Convolutional Neural Network & Backpropagation Algorithm

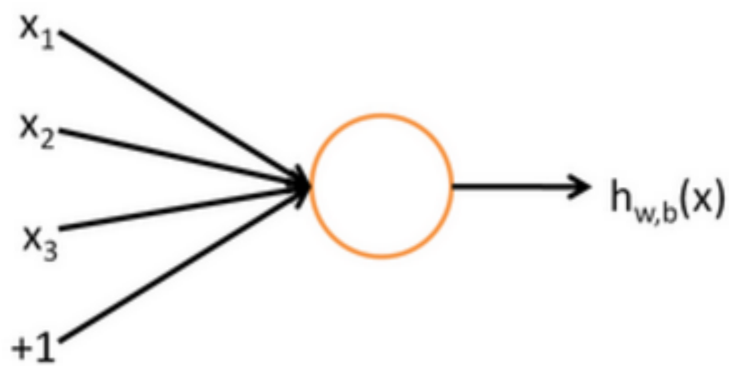
Xuelin Qian

Content

Content

1. Neural Network

"neuron"

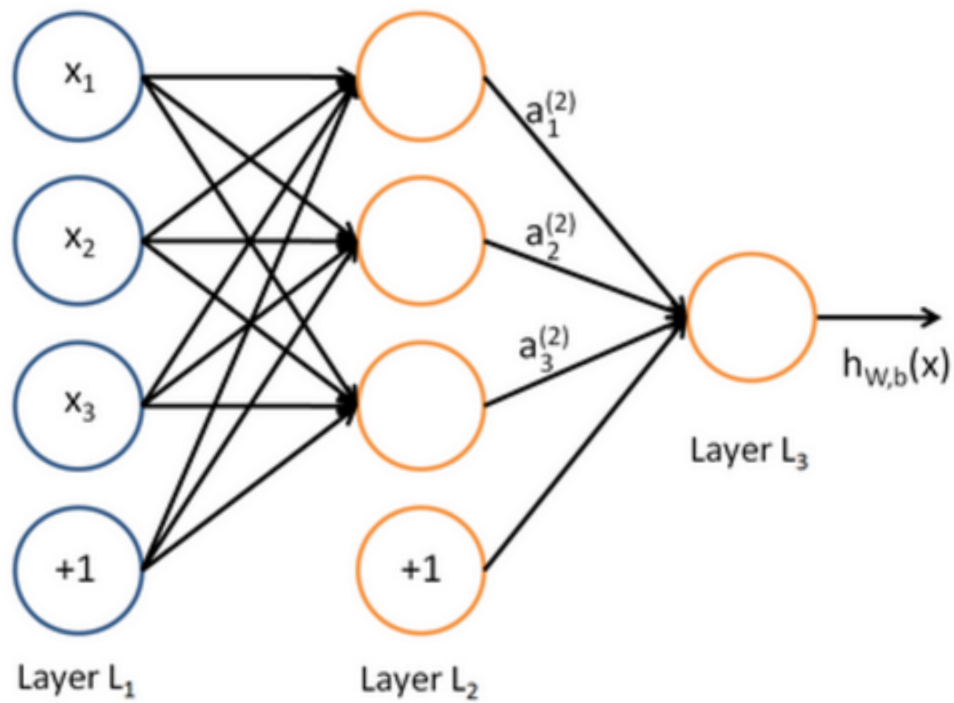


$$h_{W,b}(x) = f(W^T x) = f(\textstyle\sum_{i=1}^{3} W_i x_i + b)$$

where $f : \Re \mapsto \Re$ is called the **activation function**

1. Neural Network

neural network: consist of many simple “neurons”

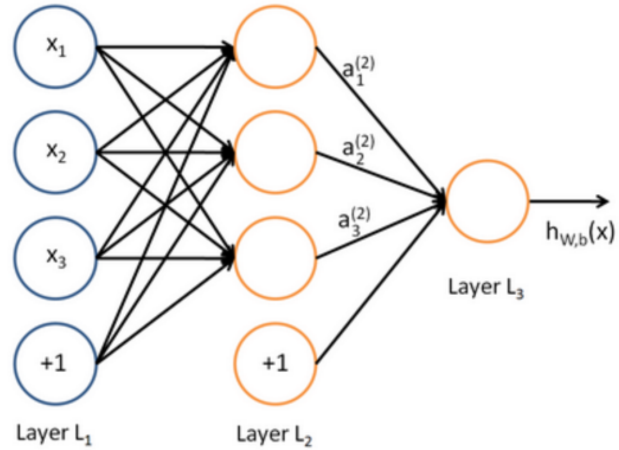


Input Layer

Hidden Layer

Output Layer

1. Neural Network



Forward Propagation

n_l : the number of layers

L_l : the layer l

$W_{ij}^{(l)}$: the weight associated with the connection between unit j in layer l, and unit i in layer $l+1$

