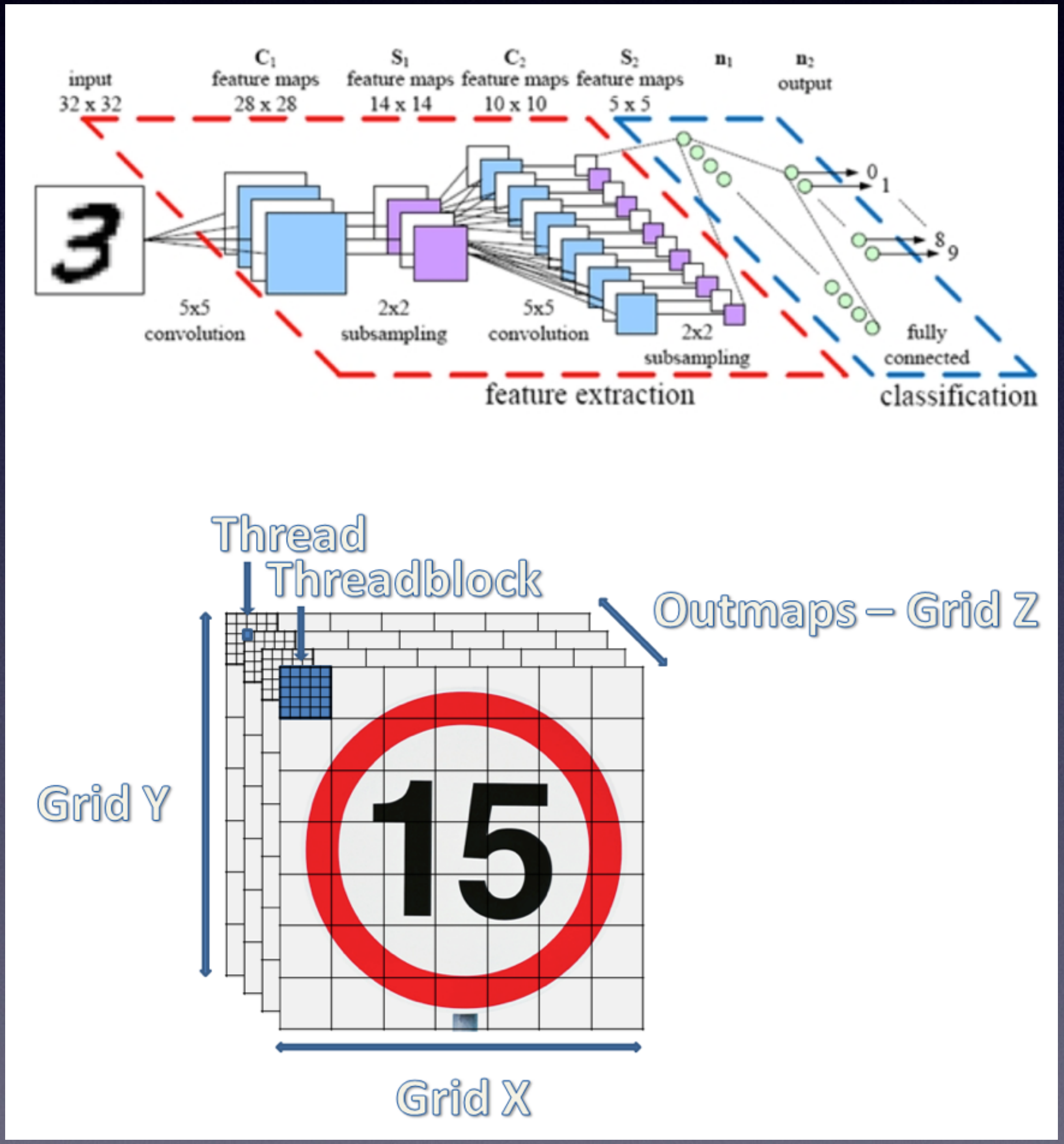
Multi Layer Perceptron

Computer & Robot Vision Lab
Sung -ju Kim
goddoe2@gmail.com

Content

- Why Neural Net came back?
- Single Layer Perceptron
- Multi Layer Perceptron
- Traffic Sign Lane Guessing



Why Neural Net Came back?

Table 2
Result overview for the final stage of the GTSRB.

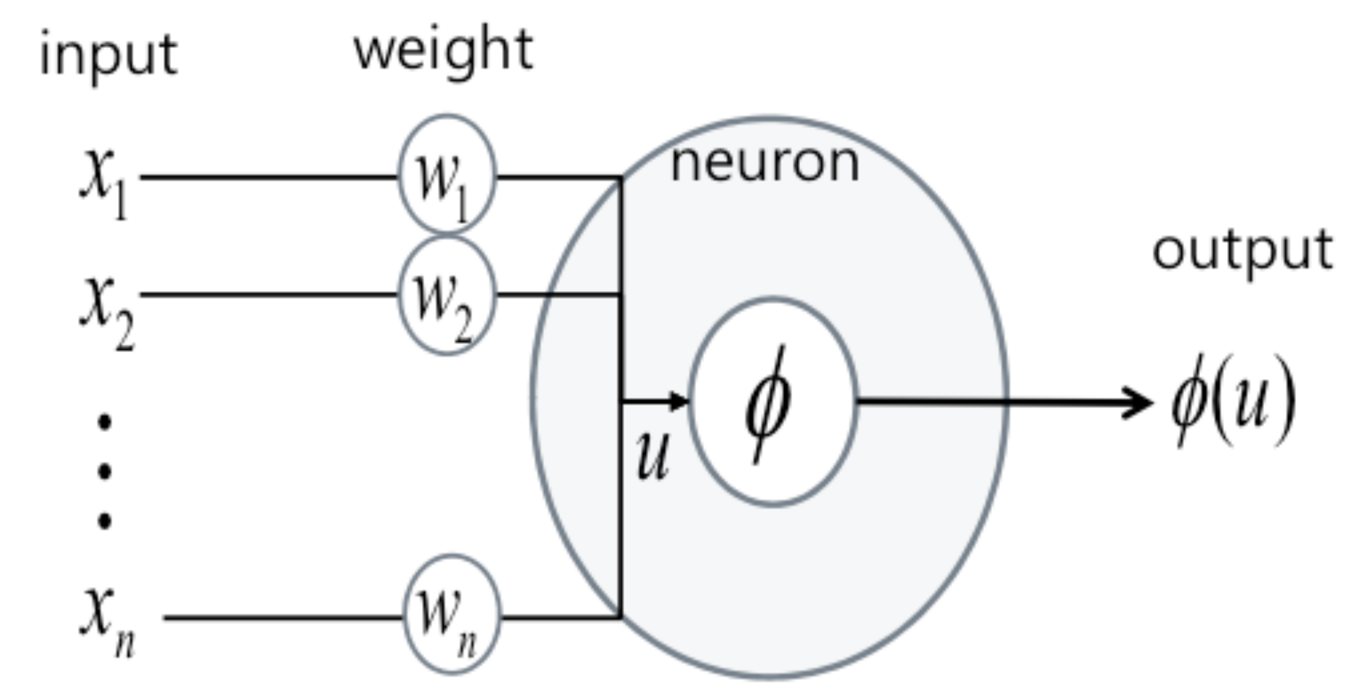
CCR (%)	Team	Method
99.46	IDSIA	Committee of CNNs
99.22	INI-RTCV	Human (best individual)
98.84	INI-RTCV	Human (average)
98.31	Sermanet	Multi-scale CNN
96.14	CAOR	Random forests
95.68	INI-RTCV	LDA (HOG 2)
93.18	INI-RTCV	LDA (HOG 1)
92.34	INI-RTCV	LDA (HOG 3)

Face net: 99.6%
Deep Face : 97.25%

Stallkamp, Johannes, et al. "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition." *Neural networks* 32 (2012): 323-332.

