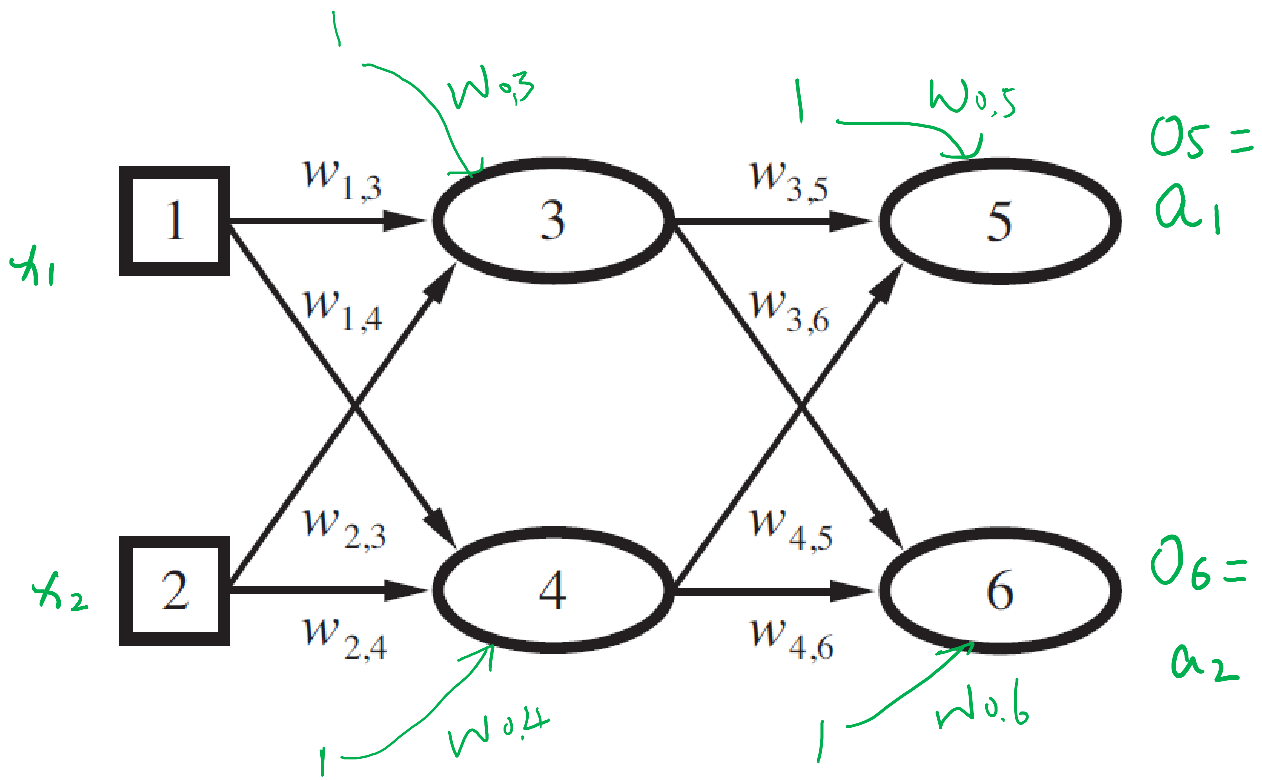
Multi-layer Neural Network
BP Algo. – example & implementation

CS385 Machine Learning – A.N.N.

