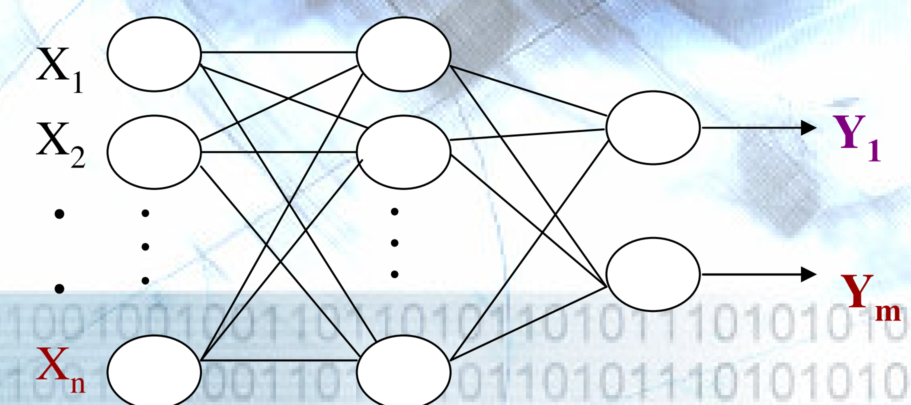
-Artificial Neural Network-
Counter Propagation Network

朝陽科技大學
資訊管理系
李麗華 教授

Introduction (1/4)

- **Counter Propagation Network(CPN)**
  - Defined by Robert Hecht-Nielsen in 1986, CPN is a network that learns a bidirectional mapping in hyper-dimensional space.
  - CPN learns both forward mapping （from n-space to m-space）and, if it exists, the inverse mapping （from m-space to n-space） for a set of pattern vectors.



Introduction (2/4)

— Counter Propagation Network (CPN) is a an unsupervised winner-take-all competitive learning network.

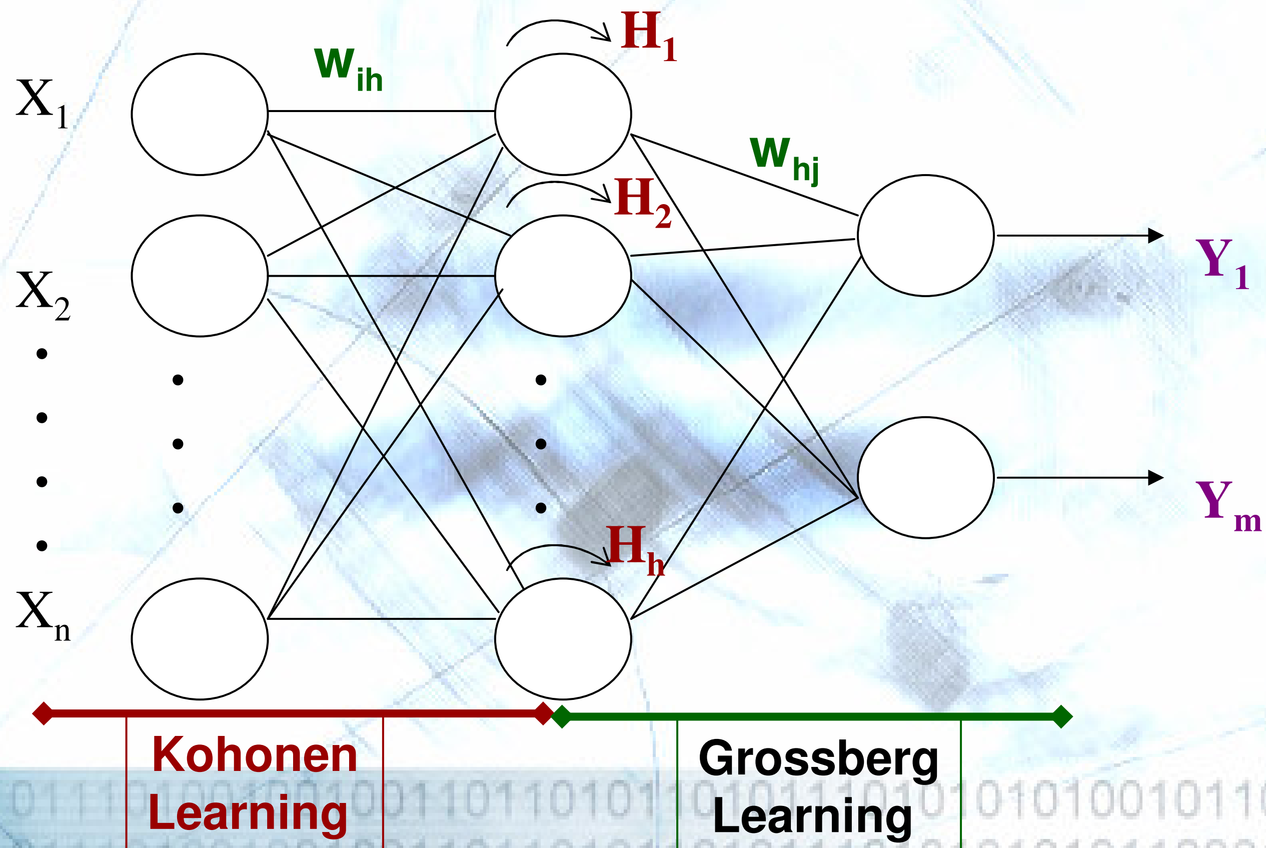
— The learning process consists 2 phases：