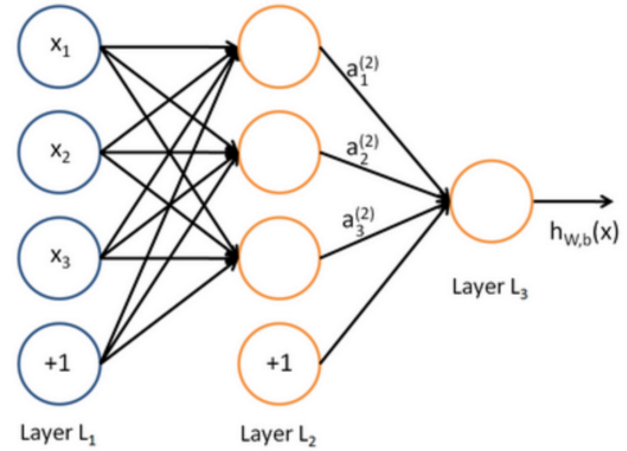
$b_i^{(l)}$: the bias associated with unit i in layer $l+1$

$a_i^{(l)}$: the acctivation unit i in layer l

$$
\begin{aligned}
a_1^{(2)} &= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\
a_2^{(2)} &= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\
a_3^{(2)} &= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\
h_{W,b}(x) = a_1^{(3)} &= f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})
\end{aligned}
$$

1. Neural Network

Forward Propagation



n_l : the number of layers

L_l : the layer l

$W_{ij}^{(l)}$: the weight associated with the connection between unit j in layer l, and unit i in layer $l+1$

$b_i^{(l)}$: the bias associated with unit i in layer $l+1$

$a_i^{(l)}$: the acctivation unit i in layer l

$z_i^{(l)}$: the total weighted sum of inputs to unit i in layer l

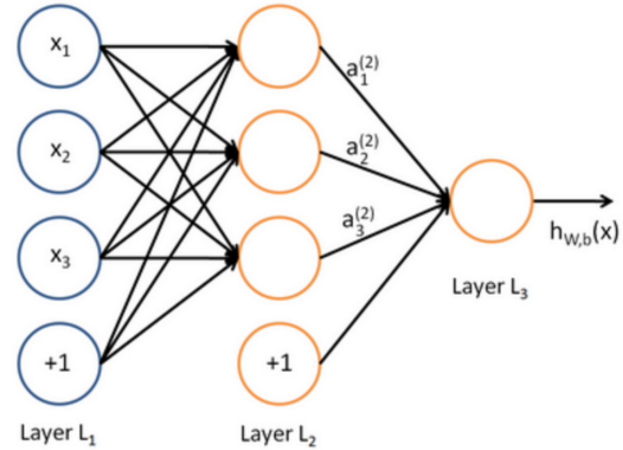
e.g.

$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$

$$a_1^{(2)} = f\left(z_1^{(2)}\right) \qquad z_1^{(2)} = \sum_{j=1}^{3} W_{1j}^{(1)}x_j + b_1^{(1)}$$

1. Neural Network

Forward Propagation



n_l : the number of layers

L_l : the layer l

$W_{ij}^{(l)}$: the weight associated with the connection between unit j in layer l, and unit i in layer $l+1$

$b_i^{(l)}$: the bias associated with unit i in layer $l+1$

$a_i^{(l)}$: the acctivation unit i in layer l

$z_i^{(l)}$: the total weighted sum of inputs to unit i in layer l

$$
\begin{aligned}
z^{(2)} &= W^{(1)}x + b^{(1)} \\
a^{(2)} &= f(z^{(2)}) \\
z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\
h_{W,b}(x) &= a^{(3)} = f(z^{(3)})
\end{aligned}
\quad \Longrightarrow \quad
\begin{aligned}
z^{(l+1)} &= W^{(l)}a^{(l)} + b^{(l)} \\
a^{(l+1)} &= f(z^{(l+1)})
\end{aligned}
$$