B.P. Example – the Network



$$in_3 = x_1 \cdot w_{1,3} + x_2 \cdot w_{2,3} + w_{0,3}$$

$$o_3 = sigmoid(in_3)$$

B.P. Example – Feed Forward

$$o_3 = \mathrm{sigmoid}(x_1 \cdot w_{1,3} + x_2 \cdot w_{2,3} + w_{0,3})$$

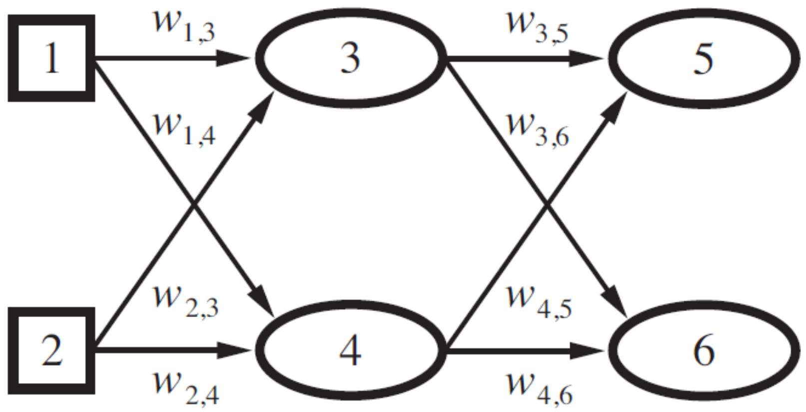
$$o_4 = \mathrm{sigmoid}(x_1 \cdot w_{1,4} + x_2 \cdot w_{2,4} + w_{0,4})$$

$$o_5 = \mathrm{sigmoid}(o_3 \cdot w_{3,5} + o_4 \cdot w_{4,5} + w_{0,5})$$

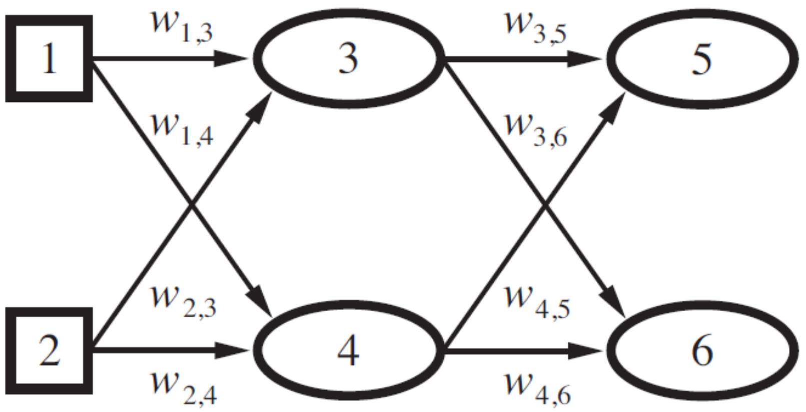
$$o_6 = \mathrm{sigmoid}(o_3 \cdot w_{3,6} + o_4 \cdot w_{4,6} + w_{0,6})$$

$$\delta_5 = (y_1 - o_5) \cdot o_5 \cdot (1 - o_5)$$

$$\delta_6 = (y_2 - o_6) \cdot o_6 \cdot (1 - o_6)$$



B.P. Example – Learning (stochastic)



$$w_{3,5} = w_{3,5} + \alpha \cdot \delta_5 \cdot o_3$$

$$w_{4,5} = w_{4,5} + \alpha \cdot \delta_5 \cdot o_4$$

$$w_{0,5} = w_{0,5} + \alpha \cdot \delta_5 \cdot 1$$

$$\delta_3 = o_3 \cdot (1 - o_3) \cdot (\delta_5 \cdot\ w_{3,5} + \delta_6 \cdot\ w_{3,6})$$

$$\delta_4 = o_4 \cdot (1 - o_4) \cdot (\delta_5 \cdot\ w_{4,5} + \delta_6 \cdot\ w_{4,6})$$

$$w_{1,3} = w_{1,3} + \alpha \cdot \delta_3 \cdot x_1 \qquad w_{1,4} = w_{1,4} + \alpha \cdot \delta_4 \cdot x_1$$

$$w_{2,3} = w_{2,3} + \alpha \cdot \delta_3 \cdot x_2 \qquad w_{2,4} = w_{2,4} + \alpha \cdot \delta_4 \cdot x_2$$

$$w_{0,3} = w_{0,3} + \alpha \cdot \delta_3 \cdot 1 \qquad w_{0,4} = w_{0,4} + \alpha \cdot \delta_4 \cdot 1$$

BP algorithm (stochastic + sigmoid)

- Step0 define # of layers, # of nodes
- Step1 initialize parameters: weights and bias
- Step2 feed forward
  - Calculate output for each non-input layer node
- Step3 back propagate
  - Compute error for each non-input layer node i
  - Update parameters
- Step4 Convergence
  - Compute cost function
  - Repeat step2 until convergence

