-Artificial Neural Network-
Chapter 5 Back Propagation Network

朝陽科技大學
資訊管理系
李麗華 教授

Introduction (1)

- BPN = Back Propagation Network
- BPN is a layered feedforward supervised network.
- BPN provides an effective means of allowing a computer to examine data patterns that may be incomplete or noisy.
- BPN can take various type of input, i.e., binary data or real data.
- The output of BPN is depending on the transfer function used.

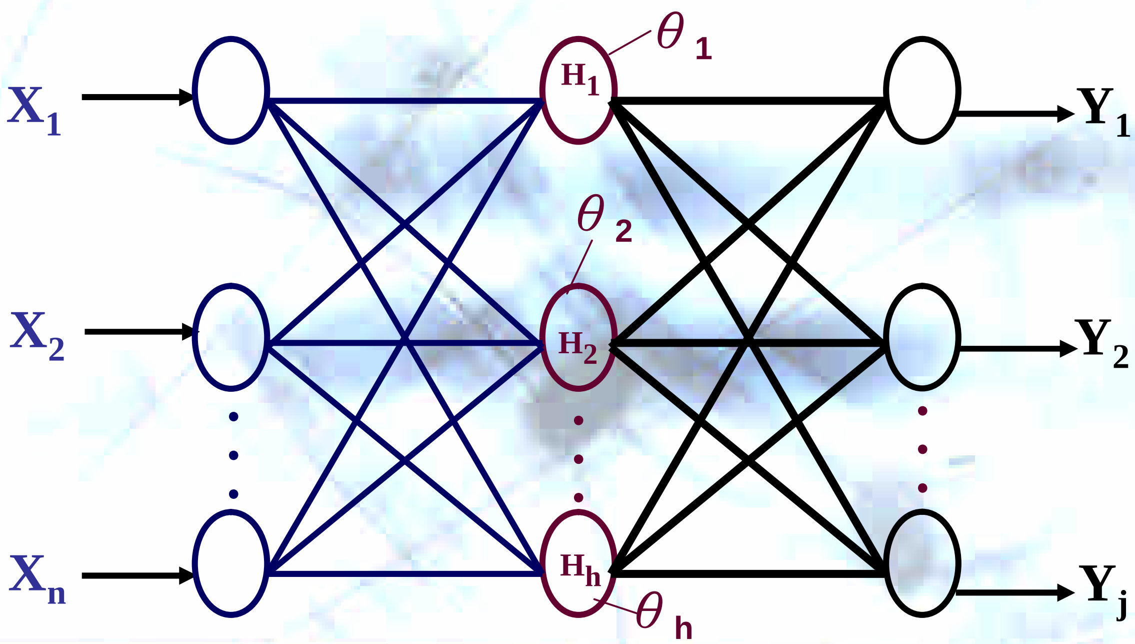
(1) If the sigmoid function is used, then the output $0 \leq y \leq 1$

(2) If the hyperbolic Tangent function is used,

then the output : $-1 \leq y \leq 1$

Introduction (2)

Architecture:



Introduction (3)

- Input layer: $[X_1, X_2, \ldots. X_n]$.
- Hidden layer: can have more than one layer.
- derive: net_1, net_2, …net_h; transfer output H_1, H_2,…,H_h,
  H_h will be used as the input to derive the result for output layer
- Output layer: $[Y_1, \ldots Y_j]$.
- Weights: W_{ij}.
- Transfer function: Nonlinear ➔ Sigmoid function

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

(*) The nodes in the hidden layers organize themselves in a way that different nodes learn to recognize different features of the total input space.

Introduction (4)

- Application of BPN is quite broad.
  - Pattern Recognition (樣本識別; 字母識別)
  - Prediction (股市預測)
  - Classification (客群分類**)**
  - Learning (資料學習)
  - Control (回饋與控制)
  - CRM (客服分群服務)

Processing Steps (1)

The processing steps can be briefly described as follows.

1. Based on the problem domain, set up the network.
2. Randomly generate weights W_{ij}.
3. Feed a training set, $[X_1, X_2, \ldots. X_n]$, into BPN.
4. Compute the weighted sum and apply the transfer function on each node in each layer. Feeding the transferred data to the next layer until the output layer is reached.
5. The output pattern is compared to the desired output and an error is computed for each unit.