Single Layer Perceptron

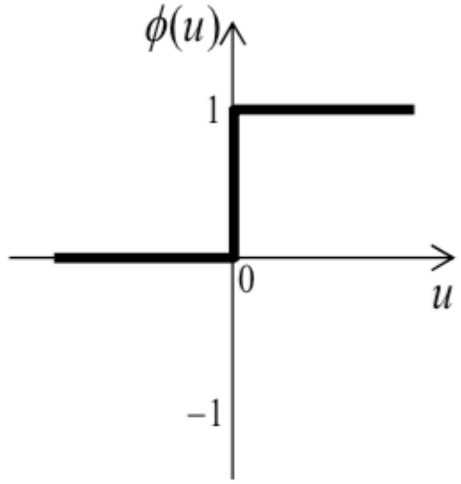
Feed Forward



$$u = \sum_{i=1}^{n} w_i x_i, \qquad \phi(u) = \begin{cases} 1 & \text{if } u \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

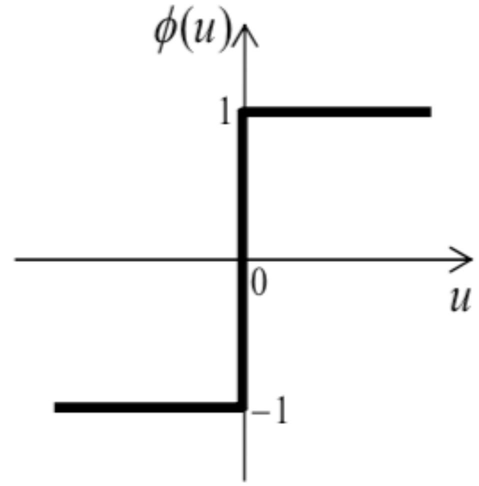
Activation Functions

step function



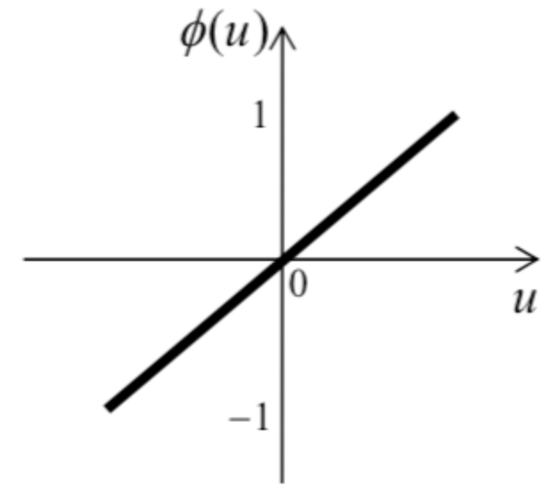
$$\phi_{step}(u) = \begin{cases} 1 & \text{if } \; u \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

sign function

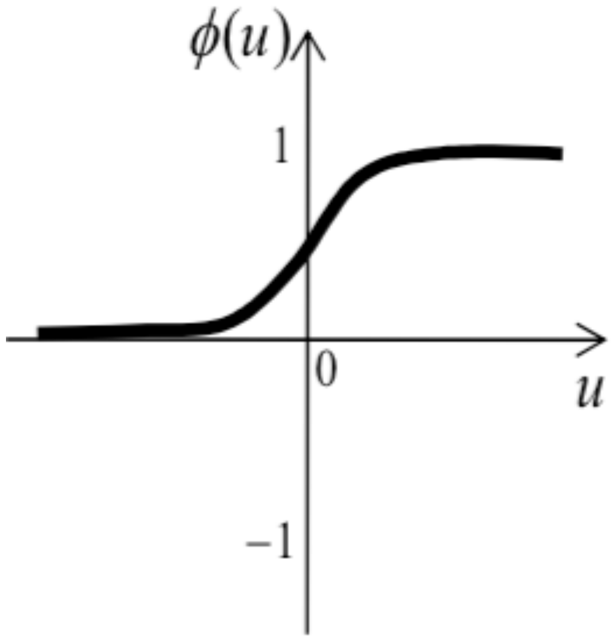


$$\phi_{sign}(u) = \begin{cases} 1 & \text{if } \; u \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

