



# *NEURAL NETWORK*

HYUN SOO PARK

# CHALLENGES OF VISUAL RECOGNITION

- **Appearance**

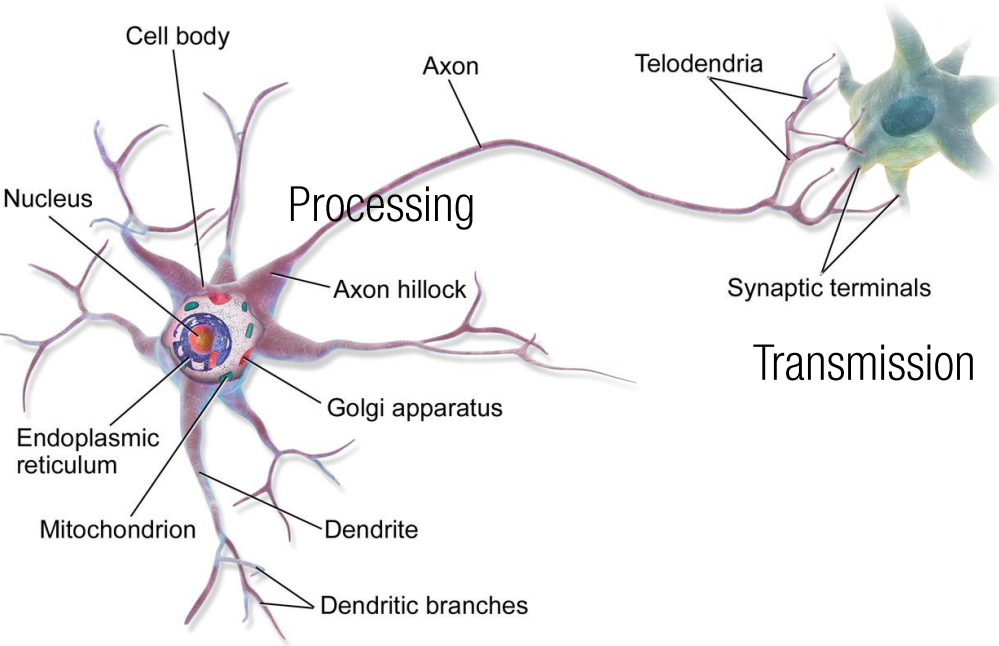
- DOF: texture, illumination, material, shading, ...

- **Shape**

- DOF: object category, geometric pose, viewpoint, ...



