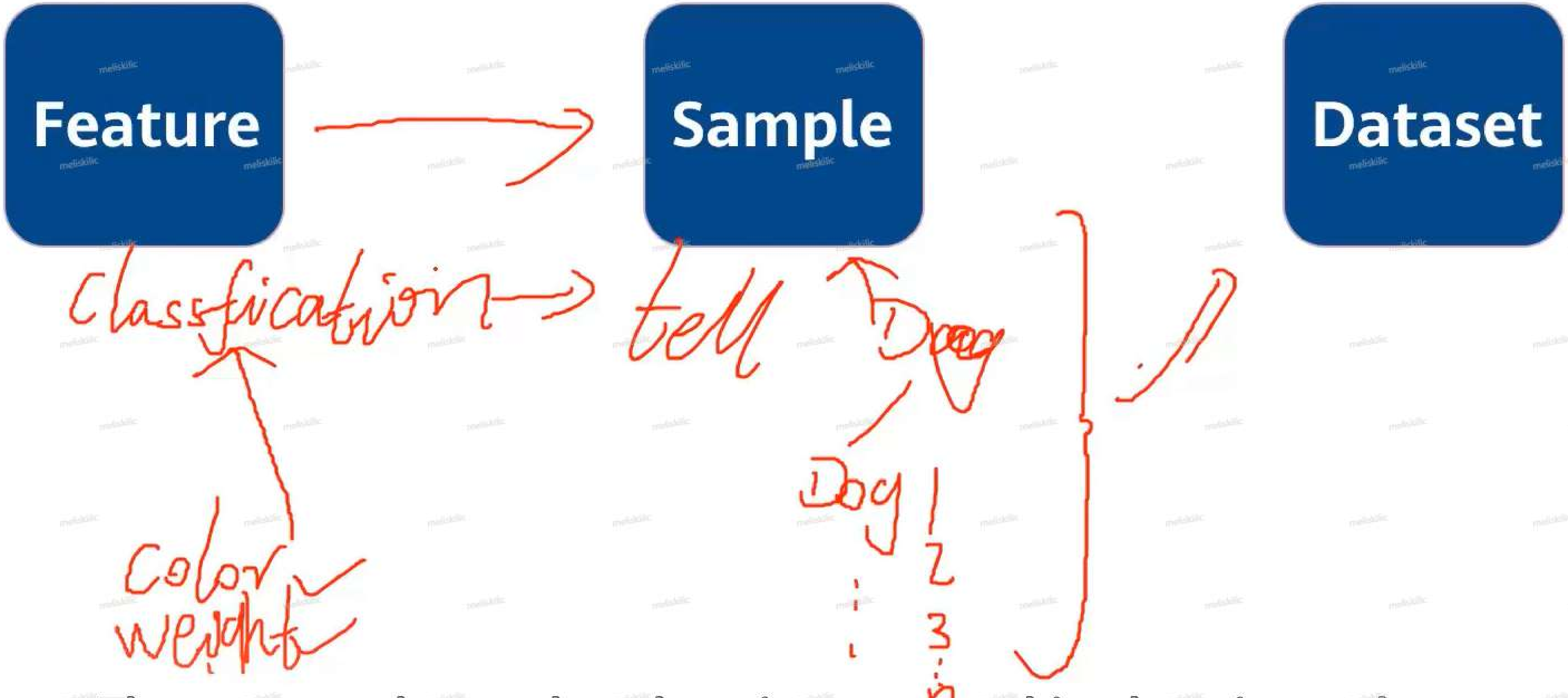
Dataset



Feature

Sample

Dataset

There we can do our deep learning or machine learning tasks

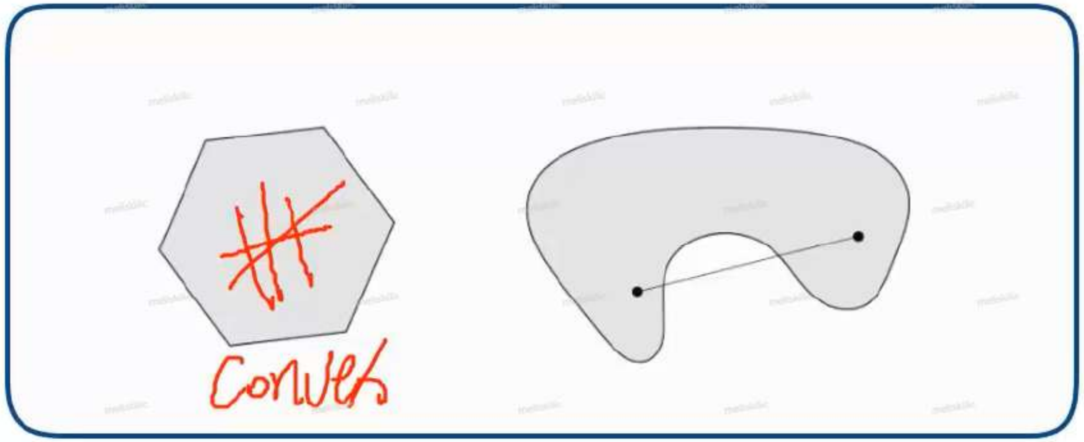
Learning Method Classification (2)