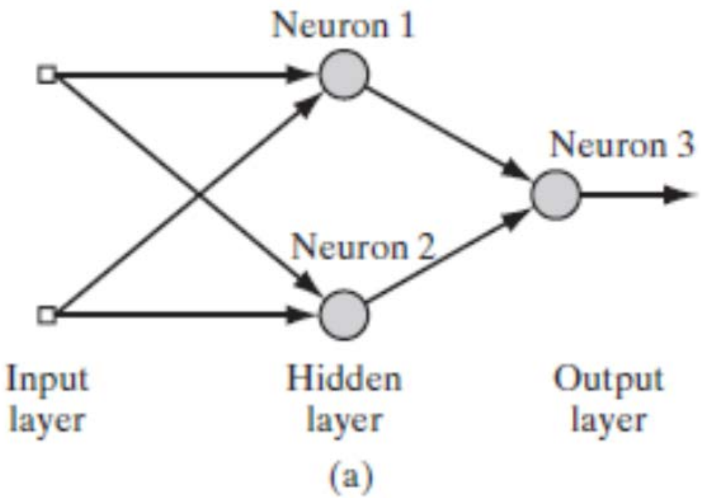
$$\delta_i = error_i \cdot o_i \cdot (1 - o_i)$$

$$error_i = (y - o_i) \qquad \text{output-layer}$$

$$error_i = \sum_j \delta_i * w_{i,j} \qquad \text{hidden-layer}$$

$$w_{k,i} = w_{k,i} + \alpha \cdot \delta_i \cdot o_k$$

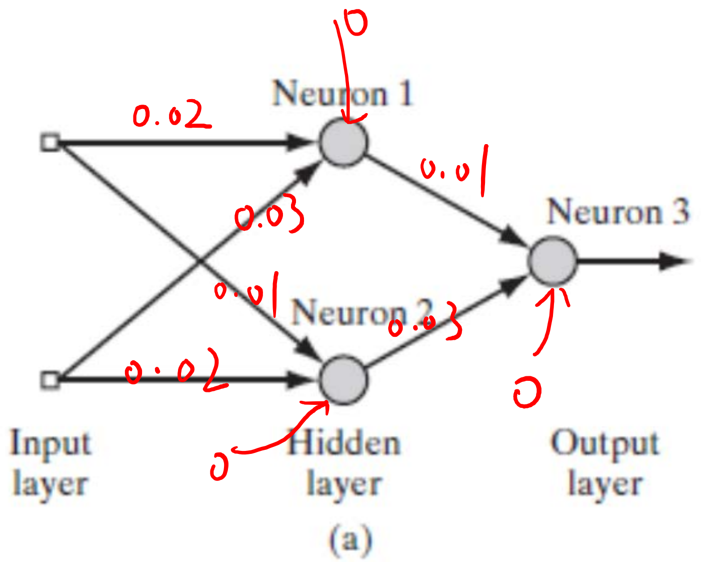
XOR with 2 nodes hidden layer



X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

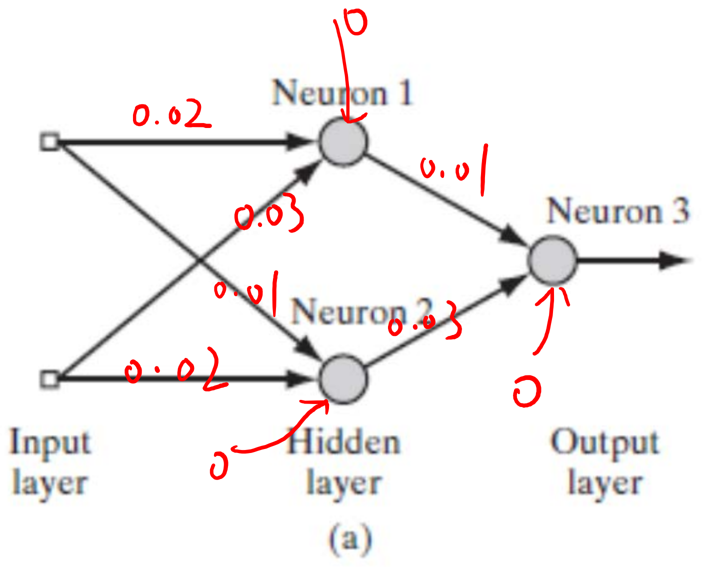
Parameter initialization

- To avoid symmetry breaking problem, the initial value for the weights would not be zeros any longer,
