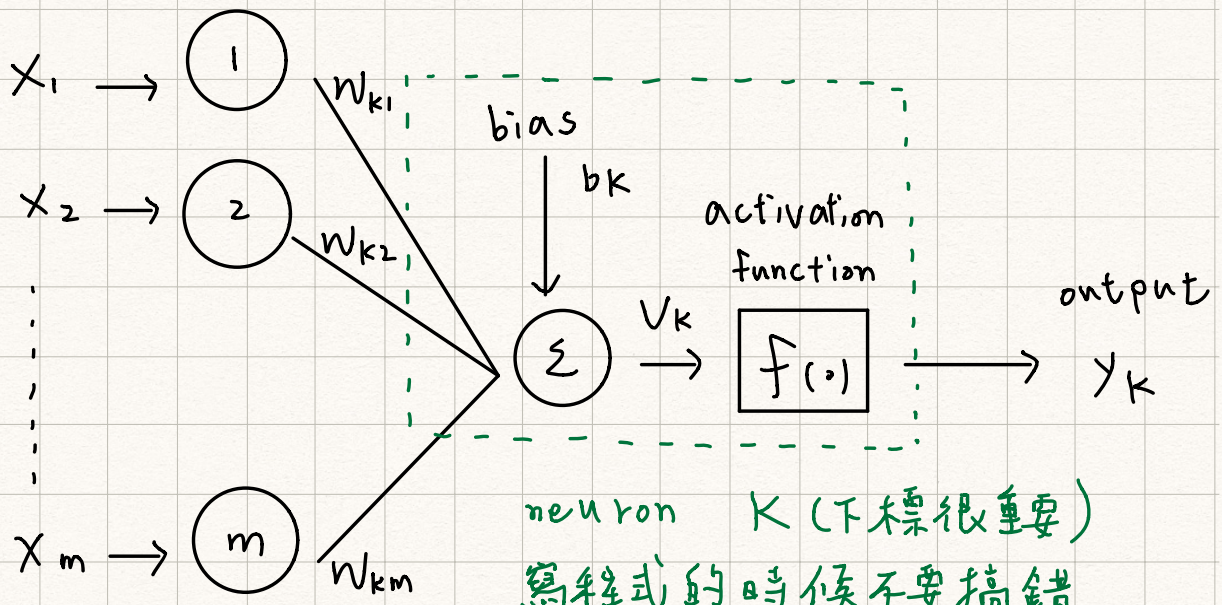
Δ. Neural Networks



neuron K (下標很重要)

寫程式的時候不要搞錯

(where $x_0 = 1$, $w_{k0} = b_k$)

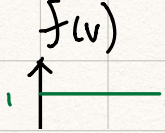
$$y_k = f(v_k)$$

$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k = \sum_{j=0}^{m} w_{kj} x_j$$

(全部的輸入和)

Δ. Types of activation function (f(v))

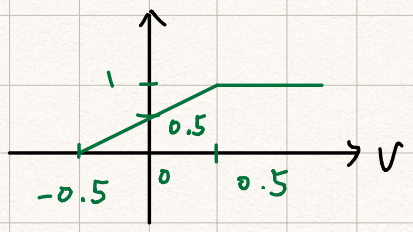
1. Threshold function



$$f(v) = \begin{cases} 1, & v \geq 0 \end{cases}$$

$0, v < 0$

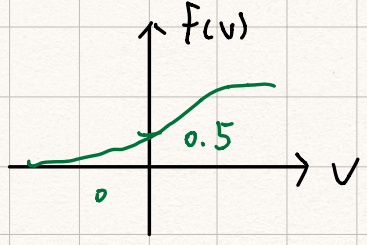
2. Piecewise-linear function



$$f(v) = \begin{cases} 1 & v \geq 0.5 \\ v + 0.5 & 0.5 > v > -0.5 \\ 0 & v \leq -0.5 \end{cases}$$

3. Sigmoid function

最常用

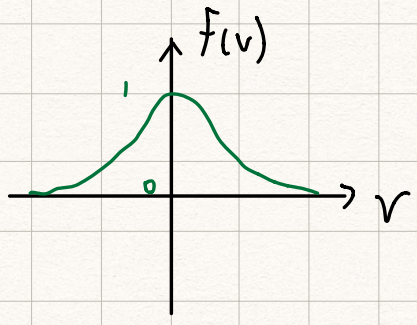


$$f(v) = \frac{1}{1 + \exp(-av)}$$

a is a constant

(表斜率)，通常令 a=1 方便計算

4. Gaussian function



$$f(v) = \exp\left(\frac{-v^2}{2\sigma^2}\right)$$

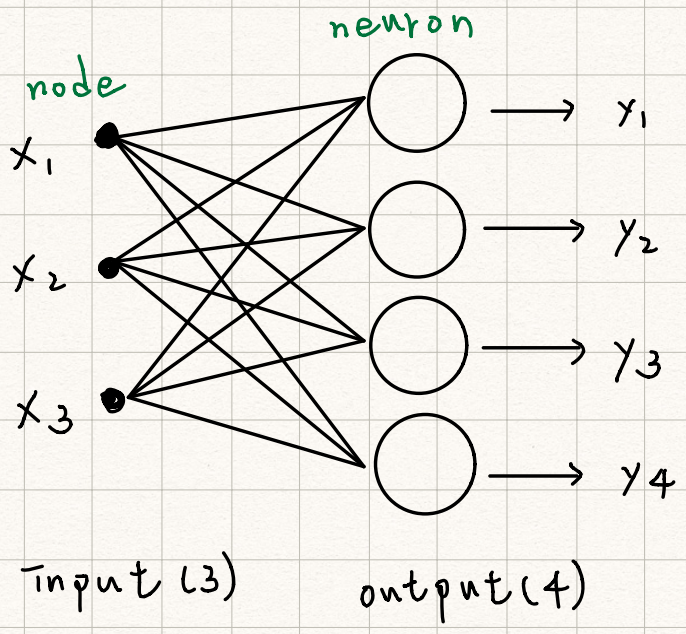
σ is a constant

(表寬度)通常令 $\sigma = 1$

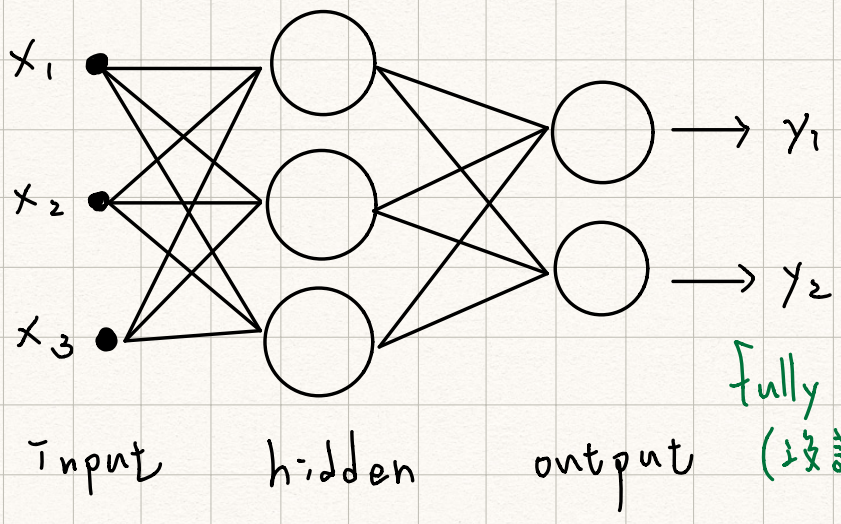
神經網路 → 分類以辨識 (用基底函數組成)