identity function

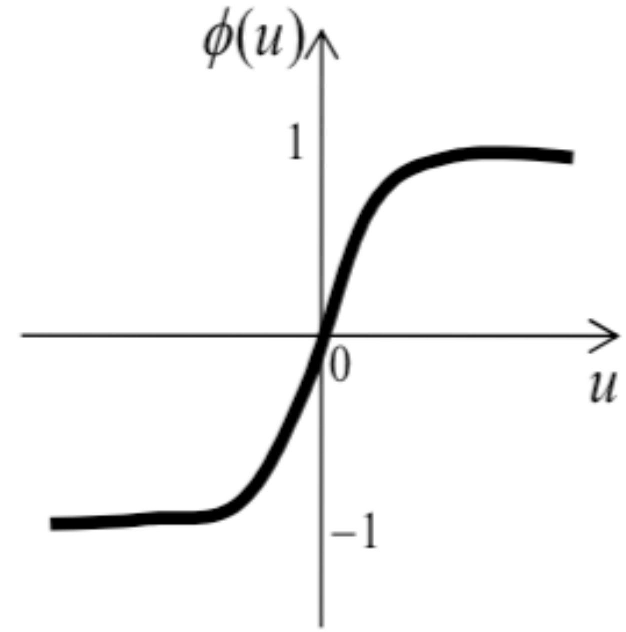


$$\phi_{id}(u) = u$$



$$\phi_{sig}(u) = \frac{1}{1+e^{-u}}$$

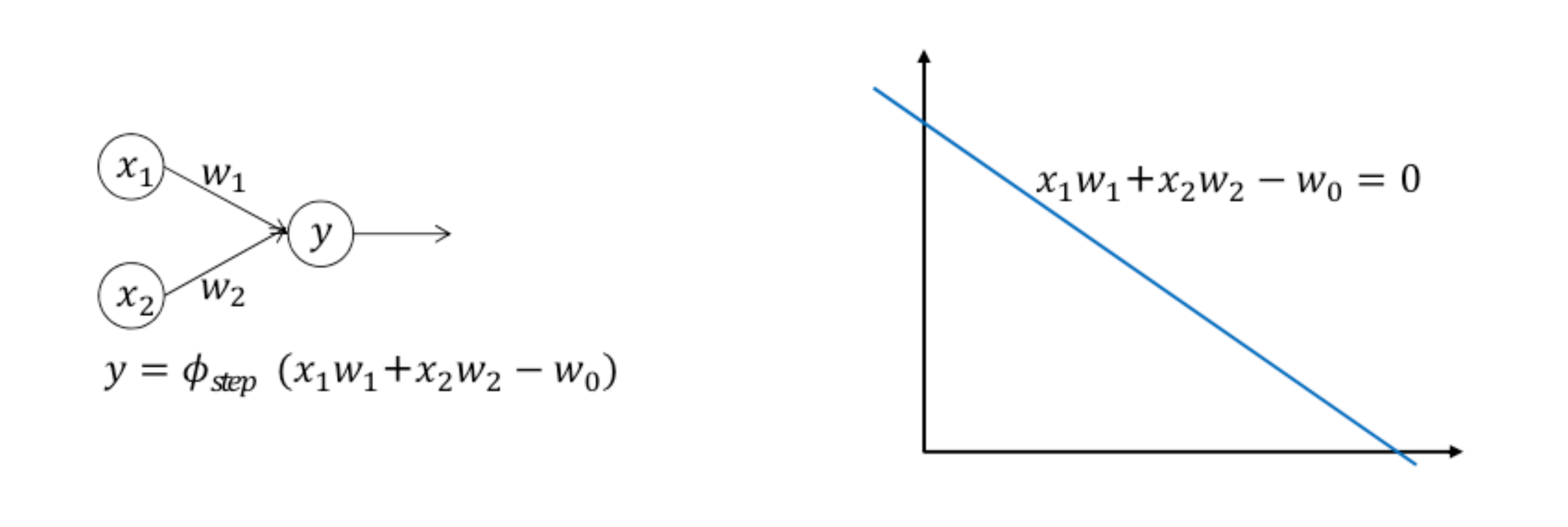
sigmoid function



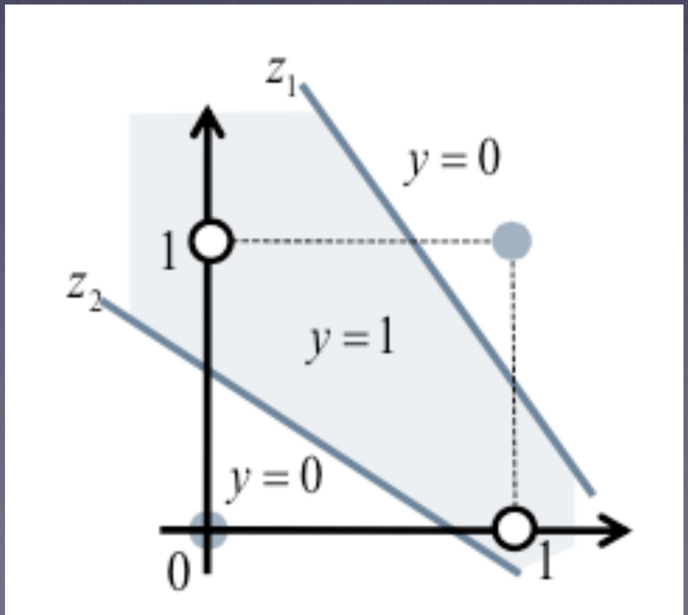
