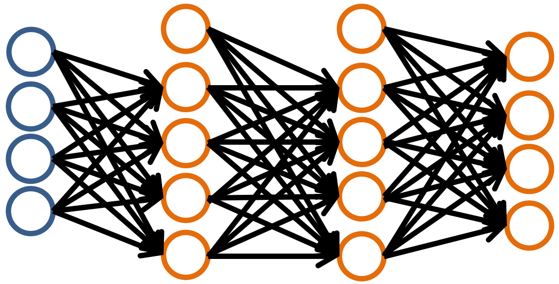
BackPropagation Algorithm

Lecturer: MaoQiang Xie
College of Software, Nankai University

本节内容主要来自于吴恩达Coursera网课

Neural Network (Classification)



$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$$

$L =$ total no. of layers in network

$s_l =$ no. of l units (not counting bias unit) in layer

Binary classification

$y = 0 \text{ or } 1$

1 output unit

Multi-class classification (K classes)

$y \in \mathbb{R}^K$ E.g. $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

pedestrian car motorcycle truck

K output units

Cost function

Logistic regression (Cross Entropy):

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Neural network (Cross Entropy & Regularization):

$$h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}$$

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1-y_k^{(i)}) \log(1-(h_\Theta(x^{(i)}))_k)\right]$$
$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

与基于平方误差的损失函数的对比

基于交叉熵的损失函数：

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1-y_k^{(i)})\log(1-(h_\Theta(x^{(i)}))_k)\right]$$
$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