△. Network structures

1. single-layer feedforward networks

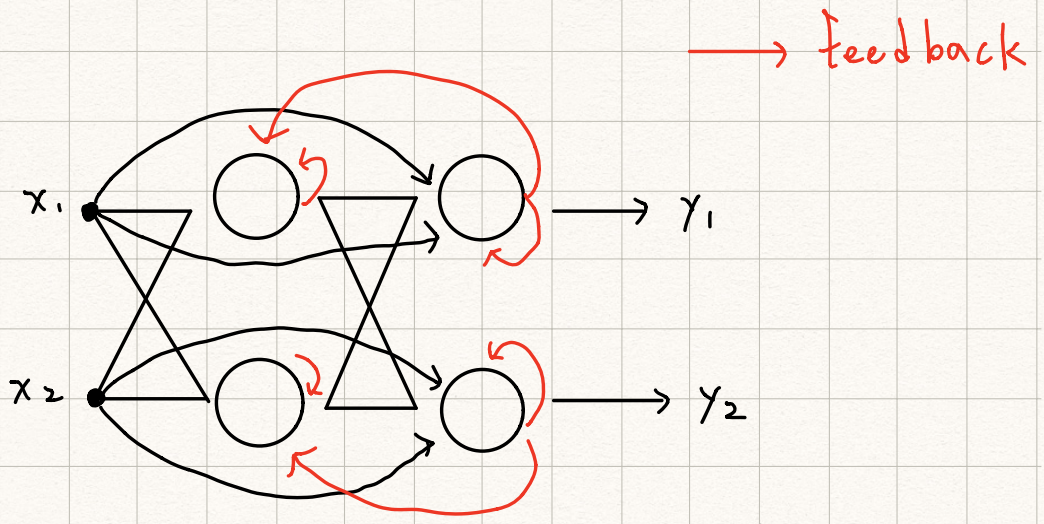


2. Multilayer feedforward networks
(由左往右)



fully connected
(攻讀者都是此種)

3. Recurrent networks



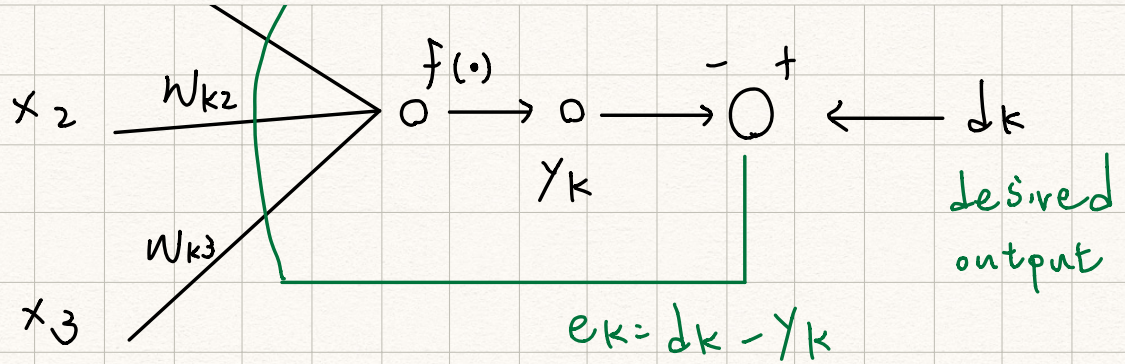
③ 適合用來處理有時序的系統

$$F(\underline{x}) = ? = w_1 f_1 + w_2 f_2 + \cdots\cdots$$

Δ. Learning Rules

The ways to chang parameters (weights) of a nenral networks

1. Error-correction learning 誤差修正法 (使誤差縮小)



try to make $e \to 0 \Rightarrow y_k = d_k$?

$w(n+1) = w(n) + \Delta w(n)$

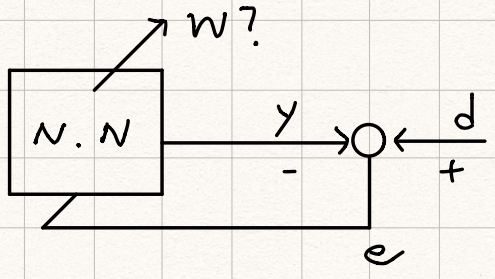
$w(new) = w(old) + \Delta w$ 更新公式

To minimize a cost function (performance index)

$E(n) = \frac{1}{2} e_k^2(n)$, e_k越小越好

期望值 - 實際輸出值 = 誤差, $d_k(n) - y_k(n) = e_k(n)$

以誤差值檢視
神經網路的品質 (性能指標) → $e = +1, -1$ → $|e|$ → e^2



gradiant descent (梯度修正法)

評估指標 Energy function

$$E = \frac{1}{2}\left[d_k(n) - y_k(n)\right]^2 = \frac{1}{2}\left(d_k(n) - \underline{w}^T(n)\,\underline{x}(n)\right)^2$$

for $\varphi(\cdot)$ is linear

$$\Delta \underline{w}(n) = -\eta \frac{\partial E}{\partial w(n)} \qquad \eta : \text{learning rate}$$

$$= \eta\left(d_k(n) - \underline{w}^T(n)\,\underline{x}(n)\right) x(n)$$

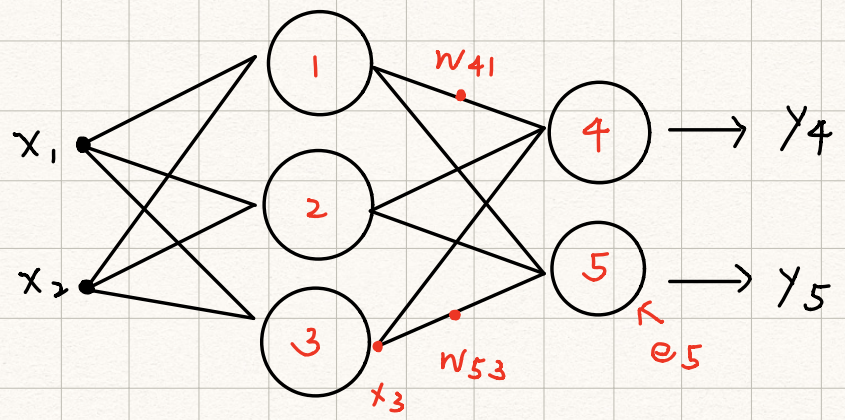
$$= \eta\left(d_k(n) - y_k(n)\right)\underline{x}(n)$$