$$\phi_h = \frac{e^u - 1}{e^u + 1}$$

hyper tangent function

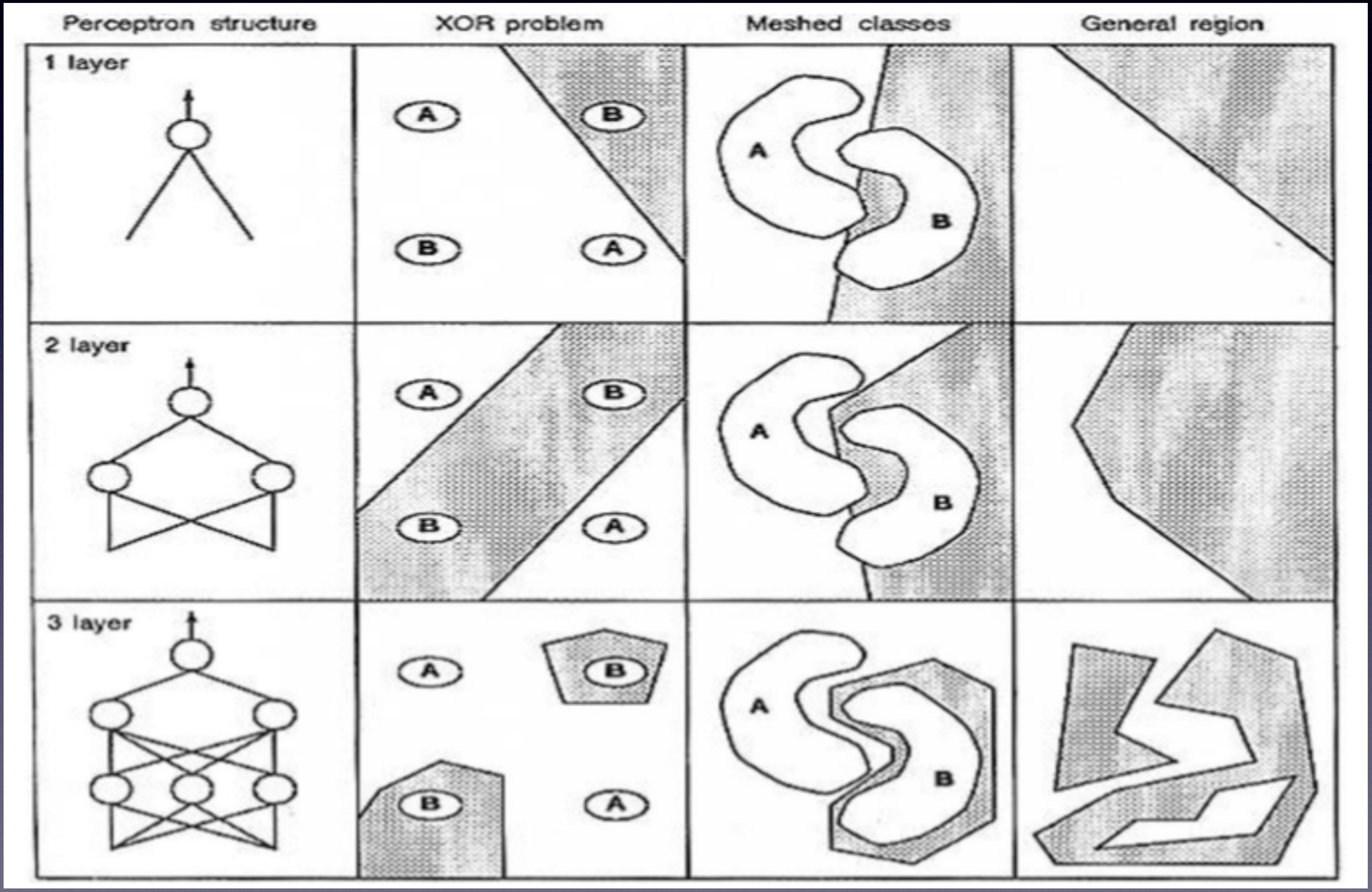
Limitation of Single Layer Perceptron



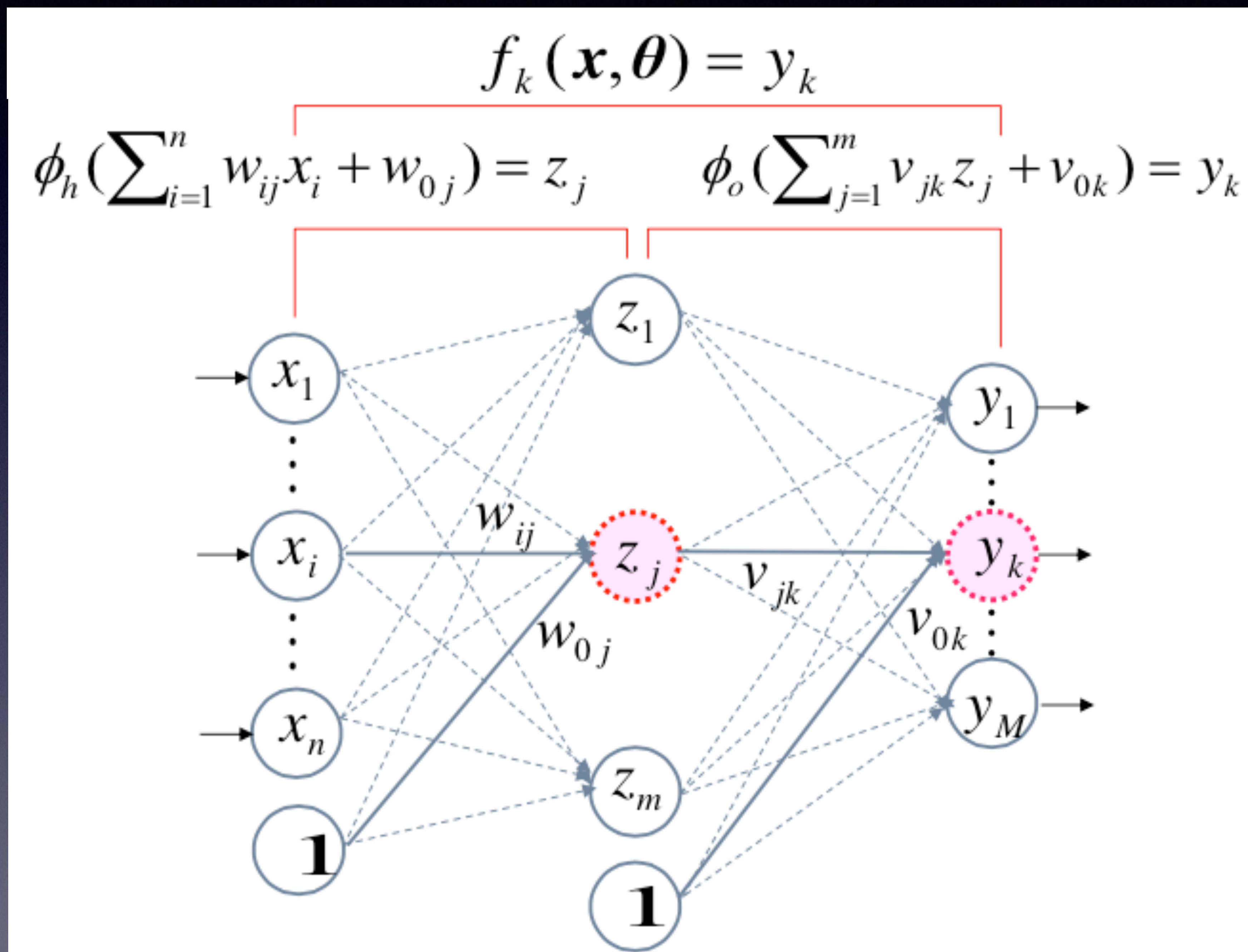
But Single Layer Perceptron cannot classify XOR Problem



