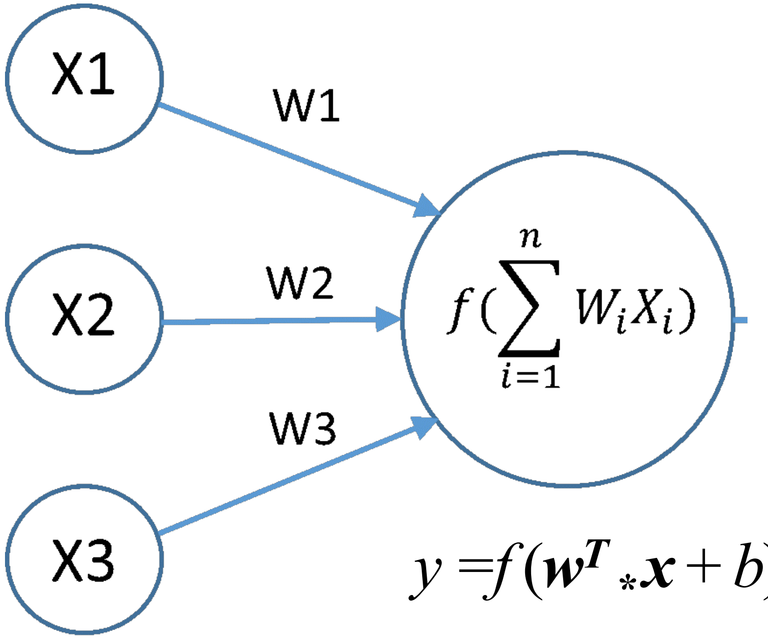
Introduction to artificial neural network (deep NN)

MIT Introduction to Deep Learning | 6.S191 https://youtu.be/5tvmMX8r_OM

Neuron (perceptron)

- Neuron performs weighted linear combination with bias and activation function

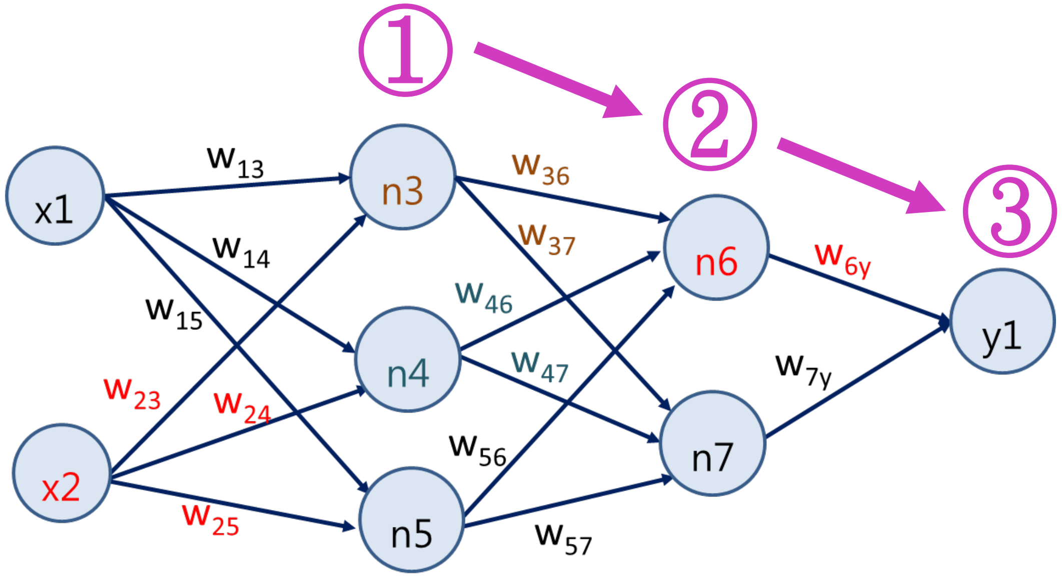
1.1. Perceptron.ipynb



$$y = f(\boldsymbol{w}^T * \boldsymbol{x} + b)$$

Multiple-layer perception (MLP)

1.2 MLP forward propagation.ipynb



$$\begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 10 & 5 \end{bmatrix} \qquad \begin{bmatrix} 7 \\ 12 \\ 40 \end{bmatrix}$$

①
$$n_3 = \sigma(x_1 * w_{13} + x_2 * w_{23} + b_3)$$
$$n_4 = \sigma(x_1 * w_{14} + x_2 * w_{24} + b_4)$$
$$n_5 = \sigma(x_1 * w_{15} + x_2 * w_{25} + b_5)$$