The kohonen learning (unsupervised) phase &

the Grossberg learning(supervised) phase.

Introduction (3/4)

– The Architecture：



Introduction (4/4)

Input layer : $X=[X_1, X_2, \ldots\ldots.X_n]$

Hidden layer: also called Cluster layer, $H=[H_1, H_2, \ldots\ldots.H_n]$

Output layer: $Y=[Y_1, Y_2, \ldots\ldots Y_m]$

Weights : From Input→Hidden: $\mathbf{W_{ih}}$,

From Hidden→Output : $\mathbf{W_{hj}}$

Transfer function: uses linear type

$$f(net_j) = net_j$$

The learning Process(1/2)

The learning Process：

Phase I: (Kohonen unsupervised learning)

(1)Computes the Euclidean distance between input vector & the weights of each hidden node.

(2)Find the winner node with the shortest distance.

(3)Adjust the weights that connected to the winner node in hidden layer with $\triangle W_{ih^*} = \eta_1 (X_i - W_{ih^*})$

Phase II: (Grossberg supervised learning）

- Some as (1)& (2)of phase I
- Let the link connected to the winner node to output node is set as 1 and the other are set to 0.
- Adjust the weights using $\triangle W_{ij} = \eta_2 \cdot \delta \cdot H_h$

The learning Process(2/2)

The recall process:

- Set up the network
- Read the trained weights.
- Input the test vector, X.
- Computes the Euclidean distance & finds the winner where the winner hidden node output 1 and the other output 0.
- Compute the weighted sum for output nodes to derive the prediction (mapping output).

The computation of CPN (1/4)

Phase I：（Kohonen unsupervised learning）

1. Set up the network.
2. Randomly assign weights, W_{ih}
3. Input training vector, X=[X_1, X_2, …….X_n]
4. Compute the Euclidean Distance to find the winner node, H^*

$$net_h = \sum_i (x_i - w_{ih})^2 \quad \text{or} \quad net_h = \sqrt{\sum_i (x_i - w_{ih})^2}$$

$$net_{h^*} = \min_h [net_h]$$

5. $H_h = \begin{cases} 1 & \text{if } h=h^* \\ 0 & \text{otherwise} \end{cases}$

The computation of CPN (2/4)

6. Update weights $\triangle W_{ih}^{*} = \eta_1 (X_1 - W_{ih}^{*})$

   $W_{ih}^{*} = W_{ih}^{*} + \triangle W_{ih}^{*}$.