Multi Layer Perceptron



Multi Layer Perceptron

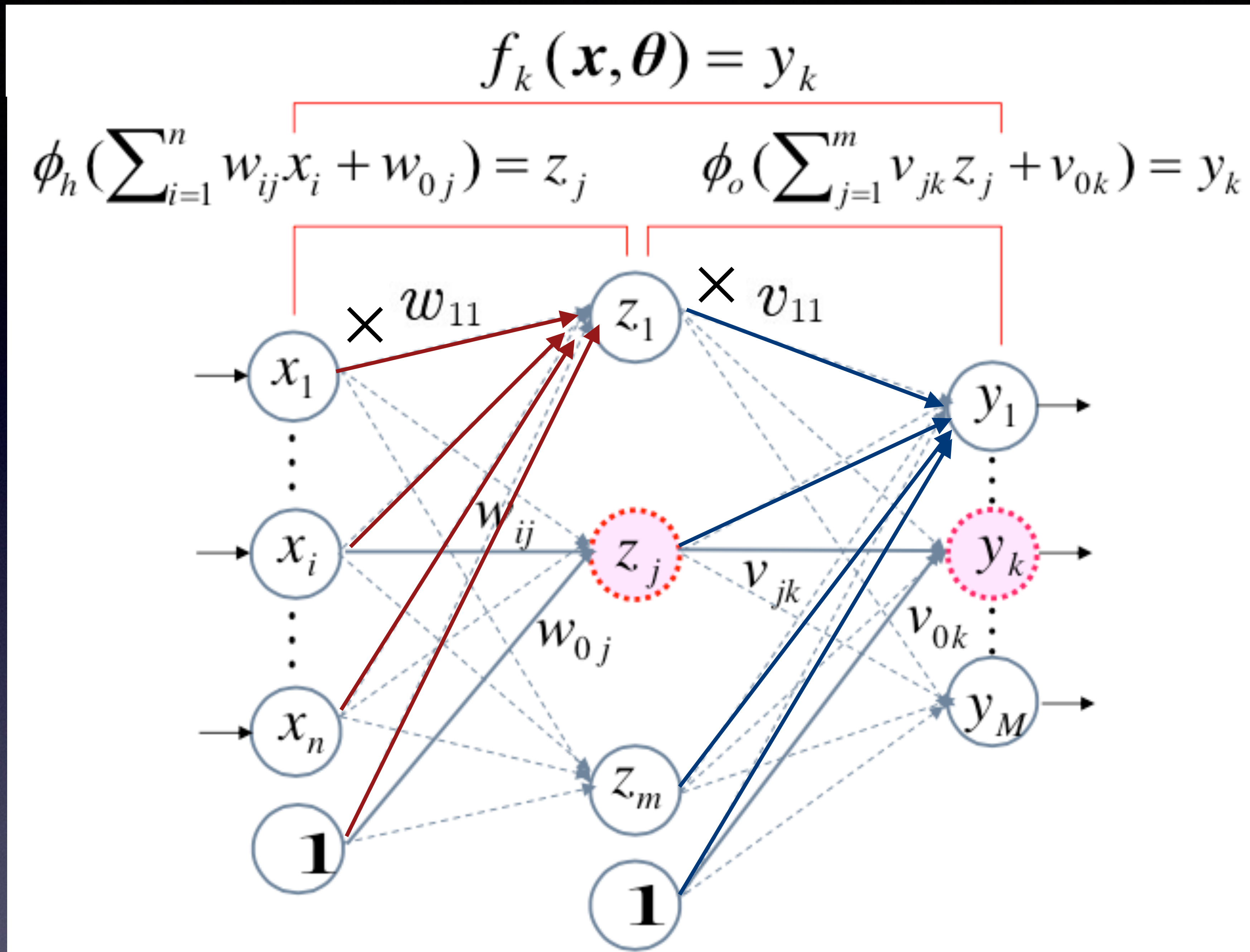


Input Layer

Hidden Layer

Output Layer

Feed Forward



Ex) $z_1 = \phi_h(x_1 w_{11} + x_2 w_{21} + \ldots + x_i w_{i1} + \ldots + x_n w_{n1} + 1 w_{01})$

$y_1 = \phi_h(z_1 v_{11} + z_2 v_{21} + \ldots + z_j v_{j1} + \ldots + z_m v_{m1} + 1 v_{01})$

$\phi_h(x) = \tanh(x)$

How to Learn Perceptron?

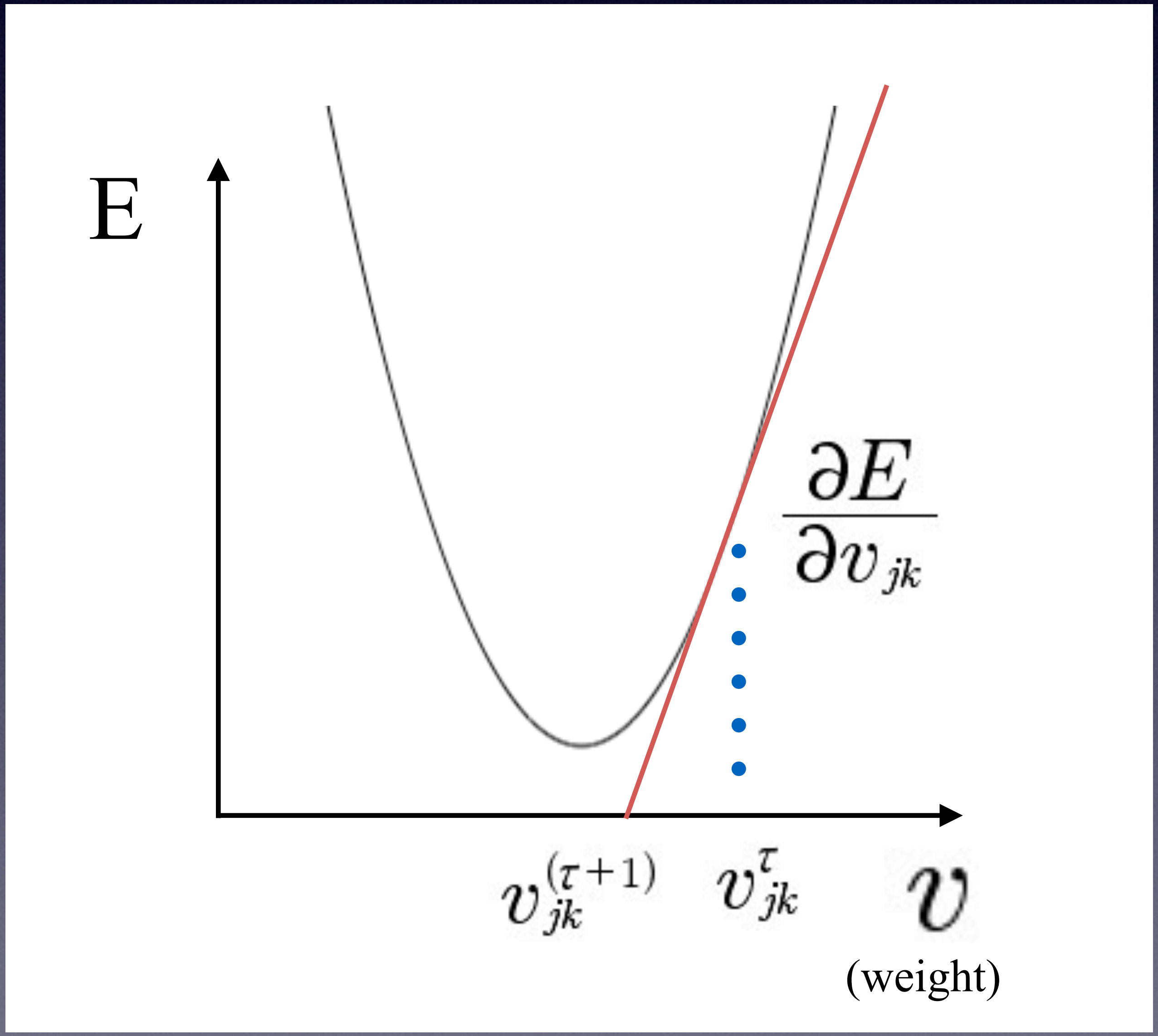
Error Function

$$E = \sum_{k=1}^{M} \frac{1}{2} (t_k - y_k)^2$$

E	Error Function
t_j	Target Value
y_j	Output Value

How to Learn Perceptron?

Delta Learning Rule



How to Learn Perceptron?