Content

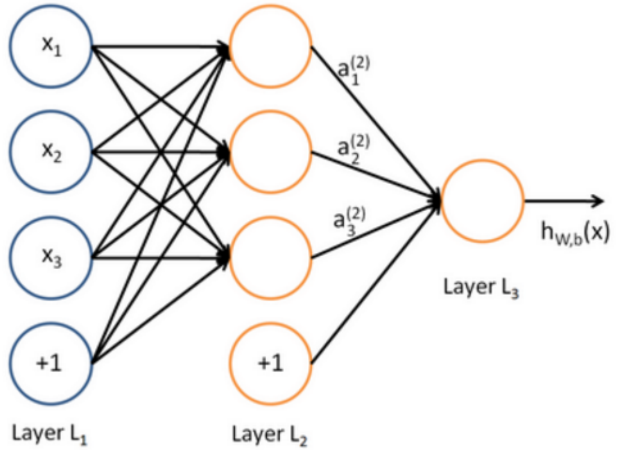
1.Neural Network

2.Backpropagation Algorithm

3.Convolutional Neural Network

4.Backpropagation on CNN

2. Backpropagation Algorithm

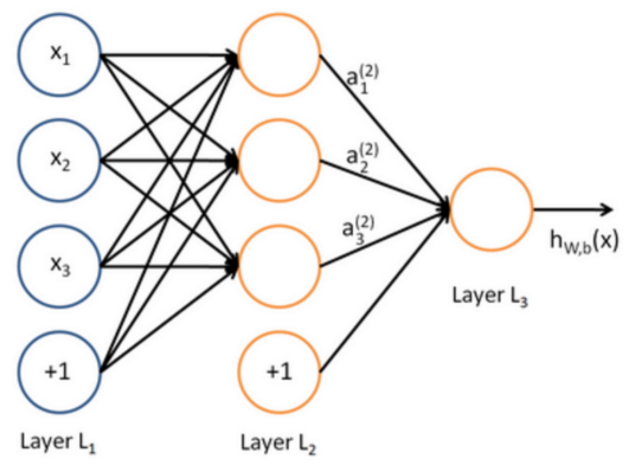


"Loss function"

$$J(W,b;x,y) = \frac{1}{2}\left\| h_{W,b}(x) - y \right\|^2 .$$

$$\begin{aligned} J(W,b) &= \left[\frac{1}{m} \sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)}) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{(l)} \right)^2 \\ &= \left[\frac{1}{m} \sum_{i=1}^{m} \left(\frac{1}{2} \left\| h_{W,b}(x^{(i)}) - y^{(i)} \right\|^2 \right) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{(l)} \right)^2 \end{aligned}$$

2. Backpropagation Algorithm



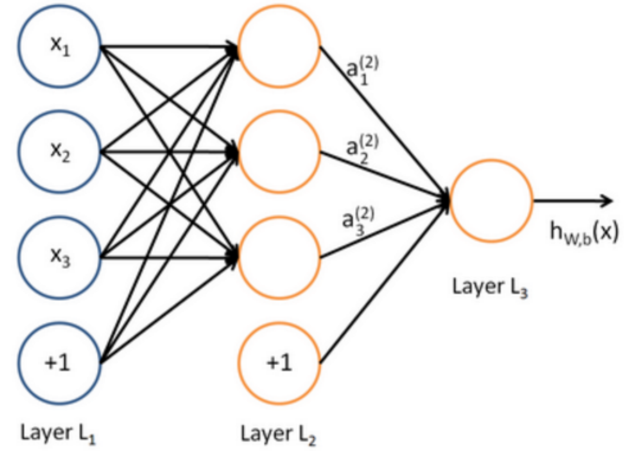
"Gradient Descent"

$$\begin{aligned} J(W,b) &= \left[\frac{1}{m}\sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2 \\ &= \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}(x^{(i)}) - y^{(i)}\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2 \end{aligned}$$

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \boxed{\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b)}$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W,b)$$

2. Backpropagation Algorithm



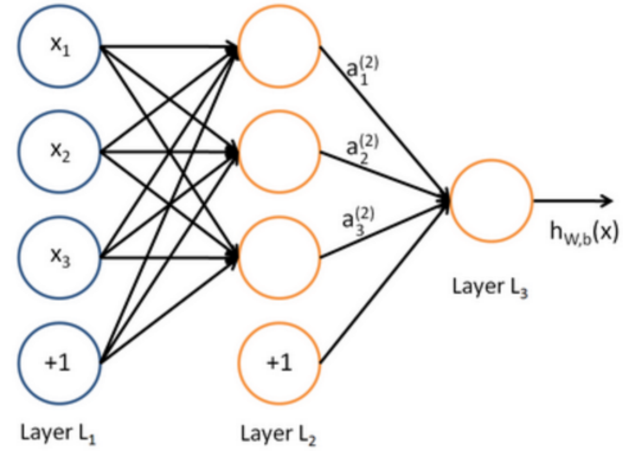
"Gradient Descent"

$$
\begin{aligned}
J(W,b) &= \left[\frac{1}{m}\sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2 \\
&= \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}(x^{(i)}) - y^{(i)}\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b) &= \left[\frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x^{(i)},y^{(i)})\right] + \lambda W_{ij}^{(l)} \\
\frac{\partial}{\partial b_{i}^{(l)}} J(W,b) &= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial b_{i}^{(l)}} J(W,b;x^{(i)},y^{(i)})
\end{aligned}
$$