# NEURON



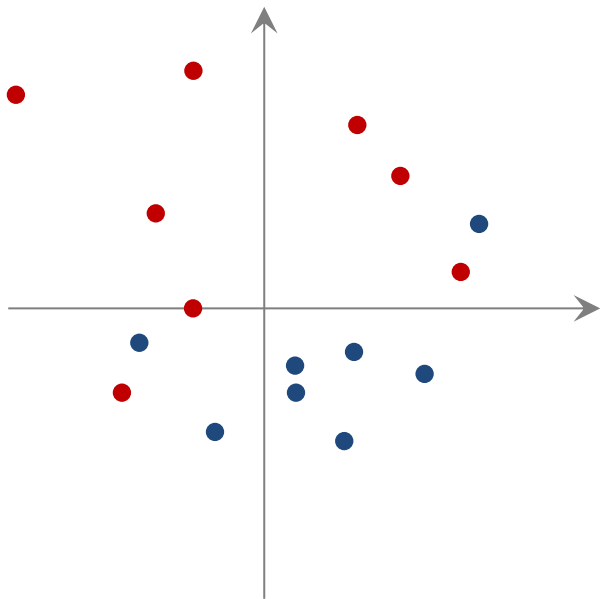
Signal reception

Processing

Transmission

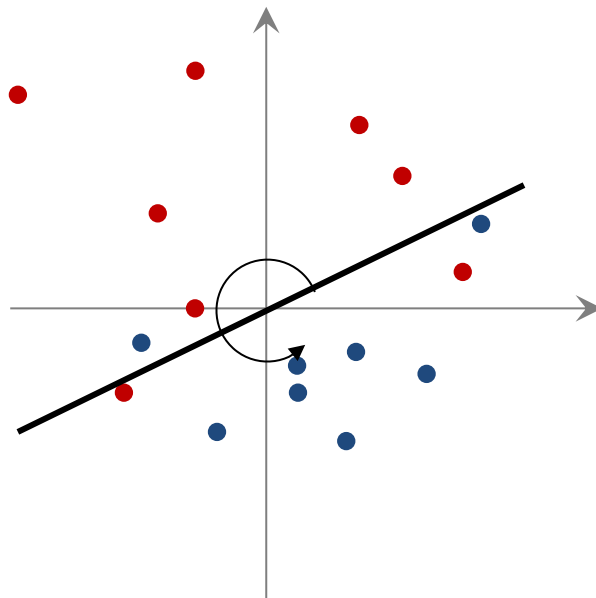
# *PERCEPTRON*

Binary classification

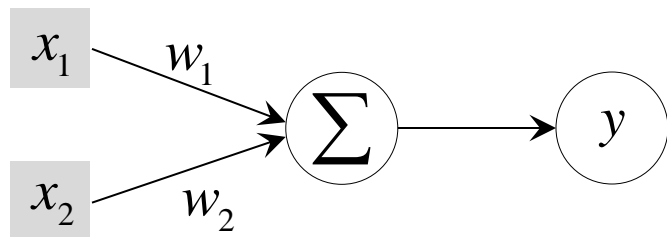


# *PERCEPTRON*

Binary classification

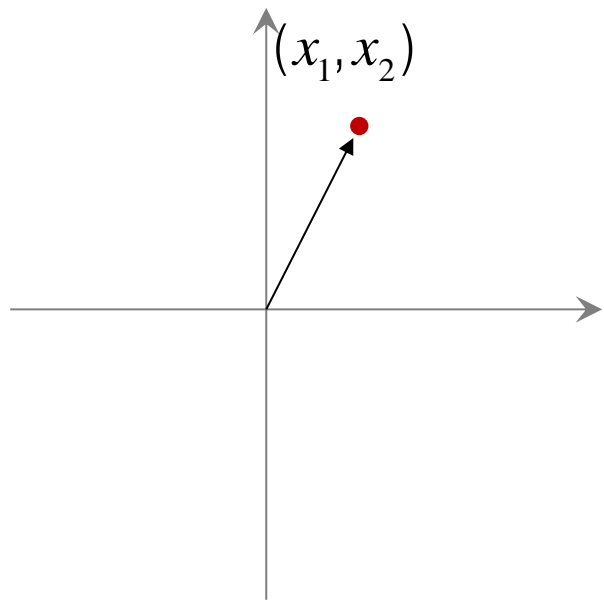


# PERCEPTRON

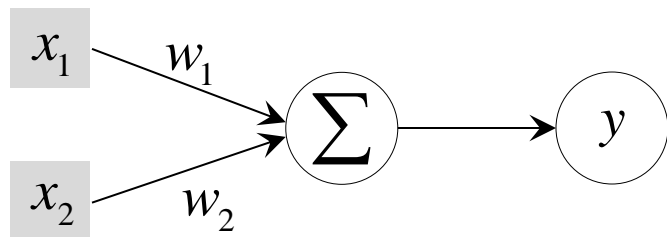


$$y = x_1 w_1 + x_2 w_2$$

Binary classification

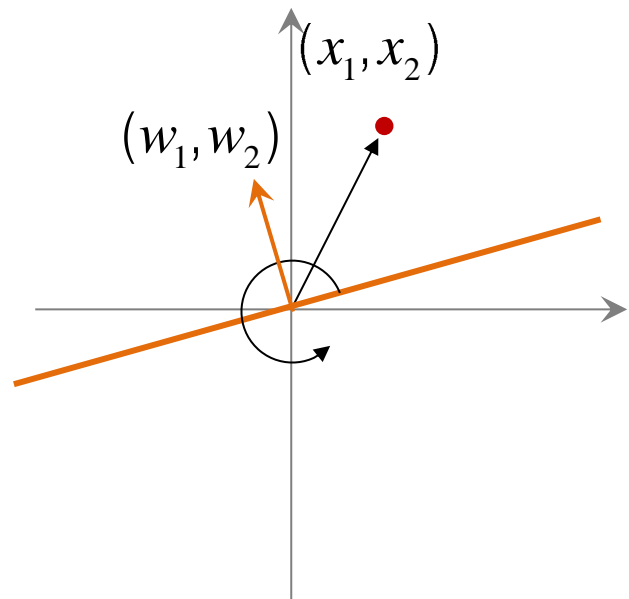


# PERCEPTRON

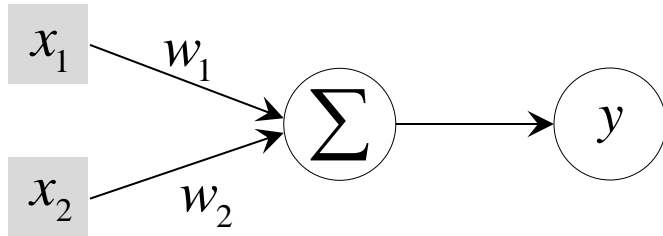


$$y = x_1 w_1 + x_2 w_2$$

Binary classification

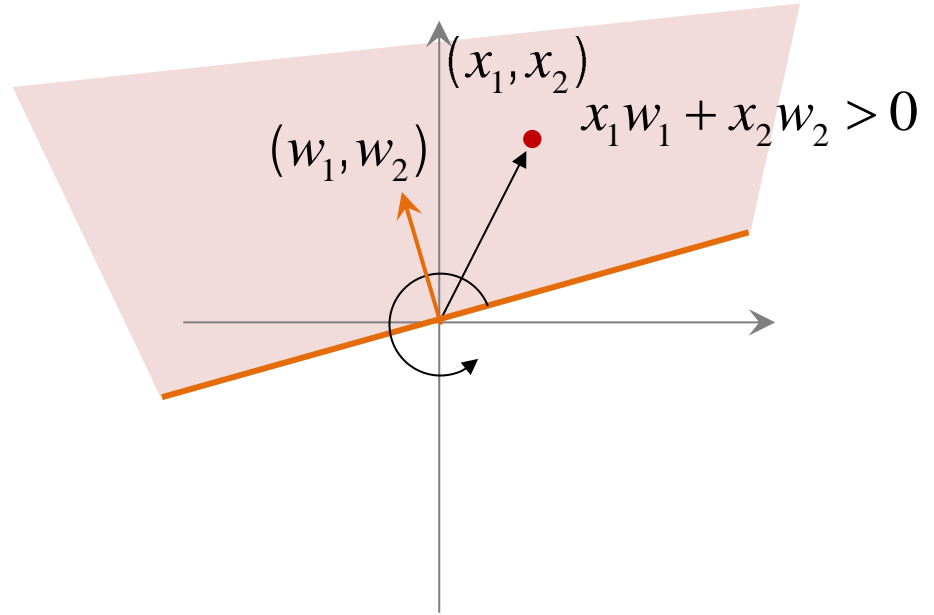


# PERCEPTRON



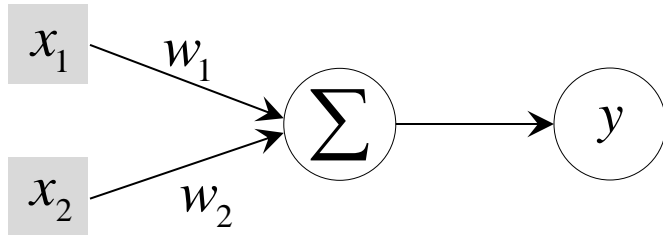
$$y = x_1 w_1 + x_2 w_2$$

Binary classification



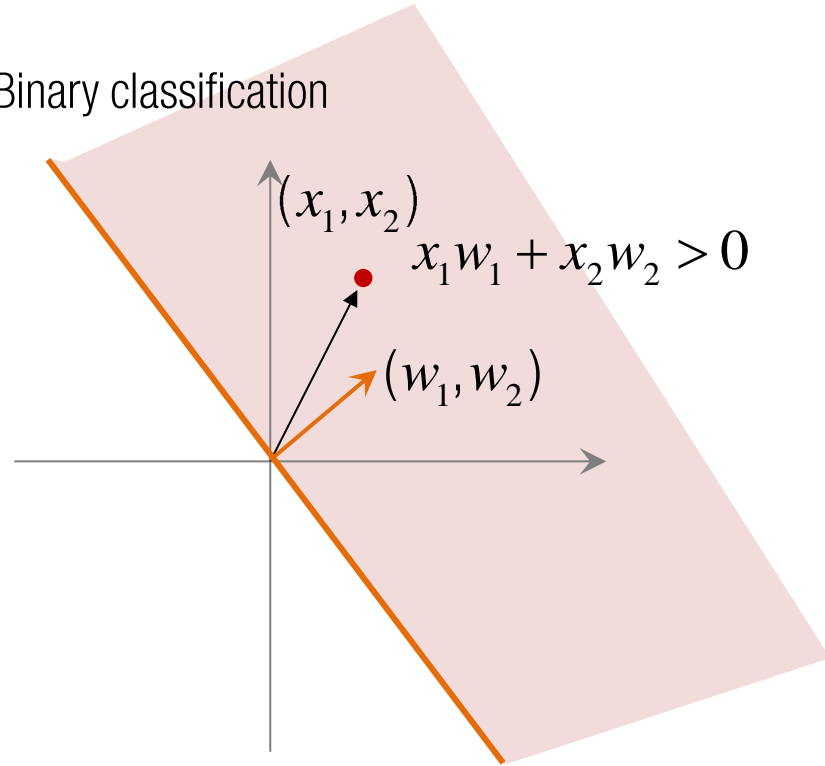


# PERCEPTRON

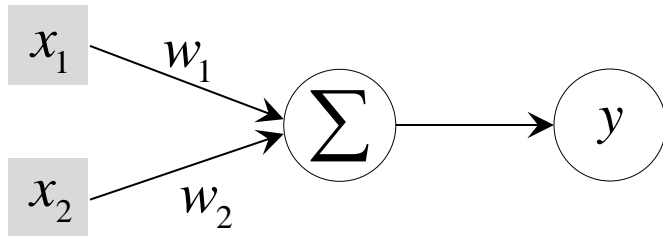


$$y = x_1 w_1 + x_2 w_2$$

Binary classification

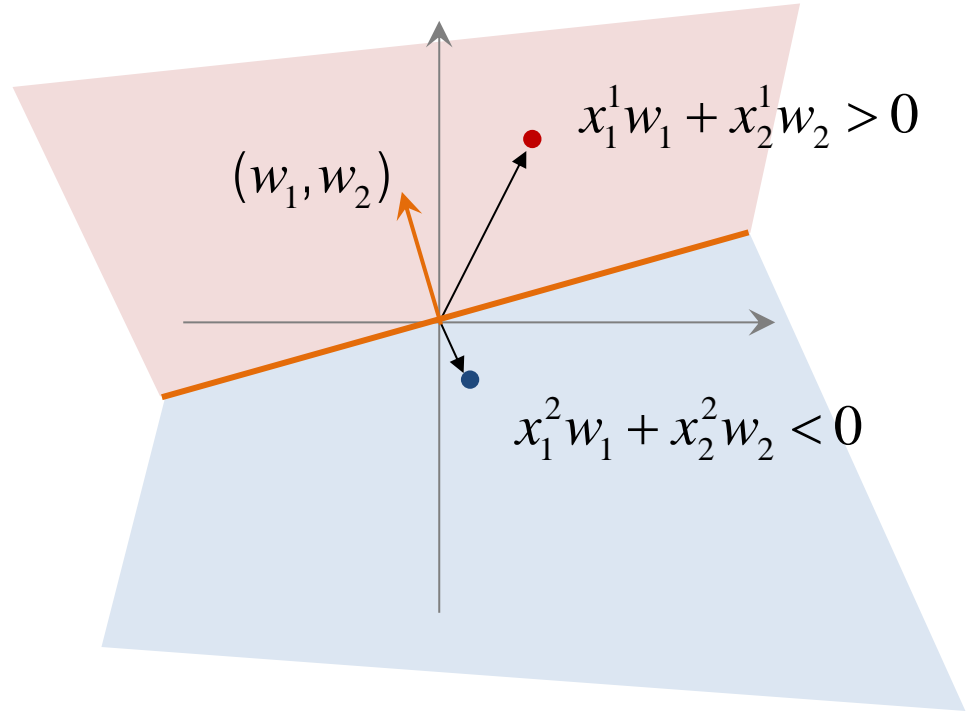


# PERCEPTRON

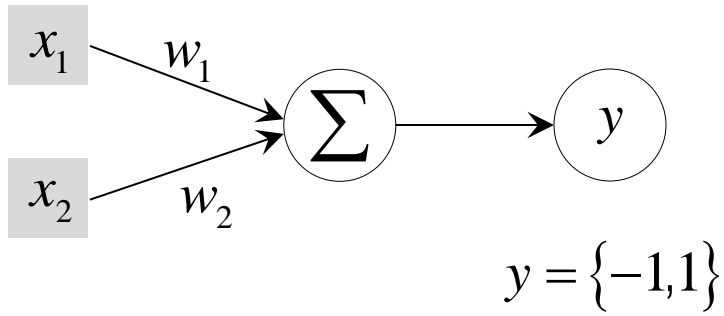


$$y = x_1 w_1 + x_2 w_2$$

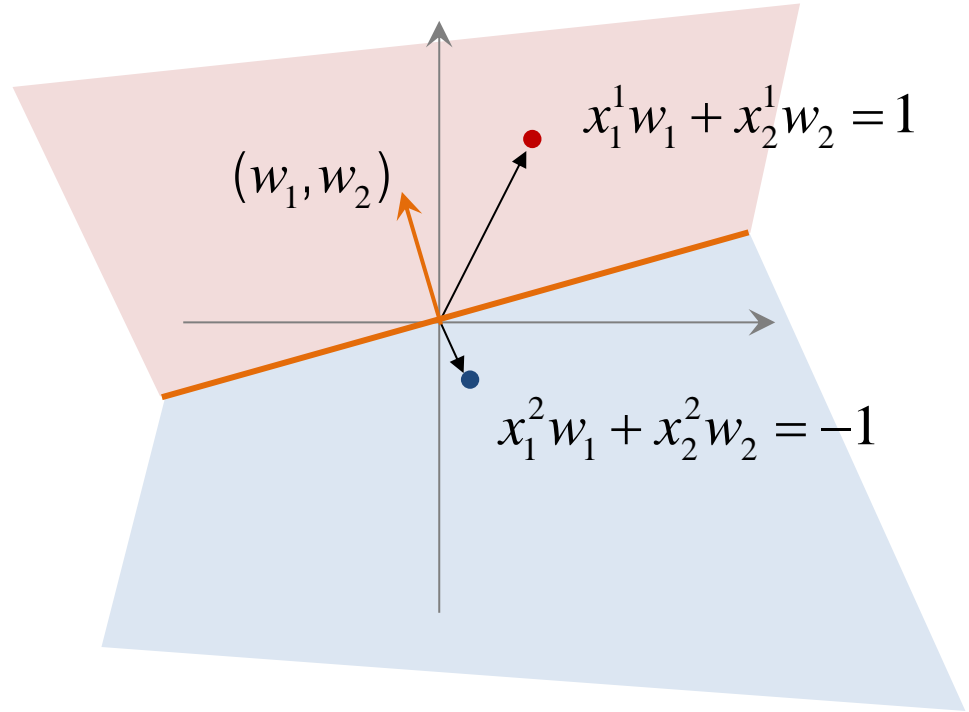
Binary classification



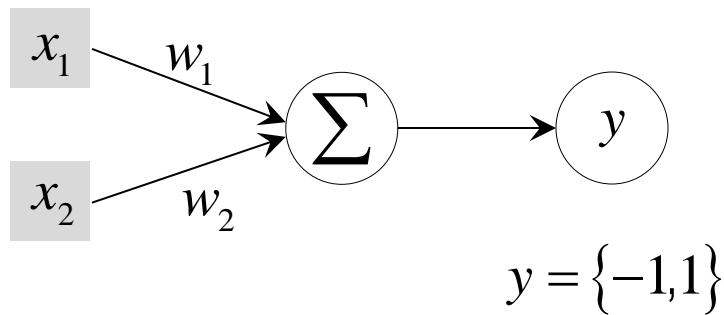
# PERCEPTRON



Binary classification



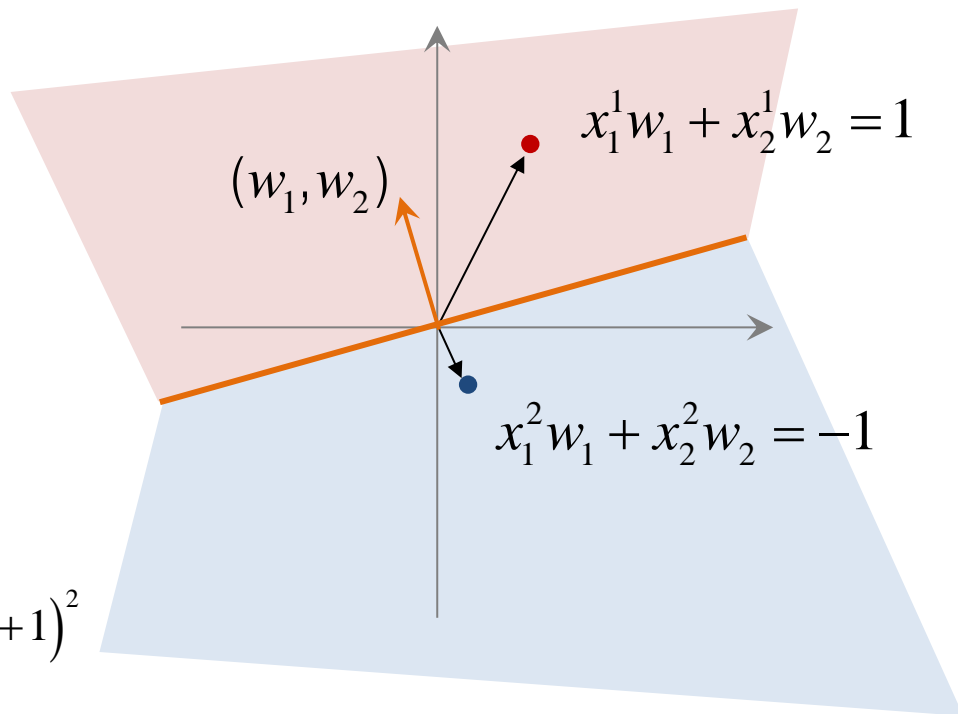
# PERCEPTRON



minimize  $L(w_1, w_2)$   
 $w_1, w_2$

$$L(w_1, w_2) = (x_1^1 w_1 + x_2^1 w_2 - 1)^2 + (x_1^2 w_1 + x_2^2 w_2 + 1)^2$$

Binary classification



# RECALL: IMAGE ALIGNMENT OBJECTIVE

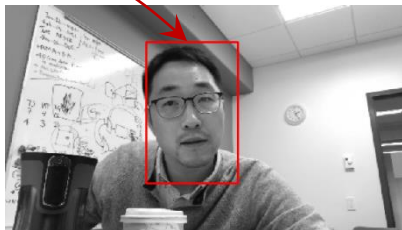
$$W(x; p)$$



$T(x)$



$I(W(x; p))$



$I(x)$

$$p^* = \underset{p}{\text{minimize}} \sum_x (I(W(x; p)) - T(x))^2$$

## Guass-Newton's method

1. Linearize the obj. function at  $p$

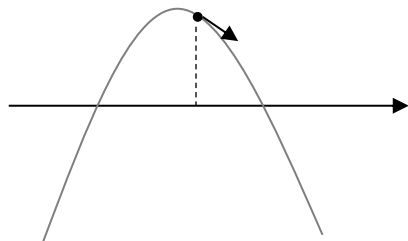
$$I(W(x; p + \Delta p)) \approx I(W(x; p)) + \frac{\partial I}{\partial p} \Delta p$$

Ex) affine transform

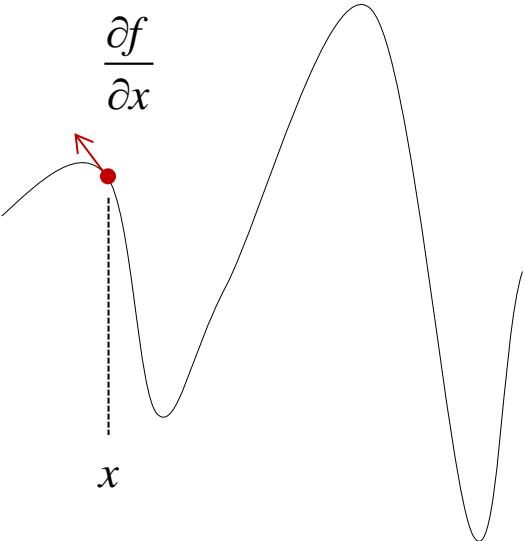
$$W(x; p) = \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} p_1 u + p_2 v + p_3 \\ p_4 u + p_5 v + p_6 \end{bmatrix}$$

$$\rightarrow \frac{\partial I}{\partial p} \Delta p = \frac{\partial I}{\partial x} \frac{\partial W}{\partial p} \Delta p = \nabla I \frac{\partial W}{\partial p} \Delta p$$

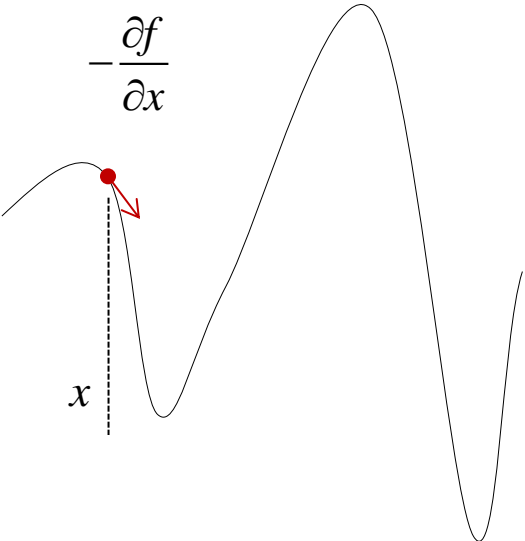
$$\text{Jacobian: } \frac{\partial W}{\partial p} = \begin{bmatrix} \frac{\partial u}{\partial p_1} & \dots & \frac{\partial u}{\partial p_6} \\ \frac{\partial v}{\partial p_1} & \dots & \frac{\partial v}{\partial p_6} \end{bmatrix}$$



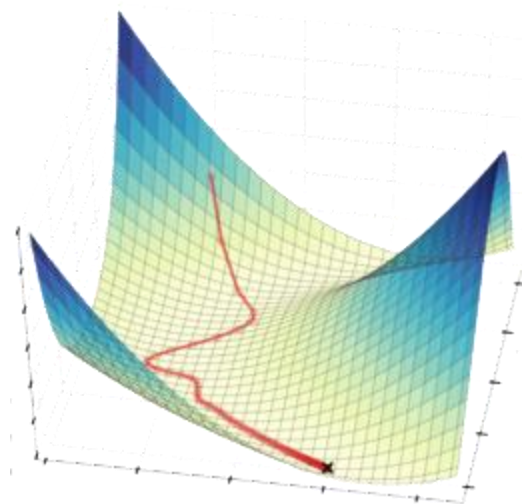
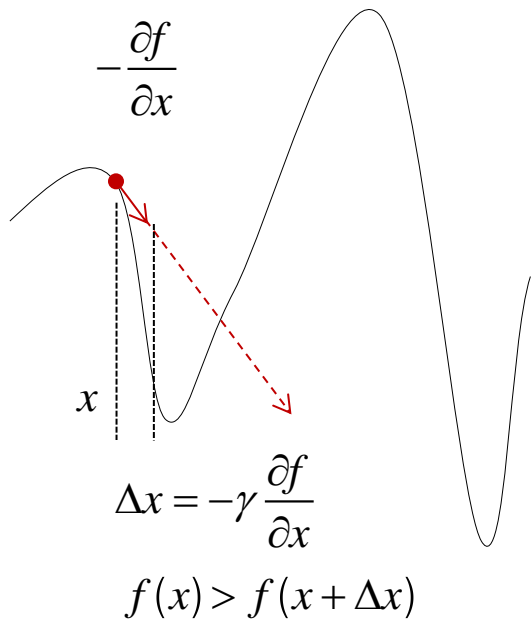
# *GRADIENT DESCENT*



# *GRADIENT DESCENT*



# GRADIENT DESCENT

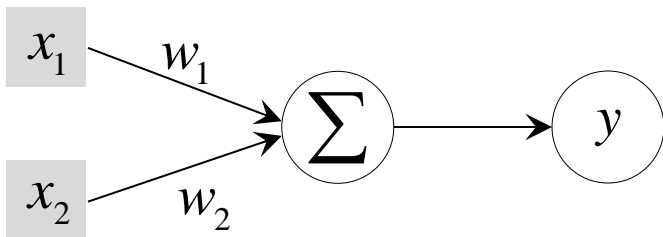


minimize  $f(x)$

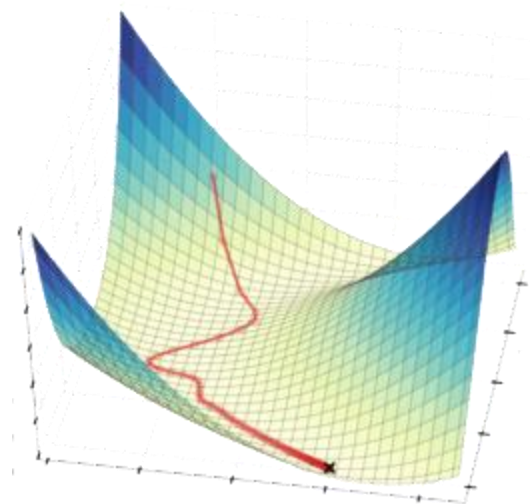
$$x \leftarrow x - \gamma \frac{\partial f}{\partial x}$$



# GRADIENT DESCENT



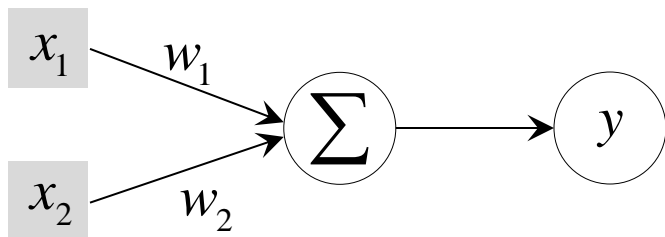
$$L(w_1, w_2) = (x_1^1 w_1 + x_2^1 w_2 - 1)^2 + (x_1^2 w_1 + x_2^2 w_2 + 1)^2$$



minimize  $f(x)$   
 $x$

$$x \leftarrow x - \gamma \frac{\partial f}{\partial x}$$

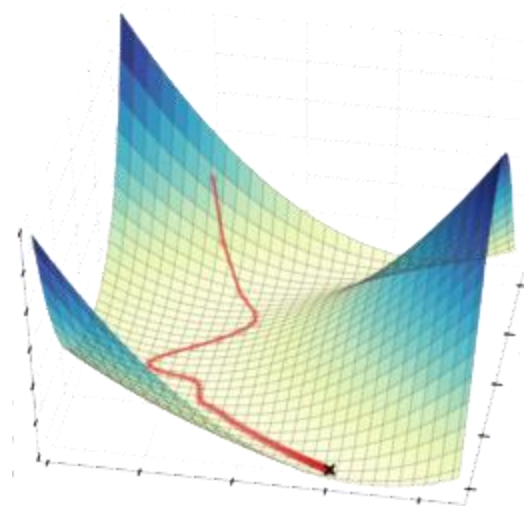
# GRADIENT DESCENT



$$L(w_1, w_2) = (x_1^1 w_1 + x_2^1 w_2 - 1)^2 + (x_1^2 w_1 + x_2^2 w_2 + 1)^2$$

Weight update rule:

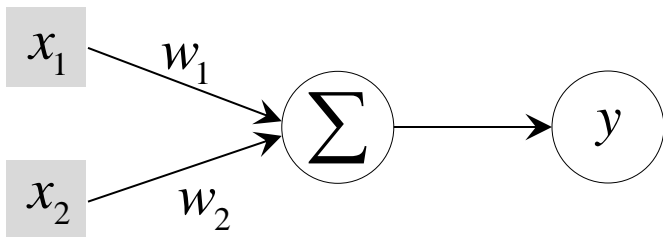
$$w_1 = w_1 - \gamma \frac{\partial L}{\partial w_1} \quad w_2 = w_2 - \gamma \frac{\partial L}{\partial w_2}$$



minimize  $f(x)$

$$x \leftarrow x - \gamma \frac{\partial f}{\partial x}$$

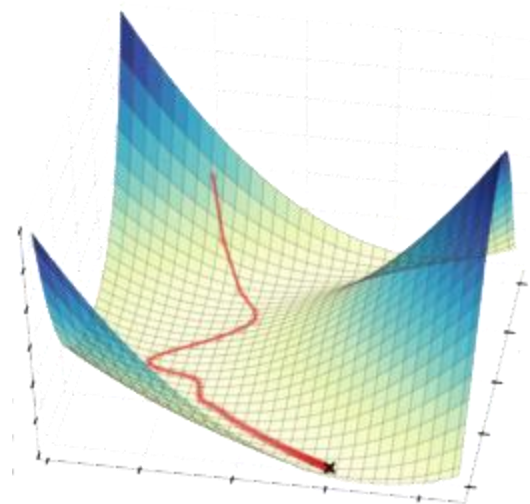
# GRADIENT DESCENT



$$L(w_1, w_2) = (x_1^1 w_1 + x_2^1 w_2 - y^1)^2 + (x_1^2 w_1 + x_2^2 w_2 - y^2)^2$$

Weight update rule:

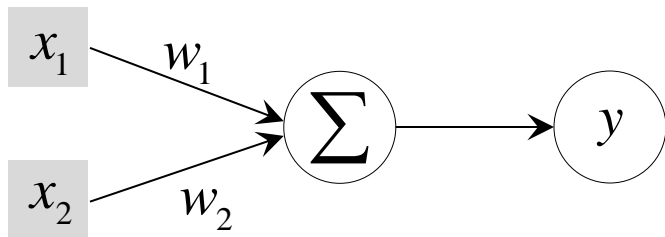
$$w_1 = w_1 - \gamma \frac{\partial L}{\partial w_1} \quad w_2 = w_2 - \gamma \frac{\partial L}{\partial w_2}$$



minimize  $f(x)$

$$x \leftarrow x - \gamma \frac{\partial f}{\partial x}$$

# GRADIENT DESCENT



$$L(w_1, w_2) = (x_1^1 w_1 + x_2^1 w_2 - y^1)^2 + (x_1^2 w_1 + x_2^2 w_2 - y^2)^2$$

Weight update rule:

$$w_1 = w_1 - \gamma \frac{\partial L}{\partial w_1} \quad w_2 = w_2 - \gamma \frac{\partial L}{\partial w_2}$$

Prediction:

$$\tilde{y} = x_1 w_1 + x_2 w_2$$

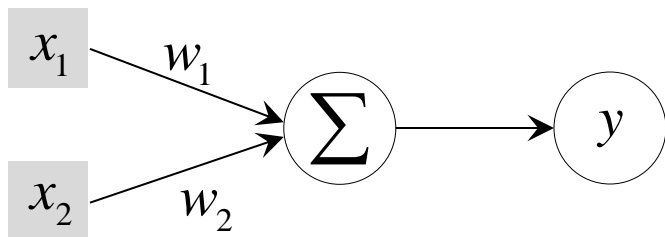
Error for each data:

$$L(w_1, w_2) = (\tilde{y} - y)^2$$

Gradient:

$$\frac{\partial L}{\partial w_1} =$$

# GRADIENT DESCENT



$$L(w_1, w_2) = (x_1^1 w_1 + x_2^1 w_2 - y^1)^2 + (x_1^2 w_1 + x_2^2 w_2 - y^2)^2$$

Weight update rule:

$$w_1 = w_1 - \gamma \frac{\partial L}{\partial w_1} \quad w_2 = w_2 - \gamma \frac{\partial L}{\partial w_2}$$

Prediction:

$$\tilde{y} = x_1 w_1 + x_2 w_2$$

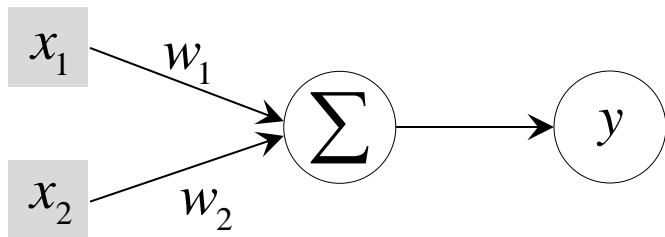
Error for each data:

$$L(w_1, w_2) = (\tilde{y} - y)^2$$

Gradient:

$$\begin{aligned} \frac{\partial L}{\partial w_1} &= (\tilde{y} - y) \frac{\partial \hat{y}}{\partial w_1} \\ &= (\tilde{y} - y) x_1 \end{aligned}$$

# GRADIENT DESCENT



$$L(w_1, w_2) = (x_1^1 w_1 + x_2^1 w_2 - y^1)^2 + (x_1^2 w_1 + x_2^2 w_2 - y^2)^2$$