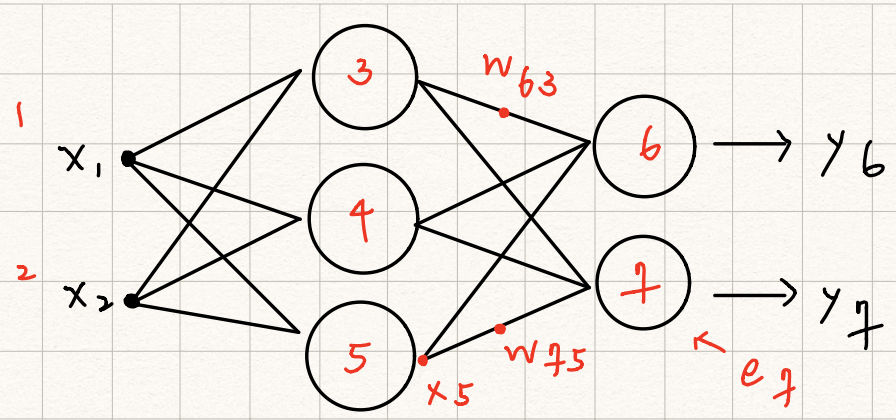
$$= \eta\, e_k(n)\, x(n)$$

$$\Rightarrow \quad \Delta w_{kj}(n) = \eta\, e_k(n)\, x_j(n) \quad ※$$

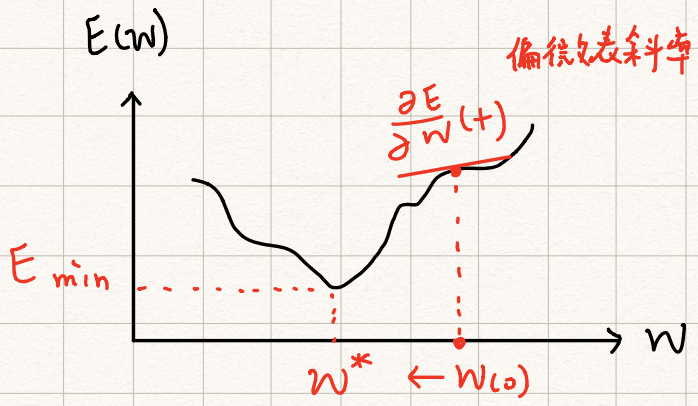
利用此公式，可使誤差越變越小

$$\Rightarrow \quad w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$$





Why $\Delta w(n) = -\eta \frac{\partial E}{\partial w(n)}$ let $e_k \downarrow$?



set $w(0) \in [-0.5, 0.5]$
(when programing)

η 表更新值的大小. w 的移動距離 ($\eta = 0.01$)

$w(0)$ 太大, $-\eta \frac{\partial E}{\partial w(n)}$ → 為負值

$- \quad + \quad +$

$E(n+1) \leq E(n) \leq \cdots \leq E(1) \leq E(0)$

要做幾次?, 才能使 $e_k = 0$ or 0.01 or .....

看系統規格誤差百分比

E(w). w 可能落入 local minimum

2. Memory - based leaning

3. Habbian leaning

4. Boltzymann leaning

5. Competitive learning

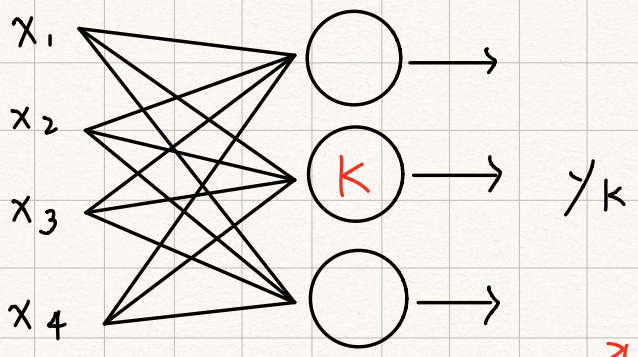
Only a single output neuron is active at any one time. Three basic elements to a competive learning rule.

① A set of neurons that are all the same except for some randomly distributed weights, and which therefore respond differently to a given set of input patterns.

② A limit imposed on the "strength" of each neuron.

③ A mechanism that permits the neurons to compete for the right to respond

to a given subset of inputs, such that only one output neuron or only one neuron per group, is active (on) at a time. The neuron that wins the competition is called a "winners-takes-all" neuron.



看誰的總輸入最大誰就 win

for neuron k is the winner, then

$$y_k = \begin{cases} 1, & \text{if } v_k > v_j, \text{ for all } j, j \neq k \\ 0, & \text{otherwise} \end{cases}$$

where v_k represents the combined action of all the forward and feedback inputs to neuron k.

A neuron learns by shifting from

its inactive to active inputs nodes.

$$\Delta w_{kj} = \begin{cases} \eta(x_j - w_{kj}) & \text{if neuro } k \text{ wins the competition} \\ 0, & \text{..... loses} \end{cases}$$

$$w_{kj}(n+1) = w_{kj}(n) + \Delta_{kj}(n) = 0$$

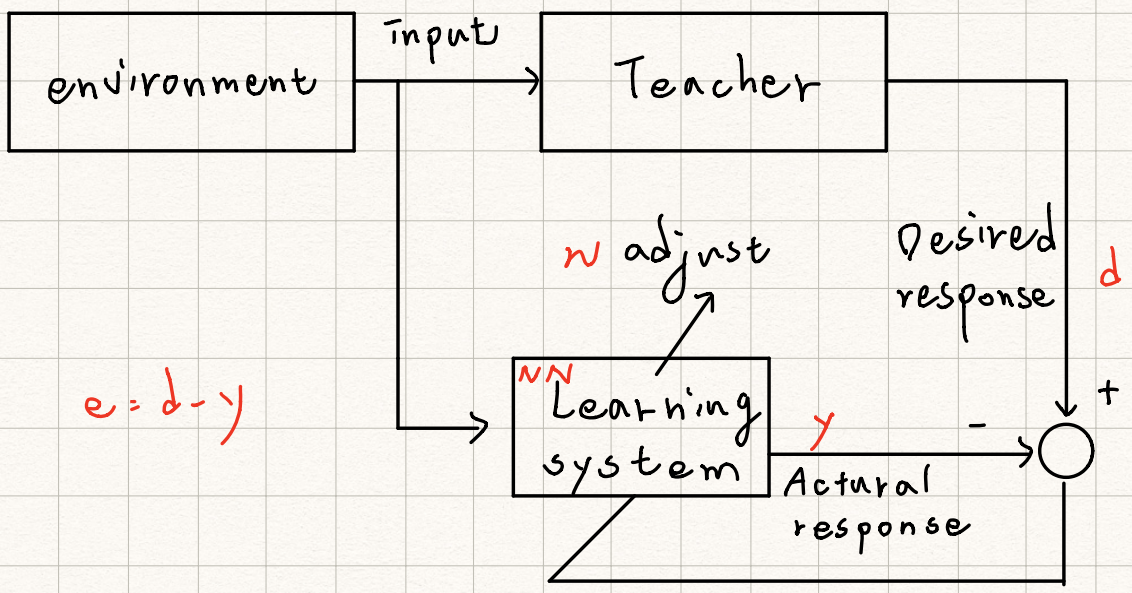
$$w(n+1) = w(n) = w(n-1) = \cdots\cdots$$