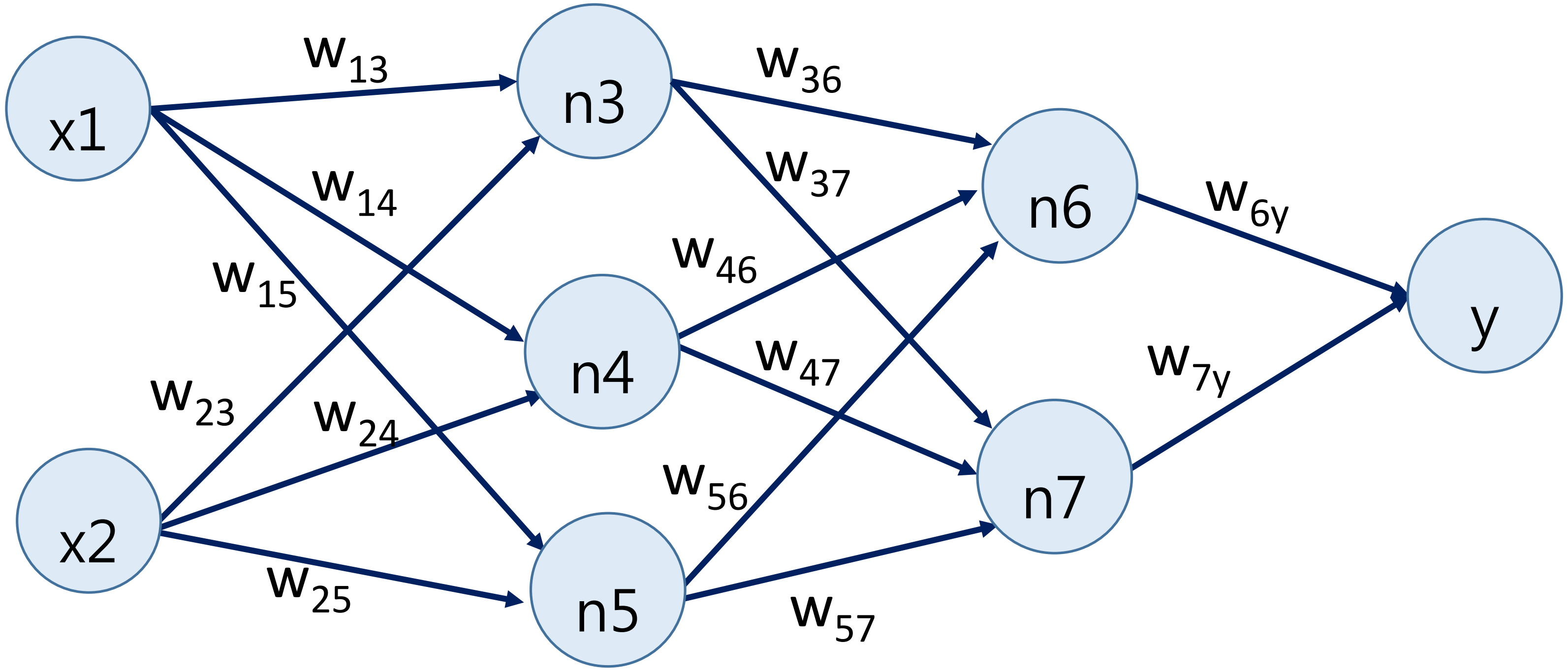
②
$$n_6 = \sigma(n_3 * w_{36} + n_4 * w_{46} + n_5 * w_{56} + b_6)$$
$$n_7 = \sigma(n_3 * w_{37} + n_4 * w_{47} + n_5 * w_{57} + b_7)$$

③
$$y_1 = \sigma(n_6 * w_{6y} + n_7 * w_{7y} + b_y)$$

Matrix operation

```
MyNet  =  nn.Sequential(
    nn.Linear(2,  3),
    nn.Linear(3,  2),
    nn.Linear(2,  1)
)
```

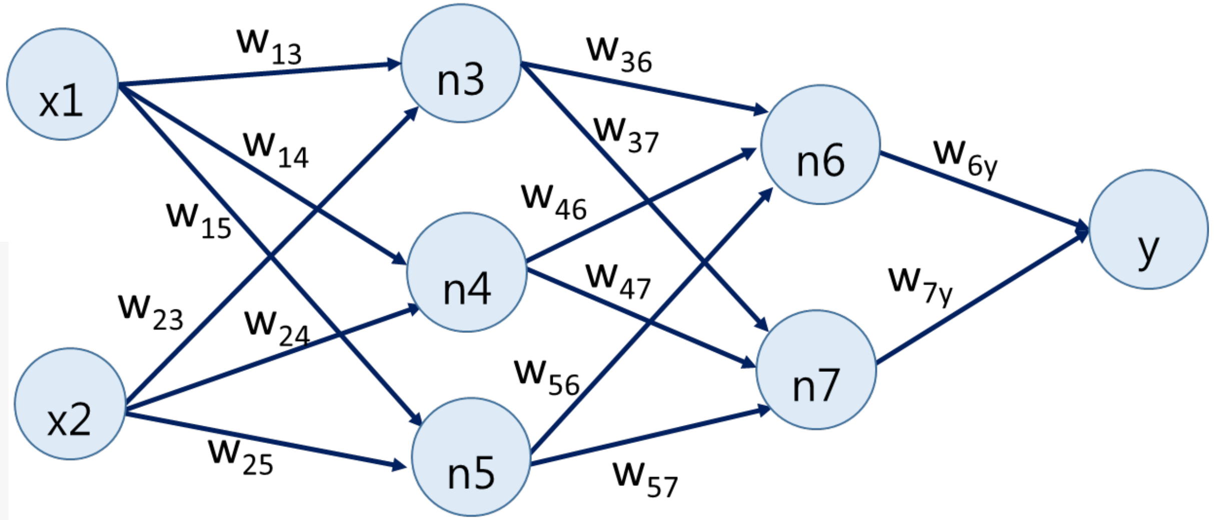
$$\vec{X} = \begin{array}{c} \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 10 & 5 \end{bmatrix} \end{array}$$



