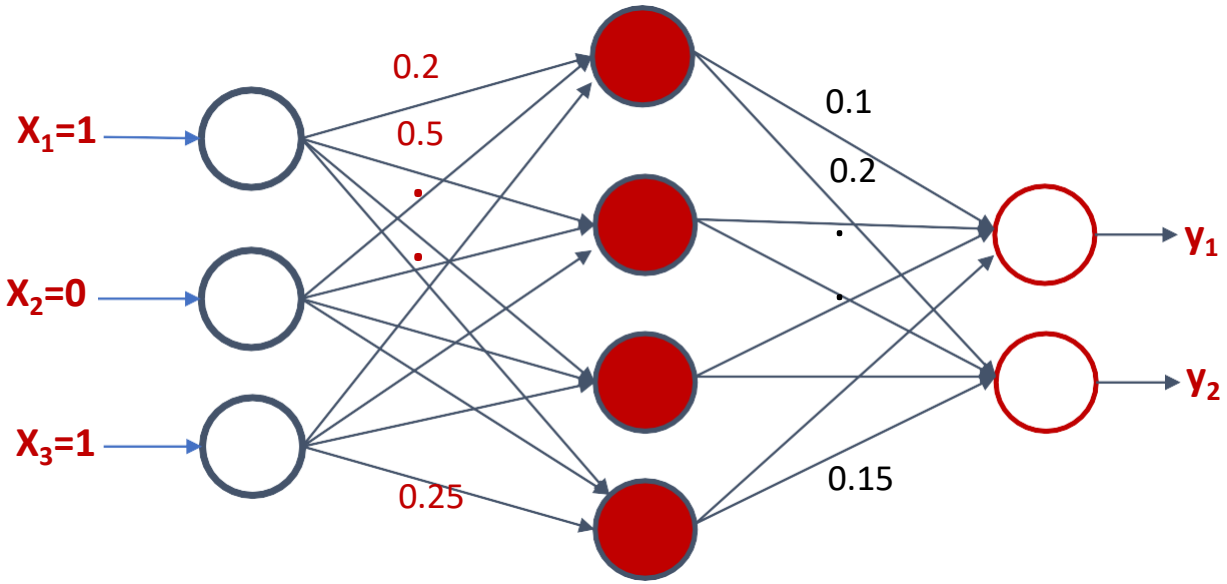


$$E = \|\bar{y} - \hat{y}\| = 0$$

or

$$E = \|\bar{y} - \hat{y}\| \leq \epsilon$$

Backpropagation Through Time

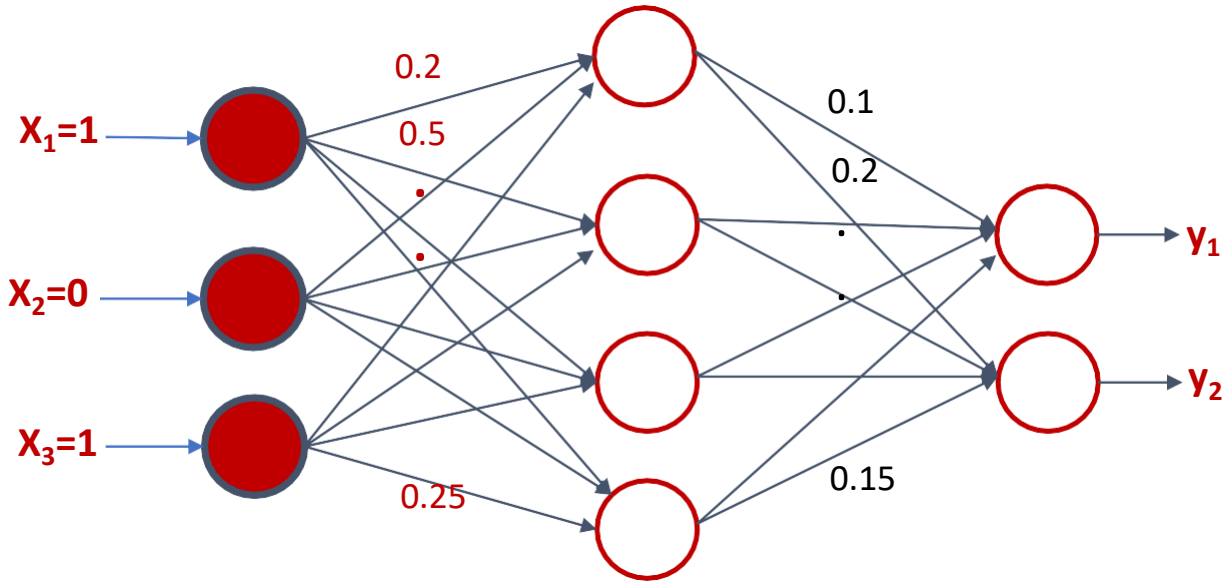


$$E = \|\bar{y} - \hat{y}\| = 0$$

or

$$E = \|\bar{y} - \hat{y}\| \leq \epsilon$$

Backpropagation Through Time

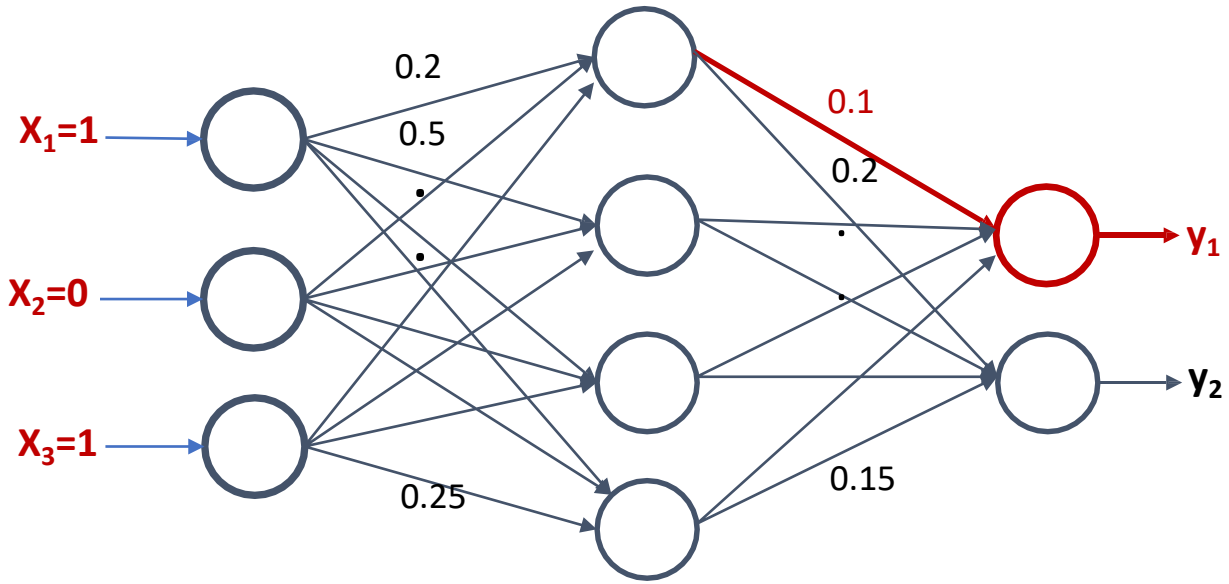


$$E = \|\bar{y} - \hat{y}\| = 0$$

or

$$E = \|\bar{y} - \hat{y}\| \leq \epsilon$$

How are the Derivatives performed



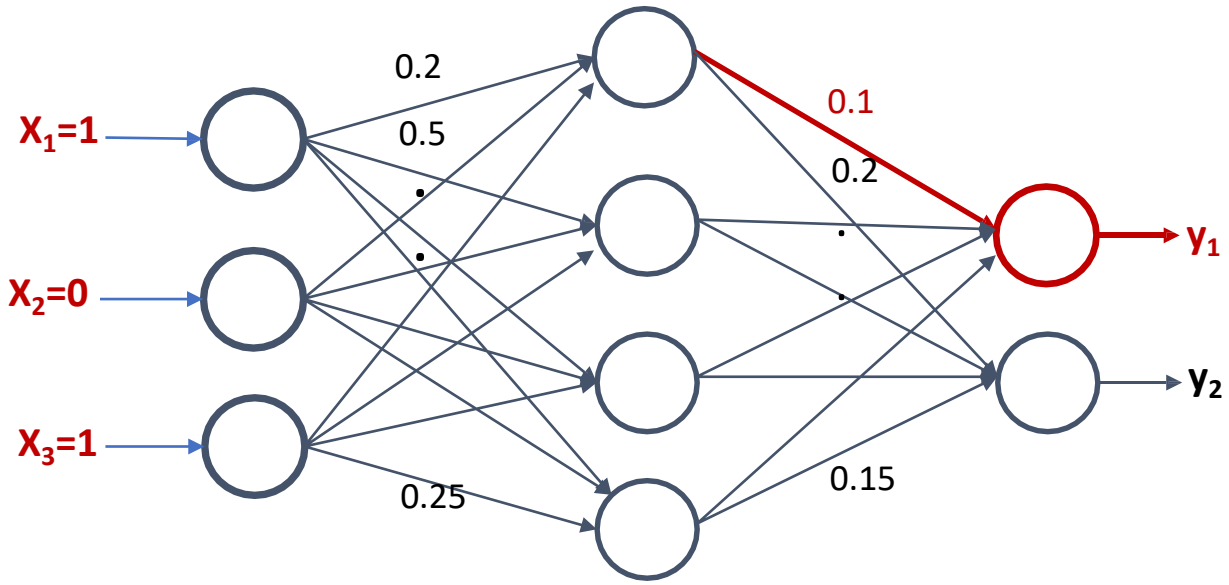
Loss function $E = \|\bar{y} - \hat{y}\|$