$\boxed{\text{different i}}$

2. Backpropagation Algorithm



"Derivative chain rule"

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b) = \left[\frac{1}{m} \sum_{i=1}^{m} \boxed{\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x^{(i)},y^{(i)})} \right] + \lambda W_{ij}^{(l)}$$

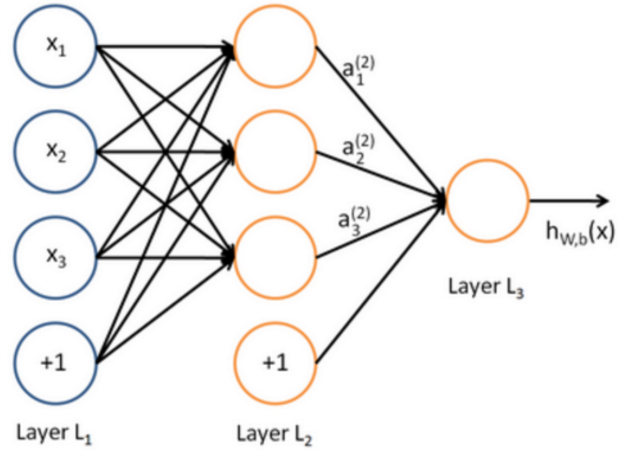
$$\frac{\partial}{\partial b_{i}^{(l)}} J(W,b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial b_{i}^{(l)}} J(W,b;x^{(i)},y^{(i)})$$

$$z_i^{(l+1)} = \sum_{j=1}^{s_l} W_{ij}^{(l)} a_j^{(l)} + b_i^{(l)} \qquad a_i^{(l+1)} = f\left(z_i^{(l+1)}\right)$$

(Page 7)

$$\frac{\partial}{\partial W_{ij}^{(l)}} J\left(W,b;x^{(i)},y^{(i)}\right) = \boxed{\frac{\partial}{\partial z_i^{(l+1)}} J\left(W,b;x^{(i)},y^{(i)}\right)} \times \frac{\partial z_i^{(l+1)}}{\partial W_{ij}^{(l)}}$$

2. Backpropagation Algorithm

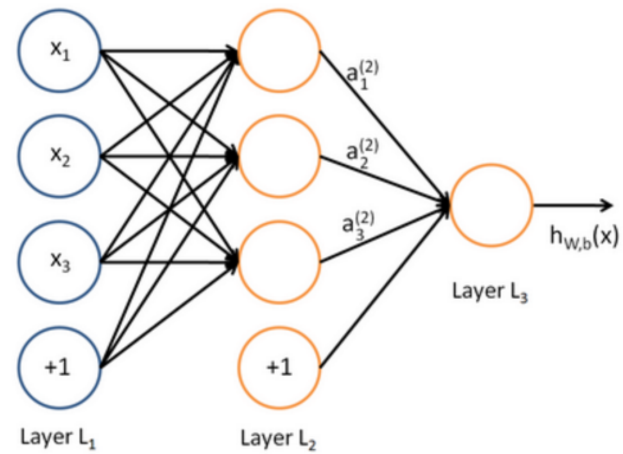


"error term"

$$\delta_i^{(l+1)} = \frac{\partial}{\partial z_i^{(l+1)}} J\left(W, b; x^{(i)}, y^{(i)}\right)$$

which measures how much that node was "responsible" for any errors in output

2. Backpropagation Algorithm



"error term"

$$\delta_i^{(l+1)} = \frac{\partial}{\partial z_i^{(l+1)}} J\left(W, b; x^{(i)}, y^{(i)}\right)$$