Weight update rule:

$$w_1 = w_1 - \gamma(\tilde{y}^1 - y^1)x_1^1 - \gamma(\tilde{y}^2 - y^2)x_1^2$$

$$w_2 = w_2 - \gamma(\tilde{y}^1 - y^1)x_2^1 - \gamma(\tilde{y}^2 - y^2)x_2^2$$

Prediction:

$$\tilde{y} = x_1 w_1 + x_2 w_2$$

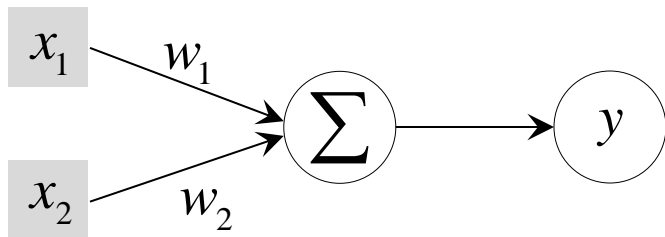
Error for each data:

$$L(w_1, w_2) = (\tilde{y} - y)^2$$

Gradient:

$$\begin{aligned} \frac{\partial L}{\partial w_1} &= (\tilde{y} - y) \frac{\partial \tilde{y}}{\partial w_1} \\ &= (\tilde{y} - y) x_1 \end{aligned}$$

# GRADIENT DESCENT

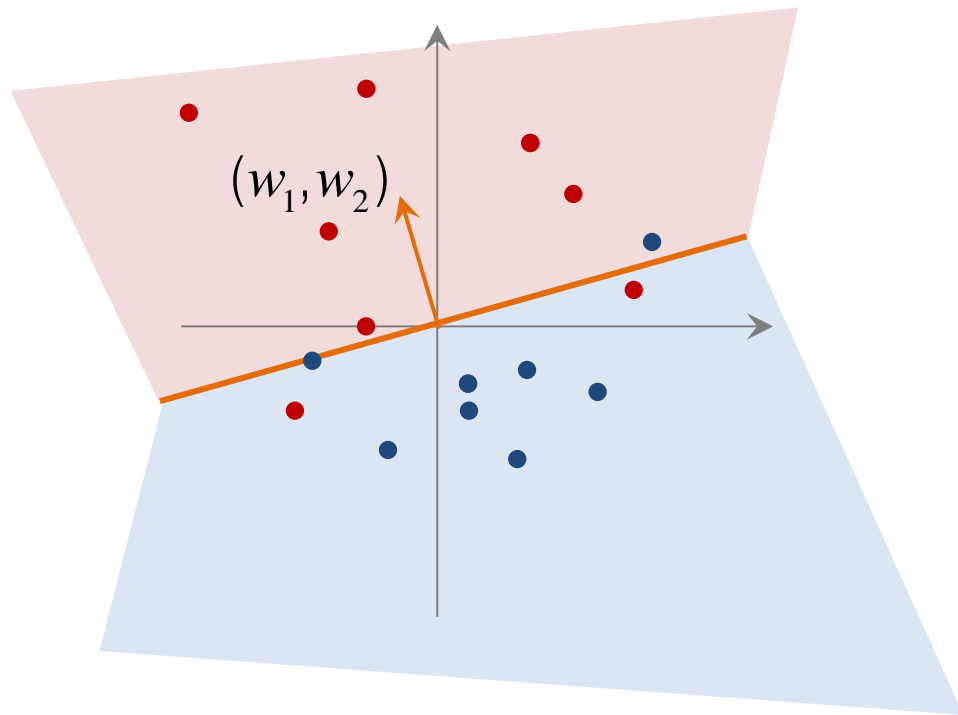


$$L(w_1, w_2) = \sum_i (x_1^i w_1 + x_2^i w_2 - y^i)^2$$

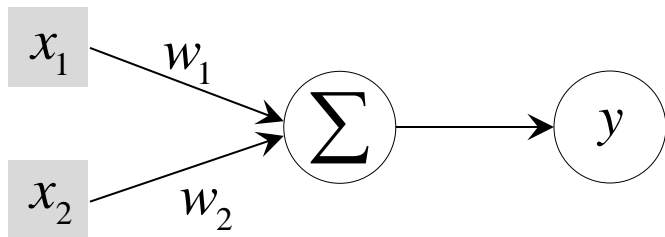
Weight update rule:

$$w_1 = w_1 - \gamma \frac{\partial L}{\partial w_1} \quad w_2 = w_2 - \gamma \frac{\partial L}{\partial w_2}$$

Binary classification



# GRADIENT DESCENT



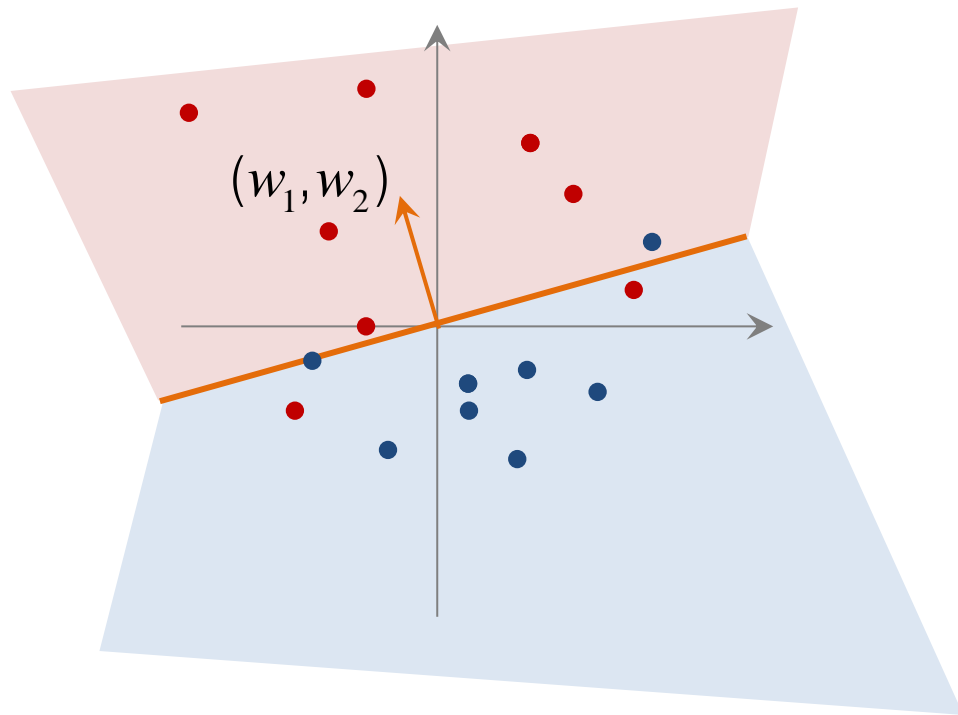
$$L(w_1, w_2) = \sum_i (x_1^i w_1 + x_2^i w_2 - y^i)^2$$

Weight update rule:

$$w_1 = w_1 - \frac{\gamma}{n} \sum_i (\tilde{y}^i - y^i) x_1^i$$

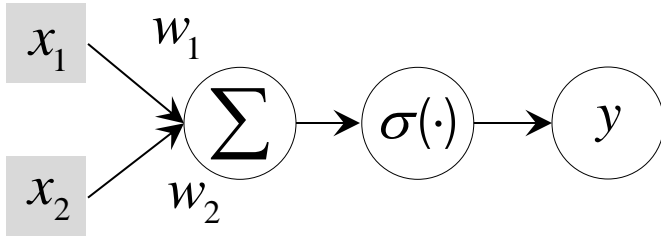
$$w_2 = w_2 - \frac{\gamma}{n} \sum_i (\tilde{y}^i - y^i) x_2^i$$

Binary classification





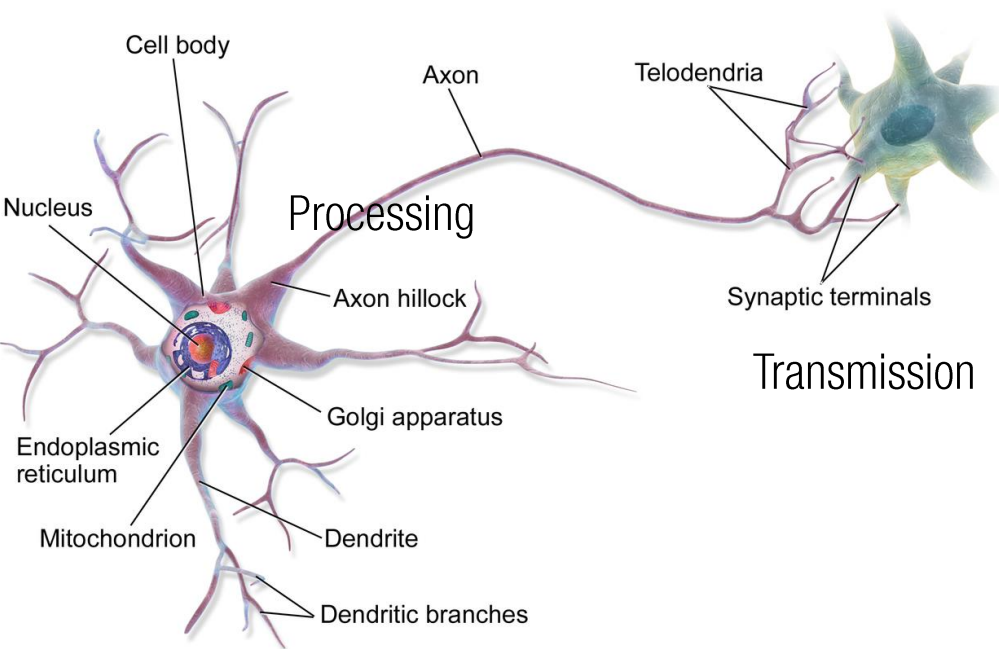
# ACTIVATION



$$y = f(a)$$

$$a = x_1 w_1 + x_2 w_2$$

# NEURON



Nucleus

Cell body

Axon

Telodendria

Processing

Axon hillock

Synaptic terminals

Transmission

Endoplasmic reticulum

Golgi apparatus

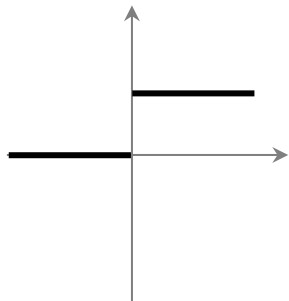
Mitochondrion

Dendrite

Dendritic branches

Signal reception

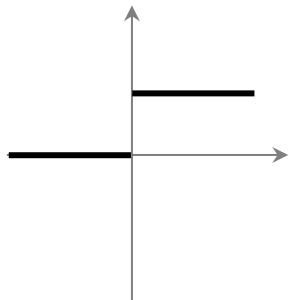
# *ACTIVATION*



Step

$$\sigma(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

# ACTIVATION



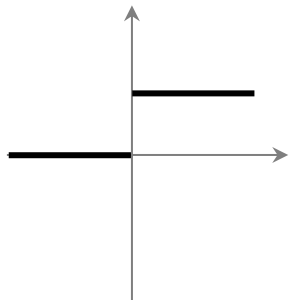
Step

$$\sigma(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

$$w_1 = w_1 - \gamma \frac{\partial L}{\partial w_1}$$

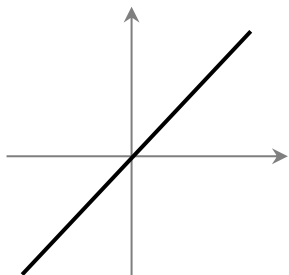
$$w_2 = w_2 - \gamma \frac{\partial L}{\partial w_2}$$

# ACTIVATION



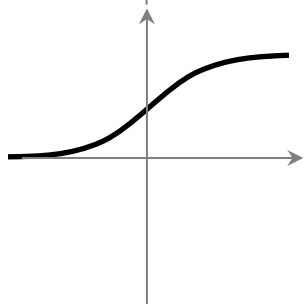
Step

$$\sigma(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$



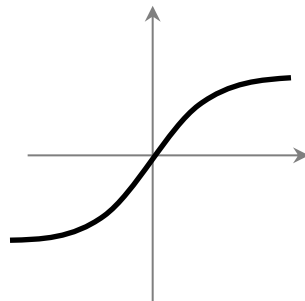
Linear

$$\sigma(x) = ax$$



Sigmoid

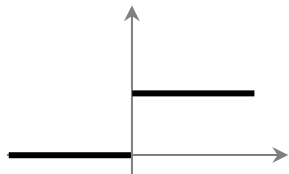
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



Tanh

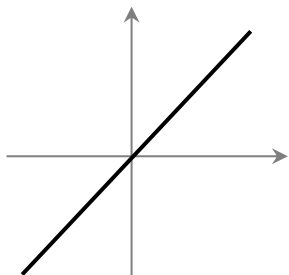
$$\sigma(x) = \tanh(x)$$

# ACTIVATION



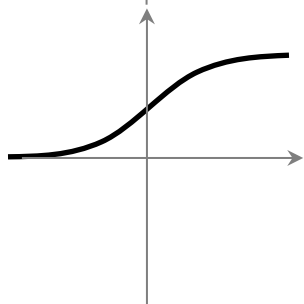
Step

$$\sigma(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$



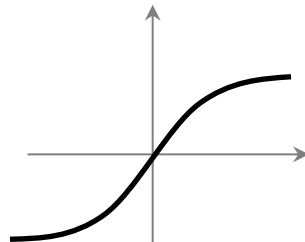
Linear

$$\sigma(x) = ax$$



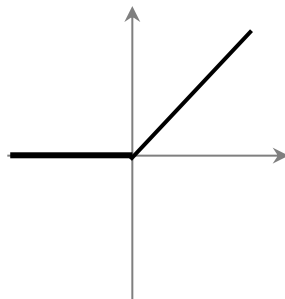
Sigmoid

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



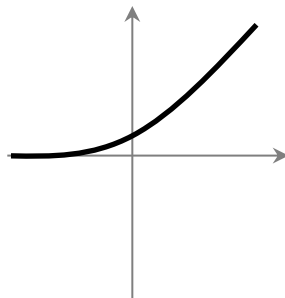
Tanh

$$\sigma(x) = \tanh(x)$$



Rect. Linear (ReLU)

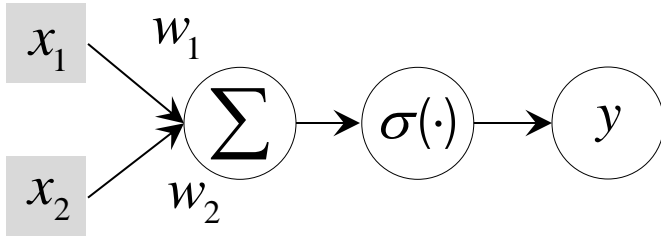
$$\sigma(x) = \max(0, x)$$



Soft Rect. Linear

$$\sigma(x) = \log(1 + e^x)$$

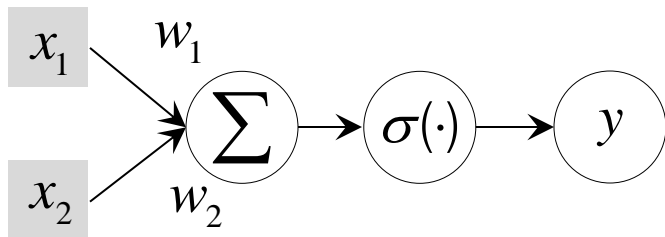
# ACTIVATION



$$f = \sigma(a)$$

$$a = x_1 w_1 + x_2 w_2$$

# ACTIVATION



$$f = \sigma(a)$$

$$a = x_1 w_1 + x_2 w_2$$

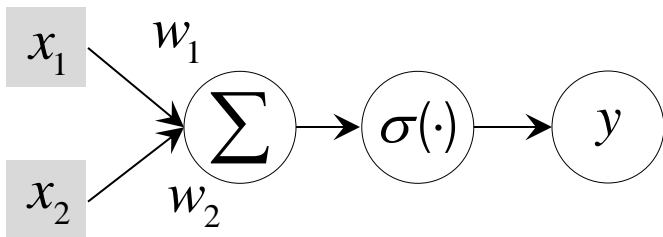
$$L(w_1, w_2) = \sum_i \left( f(x_1^i w_1 + x_2^i w_2) - y^i \right)^2$$

Weight update rule:

$$w_1 = w_1 - \gamma \frac{\partial L}{\partial w_1} \quad w_2 = w_2 - \gamma \frac{\partial L}{\partial w_2}$$



# ACTIVATION



$$f = \sigma(a)$$

$$a = x_1 w_1 + x_2 w_2$$

$$L(w_1, w_2) = \sum_i (f(x_1^i w_1 + x_2^i w_2) - y^i)^2$$

Weight update rule:

$$w_1 = w_1 - \gamma \frac{\partial L}{\partial w_1} \quad w_2 = w_2 - \gamma \frac{\partial L}{\partial w_2}$$

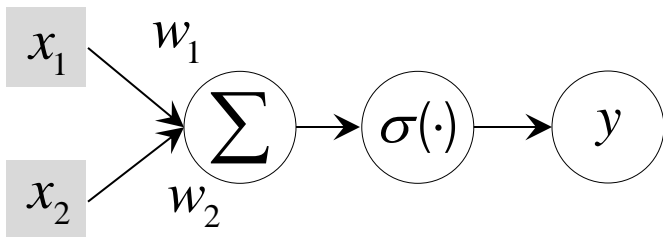
$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial a} \frac{\partial a}{\partial w} \quad \text{: chain rule}$$

Sigmoid activation

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

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

# ACTIVATION



$$f = \sigma(a)$$

$$a = x_1 w_1 + x_2 w_2$$

$$L(w_1, w_2) = \sum_i (f(x_1^i w_1 + x_2^i w_2) - y^i)^2$$

Weight update rule:

$$w_1 = w_1 - \gamma \frac{\partial L}{\partial w_1} \quad w_2 = w_2 - \gamma \frac{\partial L}{\partial w_2}$$

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial a} \frac{\partial a}{\partial w} \quad : \text{chain rule}$$

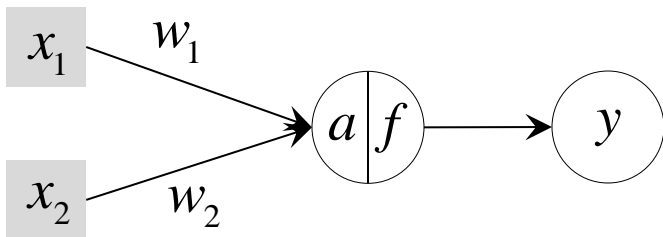
$$\begin{aligned} &= (\tilde{y} - y) \frac{\partial f}{\partial a} \frac{\partial a}{\partial w} \\ &= (\tilde{y} - y) f(1 - f)x \end{aligned}$$

Sigmoid activation

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

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

# ACTIVATION



$$f = \sigma(a)$$

$$a = x_1 w_1 + x_2 w_2$$

$$L(w_1, w_2) = \sum_i (f(x_1^i w_1 + x_2^i w_2) - y^i)^2$$

Weight update rule:

$$w_1 = w_1 - \gamma \frac{\partial L}{\partial w_1} \quad w_2 = w_2 - \gamma \frac{\partial L}{\partial w_2}$$