$$\nabla = \frac{\delta E}{\delta V} = 0$$

$$\nabla_{11} = \frac{\delta E}{\delta V_{11}} = 0$$

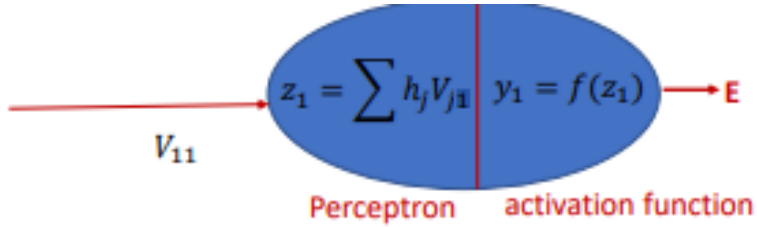
How are the Derivatives performed

Loss function $E = \|\bar{y} - \hat{y}\|$

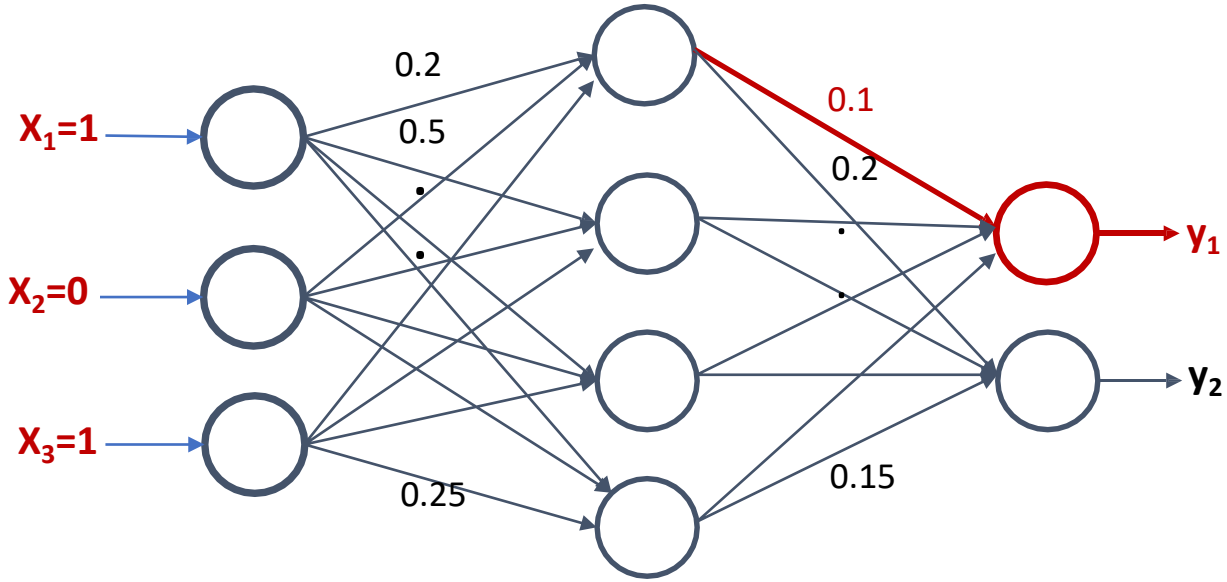


$$\nabla = \frac{\delta E}{\delta V} = 0$$

$$\nabla_{11} = \frac{\delta E}{\delta V_{11}} = 0$$



How are the Derivatives performed

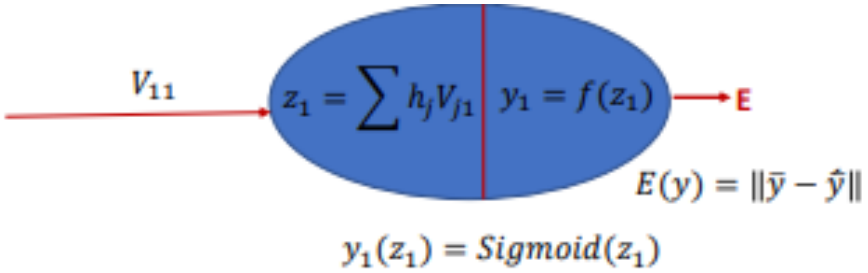


Loss function

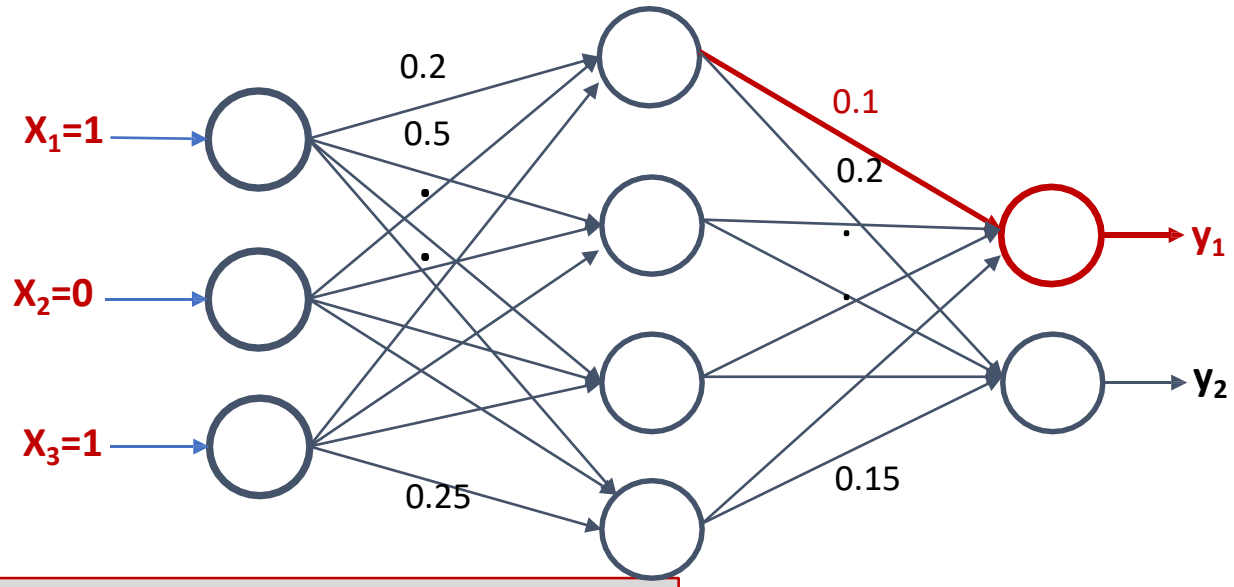
$$\nabla = \frac{\delta E}{\delta V} = 0$$

$$\nabla_{11} = \frac{\delta E}{\delta V_{11}} = 0$$

$$z_1(V_{11}) = h_1V_{11} + h_2V_{21} + h_3V_{31} + h_4V_{41}$$



How are the Derivatives performed



Loss function

$$E = \|\bar{y} - \hat{y}\|$$

$$\nabla = \frac{\delta E}{\delta V} = 0$$

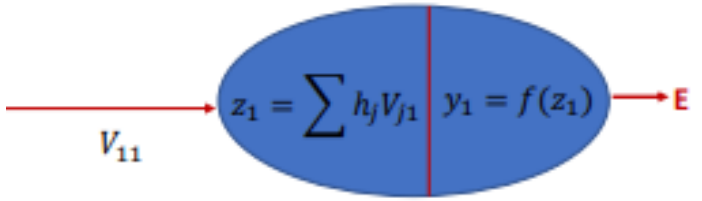
$$\nabla_{ij} = \frac{\delta E}{\delta V_{ij}} = 0$$

$$\nabla_{11} = \frac{\delta E}{\delta V_{11}} = \frac{\delta z_1}{\delta V_{11}} \cdot \frac{\delta y_1}{\delta z_1} \cdot \frac{\delta E}{\delta y_1}$$

$$z_1 = h_1 V_{11} + h_2 V_{21} + h_3 V_{31} + h_4 V_{41}$$

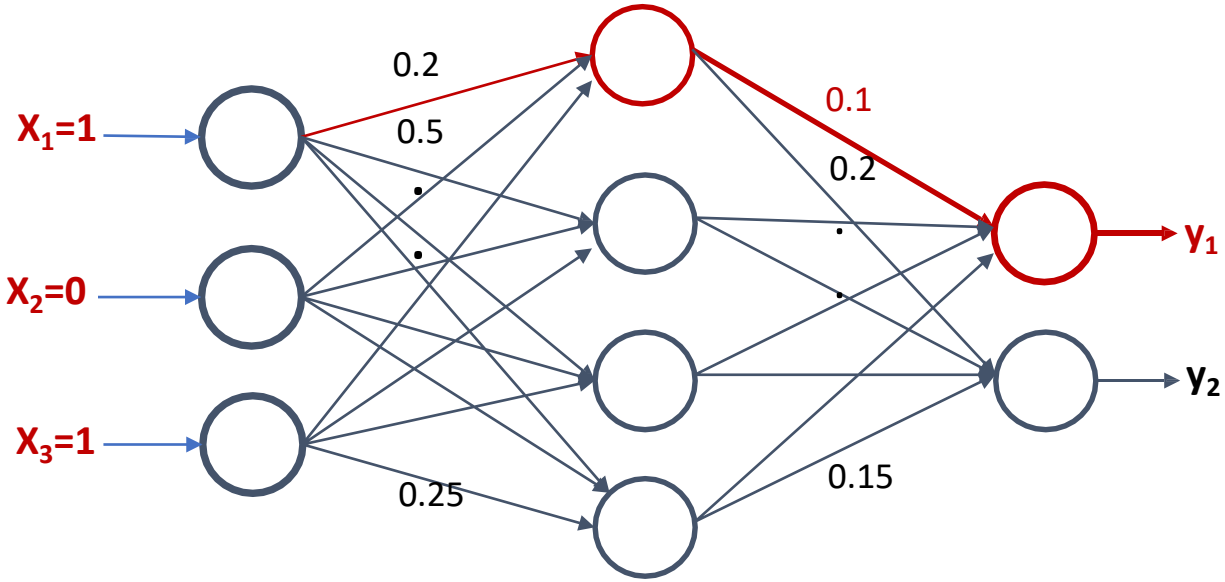
$$y_1 = Sigmoid(\mathbf{z_1})$$

$$\frac{\delta E}{\delta V_{11}} = \frac{\delta z_1}{\delta V_{11}} \times \frac{\delta y_1}{\delta z_1} \times \frac{\delta E}{\delta y_1}$$



Backpropagation

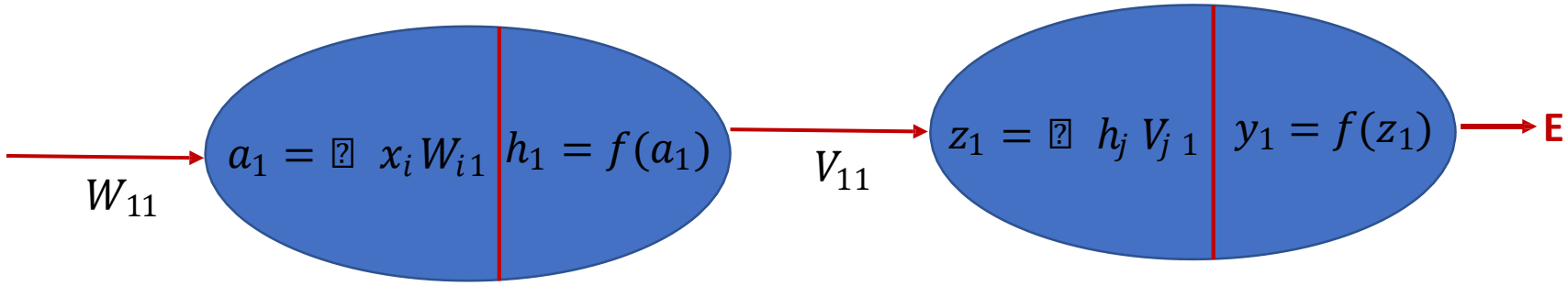
Loss function $E = \|\bar{y} - \hat{y}\|$



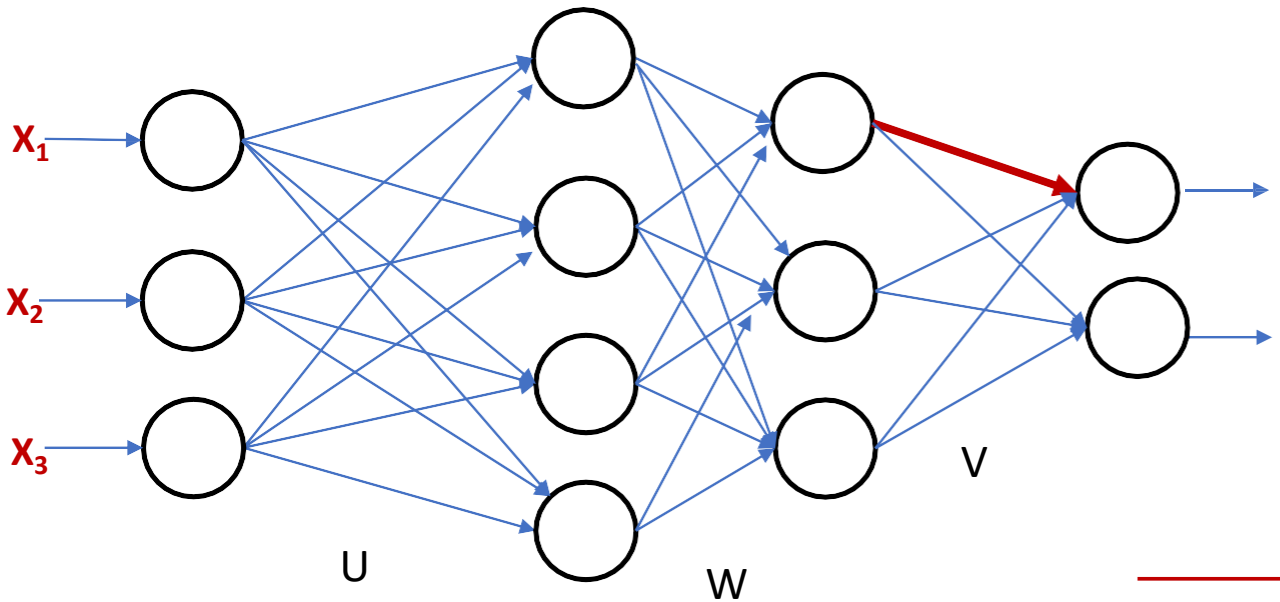
$$\nabla = \frac{\delta E}{\delta W} = 0$$

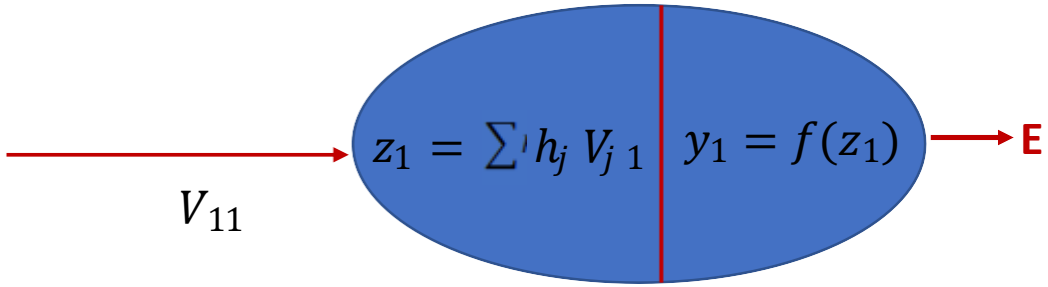
$$\nabla_i = \frac{\delta E}{\delta W_i} = 0$$

$$\frac{\delta E}{\delta W_{11}} = \frac{\delta a_1}{\delta W_{11}} \times \frac{\delta h_1}{\delta a_1} \times \frac{\delta z_1}{\delta h_1} \times \frac{\delta y_1}{\delta z_1} \times \frac{\delta E}{\delta y_1}$$



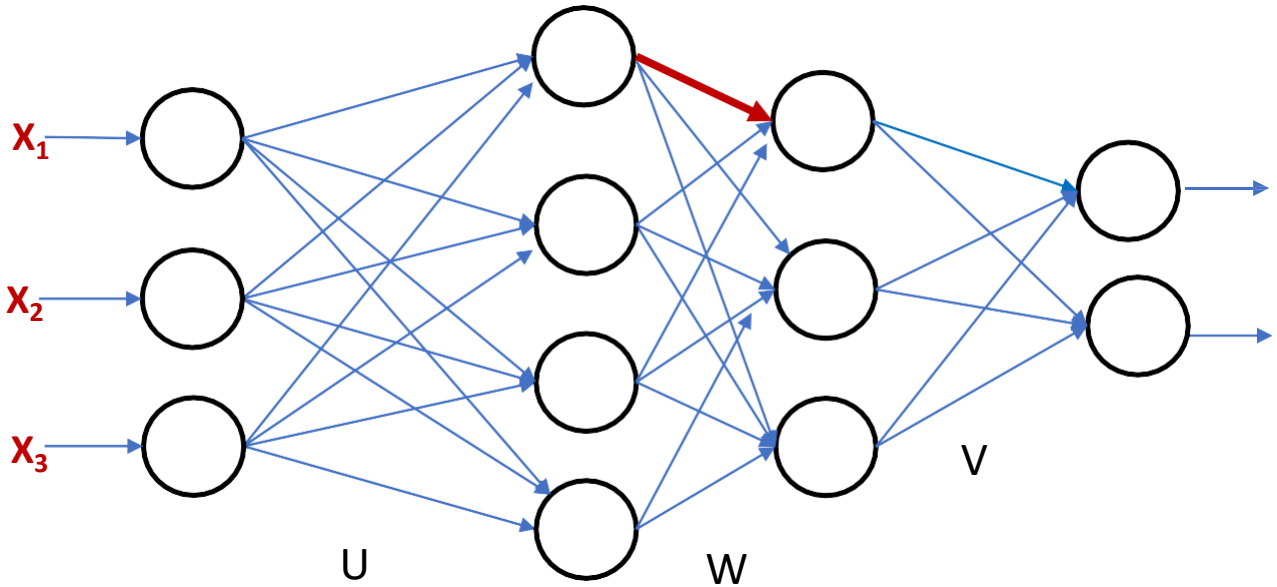
We may have multiple layers.