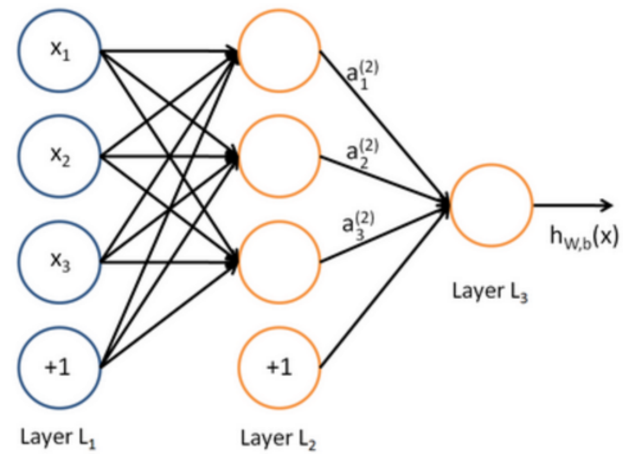
a) For each output unit i in layer n_l

$$\begin{aligned}
\delta_i^{(n_l)} &= \frac{\partial}{\partial z_i^{n_l}} J(W,b;x,y) = \frac{\partial}{\partial z_i^{n_l}} \frac{1}{2} \left\| y - h_{W,b}(x) \right\|^2 \\
&= \frac{\partial}{\partial z_i^{n_l}} \frac{1}{2} \sum_{j=1}^{S_{n_l}} (y_j - a_j^{(n_l)})^2 = \frac{\partial}{\partial z_i^{n_l}} \frac{1}{2} \sum_{j=1}^{S_{n_l}} (y_j - f(z_j^{(n_l)}))^2 \\
&= -(y_i - f(z_i^{(n_l)})) \cdot f'(z_i^{(n_l)}) = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)})
\end{aligned}$$

$$a_i^{(l+1)} = f\left(z_i^{(l+1)}\right) \quad z_i^{(l+1)} = \sum_{j=1}^{s_l} W_{ij}^{(l)} a_j^{(l)} + b_i^{(l)}$$

(Page 7)

2. Backpropagation Algorithm



"error term"

b) For $l = n_l - 1, n_l - 2, n_l - 3, \cdots, 2$

$$\delta_i^{(n_l-1)} = \frac{\partial}{\partial z_i^{n_l-1}} J(W,b;x,y) = \frac{\partial}{\partial z_i^{n_l-1}} \frac{1}{2} \left\| y - h_{W,b}(x) \right\|^2 = \frac{\partial}{\partial z_i^{n_l-1}} \frac{1}{2} \sum_{j=1}^{S_{n_l}} (y_j - a_j^{(n_l)})^2$$

$$= \frac{1}{2} \sum_{j=1}^{S_{n_l}} \frac{\partial}{\partial z_i^{n_l-1}} (y_j - a_j^{(n_l)})^2 = \frac{1}{2} \sum_{j=1}^{S_{n_l}} \frac{\partial}{\partial z_i^{n_l-1}} (y_j - f(z_j^{(n_l)}))^2$$

$$= \sum_{j=1}^{S_{n_l}} -(y_j - f(z_j^{(n_l)})) \cdot \frac{\partial}{\partial z_i^{(n_l-1)}} f(z_j^{(n_l)}) = \sum_{j=1}^{S_{n_l}} -(y_j - f(z_j^{(n_l)})) \cdot f'(z_j^{(n_l)}) \cdot \frac{\partial z_j^{(n_l)}}{\partial z_i^{(n_l-1)}}$$

$$= \sum_{j=1}^{S_{n_l}} \delta_j^{(n_l)} \cdot \frac{\partial z_j^{(n_l)}}{\partial z_i^{n_l-1}} = \sum_{j=1}^{S_{n_l}} \left(\delta_j^{(n_l)} \cdot \frac{\partial}{\partial z_i^{n_l-1}} \sum_{k=1}^{S_{n_l-1}} f(z_k^{n_l-1}) \cdot W_{jk}^{n_l-1} \right)$$

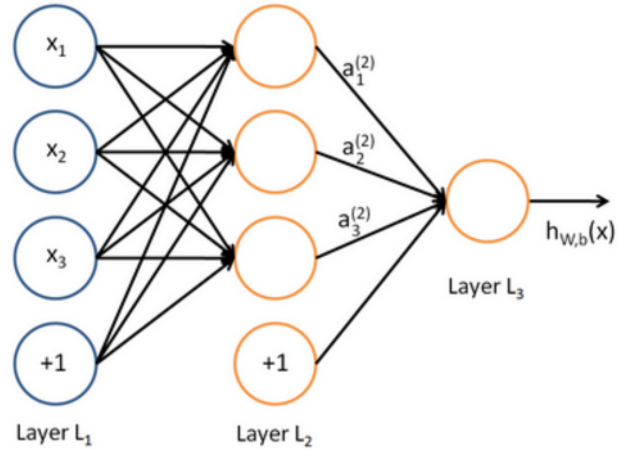
$$= \sum_{j=1}^{S_{n_l}} \delta_j^{(n_l)} \cdot W_{ji}^{n_l-1} \cdot f'(z_i^{n_l-1}) = \left(\sum_{j=1}^{S_{n_l}} W_{ji}^{n_l-1} \delta_j^{(n_l)} \right) f'(z_i^{n_l-1})$$

$$a_i^{(l+1)} = f\left(z_i^{(l+1)}\right)$$

$$z_i^{(l+1)} = \sum_{j=1}^{s_l} W_{ij}^{(l)} a_j^{(l)} + b_i^{(l)}$$

2. Backpropagation Algorithm

"error term"



b) For $l = n_l - 1, n_l - 2, n_l - 3, \cdots, 2$

$$\delta_i^{(n_l-1)} = \left(\sum_{j=1}^{S_{n_l}} W_{ji}^{n_l-1} \delta_j^{(n_l)} \right) f'(z_i^{n_l-1})$$