$$\hat{y} = \begin{bmatrix} 7 \\ 12 \\ 40 \end{bmatrix}$$

Matrix operation

```python
for param in MyNet.parameters():
    if param.requires_grad:
        print(param.data)
```




```
tensor([[ 0.4727, -0.5188],
        [-0.5681, -0.6032],
        [-0.0252, -0.3011]])
tensor([-0.6986, -0.6602, -0.4860])
tensor([[-0.5549,  0.2550,  0.4584],
        [ 0.2930,  0.0849, -0.3146]])
tensor([0.1677, 0.0736])
tensor([[ 0.4106, -0.3618]])
tensor([-0.2270])
```

$$\begin{bmatrix} w_{13} & w_{23} \\ w_{14} & w_{24} \\ w_{15} & w_{25} \end{bmatrix}$$

$$[b_3 \quad b_4 \quad b_5]$$

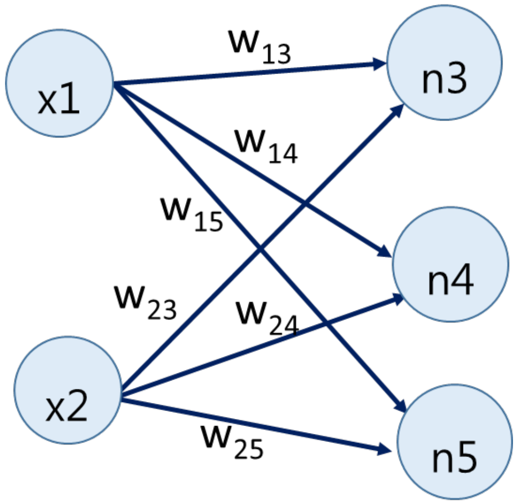
$$\begin{bmatrix} w_{36} & w_{46} & w_{56} \\ w_{37} & w_{47} & w_{57} \end{bmatrix}$$

$$[b_6 \quad b_7]$$

$$[w_{6y} \quad w_{7y}]$$

$$[b_y]$$

$$\begin{array}{cc} x_1 & x_2 \end{array}$$
$$\vec{X} = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 10 & 5 \end{bmatrix}$$




```
W1 = MyNet[0].weight
b1 = MyNet[0].bias
print(W1, W1.shape, b1)
```

```
Parameter containing:
tensor([[ 0.4727, -0.5188],
        [-0.5681, -0.6032],
        [-0.0252, -0.3011]],
tensor([-0.6986, -0.6602, -0.4860], r
```

$$\begin{bmatrix} w_{13} & w_{23} \\ w_{14} & w_{24} \\ w_{15} & w_{25} \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 10 & 5 \end{bmatrix} \begin{bmatrix} w_{13} & w_{14} & w_{15} \\ w_{23} & w_{24} & w_{25} \end{bmatrix}} + [b_3 \quad b_4 \quad b_5]$$

$$\underbrace{\begin{bmatrix} k_3^1 & k_4^1 & k_5^1 \\ k_3^2 & k_4^2 & k_5^2 \\ k_3^3 & k_4^3 & k_5^3 \end{bmatrix} + \begin{bmatrix} b_3 & b_4 & b_5 \\ b_3 & b_4 & b_5 \\ b_3 & b_4 & b_5 \end{bmatrix}}$$

$$\boxed{\begin{bmatrix} n_3^1 & n_4^1 & n_5^1 \\ n_3^2 & n_4^2 & n_5^2 \\ n_3^3 & n_4^3 & n_5^3 \end{bmatrix}}$$

Use Excel to verify

```
#Calculate n3, n4, n5
HiddenLayer1 = MyNet[0](tensorX)
print(HiddenLayer1)
```

```
tensor([[-1.2635, -2.4348, -1.1135],
        [-1.3097, -3.6061, -1.4398],
        [ 1.4340, -9.3577, -2.2441]],
```

```
#Calculate n3, n4, n5 using Pytorch matrix operation
HiddenLayer1 = tensorX.mm(torch.transpose(W1, 1, 0)) + b1
print(HiddenLayer1)
```

```
tensor([[-1.2635, -2.4348, -1.1135],
        [-1.3097, -3.6061, -1.4398],
        [ 1.4340, -9.3577, -2.2441]], grad_fn=<AddBackward0>)
```

```python
#Calculate n6, n7 using PyTorch matrix operation
W2 = MyNet[1].weight
b2 = MyNet[1].bias
HiddenLayer2 = HiddenLayer1.mm(torch.transpose(W2, 1, 0)) +b2
print(HiddenLayer2)
```

```
tensor([[-0.2625, -0.1530],
        [-0.6852, -0.1632],
        [-4.0429,  0.4054]], grad_fn=<AddBackward0>)
```