- small randomized values
  - $w_{i1,1} = 0.02$ (randomly generated)
  - $w_{i2,1} = 0.03$
  - $w_{0,1} = 0$
  - $w_{i1,2} = 0.01$
  - $w_{i2,2} = 0.02$
  - $w_{0,2} = 0$
  - $w_{1,3} = 0.01$
  - $w_{2,3} = 0.03$
  - $w_{0,3} = 0$

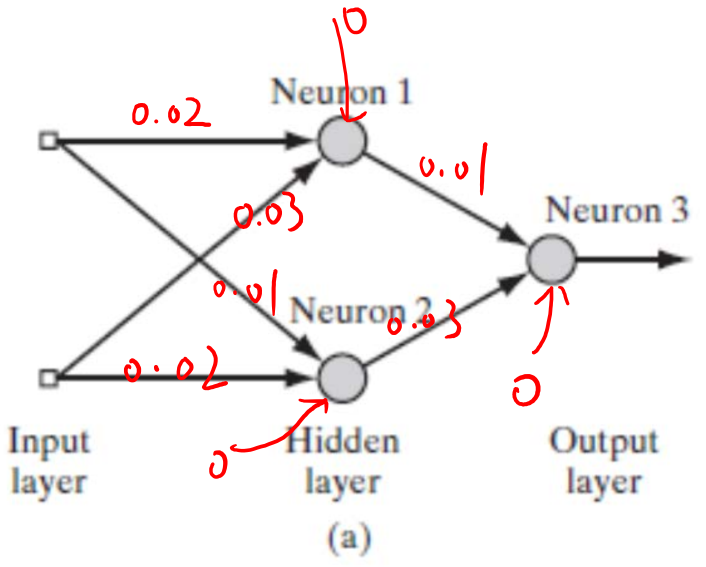


Feed forward – input(0,0)

- Instead, they would be a small randomized value
  - $o_1 = sigmoid(0) \ = 0.5$
  - $o_2 = sigmoid(0) \ = 0.5$
  - $o_3 = sigmoid(0.5 \cdot 0.01 + 0.5 \cdot 0.03 + 0) = 0.505$

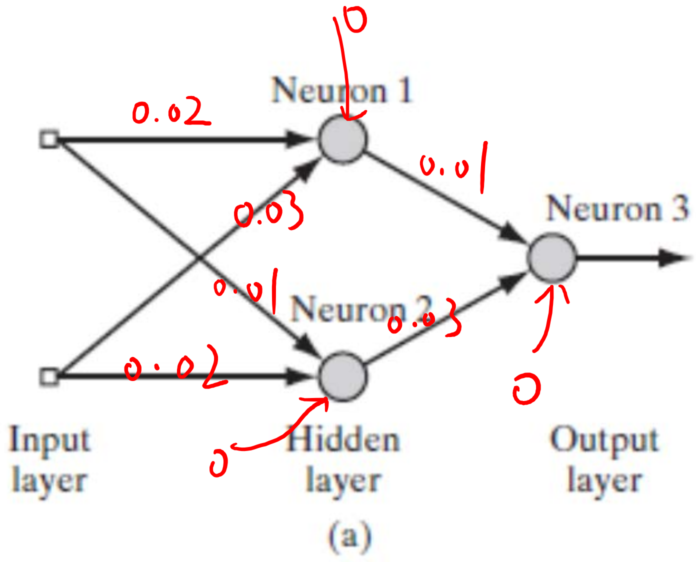


Back propagate – input(0,0)