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)})$$

2. Backpropagation Algorithm

"Gradient Descent"

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \boxed{\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b)}$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W,b)$$

"Gradient Descent"

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b) = \left[\frac{1}{m} \sum_{i=1}^{m} \boxed{\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x^{(i)},y^{(i)})} \right] + \lambda W_{ij}^{(l)}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W,b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial b_i^{(l)}} J(W,b;x^{(i)},y^{(i)})$$

"Derivative chain rule"

$$\frac{\partial}{\partial W_{ij}^{(l)}} J\left(W,b;x^{(i)},y^{(i)}\right) = \boxed{\frac{\partial}{\partial z_i^{(l+1)}} J\left(W,b;x^{(i)},y^{(i)}\right)} \times \frac{\partial z_i^{(l+1)}}{\partial W_{ij}^{(l)}}$$

$$z_i^{(l+1)} = \sum_{j=1}^{s_l} W_{ij}^{(l)} a_j^{(l)} + b_i^{(l)}$$

"error term"

$$\boxed{\delta_i^{(l+1)}} = \frac{\partial}{\partial z_i^{(l+1)}} J\left(W,b;x^{(i)},y^{(i)}\right)$$

2. Backpropagation Algorithm

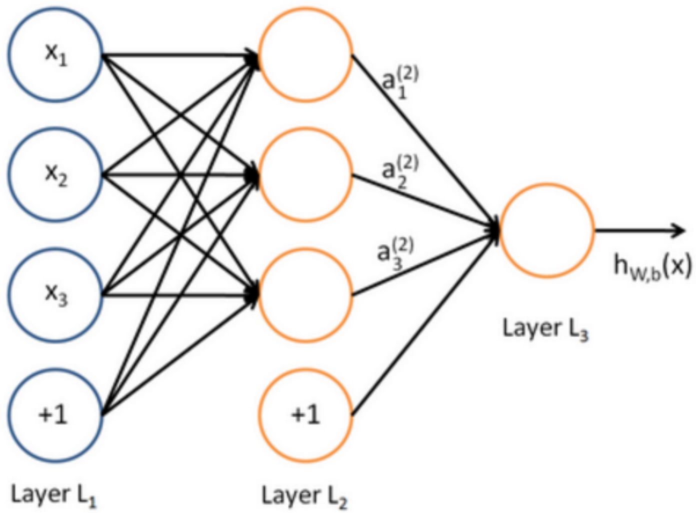
$$\boxed{\begin{aligned}\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x,y) &= a_j^{(l)} \delta_i^{(l+1)} \\ \frac{\partial}{\partial b_i^{(l)}} J(W,b;x,y) &= \delta_i^{(l+1)}.\end{aligned}}$$

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \; \frac{1}{2} \left\| y - h_{W,b}(x) \right\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)})$$

"error term"

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)})$$

2. Backpropagation Algorithm



$$\boxed{\begin{aligned}\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x,y) &= a_j^{(l)} \delta_i^{(l+1)} \\ \frac{\partial}{\partial b_i^{(l)}} J(W,b;x,y) &= \delta_i^{(l+1)}.\end{aligned}}$$

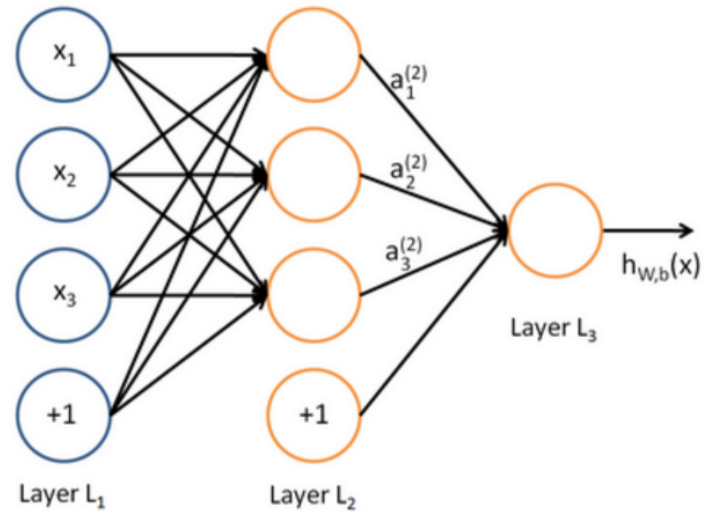
$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \; \frac{1}{2} \|y - h_{W,b}(x)\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)})$$

"error term"