Delta Learning Rule

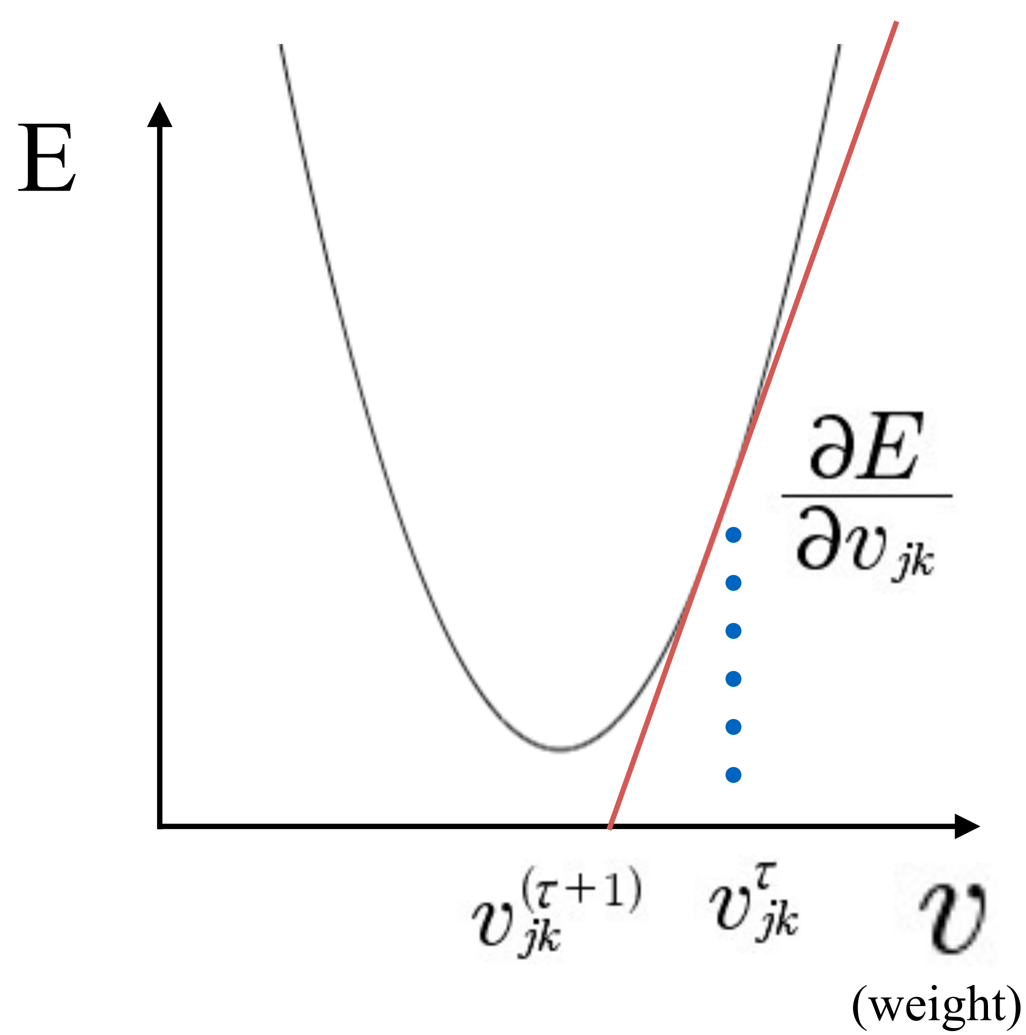
$$v_{jk}^{(\tau+1)} = v_{jk}^{\tau} + \Delta v_{jk}$$

$$\Delta v_{jk} = -\eta \frac{\partial E}{\partial v_{jk}}$$

$v_{jk}^{(\tau+1)}$	new weight
v_{jk}^{τ}	current weight
η	learning rate
E	Error Function

How to Learn Perceptron?

Delta Learning Rule

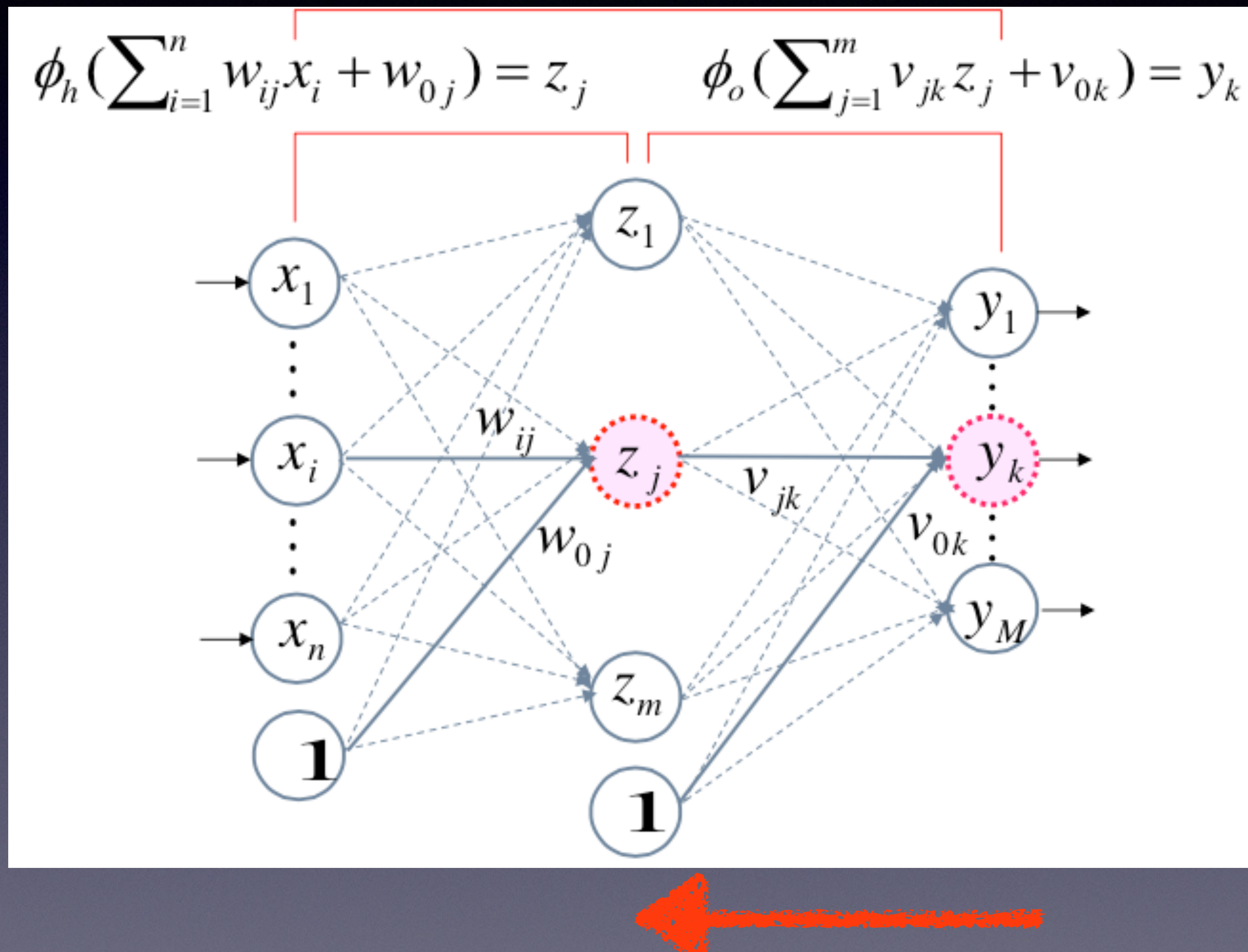


$$v_{jk}^{(\tau+1)} = v_{jk}^{\tau} + \Delta v_{jk}$$

$$\Delta v_{jk} = -\eta \frac{\partial E}{\partial v_{jk}}$$

$v_{jk}^{(\tau+1)}$	new weight
v_{jk}^{τ}	current weight
η	learning rate
E	Error Function

