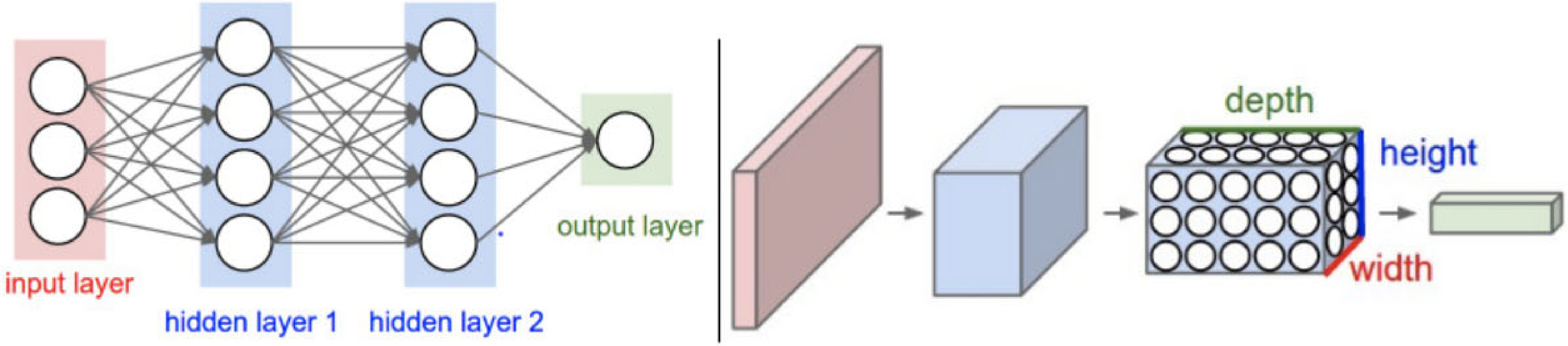
Machine Learning
Multilayer Networks

Jian Liu

Part 1: MLP

Part 2: CNN

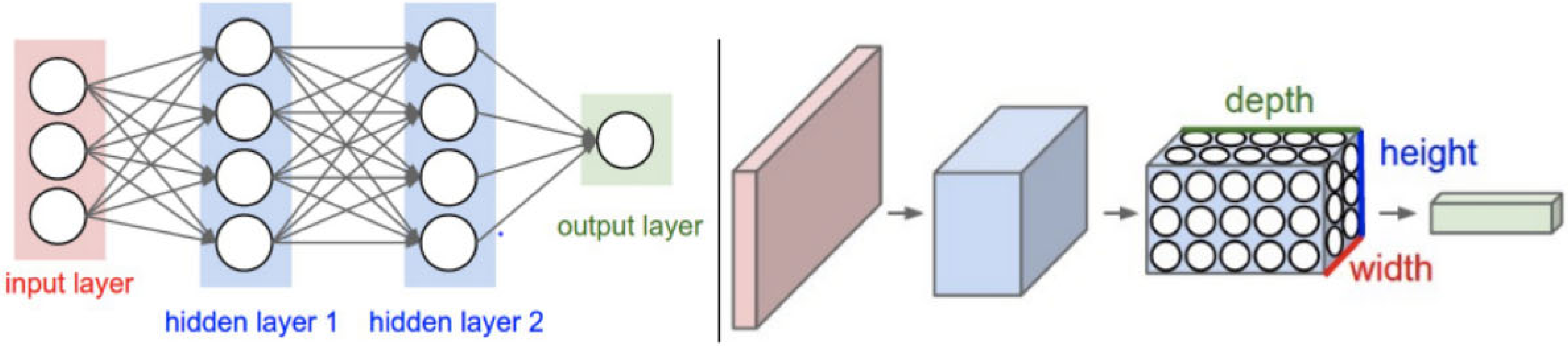


MLP [towardsdatascience] CNN

Machine Learning
Multilayer Networks

Jian Liu

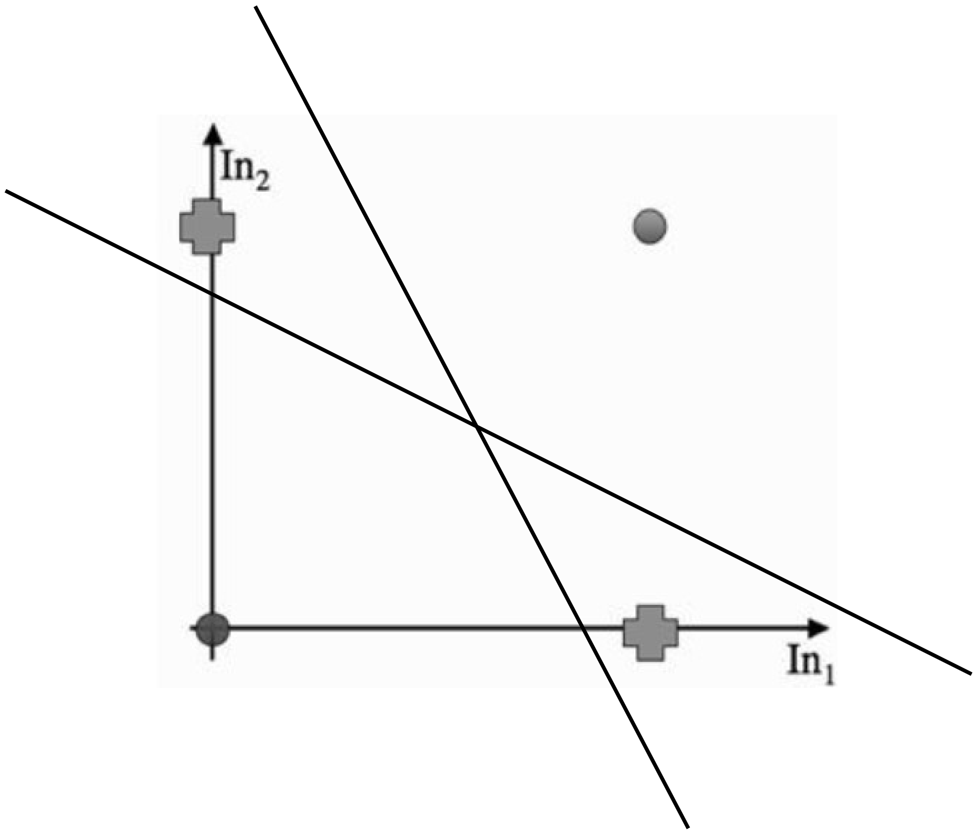
Part 1: MLP

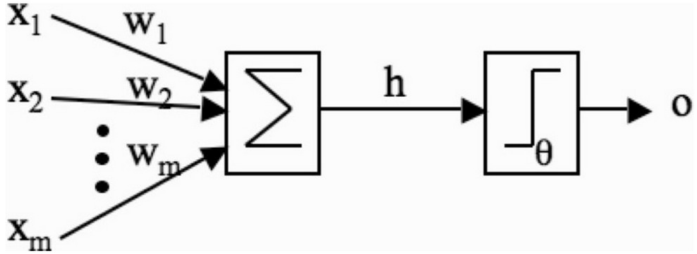


MLP towardsdatascience CNN

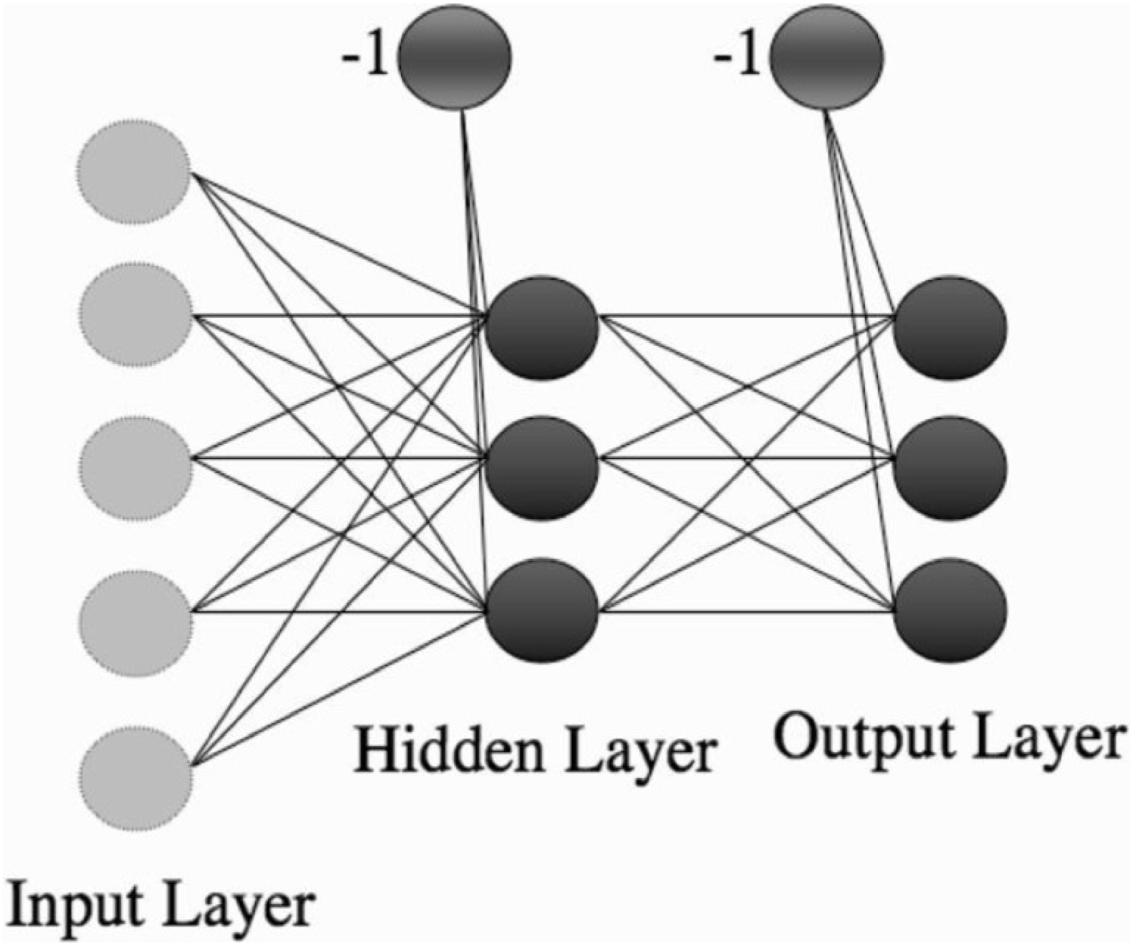