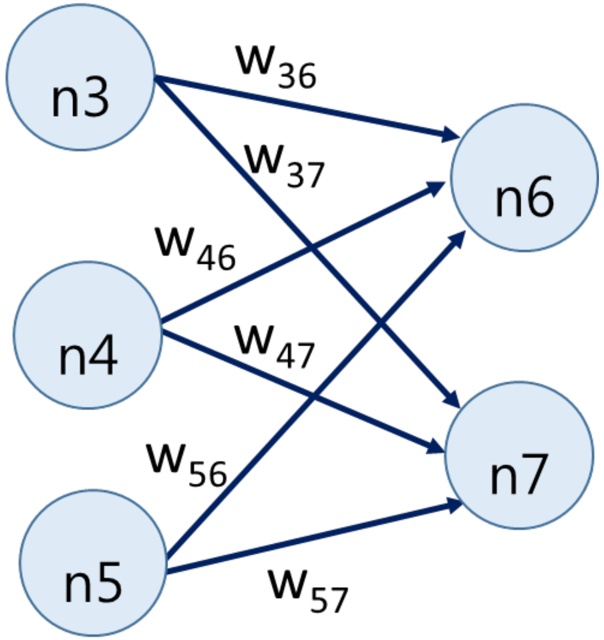
$$\begin{bmatrix} n_3^1 & n_4^1 & n_5^1 \\ n_3^2 & n_4^2 & n_5^2 \\ n_3^3 & n_4^3 & n_5^3 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} -1.2635 & -2.4348 & -1.1135 \\ -1.3097 & -3.6061 & -1.4398 \\ 1.4340 & -9.3577 & -2.2441 \end{bmatrix} \begin{bmatrix} w_{36} & w_{37} \\ w_{46} & w_{47} \\ w_{56} & w_{57} \end{bmatrix}} + [b_6 \quad b_7]$$

$$\underbrace{\begin{bmatrix} k_6^1 & k_7^1 \\ k_6^2 & k_7^2 \\ k_6^3 & k_7^3 \end{bmatrix} + \begin{bmatrix} b_6 & b_7 \\ b_6 & b_7 \\ b_6 & b_7 \end{bmatrix}}$$

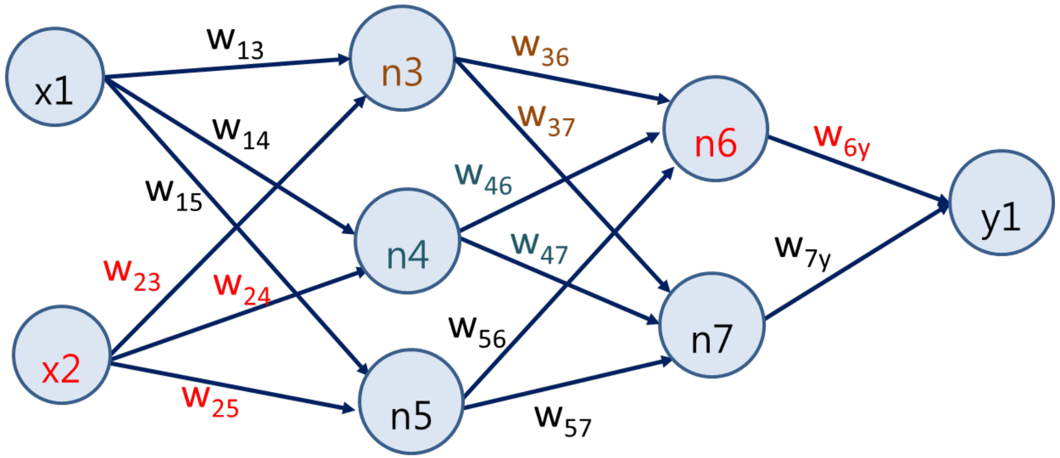
$$\begin{bmatrix} n_6^1 & n_7^1 \\ n_6^2 & n_7^2 \\ n_6^3 & n_7^3 \end{bmatrix}$$