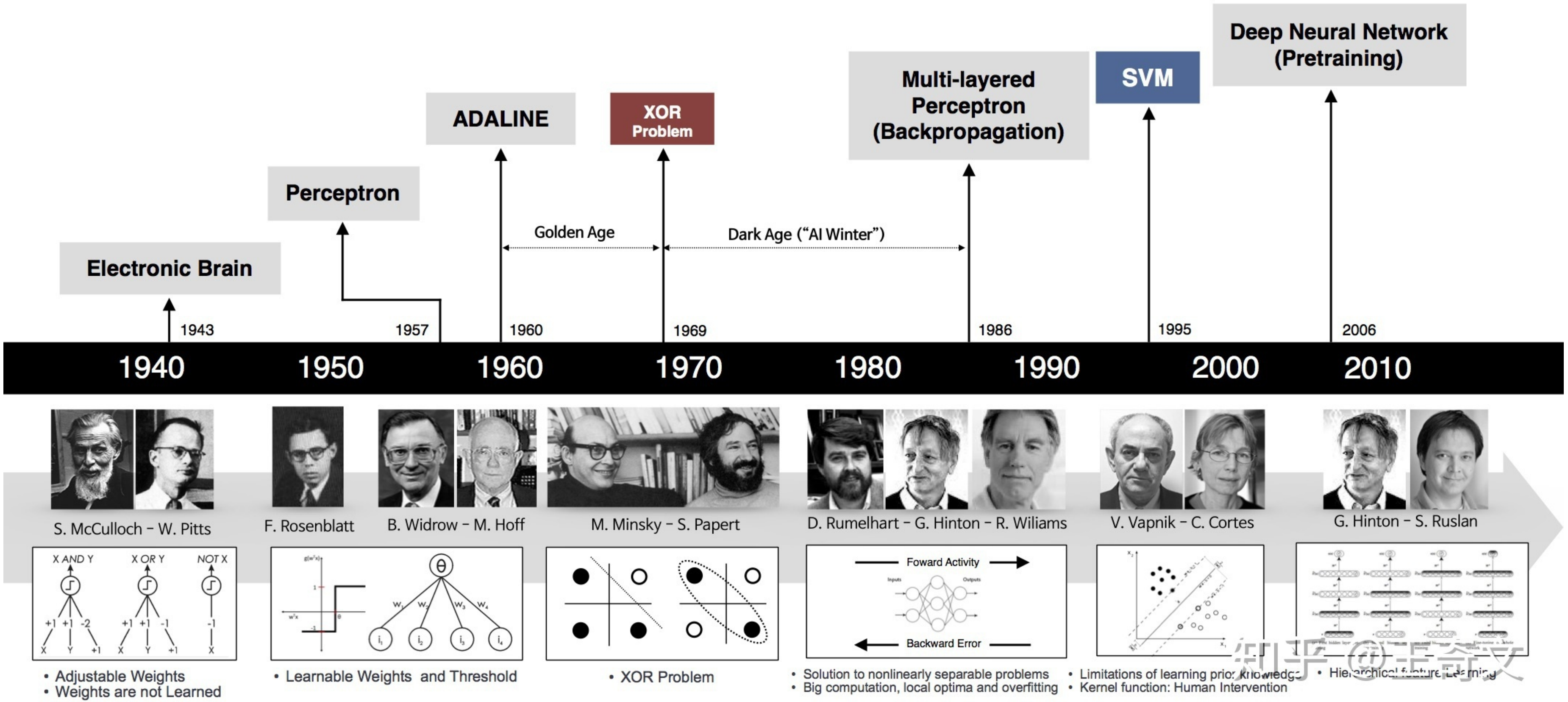
基于平方误差的损失函数：（基于教材P102公式(5.4)）

$$E_k = \frac{1}{2}\sum_{k=1}^{m}\sum_{j=1}^{l}(\hat{y}_j^k - y_j^k)^2$$

神经网络损失函数的优化

BackPropagation Algorithm (BP算法)

- 误差反向传播算法：用于多层前馈网络的训练
- [Werbos, 1974]， [Rumelhart, Hinton et al., 1986]


Deep Neural Network (Pretraining)
SVM
Multi-layered Perceptron (Backpropagation)
XOR Problem
ADALINE
Perceptron
Electronic Brain
Golden Age
Dark Age (“AI Winter”)
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch – W. Pitts
F. Rosenblatt
B. Widrow – M. Hoff
M. Minsky – S. Papert
D. Rumelhart – G. Hinton – R. Wiliams
V. Vapnik – C. Cortes
G. Hinton – S. Ruslan
X AND Y
X OR Y
NOT X
Foward Activity
Backward Error
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
• Solution to nonlinearly separable problems
• Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning

神经网络损失函数的优化

BackPropagation Algorithm (BP算法)

- 误差反向传播算法：用于多层前馈网络的训练
- [Werbos, 1974]， [Rumelhart, Hinton et al., 1986]

算法价值在于：最终层的误差如何分解到每个节点的输出上。

- 将误差分解到各节点的输出上，才能够通过梯度调整各节点的输入权重。

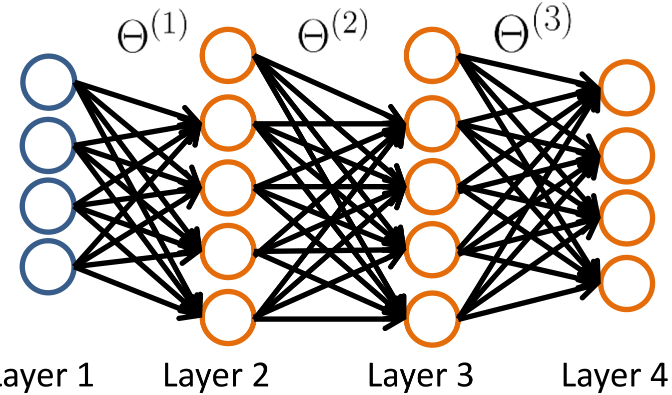
Gradient computation (基于交叉熵)

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log h_\theta(x^{(i)})_k + (1-y_k^{(i)})\log(1-h_\theta(x^{(i)})_k)\right]$$
$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_j^{(l)})^2$$

$$\min_{\Theta} J(\Theta)$$

Need code to compute:

- $J(\Theta)$
- $\frac{\partial}{\partial\Theta_{ij}^{(l)}} J(\Theta)$



$z^{(2)} = \Theta^{(1)}a^{(1)}$ $z^{(3)} = \Theta^{(2)}a^{(2)}$ $z^{(4)} = \Theta^{(3)}a^{(3)}$