- We have
  - o[3]={0.5, 0.5, 0.505}
  - $o_3 = 0.505$, and y = 0
- Error (output)
  - $\delta_3 = (y - o_3) \cdot o_3 \cdot (1 - o_3) = (0 - 0.505)0.505(1 - 0.505) = -0.126$
- Hidden-Output Weight
  - $w_{1,3} = w_{1,3} + \alpha \cdot \delta_3 \cdot o_1 = 0.01 + 0.1 \cdot -0.126 \cdot 0.5 = -0.0037$
  - $w_{2,3} = w_{2,3} + \alpha \cdot \delta_3 \cdot o_2 = 0.03 + 0.1 \cdot -0.126 \cdot 0.5 = -0.0237$
  - $w_{0,3} = w_{0,3} + \alpha \cdot \delta_3 = 0 + 0.1 \cdot -0.126 = -0.0126$

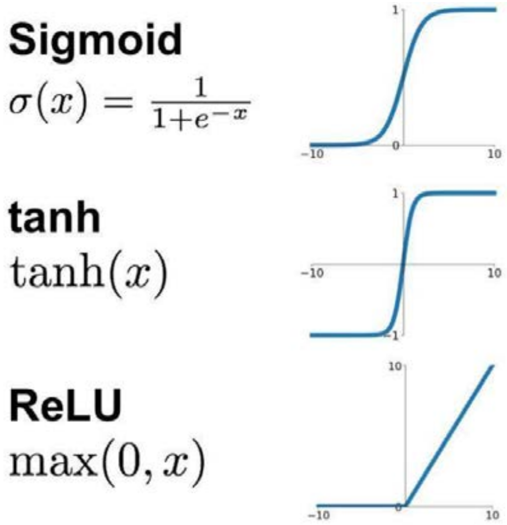
Back propagate – input(0,0)



- We have
  - o[3]={0.5, 0.5, 0.505}
  - δ[3]={?, ?, -0.126}
- Error (hidden-output)
  - $\delta_1 = (\delta_3 w_{1,3}) o_1 (1 - o_1) = (-0.126) \cdot 0.01 \cdot 0.5 \cdot (1 - 0.5) = -0.000315$
  - $\delta_2 = (\delta_3 w_{2,3}) o_2 (1 - o_2) = (-0.126) \cdot 0.03 \cdot 0.5 \cdot (1 - 0.5) = -0.000945$
- Input-Hidden
  - $w_{i1,1} = w_{i1,1} + \alpha \cdot \delta_1 \cdot x_1 = 0.02 + 0.1 \cdot -0.000315 \cdot 0 = 0.02$
  - $w_{i2,1} = w_{i2,1} + \alpha \cdot \delta_1 \cdot x_2 = 0.03 + 0.1 \cdot -0.000315 \cdot 0 = 0.03$
  - $w_{0,1} = w_{0,1} + \alpha \cdot \delta_1 = 0 + 0.1 \cdot -0.000315 = -0.0000315$
  - $w_{i1,2} = w_{i1,2} + \alpha \cdot \delta_2 \cdot x_1 = 0.01 + 0.1 \cdot -0.000945 \cdot 0 = 0.01$
  - $w_{i2,2} = w_{i2,2} + \alpha \cdot \delta_2 \cdot x_2 = 0.02 + 0.1 \cdot -0.000945 \cdot 0 = 0.02$
  - $w_{0,2} = w_{0,2} + \alpha \cdot \delta_2 = 0 + 0.1 \cdot -0.000945 = -0.0000945$

Activation Functions

- sigmoid
  - Gradient vanishing
  - Output is not zero-centered – slow down the learning
  - Power operation – time cost
- tanh (Hyperbolic Tangent)
  - Gradient vanishing
  - Output is not zero-centered
  - Power operation
- Relu
  - Gradient vanishing
  - Output is not zero-centered
  - Power operation
  - Dead Relu Problem