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)})$$

2. Backpropagation Algorithm

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x,y) = a_j^{(l)} \delta_i^{(l+1)}$$
$$\frac{\partial}{\partial b_i^{(l)}} J(W,b;x,y) = \delta_i^{(l+1)}.$$



“error term”

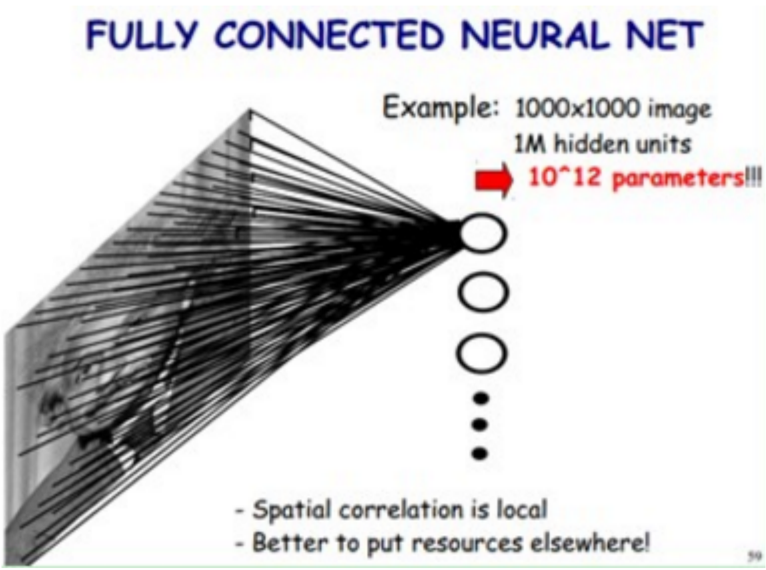
$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \|y - h_{W,b}(x)\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)})$$

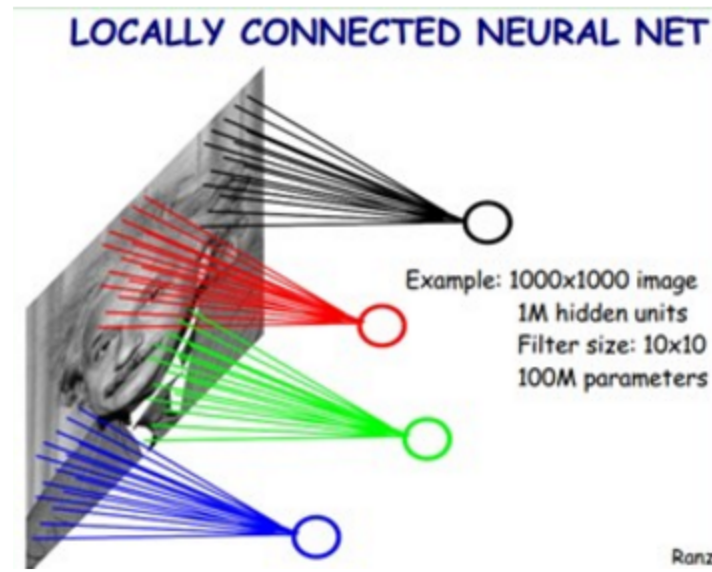
$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)})$$