Semi-supervised learning

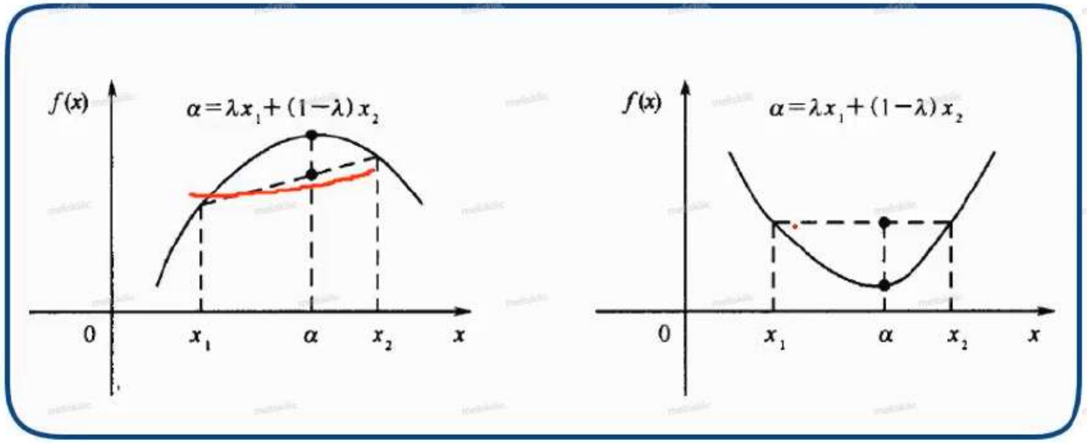
Reinforcement learning

rewards and punishment

Concave and Convex Functions



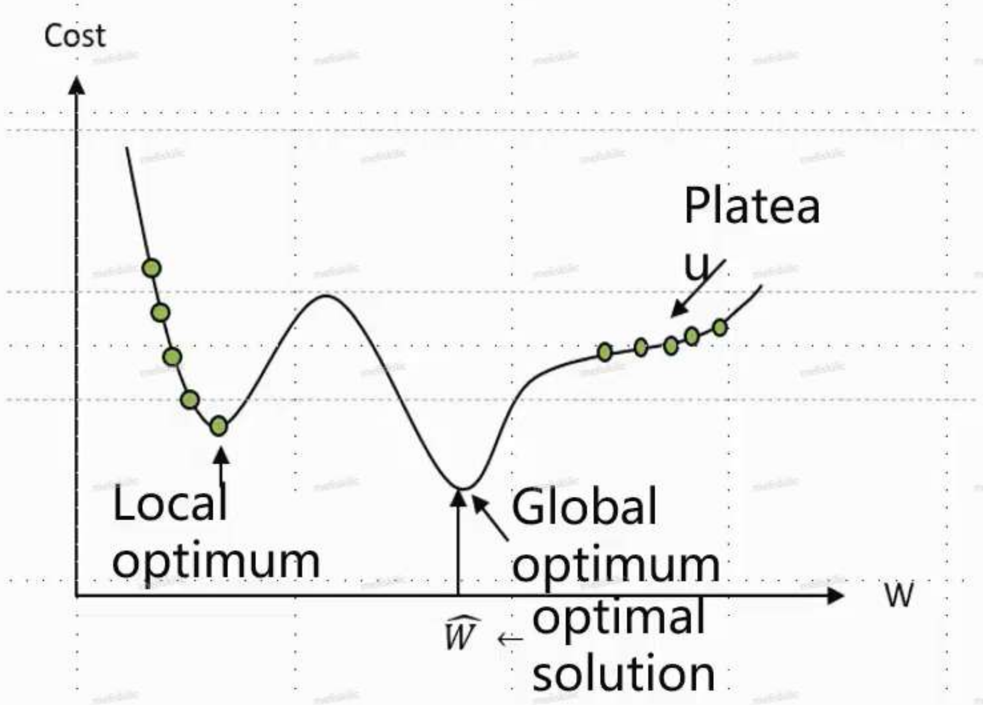
Convex set & Concave set



Concave function&Convex function

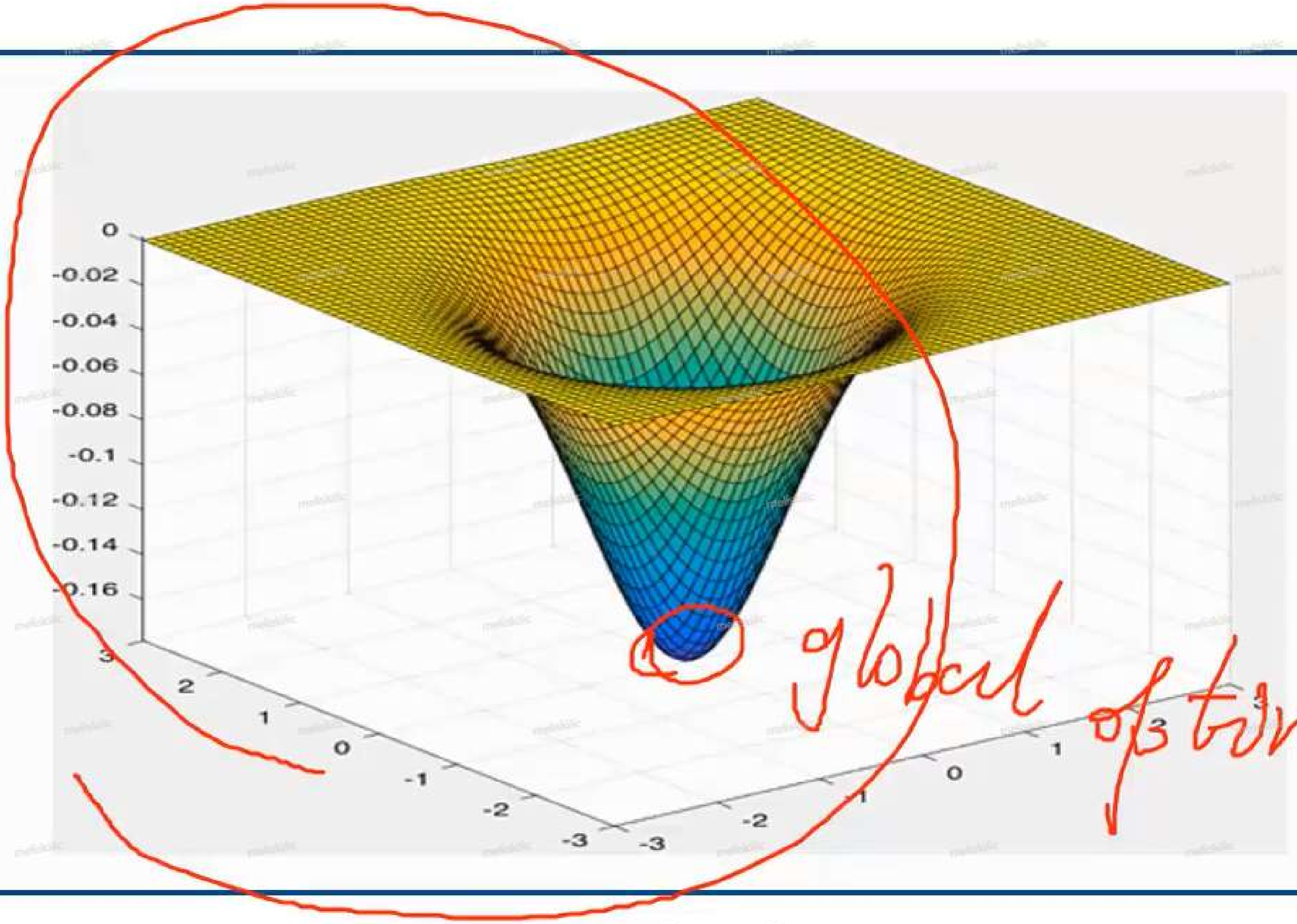
Non-Convex Function

prediction <=> real result



high-order functions

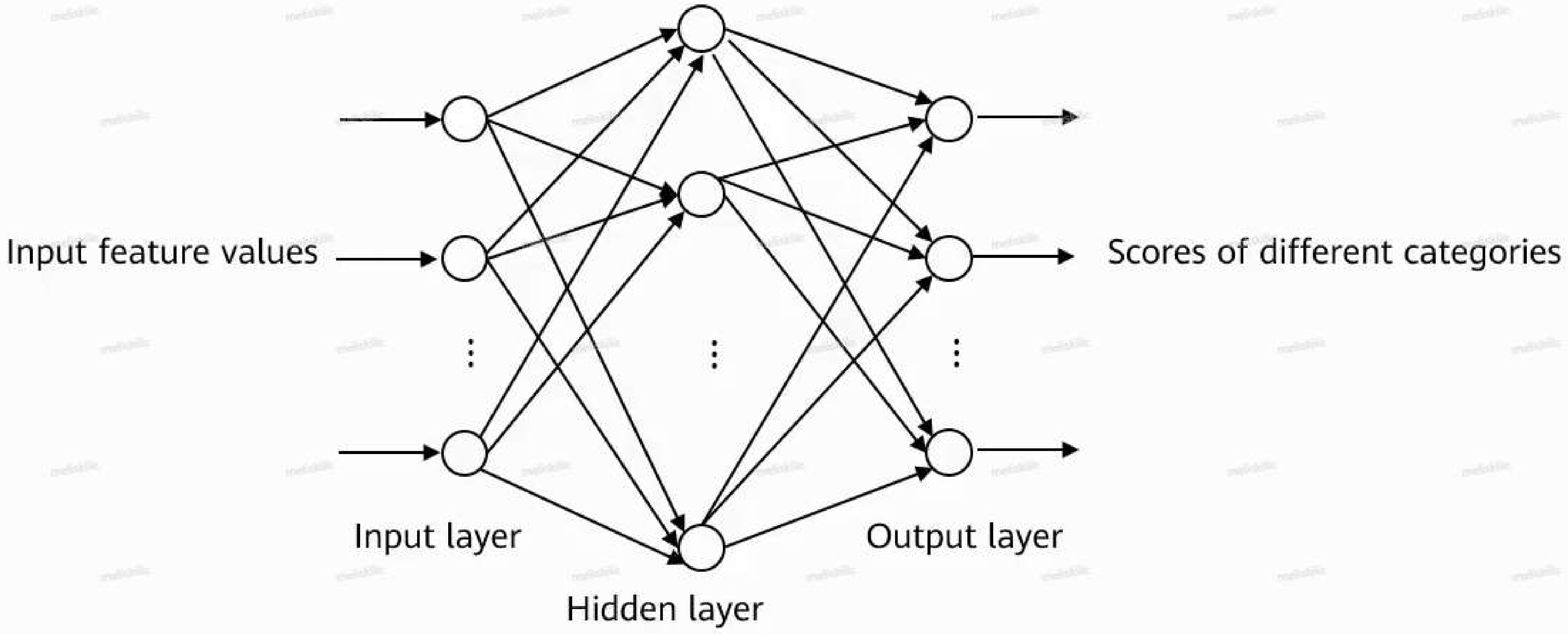
Convex Function



global optimal point

convex function

Loss Function



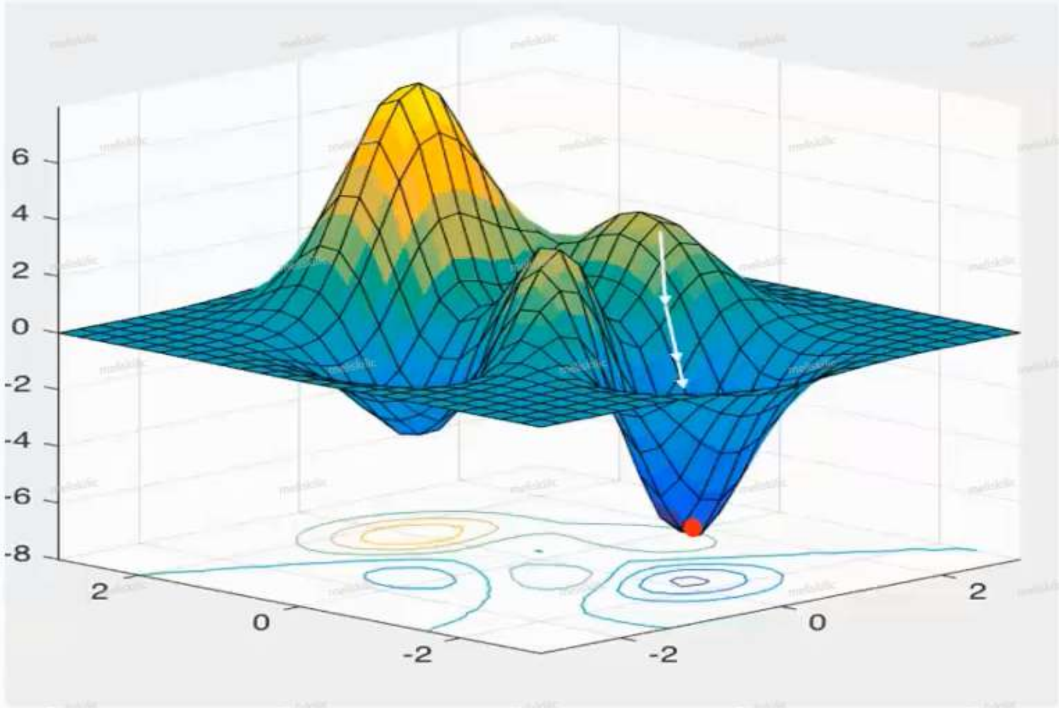
Cross entropy/Softmax Loss Function

$$Loss_i = -\sum_k p_k \log(q_k) = -\sum_k p_k \log\left(\frac{e^{f_k}}{\sum_j e^{f_j}}\right)$$

In the preceding function, p_k is the probability that x_i belongs to class k:

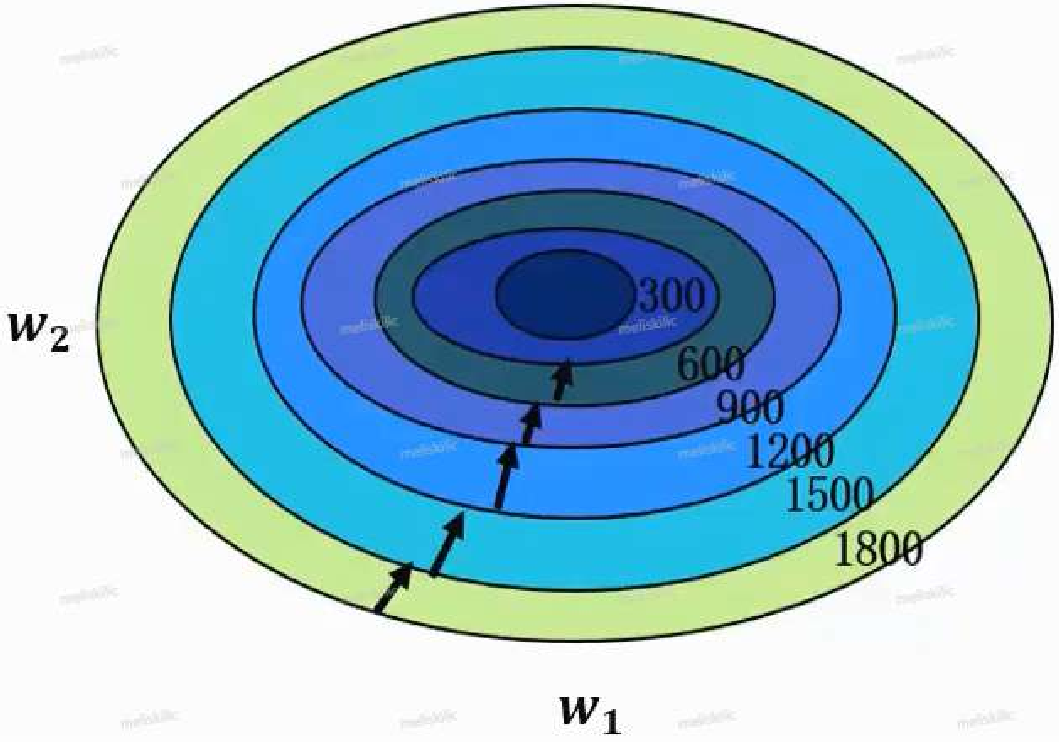
$$p_{k=y_i} = 1,\ p_{k\neq y_i} = 0.$$

Gradient Descent (1)



Gradient Descent (2)

$$w^{+} = w - \underline{\eta} * \boxed{\frac{\partial Loss}{\partial w}}$$



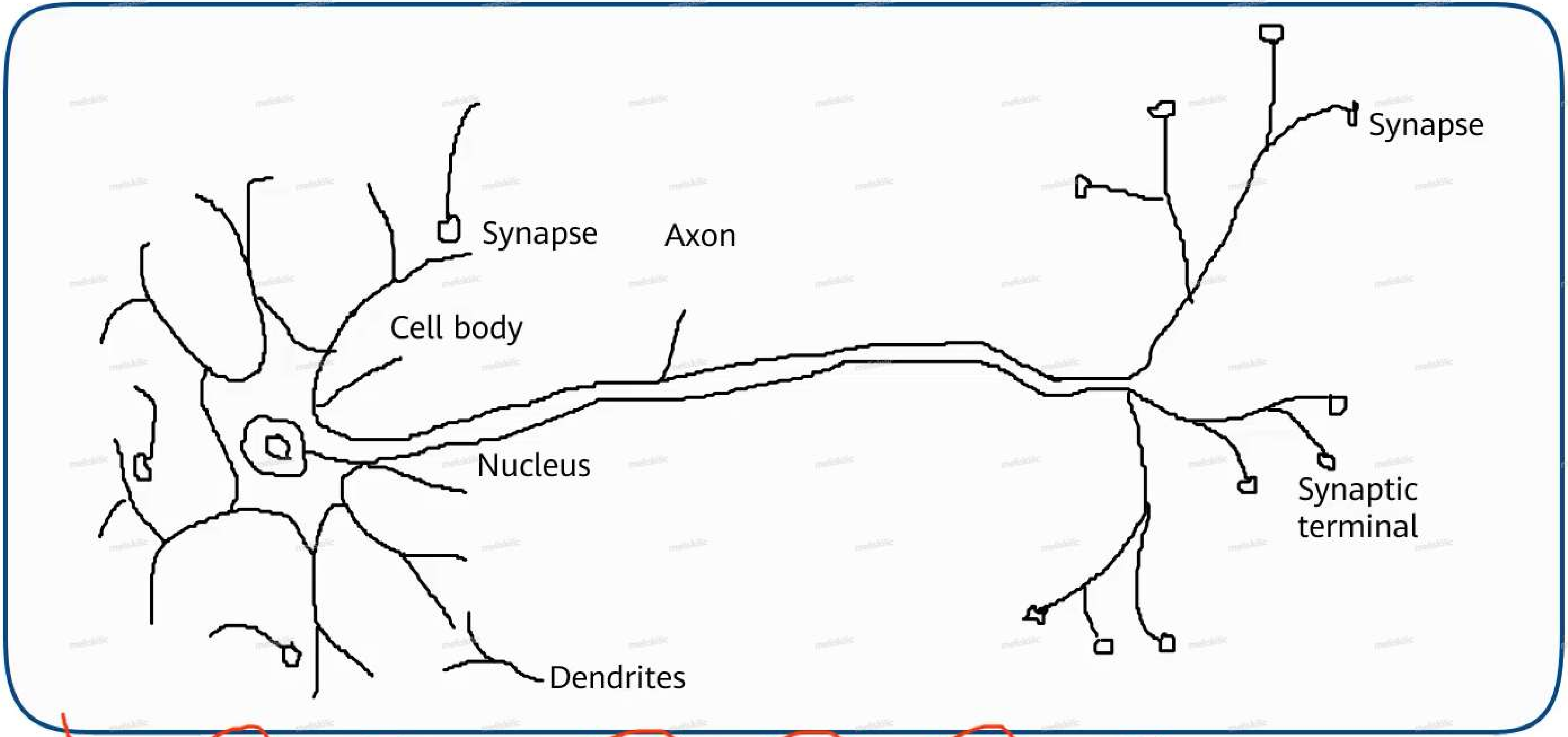
Gradient Descent (3)

Batch gradient descent (BGD)

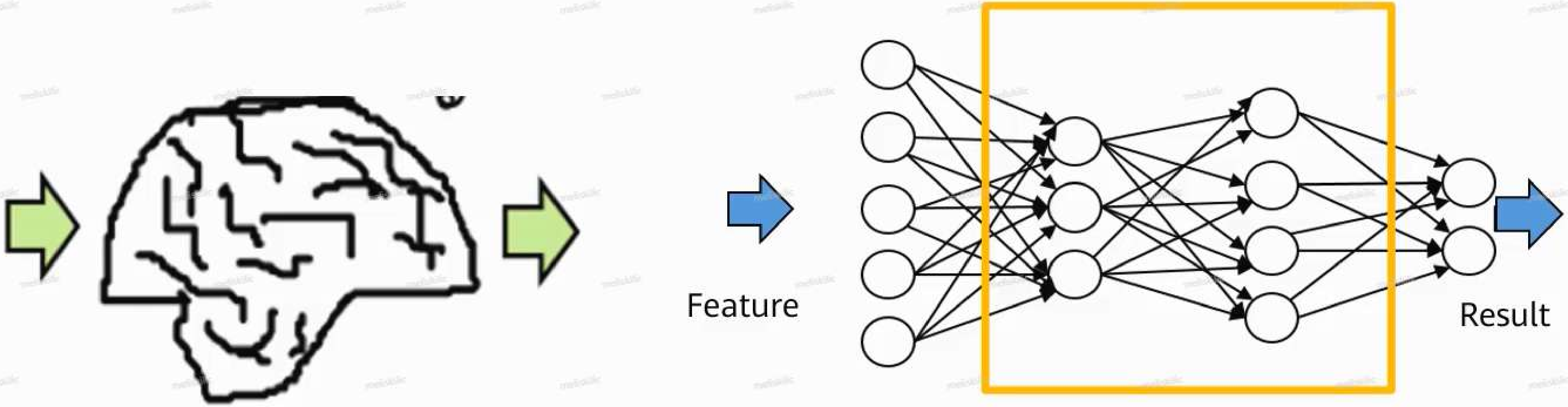
Stochastic gradient descent (SGD)

Mini-batch gradient descent (MBGD)