Use Excel to verify



Calculate prediction error

1.3 MLP backward propagation.ipynb



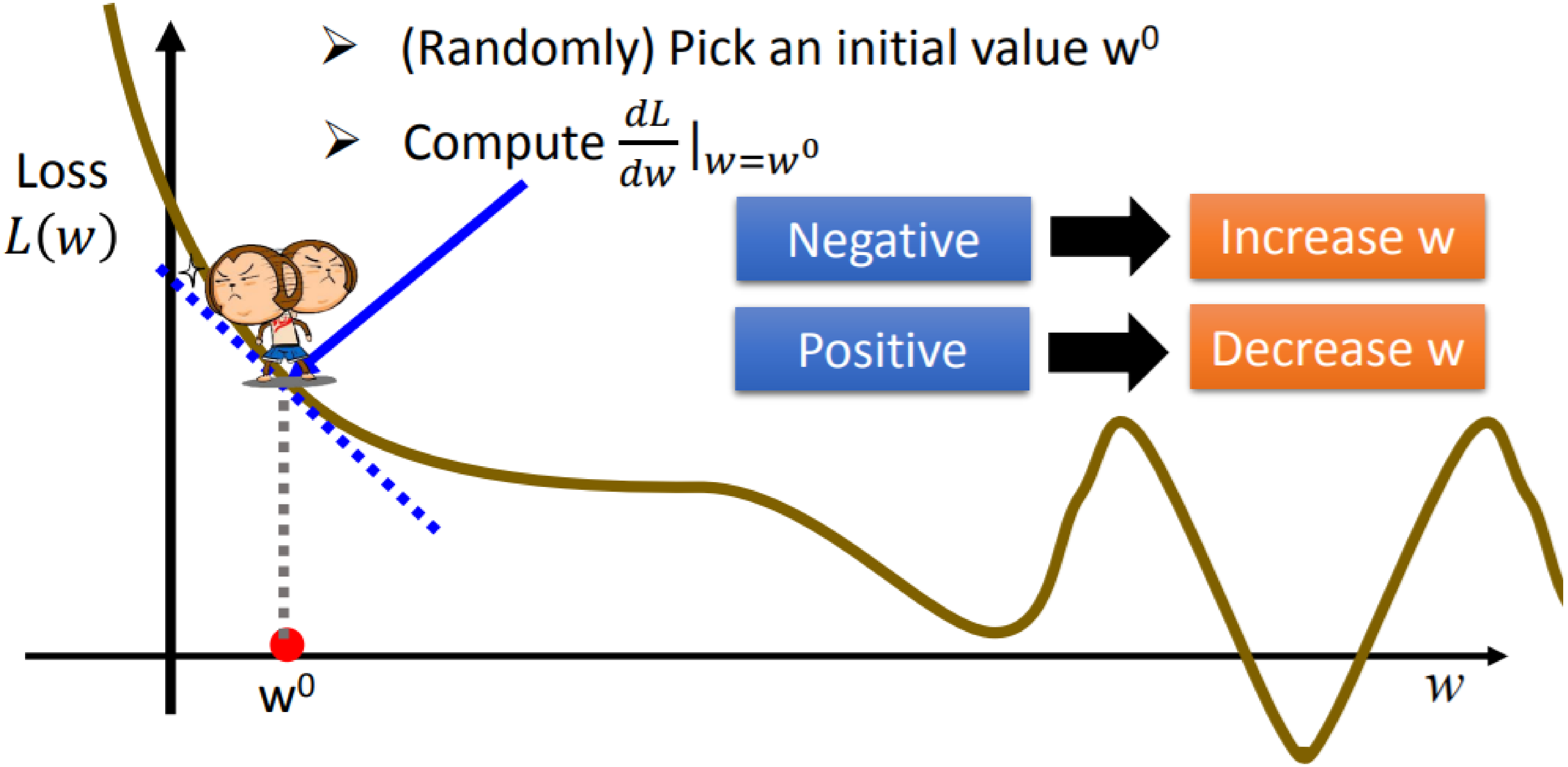
$$L = \frac{1}{N}\sum_{i=1}^{N}(y^i - \hat{y}^i)^2$$

Use gradient decent to find optimal parameters

3. Find the optimal parameters that minimize $\mathcal{L}(f)$

$$w^* = arg \min_{w} L(w)$$

- Consider loss function $L(w)$ with one parameter w:
  - (Randomly) Pick an initial value w^0
  - Compute $\frac{dL}{dw}|_{w=w^0}$