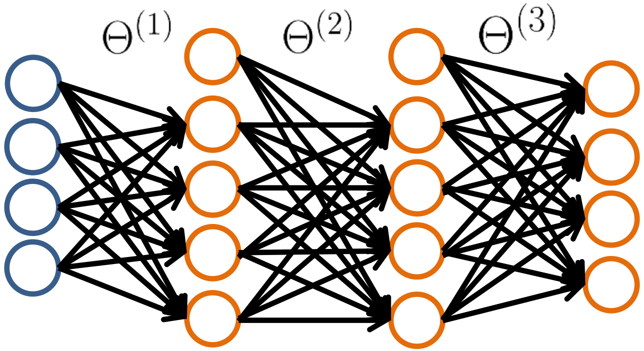
$a^{(1)} = x$ $a^{(2)} = g(z^{(2)})$ $a^{(3)} = g(z^{(3)})$ $a^{(4)} = g(z^{(4)}) = h_\Theta(\boldsymbol{x})$

$(\text{add } a_0^{(1)})$ $(\text{add } a_0^{(2)})$ $(\text{add } a_0^{(3)})$

Gradient computation: Backpropagation

Intuition: $\delta_j^{(l)}$ = "error" of node j in layer l.

损失函数相对于模型参数(网络连接权重)的梯度使用链式求导法计算



Layer 1 Layer 2 Layer 3 Layer 4

$$z^{(2)} = \Theta^{(1)} a^{(1)} \quad z^{(3)} = \Theta^{(2)} a^{(2)} \quad z^{(4)} = \Theta^{(3)} a^{(3)}$$

$$a^{(1)} = x \quad a^{(2)} = g(z^{(2)}) \quad a^{(3)} = g(z^{(3)}) \quad a^{(4)} = g(z^{(4)}) = h_\Theta(\boldsymbol{x})$$

$$(\text{add } a_0^{(1)}) \quad (\text{add } a_0^{(2)}) \quad (\text{add } a_0^{(3)})$$

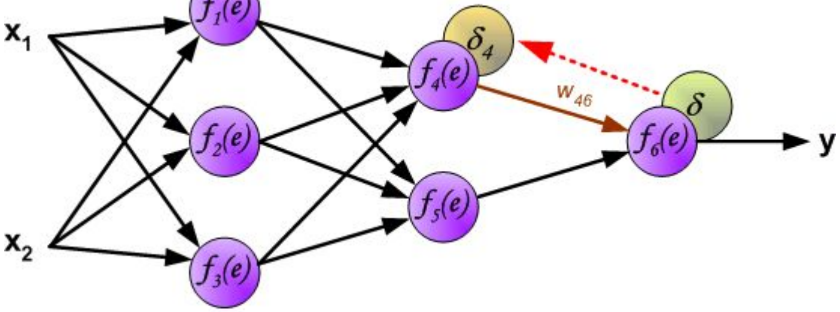
$$\delta^{(4)} = \frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}}$$

$$\delta^{(3)} = \underbrace{\frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}}}_{\delta^{(4)}} \frac{\partial z^{(4)}}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(3)}}$$

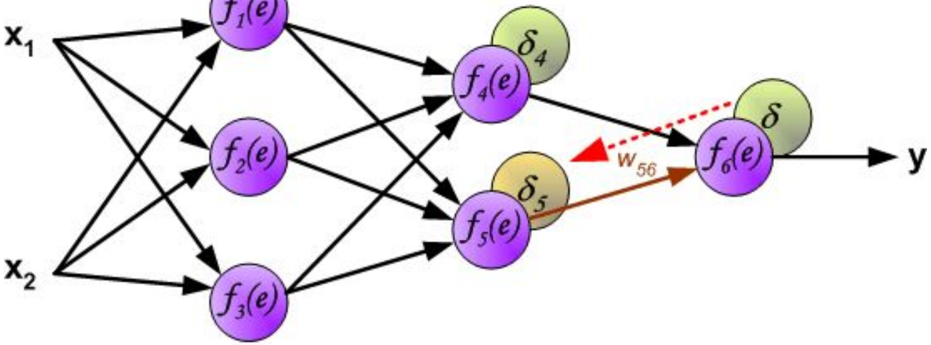
$$\delta^{(2)} = \underbrace{\frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}} \cdot \frac{\partial z^{(4)}}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(3)}}}_{\delta^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}}$$