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

$$\sigma(x) \cdot (1 - \sigma(x))$$

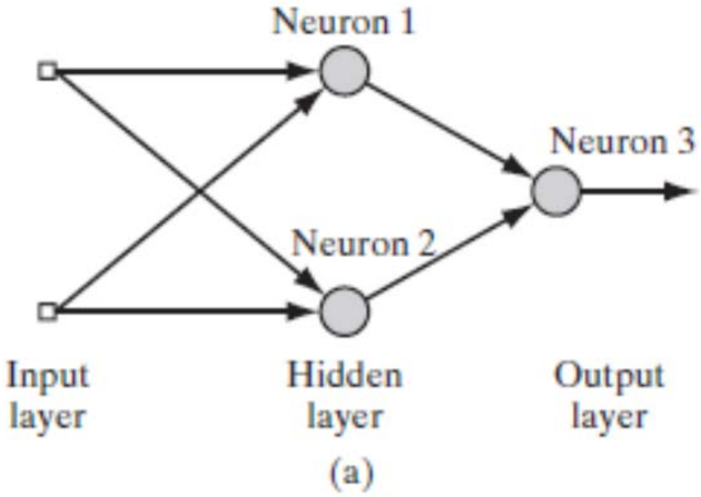
$$\tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

$$1 - f(x)^2$$

$$\mathrm{ReLU} = \max(0, x)$$

$$f'(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases}$$

XOR with 2 nodes hidden layer - vectorization



- Hidden layer
  - tanh
- Output layer
  - sigmoid

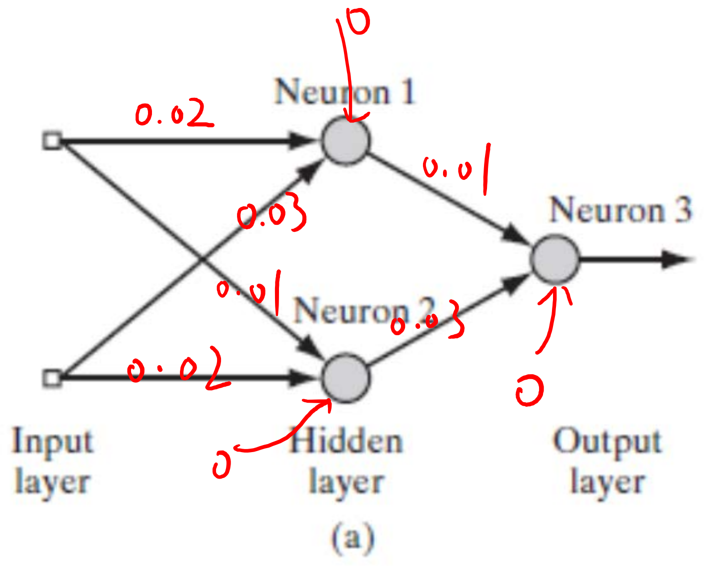
X0	X1	X2	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

- X - 4×3
- Y - 4×1

Parameter initialization

- Hidden layer
  - Wh = $\begin{bmatrix} 0 & 0.02 & 0.03 \\ 0 & 0.01 & 0.02 \end{bmatrix}$
- Output layer
  - Wo = $[0 \quad 0.01 \quad 0.03]$

$$
\begin{aligned}
w_{i1,1} &= 0.02 \\
w_{i2,1} &= 0.03 \\
w_{0,1} &= 0 \\
w_{i1,2} &= 0.01 \\
w_{i2,2} &= 0.02 \\
w_{0,2} &= 0 \\
w_{1,3} &= 0.01 \\
w_{2,3} &= 0.03 \\
w_{0,3} &= 0
\end{aligned}
$$