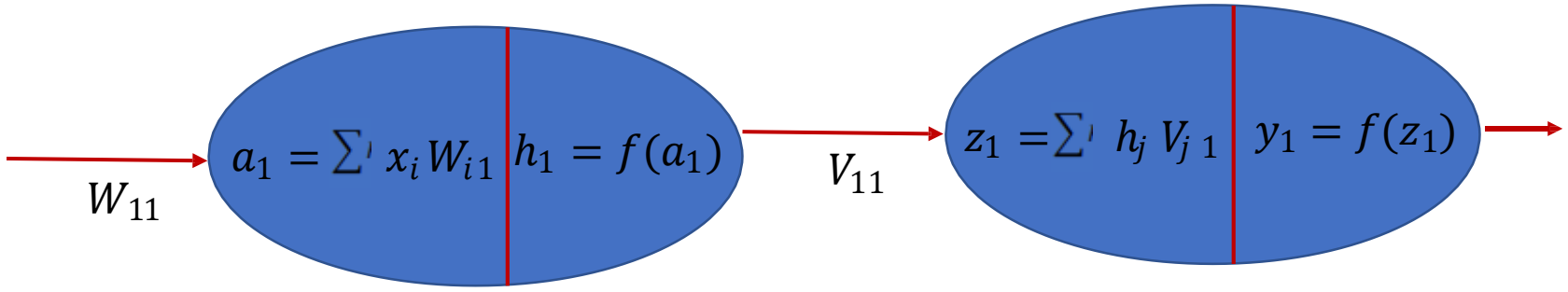


$$\frac{\delta E}{\delta V_{11}} = \frac{\delta z_1}{\delta V_{11}} \times \frac{\delta y_1}{\delta z_1} \times \frac{\delta E}{\delta y_1}$$

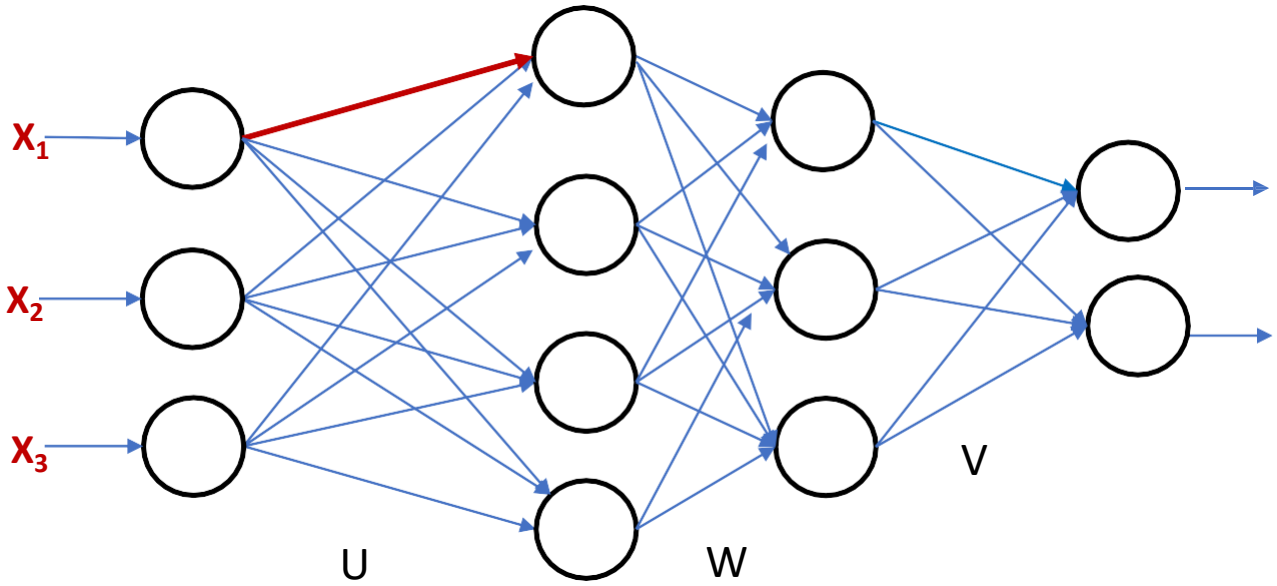
We may have multiple layers



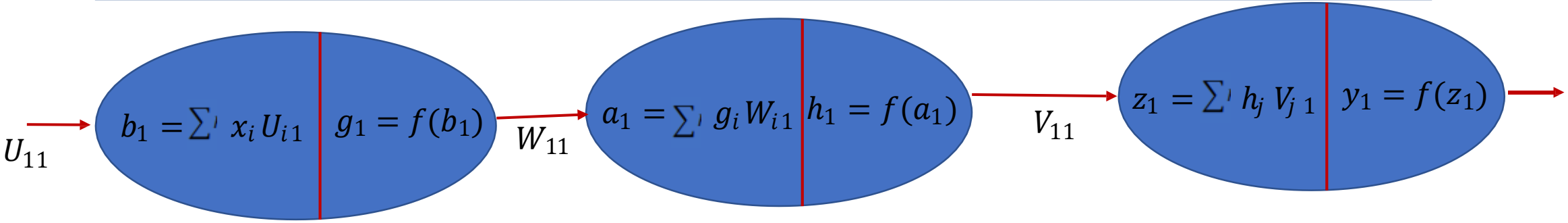
$$\frac{\delta E}{\delta W_{11}} = \frac{\delta a_1}{\delta W_{11}} \times \frac{\delta h_1}{\delta a_1} \times \frac{\delta z_1}{\delta h_1} \times \frac{\delta y_1}{\delta z_1} \times \frac{\delta E}{\delta y_1}$$

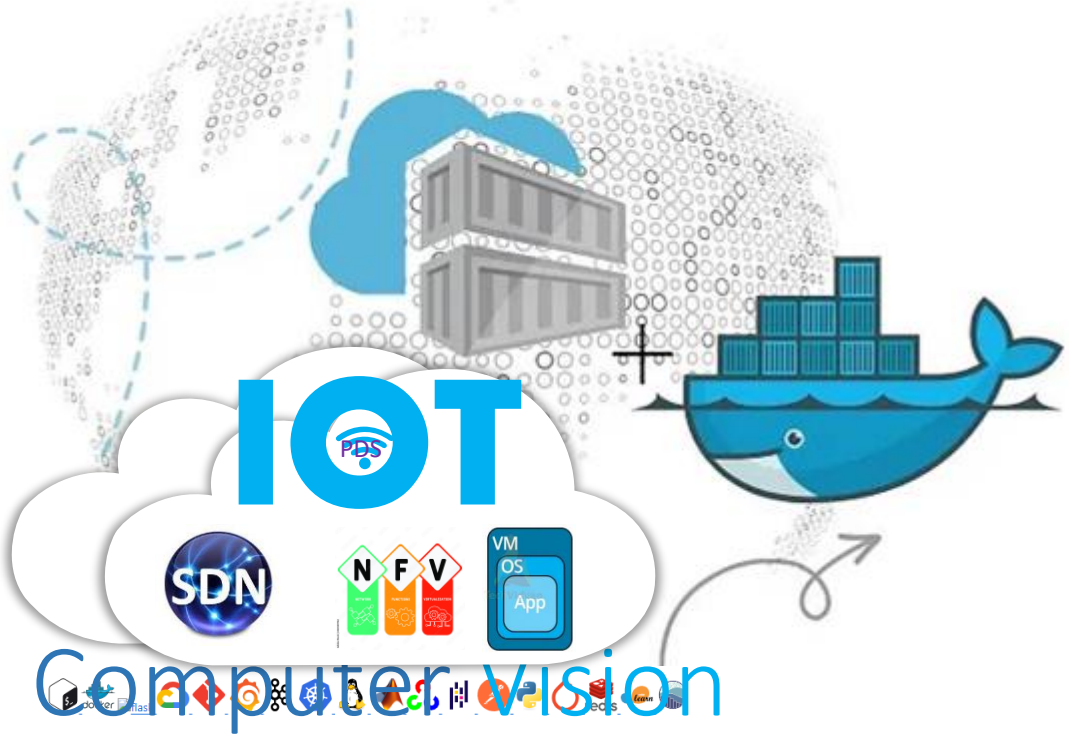


We may have multiple layers



$$\frac{\delta E}{\delta U_{11}} = \frac{\delta b_1}{\delta U_{11}} \times \frac{\delta g_1}{\delta b_1} \times \frac{\delta a_1}{\delta g_1} \times \frac{\delta h_1}{\delta a_1} \times \frac{\delta z_1}{\delta h_1} \times \frac{\delta y_1}{\delta z_1} \times \frac{\delta E}{\delta y_1}$$





Chapter 5-5:

Learning with different Loss Functions and Their Derivatives

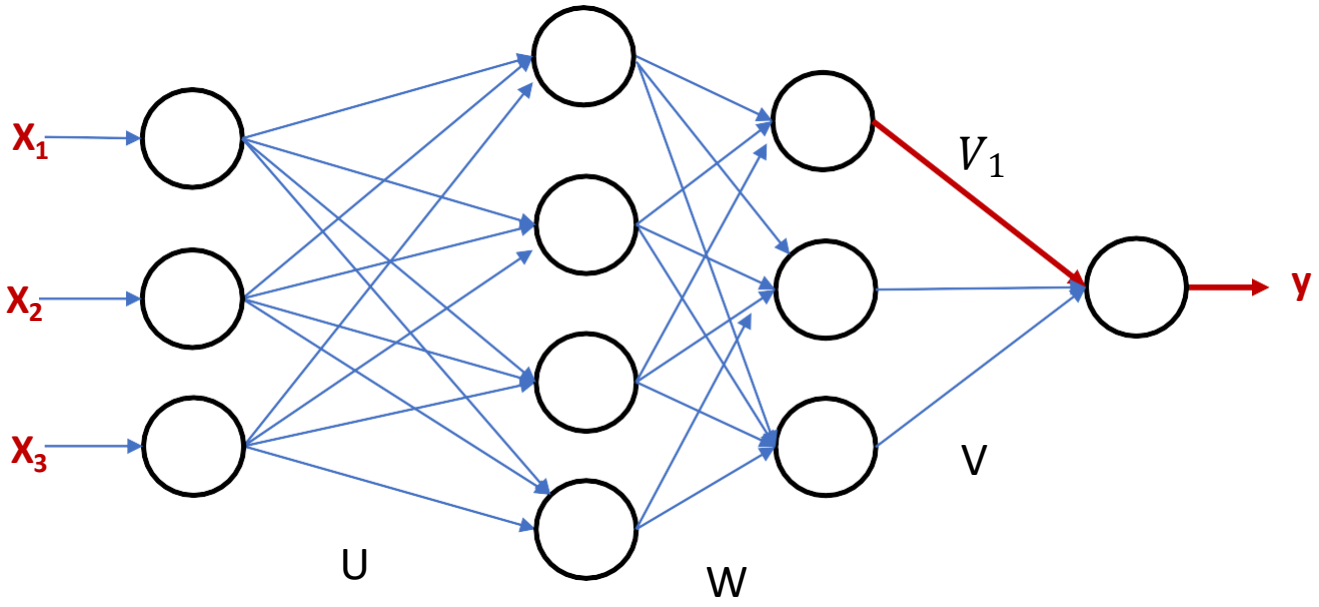
https://github.com/amelcharolinesgn2/IoT_simulator-mqtt-NodeRed

https://github.com/siagianp
https://www.youtube.com/@AmelOline/videos

Two Commonly used Loss Functions are

- **Mean Square Error – Standard Loss Function for Regression**
- Cross Entropy Loss - Standard Loss Function for Classification

Mean Square Error (MSE)



For the Single Sample

$$\text{MSE } E = (y - \hat{y})^2$$

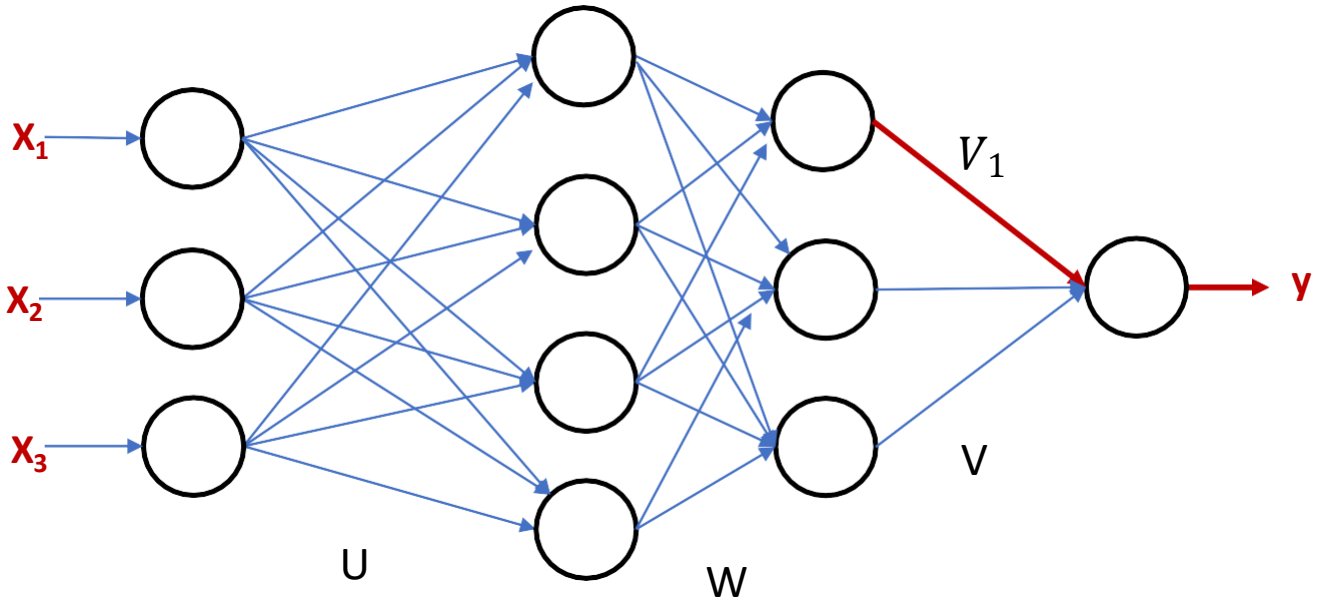
Ground truth is

y

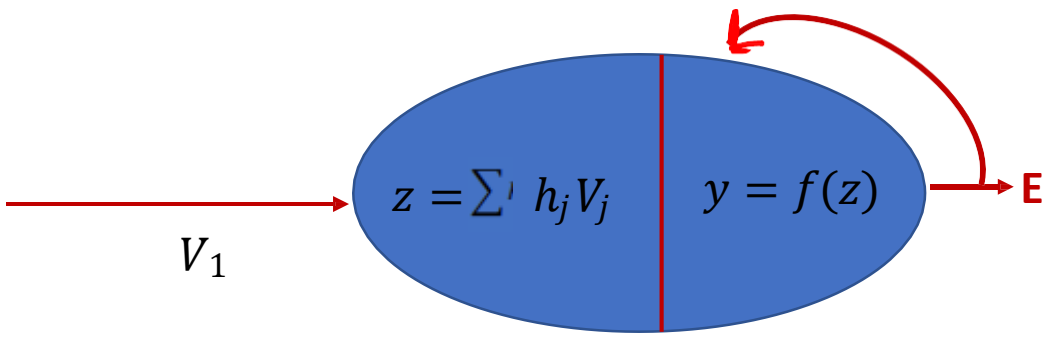
Mean Square Error (MSE)