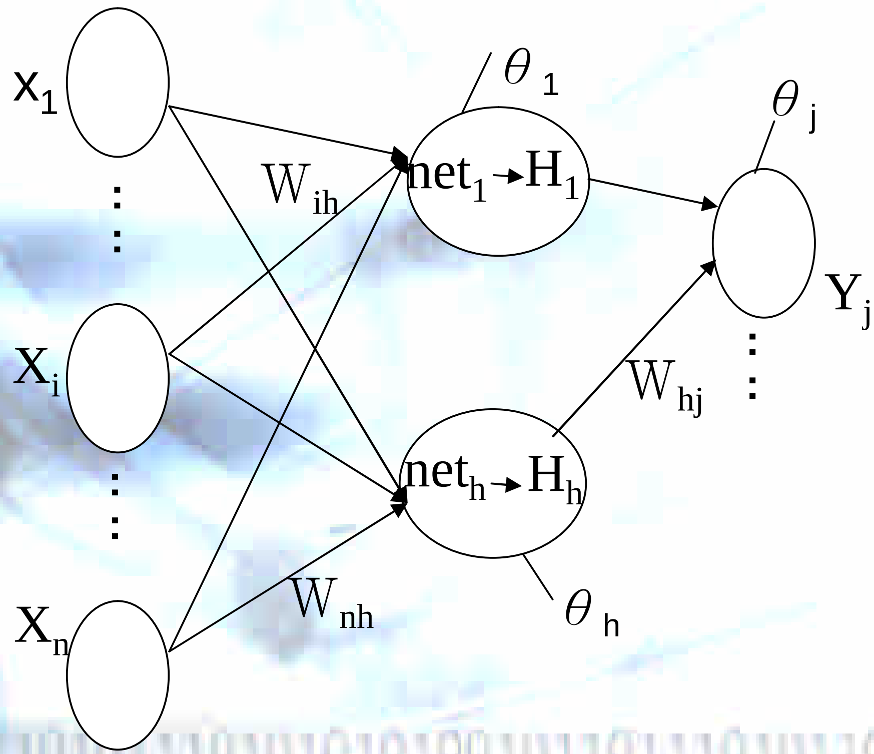
Processing Steps (2)

6. Feedback the error back to each node in the hidden layer.
7. Each unit in hidden layer receives only a portion of total errors and these errors then feedback to the input layer.
8. Go to step 4 until the error is very small.
9. Repeat from step 3 again for another training set.

Computation Processes(1/10)

- The detailed computation processes of BPN.

1. Set up the network according to the input nodes and the output nodes required. Also, properly choosing the hidden layers and nodes.
2. Randomly assigned the weights.
3. Feed the training pattern (set) into the network and do the following computation.



Computation Processes(2/10)

4. Compute from the Input layer to hidden layer for each node.

$$net_h = \sum_{\mathbf{i}} \mathbf{W_{ih}} \cdot \mathbf{X_i} - \theta_{\mathbf{h}}$$

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

5. Compute from the hidden layer to output layer for each node.

$$net_j = \sum_{\mathrm{i}} \mathrm{W_{hj}} \cdot \mathrm{H_h} - \theta_{\mathrm{j}}$$

$$Y_j = f(net_h) = \frac{1}{1+e^{-net_j}}$$

Computation Processes(3/10)

6. Calculate the total error & find the difference for correction

$$\delta_j = Y_j(1-Y_j)(T_j - Y_j)$$

$$\delta_h = H_h(1-H_h)\sum_j W_{hj}\,\delta_j$$

7. $\Delta W_{hj} = \eta\,\delta_j H_h \quad \Delta\Theta_j = -\eta\,\delta_j$

   $\Delta W_{ih} = \eta\,\delta_h X_i \quad \Delta\Theta_h = -\eta\,\delta_h$

8. update weights

   $W_{hj} = W_{hj} + \Delta W_{hj}$，$W_{ih} = W_{ih} + \Delta W_{ih}$，

   $\Theta_j = \Theta_j + \Delta\Theta_j$，$\Theta_h = \Theta_h + \Delta\Theta_h$

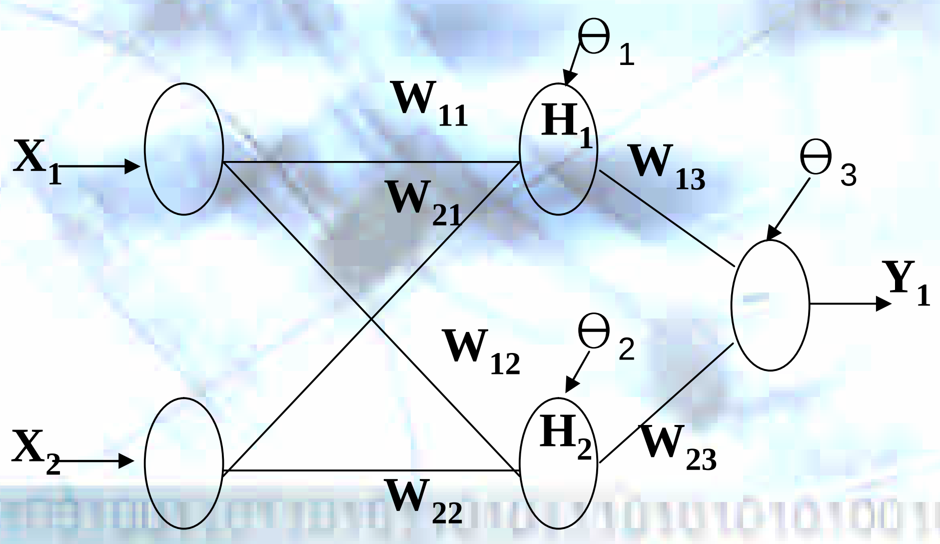
9. Repeat steps 4~8, until the error is very small.