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial a} \frac{\partial a}{\partial w} \quad : \text{chain rule}$$

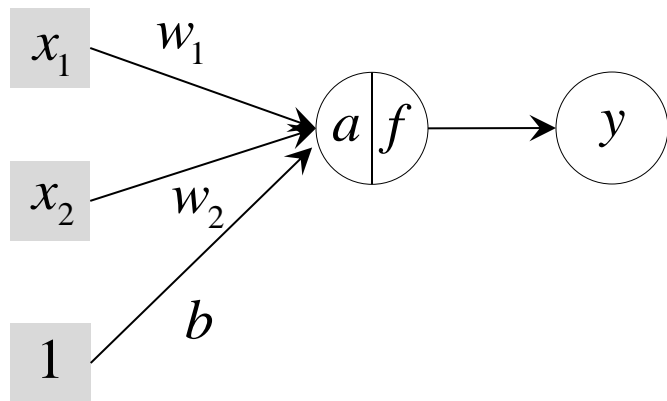
$$\begin{aligned} &= (\tilde{y} - y) \frac{\partial f}{\partial a} \frac{\partial a}{\partial w} \\ &= (\tilde{y} - y) f(1 - f)x \end{aligned}$$

Sigmoid activation

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

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

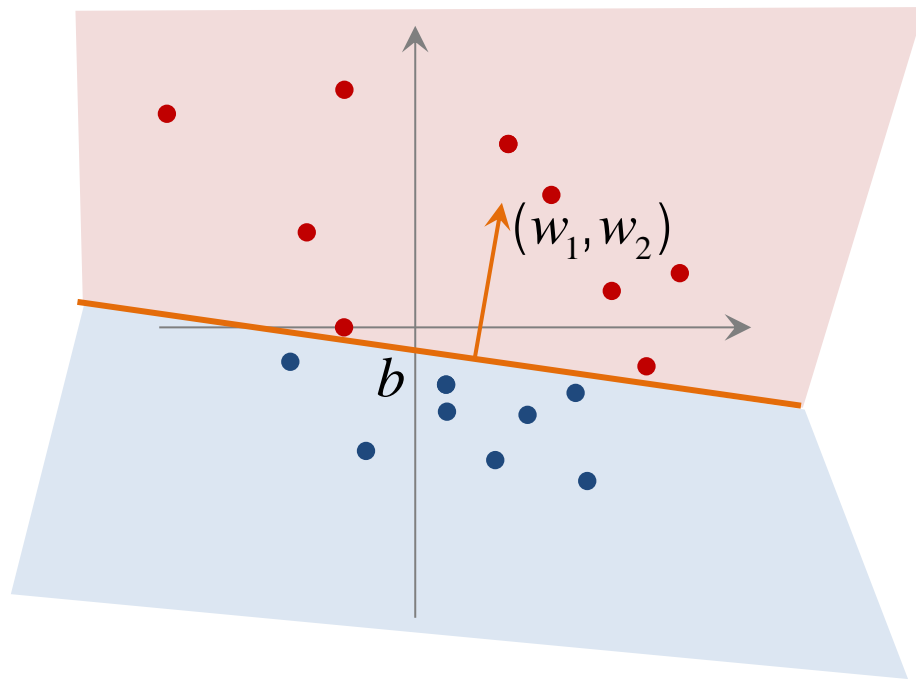
# BIAS



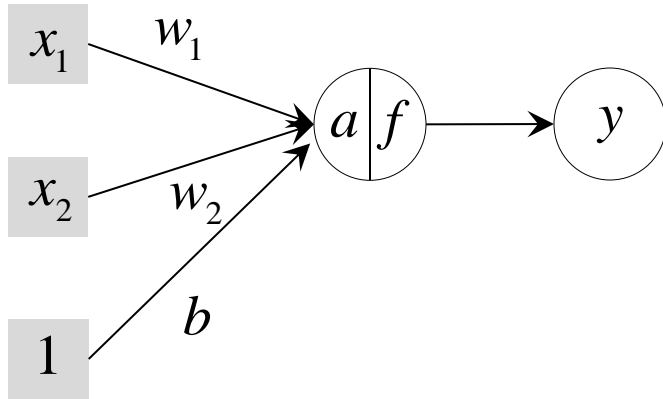
$$f = \sigma(a)$$

$$a = x_1 w_1 + x_2 w_2 + 1 \cdot b$$

Binary classification



# BIAS



$$f = \sigma(a)$$

$$a = x_1 w_1 + x_2 w_2 + 1 \cdot b$$

$$L(w_1, w_2) = \sum_i (f(x_1^i w_1 + x_2^i w_2) - y^i)^2$$

Weight update rule:

$$w_1 = w_1 - \gamma \frac{\partial L}{\partial w_1} \quad w_2 = w_2 - \gamma \frac{\partial L}{\partial w_2}$$

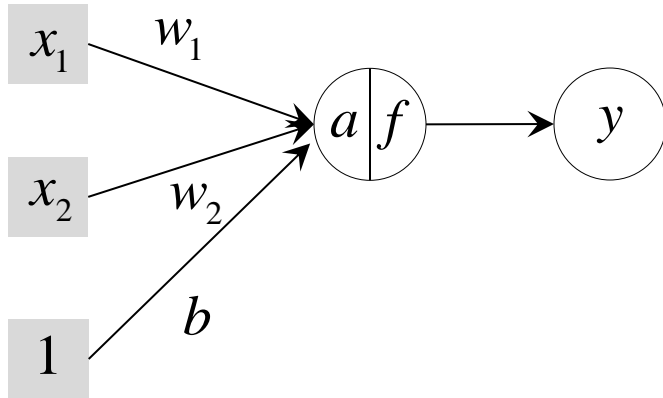
$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial a} \frac{\partial a}{\partial b} \quad \text{: chain rule}$$

Sigmoid activation

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

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

# BIAS



$$f = \sigma(a)$$

$$a = x_1 w_1 + x_2 w_2 + 1 \cdot b$$

$$L(w_1, w_2) = \sum_i (f(x_1^i w_1 + x_2^i w_2) - y^i)^2$$

Weight update rule:

$$w_1 = w_1 - \gamma \frac{\partial L}{\partial w_1} \quad w_2 = w_2 - \gamma \frac{\partial L}{\partial w_2}$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial a} \frac{\partial a}{\partial b} \quad \text{: chain rule}$$

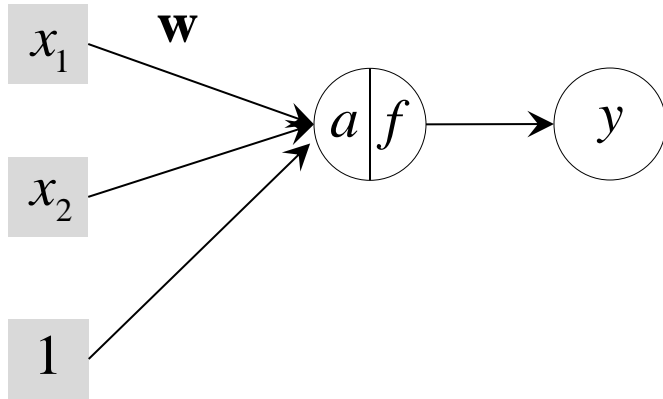
$$\begin{aligned} &= (\tilde{y} - y) \frac{\partial f}{\partial a} \frac{\partial a}{\partial b} \\ &= (\tilde{y} - y) f(1 - f) \end{aligned}$$

Sigmoid activation

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

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

# EXPRESSION



$$f = \sigma(\mathbf{w} \cdot \mathbf{x})$$

$$L(w_1, w_2) = \sum_i (f(x_1^i w_1 + x_2^i w_2) - y^i)^2$$

Weight update rule:

$$w_1 = w_1 - \gamma \frac{\partial L}{\partial w_1} \quad w_2 = w_2 - \gamma \frac{\partial L}{\partial w_2}$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial a} \frac{\partial a}{\partial b} \quad : \text{chain rule}$$

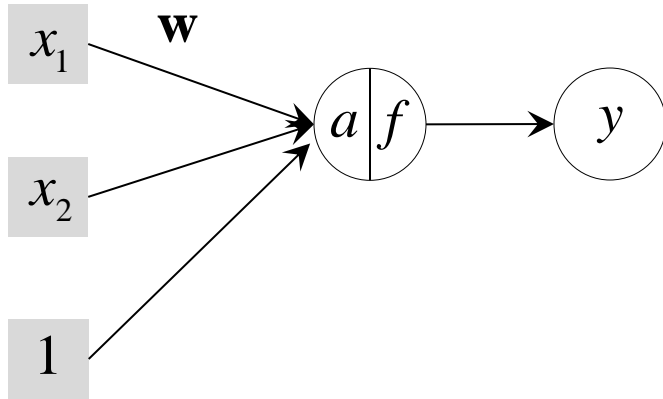
$$\begin{aligned} &= (\tilde{y} - y) \frac{\partial f}{\partial a} \frac{\partial a}{\partial b} \\ &= (\tilde{y} - y) f(1 - f) \end{aligned}$$

Sigmoid activation

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

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

# EXPRESSION

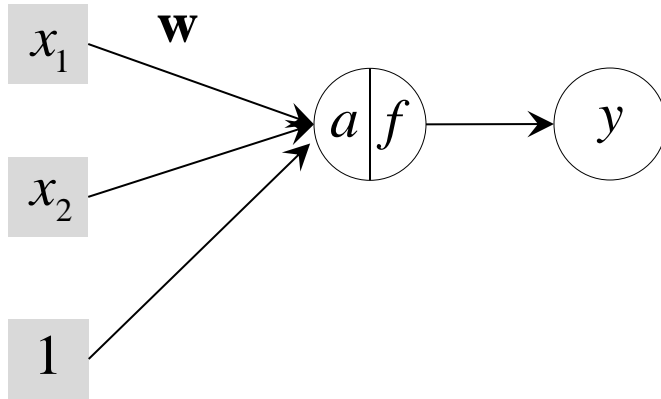


$$f = \sigma(\mathbf{w} \cdot \mathbf{x})$$

How many variables?



# EXPRESSION



$$f = \sigma(\mathbf{w} \cdot \mathbf{x})$$

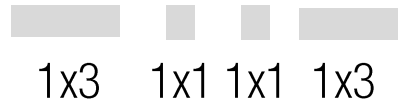
How many variables?

$n+1$

$$\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial a} \frac{\partial a}{\partial \mathbf{w}}$$

$$f = \sigma(a)$$

$$a = x_1 w_1 + x_2 w_2 + 1 \cdot b$$



# MNIST DIGIT RECOGNITION

0000000000000000  
1111111111111111  
2222222222222222  
3333333333333333  
4444444444444444  
5555555555555555  
6666666666666666  
7777777777777777  
8888888888888888  
9999999999999999

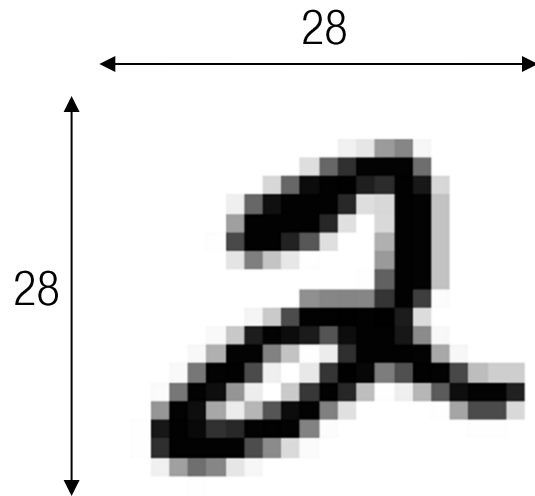
Data & Labels



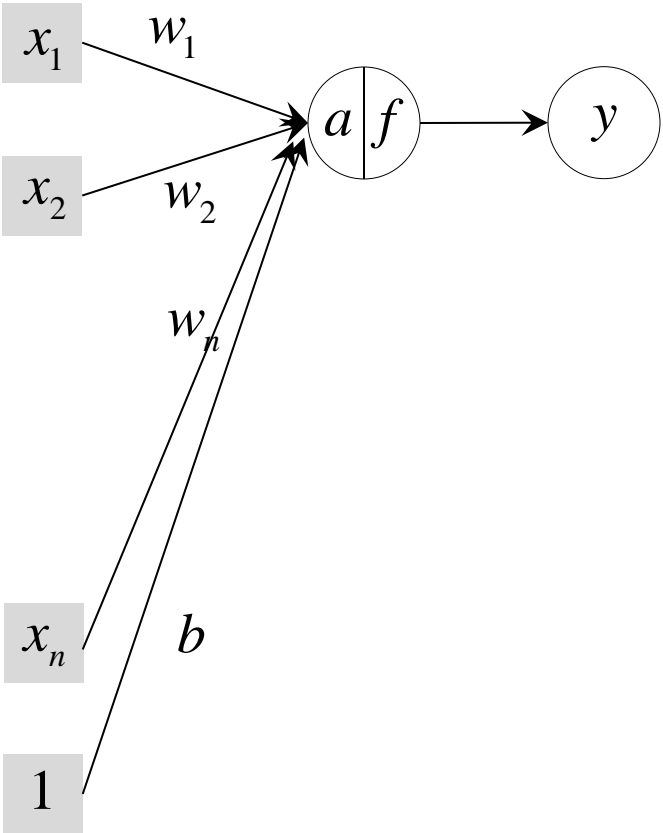
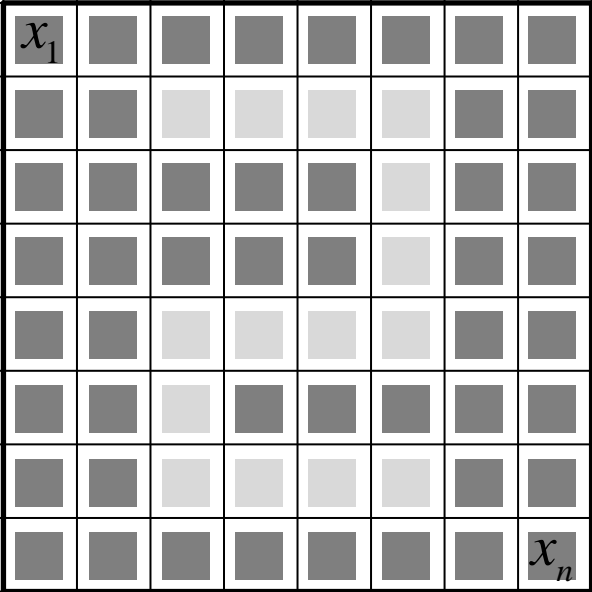
0  
1  
2  
3  
4  
5  
6  
7  
8  
9

Data

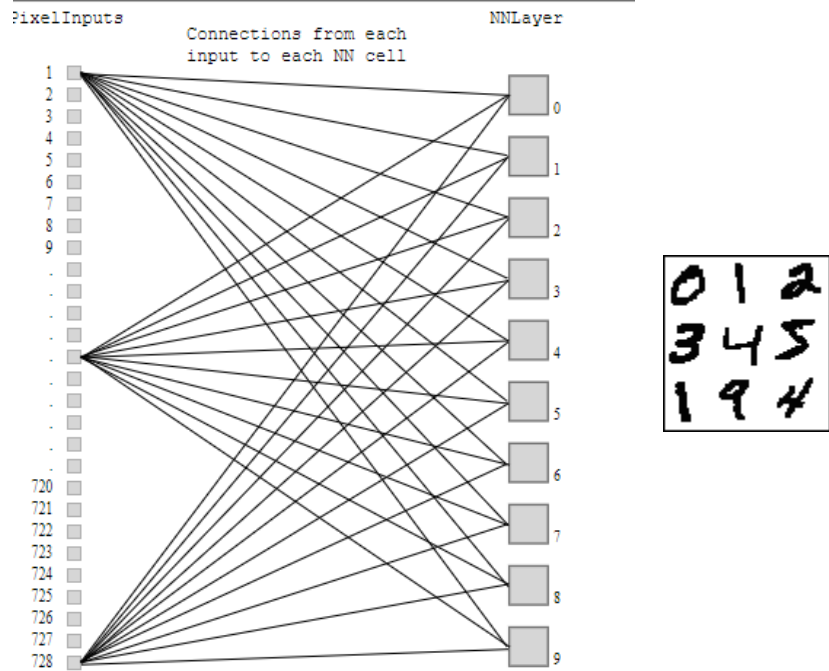
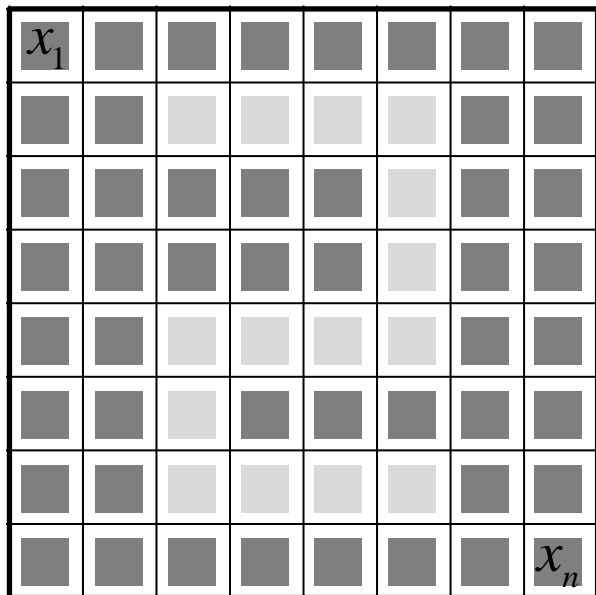
- 60,000 training
- 10,000 testing



# ***DIGIT RECOGNITION***



# ***DIGIT RECOGNITION***

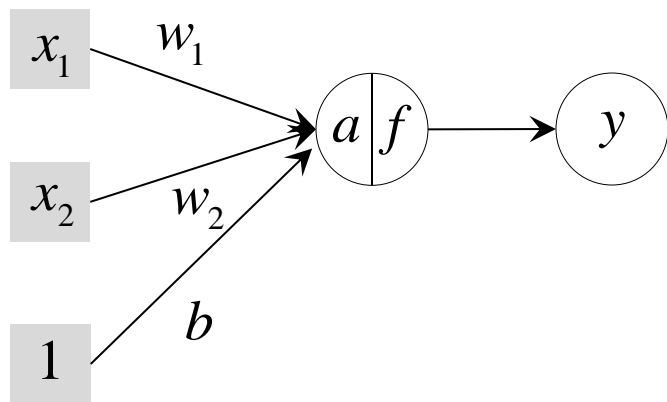


Training error: 99.97%

Testing error: 85% (88% by LeCunn)

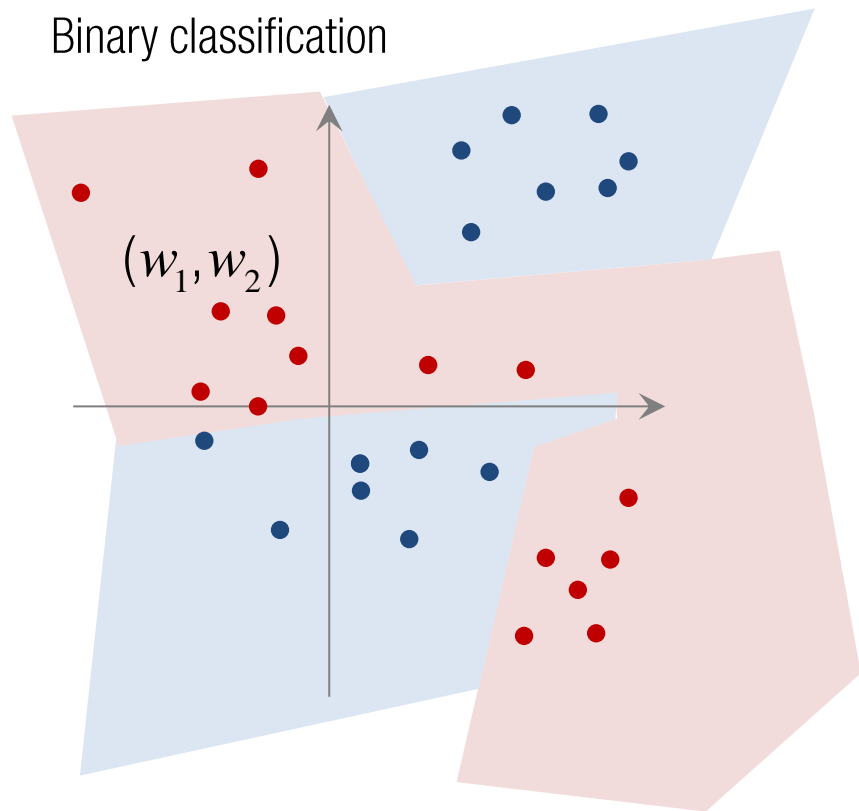
Can we do better?