误差的反向传递示意

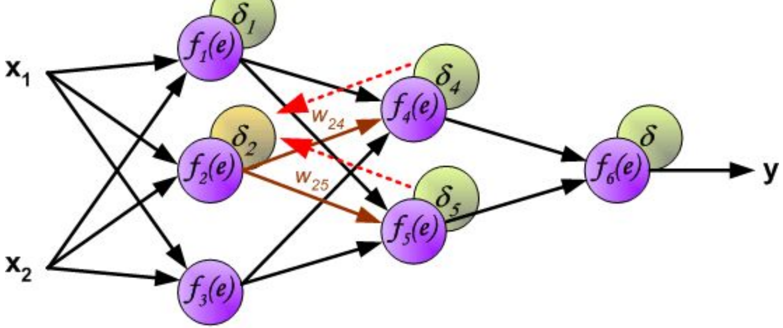
$$\delta_4 = w_{46}\delta$$



$$\delta_5 = w_{56}\delta$$



$$\delta_2 = w_{24}\delta_4 + w_{25}\delta_5$$



Gradient computation

对于一个样本和一个输出层节点的误差相对于模型参数的梯度：

$$\frac{\partial J}{\partial \Theta^{(2)}} = \underbrace{\frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}} \cdot \frac{\partial z^{(4)}}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(3)}}}_{\delta^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial \Theta^{(2)}}$$

$$\Theta_{ij}^{(2)}(t+1) = \Theta_{ij}^{(2)}(t) - \alpha \times \frac{\partial J}{\partial \Theta_{ij}^{(2)}}$$

Gradient computation: Backpropagation algorithm(cross entropy cost function)

$$\delta^{(4)} = \frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}} = a^{(4)} - y$$

$$\frac{\partial J}{\partial a^{(4)}} = \frac{a^{(4)} - y^{(i)}}{a^{(4)}(1 - a^{(4)})}$$

求导见下一页

$$\frac{\partial a^{(4)}}{\partial z^{(4)}} = a^{(4)}(1 - a^{(4)})$$

求导见下下页

GOAL: $$\delta^{(4)} = \frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}} = a^{(4)} - y$$

$$\begin{aligned}
\frac{\partial J}{\partial a^{(4)}} &= \frac{\partial}{\partial a^{(4)}} \left[- \left(y^{(i)} \mathbf{log} a^{(4)} + (1 - y^{(i)}) \mathbf{log}(1 - a^{(4)}) \right) \right] \\
&= -y^{(i)} \frac{1}{a^{(4)}} - (1 - y^{(i)}) \cdot \frac{1}{1 - a^{(4)}} \cdot (-1) \\
&= -\frac{y^{(i)}}{a^{(4)}} + \frac{1 - y^{(i)}}{1 - a^{(4)}} \\
&= \frac{-y^{(i)} + y^{(i)} a^{(4)} + a^{(4)} - a^{(4)} y^{(i)}}{a^{(4)}(1 - a^{(4)})} \\
&= \frac{a^{(4)} - y^{(i)}}{a^{(4)}(1 - a^{(4)})}
\end{aligned}$$

GOAL: $$\delta^{(4)} = \frac{\partial J}{\partial a^{(4)}}\frac{\partial a^{(4)}}{\partial z^{(4)}} = a^{(4)} - y$$

$$\begin{aligned}\frac{\partial a^{(4)}}{\partial z^{(4)}} &= \frac{\partial}{\partial z^{(4)}}\left(\frac{1}{1+e^{-z^{(4)}}}\right)\\ &= \frac{-1}{(1+e^{-z^{(4)}})^2}\cdot\frac{\partial}{\partial z^{(4)}}(e^{-z^{(4)}})\\ &= \frac{-1}{(1+e^{-z^{(4)}})^2}\cdot e^{-z^{(4)}}\cdot(-1)\\ &= \frac{1}{1+e^{-z^{(4)}}}\cdot\frac{e^{-z^{(4)}}}{1+e^{-z^{(4)}}}\\ &= a^{(4)}(1-a^{(4)})\end{aligned}$$

对于sigmoid函数

$$g(z) = \frac{1}{1+e^{-z}}$$

其导数为：

$$g'(z) = g(z)(1-g(z))$$

汇总整理

$$\delta^{(4)} = \frac{\partial J}{\partial a^{(4)}}\frac{\partial a^{(4)}}{\partial z^{(4)}} = a^{(4)} - y$$

$$\delta^{(3)} = \frac{\partial J}{\partial a^{(4)}}\frac{\partial a^{(4)}}{\partial z^{(4)}}\frac{\partial z^{(4)}}{\partial a^{(3)}}\frac{\partial a^{(3)}}{\partial z^{(3)}}$$