Perceptron limitations

XOR

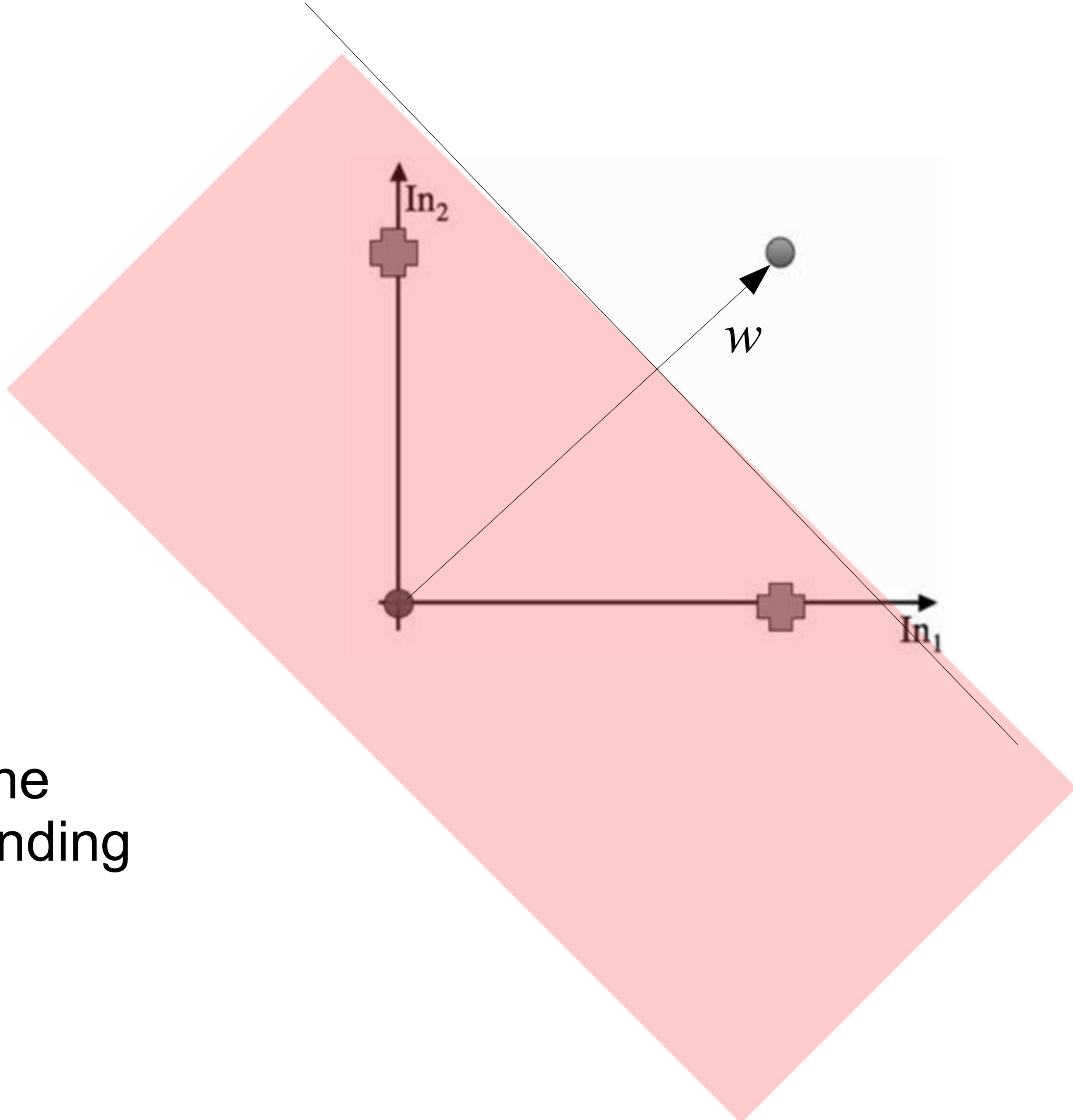




Multi-layer Perceptron (MLP)

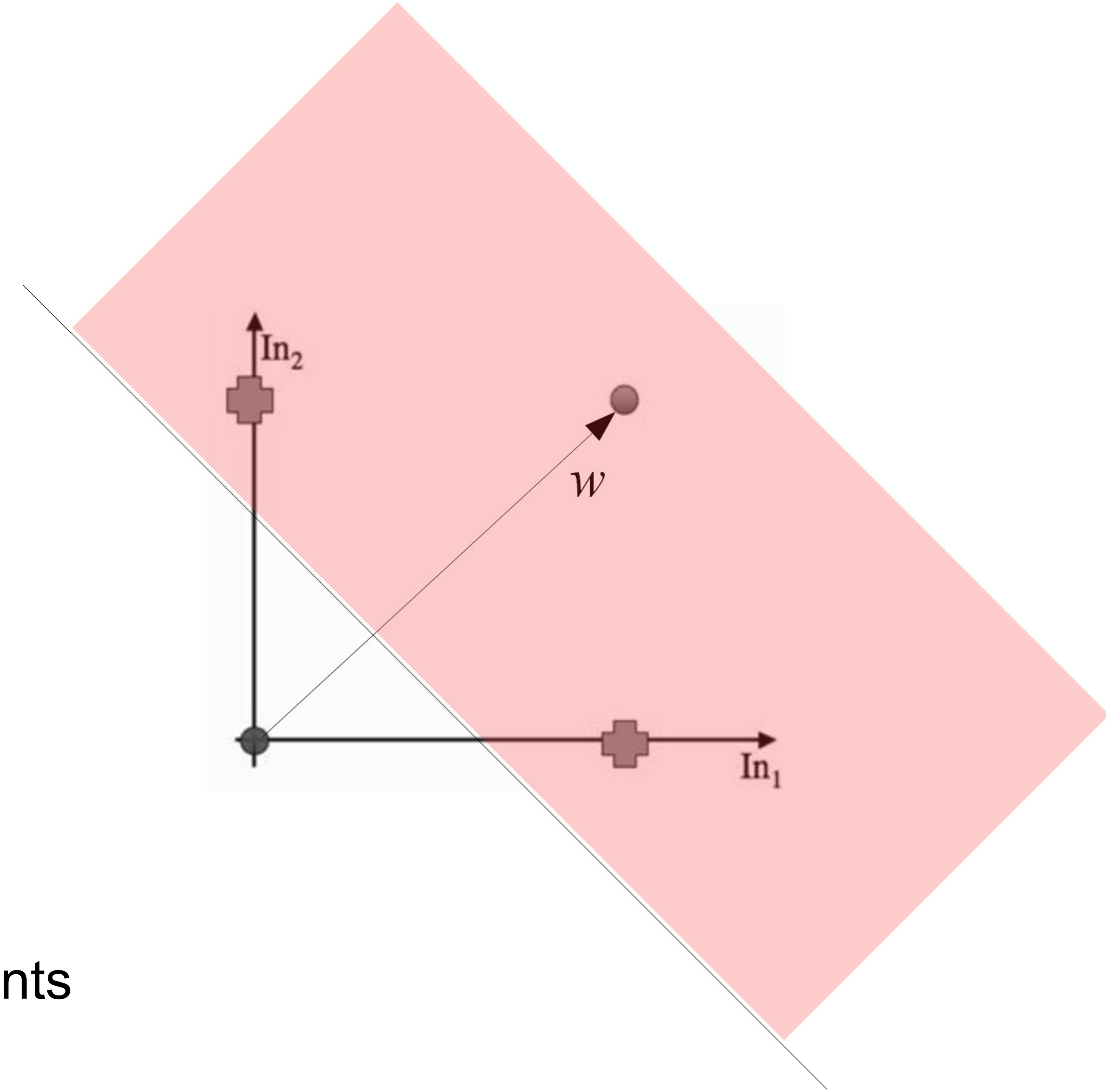


MLP and XOR



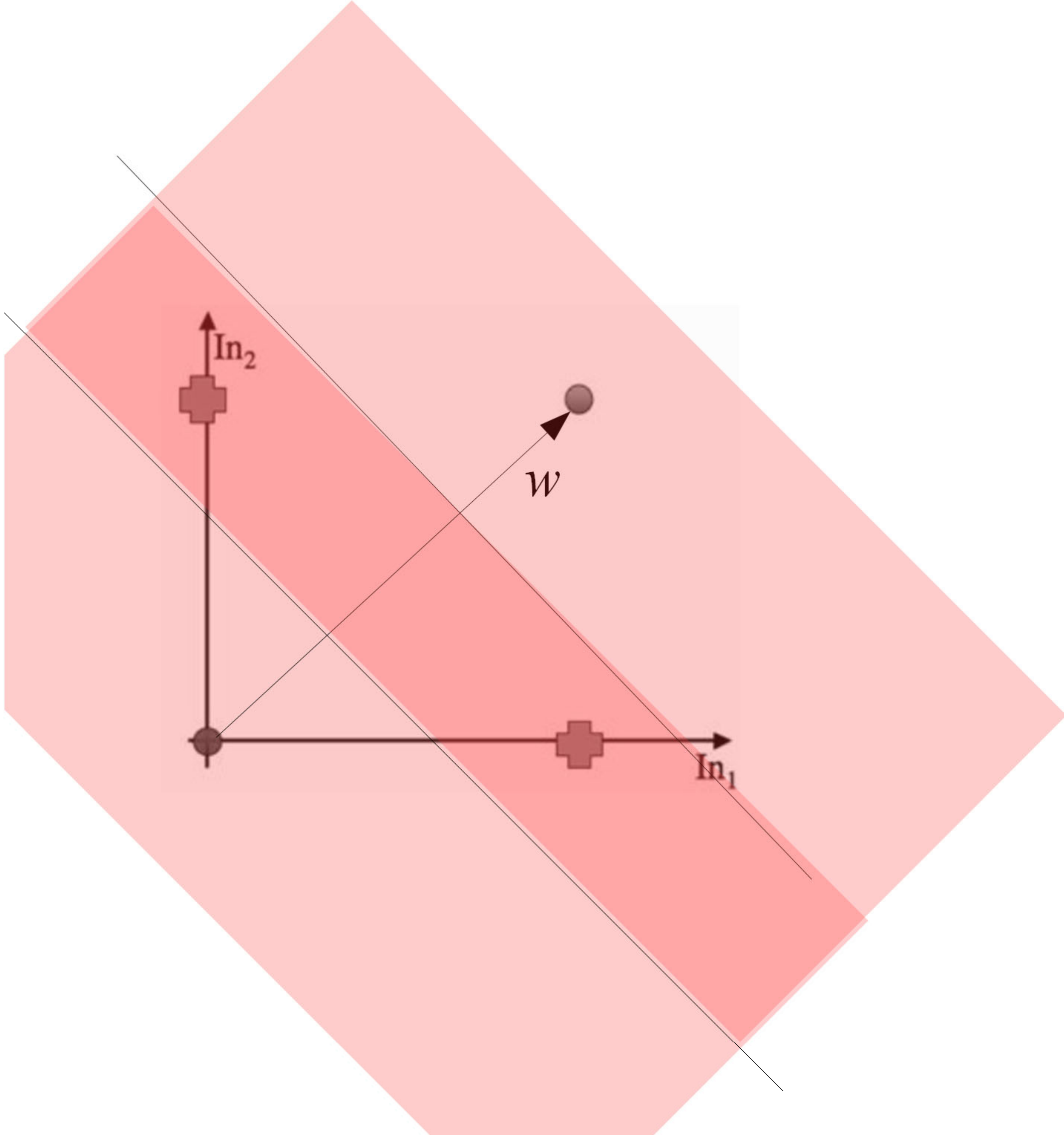
Choose a straight line that separates the points as in the figure, and whose corresponding perceptron returns 1 in the highlighted area

MLP and XOR



Now for the other points

MLP and XOR



Possible solution:

$-x_1 - x_2 + 2.5 = 0$

$w = \langle -1, -1 \rangle$

$x_1 + x_2 - 0.5 = 0$

$w = \langle 1, 1 \rangle$

MLP and XOR

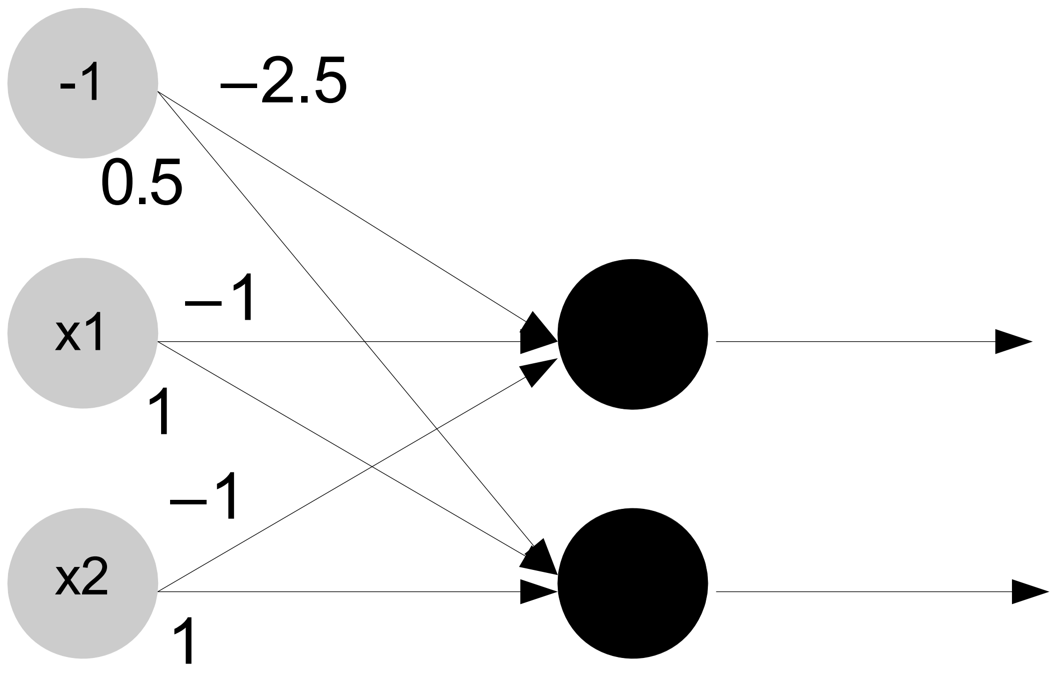
$-x_1 - x_2 + 2.5 \geq 0$

$x_1 + x_2 - 0.5 \geq 0$

These are 2 perceptrons with weights:

$\langle -1, -1, -2.5 \rangle$

$\langle 1, 1, 0.5 \rangle$



MLP and XOR

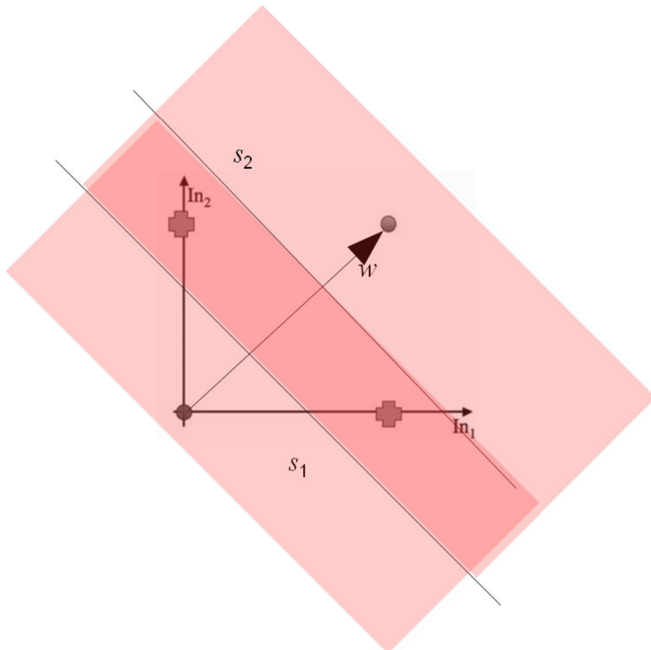
Their outputs are:

What we want

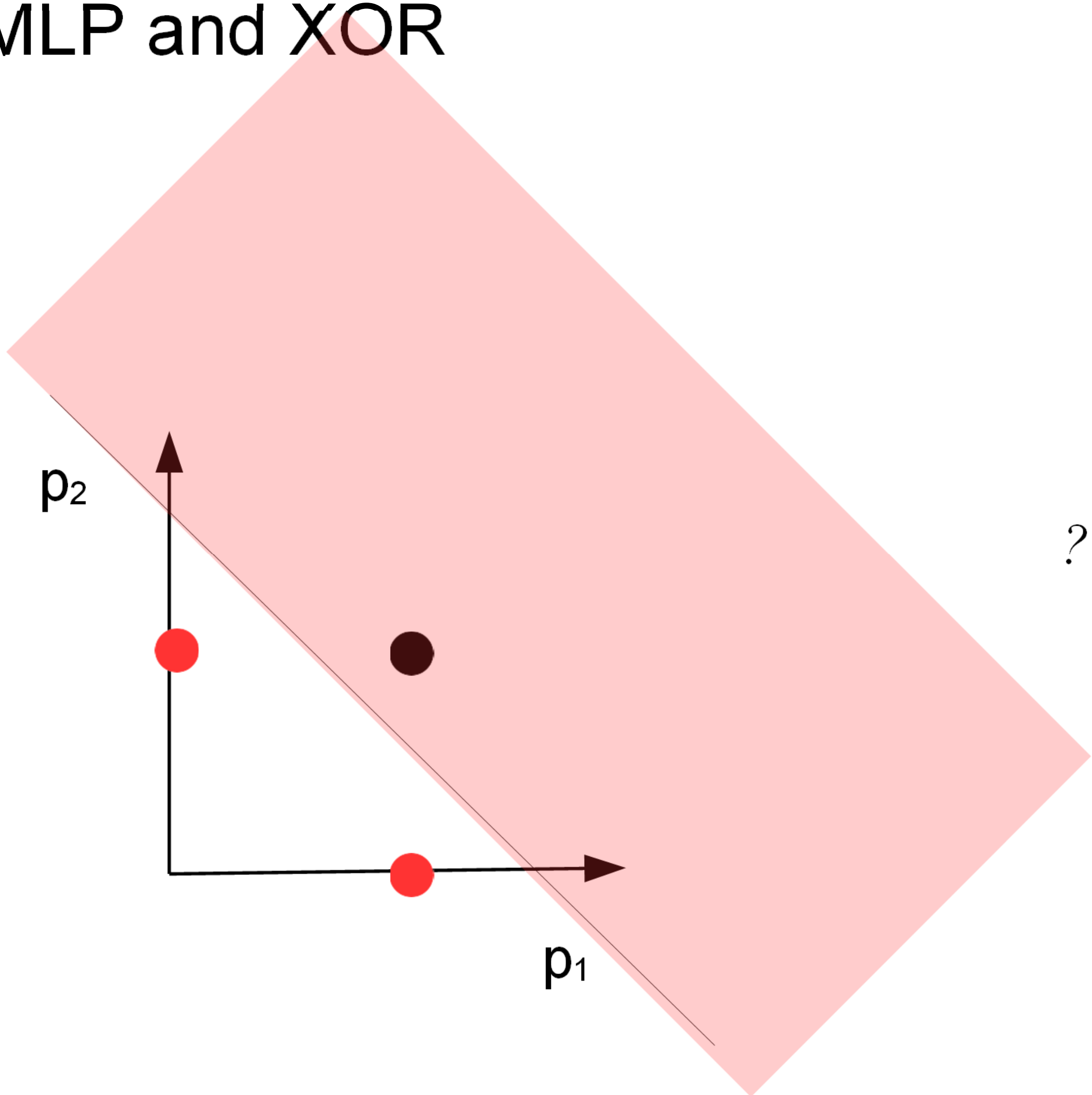
x1	x2	p1	p2	o
0	0	1	0	0
0	1	1	1	1
1	0	1	1	1
1	1	0	1	0