# *LINEARLY NON-SEPARABLE DATA*

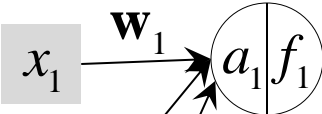


$$f = \sigma(\mathbf{w} \cdot \mathbf{x})$$

Binary classification



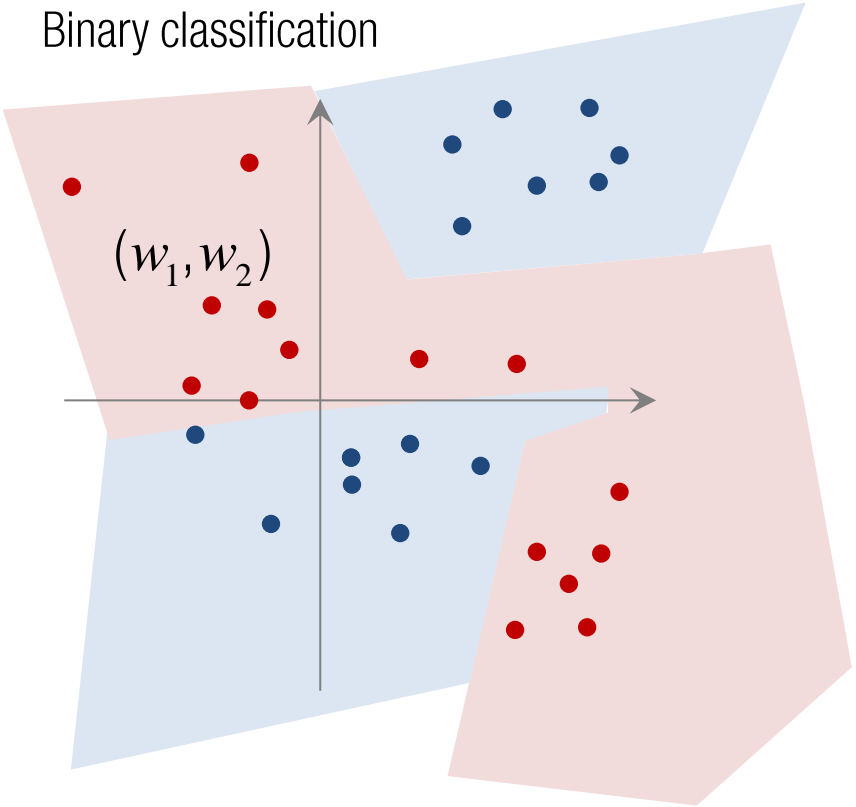
# ***MULTI-LAYER PERCEPTRON***



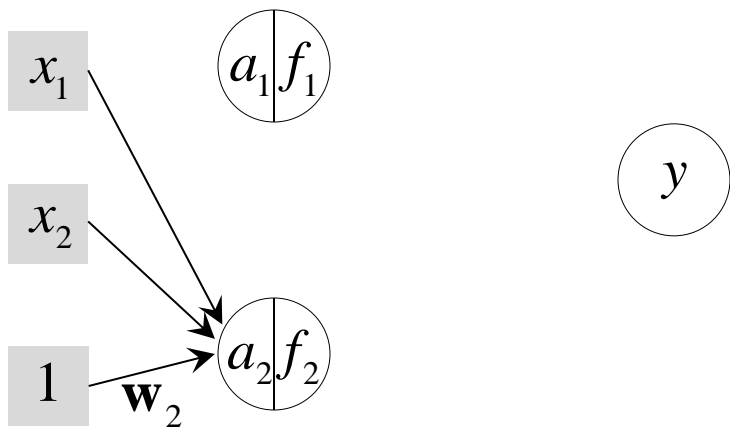
$y$

$$f_1 = \sigma(\mathbf{w}_1 \cdot \mathbf{x})$$

Binary classification



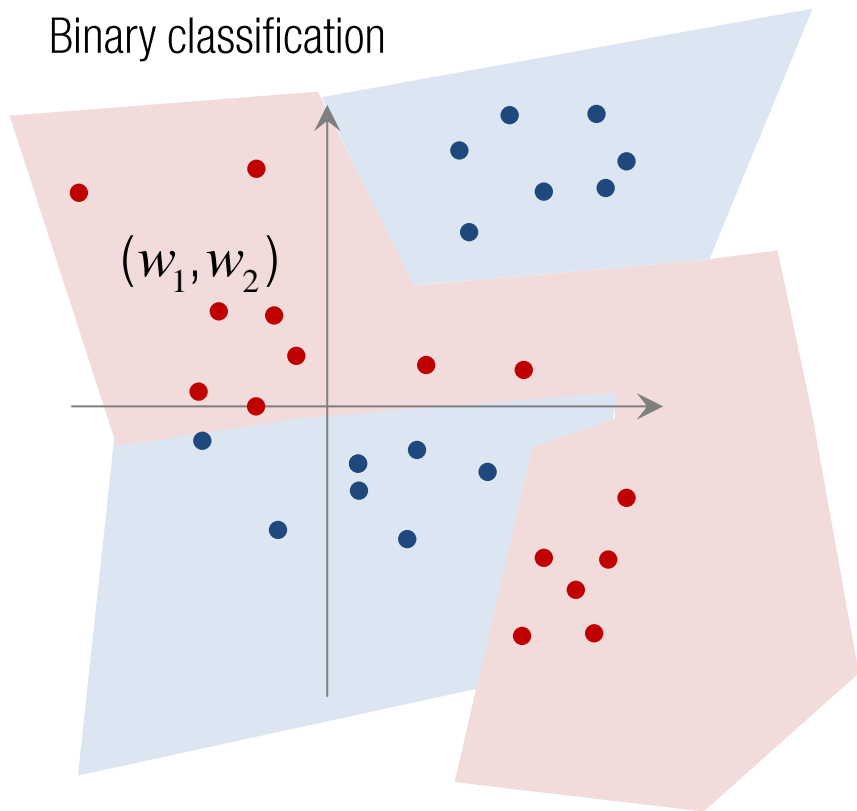
# MULTI-LAYER PERCEPTRON



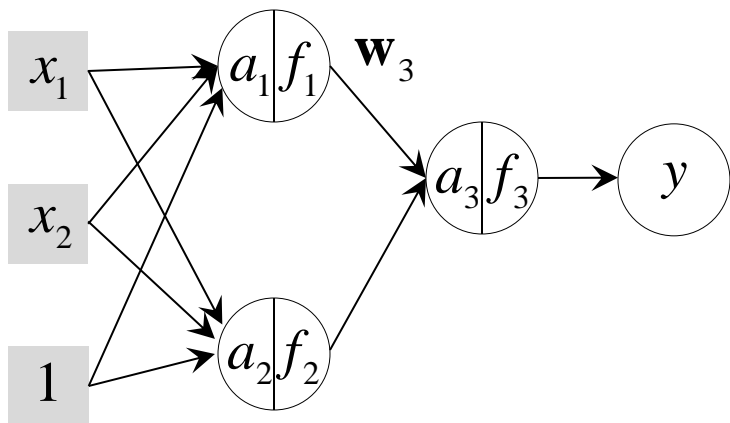
$$f_1 = \sigma(\mathbf{w}_1 \cdot \mathbf{x})$$

$$f_2 = \sigma(\mathbf{w}_2 \cdot \mathbf{x})$$

Binary classification



# MULTI-LAYER PERCEPTRON

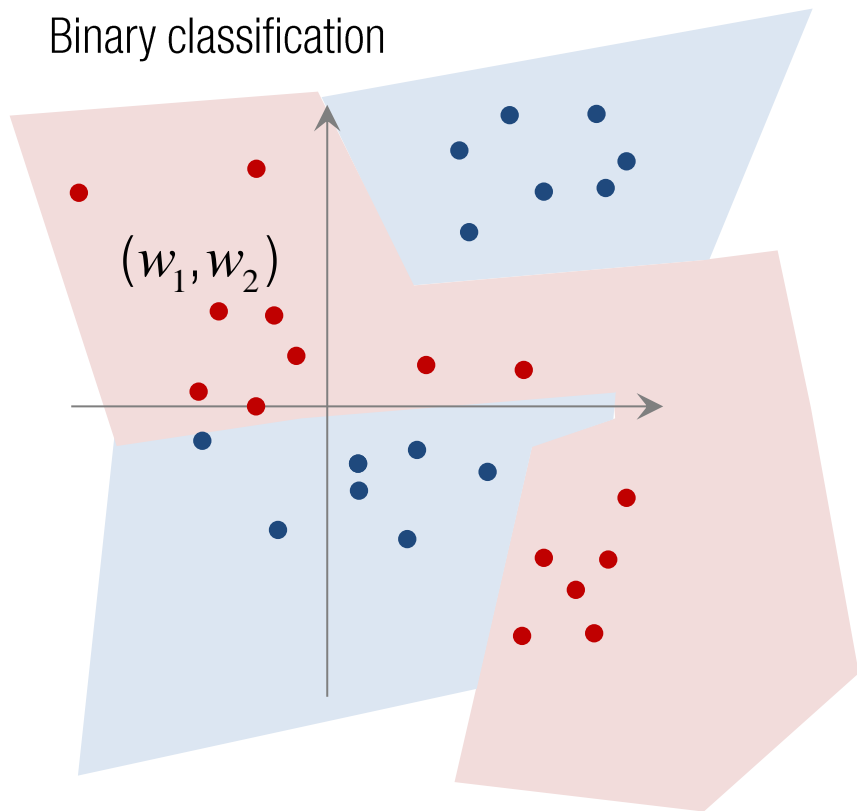


$$f_1 = \sigma(\mathbf{w}_1 \cdot \mathbf{x})$$

$$f_2 = \sigma(\mathbf{w}_2 \cdot \mathbf{x})$$

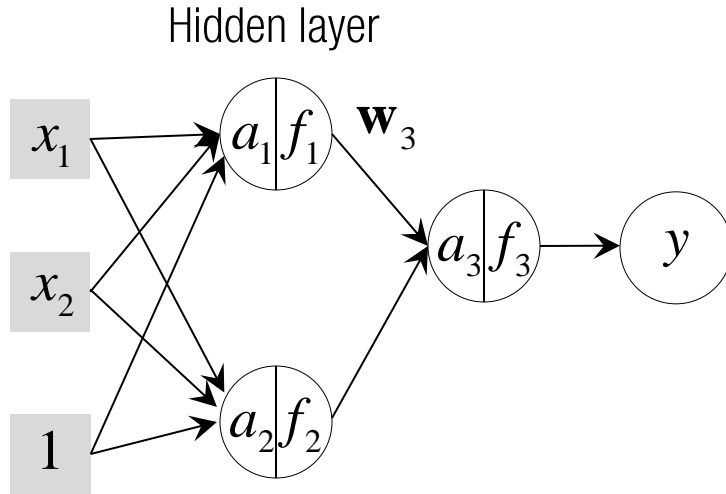
$$f_3 = \sigma\left(\mathbf{w}_3 \cdot \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}\right)$$

Binary classification





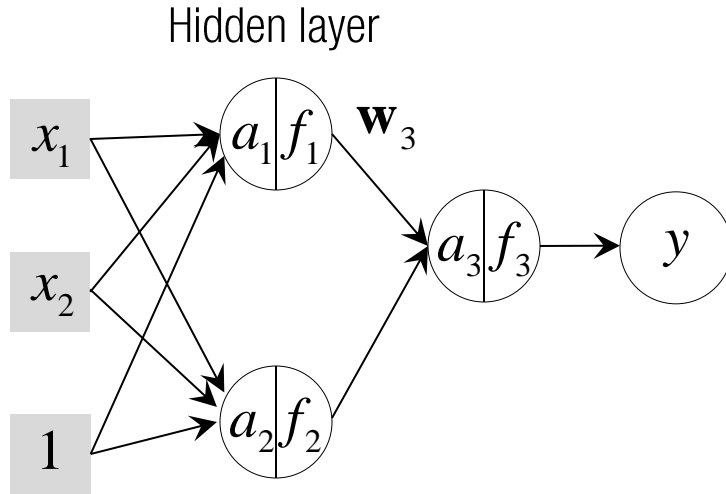
# MULTI-LAYER PERCEPTRON



How many variables?

$$\begin{aligned} f_1 &= \sigma(\mathbf{w}_1 \cdot \mathbf{x}) \\ f_2 &= \sigma(\mathbf{w}_2 \cdot \mathbf{x}) \\ f_3 &= \sigma\left(\mathbf{w}_3 \cdot \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}\right) \end{aligned}$$

# MULTI-LAYER PERCEPTRON



$$f_1 = \sigma(\mathbf{w}_1 \cdot \mathbf{x})$$

$$f_2 = \sigma(\mathbf{w}_2 \cdot \mathbf{x})$$

$$f_3 = \sigma\left(\mathbf{w}_3 \cdot \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}\right)$$

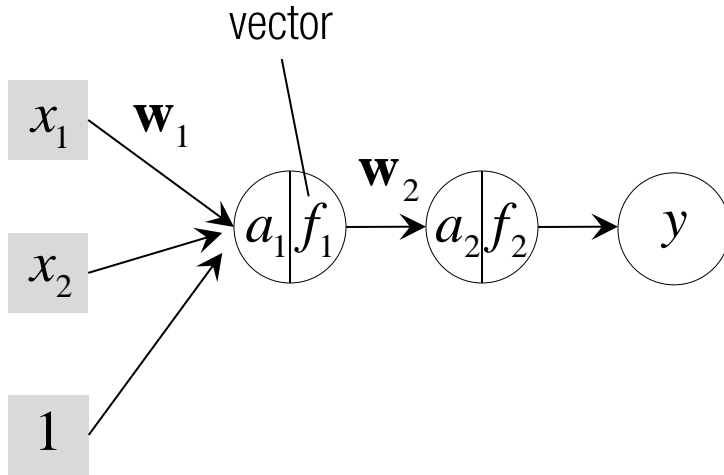
How many variables?

$$m(n+1) + m$$

$n$ : input dimension

$m$ : # of hidden variables

# ***MULTI-LAYER PERCEPTRON***



$$f_2 = \sigma(\mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 \cdot \mathbf{x}))$$

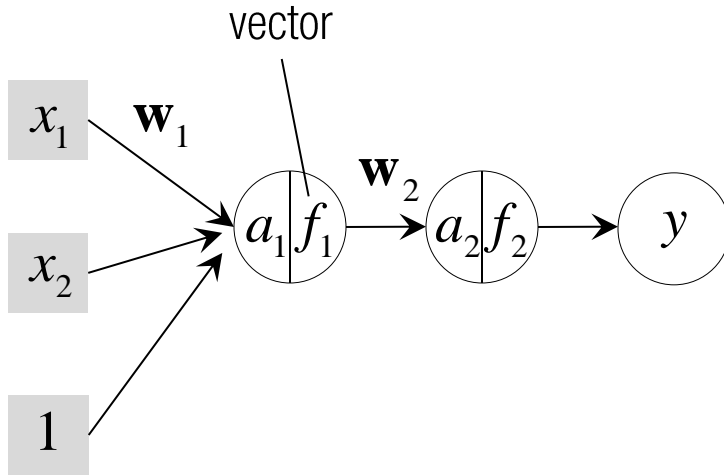
How many variables?

$$m(n+1) + m$$

$n$ : input dimension

$m$ : # of hidden variables

# ***MULTI-LAYER PERCEPTRON***



$$f_2 = \sigma(\mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 \cdot \mathbf{x}))$$

How many variables?