$$= (\Theta^{(3)})^T \times \delta^{(4)} \cdot *g'(z^{(3)})$$

$$\delta^{(2)} = \frac{\partial J}{\partial a^{(4)}}\frac{\partial a^{(4)}}{\partial z^{(4)}} \cdot \frac{\partial z^{(4)}}{\partial a^{(3)}}\frac{\partial a^{(3)}}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial a^{(2)}}\frac{\partial a^{(2)}}{\partial z^{(2)}}$$

$$= (\Theta^{(2)})^T \times \delta^{(3)} \cdot *g'(z^{(2)})$$

$$\frac{\partial J}{\partial \Theta^{(3)}} = \delta^{(4)} \bullet \frac{\partial z^{(4)}}{\partial \Theta^{(3)}} = \delta^{(4)} \bullet a^{(3)}$$

$$\frac{\partial J}{\partial \Theta^{(2)}} = \delta^{(3)} \bullet \frac{\partial z^{(3)}}{\partial \Theta^{(2)}} = \delta^{(3)} \bullet a^{(2)}$$

$$\frac{\partial J}{\partial \Theta^{(1)}} = \delta^{(2)} \cdot \frac{\partial z^{(2)}}{\partial \Theta^{(1)}} = \delta^{(2)} \cdot a^{(1)}$$

Gradient Computation for Backpropagation Algorithm

Training set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

1. Set $\triangle_{ij}^{(l)} = 0$ (for all l, i, j).
2. For $i = 1$ to m

   Set $a^{(1)} = x^{(i)}$

   Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \ldots, L$

   Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$

   Compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$

   $\triangle_{ij}^{(l)} := \triangle_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$ 为每个样本累计误差delta
3. $$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)} = \begin{cases} \frac{1}{m}\Delta_{ij}^{(l)} + \frac{\lambda}{m}\theta_{ij}^{(l)} & \text{if } j \neq 0 \\ \frac{1}{m}\Delta_{ij}^{(l)} & \text{if } j = 0 \end{cases}$$

Backpropagation Algorithm (by Gradient Descent)

Input： Training set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

Learning Rate： η

Initialize the weights of links (`randInitializeWeights.m`)

repeat

for all $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$in training set

Compute all gradients $\dfrac{\partial J(\Theta)}{\partial \Theta_{ij}^{(l)}}$

Update all link weights $\Theta_{ij}^{(l)}(t+1) := \Theta_{ij}^{(l)}(t) - \eta \dfrac{\partial J(\Theta)}{\partial \Theta_{ij}^{(l)}}$

end

until converge or reach maximal iterations

Backpropagation Algorithm 2

- 输入与前向传播：

1. Set the input layer's values ($a^{(1)}$) to the t-th training example $x^{(t)}$. Perform a feedforward pass (Figure 2), computing the activations ($z^{(2)}$, $a^{(2)}, z^{(3)}, a^{(3)}$) for layers 2 and 3.

- 输出层误差计算：

2. For each output unit k in layer 3 (the output layer), set

$$\delta_k^{(3)} = (a_k^{(3)} - y_k),$$

- 面向隐含层的误差反向传播：

3. For the hidden layer $l = 2$, set

$$\delta^{(2)} = \left(\Theta^{(2)}\right)^T \delta^{(3)} .* g'(z^{(2)})$$

Backpropagation Algorithm 2

- 使用误差梯度计算模型参数梯度

4. Accumulate the gradient from this example using the following formula.

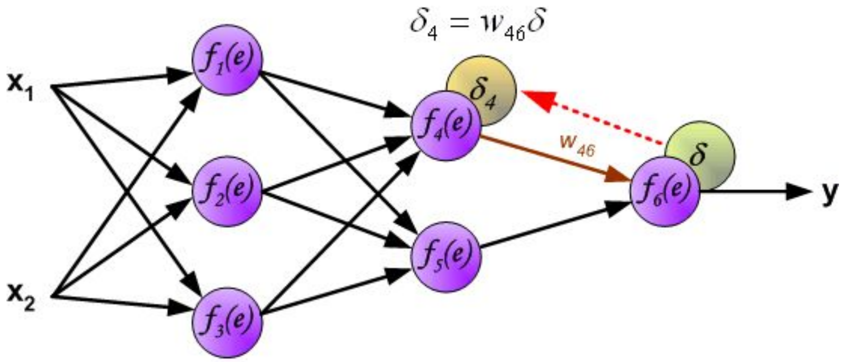
$$\Delta^{(l)} = \Delta^{(l)} + \delta^{(l+1)}(a^{(l)})^T$$