$$p_1 = -x_1 - x_2 + 2.5 \geq 0$$

$$p_2 = x_1 + x_2 - 0.5 \geq 0$$



MLP and XOR

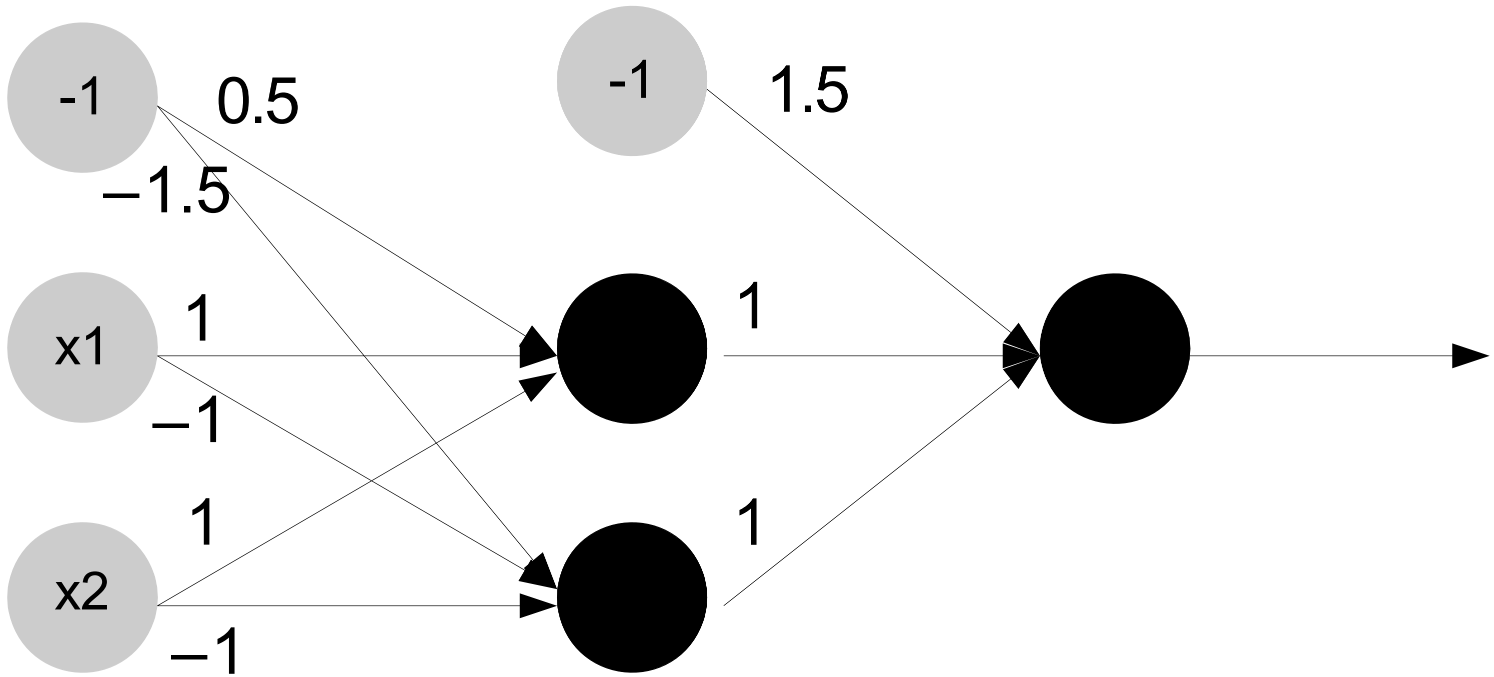


MLP and XOR

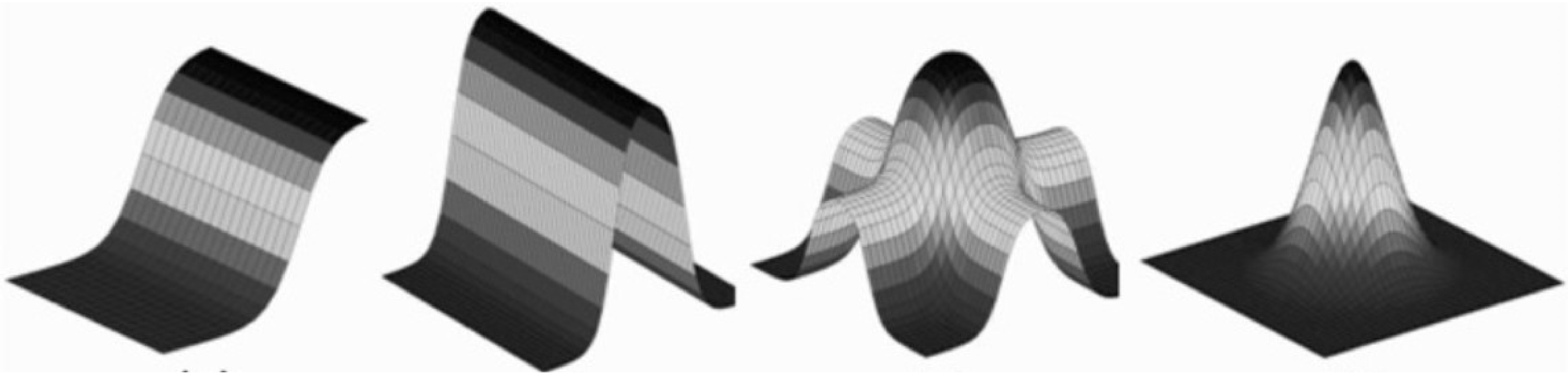
$$p_1 \equiv x_1 + x_2 - 0.5 \geq 0$$

$$p_2 \equiv -x_1 - x_2 + 1.5 \geq 0$$

$$o \equiv p_1 + p_2 - 1.5 \geq 0$$



A Universal Approximator



$$g(x)=\sum_{j}^{N} w_j \sigma(y_j^T x+\theta_j)$$ given $q(x)$ $\epsilon>0$

$$|g(x)-q(x)|<\epsilon$$

Error definition

$$E(\boldsymbol{X}) = \sum_{x_n \in \boldsymbol{X}} |y_n - t_n|$$

Number of errors on the training set

$$E_p(\boldsymbol{X}) = \sum_{x_n \in \boldsymbol{X}} \boldsymbol{w}^T \boldsymbol{x}_n (y_n - t_n)$$

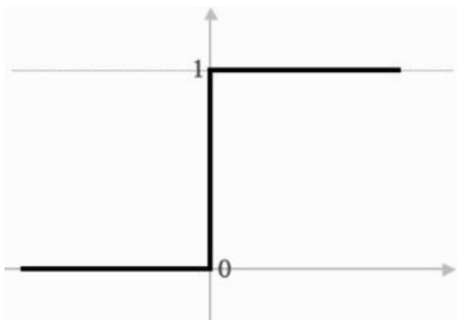
The Perceptron error

$$E_m(\boldsymbol{X}) = \frac{1}{2} \sum_{x_n \in \boldsymbol{X}} (y_n - t_n)^2$$

Squared error function (differentiable!)
Usually known as the Mean Squared Error (MSE)

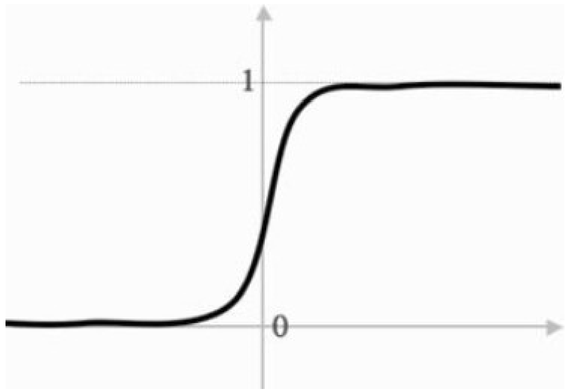
$$y = f\left(\sum_{i=1}^{M} w_i x_i\right)$$

Output is differentiable if f is

$f =$ 

Not good

A different activation function



The sigmoid function: $$f(x)=\frac{1}{1+e^{-\beta x}}\equiv\sigma_{\beta}$$

$$\sigma_{\beta}{}'(x)=?$$

The derivative of the sigmoid

The sigmoid function: $f(x) = \dfrac{1}{1+e^{-\beta x}} \equiv \sigma_\beta$

$$\sigma_\beta'(x) = ?$$

Two useful properties of derivatives:

$$f(x) = e^x \qquad f'(x) = e^x$$

Example: $(e^{x^2})' = e^{x^2} \cdot 2x$

Chain rule: $(f \circ g)'(x) = f'(g(x)) \cdot g'(x)$

Hint: $\dfrac{1}{1+e^{-\beta x}} = (1+e^{-\beta x})^{-1}$

The derivative of the sigmoid

The sigmoid function: $f(x) = \dfrac{1}{1+e^{-x}} \equiv \sigma$ where $\beta = 1$ for simplicity

We derive the most external function first

Then this

$$\sigma'(x) = \left((1+e^{-x})^{-1}\right)' = -1(1+e^{-x})^{-2} \cdot (1+e^{-x})' =$$

$$= -1(1+e^{-x})^{-2} \cdot e^{-x} \cdot (-x)' = -1(1+e^{-x})^{-2} \cdot e^{-x} \cdot (-1)$$

and finally this one

$$\sigma'(x) = -1(1+e^{-x})^{-2} \cdot e^{-x} \cdot (-1) = \frac{e^{-x}}{(1+e^{-x})^2}$$

Let's note that:

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

The same thing with β, FYI

The sigmoid function: $h(x) = \dfrac{1}{1+e^{-\beta x}} \equiv \sigma_\beta$

We derive the most external function first

Then this

$$\sigma_\beta'(x) = \left(\left(1+e^{-\beta x}\right)^{-1}\right)' = -1\left(1+e^{-\beta x}\right)^{-2} \cdot \left(1+e^{-\beta x}\right)' =$$

$$= -1\left(1+e^{-\beta x}\right)^{-2} \cdot e^{-\beta x} \cdot (-\beta x)' = -1\left(1+e^{-\beta x}\right)^{-2} \cdot e^{-\beta x} \cdot (-\beta)$$

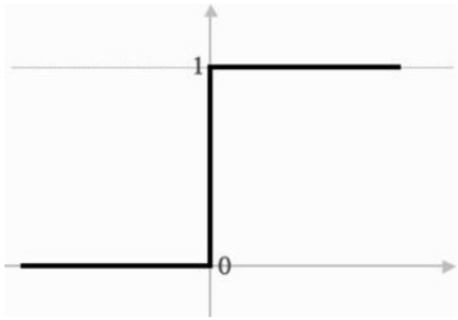
and finally this one

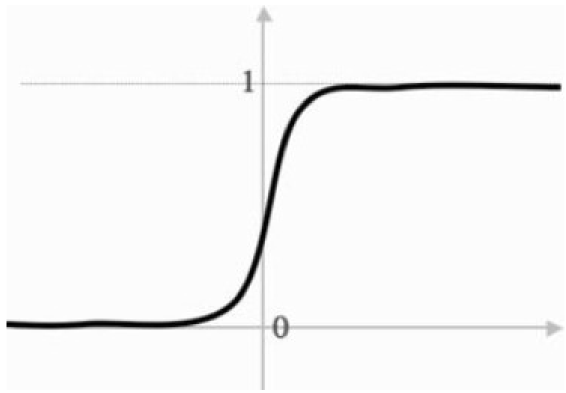
$$\sigma_\beta'(x) = -1\left(1+e^{-\beta x}\right)^{-2} \cdot e^{-\beta x} \cdot (-\beta) = \frac{\beta e^{-\beta x}}{\left(1+e^{-\beta x}\right)^2}$$

Let's note that:

$$1-\sigma_\beta = 1 - \frac{1}{1+e^{-\beta x}} = \frac{1+e^{-\beta x}-1}{1+e^{-\beta x}} = \frac{e^{-\beta x}}{1+e^{-\beta x}} \qquad \Rightarrow \sigma_\beta' = \beta\sigma_\beta(1-\sigma_\beta)$$

A different activation function





before

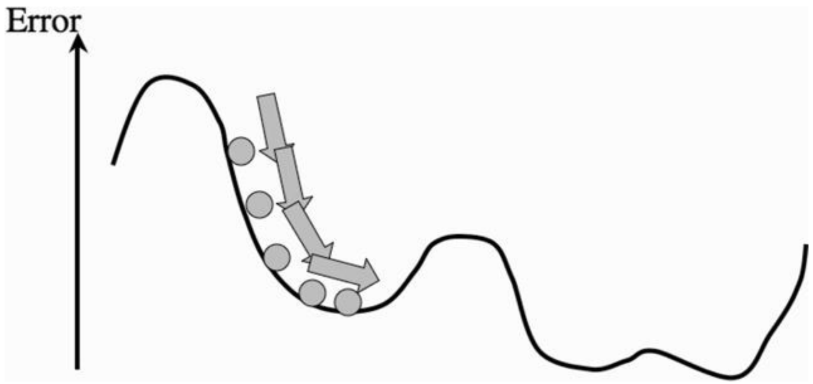
after

$$y(\boldsymbol{w}^T\boldsymbol{x}) = \begin{cases} 1 & \text{if} \quad \boldsymbol{w}^T\boldsymbol{x} > 0 \\ 0 & \text{if} \quad \boldsymbol{w}^T\boldsymbol{x} \le 0 \end{cases}$$

$$y(\boldsymbol{w}^T\boldsymbol{x}) = \frac{1}{1+e^{-\beta \boldsymbol{w}^T \boldsymbol{x}}}$$

$$\sigma_\beta{}'(x) = \beta \frac{e^{-\beta x}}{(1+e^{-\beta x})^2} = \beta\, \sigma_\beta(x)(1-\sigma_\beta(x))$$

Gradient descent (again)