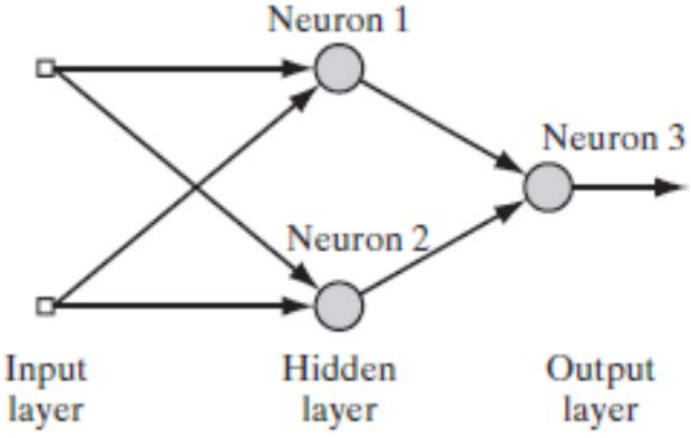


Feed forward – Hidden layer output

- X = $\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

- Wh = $\begin{bmatrix} 0 & 0.02 & 0.03 \\ 0 & 0.01 & 0.02 \end{bmatrix}$

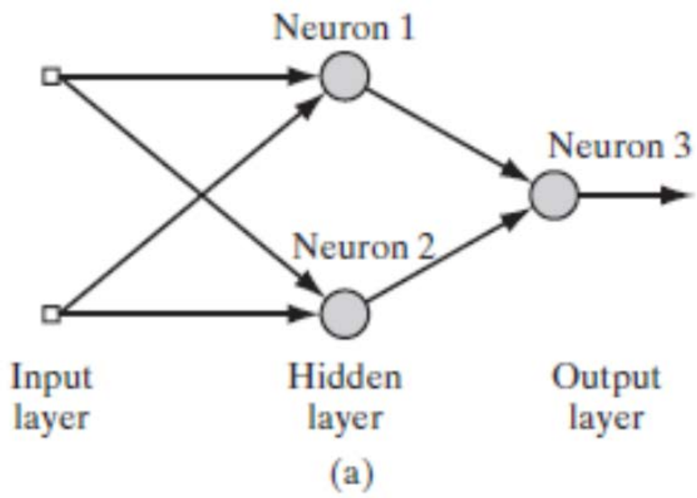
- Oh=tanh(Wh $\cdot X^T$) $= \begin{bmatrix} o_1^{(1)} & o_1^{(2)} & o_1^{(3)} & o_1^{(4)} \\ o_2^{(1)} & o_2^{(2)} & o_2^{(3)} & o_2^{(4)} \end{bmatrix}$



(a)

Feed forward – output layer output

- Oh= $\begin{bmatrix} o_1^{(1)} & o_1^{(2)} & o_1^{(3)} & o_1^{(4)} \\ o_2^{(1)} & o_2^{(2)} & o_2^{(3)} & o_2^{(4)} \end{bmatrix}$

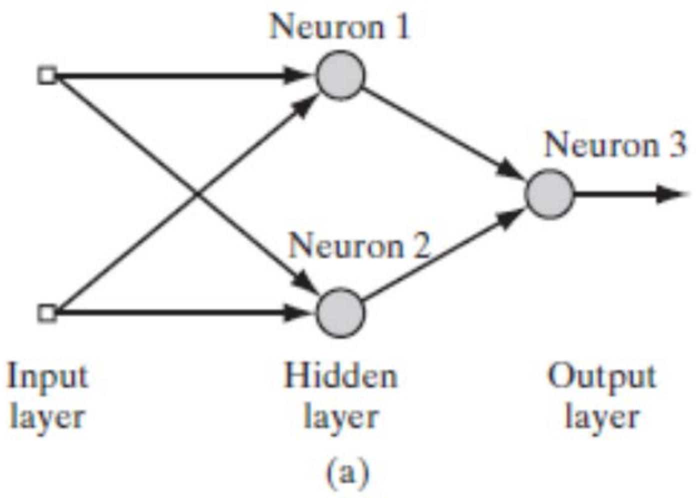


- Ino= $\begin{bmatrix} 1 & 1 & 1 & 1 \\ o_1^{(1)} & o_1^{(2)} & o_1^{(3)} & o_1^{(4)} \\ o_2^{(1)} & o_2^{(2)} & o_2^{(3)} & o_2^{(4)} \end{bmatrix}$

Wo =$[0 \quad 0.01 \quad 0.03]$

- Oo=sigmoid(Wo $\cdot\mathrm{Ino}) = \begin{bmatrix} o_3^{(1)} & o_3^{(2)} & o_3^{(3)} & o_3^{(4)} \end{bmatrix}$