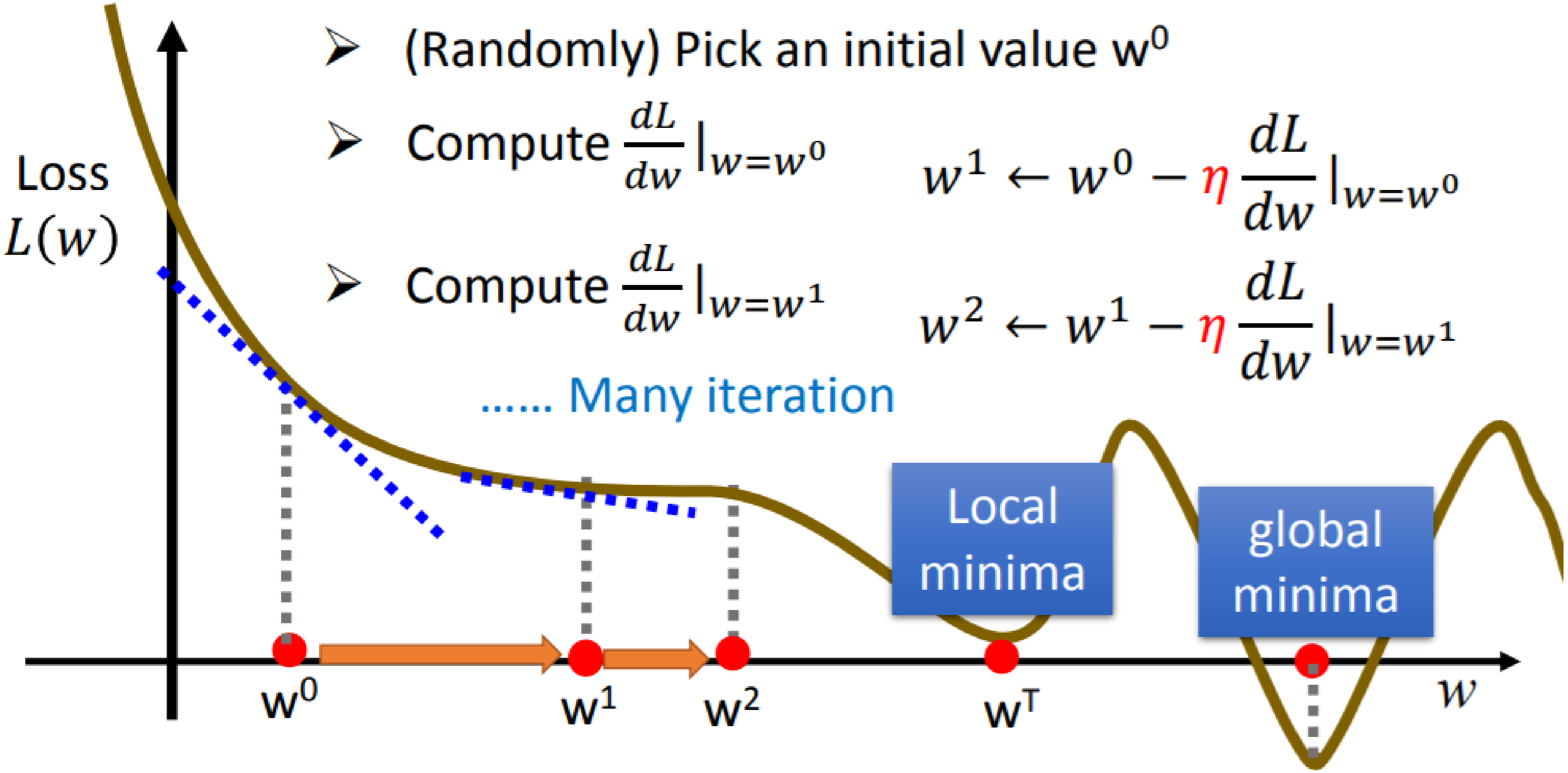


Use gradient decent to find optimal parameters

$$w^* = arg \min_{w} L(w)$$

- Consider loss function $L(w)$ with one parameter w:
  - (Randomly) Pick an initial value w^0
  - Compute $\frac{dL}{dw}|_{w=w^0}$ $\quad w^1 \leftarrow w^0 - \eta \frac{dL}{dw}|_{w=w^0}$
  - Compute $\frac{dL}{dw}|_{w=w^1}$ $\quad w^2 \leftarrow w^1 - \eta \frac{dL}{dw}|_{w=w^1}$

…… Many iteration



Gradient decent to find two parameters w^* and b^*

- How about two parameters? $w^*, b^* = arg \min_{w,b} L(w, b)$

  - (Randomly) Pick an initial value w^0, b^0

  - Compute $\frac{\partial L}{\partial w}|_{w=w^0,b=b^0}, \frac{\partial L}{\partial b}|_{w=w^0,b=b^0}$

$$\boxed{\begin{bmatrix} \frac{\partial L}{\partial w} \\ \frac{\partial L}{\partial b} \end{bmatrix} \text{gradient}}$$

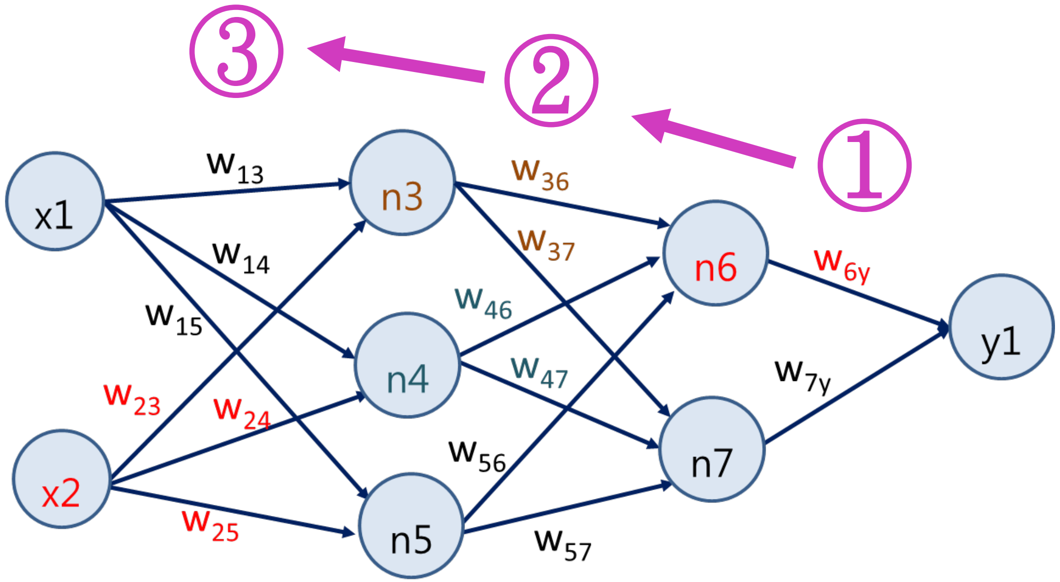
$$w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}|_{w=w^0,b=b^0} \qquad b^1 \leftarrow b^0 - \eta \frac{\partial L}{\partial b}|_{w=w^0,b=b^0}$$