$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla E(\mathbf{x})$$

Perceptron

$$E_p(\mathbf{X}) = \sum_{x_n \in X} \mathbf{w}^t \mathbf{x}_n (y_n - t_n)$$

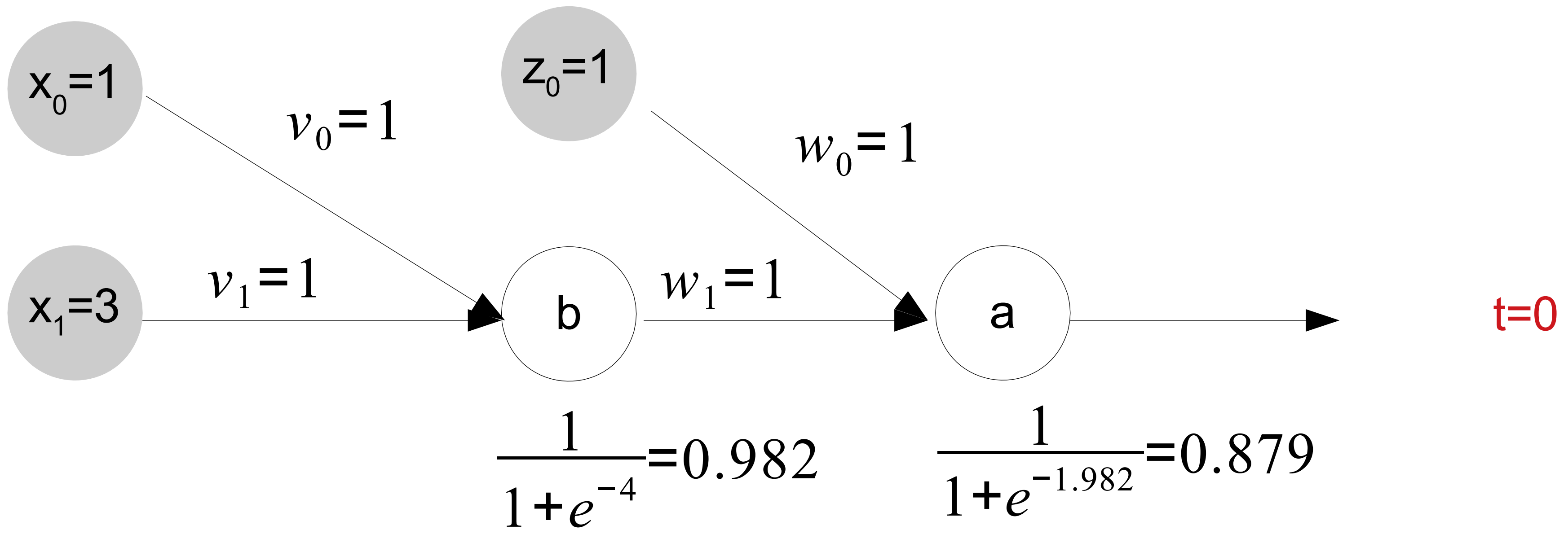
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\, \mathbf{x}(y - t)$$

Multi-Layer P

$$E_m(\mathbf{X}) = \frac{1}{2} \sum_{x_n \in X} (y_n - t_n)^2$$

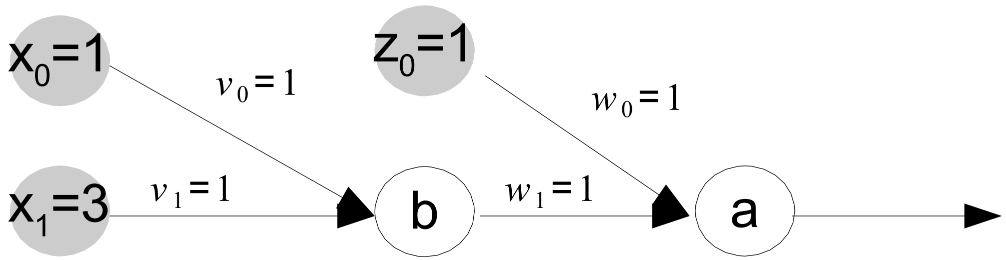
?

Example



x_0=1
x_1=3
z_0=1
$v_0 = 1$
$v_1 = 1$
$w_0 = 1$
$w_1 = 1$
b
a
$\frac{1}{1+e^{-4}} = 0.982$
$\frac{1}{1+e^{-1.982}} = 0.879$
t=0

Example



$$b=v_0 x_0+v_1 x \qquad z_1=\sigma(b) \qquad a=w_0 z_0+w_1 z_1 \qquad y=\sigma(a)$$

$$\frac{\partial E}{\partial w_0}=\frac{\partial E}{\partial a}\frac{\partial a}{\partial w_0}$$

chain rule

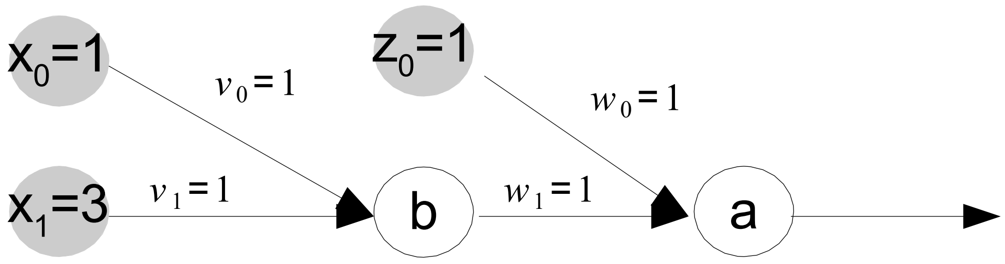
$$\frac{\partial E}{\partial a}=\frac{\partial}{\partial a}\frac{1}{2}(\sigma(a)-t)^2=(\sigma(a)-t)\cdot\sigma(a)(1-\sigma(a))$$

$$\frac{\partial a}{\partial w_0}=\frac{\partial}{\partial w_0}w_0 z_0+w_1 z= z_0$$

$$\boxed{\frac{\partial E}{\partial w_0}=(y-t)y(1-y)z_0}$$

$$\boxed{\frac{\partial E}{\partial w_1}=(y-t)y(1-y)z_1}$$

Example



$$b=v_0 x_0+v_1 x \qquad z_1=\sigma(b) \qquad a=w_0 z_0+w_1 z_1 \qquad y=\sigma(a)$$

$$a=w_0 z_0+w_1 \sigma(b)$$

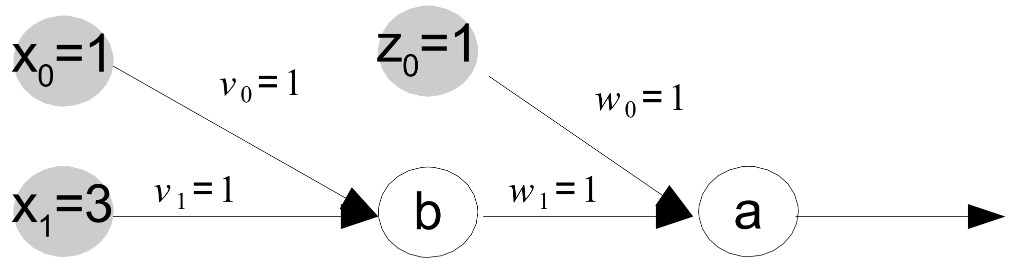
$$\frac{\partial E}{\partial v_0}=\frac{\partial E}{\partial a}\frac{\partial a}{\partial b}\frac{\partial b}{\partial v_0} \qquad \frac{\partial E}{\partial a}=(y-t)y(1-y)$$ from before

$$\frac{\partial a}{\partial b}=\frac{\partial}{\partial b}w_0 z_0+w_1\sigma(b)=w_1\sigma(b)(1-\sigma(b))=w_1 z_1(1-z_1)$$

$$\frac{\partial}{\partial v_0}v_0 x_0+v_1 x_1=x_0$$

$$\boxed{\frac{\partial E}{\partial v_0}=(y-t)y(1-y)w_1 z_1(1-z_1)x_0} \qquad \boxed{\frac{\partial E}{\partial v_1}=(y-t)y(1-y)w_1 z_1(1-z_1)x_1}$$

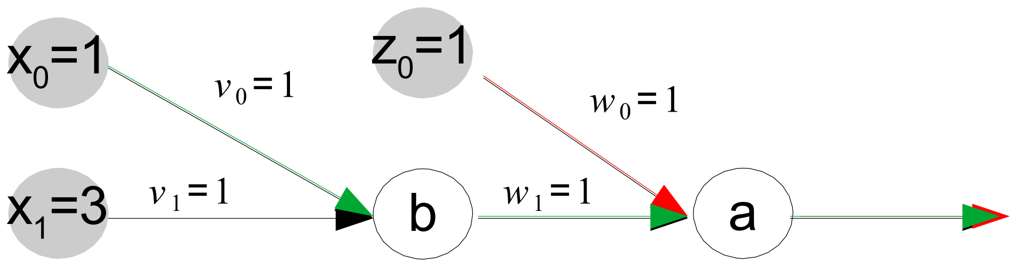
Example



$b = v_0 x_0 + v_1 x \qquad z_1 = \sigma(b) \qquad a = w_0 z_0 + w_1 z_1 \qquad y = \sigma(a)$

$$\nabla E = \begin{bmatrix} \dfrac{\partial E}{\partial w_0} \\ \dfrac{\partial E}{\partial w_1} \\ \dfrac{\partial E}{\partial v_0} \\ \dfrac{\partial E}{\partial v_1} \end{bmatrix} = \begin{bmatrix} (y-t)\,y(1-y)\,z_0 \\ (y-t)\,y(1-y)\,z_1 \\ (y-t)\,y(1-y)\,w_1 z_1 (1-z_1)\,x_0 \\ (y-t)\,y(1-y)\,w_1 z_1 (1-z_1)\,x_1 \end{bmatrix} \nabla E(\boldsymbol{w}) = \begin{bmatrix} 0.09 \\ 0.09 \\ 0.002 \\ 0.002 \end{bmatrix}$$

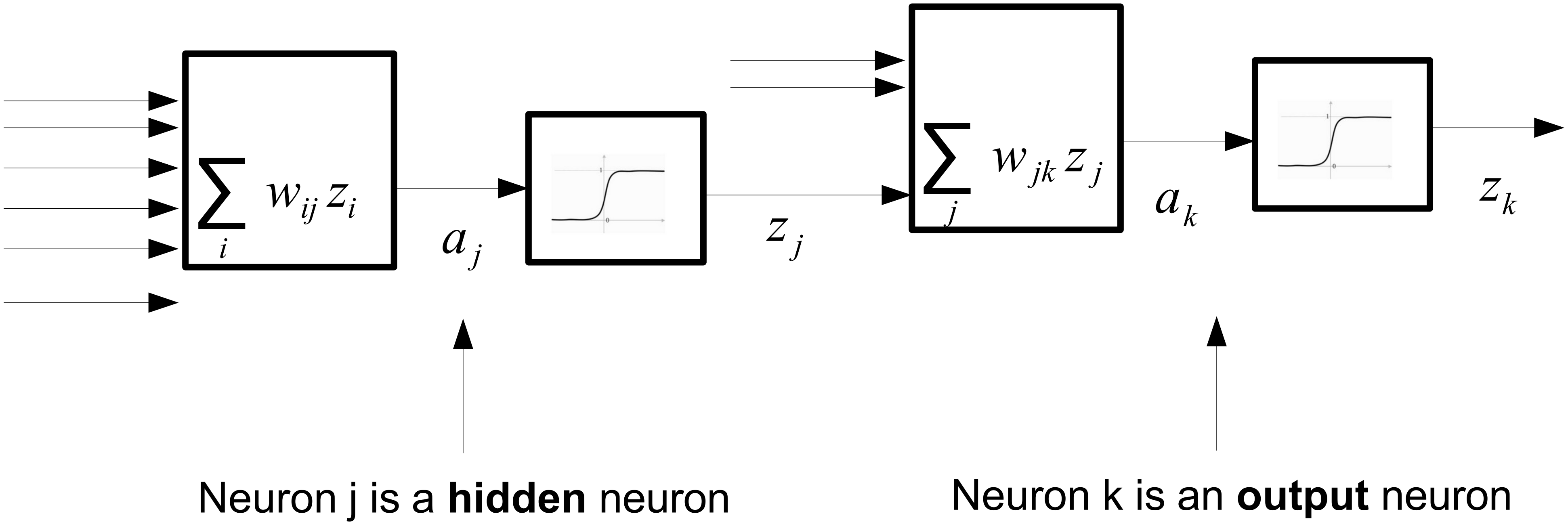
Summary



$$\frac{\partial E}{\partial w_0} = \frac{\partial E}{\partial a}\frac{\partial a}{\partial w_0}$$