Back Propagation



Input Layer

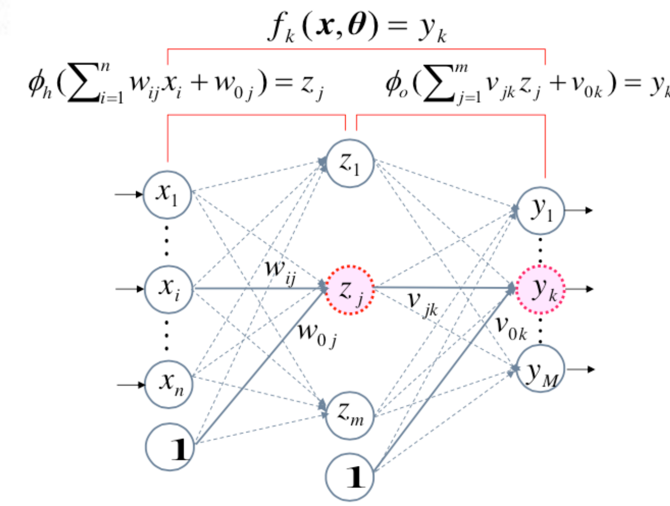
Hidden Layer

Output Layer

$$E = \sum_{k=1}^{M} \frac{1}{2}(t_k - y_k)^2 = \frac{1}{2}\sum_{k=1}^{M}\left\{t_k - \phi_h\Big(\underbrace{\sum_{j=1}^{m} v_{jk} z_j + v_{0k}}_{u_k^o}\Big)\right\}^2 \quad \cdots (1)$$

$$v_{jk}^{(\tau+1)} = v_{jk}^{\tau} + \Delta v_{jk} \quad \cdots (2)$$

$$\Delta v_{jk} = -\eta \frac{\partial E}{\partial v_{jk}} \quad \cdots (3)$$

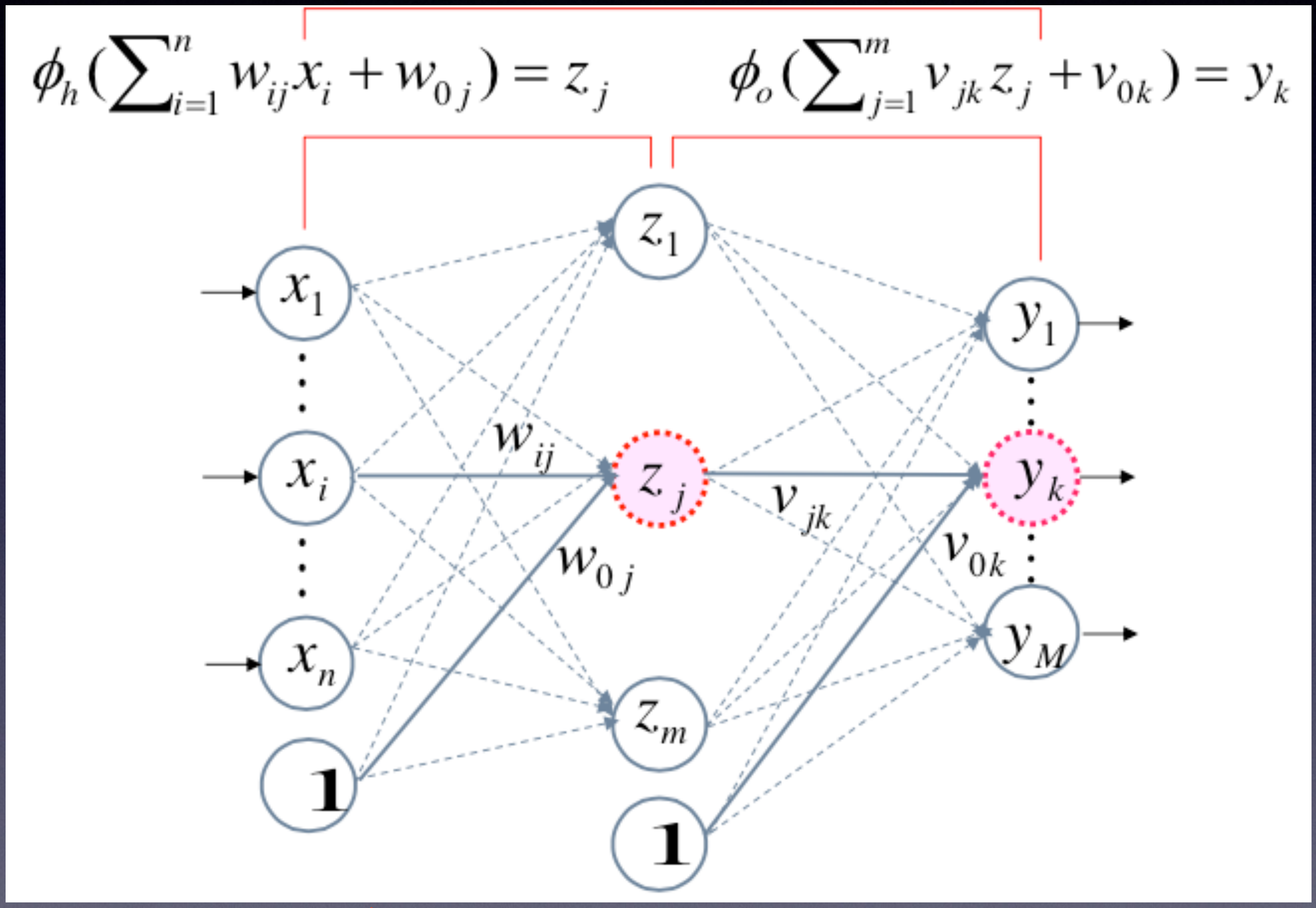


$$\frac{\partial E}{\partial v_{jk}} = \frac{\partial E}{\partial u_k^o}\frac{\partial u_k^o}{\partial v_{jk}} = \delta_k z_j = -\phi_h{}'(u_k^o)(t_k - y_k) z_j \quad \cdots (4)$$

$$\boxed{\therefore\ v_{jk}^{(\tau+1)} = v_{jk}^{\tau} + \eta \phi_h{}'(u_k^o)(t_k - y_k) z_j \quad \cdots (5)}$$

$v_{jk}^{(\tau+1)}$	new weight	η	learning rate	t_j	Target Value	$\phi_h(x) = \tanh(x)$
v_{jk}^{τ}	current weight	E	Error Function	y_j	Output Value	$\phi_h(x)' = (1-x)(1+x)$