For the Single Sample

MSE $E = (y - \hat{y})^2$

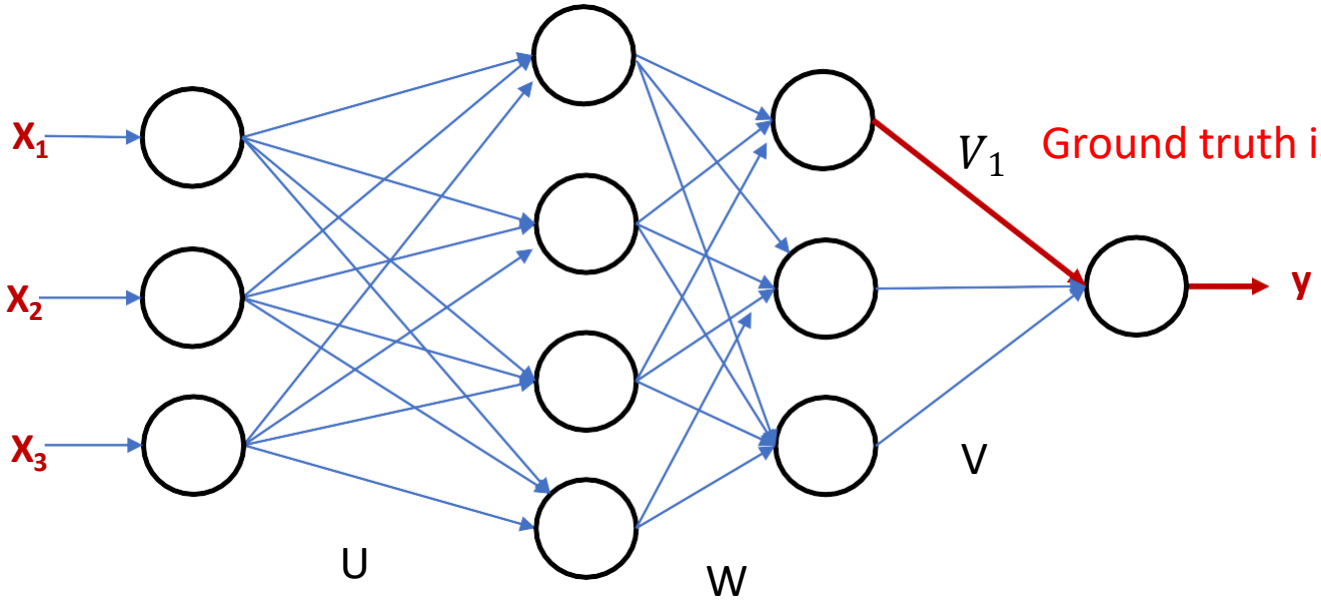


Ground truth is y



$$\frac{\delta E}{\delta V_1} = \frac{\delta z}{\delta V_1} \times \frac{\delta y}{\delta z} \times \frac{\delta E}{\delta y}$$

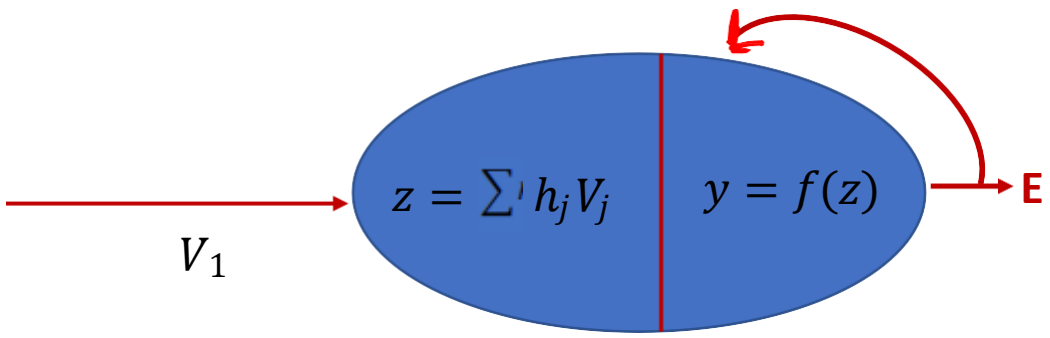
Mean Square Errors



For the Single Sample

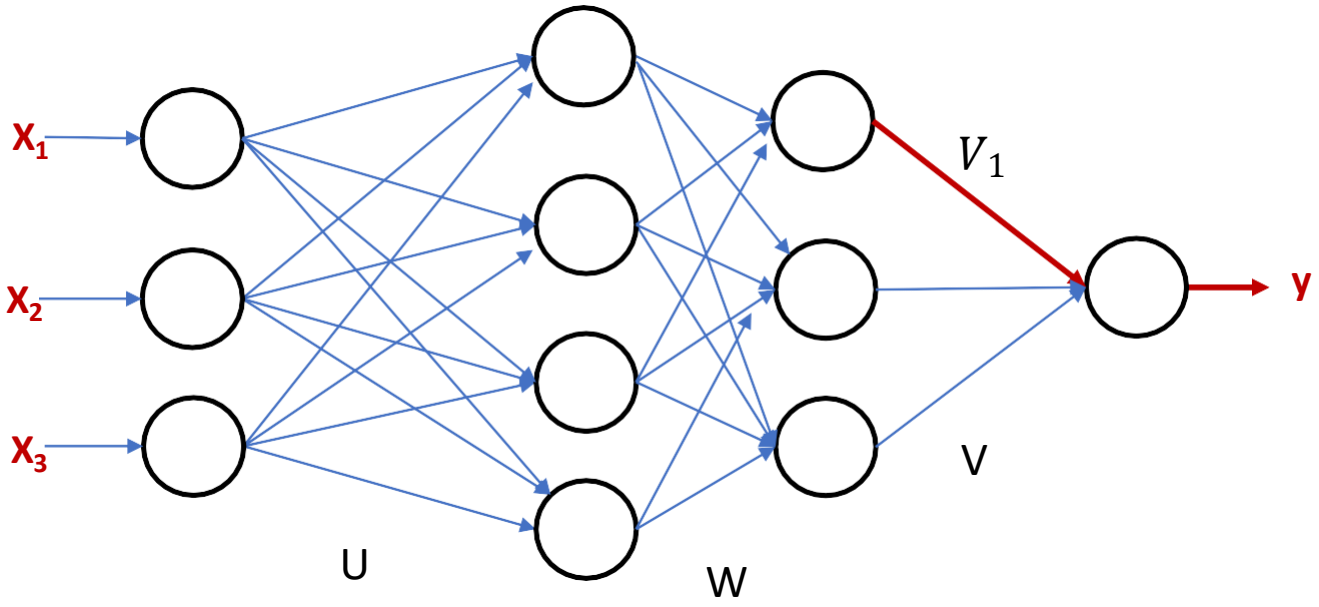
MSE $E = (y - \hat{y})^2$

$$\frac{\delta E}{\delta y} = \frac{\delta(y - \hat{y})^2}{\delta y} = 2(y - \hat{y})$$



$$\frac{\delta E}{\delta V_1} = \frac{\delta z}{\delta V_1} \times \frac{\delta y}{\delta z} \times \frac{\delta E}{\delta y}$$

Mean Square Errors

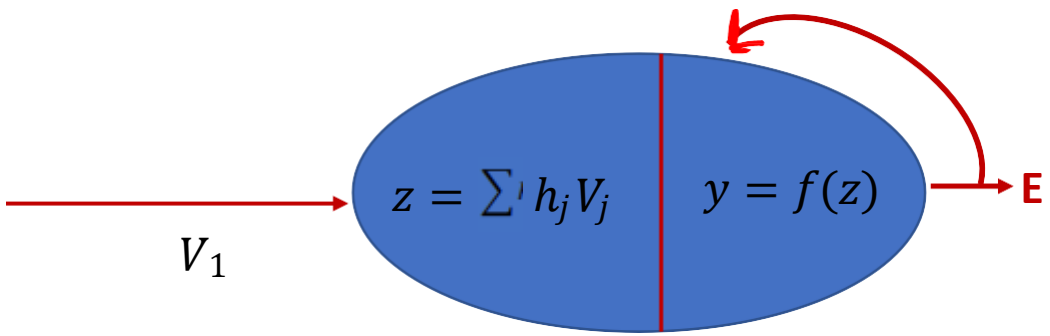


If the ground truth is y

For n Samples

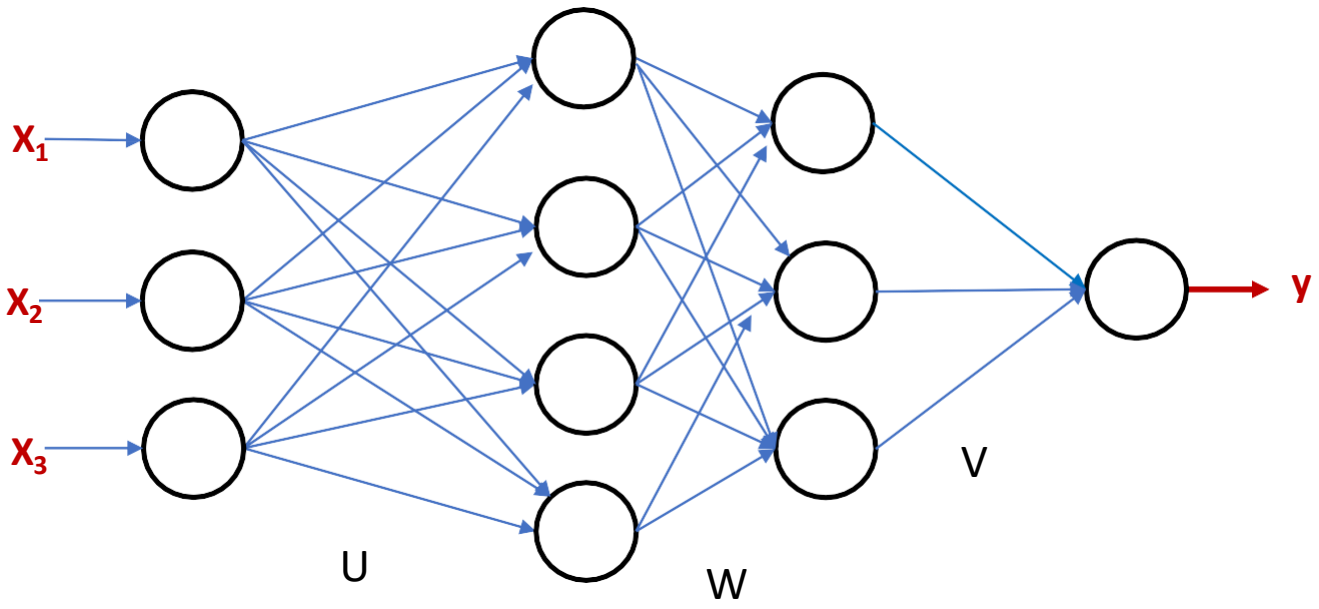
$$E = \frac{1}{n}\sum_{i=1}^{n}(y^i - \hat{y}^i)^2$$

$$\frac{\delta E}{\delta y} = \frac{\delta \frac{1}{n}\sigma_{i=1}^{n}(y^i - \hat{y}^i)^2}{\delta y} = \frac{2}{n}\sum_{i=1}^{n}(y^i - \hat{y}^i)$$



$$\frac{\delta E}{\delta V_1} = \frac{\delta z}{\delta V_1} \times \frac{\delta y}{\delta z} \times \frac{\delta E}{\delta y}$$