5. Obtain the (unregularized) gradient for the neural network cost function by dividing the accumulated gradients by $\frac{1}{m}$:

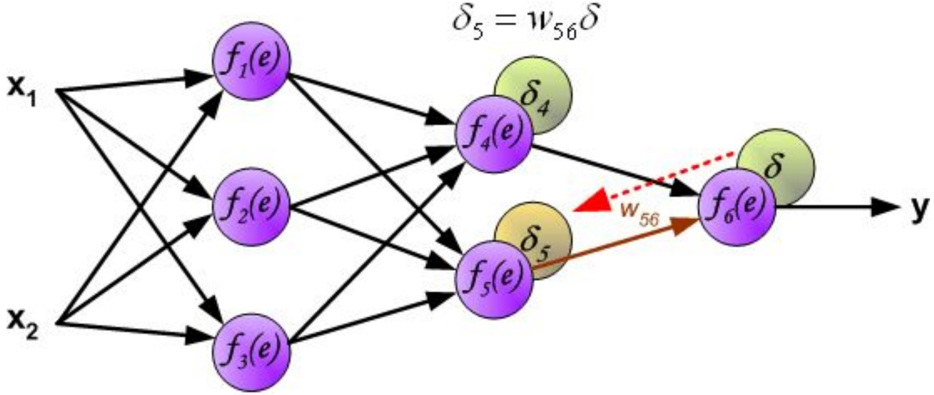
$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)} = \frac{1}{m}\Delta_{ij}^{(l)}$$

- 使用梯度下降法或梯度共轭法优化模型参数

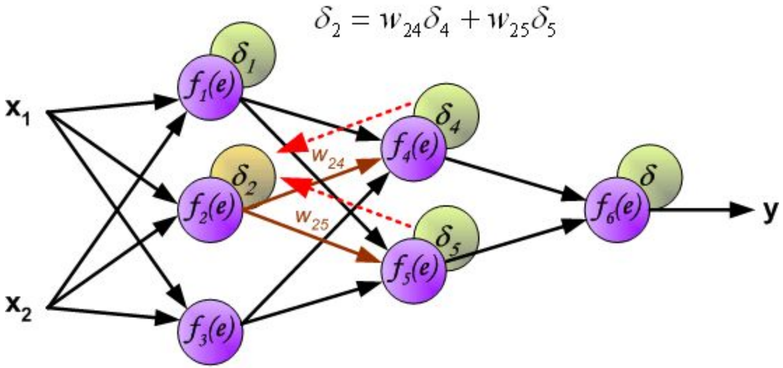
误差的反向传递示意



$\delta_4 = w_{46}\delta$
x_1
x_2
$f_1(e)$
$f_2(e)$
$f_3(e)$
$f_4(e)$
$f_5(e)$
$f_6(e)$
δ_4
δ
w_{46}
y



$\delta_5 = w_{56}\delta$
x_1
x_2
$f_1(e)$
$f_2(e)$
$f_3(e)$
$f_4(e)$
$f_5(e)$
$f_6(e)$
δ_4
δ_5
δ
w_{56}
y



$\delta_2 = w_{24}\delta_4 + w_{25}\delta_5$
x_1
x_2
$f_1(e)$
$f_2(e)$
$f_3(e)$
$f_4(e)$
$f_5(e)$
$f_6(e)$
δ_1
δ_2
δ_4
δ_5
δ
w_{24}
w_{25}
y

问题：如果将损失函数更换为误差平方和

$$E=\frac{1}{2}\sum_{i=1}^{m}\left(a^{(4)}-y^{(i)}\right)^2$$

请求解反向传递误差和对应的梯度

$$\delta=\frac{\partial E}{\partial z^{(4)}}=\frac{\partial E}{\partial a^{(4)}}\frac{\partial a^{(4)}}{\partial z^{(4)}} \qquad \frac{\partial E}{\partial \Theta_{ij}^{(3)}}=\delta^{(4)}\frac{\partial z^{(4)}}{\partial \Theta_{ij}^{(3)}}$$

请参见周志华《机器学习》第5章