Back Propagation



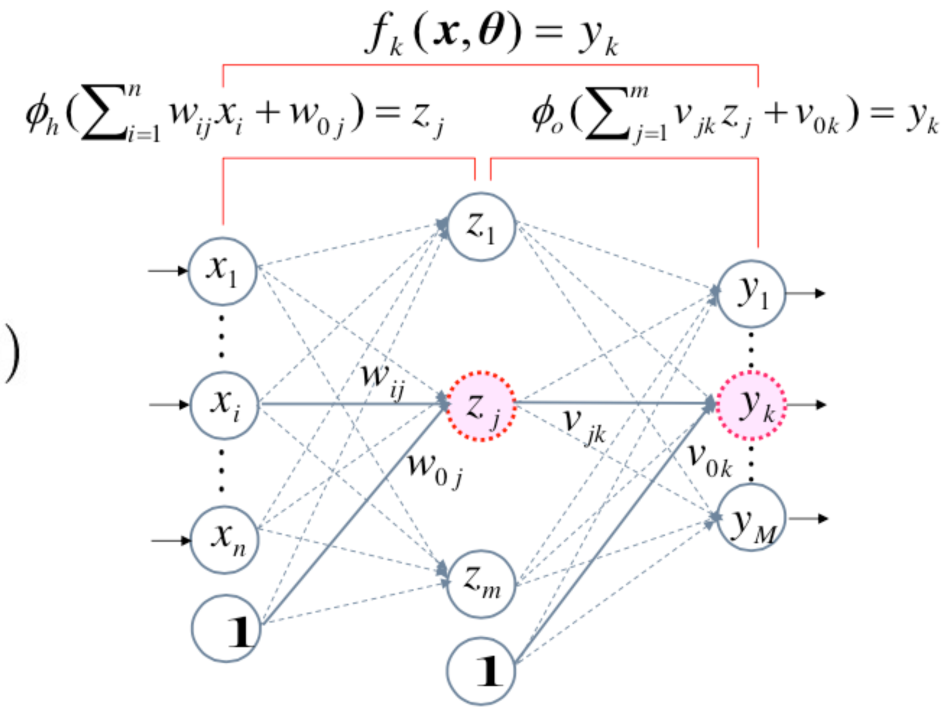
Input Layer

Hidden Layer

Output Layer

$$u_k^o = \sum_{j=1}^{m} v_{jk} z_j + v_{ok} = \sum_{j}^{m} v_{jk} \phi_h (u_j^h) + v_{ok} \quad \cdots (1)$$

$$w_{ij}^{(\tau+1)} = w_{ij}^{\tau} + \Delta w_{ij} \quad \cdots (2)$$



$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \frac{\partial E}{\partial u_j^h} \frac{\partial u_j^h}{\partial w_{ij}} \quad \cdots (3)$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial u_j^h} \frac{\partial u_j^h}{\partial w_{ij}} = \delta_j x_i \quad \cdots (4)$$

$$\delta_j = \frac{\partial E}{\partial u_j^h} = \sum_{k=1}^{M} \frac{\partial E}{\partial u_k^o} \frac{\partial u_k^o}{\partial u_j^h} = \phi_h{}' (u_j^h) \sum_{k=1}^{M} v_{jk} \delta_k \quad \cdots (5)$$

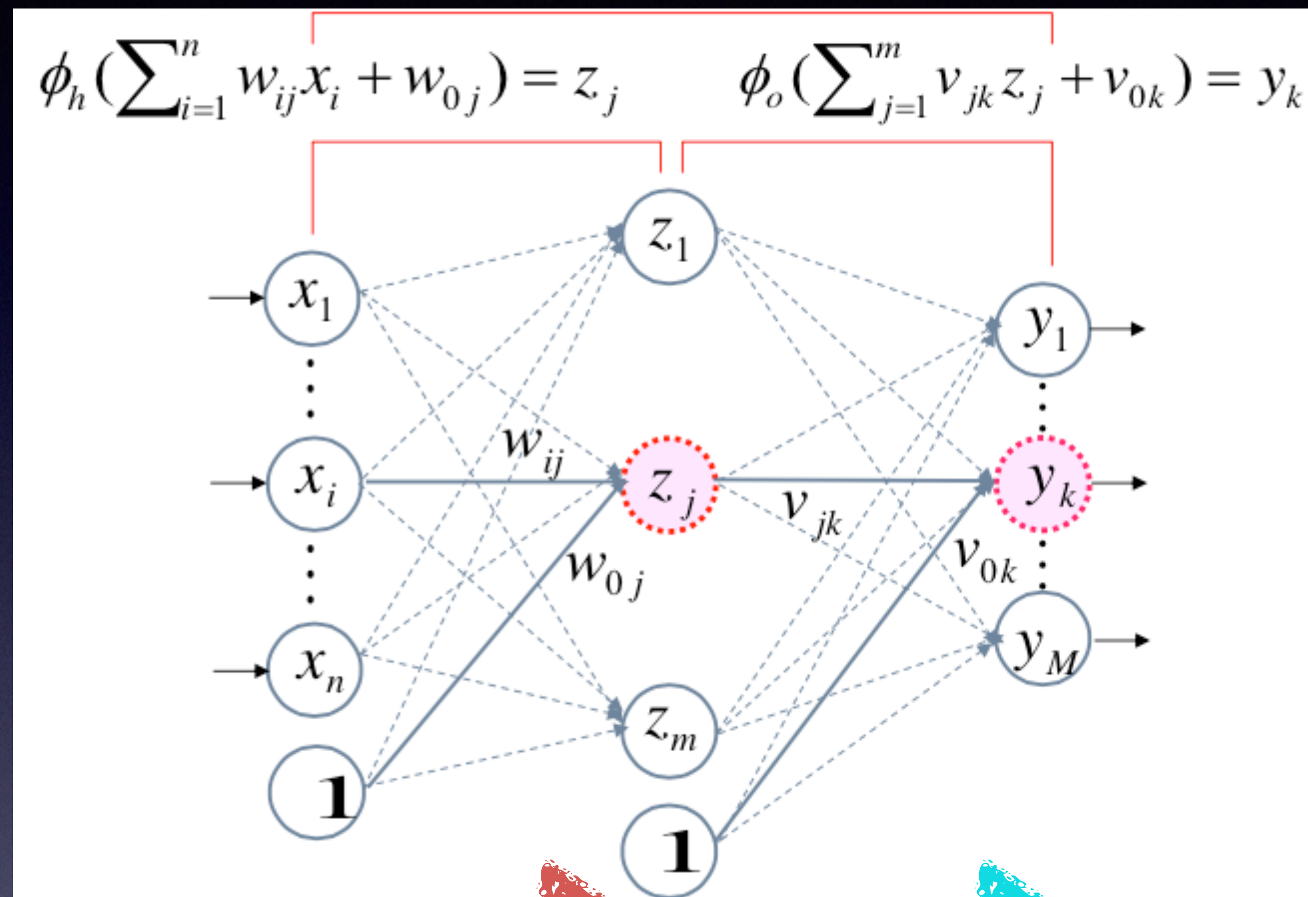
$$\boxed{\therefore\ w_{ij}^{(\tau+1)} = w_{ij}^{\tau} - \eta \left\{ \phi_h{}' (u_j^h) \sum_{k=1}^{M} v_{jk} \delta_k \right\} x_i \quad \cdots (6)}$$

$w_{jk}^{(\tau+1)}$ new weight $\quad \eta$ learning rate

w_{ij}^{τ} current weight $\quad E$ Error Function

$$\frac{\partial u_k^o}{\partial u_j^h} = \phi_h{}' (u_j^h) v_{jk} \qquad \frac{\partial E}{\partial u_k^o} = \delta_k$$

Good for Design in Parallels Architecture



$$\begin{bmatrix} w_{11} & w_{21} & \cdots & w_{(n+1)1} \\ w_{12} & w_{22} & \cdots & w_{(n+1)2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1m} & w_{2m} & \cdots & w_{(n+1)m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} u_1^h \\ u_2^h \\ \vdots \\ u_m^h \end{bmatrix}$$

$$\begin{bmatrix} v_{11} & v_{21} & \cdots & v_{(m+1)1} \\ v_{12} & v_{22} & \cdots & v_{(m+1)2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1M} & v_{2M} & \cdots & v_{(m+1)M} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} u_1^o \\ u_2^o \\ \vdots \\ u_M^o \end{bmatrix}$$