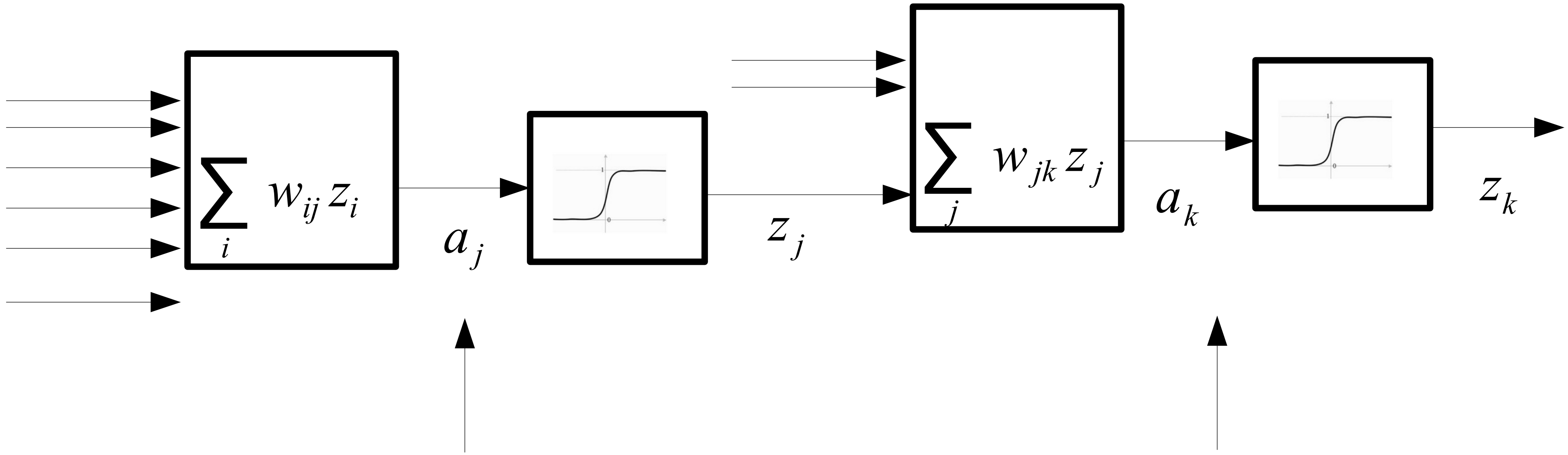
$$\frac{\partial E}{\partial v_0} = \frac{\partial E}{\partial a}\frac{\partial a}{\partial b}\frac{\partial b}{\partial v_0}$$

Backpropagation of errors, notation



$$\ldots \qquad a_j = \sum_{i=1}^{N} w_{ij} z_i \qquad z_j = f(a_j) \qquad a_k = \sum_{j=1}^{M} w_{jk} z_j \qquad z_k = f(a_k)$$

Forward pass



Neuron j is a **hidden** neuron

Neuron k is an **output** neuron

... $a_j = \sum_{i=1}^{N} w_{ij} z_i$ $\quad z_j = f(a_j)$ $\quad a_k = \sum_{j=1}^{M} w_{jk} z_j$ $\quad z_k = f(a_k)$

Forward pass: compute all the z

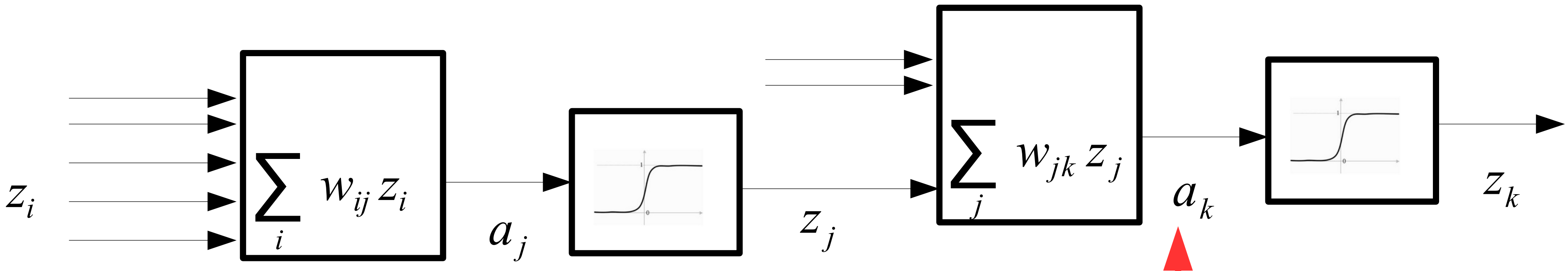
Backward pass, output neuron

How does a_k affect the error?

$$E(\boldsymbol{x}) = \frac{1}{2}(y-t)^2 = \frac{1}{2}(z_k - t)^2$$

$$\frac{\partial E}{\partial a_k} = \frac{\partial}{\partial a_k}\frac{1}{2}(z_k - t_k)^2 = \frac{\partial}{\partial a_k}\frac{1}{2}(\sigma(a_k) - t_k)^2 = (\sigma(a_k) - t_k)\sigma(a_k)(1-\sigma(a_k))$$

Now this useful, because it cancels out the exponent in the derivation



We call this derivative δ: $\delta_k = (z_k - t_k) z_k (1 - z_k)$

Backward pass

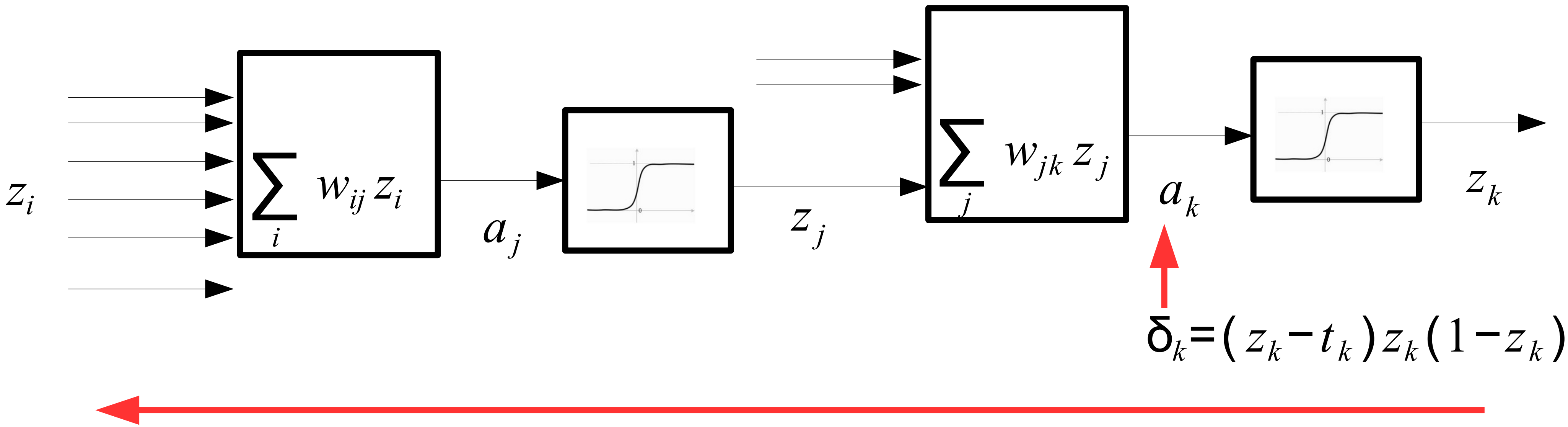
Backward pass, output neuron

One step backward, inside the box: how does w_{jk} affect the error?

$$E(\mathbf{x})=\frac{1}{2}(z_k-t)^2=\frac{1}{2}(\sigma(a_k)-t)^2 \qquad a_k=\sum_j w_{jk} z_j$$

We apply the chain rule again: $\frac{\partial E}{\partial w_{jk}}=\frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial w_{jk}}=\delta_k\cdot ?$

$$\frac{\partial a_k}{\partial w_{jk}}=\frac{\partial}{\partial w_{jk}} w_{0k} z_0+w_{1k} z_1+w_{2k} z_2+\cdots+w_{jk} z_j=?$$

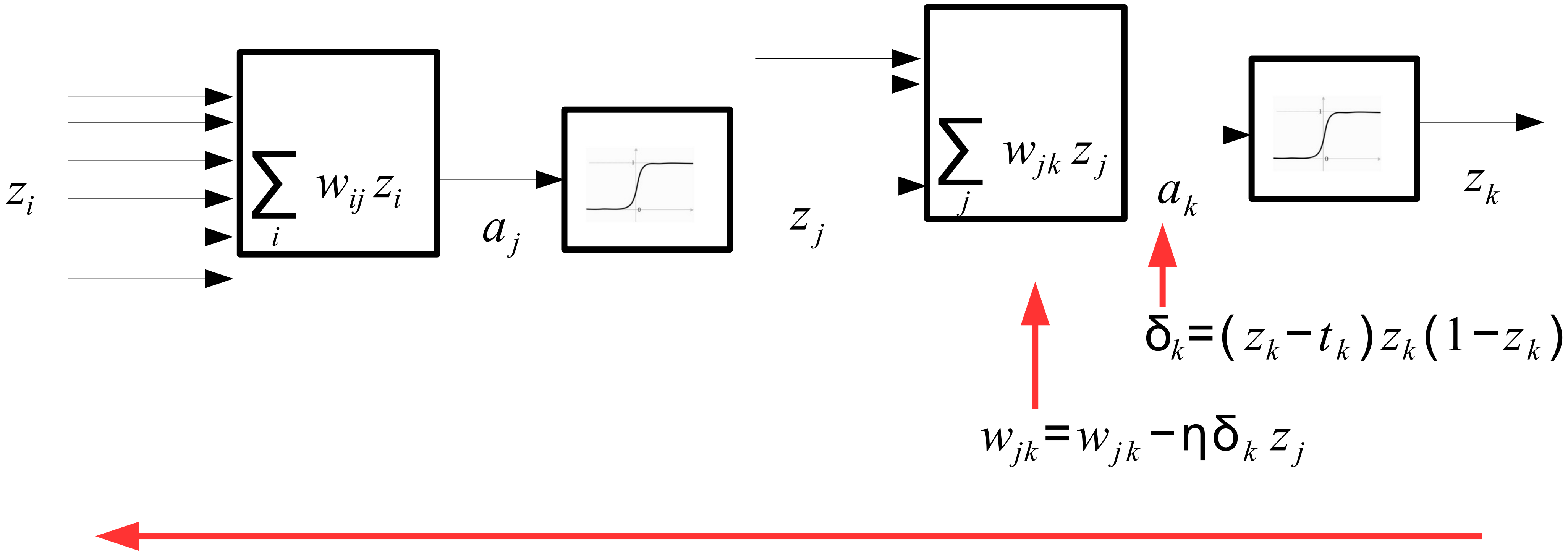


Backward pass, output neuron

One step backward, inside the box: how does w_{jk} affect the error?