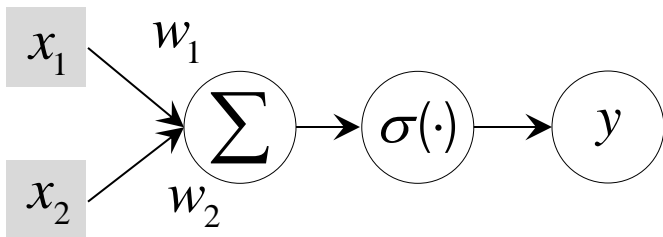
$$m(n+1) + m$$

$n$ : input dimension

$m$ : # of hidden variables

**How to train?**

# RECALL: SINGLE PERCEPTRON



$$f = \sigma(a)$$

$$a = x_1 w_1 + x_2 w_2$$

$$L(w_1, w_2) = \sum_i (f(x_1^i w_1 + x_2^i w_2) - y^i)^2$$

Weight update rule:

$$w_1 = w_1 - \gamma \frac{\partial L}{\partial w_1} \quad w_2 = w_2 - \gamma \frac{\partial L}{\partial w_2}$$

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial a} \frac{\partial a}{\partial w} \quad : \text{chain rule}$$

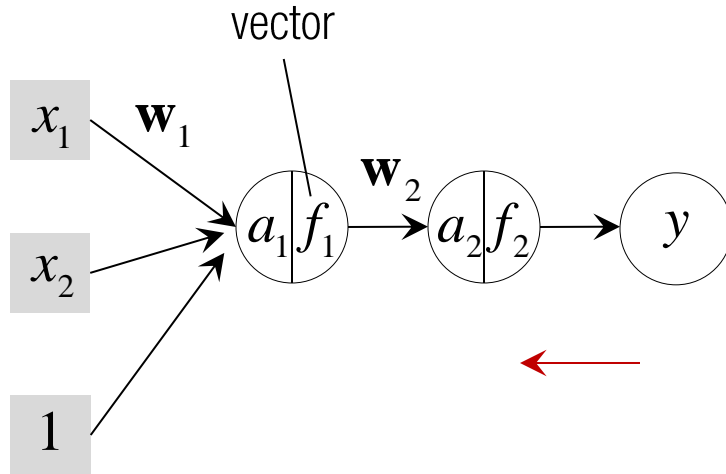
$$\begin{aligned} &= (\tilde{y} - y) \frac{\partial f}{\partial a} \frac{\partial a}{\partial w} \\ &= (\tilde{y} - y) f(1 - f)x \end{aligned}$$

Sigmoid activation

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

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

# BACK-PROPAGATION



$$f_2 = \sigma(\mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 \cdot \mathbf{x}))$$

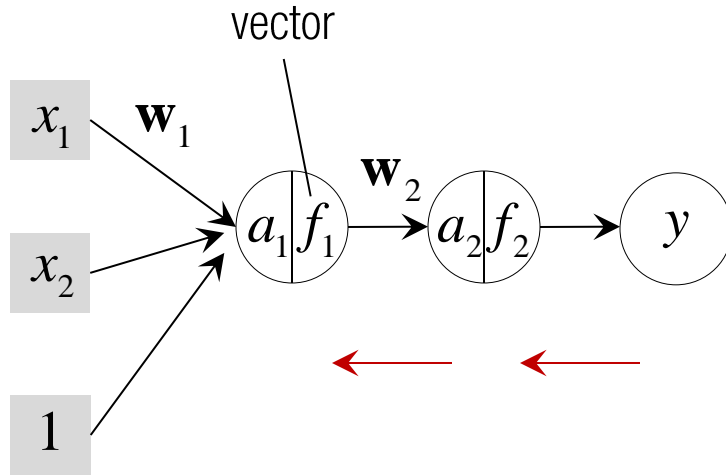
$$L(\mathbf{w}_1, \mathbf{w}_2) = \sum_i (\tilde{y}^i - y^i)^2$$

Weight update rule:

$$\mathbf{w} = \mathbf{w} - \gamma \frac{\partial L}{\partial \mathbf{w}}$$

$$\frac{\partial L}{\partial \mathbf{w}_2} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial \mathbf{w}_2}$$

# BACK-PROPAGATION



$$f_2 = \sigma(\mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 \cdot \mathbf{x}))$$

$$L(\mathbf{w}_1, \mathbf{w}_2) = \sum_i (\tilde{y}^i - y^i)^2$$

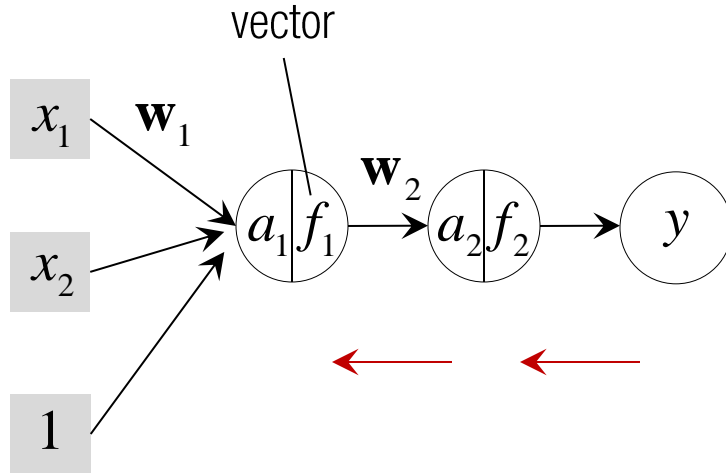
Weight update rule:

$$\mathbf{w} = \mathbf{w} - \gamma \frac{\partial L}{\partial \mathbf{w}}$$

$$\frac{\partial L}{\partial \mathbf{w}_2} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial \mathbf{w}_2}$$

$$\frac{\partial L}{\partial \mathbf{w}_1} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial f_1} \frac{\partial f_1}{\partial a_1} \frac{\partial a_1}{\partial \mathbf{w}_1}$$

# BACK-PROPAGATION



$$f_2 = \sigma(\mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 \cdot \mathbf{x}))$$

$$L(\mathbf{w}_1, \mathbf{w}_2) = \sum_i (\tilde{y}^i - y^i)^2$$

Weight update rule:

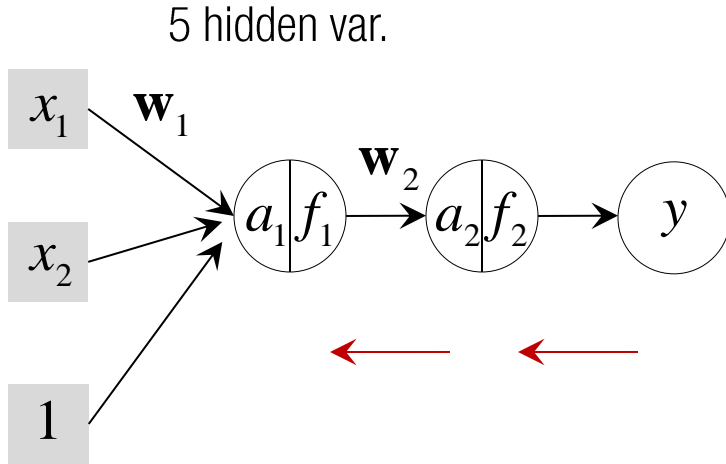
$$\mathbf{w} = \mathbf{w} - \gamma \frac{\partial L}{\partial \mathbf{w}}$$

$$\frac{\partial L}{\partial \mathbf{w}_2} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial \mathbf{w}_2}$$

$$\frac{\partial L}{\partial \mathbf{w}_1} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial f_1} \frac{\partial f_1}{\partial a_1} \frac{\partial a_1}{\partial \mathbf{w}_1}$$



# BACK-PROPAGATION



$$f_2 = \sigma(\mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 \cdot \mathbf{x}))$$

$$L(\mathbf{w}_1, \mathbf{w}_2) = \sum_i (\tilde{y}^i - y^i)^2$$

Weight update rule:

$$\mathbf{w} = \mathbf{w} - \gamma \frac{\partial L}{\partial \mathbf{w}}$$

$$\frac{\partial L}{\partial \mathbf{w}_2} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial \mathbf{w}_2}$$

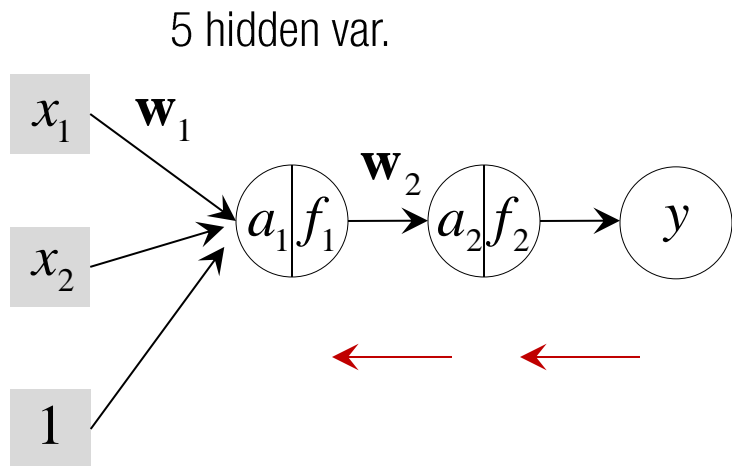
$$f_2 = \sigma(a_2)$$

$$a_2 = \mathbf{w}_2 \cdot f_1$$

$$f_1 = \sigma(a_1)$$

$$a_1 = \mathbf{w}_1 \cdot \mathbf{x}$$

# BACK-PROPAGATION



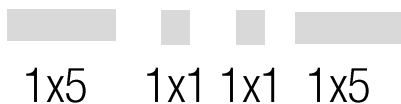
$$f_2 = \sigma(\mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 \cdot \mathbf{x}))$$

$$L(\mathbf{w}_1, \mathbf{w}_2) = \sum_i (\tilde{y}^i - y^i)^2$$

Weight update rule:

$$\mathbf{w} = \mathbf{w} - \gamma \frac{\partial L}{\partial \mathbf{w}}$$

$$\frac{\partial L}{\partial \mathbf{w}_2} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial \mathbf{w}_2}$$



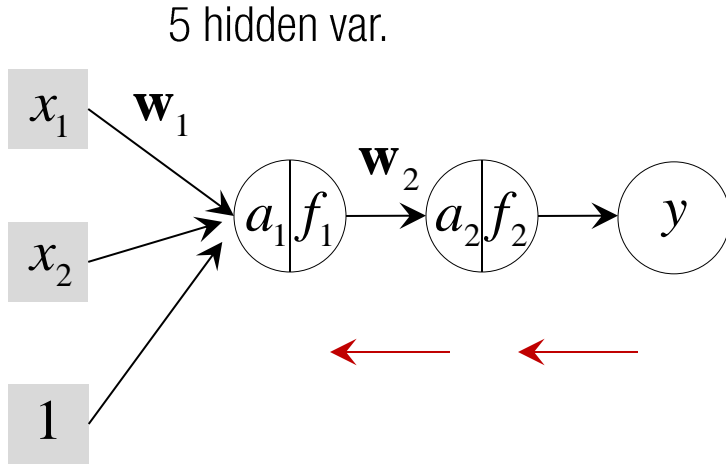
$$f_2 = \sigma(a_2)$$

$$a_2 = \mathbf{w}_2 \cdot f_1$$

$$f_1 = \sigma(a_1)$$

$$a_1 = \mathbf{w}_1 \cdot \mathbf{x}$$

# BACK-PROPAGATION



$$f_2 = \sigma(\mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 \cdot \mathbf{x}))$$

$$L(\mathbf{w}_1, \mathbf{w}_2) = \sum_i (\tilde{y}^i - y^i)^2$$

Weight update rule:

$$\mathbf{w} = \mathbf{w} - \gamma \frac{\partial L}{\partial \mathbf{w}}$$

$$\frac{\partial L}{\partial \mathbf{w}_1} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial f_1} \frac{\partial f_1}{\partial a_1} \frac{\partial a_1}{\partial \mathbf{w}_1}$$

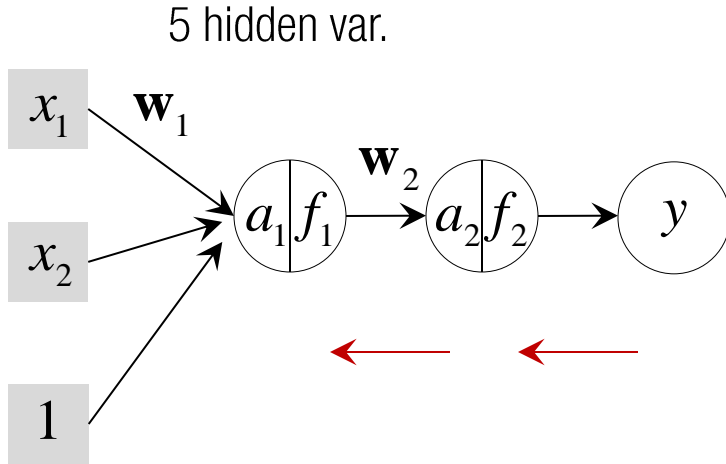
$$f_2 = \sigma(a_2)$$

$$a_2 = \mathbf{w}_2 \cdot f_1$$

$$f_1 = \sigma(a_1)$$

$$a_1 = \mathbf{w}_1 \cdot \mathbf{x}$$

# BACK-PROPAGATION



$$f_2 = \sigma(\mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 \cdot \mathbf{x}))$$

$$L(\mathbf{w}_1, \mathbf{w}_2) = \sum_i (\tilde{y}^i - y^i)^2$$

Weight update rule:

$$\mathbf{w} = \mathbf{w} - \gamma \frac{\partial L}{\partial \mathbf{w}}$$

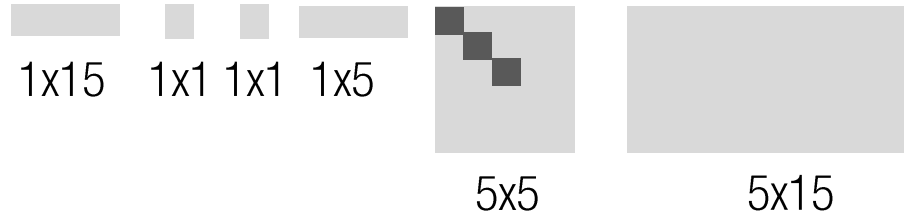
$$\frac{\partial L}{\partial \mathbf{w}_1} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial f_1} \frac{\partial f_1}{\partial a_1} \frac{\partial a_1}{\partial \mathbf{w}_1}$$

$$f_2 = \sigma(a_2)$$

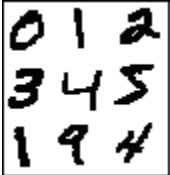
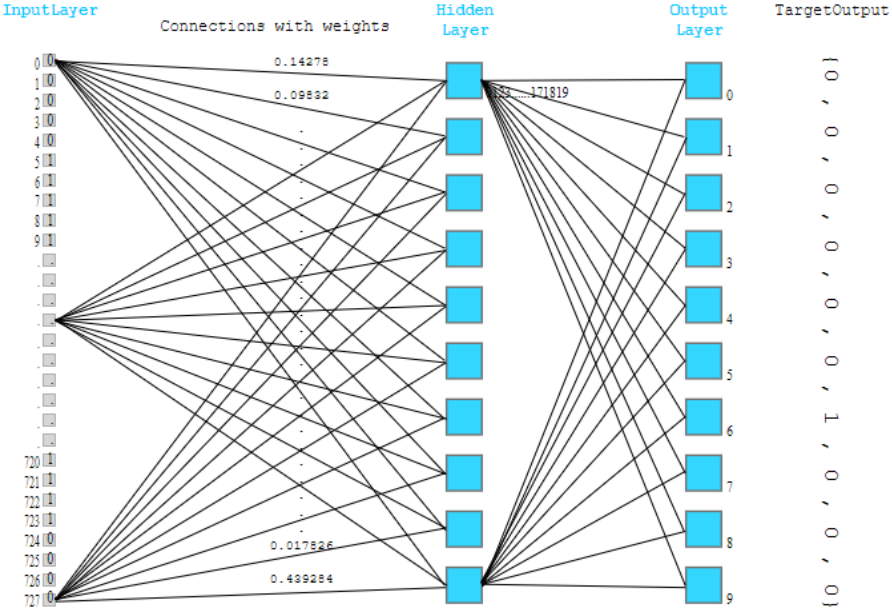
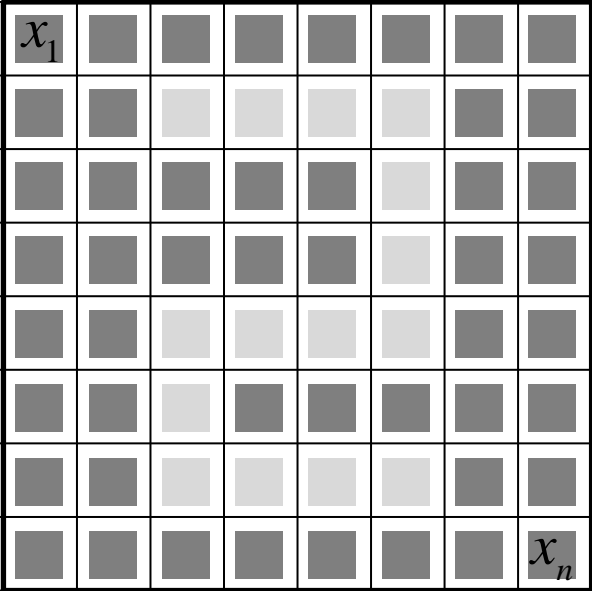
$$a_2 = \mathbf{w}_2 \cdot f_1$$

$$f_1 = \sigma(a_1)$$

$$a_1 = \mathbf{w}_1 \cdot \mathbf{x}$$

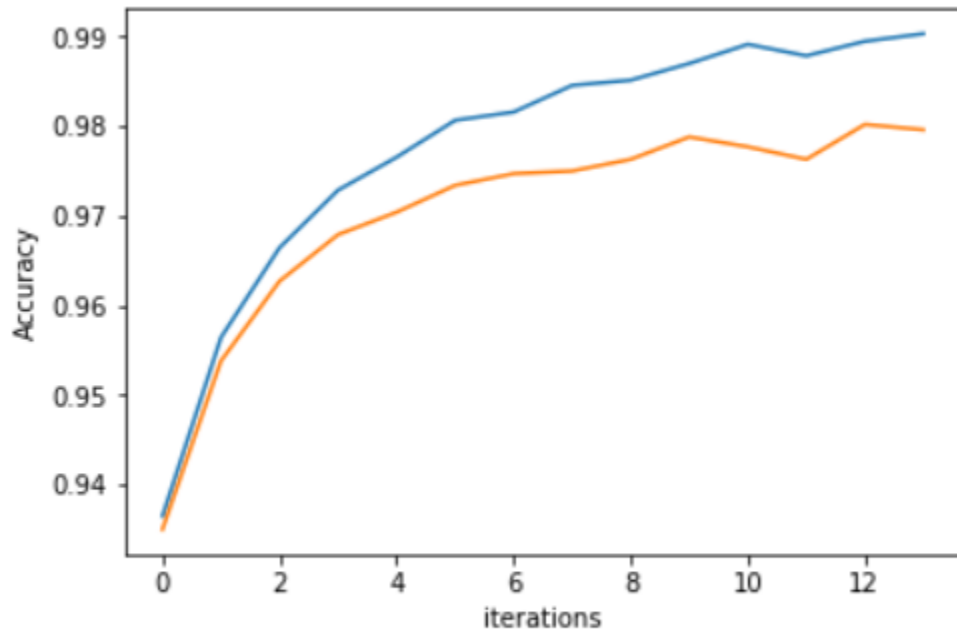


# DIGIT RECOGNITION



# *DIGIT RECOGNITION*

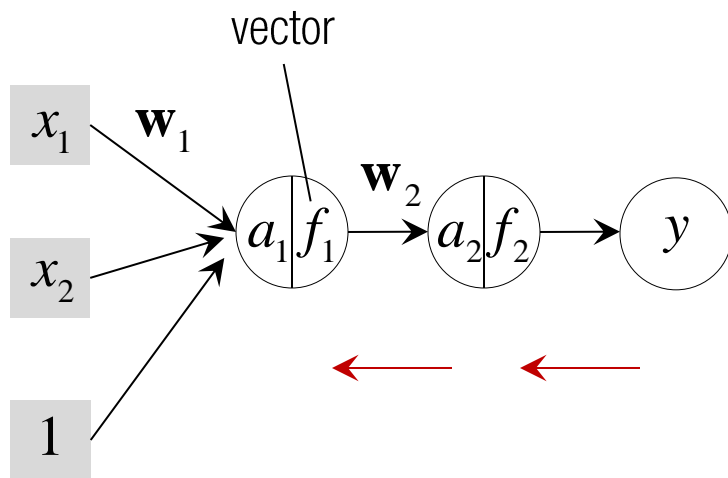
$x_1$	■	■	■	■	■	■	■
■	■	■	■	■	■	■	■
■	■	■	■	■	■	■	■
■	■	■	■	■	■	■	■
■	■	■	■	■	■	■	■
■	■	■	■	■	■	■	■
■	■	■	■	■	■	■	■
■	■	■	■	■	■	■	$x_n$