7. Repeats 3 ~ 6 until the error value is small & stable or the number of training cycle is reached.

The computation of CPN (3/4)

- Phase II：（Grossberg supervised learning）
  1. Input training vector
  2. Computes 4 & 5 of Phase I

$$net_h = \sum_i (x_i - w_{ih})^2$$

$$net_{h^*} = \min_h [net_h]$$

$$net_j = \sum W_{hj} \cdot H_h$$

$$Y_j = net_j$$

The computation of CPN (3/4)

3. Computes error ： $\delta_j = (T_j - Y_j)$

4. Updates weights： $\triangle W^*_{hj} = \eta_2 \cdot \delta_j \cdot H_{h^*}$

   $W_{h^* j} = W_{h^* j} + \triangle W_{h^* j}$

5. Repeats 1 to 4 of Phase II until the error is very small & stable or the number of training cycle is reached.

The recall computation

1. Set up the network.
2. Read the trained weights, W_{ih}
3. Input testing vector (pattern), X=[X_1, X_2, …….X_n]
4. Compute the Euclidean Distance to find the winner node, H^*

$$net_h = \sum_i (x_i - w_{ih})^2 \rightarrow net_{h^*} = \min_h [net_h]$$

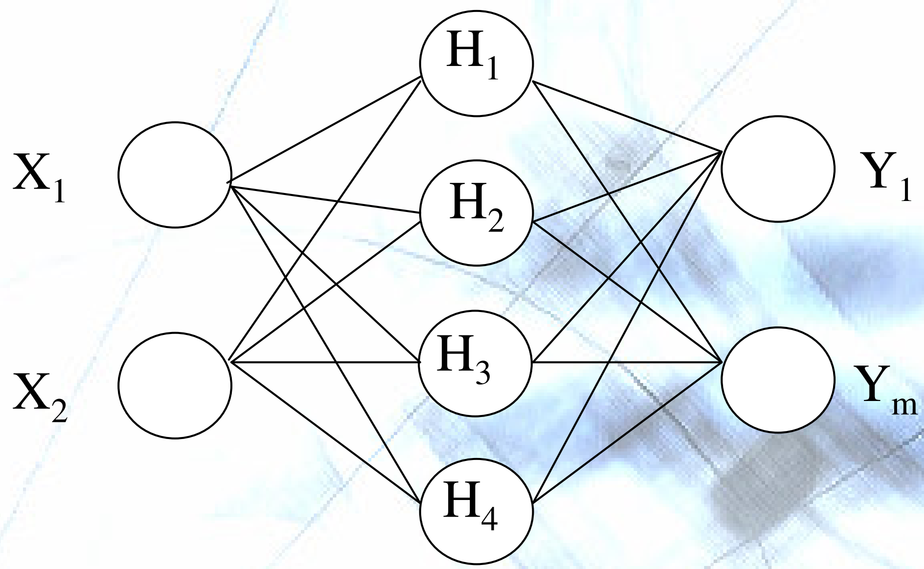
$$\rightarrow H_h = \begin{cases} 1 & \text{if } h = h^* \\ 0 & \text{otherwise} \end{cases}$$

$$\rightarrow net_j = \sum_j W_{hj} \cdot H_h \rightarrow Y_j = net$$

The example of CPN (1/2)

- Ex：Use CPN to solve XOR problem

X_1	X_2	T_1	T_2
-1	-1	0	1
-1	1	1	0
1	-1	1	0
1	1	0	1



Randomly set up the weights of W_{ih} & W_{hj}

The example of CPN (2/2)

Sol: 以下僅介紹如何計算Phase I (Phase II 計算上課說明)

(1) 代入X=[-1,-1] T=[0,1]

net_1 =[-1-(-0.5)]^2 + [-1-(-0.5)]^2 = (-0.5^2) + (-0.5^2) = 0.5

net_2 =[-1-(-0.5)]^2 + [1-(-0.5)]^2 = (-0.5^2) + (1.5^2) = 2.5

net_3 = 2.5

net_4 = 4.5

∴ net_1 has minimum distance and the winner is h* = 1

(2) Update weights of W_{ih^*}

$\triangle W_{11} = (0.5)\,[-1-(-0.5)] = -0.25$

$\triangle W_{21} = (0.5)\,[-1-(-0.5)] = -0.25$

∴ $W_{11} = \triangle W_{11} + W_{11} = -0.75$,

$W_{21} = \triangle W_{21} + W_{21} = -0.75$