We apply the chain rule again: $\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial w_{jk}} = \delta_k z_j$

$$\frac{\partial a_k}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} w_{0k} z_0 + w_{1k} z_1 + w_{2k} z_2 + \cdots + w_{jk} z_j = z_j$$

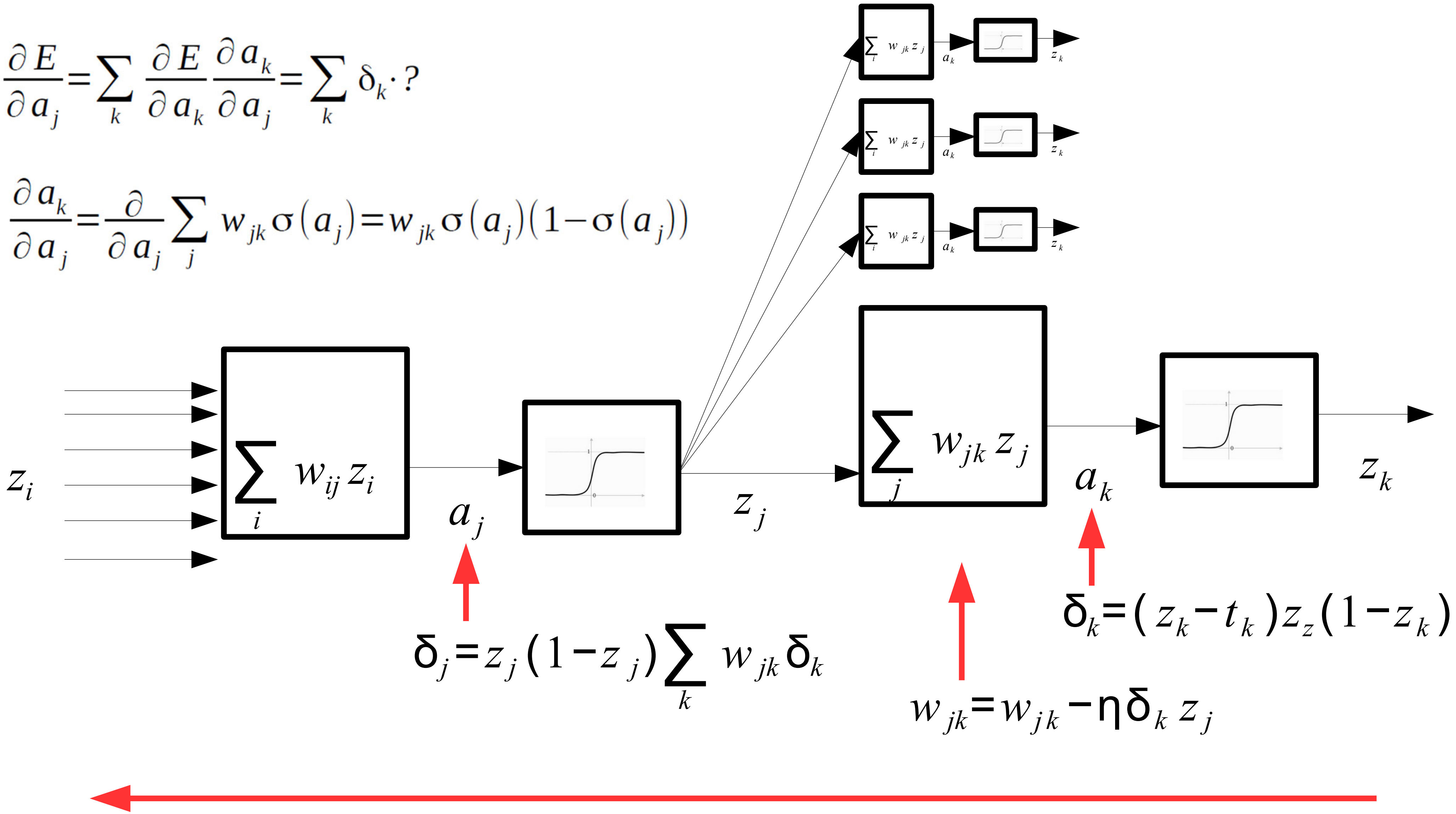


Backward pass, hidden neuron

One step backward: how does a_j affect the error?

$$\frac{\partial E}{\partial a_j} = \sum_k \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial a_j} = \sum_k \delta_k \cdot ?$$

$$\frac{\partial a_k}{\partial a_j} = \frac{\partial}{\partial a_j} \sum_j w_{jk} \sigma(a_j) = w_{jk} \sigma(a_j)(1 - \sigma(a_j))$$



$$\delta_j = z_j (1 - z_j) \sum_k w_{jk} \delta_k$$

$$w_{jk} = w_{jk} - \eta \delta_k z_j$$

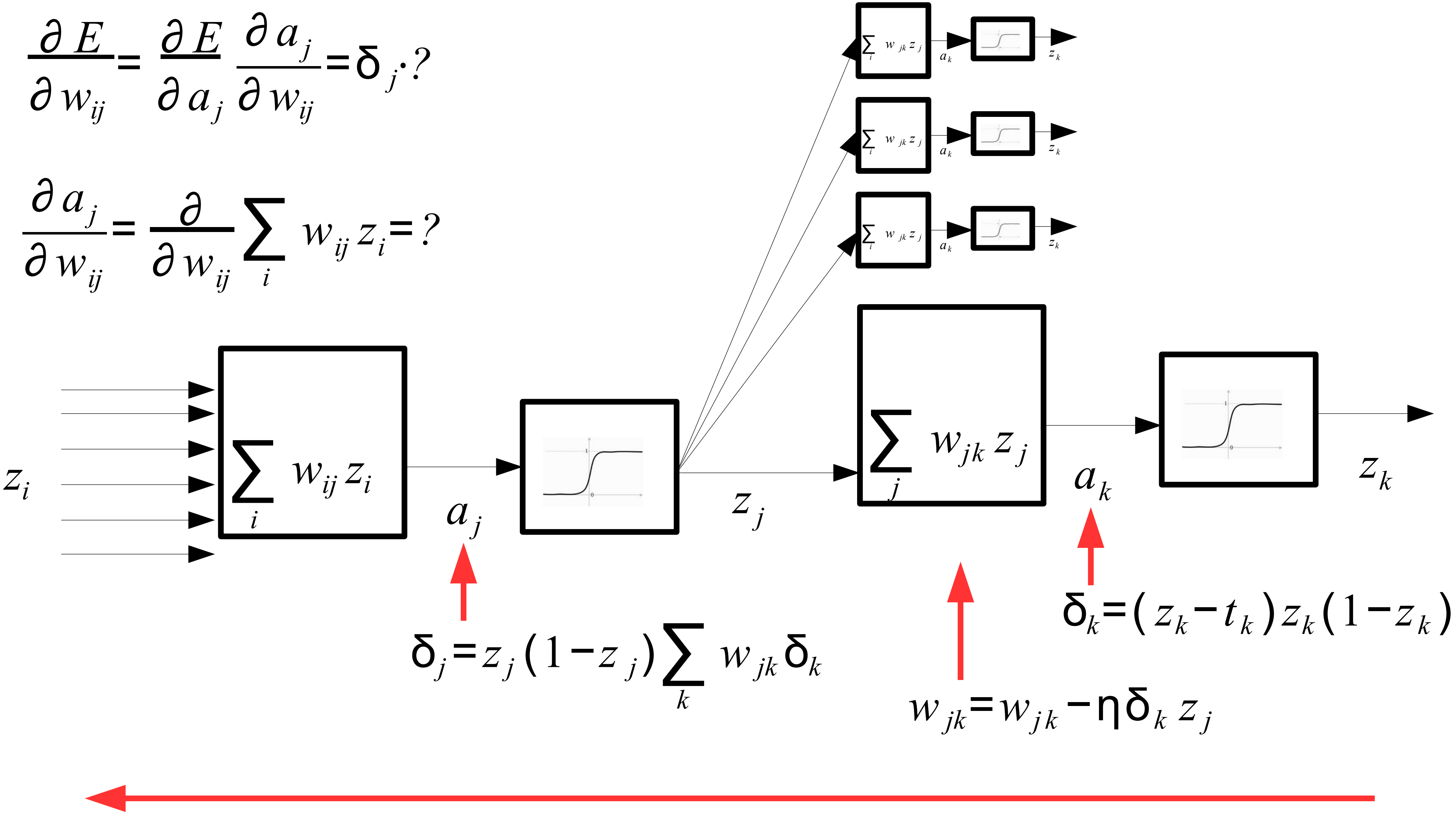
$$\delta_k = (z_k - t_k) z_z (1 - z_k)$$

Computing delta

One step backward, inside the box: how does w_{ij} affect the error?

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial w_{ij}} = \delta_j \cdot ?$$

$$\frac{\partial a_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}}\sum_i w_{ij} z_i = ?$$



$$\delta_j = z_j(1 - z_j)\sum_k w_{jk}\delta_k$$

$$w_{jk} = w_{jk} - \eta\delta_k z_j$$

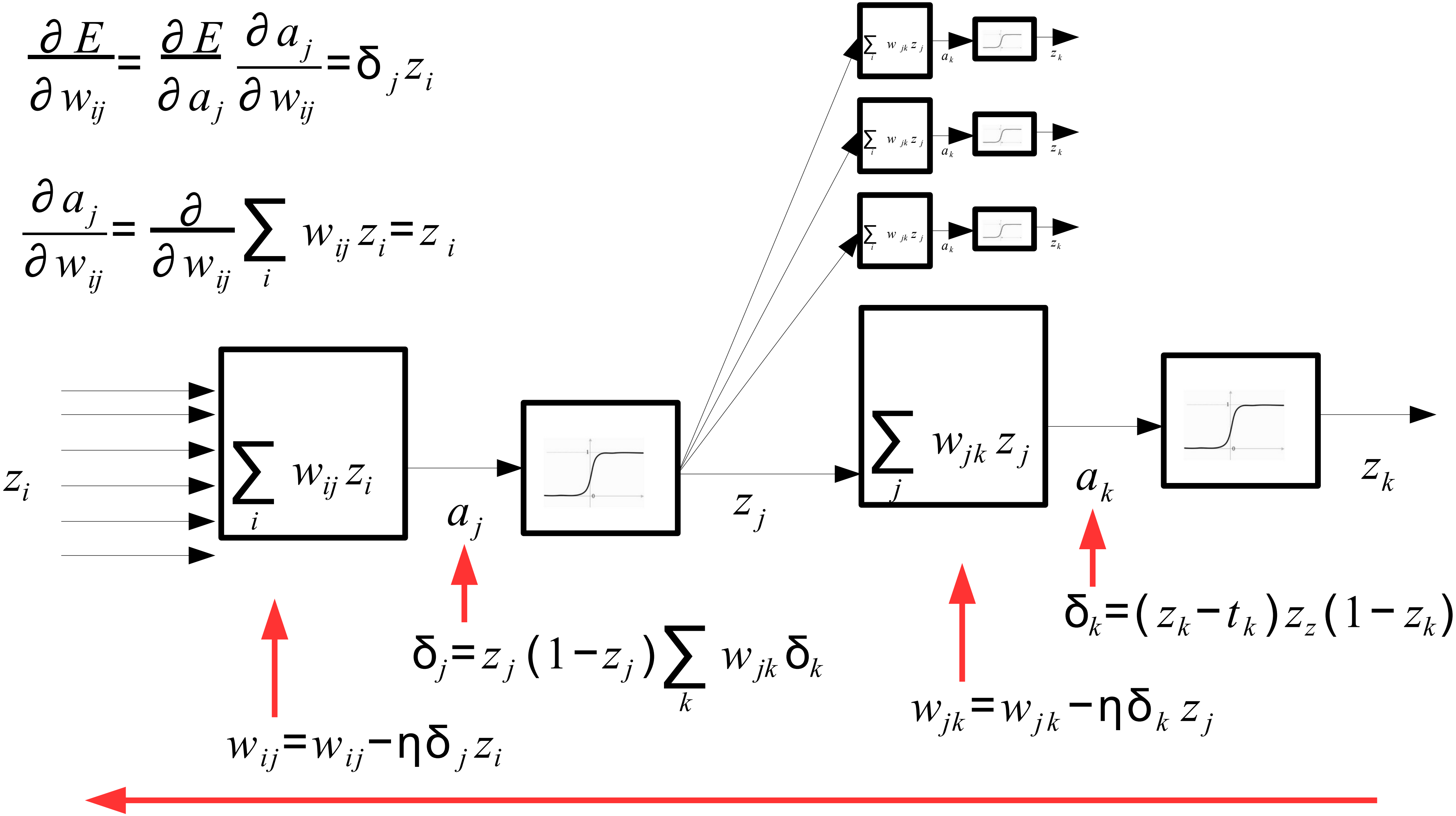
$$\delta_k = (z_k - t_k) z_k (1 - z_k)$$

Computing delta

One step backward, inside the box: how does w_{ij} affect the error?