Train Accuracy: 0.99

Test Accuracy: 0.98

# BACK-PROPAGATION ALGORITHM W/ GRADIENT DESCENT



$$f_2 = \sigma(\mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 \cdot \mathbf{x}))$$

While until converges:

For each data sample,

1. Prediction

$$f_2 = \sigma(\mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 \cdot \mathbf{x}))$$

2. Measure error

$$L(\mathbf{w}_1, \mathbf{w}_2) = \sum_i (\tilde{y}^i - y^i)^2$$

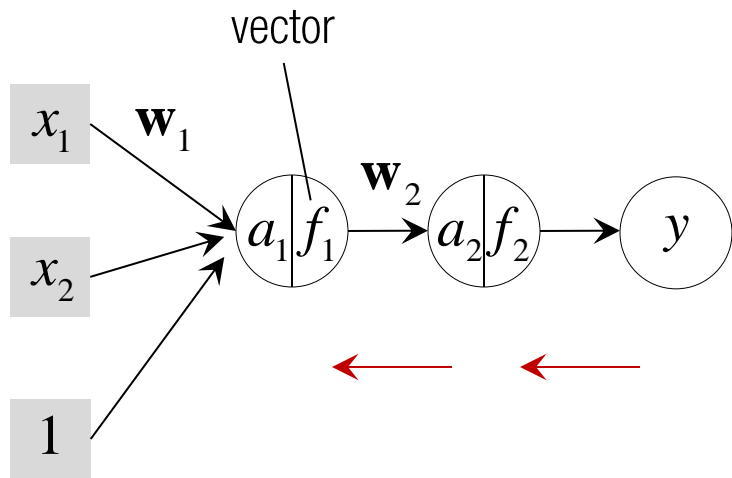
3. Back-propagation

$$\frac{\partial L}{\partial \mathbf{w}_2} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial \mathbf{w}_2} \quad \frac{\partial L}{\partial \mathbf{w}_1} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial f_1} \frac{\partial f_1}{\partial a_1} \frac{\partial a_1}{\partial \mathbf{w}_1}$$

Update gradient

$$\mathbf{w}_1 = \mathbf{w}_1 - \gamma \frac{\partial L}{\partial \mathbf{w}_1} \quad \mathbf{w}_2 = \mathbf{w}_2 - \gamma \frac{\partial L}{\partial \mathbf{w}_2}$$

# GRADIENT DESCENT



Weight update rule:

$$\mathbf{w}_1 = \mathbf{w}_1 - \gamma \frac{\partial L}{\partial \mathbf{w}_1}$$

$$\mathbf{w}_2 = \mathbf{w}_2 - \gamma \frac{\partial L}{\partial \mathbf{w}_2}$$

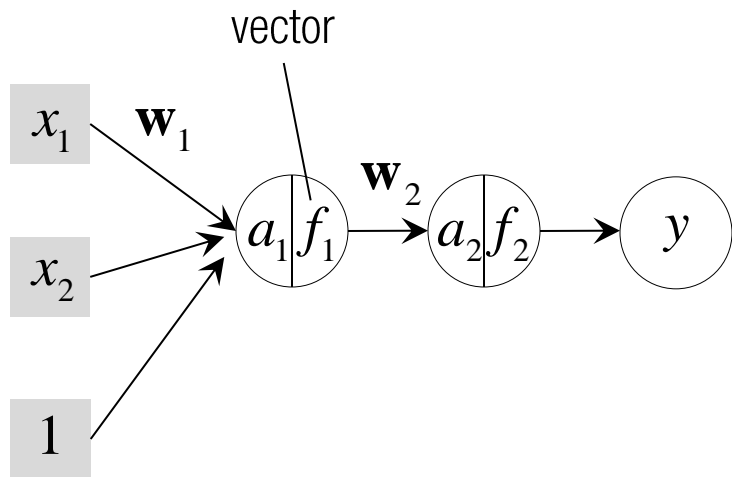
where

$$\frac{\partial L}{\partial \mathbf{w}_1} = \frac{1}{n} \sum_{i=1}^n \frac{\partial L_i}{\partial \mathbf{w}_1}$$
$$\frac{\partial L}{\partial \mathbf{w}_2} = \frac{1}{n} \sum_{i=1}^n \frac{\partial L_i}{\partial \mathbf{w}_2}$$

$n$  : the number of training data (> millions)



# STOCHASTIC GRADIENT DESCENT (SGD)



Weight update rule:

$$\mathbf{w}_1 = \mathbf{w}_1 - \gamma \frac{\partial L}{\partial \mathbf{w}_1}$$

$$\mathbf{w}_2 = \mathbf{w}_2 - \gamma \frac{\partial L}{\partial \mathbf{w}_2}$$

where 
$$\frac{\partial L}{\partial \mathbf{w}_1} = \frac{1}{n} \sum_{i=1}^n \frac{\partial L_i}{\partial \mathbf{w}_1}$$

$$\frac{\partial L}{\partial \mathbf{w}_2} = \frac{1}{n} \sum_{i=1}^n \frac{\partial L_i}{\partial \mathbf{w}_2}$$

$n$  : the number of training data (> millions)

$$\frac{\partial L}{\partial \mathbf{w}_1} \approx \frac{\partial L_i}{\partial \mathbf{w}_1}$$

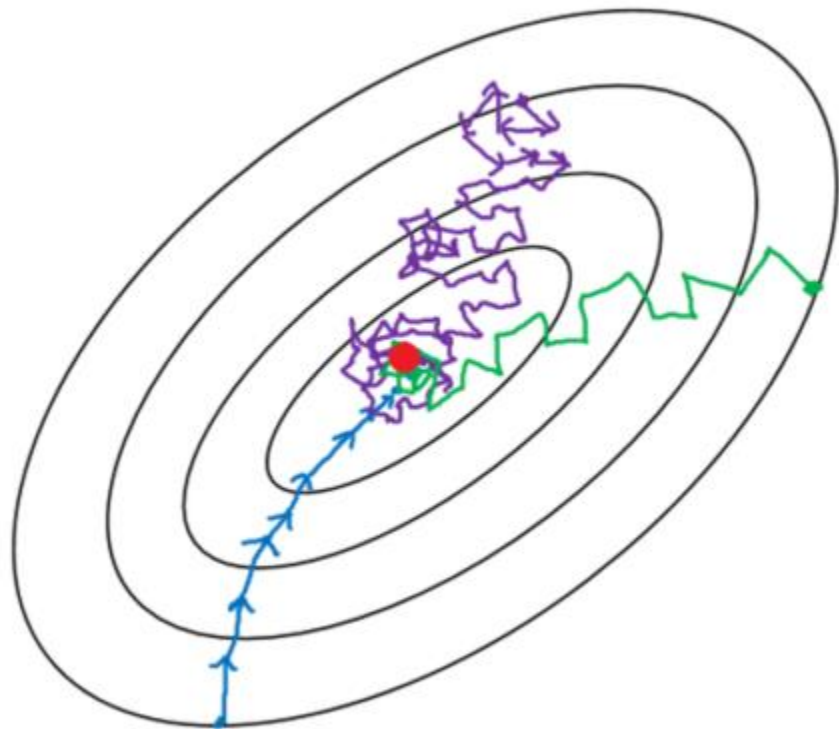
$$\frac{\partial L}{\partial \mathbf{w}_2} \approx \frac{\partial L_i}{\partial \mathbf{w}_2}$$

or 
$$\frac{\partial L}{\partial \mathbf{w}_1} \approx \sum_{i \in \mathcal{R}} \frac{\partial L_i}{\partial \mathbf{w}_1} \quad |\mathcal{R}| \ll n$$

$$\frac{\partial L}{\partial \mathbf{w}_2} \approx \sum_{i \in \mathcal{R}} \frac{\partial L_i}{\partial \mathbf{w}_2}$$

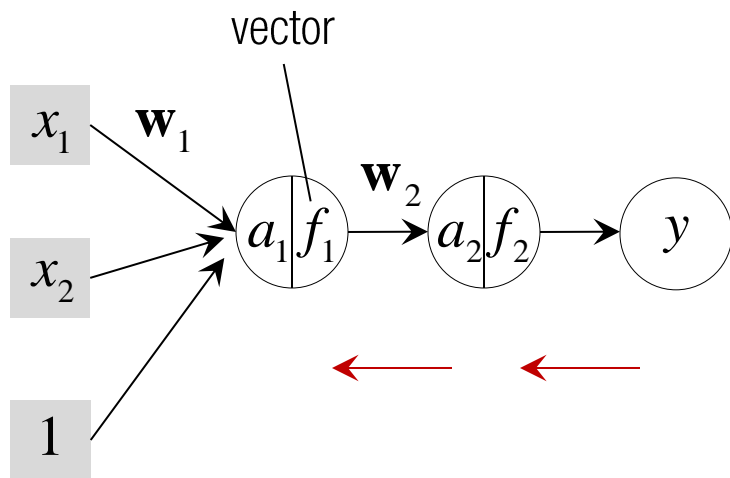
$\mathcal{R}$  : random subset of training data,  
a.k.a. mini-batch

# *STOCHASTIC GRADIENT DESCENT (SGD)*



- Batch gradient descent
- Mini-batch gradient Descent
- Stochastic gradient descent

# BACK-PROPAGATION ALGORITHM W/ GRADIENT DESCENT



$$f_2 = \sigma(\mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 \cdot \mathbf{x}))$$

While until converges:

For each data sample,

1. Prediction

$$f_2 = \sigma(\mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 \cdot \mathbf{x}))$$

2. Measure error

$$L(\mathbf{w}_1, \mathbf{w}_2) = \sum_i (\tilde{y}^i - y^i)^2$$

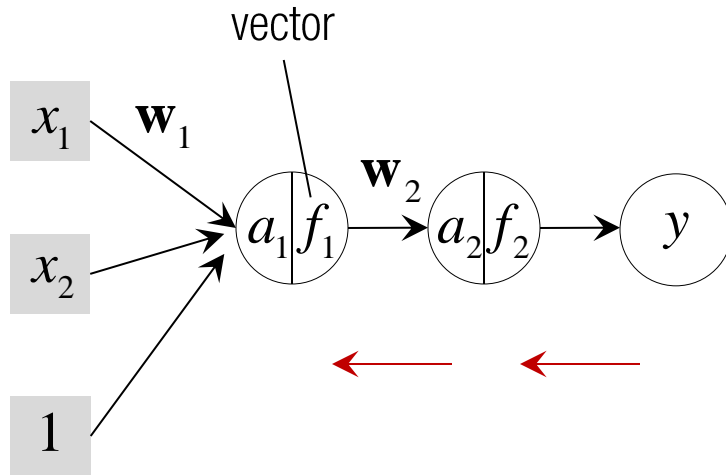
3. Back-propagation

$$\frac{\partial L}{\partial \mathbf{w}_2} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial \mathbf{w}_2} \quad \frac{\partial L}{\partial \mathbf{w}_1} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial f_1} \frac{\partial f_1}{\partial a_1} \frac{\partial a_1}{\partial \mathbf{w}_1}$$

Update gradient

$$\mathbf{w}_1 = \mathbf{w}_1 - \gamma \frac{\partial L}{\partial \mathbf{w}_1} \quad \mathbf{w}_2 = \mathbf{w}_2 - \gamma \frac{\partial L}{\partial \mathbf{w}_2}$$

# BACK-PROPAGATION ALGORITHM W/ STOCHASTIC GRADIENT DESCENT



$$f_2 = \sigma(\mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 \cdot \mathbf{x}))$$

While until converges:

For each data sample,

1. Prediction

$$f_2 = \sigma(\mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 \cdot \mathbf{x}))$$

2. Measure error

$$L(\mathbf{w}_1, \mathbf{w}_2) = \sum_i (\tilde{y}^i - y^i)^2$$

3. Back-propagation

$$\frac{\partial L}{\partial \mathbf{w}_2} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial \mathbf{w}_2} \quad \frac{\partial L}{\partial \mathbf{w}_1} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial f_1} \frac{\partial f_1}{\partial a_1} \frac{\partial a_1}{\partial \mathbf{w}_1}$$

4. Update gradient

$$\mathbf{w}_1 = \mathbf{w}_1 - \gamma \frac{\partial L}{\partial \mathbf{w}_1} \quad \mathbf{w}_2 = \mathbf{w}_2 - \gamma \frac{\partial L}{\partial \mathbf{w}_2}$$

# BACK-PROPAGATION ALGORITHM W/ STOCHASTIC GRADIENT DESCENT

While until converges:

Sample mini-batch

For each data sample in mini-batch,

1. Prediction

$$f_2 = \sigma(\mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 \cdot \mathbf{x}))$$

2. Measure error

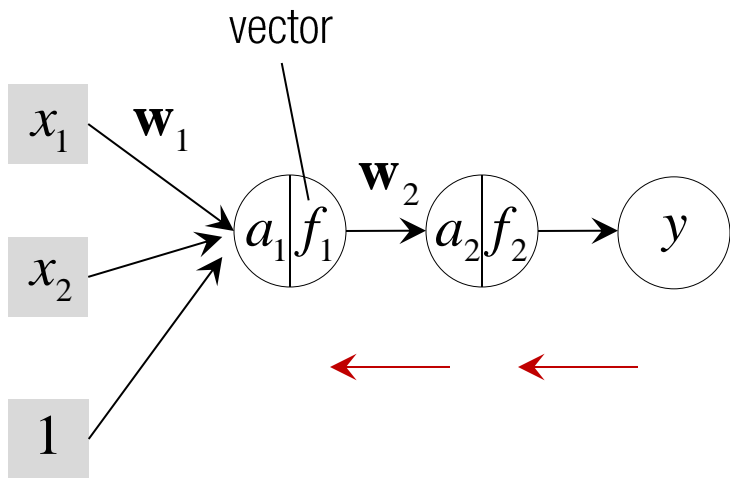
$$L(\mathbf{w}_1, \mathbf{w}_2) = \sum_i (\tilde{y}^i - y^i)^2$$

3. Back-propagation

$$\frac{\partial L}{\partial \mathbf{w}_2} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial \mathbf{w}_2} \quad \frac{\partial L}{\partial \mathbf{w}_1} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial f_1} \frac{\partial f_1}{\partial a_1} \frac{\partial a_1}{\partial \mathbf{w}_1}$$

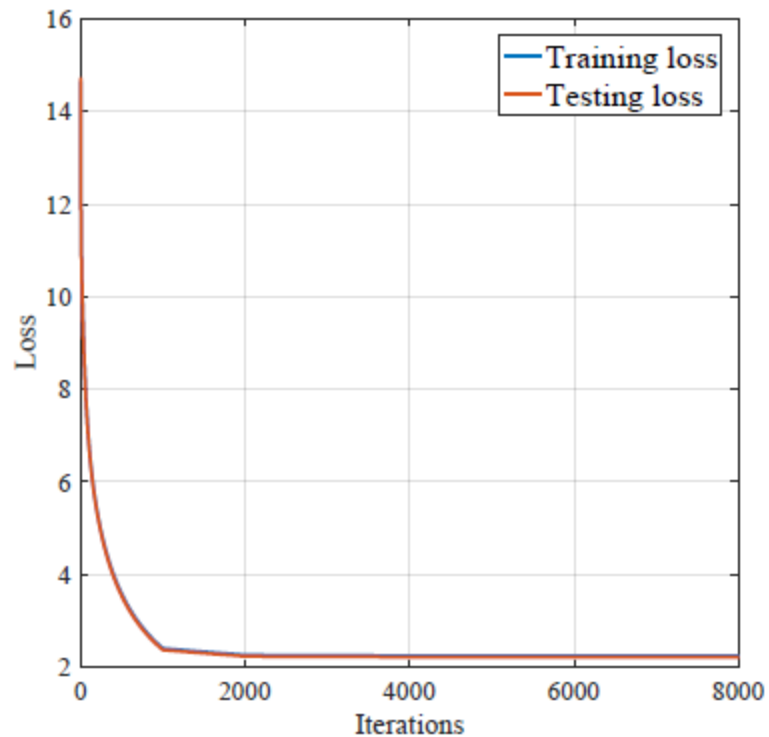
Update gradient

$$\mathbf{w}_1 = \mathbf{w}_1 - \gamma \frac{\partial L}{\partial \mathbf{w}_1} \quad \mathbf{w}_2 = \mathbf{w}_2 - \gamma \frac{\partial L}{\partial \mathbf{w}_2}$$

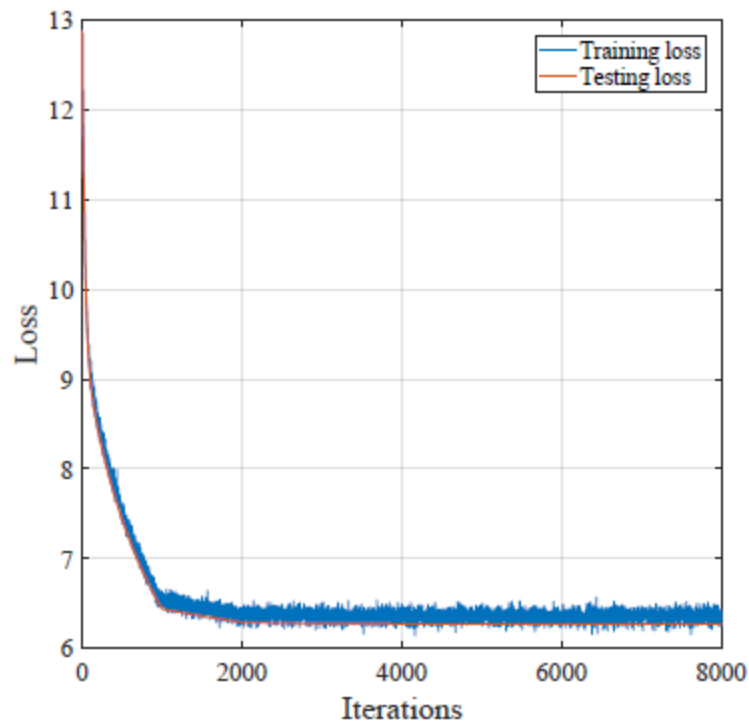


$$f_2 = \sigma(\mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 \cdot \mathbf{x}))$$