(停止條件) 時

5. Learning Paradigms

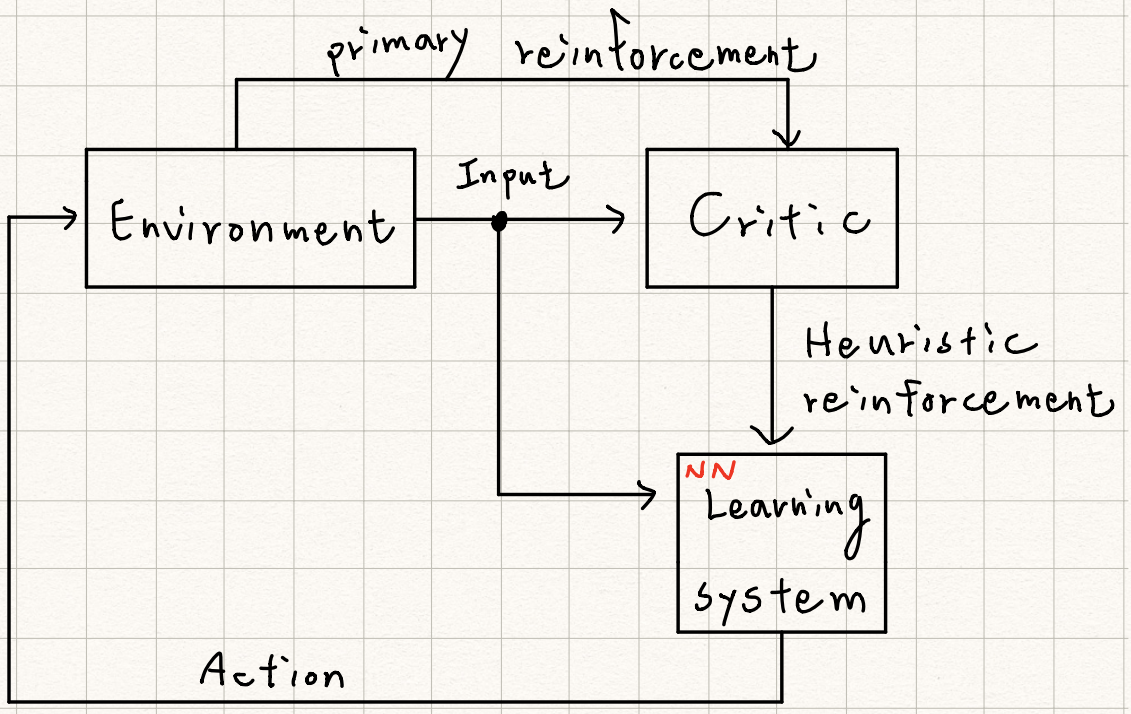
1. learning with a teacher (會給標準答案)

如誤差修正



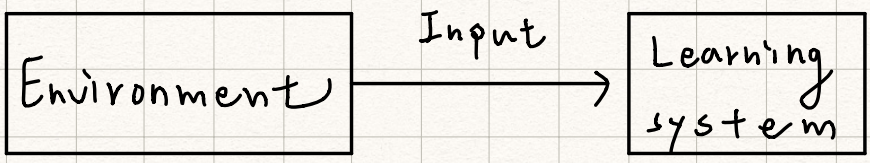
2. learning without teacher

① Reinforcement learning



② Unsupervise learning (不給標準答案) 如競爭學習

self-organized learning



4. Leaning Tasks

The choice of a particular learning algorithm is influenced by the learning task.