Biological Neural Networks

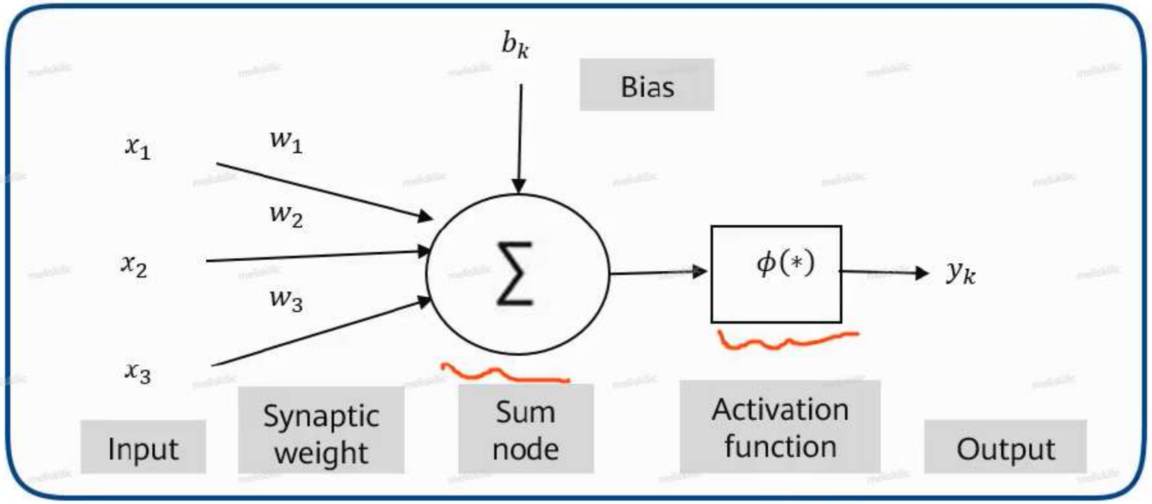


Artificial Neural Networks



Dentrites nuclei axone

Neuron (1)



Linear function: If the input is the n-dimensional feature vector X, the calculation formula is as follows:

$$f(X, W, b) = WX + b = \sum_{n} w_i x_i + b = [\mathrm{W}; \mathrm{b}][\mathrm{X}; 1]^{\mathrm{T}}$$

Activation function: $O = sign(net) = \begin{cases} 1, net > 0, \\ -1, otherwise. \end{cases}$

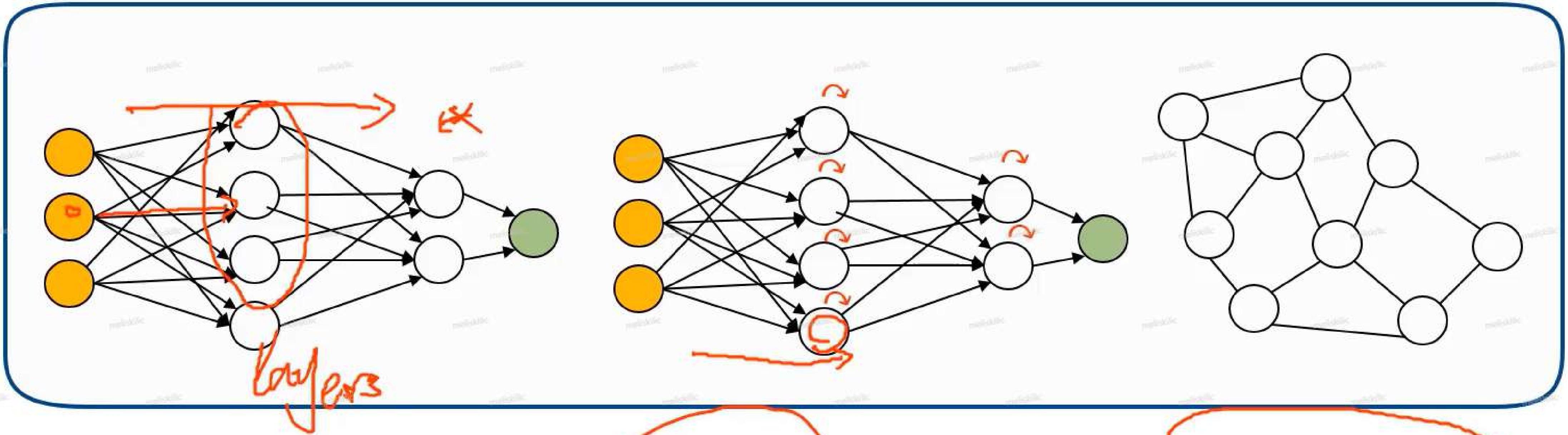
Neuron (2)

Neural Network Topology

Activation Rule

Learning Algorithm

Neural Network Topologies



Feedforward neural network

Feedback neural network

Graph neural network

2009

Perceptron — Algorithm Formula

1957

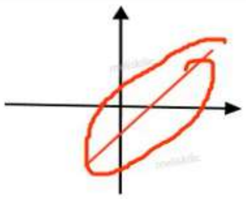
Input Value

$$Z = W_1X_1 + W_2X_2 + \ldots\ldots + W_nX_n = \sum_{n_i=1} W_iX_i = WX$$

Activation function: $sign(\boldsymbol{Z}) = \begin{cases} 1 & Z \ge \theta \\ -1 & Z \le \theta' \end{cases}$, where θ is the threshold.