  - Compute $\frac{\partial L}{\partial w}|_{w=w^1,b=b^1}, \frac{\partial L}{\partial b}|_{w=w^1,b=b^1}$

$$w^2 \leftarrow w^1 - \eta \frac{\partial L}{\partial w}|_{w=w^1,b=b^1} \qquad b^2 \leftarrow b^1 - \eta \frac{\partial L}{\partial b}|_{w=w^1,b=b^1}$$

Reference: 李宏毅 ML Lecture 1 https://youtu.be/CXgbekl66jc

Use gradient decent to find optimal NN weights



$L = g(y - \hat{y})$ $\quad y = \sigma(n_6 * w_{6y} + n_7 * w_{7y} + b_y)$

①

$$w_{6y} \leftarrow w_{6y} - \eta \frac{\partial L}{\partial w_{6y}} \qquad \frac{\partial L}{\partial w_{6y}} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w_{6y}}$$

$$w_{7y} \leftarrow w_{7y} - \eta \frac{\partial L}{\partial w_{7y}} \qquad \frac{\partial L}{\partial w_{7y}} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w_{7y}}$$

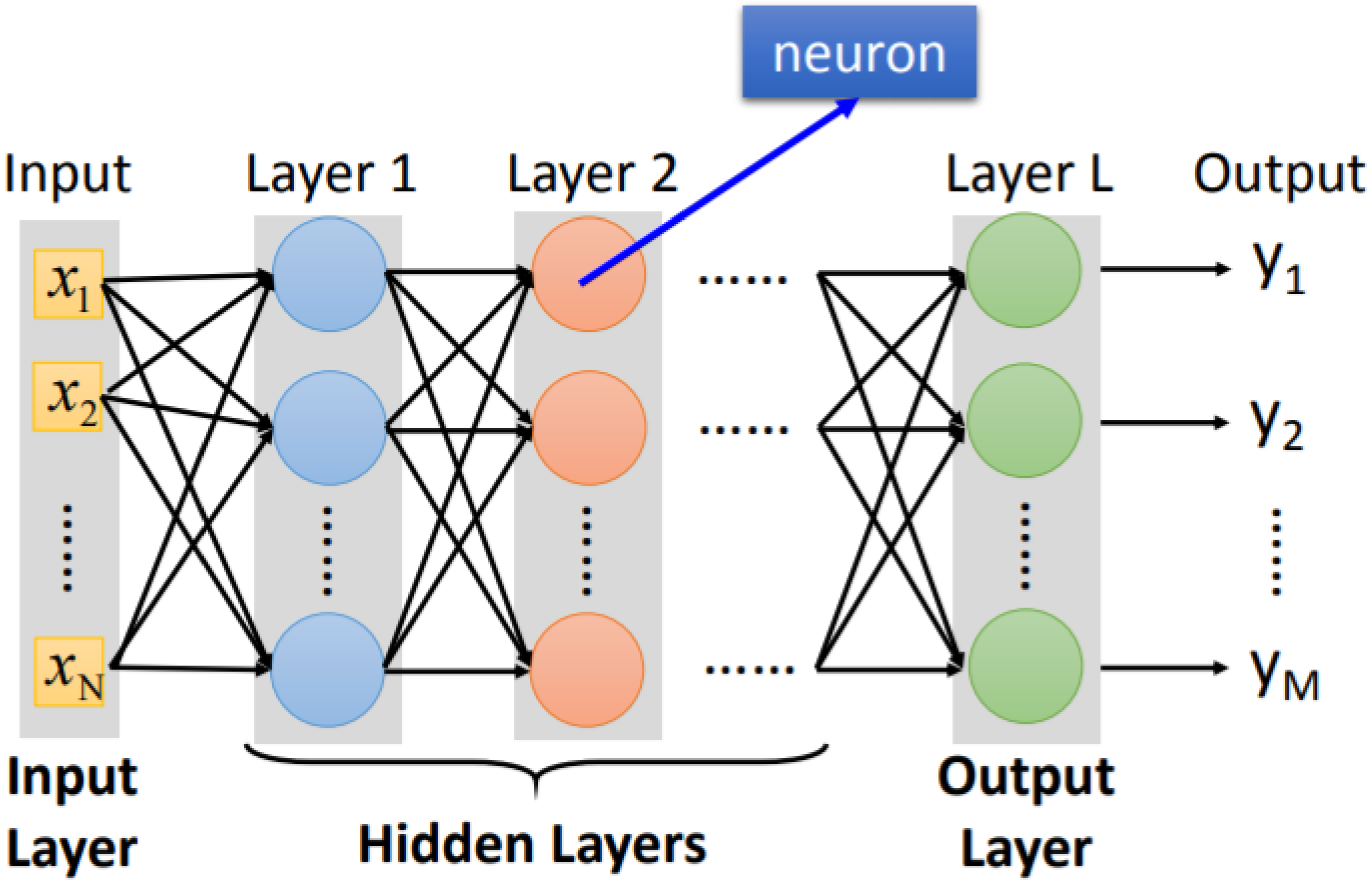
$$w_i \leftarrow w_i - \eta \frac{\partial e}{\partial w_i}$$