1. Pattern Association

store

recall

① autoassociation 

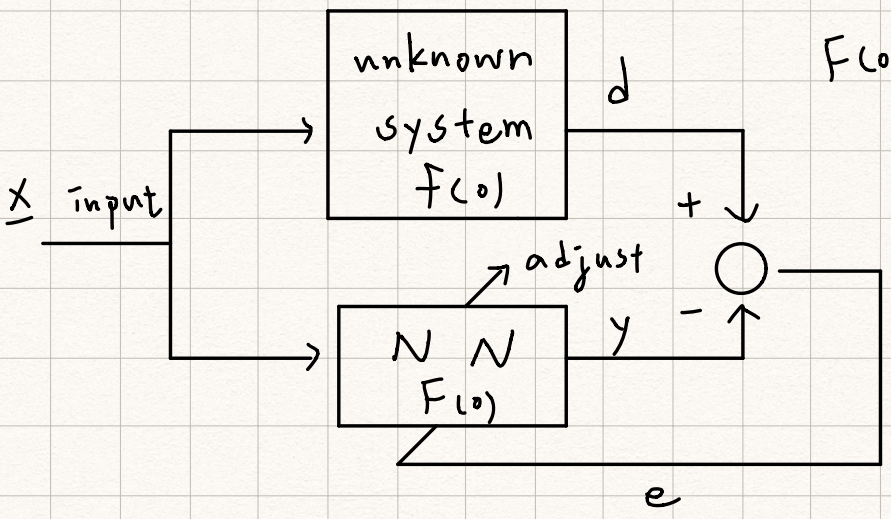
② heteroassociation 

2. Function Approximation (找系統函數)

① system identification

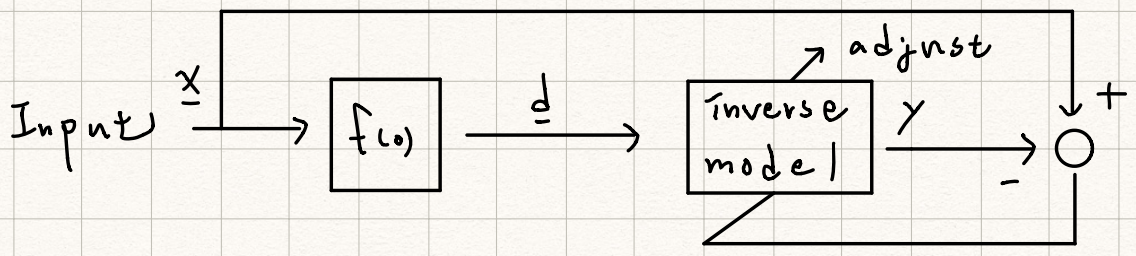
when $e = 0$

$F(0) = f(0)$

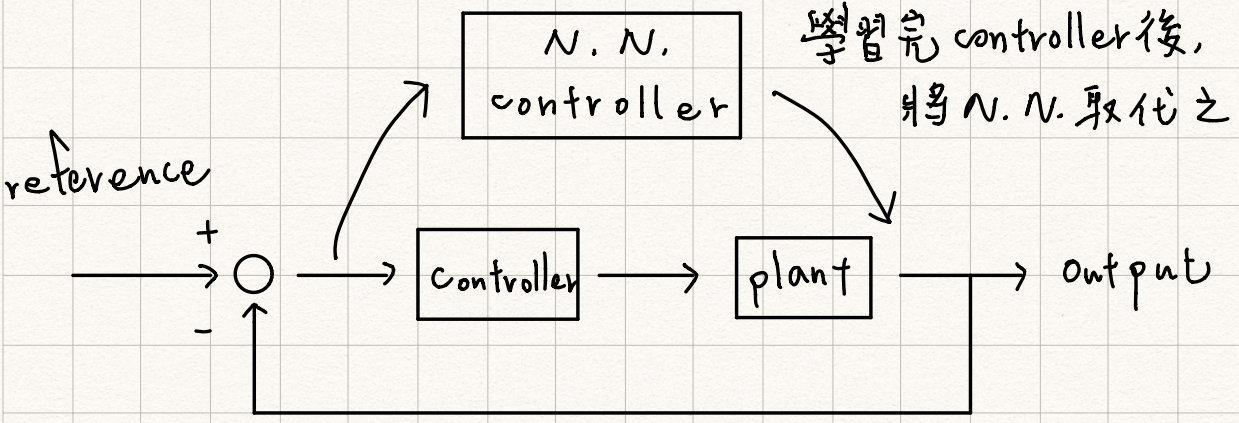


② inverse system : $\underline{x} = f^{-1}(\underline{d})$

↑ ?　　↑ 1000 rpm



3. control system



4. Filtering

① filtring ② smoothing 內差 ③ prediction 外差

↑

△. Iterative Descent

$$E(\underline{w}(n+1)) \leq E(\underline{w}(n))$$

method of steepest Descend

$$\underline{w}(n+1) = \underline{w}(n) - \eta \nabla E(\underline{w}(n))$$

$$\nabla E(\underline{w}(n)) = \left[\frac{\partial E}{\partial w_1} \quad \frac{\partial E}{\partial w_2} \quad \cdots \quad \frac{\partial E}{\partial w_m} \right]^T$$

w很多，要用偏微分

η is a positive constant. called "learning rate"

$$\boxed{E = \frac{1}{2} \sum e_i^2} \qquad e_i = d_i - y_i$$

proof: let $g \triangleq \nabla E(\underline{w})$

$$\Rightarrow \underline{w}(n+1) - \underline{w}(n) = \Delta \underline{w}(n) = -\eta g(n)$$

use a first-order Talor series expansion around w(n) to approximate

$E(w(n+1))$

$\Rightarrow E(\underline{w}(n+1)) \simeq E(\underline{w}(n)) + \left[\frac{\partial E}{\partial \underline{w}}\right]^T \Delta \underline{w}(n)$

$\Rightarrow E(\underline{w}(n+1)) \simeq E(\underline{w}(n)) + g^T(-\eta g)$

$= E(\underline{w}(n)) - \eta g^T g = E(\underline{w}(n)) - \eta \|g\|^2$

$\Rightarrow E(\underline{w}(n+1)) < E(\underline{w}(n))$

n : timing length

修正 η 過大而造成求解震蕩的情況

Δ. Learning-rate annealing schedules

1. $\eta(n) = \eta_0$, for all n

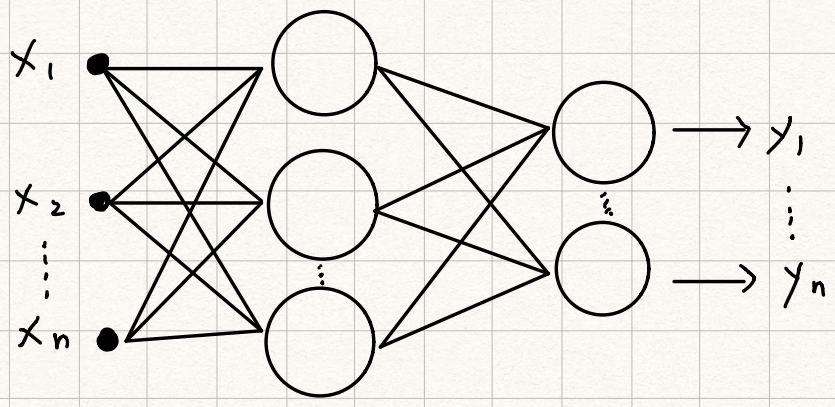
2. $\eta(n) = c/n$, c is a constant

3. $\eta(n) = \eta_0 / (1 + \frac{n}{\tau})$, η_0, τ are constants

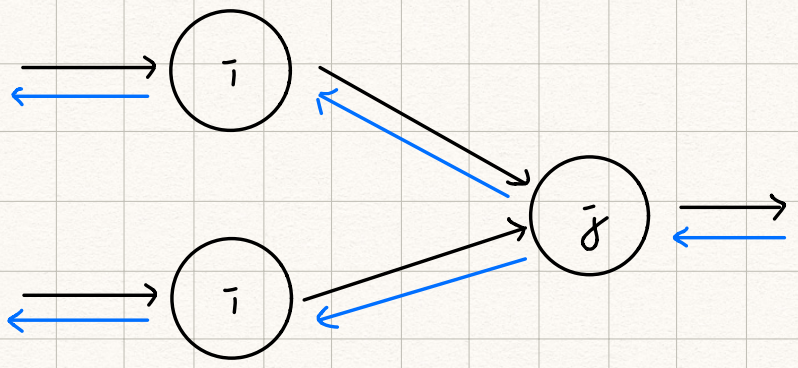
① for small $n \Rightarrow \frac{n}{\tau} < 1 \Rightarrow \eta(n) \simeq \eta_0$

② for large $\Rightarrow \frac{n}{\tau} > 1 \Rightarrow \eta(n) \simeq \frac{\eta_0 \tau}{n} = \frac{c}{n}$

Δ. Multilayer N.N.