Gradient descent



SGD with mini-batch



# HW #4: NEURAL NETWORK IMPLEMENTATION

---

**Algorithm 2** Stochastic Gradient Descent based Training

---

- 1: Set the learning rate  $\gamma$
  - 2: Set the decay rate  $\lambda \in (0, 1]$
  - 3: Initialize the weights with a Gaussian noise  $\mathbf{w} \in \mathcal{N}(0, 1)$
  - 4: **for**  $\text{ilter} = 1 : \text{nIters}$  **do**
  - 5:     At every 100<sup>th</sup> iteration,  $\gamma \leftarrow \lambda\gamma$
  - 6:      $\frac{\partial L}{\partial \mathbf{w}} \leftarrow 0$
  - 7:     Sample  $R$  images randomly.
  - 8:     **for**  $i = 1 : R$  **do**
  - 9:         Label prediction of  $\mathbf{x}_i$
  - 10:         Loss computation  $l$
  - 11:         Gradient back-propagation of  $\mathbf{x}_i$ ,  $\frac{\partial l}{\partial \mathbf{w}}$  using back-propagation.
  - 12:          $\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial \mathbf{w}} + \frac{\partial l}{\partial \mathbf{w}}$
  - 13:     **end for**
  - 14:     Update the weights,  $\mathbf{w} \leftarrow \mathbf{w} - \frac{\gamma}{R} \frac{\partial L}{\partial \mathbf{w}}$
  - 15: **end for**
-

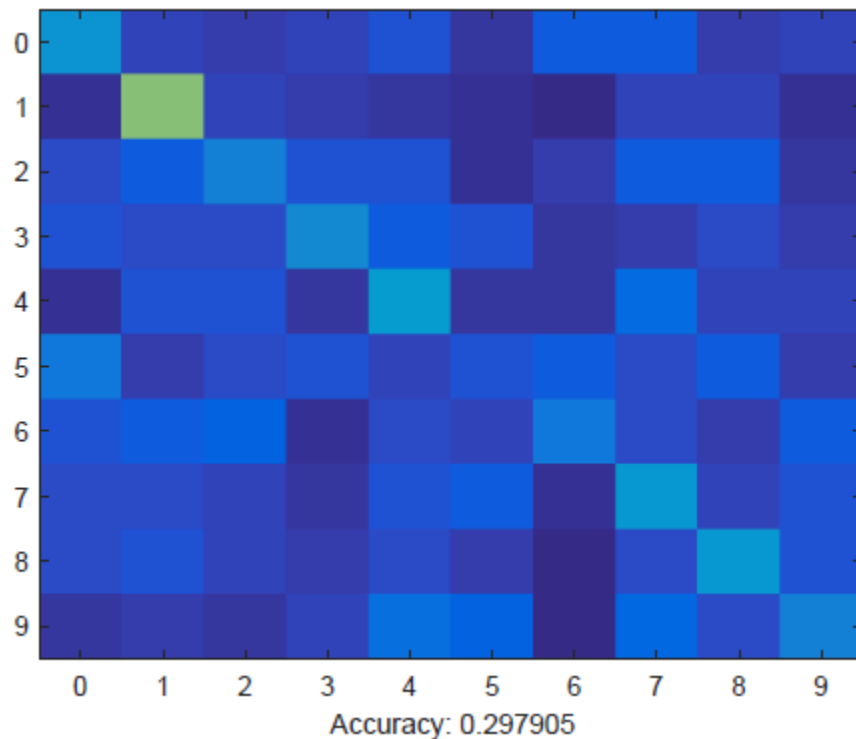
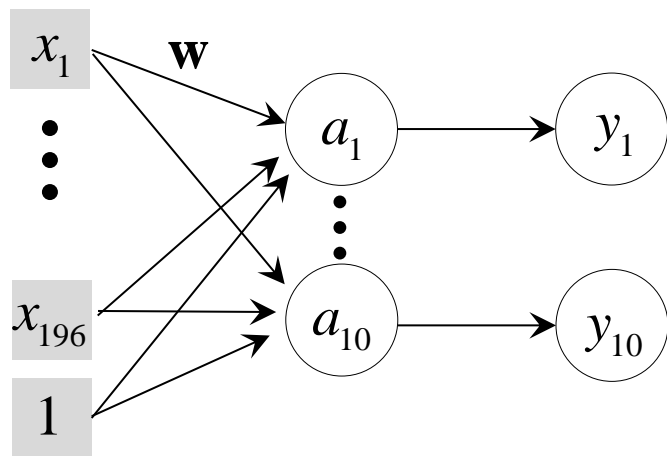
# *HW #4: NEURAL NETWORK IMPLEMENTATION*

0 0 0 0 0 0 0 0 0 0  
1 1 1 1 1 1 1 1 1 1  
2 2 2 2 2 2 2 2 2 2  
3 3 3 3 3 3 3 3 3 3  
4 4 4 4 4 4 4 4 4 4  
5 5 5 5 5 5 5 5 5 5  
6 6 6 6 6 6 6 6 6 6  
7 7 7 7 7 7 7 7 7 7  
8 8 8 8 8 8 8 8 8 8  
9 9 9 9 9 9 9 9 9 9



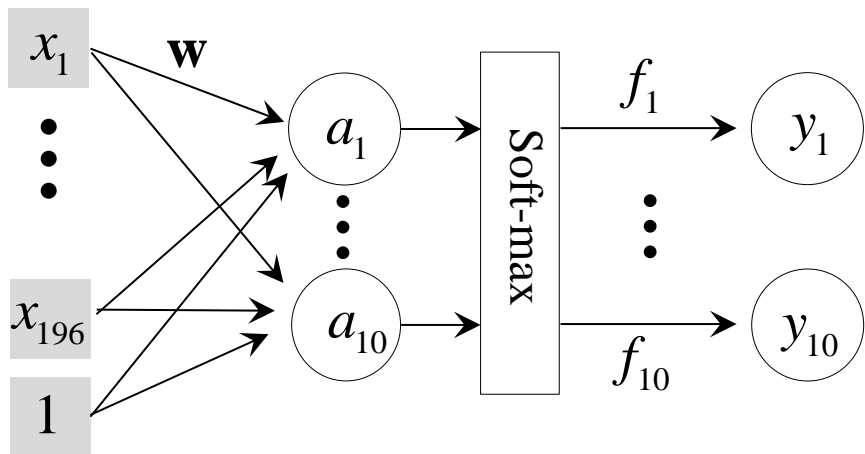
# HW #4: NEURAL NETWORK IMPLEMENTATION

## STEP 1: SINGLE LINEAR PERCEPTRON



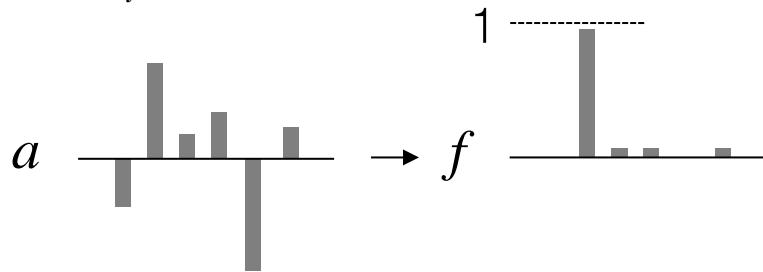
# HW #4: NEURAL NETWORK IMPLEMENTATION

## STEP 2: SINGLE PERCEPTRON



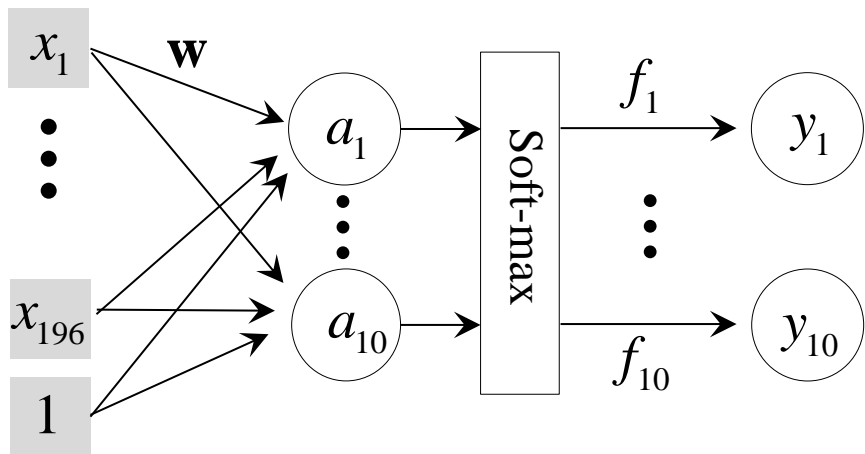
Soft-max (winners-take-all):

$$f_i = \frac{e^{a_i}}{\sum_i e^{a_i}}$$



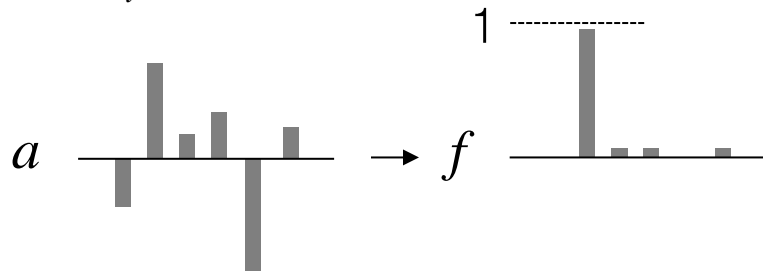
# HW #4: NEURAL NETWORK IMPLEMENTATION

## STEP 2: SINGLE PERCEPTRON



Soft-max (winners-take-all):

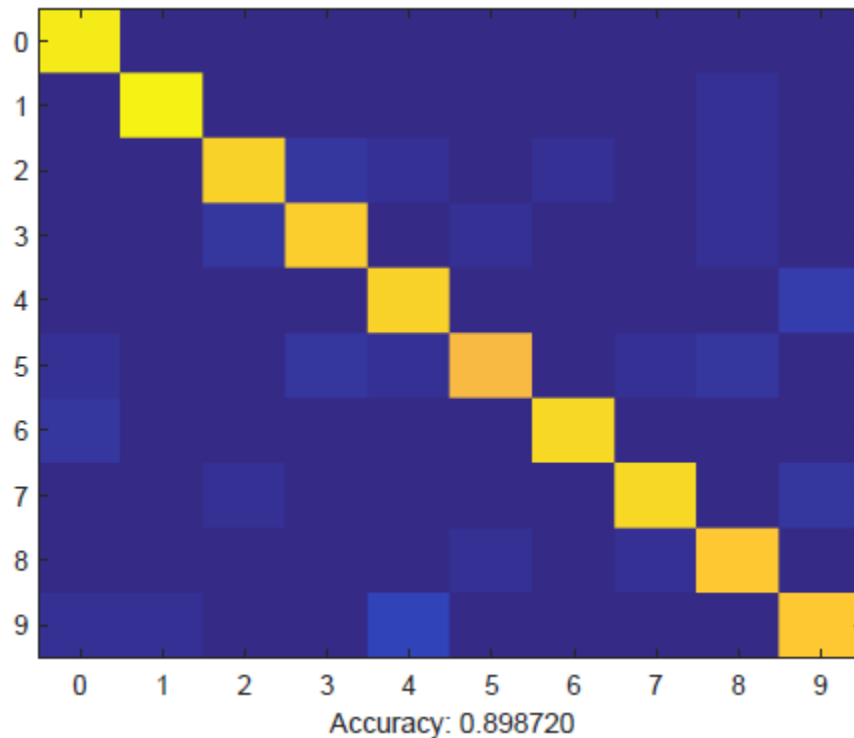
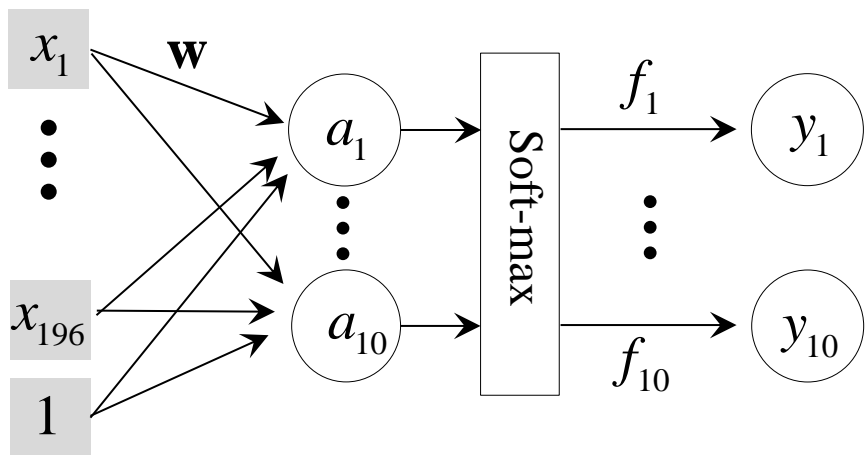
$$f_i = \frac{e^{a_i}}{\sum_i e^{a_i}}$$



$$\frac{\partial f_i}{\partial a_j} = \begin{cases} f_i(1 - f_i) & i = j \\ -f_i f_j & i \neq j \end{cases}$$

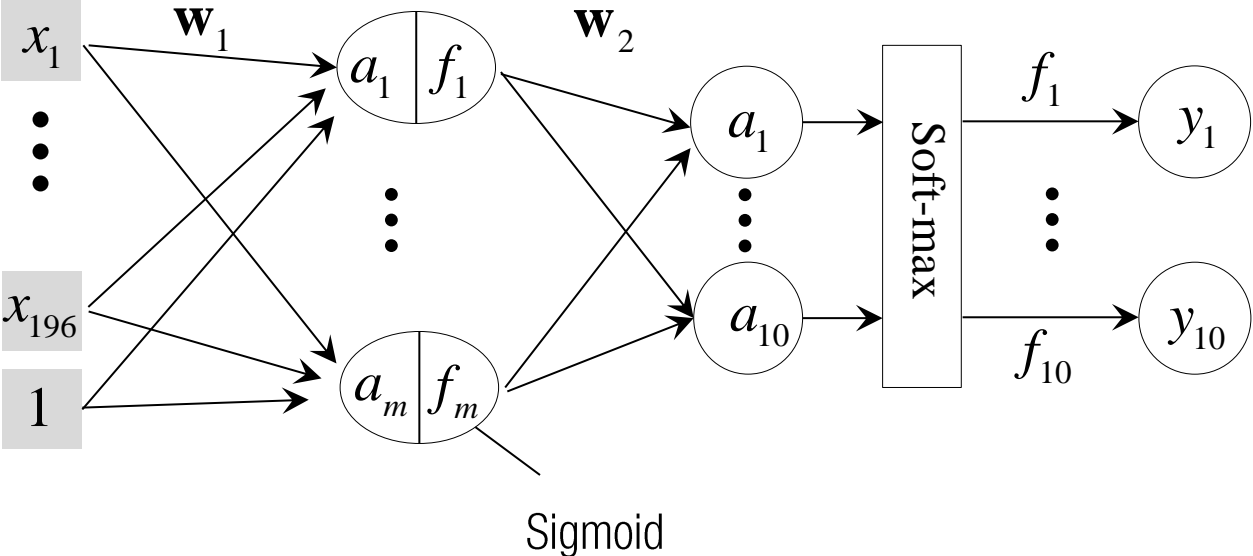
# HW #4: NEURAL NETWORK IMPLEMENTATION

## STEP 2: SINGLE PERCEPTRON



# HW #4: NEURAL NETWORK IMPLEMENTATION

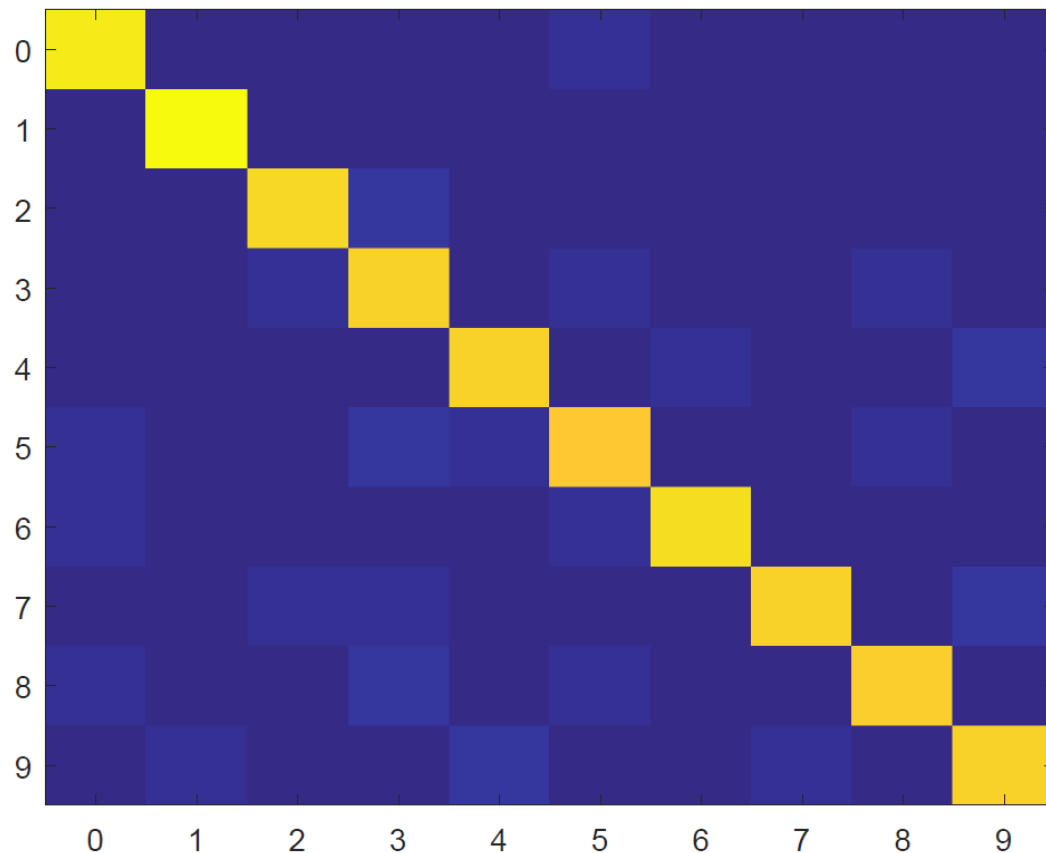
## STEP 3: MULTI-LAYER PERCEPTRON



Dimensions?

# *HW #4: NEURAL NETWORK IMPLEMENTATION*

## *STEP 3: MULTI-LAYER PERCEPTRON*



Accuracy: 0.914553