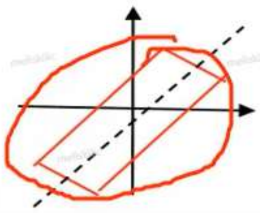
Separating point hyperplane

$Ax+B=0$



Separating line

$Ax+By+C=0$

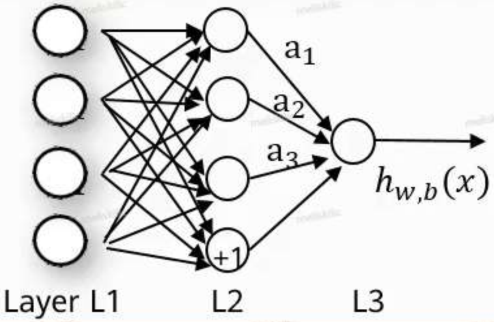


Separating plane

$Ax+By+Cz+D=0$

Separating

$WX+b=0$



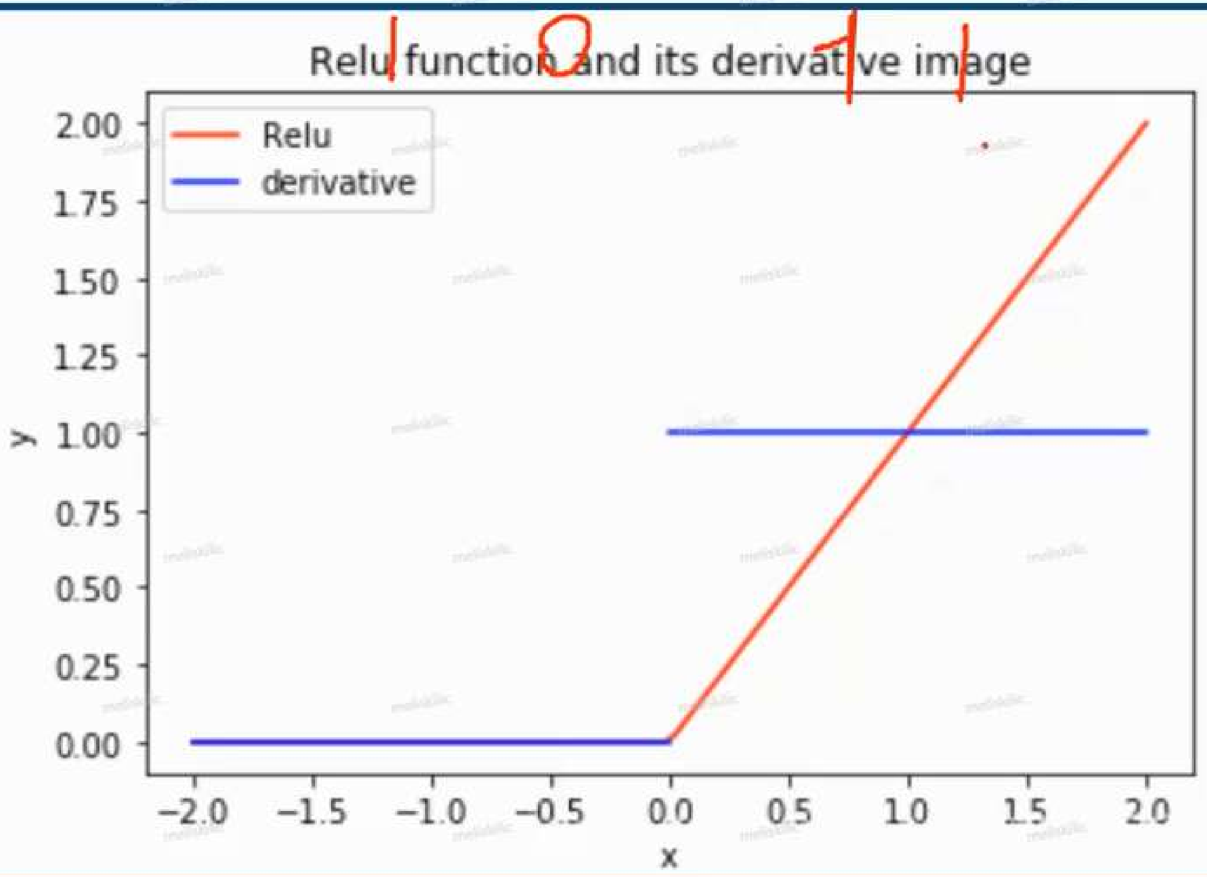
Perceptron

Perceptron — Loss Function

$$L(w, b) = -\sum_{X_i \varepsilon M} y_i(w * x_0 + b)$$

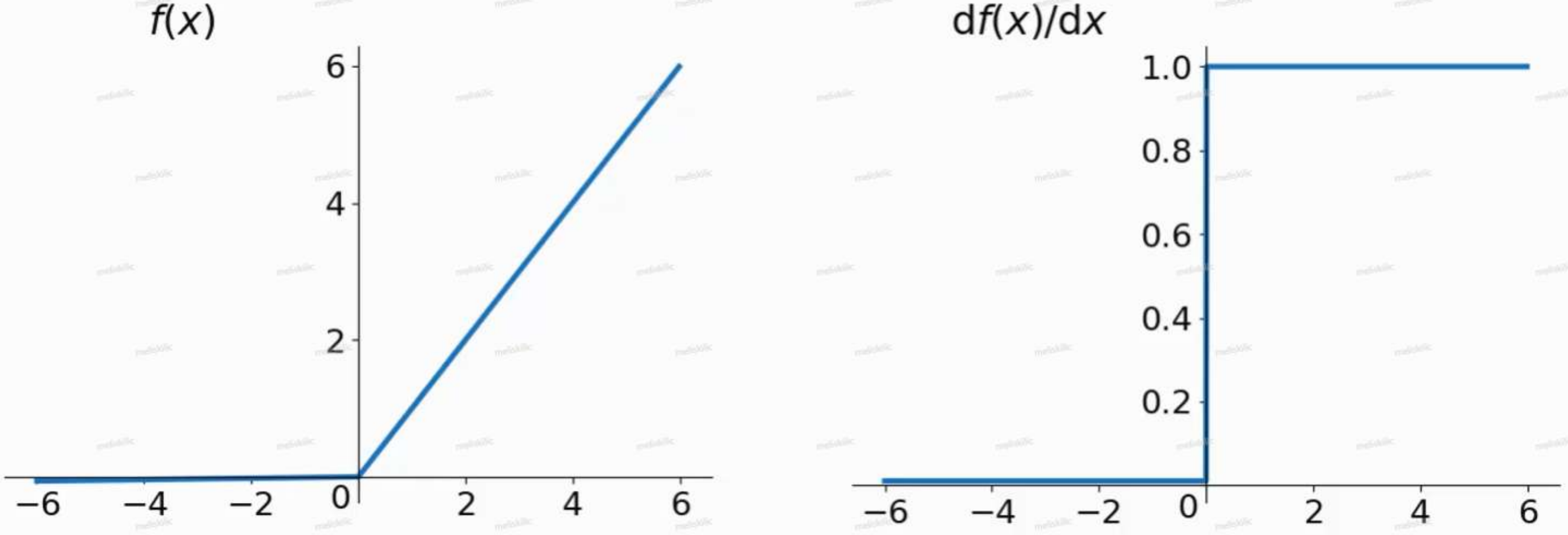
ReLU Function

$$ReLU(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} = \max(0, x)$$



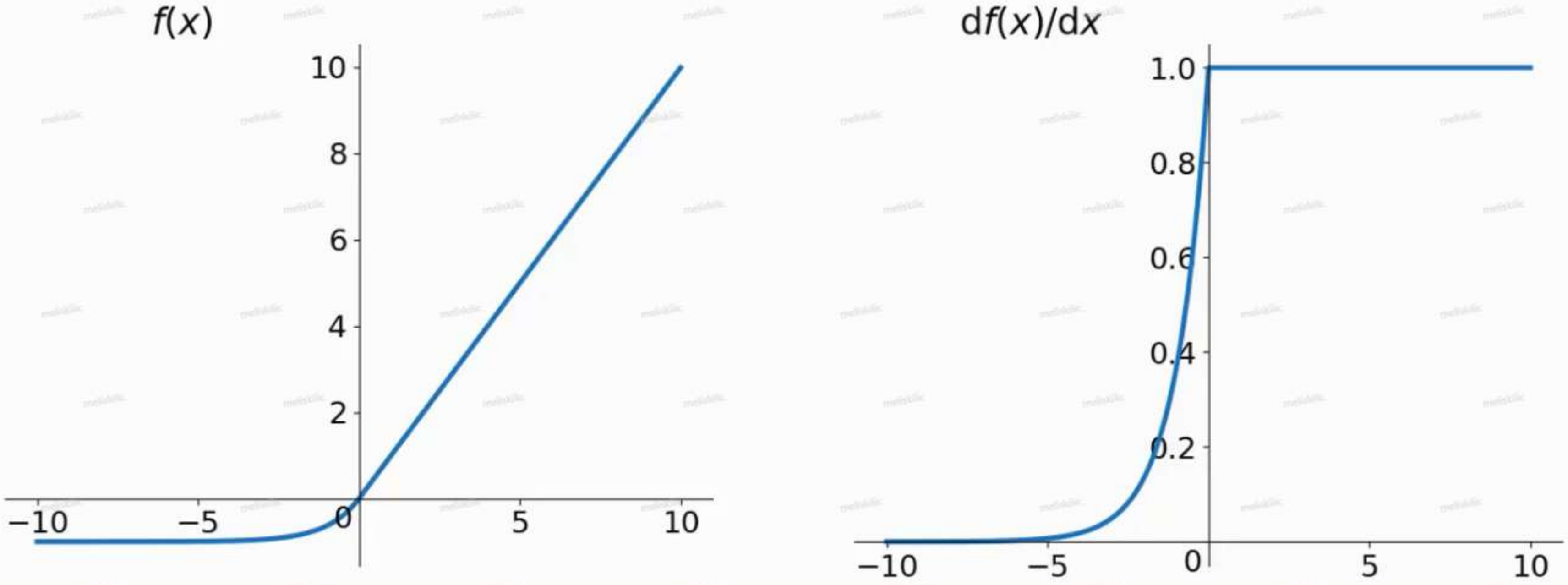
Leaky ReLU Function

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



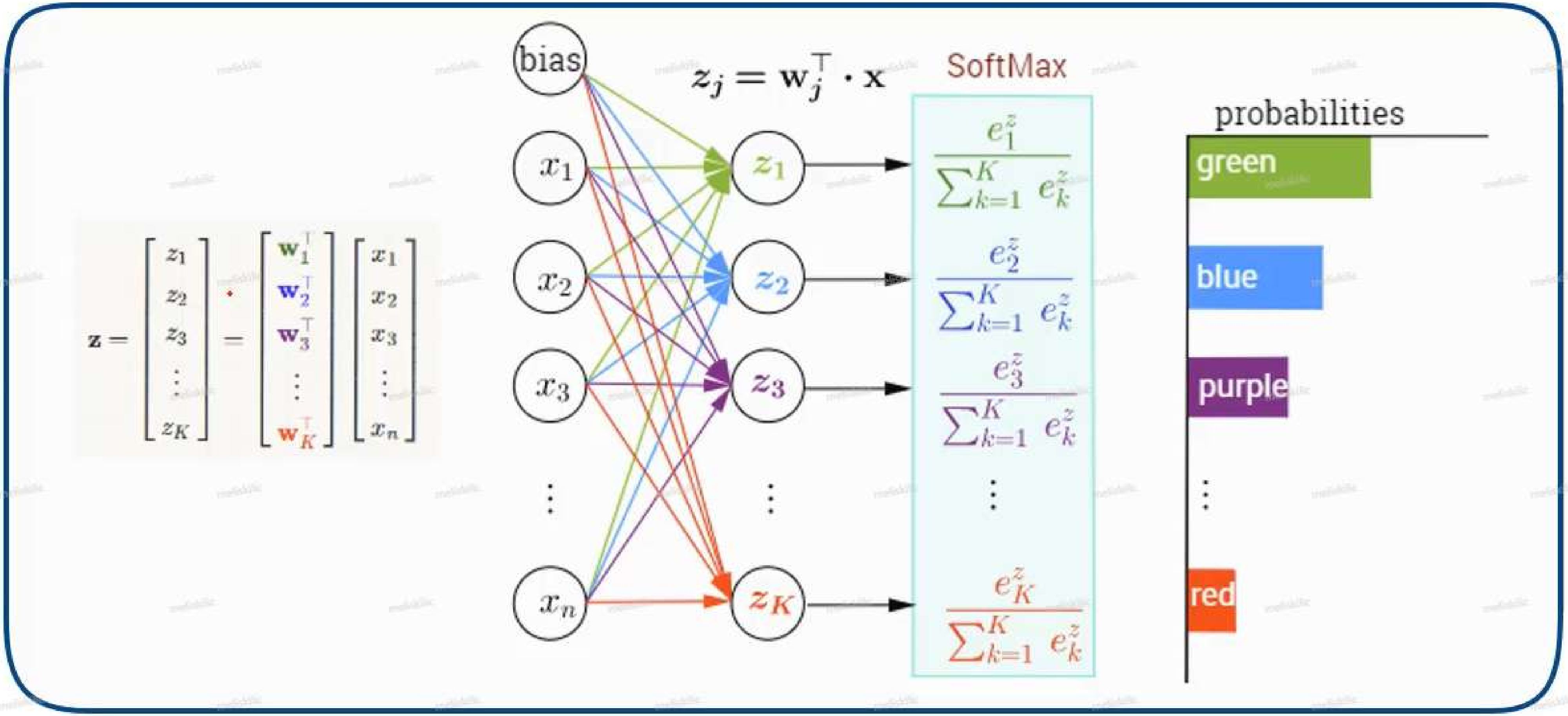
ELU (Exponential Linear Units) Function

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases}$$



Softmax Function

$$\sigma(\mathrm{z})_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$$


$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_K \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \mathbf{w}_3^\top \\ \vdots \\ \mathbf{w}_K^\top \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$
bias
x_1
x_2
x_3
$\vdots$
x_n
$z_j = \mathbf{w}_j^\top \cdot \mathbf{x}$
z_1
z_2
z_3
$\vdots$
z_K
SoftMax
$\frac{e_1^z}{\sum_{k=1}^K e_k^z}$
$\frac{e_2^z}{\sum_{k=1}^K e_k^z}$
$\frac{e_3^z}{\sum_{k=1}^K e_k^z}$
$\vdots$
$\frac{e_K^z}{\sum_{k=1}^K e_k^z}$
probabilities
green
blue
purple
$\vdots$
red