Input layer　　hidden layer　　output layer



→ function signals

← error signals

Back-propagation algorithm :

$e_j(n) = d_j(n) - y_j(n)$, 誤差 e

$$E(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n) = \frac{1}{2}(e_1^2 + \cdots e_j^2 + \cdots e_n^2)$$ 評估指標

"C includes all the output neurons"

到時模擬作業要畫出此圖

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n)\, y_i(n)$$ 輸出函數自變數

$$y_j(n) = \varphi_j(v_j(n))$$ 輸出 y_j，輸出函數 φ_j

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n)$$ 權重 w 更新公式

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)}$$ ，Δw 公式

w 該如何修正？

for output unit j:

???

利用 chain rule

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n), \quad \frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n))$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1, \quad \frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n)$$

由之: $\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j'(v_j(n))y_i(n)$

Δw 中的某项公式

$$\Rightarrow \Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} = \eta \delta_j(n) y_i(n)$$

where $\delta_j = e_j(n)\varphi'(v_j(n))$, Δw 具体公式之另一形式

local gradient

for hidden unit j: → 没有期望值输出

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial v_j(n)} y_i(n)$$

$$\delta_j(n) = -\frac{\partial E(n)}{\partial v_j(n)} = -\frac{\partial E(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial}$$

$$\frac{\partial y_j(n)}{\partial v_j(n)}$$

$$= -\frac{\partial E(n)}{\partial y_j(n)} \varphi_j'(v_j(n))$$

??? (the circled factor)

$\because E(n) = \frac{1}{2}\sum_{k\in C} e_k^2(n)$ neuron k is an output unit

下標不能用 j，此時 j 表 hidden unit j

$$\therefore \frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)}$$

$\because e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_k(v_k(n))$

$$\Rightarrow \frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n))$$

$\because v_k(n) = \sum_{j=0} w_{kj}(n) y_j(n) \Rightarrow \frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n)$

$$\Rightarrow \frac{\partial E(n)}{\partial y_j(n)} = -\sum_k e_k(n) w_{kj}(n) \varphi_k'(v_k(n))$$

$$= -\sum_k \delta_k(n) w_{kj}(n)$$

$$\Rightarrow \delta_j(n) = \left[\sum_k \delta_k(n) w_{kj}(n)\right] \varphi_j'(v_j(n))$$

相當於 hidden layer 的 e_k

整理：

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n)$$

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)$$

$$\delta_j(n) = -\frac{\partial E(n)}{\partial y_j(n)} \varphi_j'(v_j(n))$$

for j is

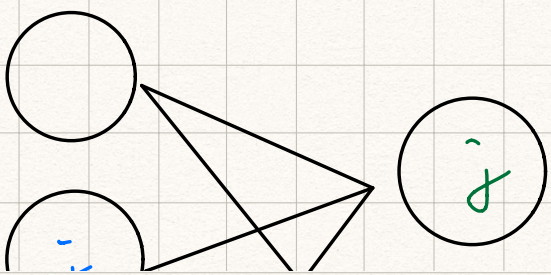
output unit: $-\frac{\partial E(n)}{\partial y_j(n)} = e_j(n)$

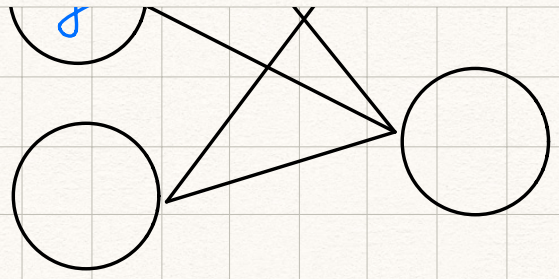
for j is

hidden unit: $-\frac{\partial E(n)}{\partial y_j(n)} = \sum_k \delta_k(n) w_{kj}(n)$

if j 表 hidden

k 表 output(j)





3. Learning mode:

1. sequential mode: weight updating is performed after the presentation of each training example (pattern)

for on-line real-time

2. Batch mode: weight updating is performed after the presentation of all the training patterns that constitute an epoch,

$\Delta w_{ji} = \frac{1}{N} \sum_{n=1}^{N} \Delta w_{ji}(n)$, N is the total number of the training example

for off-line

4. stopping Criteria

1. The BP is considered to have converged

when the Euclidean norm of the gradient vector reaches a sufficiently small threshold. (夠小的意思)

2. The ------ absolute rate of change in the average squared error per epoch is sufficiently small. (足夠小的意思)

3. Set up a number of training cycles as the threshold. (自己設一個)

4. Normalized data

normalize $\{\min \quad \max\} \longrightarrow \{a \quad b\}$

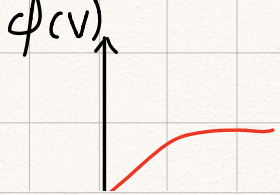
-10 +10

0.1 0.9

0.2 0.8 ✓

$$X_{new} = \frac{X - \min}{\max - \min}(b-a) + a$$

10



$v \to \infty \Rightarrow \varphi(v) = 1$

基底函數

若正規化設至1，表明不可能達到，故要改至0.2~0.8

4. Radial-Basis Function Networks (RBF)

1. Regularization RBF:

↓ ex: XOR的題目

A binary partition $\{C_1, C_2\}$ of C is said to be φ-separable if there exists an n_1-dimensional vector $\underline{w}$ such that

$$\underline{w}^T \underline{\Phi}(\underline{x}) > 0, \quad \underline{x} \in C_1 \text{ (class 1)}$$

$$\underline{w}^T \underline{\Phi}(\underline{x}) < 0, \quad \underline{x} \in C_2 \text{ (class 2)}$$

超平面

The hyperplane defined by $\underline{w}^T \underline{\Phi}(\underline{x}) = 0$ is the seperating surface in the φ-space. The inverse of this hyperplane is the seperating surface in the input space.

C is set of N patterns (vectors) $\underline{x}_1, \underline{x}_2, \cdots, \underline{x}_N$, and $\underline{x}_i$ belongs to either C_1 or C_2.