Mean Square Errors



If the ground truth is y

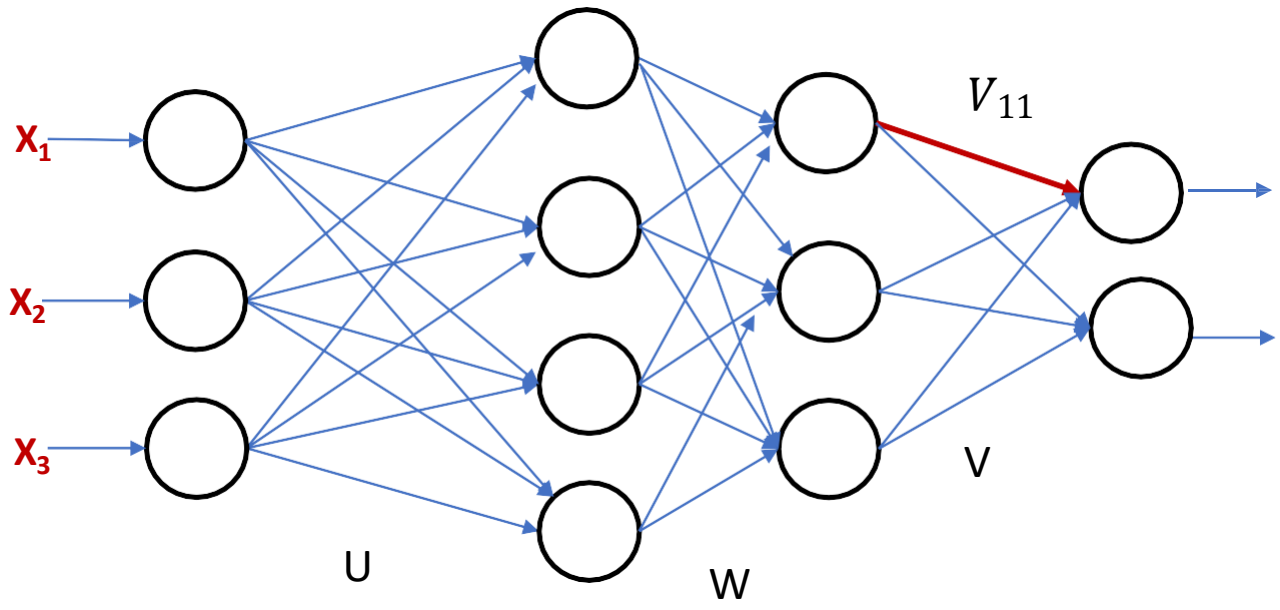
For n Samples

$$E = \frac{1}{n}\sum_{i=1}^{n}(y^i - \hat{y}^i)^2$$

$$\frac{\delta E}{\delta y} = \frac{\delta \frac{1}{n}\sum_{i=1}^{n}(y^i - \hat{y}^i)^2}{\delta y} = \frac{2}{n}\sum_{i=1}^{n}(y^i - \hat{y}^i)$$

Backpropagation will be done after a batch of n Samples

Mean Square Errors

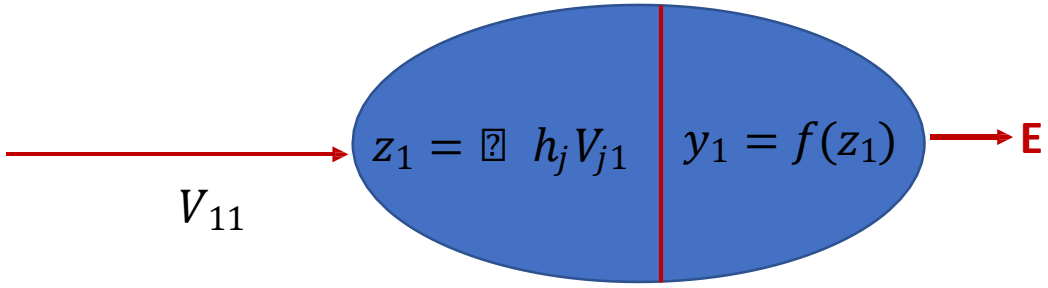


If the ground truth is y

For the n Sample

$$\boldsymbol{E} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{y}^i - \boldsymbol{y}^i)^2$$

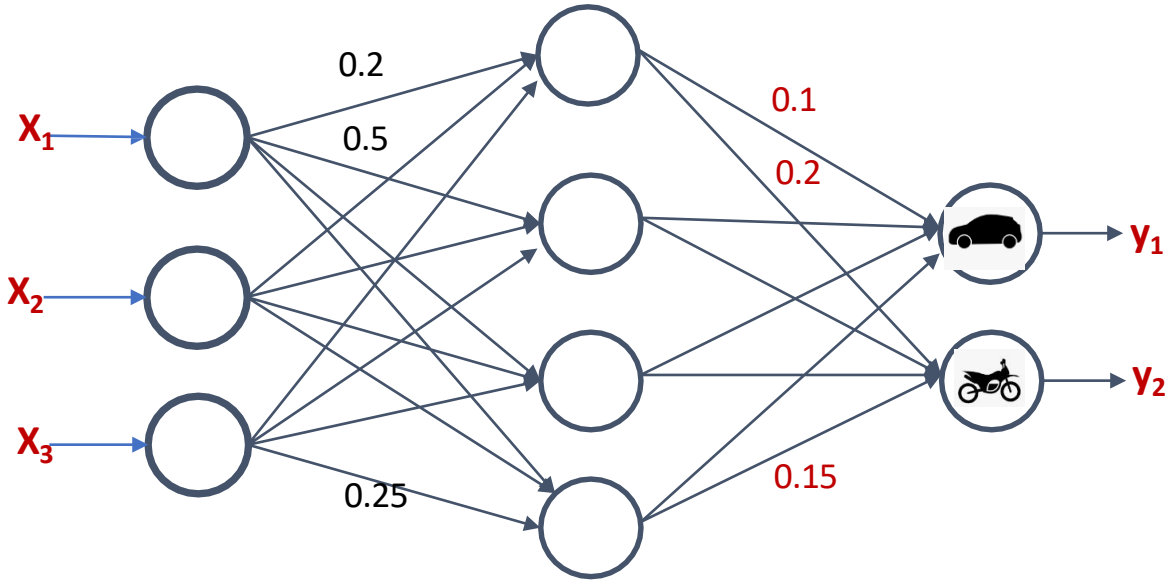
$$\frac{\delta \boldsymbol{E}}{\delta y_1} = \frac{\delta \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{y}^i - \boldsymbol{y}^i)^2}{\delta y_1} = \frac{2}{n}\sum_{i=1}^{n}(y_1{}^i - y^i)$$



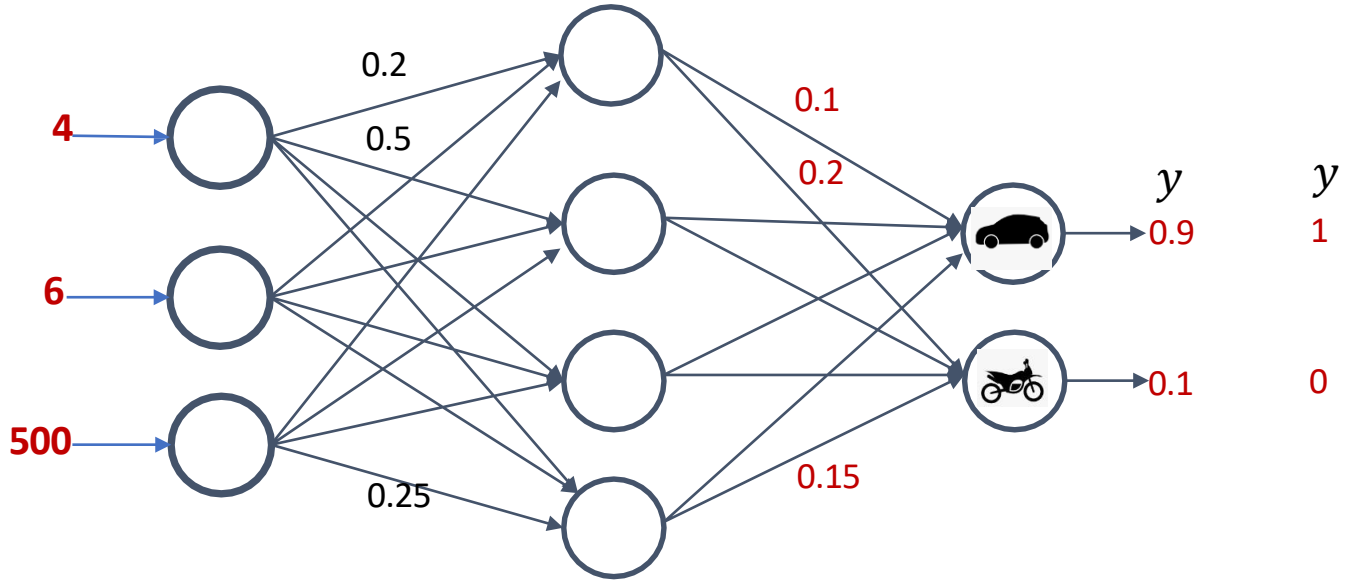
$$\frac{\delta E}{\delta V_{11}} = \frac{\delta z_1}{\delta V_{11}} \times \frac{\delta y_1}{\delta z_1} \times \frac{\delta \boldsymbol{E}}{\delta y_1}$$

Let us illustrate with a toy example

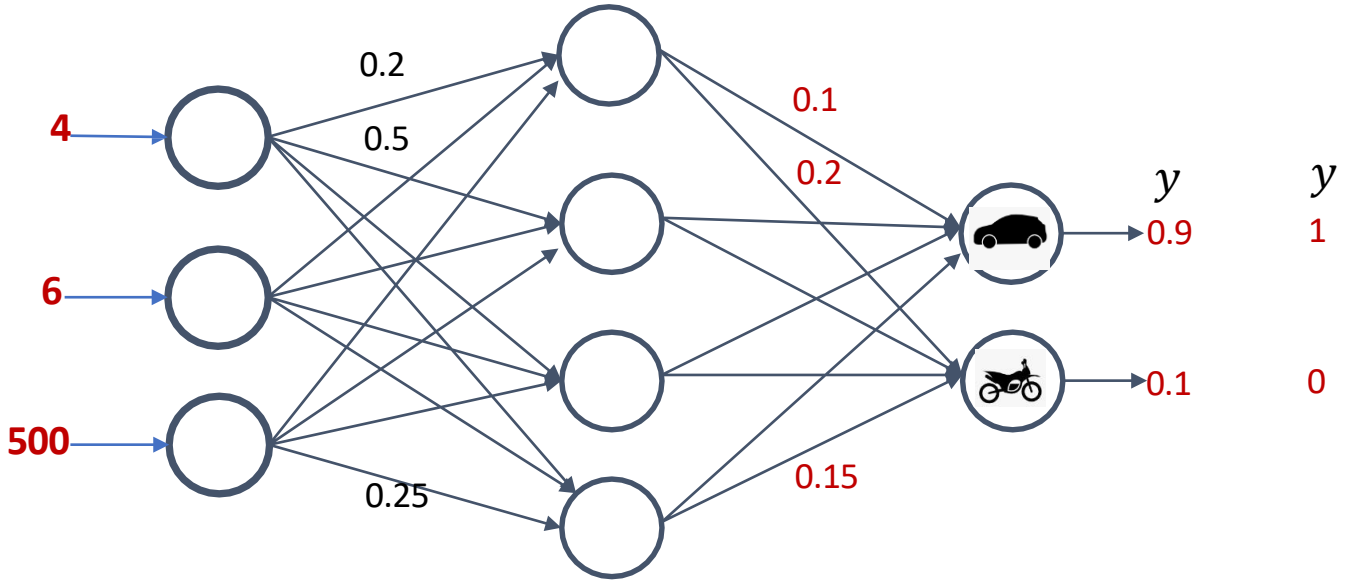
#Wheel	Height	Weight	
4	6	500	
4	5.5	600	
4	5	550	
2	3	200	
2	3.5	150	
2	4	250	



#Wheel	Height	Weight	
4	6	500	
4	5.5	600	
4	5	550	
2	3	200	
2	3.5	150	
2	4	250	



#Wheel	Height	Weight	
4	6	500	
4	5.5	600	
4	5	550	
2	3	200	
2	3.5	150	
2	4	250	



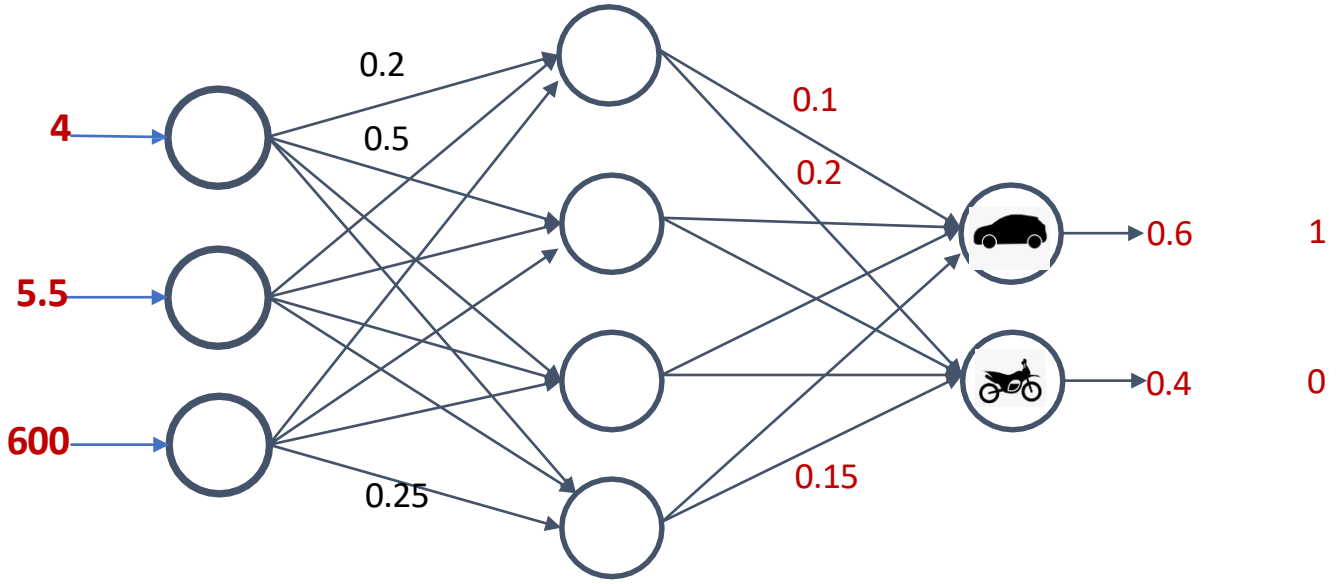
Error $E = (y - \hat{y})^2$

Error $E_{y_1} = (0.9 - 1)^2 = 0.01$

Error $E_{y_2} = (0.1 - 0)^2 = 0.01$

If these errors are not acceptable, then Backpropagate.