Factors

Non-linear

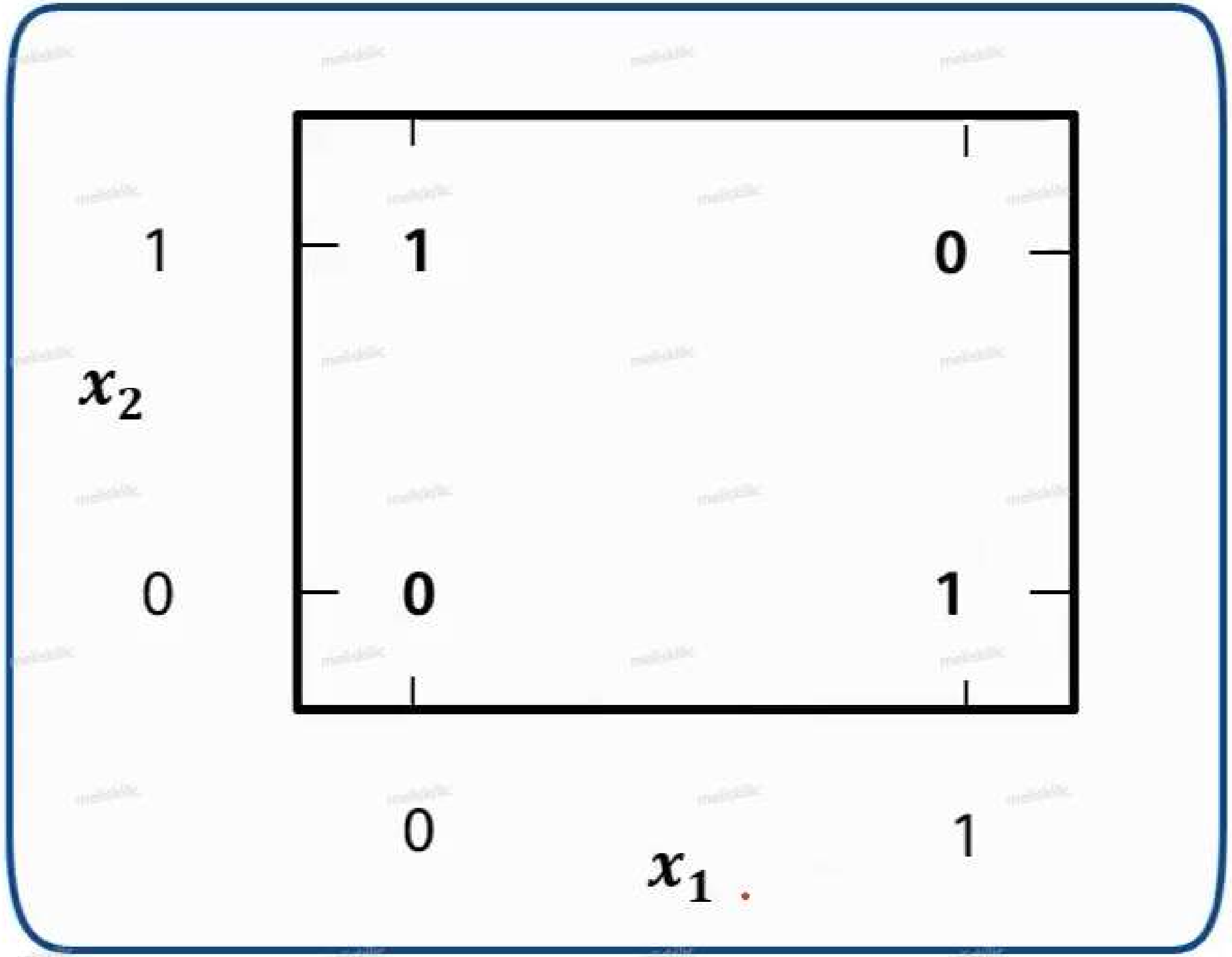
Continuously differentiable

Range

Monotonic

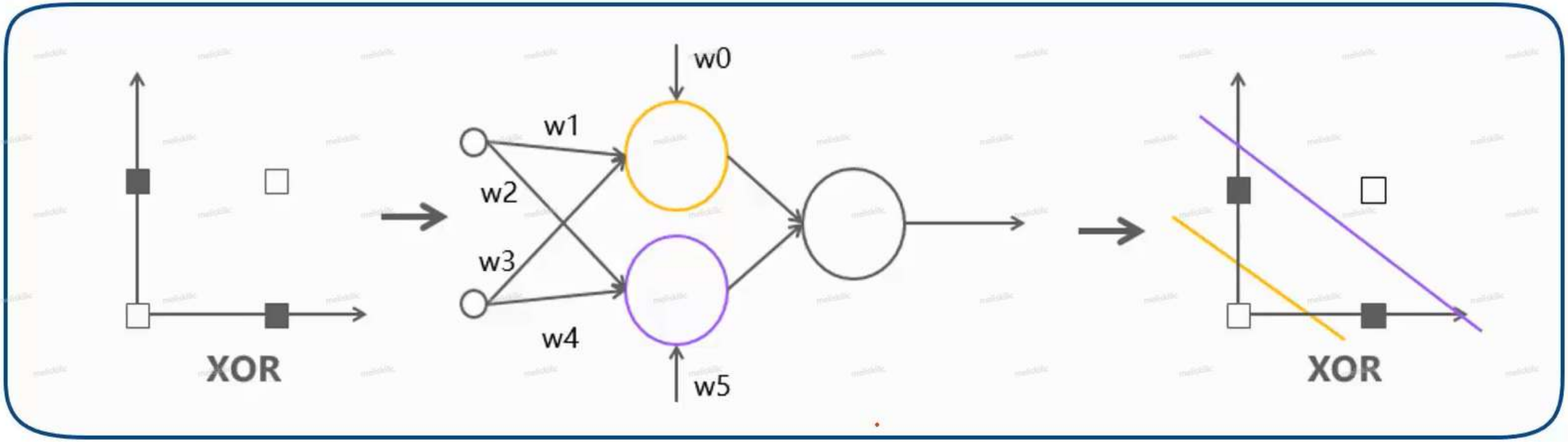
Smooth

XOR Problem

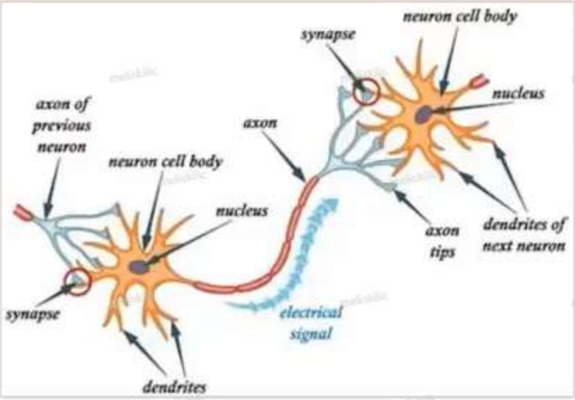


Oranginal **x** Space

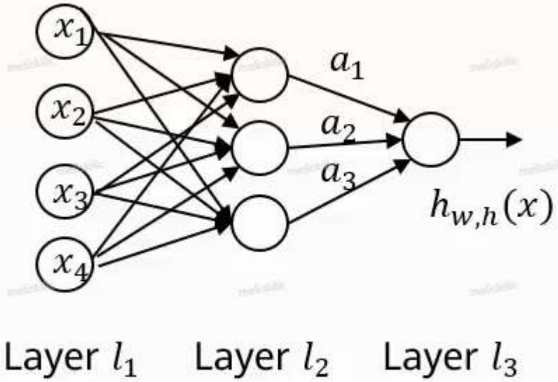
Multi-layer Fully-Connected ANN



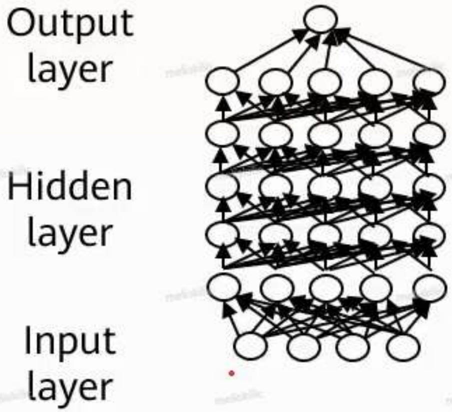
Deep Learning



Human neural network

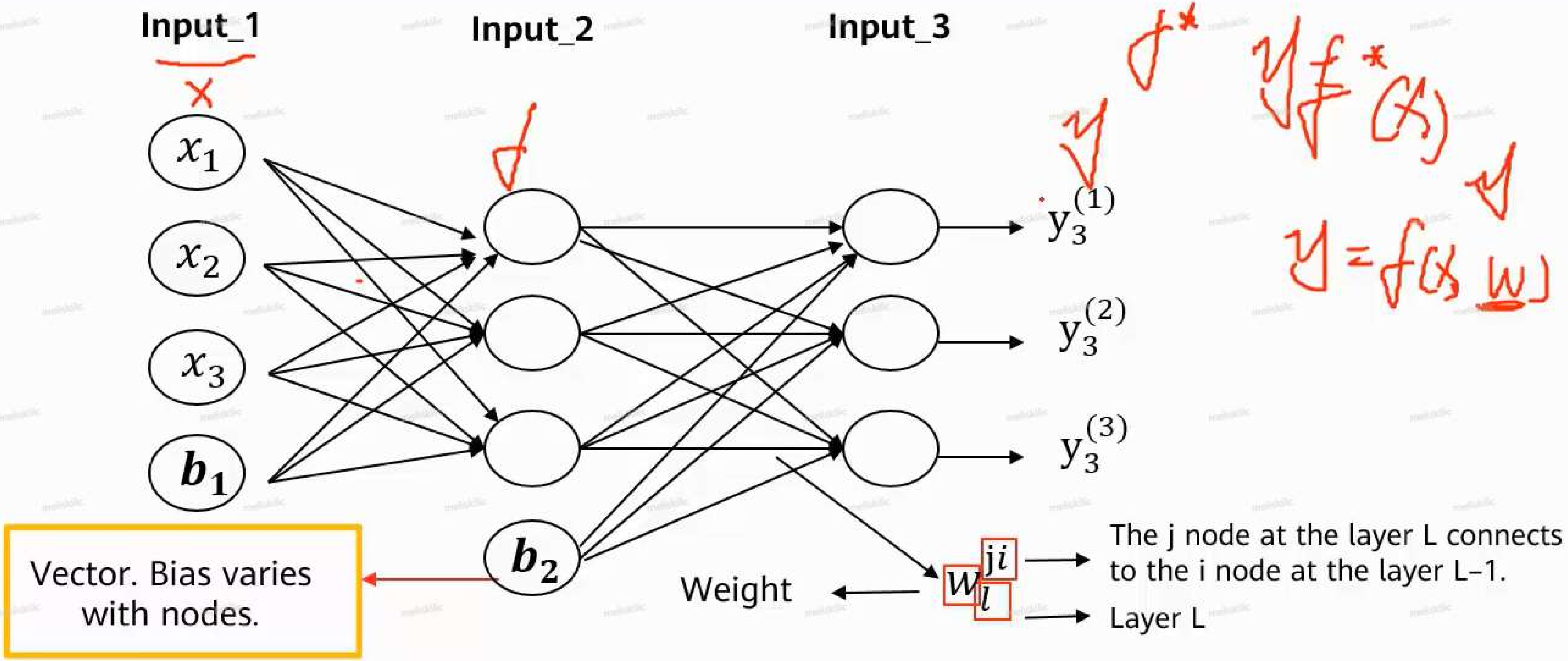


Perceptron



Deep feedforward network

Deep Feedforward Network



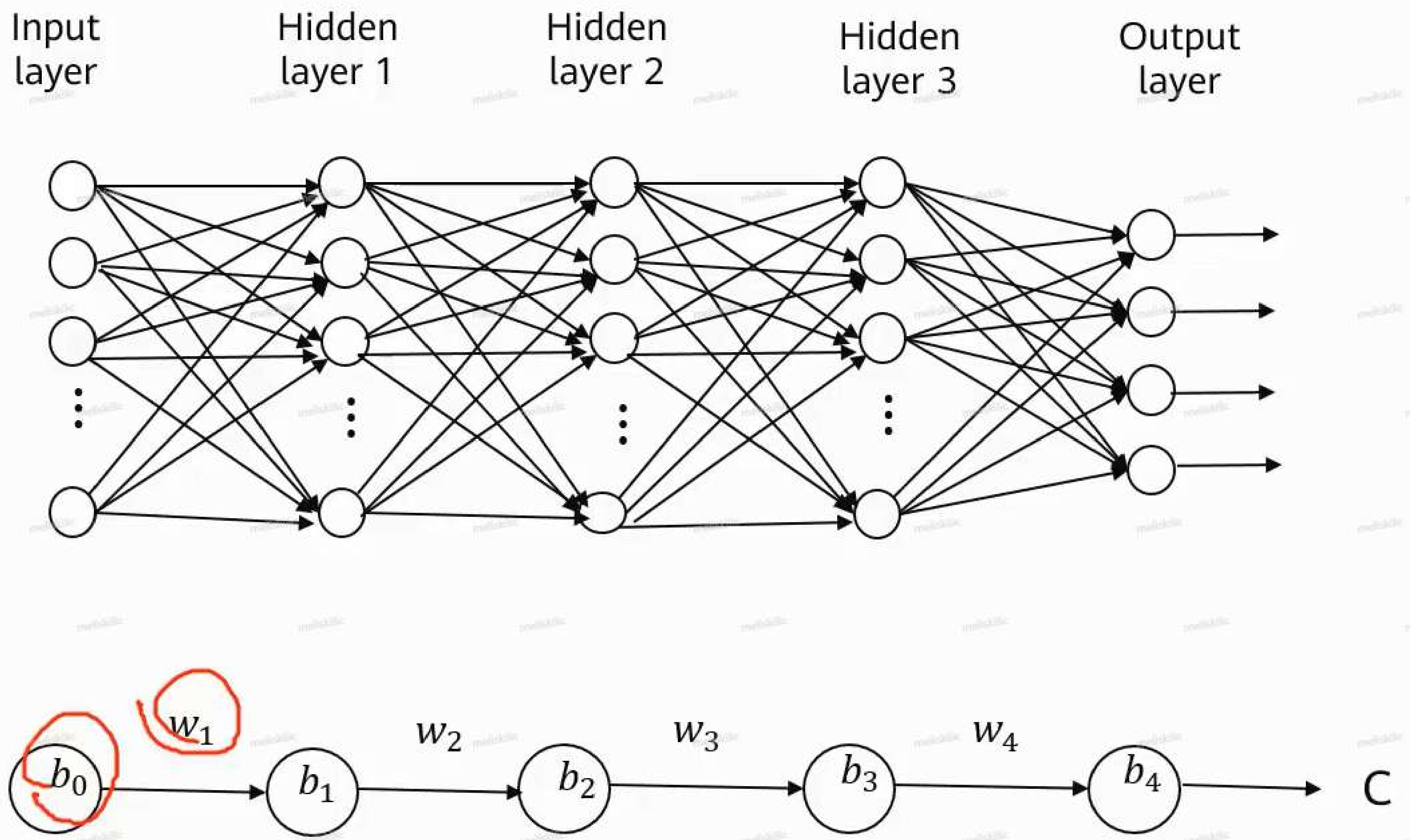
Vector. Bias varies with nodes.

Weight

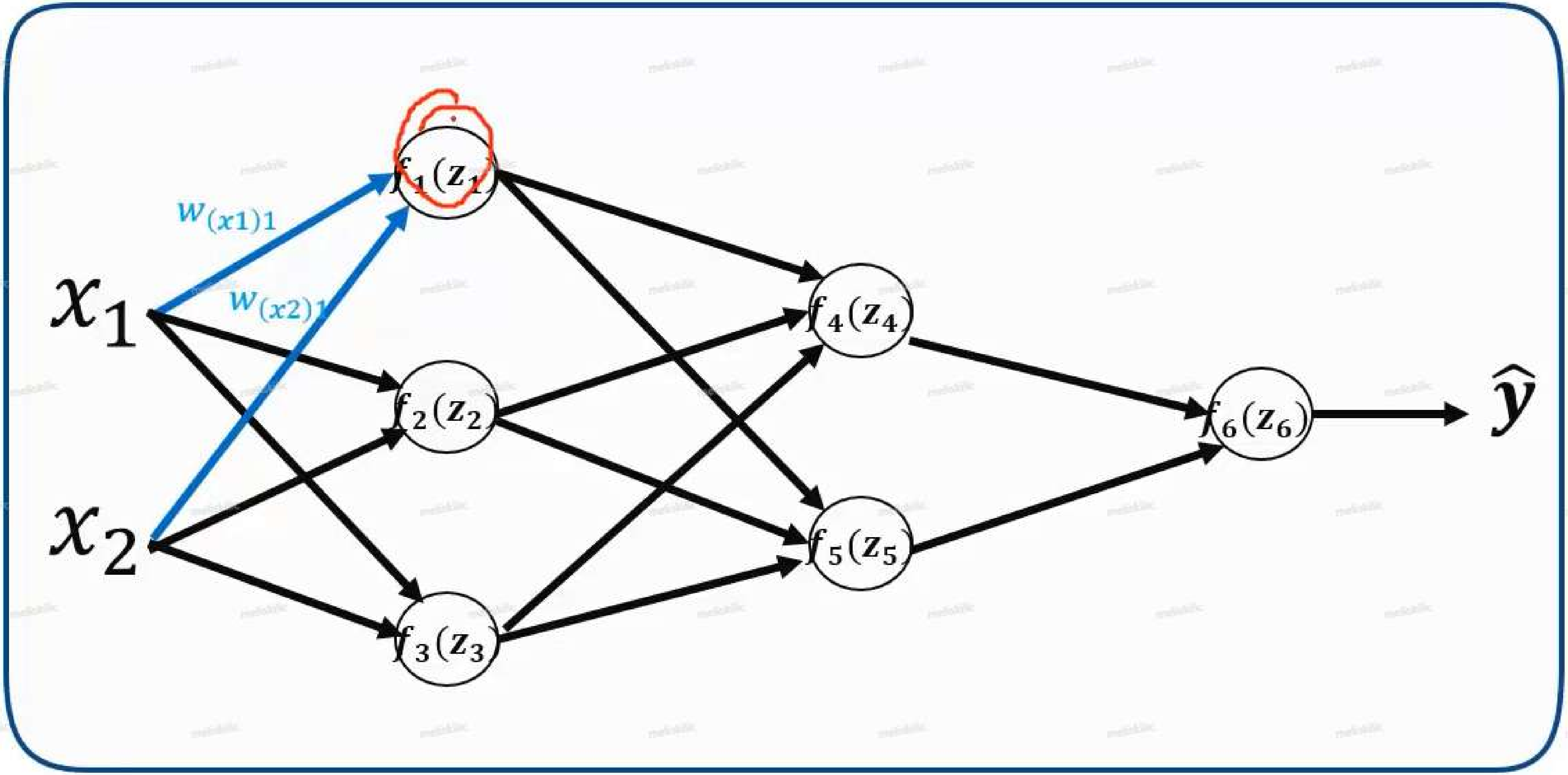
The j node at the layer L connects to the i node at the layer L–1.

Layer L

Back propagation

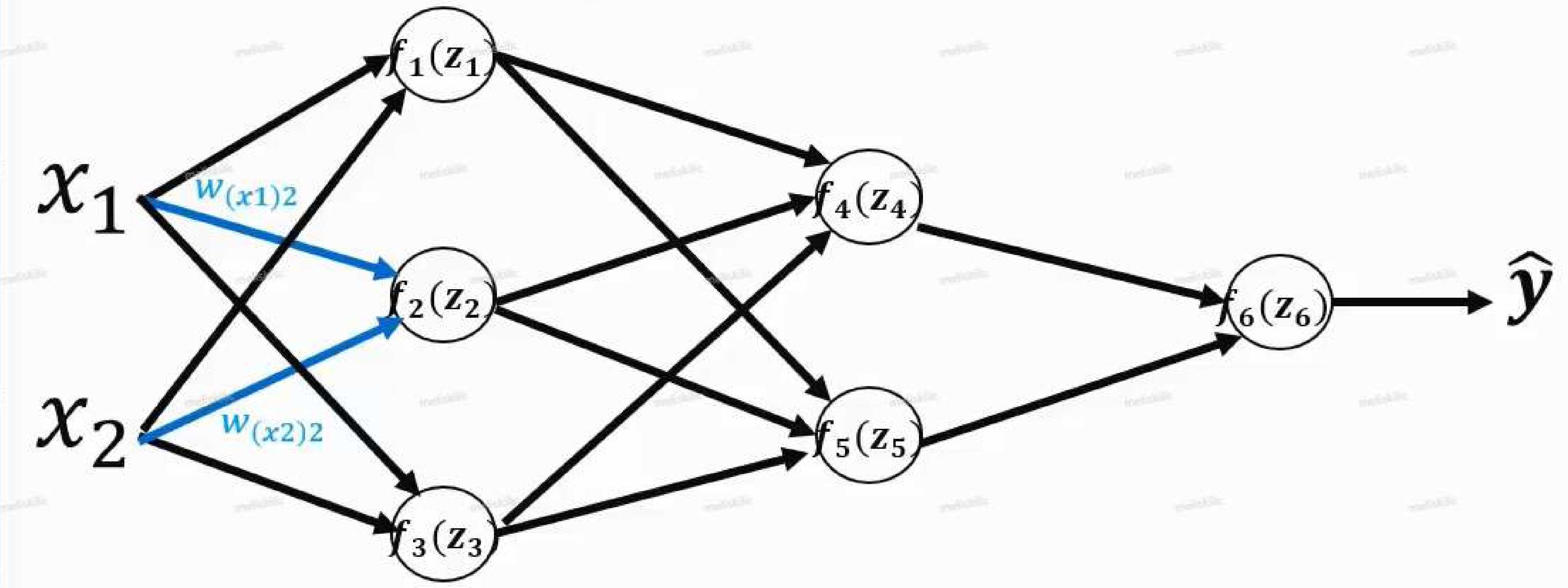


Forward Propagation (1)