Output layer - Error computing



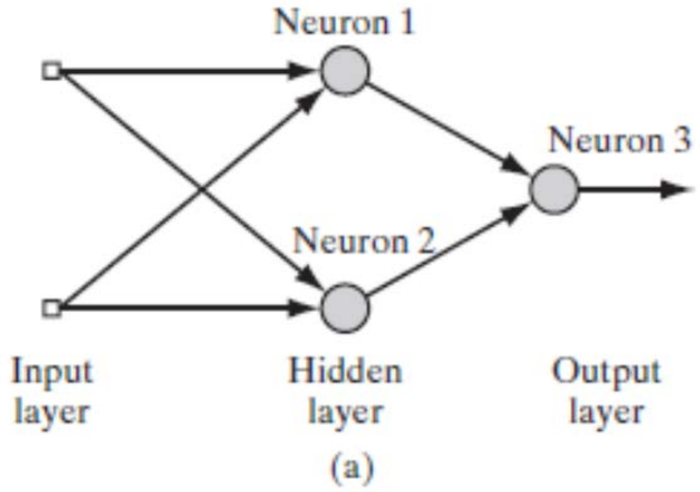
- Oo=$\begin{bmatrix} o_3^{(1)} & o_3^{(2)} & o_3^{(3)} & o_3^{(4)} \end{bmatrix}$

- $\delta o = (Y^T - Oo) \circ Oo \circ (1 - Oo)$ = $\begin{bmatrix} \delta_3^{(1)} & \delta_3^{(2)} & \delta_3^{(3)} & \delta_3^{(4)} \end{bmatrix}$
  - $(\circ\text{:}\ element\ wise\ product)$

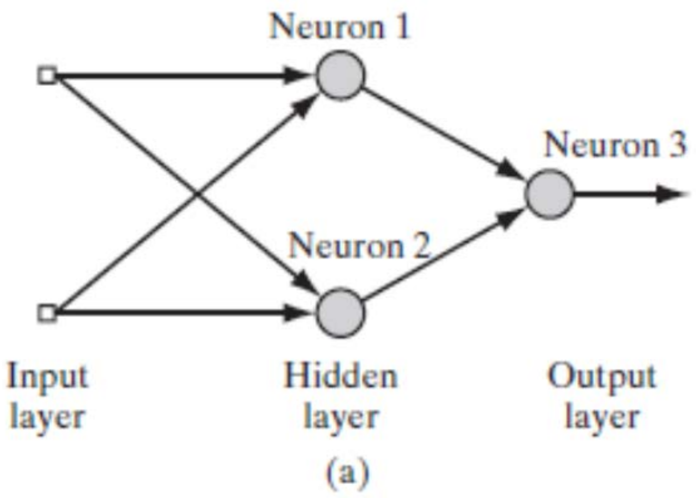
Hidden – output weight update

- $\delta o = \begin{bmatrix} \delta_3^{(1)} & \delta_3^{(2)} & \delta_3^{(3)} & \delta_3^{(4)} \end{bmatrix}$

- Wo = Wo + α·(δo · Ino^T)/4



Hidden layer - Error computing



- $\delta o = \begin{bmatrix} \delta_3^{(1)} & \delta_3^{(2)} & \delta_3^{(3)} & \delta_3^{(4)} \end{bmatrix}$

Wo =$[0 \quad 0.01 \quad 0.03]$ Wo' =$[0.01 \quad 0.03]$

Oh= $\begin{bmatrix} o_1^{(1)} & o_1^{(2)} & o_1^{(3)} & o_1^{(4)} \\ o_2^{(1)} & o_2^{(2)} & o_2^{(3)} & o_2^{(4)} \end{bmatrix}$

- $\delta h = (Wo'^T \cdot \delta o)$ ∘(1-Oh ∘ Oh)= $\begin{bmatrix} \delta_1^{(1)} & \delta_1^{(2)} & \delta_1^{(3)} & \delta_1^{(4)} \\ \delta_2^{(1)} & \delta_2^{(2)} & \delta_2^{(3)} & \delta_2^{(4)} \end{bmatrix}$

input – hidden weight update

- $\delta h = \begin{bmatrix} \delta_1^{(1)} & \delta_1^{(2)} & \delta_1^{(3)} & \delta_1^{(4)} \\ \delta_2^{(1)} & \delta_2^{(2)} & \delta_2^{(3)} & \delta_2^{(4)} \end{bmatrix}$

- Wh = Wh + α·(δh · X)/4