10. Repeat steps 3~9, until all the training patterns are learned.

EX: Use BPN to solve XOR (1)

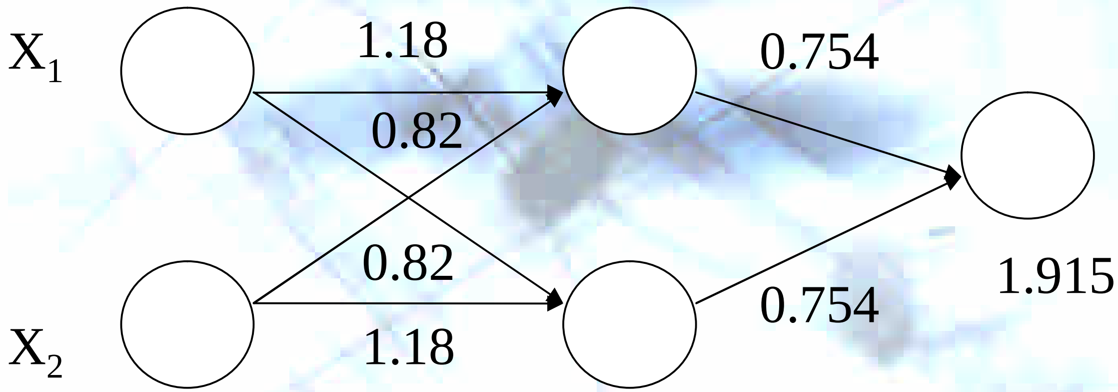
X_1	X_2	T
-1	-1	0
-1	1	1
1	-1	1
1	1	0

- Use BPN to solve the XOR problem
- Let $W_{11}=1$, $W_{21}= -1$, $W_{12}= -1$, $W_{22}=1$, $W_{13}=1$, $W_{23}=1$, $\Theta_1=1$, $\Theta_2=1$, $\Theta_3=1$, $\eta=10$



EX: BPN Solve XOR (2)

- $\Delta W_{12} = \eta\ \delta_1\ X_1$ =(10)(-0.018)(-1)=0.18
- $\Delta W_{21} = \eta\ \delta_1\ X_2$ =(10)(-0.018)(-1)=0.18
- $\Delta\Theta_1 = -\eta\ \delta_1$ = -(10)(-0.018)=0.18
- 以下爲第一次修正後的權重值.



BPN Discussion

1. Number of hidden nodes increase, the convergence will get slower. But the error can be minimized.
2. The general concept of designing the number of hidden node uses:

   # of hidden nodes=(Input nodes + Output nodes)/2, or

   # of hidden nodes=(Input nodes * Output nodes)$^{1/2}$
3. Usually, 1~2 hidden layer is enough for learning a complex problem. Too many layers will cause the learning very slow. When the problem is hyper-dimension and very complex, then an extra layer could be used
4. Learning rate, η, usually set between [0.1, 1.0], but it depends on how fast and how detail the network shall learn.

The Gradient Steepest Descent Method(SDM) (1)

- The gradient steepest descent method
- Recall:

$$net_j^n = \sum_j W_{ij} A_i^{n-1} - \theta_j$$

- We want the difference of computed output and expected output getting close to 0.

$$E = (1/2)\sum_j (T_j - A_j)^2 \Rightarrow \Delta W_{ij} = -\eta \frac{\partial E}{\partial W_{ij}}$$

- Therefore, we want to obtain $\frac{\partial E}{\partial W_{ij}}$ so that we can update weights to improve the network results.