#Wheel	Height	Weight	
4	6	500	
4	5.5	600	
4	5	550	
2	3	200	
2	3.5	150	
2	4	250	



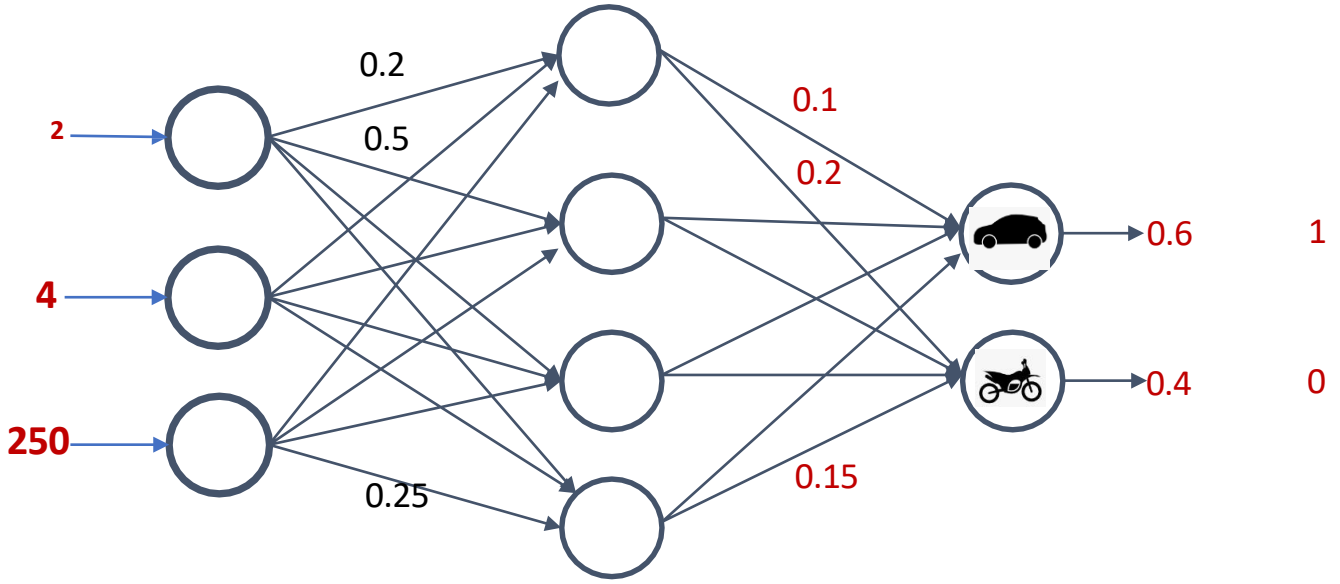
Error $E = (y - yy)^2$

Error $E_{y_1} = (0.6 - 1)^2 = 0.16$

Error $E_{y_2} = (0.4 - 0)^2 = 0.36$

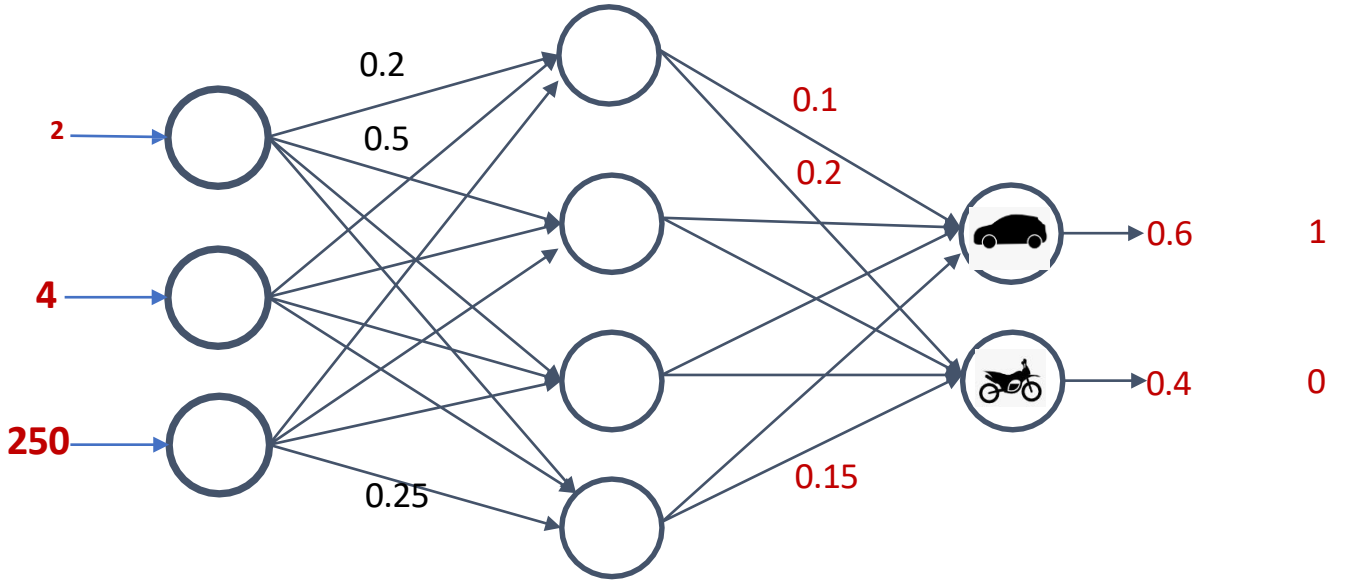
If these errors are not acceptable, then Backpropagate.

#Wheel	Height	Weight	
4	6	500	
4	5.5	600	
4	5	550	
2	3	200	
2	3.5	150	
2	4	250	



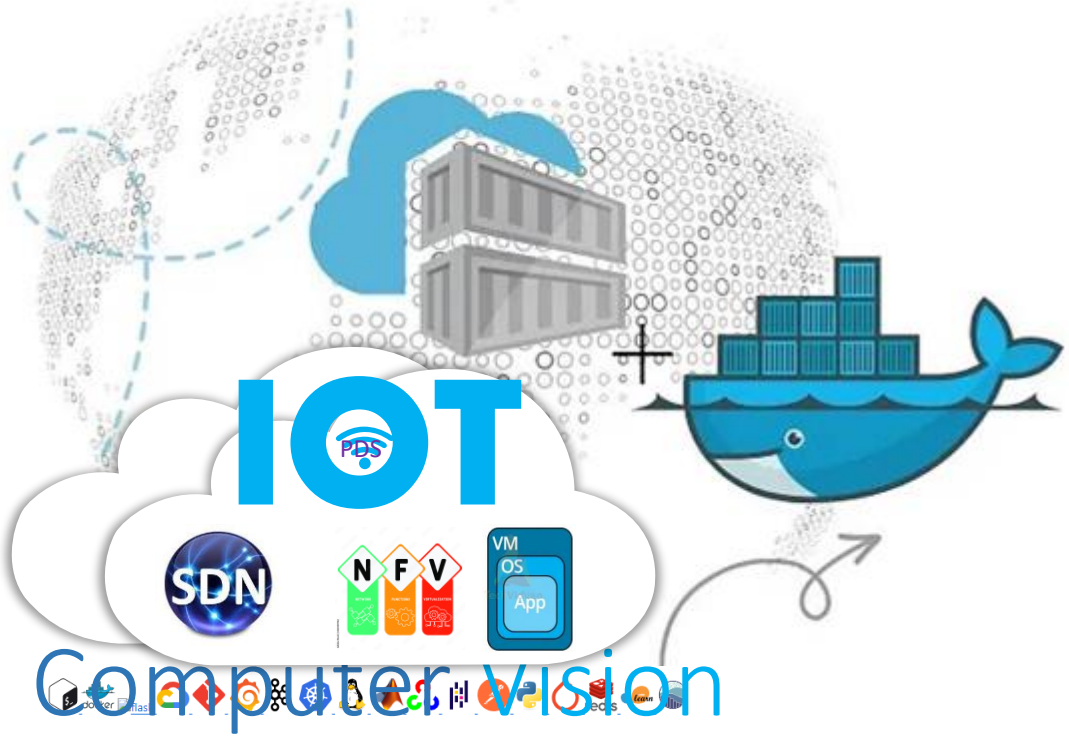
One complete cycle of training is called Epoch

#Wheel	Height	Weight	
4	6	500	
4	5.5	600	
4	5	550	
2	3	200	
2	3.5	150	
2	4	250	



Backpropagation after every sample is expensive.

Do it in batches.



Chapter 5-6:

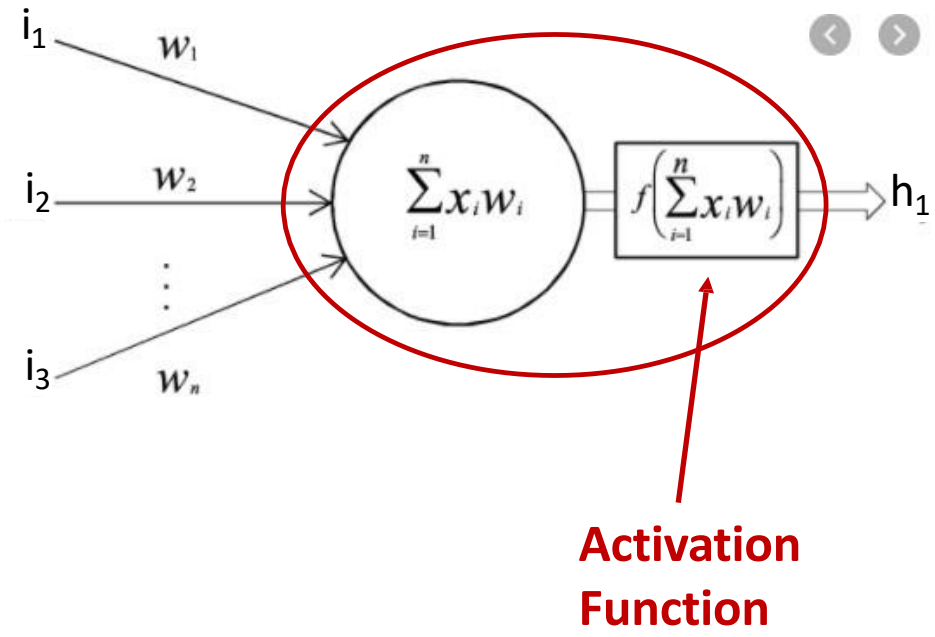
Activation Functions and Their Derivatives

https://github.com/amelcharolinesgn2/IoT_simulator-mqtt-NodeRed

https://github.com/siagianp
https://www.youtube.com/@AmelOline/videos

- Characteristics of different Activation Functions and their gradient
- The choice of activation function depend on the nature of the problem, nature of the target output and the deepness of the network.

Activation Functions



Activation Function is applied over the linear weighted summation of the incoming information to a node.

Convert linear input signals from perceptron to a linear/non-linear output signal.

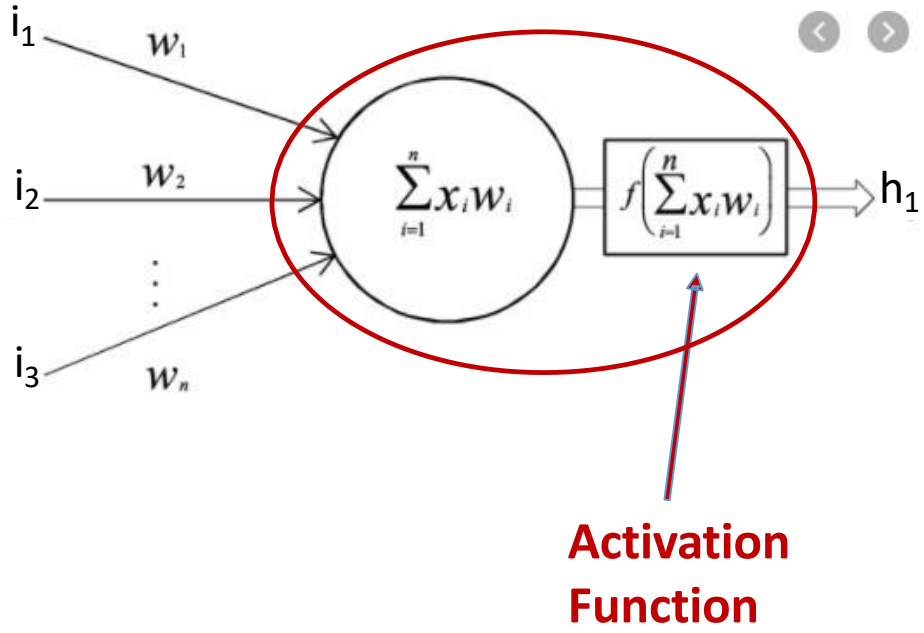
It decides whether to activate a node or not.

Activation Functions

Activation functions must be monotonic, differentiable, and quickly converging.