$\underline{\Phi}(\underline{x}_i) = [\varphi_1(\underline{x}_i), \varphi_2(\underline{x}_i), \cdots, \varphi_N(\underline{x}_i)]$ is the hidden function

Let $F(\underline{x}_i) = \sum_{j=1}^{N} w_j \varphi_j(\underline{x}_i)$

basis function

$\underline{d}_{N\times 1} = [d_1, d_2, d_3 \cdots, d_N]^T$

$\underline{w}_{N\times 1} = [w_1, w_2, \cdots w_N]^T$

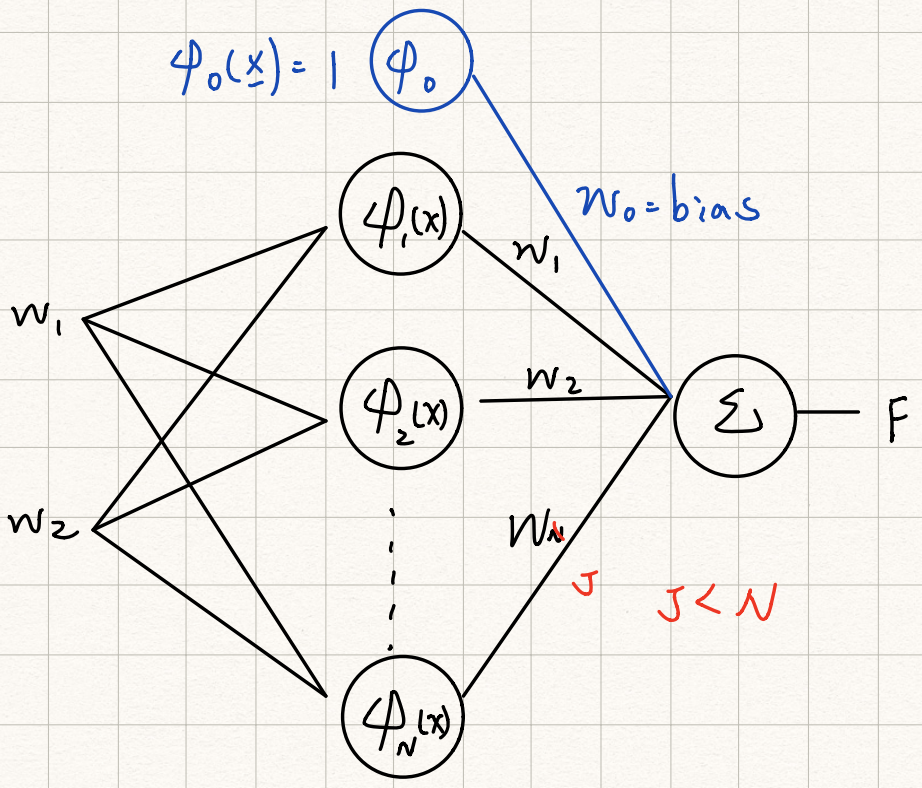
$\underline{\Phi}_{N\times N} = \{\varphi_{ji},\ j, i = 1, 2, \cdots, N\}$

then

解法 $\underline{\Phi}\,\underline{w} = \underline{d} \Rightarrow \underline{w} = \underline{\Phi}^{-1}\underline{d}$

↓ given ↓ measured 解出 w $F(\underline{x})$?

可將 Radial - Basis 轉成神經網路：



$$F(\underline{x}_i) = \sum_{j=1}^{J} w_j \varphi_j(\underline{x}_i) = w_1\varphi_1 + w_2\varphi_2 + \cdots + w_N\varphi_J$$

$$E(n) = \frac{1}{2} e^2(n) = \frac{1}{2}\left(d_n - F(\underline{x}_n)\right)^2$$

非 bias 的修正法：

$$w_j(n+1) = w_j(n) + \Delta w_j(n)$$

$$\Delta w_j(n) = -\eta \cdot \frac{\partial E(n)}{\partial w_j(n)} = -\eta \cdot \frac{\partial E(n)}{\partial F(\underline{x}_n)} \cdot \frac{\partial F(\underline{x}_n)}{\partial w_j(n)}$$

$$= \eta (d_n - F(\underline{x}_n)) \varphi_j(\underline{x}_n) \quad ※$$

$$E(n+1) < E(n) < \cdots < E(1) < E(0)$$

bias 的修正法：

$$b(n+1) = b(n) + \Delta b(n)$$

$$\Delta b(n) = -\eta \frac{\partial E(n)}{\partial b(n)} = \eta (d_n - F(\underline{x}_n)) \cdot 1 \quad ※$$

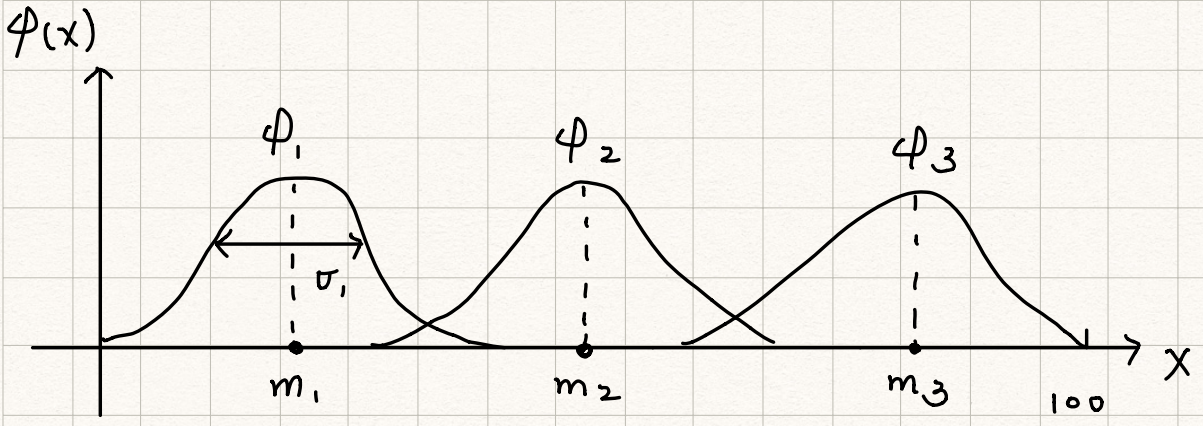
Three kinds of radial-basis function that are used in the RBF networks

1. $\varphi_j(\underline{x}) = \dfrac{1}{(\|\underline{x} - m_j\|^2 + c^2)^{\frac{1}{2}}}$

2. $\varphi_j(\underline{x}) = \exp\left[-\dfrac{\|\underline{x} - m_j\|^2}{2\sigma_j^2}\right]$

最常用（高斯函数）

3. $\varphi_j(\underline{x}) = \exp\left\{-\frac{1}{2}(\underline{x}-\underline{m}_j)^T \Sigma_j^{-1}(\underline{x}-\underline{m}_j)\right\}$



重疊的地方不能太少

高斯函數的 m 參數修正：

$$\underline{m}_j(n+1) = \underline{m}_j(n) + \Delta \underline{m}_j(n)$$

$$\Delta \underline{m}_j(n) = -\eta \frac{\partial E(n)}{\partial \underline{m}_j(n)} = -\eta \frac{\partial E(n)}{\partial F(\underline{x}_n)} \cdot \frac{\partial F(\underline{x}_n)}{\partial \varphi_j(\underline{x}_n)} \cdot \frac{\partial \varphi_j(\underline{x}_n)}{\partial \underline{m}_j(n)}$$

$$= \underbrace{\eta(d_n - F(\underline{x}_n))}_{①} \underbrace{w_j(n)}_{②} \underbrace{\varphi_j(x_n) \frac{1}{\sigma_j^2}(\underline{x}_n - \underline{m}_j(n))}_{③}$$

$$\sigma_j(n+1) = \sigma_j(n) + \Delta\sigma_j(n)$$

$$\Delta\sigma_j(n) = -\eta \frac{\partial E(n)}{\partial \sigma_j(n)}$$

$$= -\eta \frac{\partial E(n)}{\partial F(\underline{x}_n)} \cdot \frac{\partial F(\underline{x}_n)}{\partial \varphi_j(x_n)} \cdot \frac{\partial \varphi_j(\underline{x}_n)}{\partial \underline{m}_j(n)}$$

$$= \eta (d_n - F_x(x_n)) w_j(n) \varphi_j(\underline{x}_n) \frac{1}{\sigma_j^2} (\underline{x}_n - \underline{m}_j(n))$$