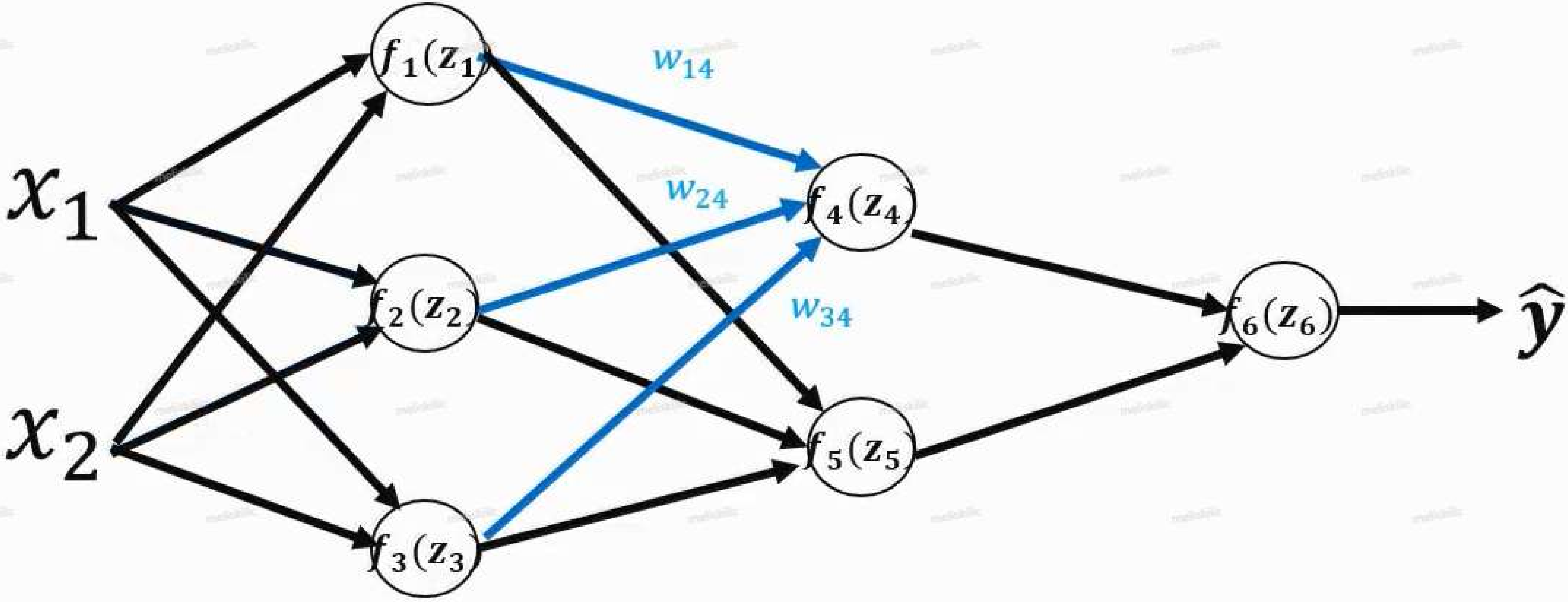
Forward Propagation (2)

$$y_2 = f_2(w_{(x1)2}x_1 + w_{(x2)2}x_2)$$



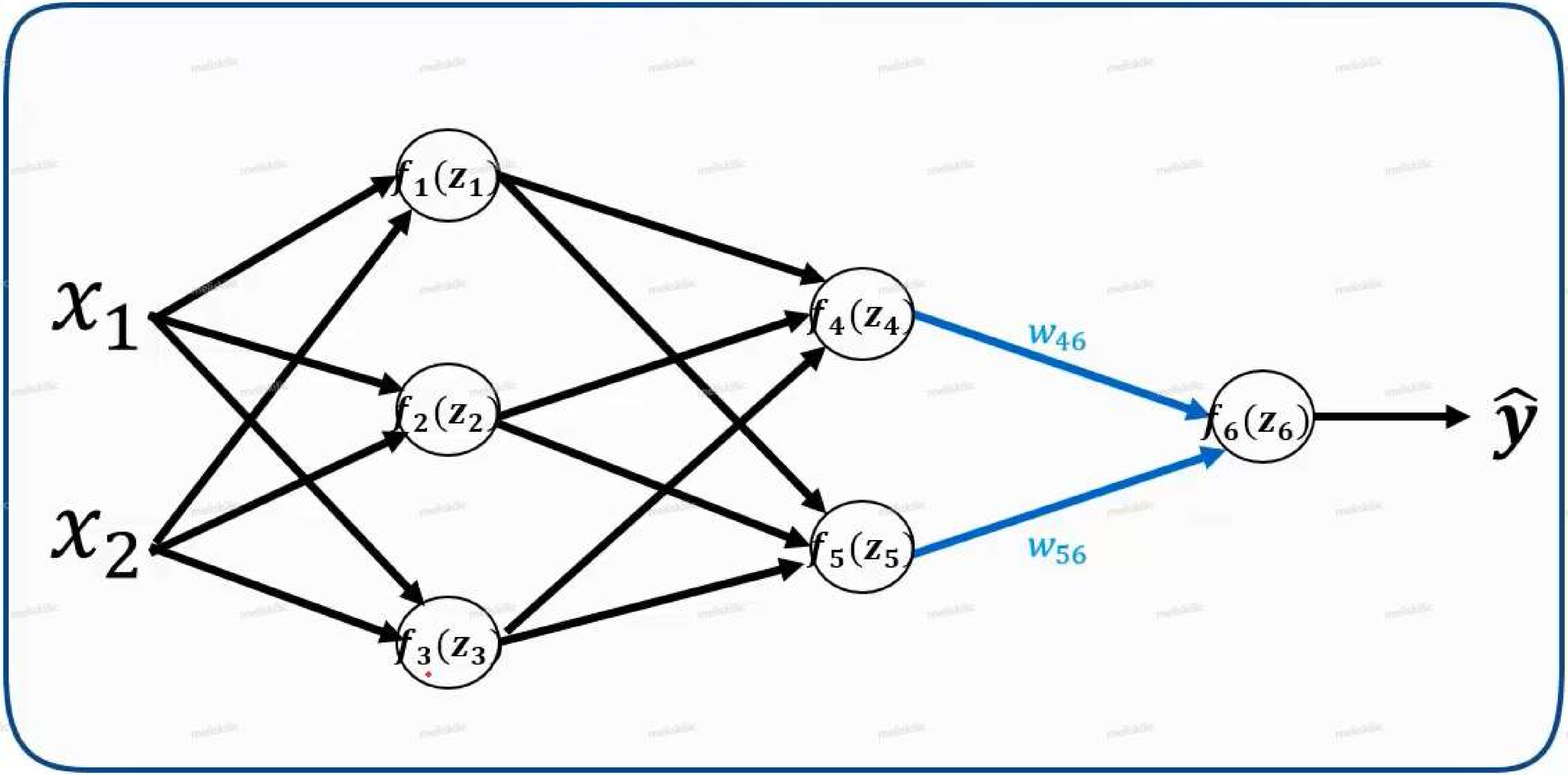
Forward Propagation (3)

$$y_4 = f_4(w_{14}y_1 + w_{24}y_2 + w_{34}y_3)$$



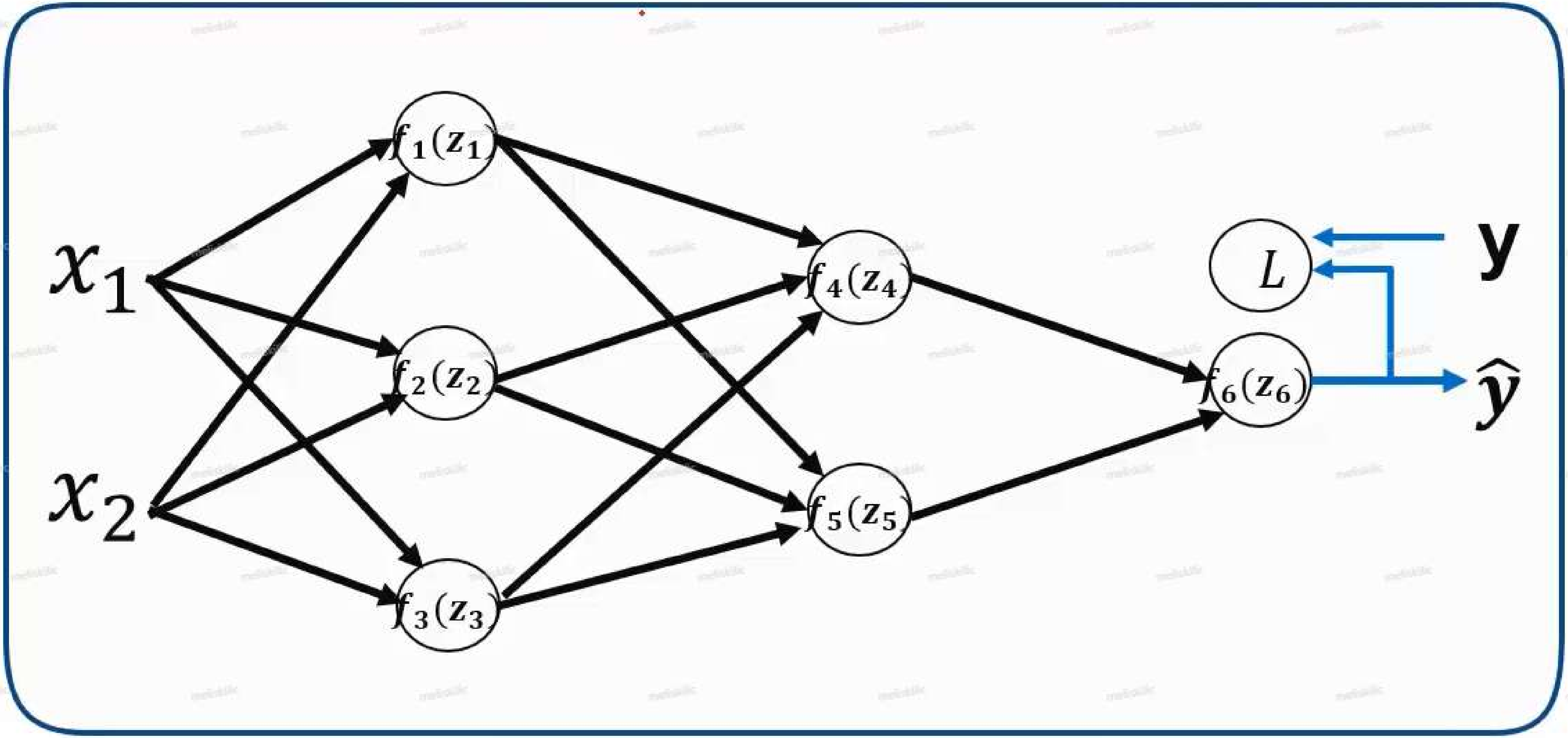
Forward Propagation (5)

$$y = f_6(w_{46}y_4 + w_{56}y_5)$$



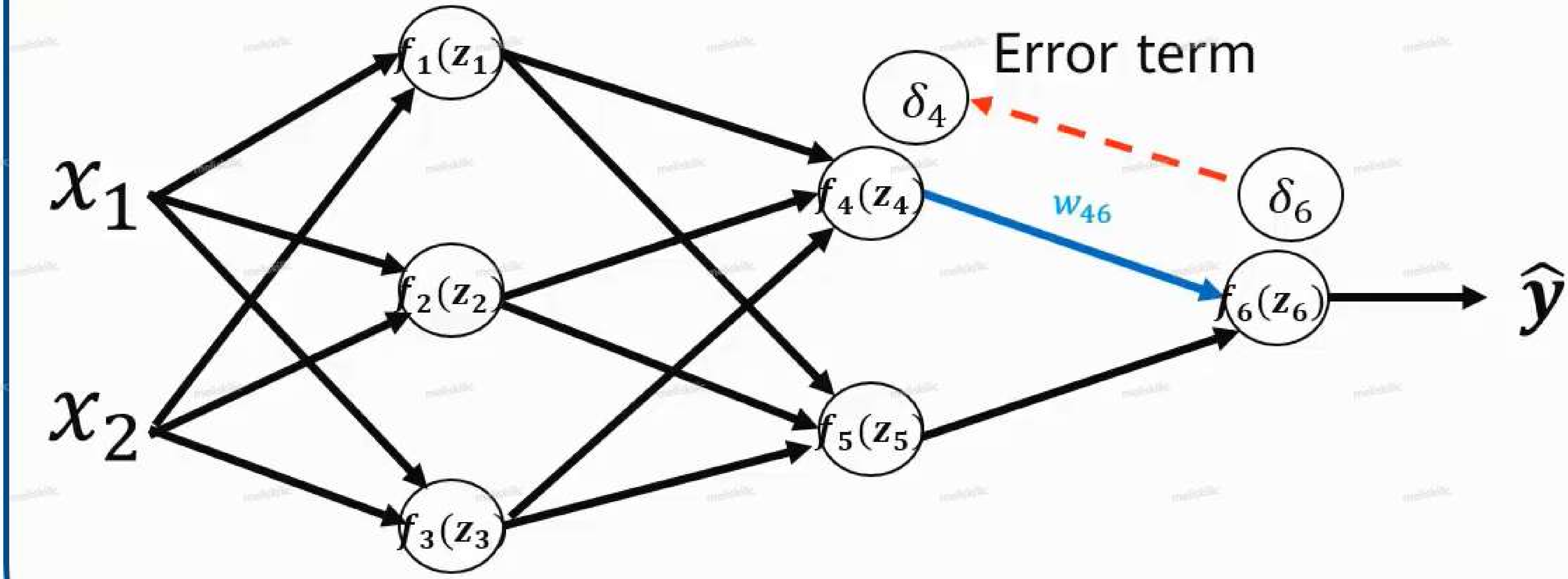
Error Backpropagation (1)

$$\delta_6 = \frac{\partial L(y,\hat{y})}{\partial z_6} = \frac{\partial L(y, f_6\,(z_6))}{\partial z_6} = \frac{\partial L(y, f_6(z_6))}{\partial f_6\,(z_6)} * \frac{\partial f_6(z_6)}{\partial z_6}$$