The Gradient Steepest Descent Method(SDM) (2)

$$\frac{\partial E}{\partial W_{ij}} = \left(\frac{\partial E}{\partial net_j^n}\right)\left(\frac{\partial net_j^n}{\partial W_{ij}}\right) = \underset{(3)}{\left(\frac{\partial E}{\partial A_j^n}\right)}\underset{(2)}{\left(\frac{\partial A_j^n}{\partial net_j^n}\right)}\underset{(1)}{\left(\frac{\partial net_j^n}{\partial W_{ij}}\right)}$$

For (1)

$$\frac{\partial net_j^n}{\partial W_{ij}} = \frac{\partial\left(\sum_k W_{kj} A_k^{n-1} - \theta_j\right)}{\partial W_{ij}}$$

For (2)

$$\frac{\partial A_j^n}{\partial net_j^n} = \frac{\partial f(net_j^n)}{\partial net_j^n} = f'(net_j^n)$$

For (3-1): when n is the output layer

$$\frac{\partial E}{\partial A_j^n} = \frac{\partial[1/2\sum_k (T_k - A_k^n)^2]}{\partial A_j^n} = -(Tj - A_j^n)$$

For (3-2) when n is the hidden layer

$$\frac{\partial E}{\partial A_j^n} = \sum_k \left(\frac{\partial E}{\partial net_k^{n+1}}\right)\left(\frac{\partial net_k^{n+1}}{\partial A_j^n}\right) = -\sum_k \delta_k^{n+1} W_{jk}$$

The Gradient Steepest Descent Method(SDM) (3)

From (1)(2)(3) we have two types of values:

∗When n is output layer

$$\frac{\partial E}{\partial W_{ij}} = -(T_j - A_j^n)\, f^{\,t}(A_j^n) A_i^{n-1} \qquad (代入(B))$$

or $\quad = -\delta_j^n A_i^{n-1} \qquad$ (代入(A))

we get $\delta_{\mathrm{j}}^{\mathrm{n}} = (T_j - A_i^{n-1})\, f^{\,t}(\, net_j^n)$

The Gradient Steepest Descent Method(SDM)　(4)

* When n is hidden layer

$$\frac{\partial E}{\partial W_{ij}} = -[\sum_k \delta_k^{n+1} W_{jk}]f^{\,t}(net_j^n)A_i^{n-1} \quad (\text{代入(B)})$$

$$\text{or} \quad = -\delta_j^n A_i^{n-1} \quad (\text{代入(A)})$$

$$\text{we get} \quad \delta_j^n = [\sum_k \delta_k^{n+1} W_{jk}]f^{\,t}(net_j^n)$$

$$\left.\begin{array}{l} \Rightarrow \dfrac{\partial E}{\partial W_{ij}} = -\delta_j^n A_i^{n-1} \\ \Rightarrow \Delta W_{ij} = +\eta \delta_j^n A_i^{n-1} \\ \Rightarrow \theta = -\eta - \delta_j^n \end{array}\right\} \begin{array}{l} W_{ij} = W_{ij} + \varDelta W_{ij} \\ \theta_j = \theta_j + \varDelta\theta_j \end{array}$$

The Gradient Steepest Descent Method(SDM) (5)

$$f(net_j^n) = \frac{1}{1+e^{-net_j}} = (1+\mathrm{e}^{\text{-netj}})^{-1}$$

$$f^{t}(net_j^n) = [(1+\mathrm{e}^{\text{-net}_j})^{-1}]^{-2}][-(\mathrm{e}^{\text{-net}_j})]$$

$$= \frac{e^{-netj}}{(1+e^{-net_j})^2} = \frac{e^{-netj}}{(1+e^{-net_j})} \cdot \frac{1}{1+e^{-net_j}}$$

$$= f(net_j)(1- f(net_j))$$

$$\delta_j^n = \begin{cases} (\mathrm{T_j - Y_j)Y_j(1 - Y_j)} & \text{if n is output layer} \\ [\sum_k \delta_j^{n+1} W_{ik}] \bullet H_j(1-H_j) & \text{if n is hidden layer} \end{cases}$$

The Gradient Steepest Descent Method(SDM) (6)

- Learning computation

1. $net_j = \sum_i W_{ih} \bullet X_i - \theta_h$ Compute value of the hidden layer

$$\mathrm{H_h} = f(net_h) = \frac{1}{1+e^{-net_h}}$$

2. $net_j = \sum_i W_{hj} \bullet H_h - \theta_j$ Compute value of the output layer

$$\mathrm{Y_j} = f(net_j) = \frac{1}{1+e^{-net_j}}$$

3. $\delta_j = \mathrm{Y_j(1- Y_j)(T_j- Y_j)}$ Compute the value difference for correction

$$\delta_h = H_h(1\text{-}H_h)\sum_j W_{hj}\delta_j$$

The Gradient Steepest Descent Method(SDM) (7)

4. $\Delta W_{hj} = \Delta\eta\ \delta_j \mathrm{H}_\mathrm{h}$ $\quad \Delta\theta = \eta\ \delta_j$ $\quad$ Compute the value to be updated

$\Delta W_{ih} = \eta\ \delta_h \mathrm{H}_\mathrm{i}$

5. $W_{hj} = W_{hj} + \Delta W_{hj}$ $\quad \theta_j = \theta_j + \Delta\theta_j$

$W_{ih} = W_{ih} + \Delta W_{ih}$ $\quad \theta_h = \theta_h + \Delta\theta_h$