只有winner才能更新w，ART適合處理圖像分類

Δ. Adaptive Resonance Theory (ART)

competitive learning — winner-takes-all

1. Initialize all weights to "1" and enable all neurons

2. Input $\underline{x}$

3. Compare first similarity with all enabled neurons by S_i

$$S_1(\underline{w}_j, \underline{x}) = \frac{\underline{w}_j \underline{x}}{\beta + \sum_{i=1}^{n} w_{ij}} \qquad 0 < \beta < 1$$

恒分母是零

4. If the winner from step 3 is j^{th} neuron, then check the second similarity by S_2

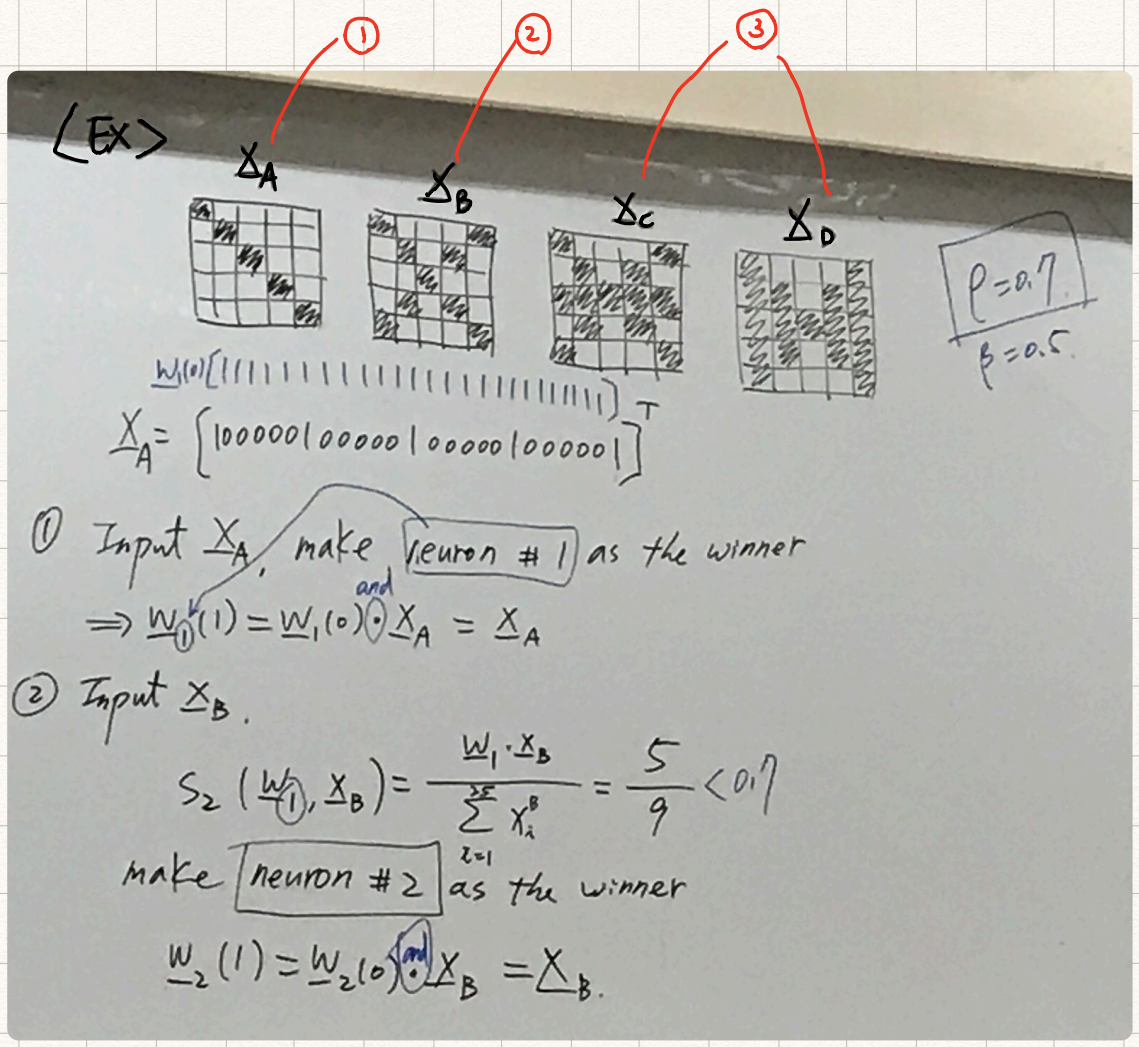
$$S_2(\underline{w}_j, \underline{x}) = \frac{\underline{w}_j \cdot \underline{x}}{\sum_{i=1}^{n} x_i}$$

if $S_2 \geq \rho$ go to step 5. $\rho > 0$, given
$S_2 < \rho$ disable j^{th} neuron and go to step 3

5. updata winner's weight by
$w_{ij}(t+1) = w_{ij}(t) \cdot x_i, \quad 1 \leq i \leq n$

Indicate this input $\underline{x}$ as cluster j

Go to step 2



③ X_C

$$S_1(W_1, X_C) = \frac{5}{0.5+5} = 0.91$$

$$S_1(W_2, X_C) = \frac{9}{0.5+9} = 0.95$$

$$S_2(W_2, X_C) = \frac{9}{13} < 0.7$$

$S_2(\underline{w}_1, \underline{x}_C) = \frac{5}{13} < 0.7$

make **neuron ※3** as the winner

沒神經細胞可用，故新增其他細胞使用

$\underline{w}_3(1) = \underline{w}_3(0) \cdot \underline{x}_C = \underline{x}_C$

④ $\underline{x}_D$

$S_1(\underline{w}_1, \underline{x}_D) = 0.91$, $S_1(\underline{w}_2, \underline{x}_D) = 0.95$

$S_1(\underline{w}_3, \underline{x}_D) = 0.96$. $S_2(\underline{w}_3, \underline{x}_D) = \frac{13}{17} > 0.7$

$\underline{w}_3(2) = \underline{w}_3(1) \cdot \underline{x}_D = \underline{x}_C = \underline{w}_3(1)$

最後，ART將 X_A-類，X_B-類，X_C X_D X-類