②

$$w_{57} \leftarrow w_{57} - \eta \frac{\partial L}{\partial w_{57}} \qquad \frac{\partial L}{\partial w_{57}} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial n_7} \frac{\partial n_7}{\partial w_{57}}$$

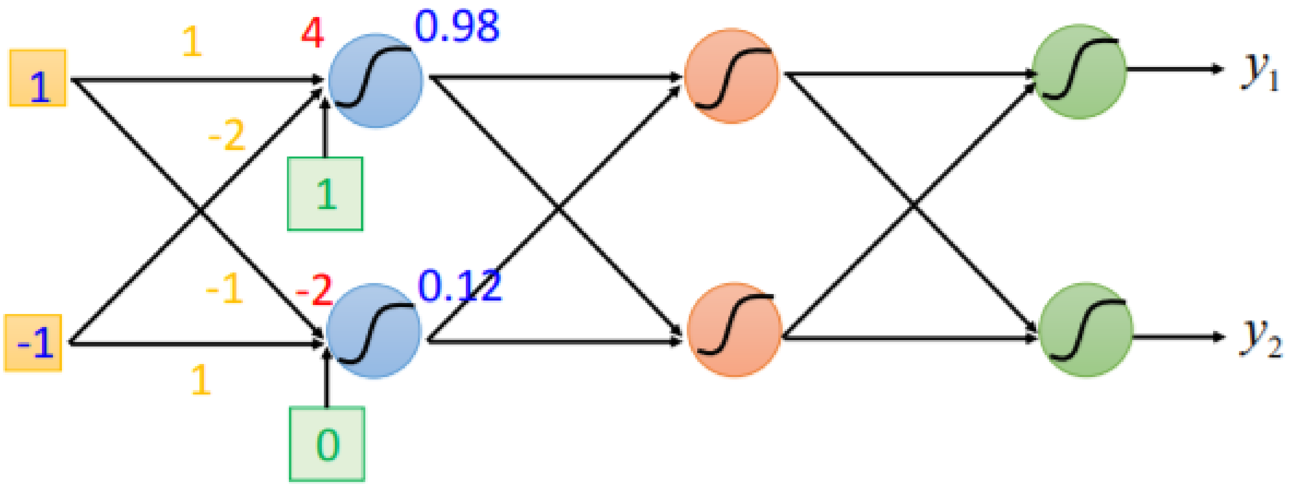
$$n_7 = f(n_3 * w_{37} + n_4 * w_{47} + n_5 * w_{57} + b_7)$$

MLP is a fully connected feedforward network



Reference: 李弘毅 ML Lecture 6 https://youtu.be/Dr-WRlEFefw

Fully connected feed forward network is implemented as matrix operation

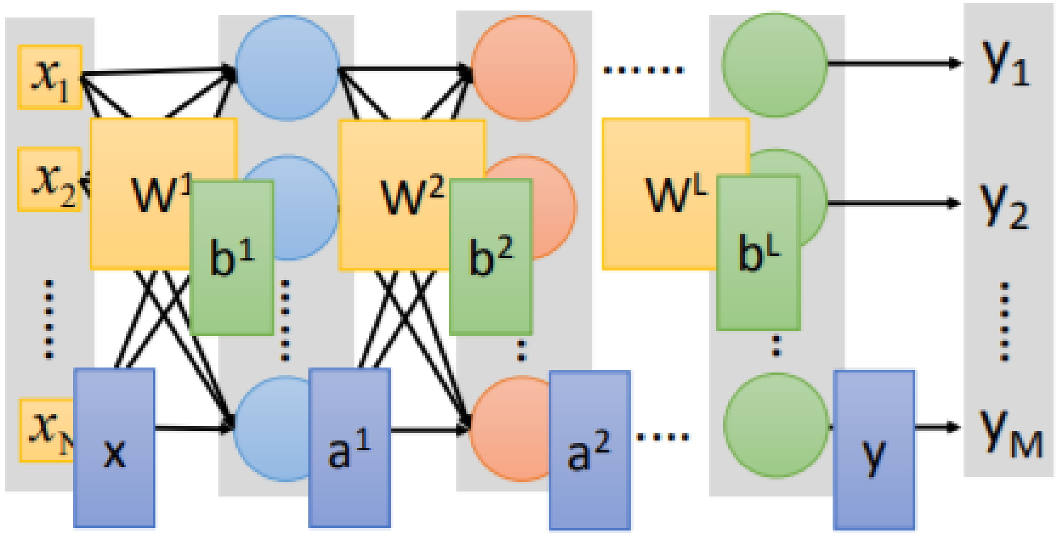


$$y = \sigma(w \cdot x + b)$$

$$\sigma\Big(\underbrace{\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\begin{bmatrix} 4 \\ -2 \end{bmatrix}}\Big) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

Reference: 李弘毅 ML Lecture 6 https://youtu.be/Dr-WRlEFefw

Use parallel computing to speed up matrix operation



$$y = f(x)$$

Using parallel computing techniques to speed up matrix operation

$$= \sigma(W^L \cdots \sigma(W^2 \; \sigma(W^1 \; x + b^1) + b^2) \cdots + b^L)$$

Reference: 李弘毅 ML Lecture 6 https://youtu.be/Dr-WRlEFefw