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial w_{ij}} = \delta_j z_i$$

$$\frac{\partial a_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_i w_{ij} z_i = z_i$$



$$\delta_j = z_j (1 - z_j) \sum_k w_{jk} \delta_k$$

$$\delta_k = (z_k - t_k) z_z (1 - z_k)$$

$$w_{jk} = w_{jk} - \eta \delta_k z_j$$

$$w_{ij} = w_{ij} - \eta \delta_j z_i$$

Gradient descent (again)

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \nabla E(\boldsymbol{x})$$

Perceptron

$$E_p(\boldsymbol{X}) = \sum_{\boldsymbol{x}_n \in \boldsymbol{X}} \boldsymbol{w}^t \boldsymbol{x}_n (y_n - t_n)$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta (y - t) \boldsymbol{x}$$

Multi-Layer P

$$E_m(\boldsymbol{X}) = \frac{1}{2} \sum_{\boldsymbol{x}_n \in \boldsymbol{X}} (y_n - t_n)^2$$

Output:

$$\delta_{\text{output}} = (y - t) y (1 - y)$$

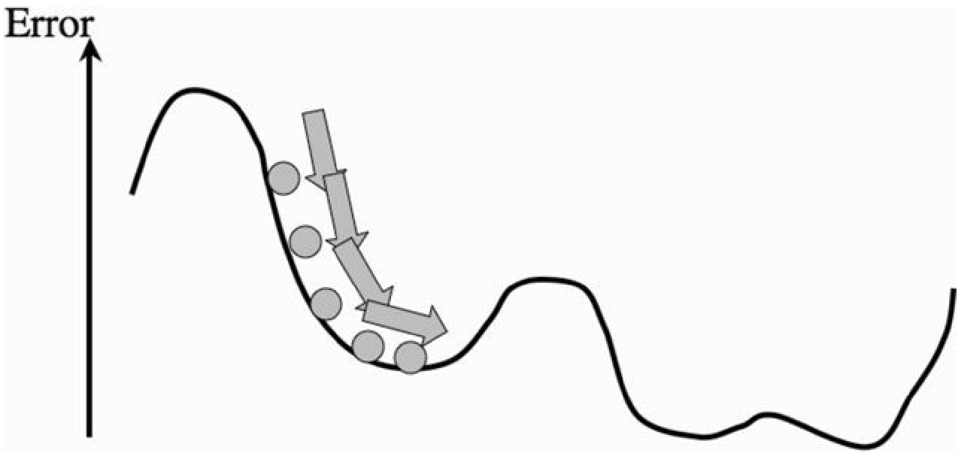
$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \delta_{\text{output}} \boldsymbol{x}$$

Hidden:

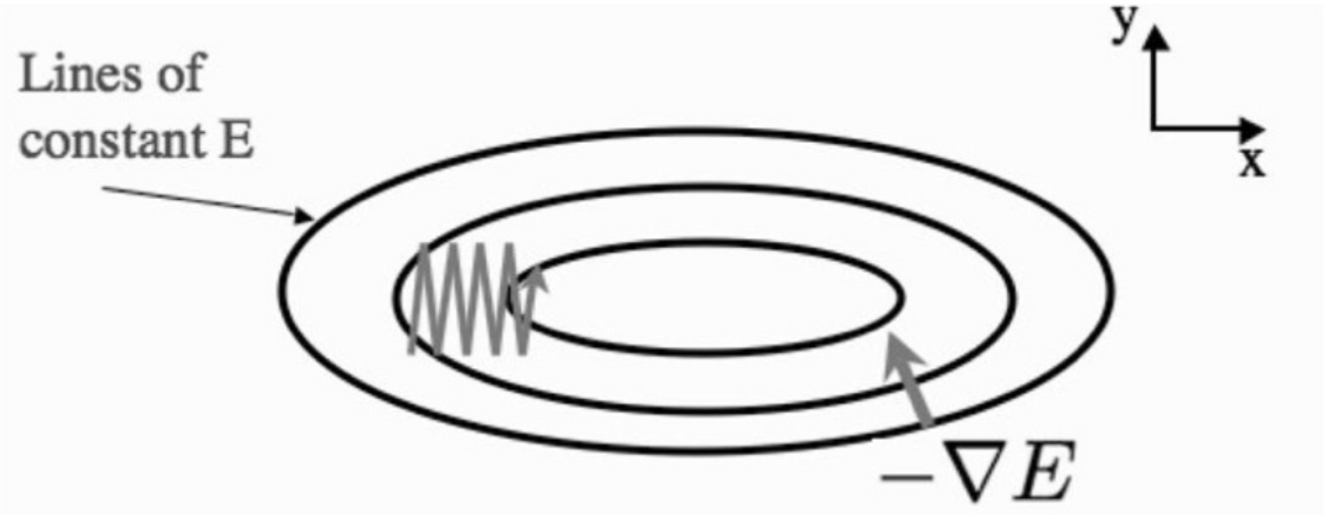
$$\delta_{\text{hidden}} = \sum_k w_k \delta_k$$

$$\boldsymbol{v}_{t+1} = \boldsymbol{v}_t - \eta \delta_{\text{hidden}} \boldsymbol{x}$$

Local Minima



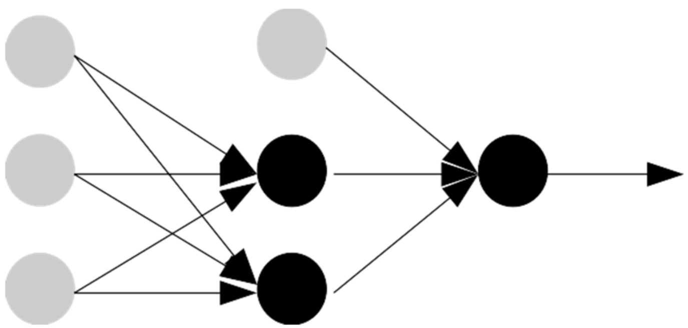
Start with weights close to 0: where the decision is actually made



Multiple random restarts

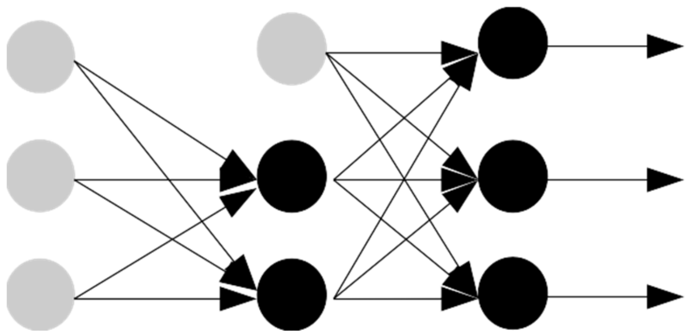
Using MLPs

Regression



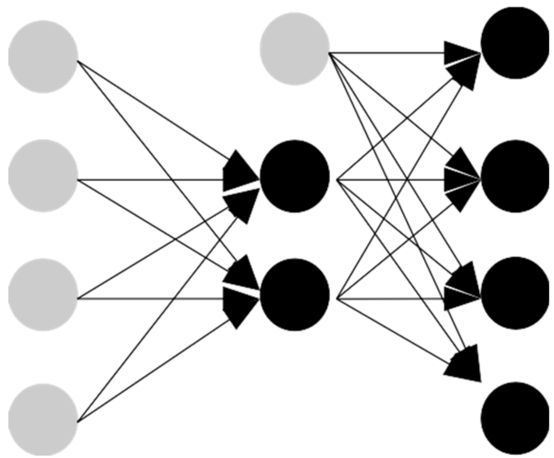
Last neuron
linear

Classification



One output
per class, pick
highest

Compression
(Autoencoder)



Middle
"bottleneck"
layer

Training "recipe"

Choose features

Normalize (rescale) data:

$$x' = \frac{x - \bar{x}}{\sigma}$$

Zero mean, unit variance

or

$$x' = \frac{x - min(x)}{max(x) - min(x)}$$

in [0,1]

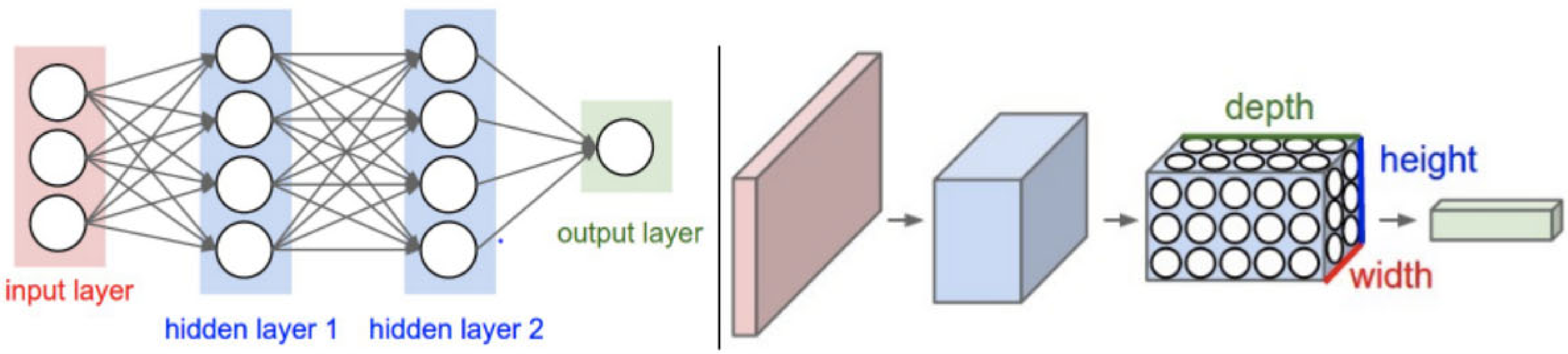
Create training, validation, and test sets

Decide whether you need hidden layers and how big.
Try several ones.

Train

Test

Deep learning



MLP

CNN