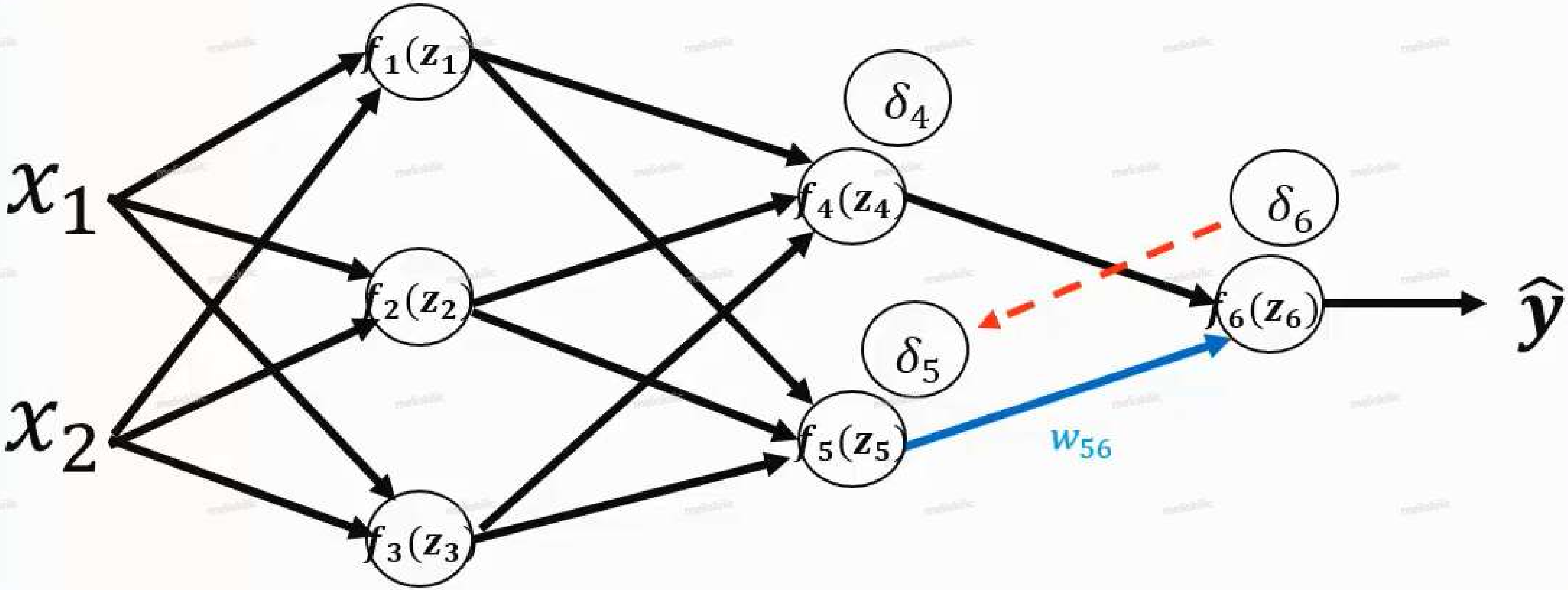
Error Backpropagation (2)

$$\delta_4 = \frac{\partial L}{\partial z_4} = \delta_6 * \underline{w_{46}} * f_4'(z_4)$$

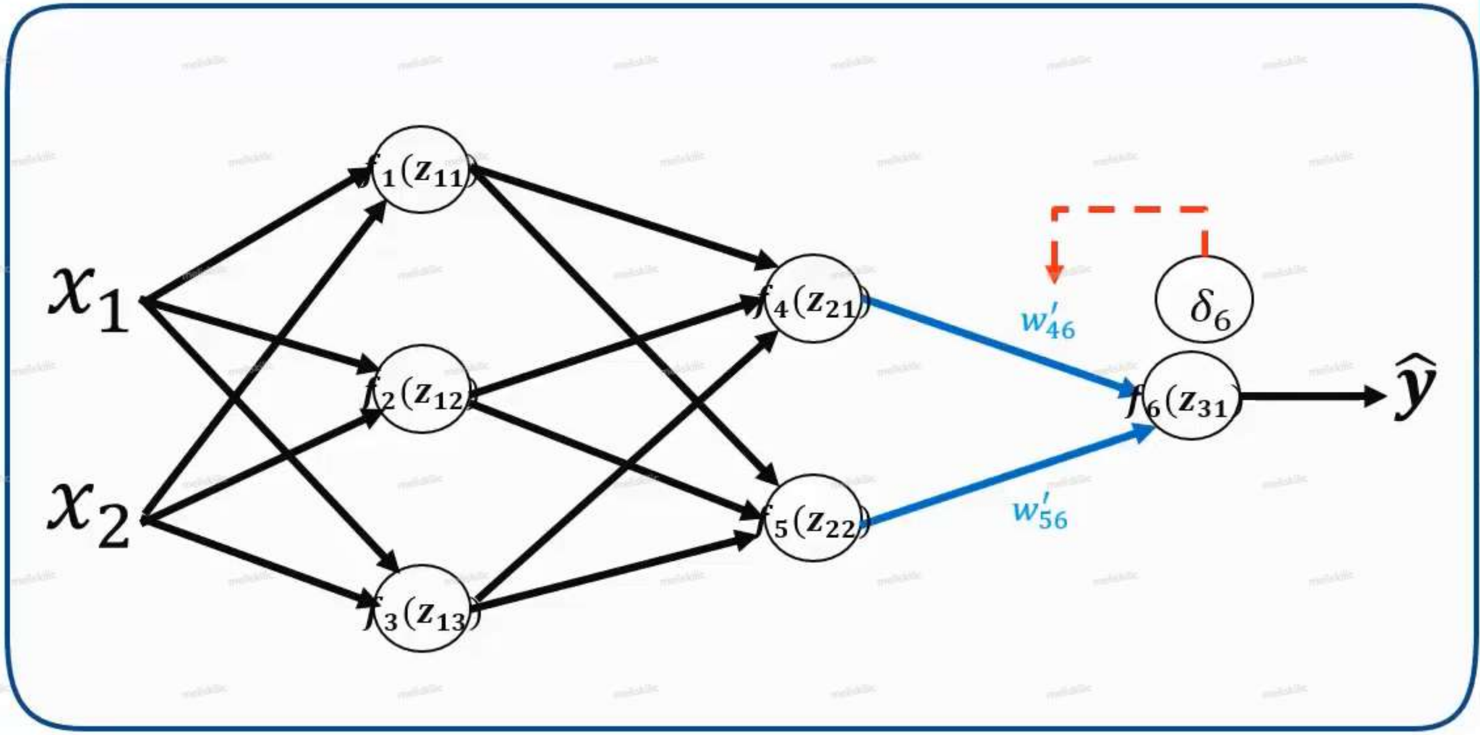


Error Backpropagation (3)

$$\delta_5 = \frac{\partial L}{\partial z_5} = \delta_6 * w_{56} * f_5'(z_5)$$

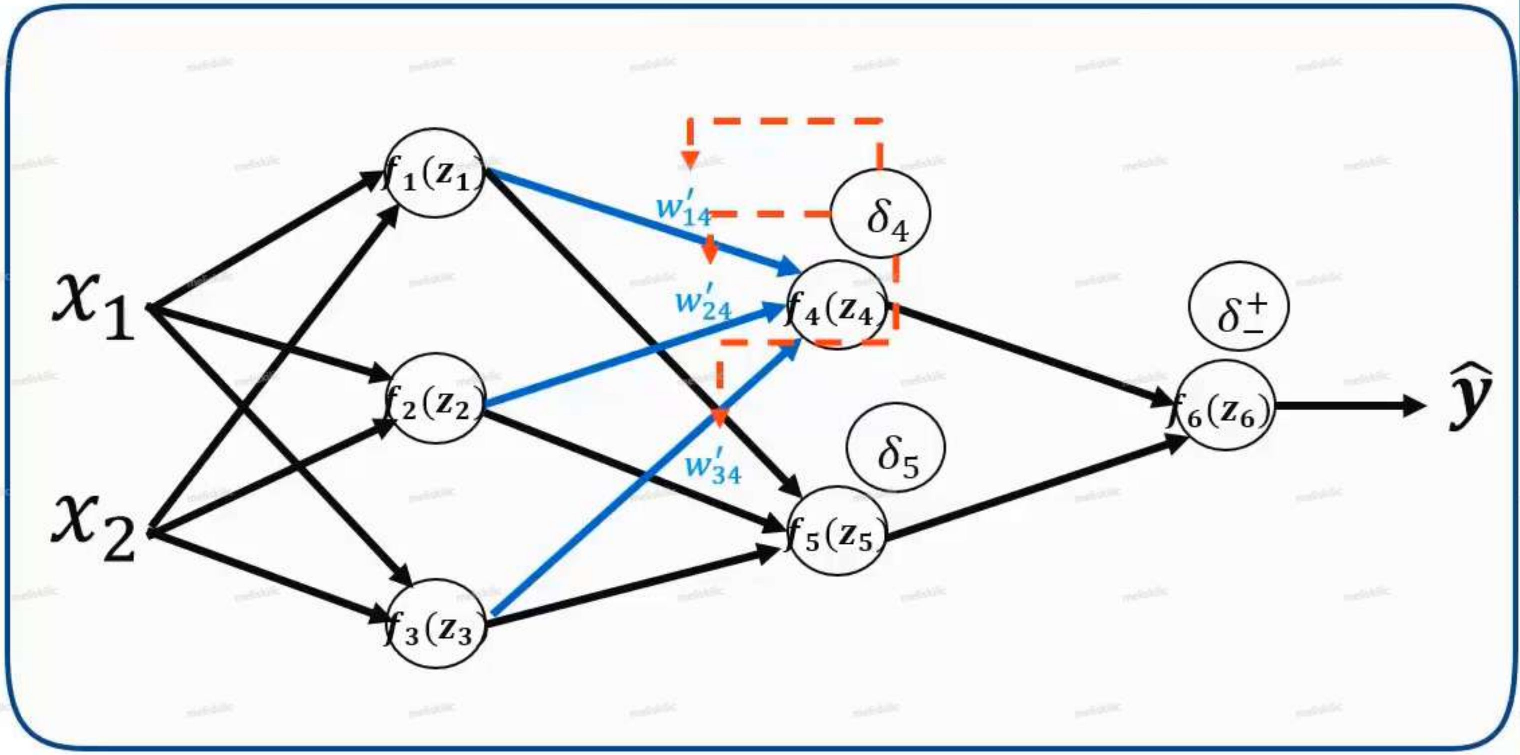


Weight Update (1)



$$
\begin{aligned}
w'_{46} &= w_{46} - \eta * \frac{\partial L(y,\hat{y})}{\partial w_{46}} \\
&= w_{46} - \eta * \frac{\partial L(y,\hat{y})}{\partial z_6} * \frac{\partial z_6}{\partial w_{46}} \\
&= w_{46} - \eta * \delta_6 * \frac{\partial (w_{46}y_4 + w_{56}y_5)}{\partial w_{46}} \\
&= w_{46} - \eta * \delta_6 * y_4
\end{aligned}
$$

Weight Update (2)