Types of Activation Functions:

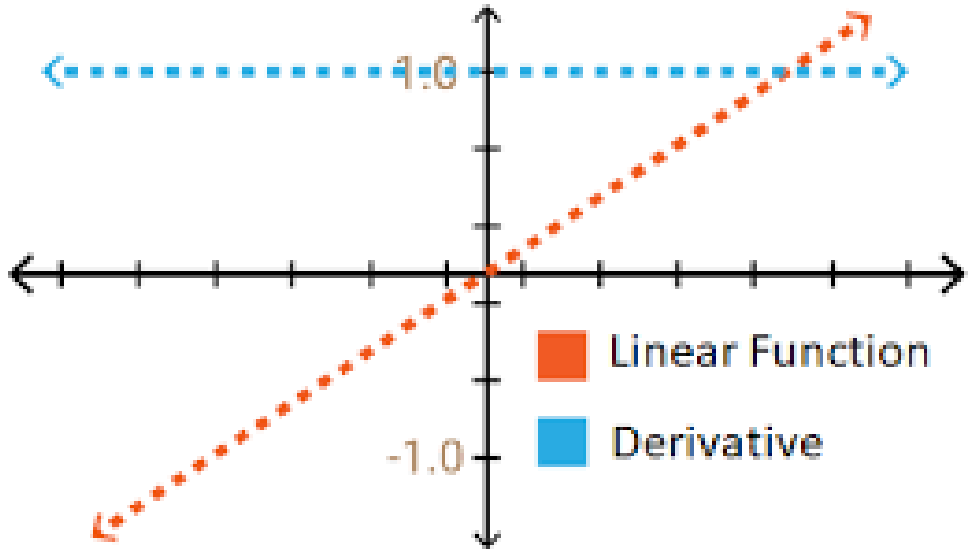
- Linear
- Non-Linear



Linear

$$f(x) = ax + b$$

$$\frac{df(x)}{dx} = a$$



Observations:

- Constant gradient
- Gradient does not depend on the change in the input

Linear

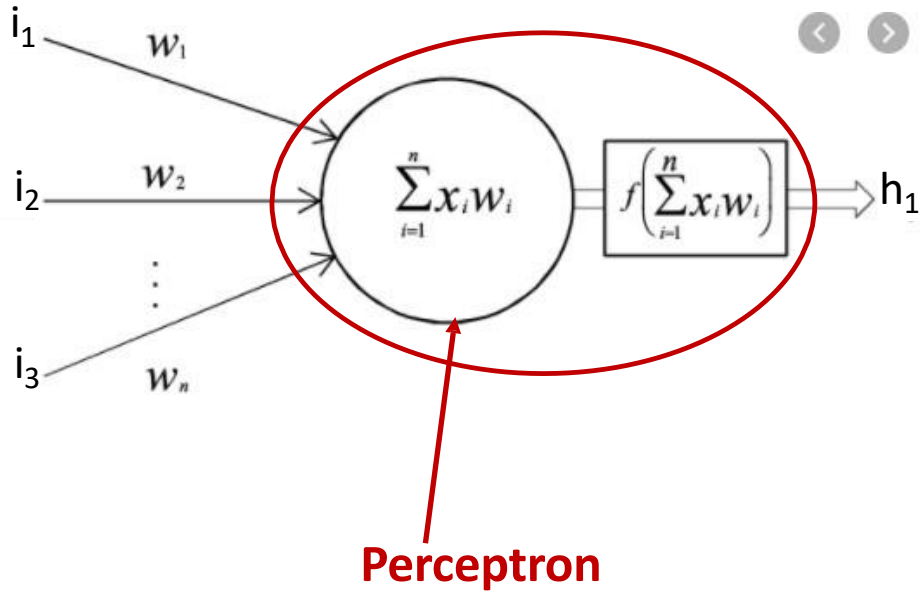
$$f(x) = ax + b$$

$$f(x) = a_1x_1 + a_2x_2 + a_3x_3 + \cdots + b$$

Linear

$$f(x) = ax + b$$

$$f(x) = a_1x_1 + a_2x_2 + a_3x_3 + \cdots + b$$



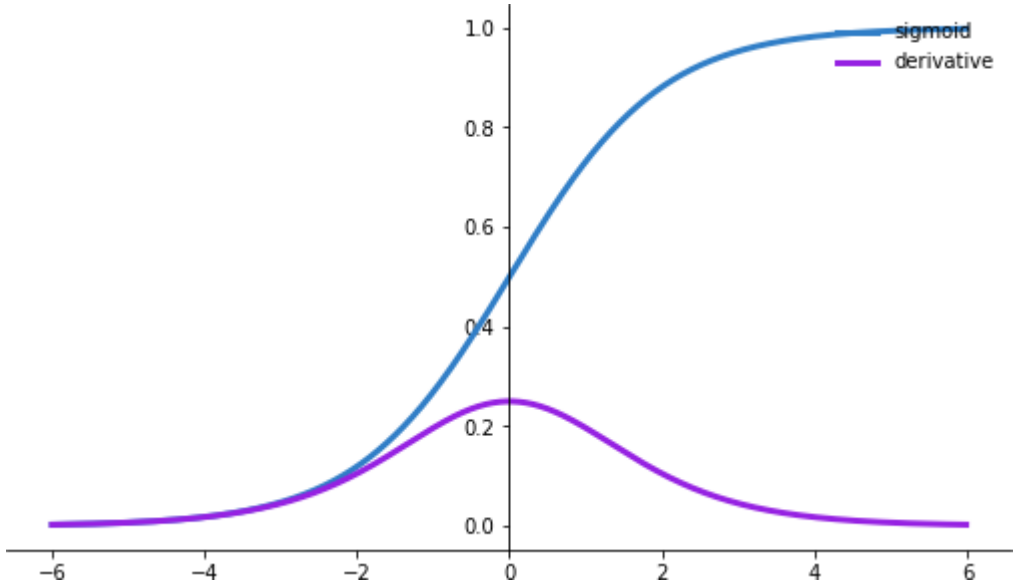
Non-Linear

- Sigmoid (Logistic)
- Hyperbolic Tangent (Tanh)
- Rectified Linear Unit (ReLU)
  - *Leaky Relu*
  - *Parametric Relu*
- Exponential Linear Unit (ELU)

Sigmoid Activation Functions (Logistics)

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

$$\frac{df(x)}{dx} = f(x)(1 - f(x))$$



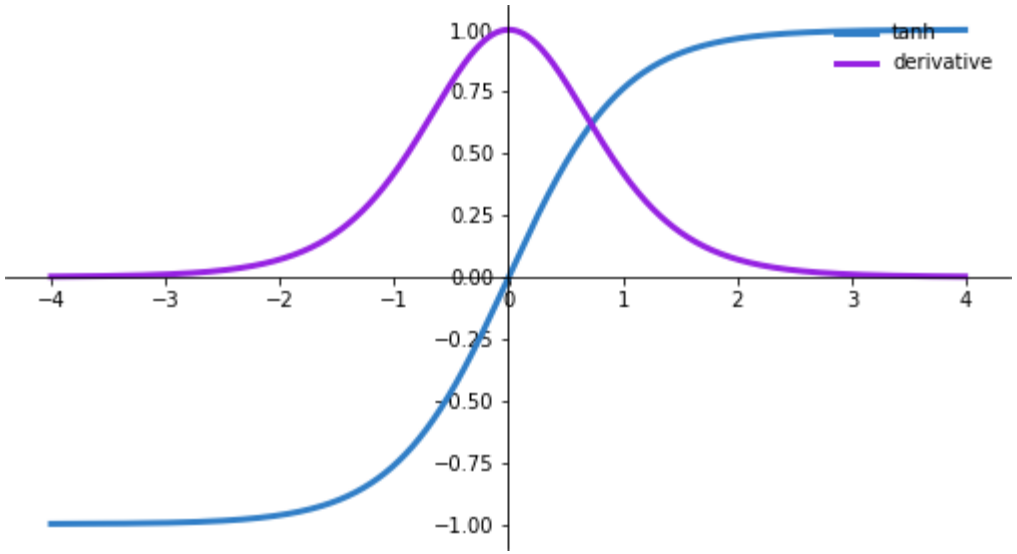
Observations:

- Output: 0 to 1
- Outputs are not zero-centered
- Can saturate and kill (vanish) gradients

Tanh Activation Function

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

$$\frac{df(x)}{dx} = 1 - f(x)^2$$



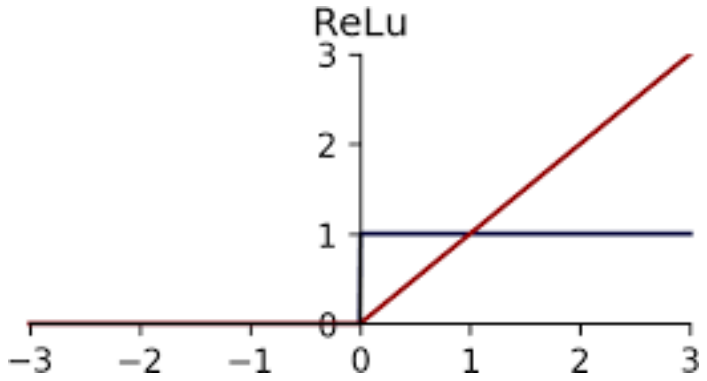
Observations:

- Output: -1 to +1
- Outputs are zero-centered
- Can Saturate and kill (vanish) gradients
- Gradient is more steeped than Sigmoid, resulting in faster convergence

Rectified Linear Unit(ReLU)

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

$$\frac{df(x)}{dx} = 1$$



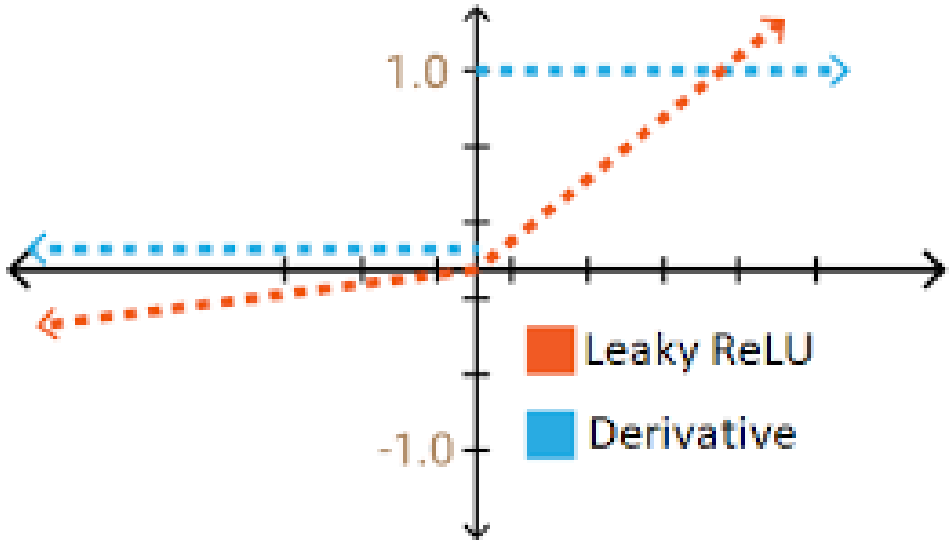
Observations:

- Greatly increase training speed compared to tanh and sigmoid
- Reduces likelihood of killing(vanishing) gradient
- It can blow up activation
- Dead nodes

Leaky-ReLU

$$f(x) = \max(0.01x, x)$$

$$\frac{df(x)}{dx} = \begin{cases} 0.01, & x < 0 \\ 1, & x \geq 0 \end{cases}$$



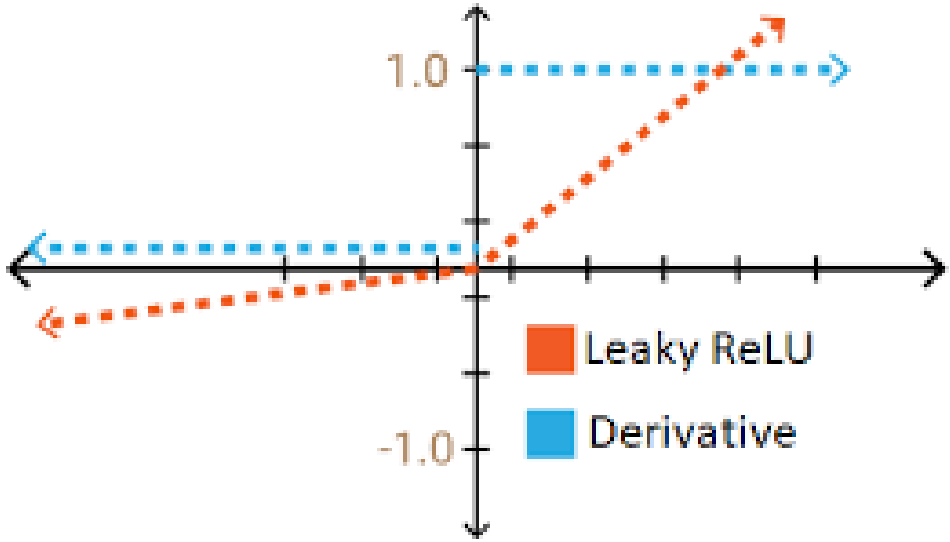
Observations:

- Fixed dying ReLU

Parameterized-ReLU

$$f(x) = \max(\alpha x, x)$$

$$\frac{df(x)}{dx} = \begin{cases} \alpha, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

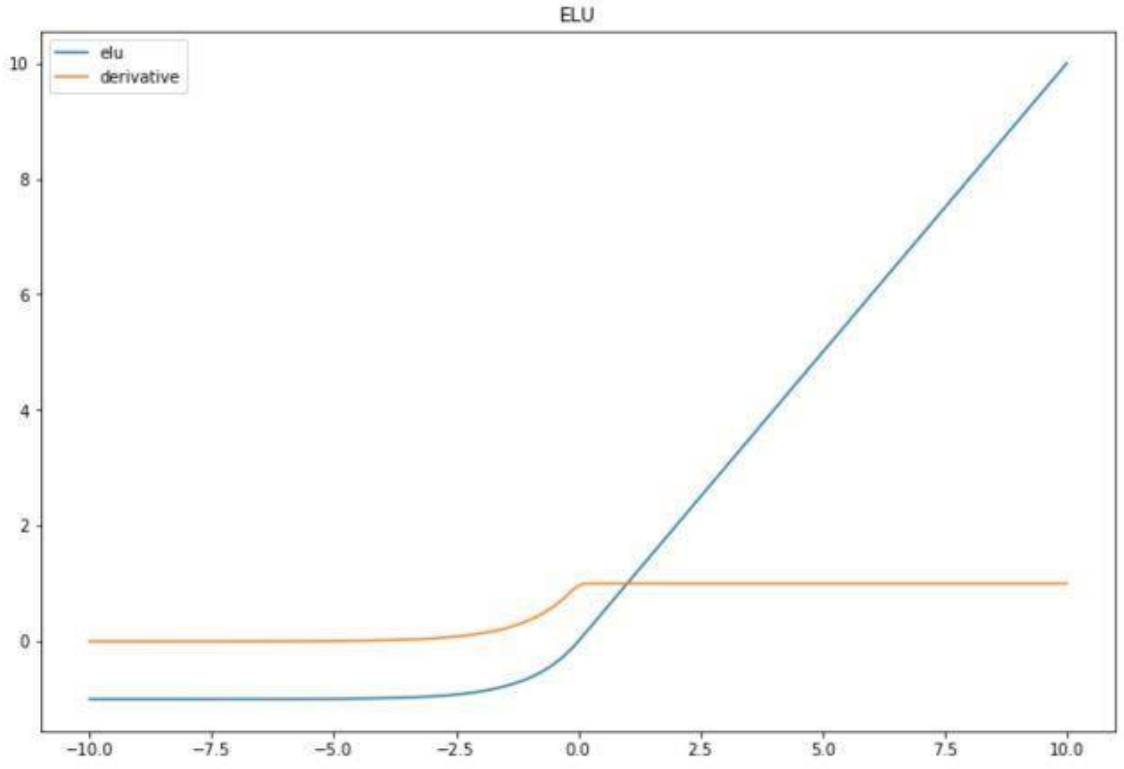


Observations:

Exponential Linear Unit (ELU)

$$f(x) = \begin{cases} \alpha(e^x - 1), & x < 0 \\ 1x & x \geq 0 \end{cases}$$

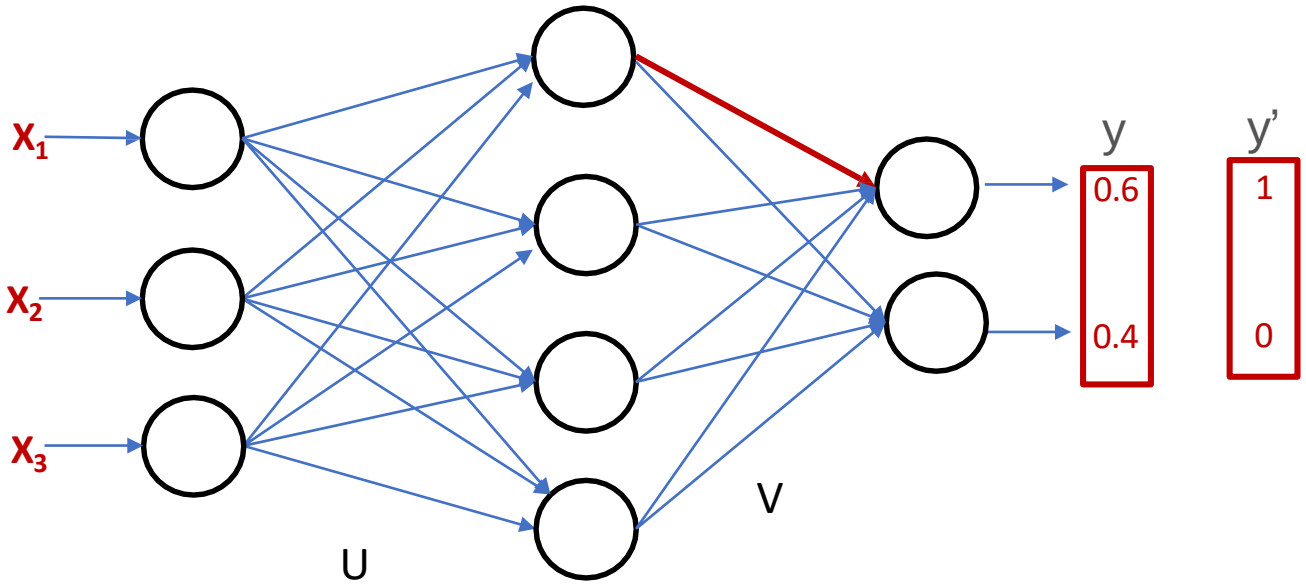
$$\frac{df(x)}{dx} = \begin{cases} f(x) + \alpha, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

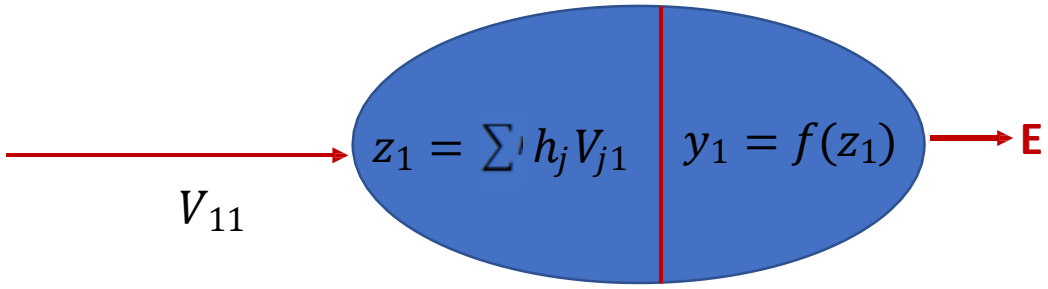


Observations:

- **It can produce –ve output**
- **It can blow up activation function**

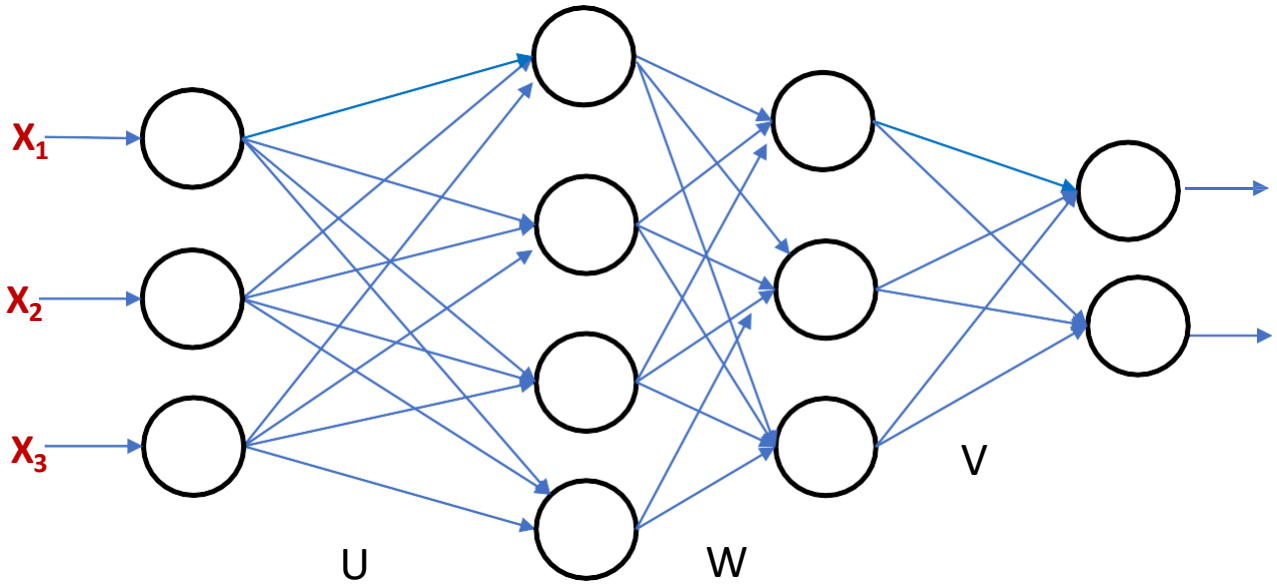
Complete Chain



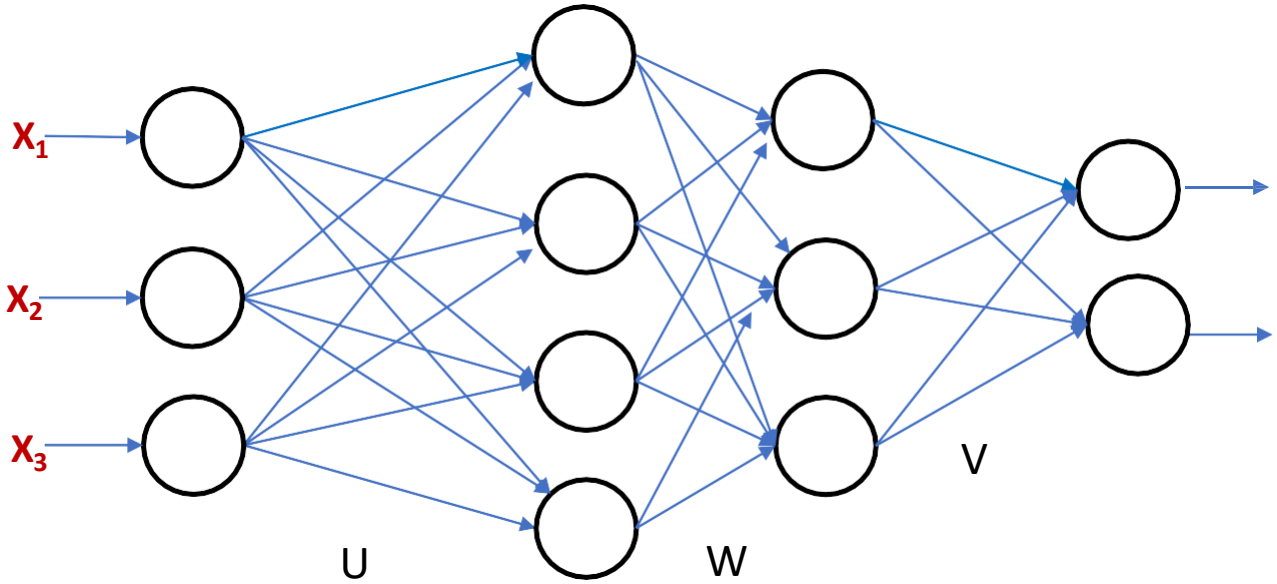


$$\frac{\delta E}{\delta V_{11}} = \frac{\delta z_1}{\delta V_{11}} \times \frac{\delta y_1}{\delta z_1} \times \frac{\delta E}{\delta y_1}$$

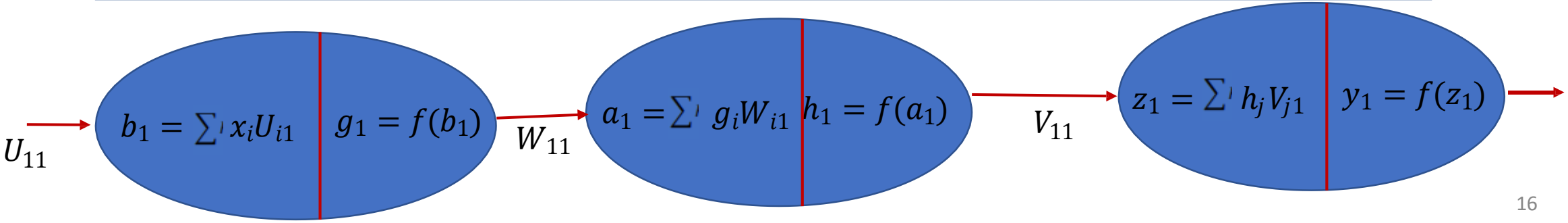
Deep Network

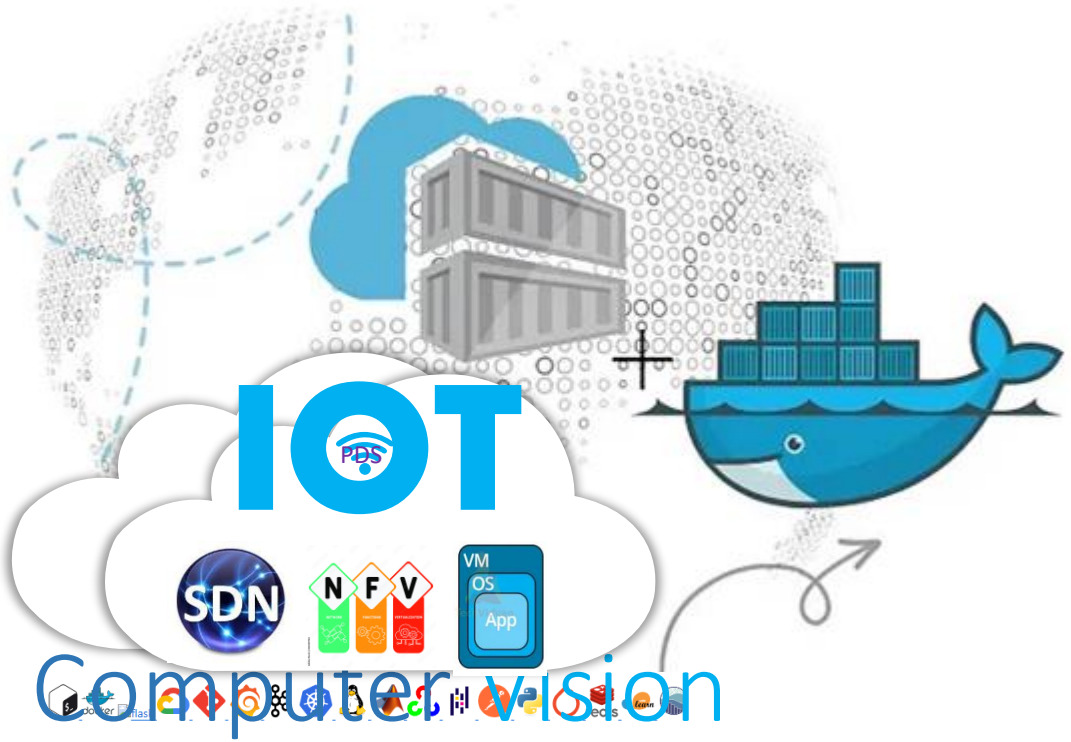


Deep Network - Vanishing/Exploding Gradient



$$\frac{\delta E}{\delta U_{11}} = \frac{\delta b_1}{\delta U_{11}} \times \frac{\delta g_1}{\delta b_1} \times \frac{\delta a_1}{\delta g_1} \times \frac{\delta h_1}{\delta a_1} \times \frac{\delta z_1}{\delta h_1} \times \frac{\delta y_1}{\delta z_1} \times \frac{\delta E}{\delta y_1}$$





Thank you!

Thank you For being a great class!

https://github.com/siagianp

https://github.com/amelcharolinesgn2/IoT_simulator-mqtt-NodeRed

https://www.youtube.com/@AmelOline/videos