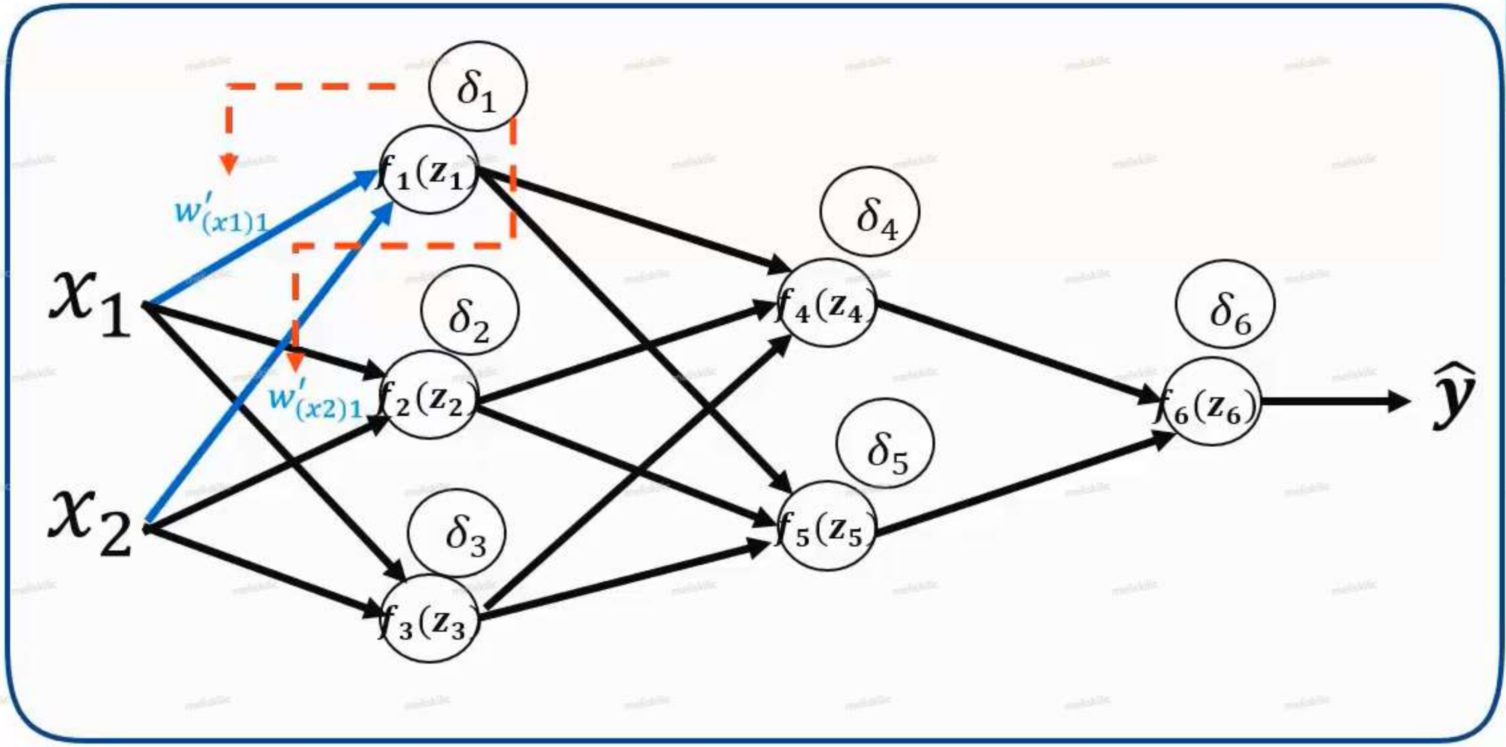


$$w'_{14} = w_{14} - \eta\delta_4 y_1$$

$$w'_{24} = w_{24} - \eta\delta_4 y_2$$

$$w'_{34} = w_{34} - \eta\delta_4 y_3$$

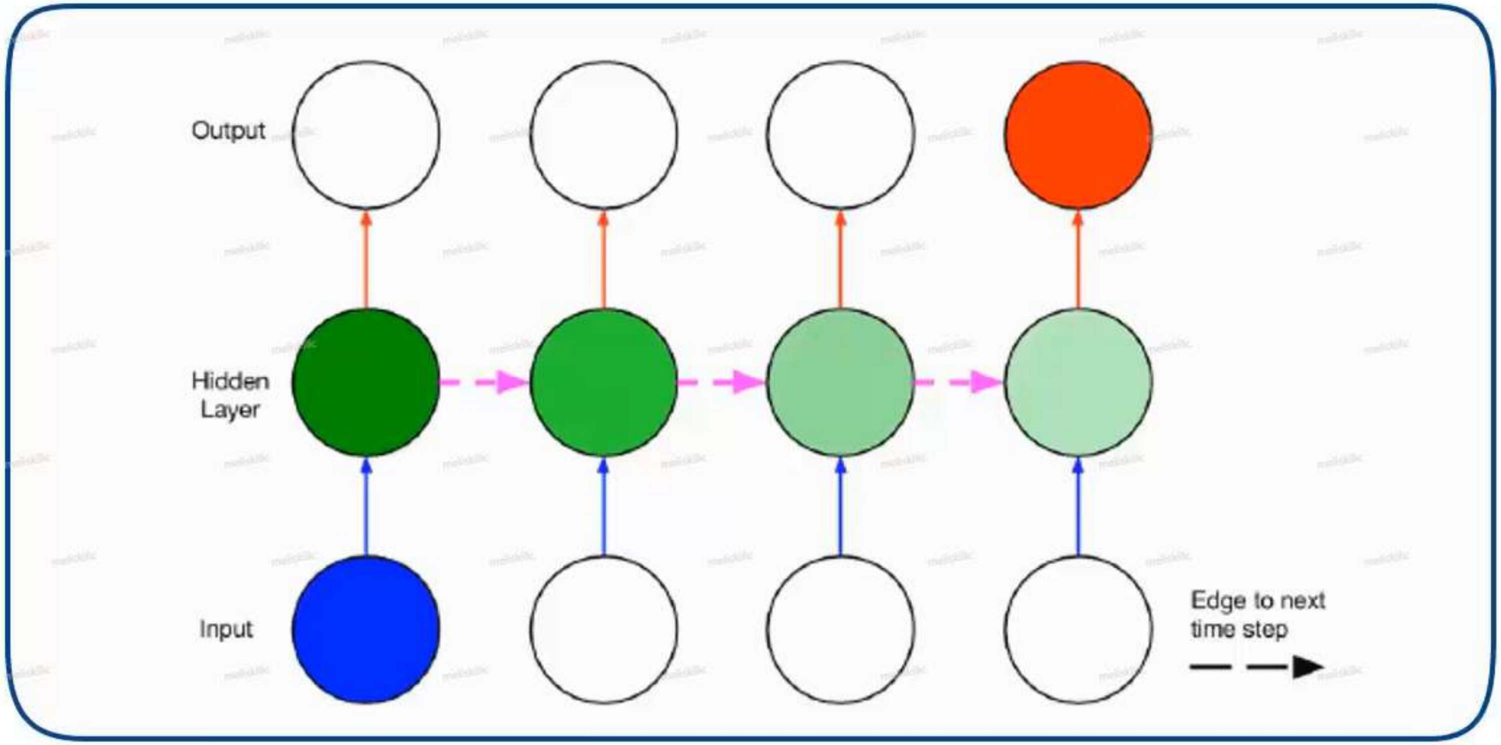
Weight Update (3)



$$w'_{(x1)1} = w_{(x1)1} - \eta\delta_1 x_1$$

$$w'_{(x2)1} = w_{(x2)1} - \eta\delta_1 x_2$$

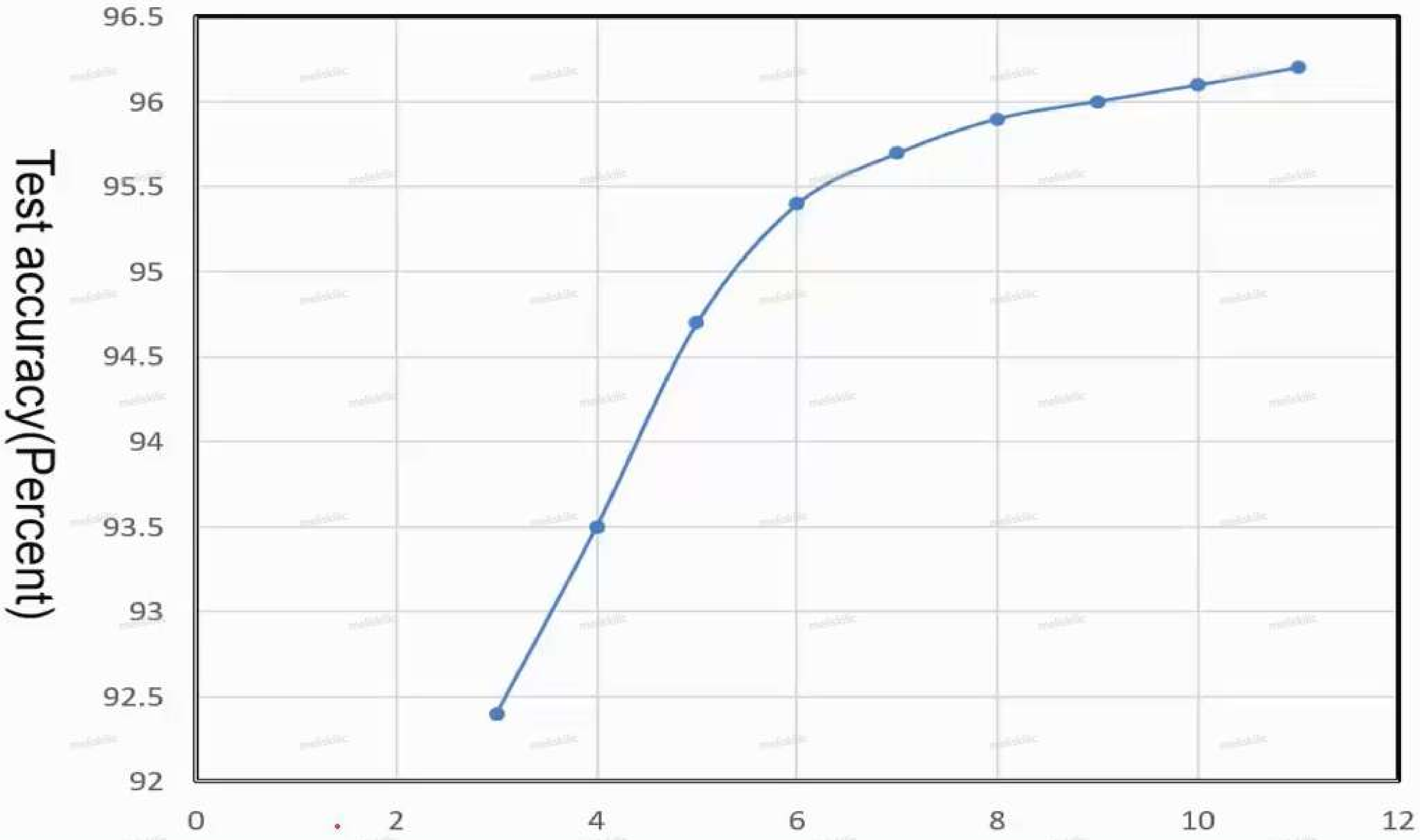
Gradient Problems



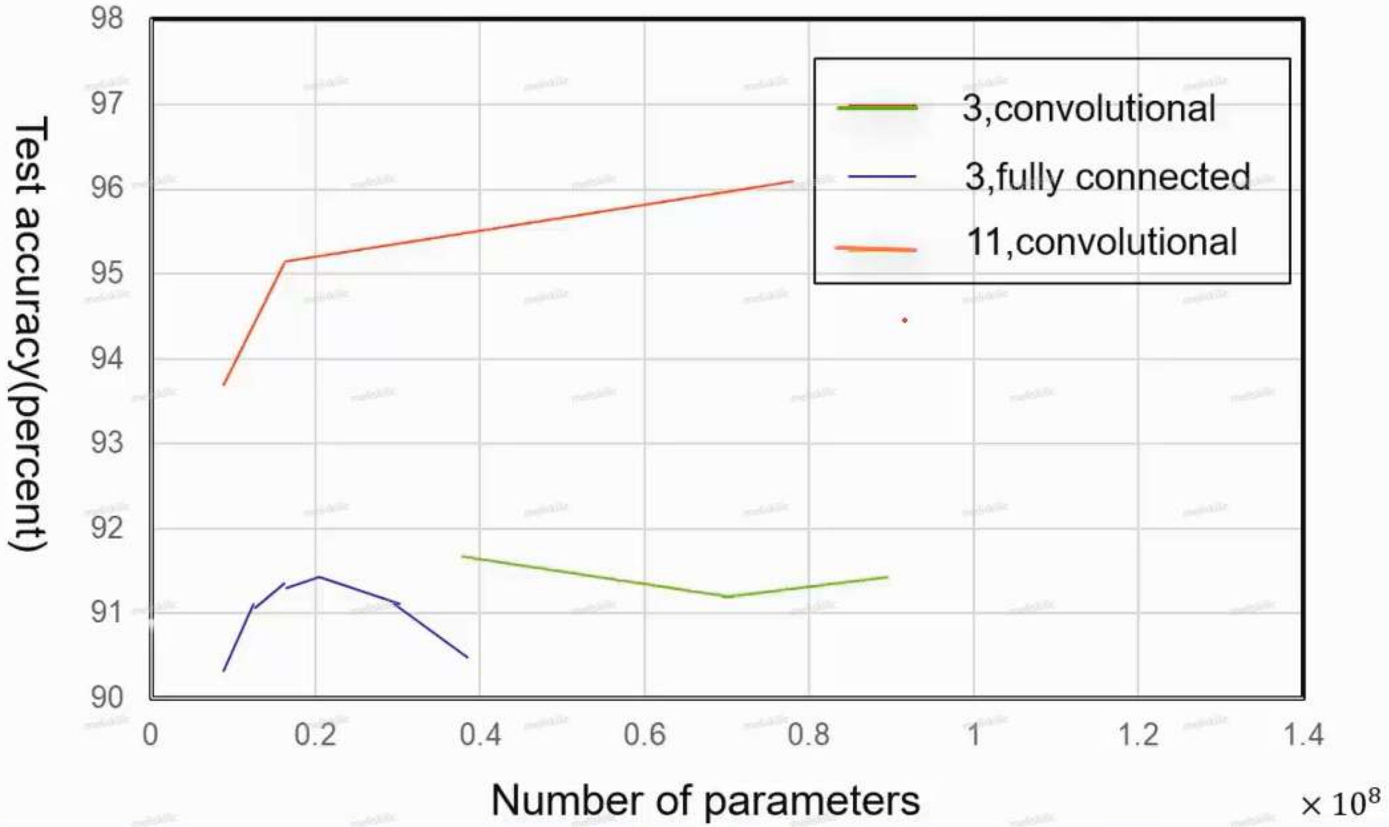
W is larger ➡ Exploding

W is small ➡ Vanishing

Effect of Neural Network Depth



Effect of the Number of Neural Network Parameters


Test accuracy(percent)
Number of parameters
$\times 10^8$
3,convolutional
3,fully connected
11,convolutional