结论：一个点（神经元）的残差=前向的时候下一层中使用过该点的神经单元的残差按权重求和

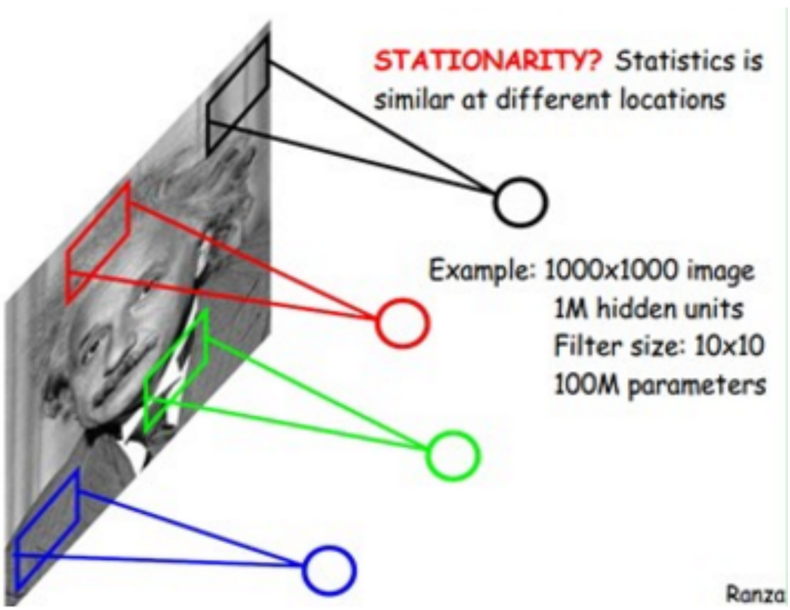
Content

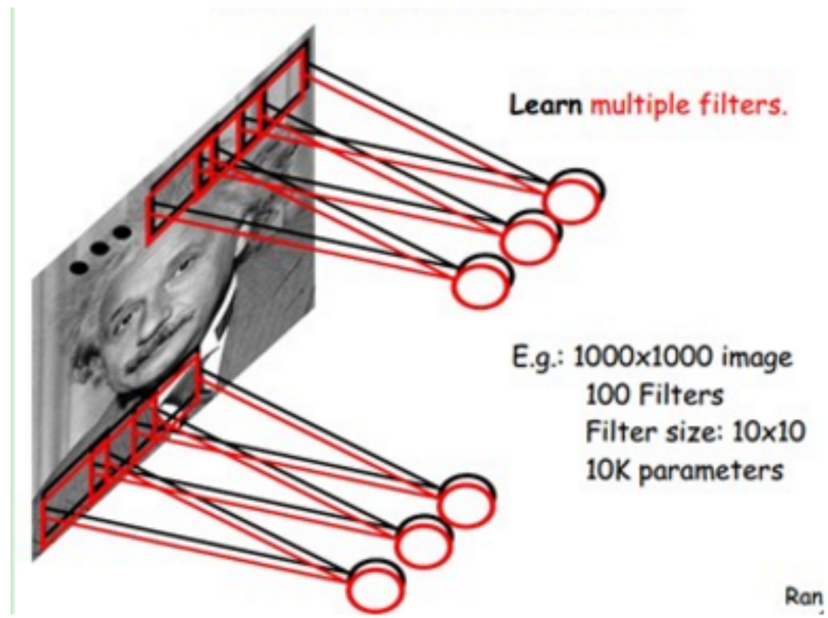
3. Convolutional Neural Network





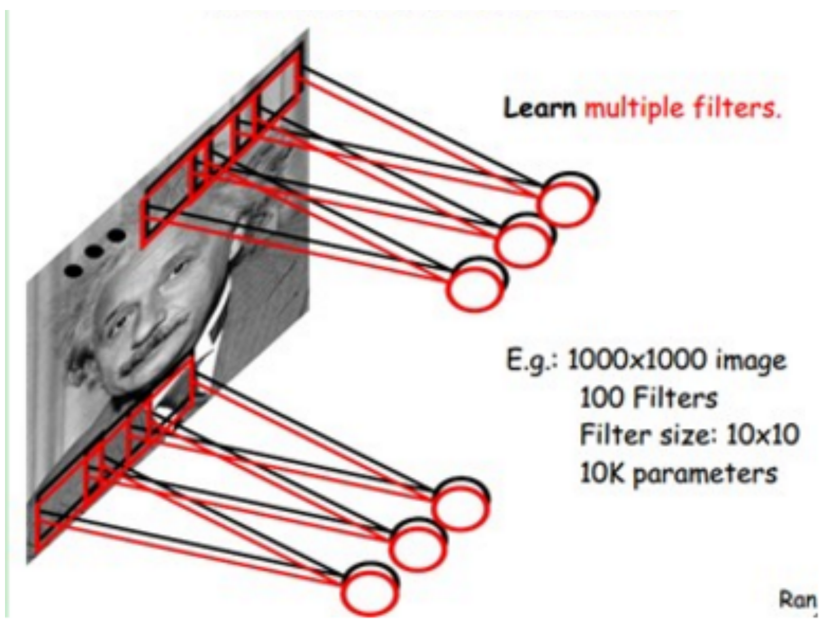
3. Convolutional Neural Network





3. Convolutional Neural Network

convolution



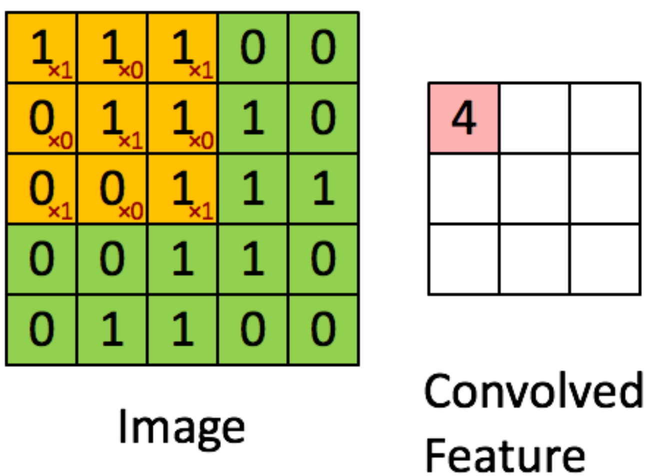
How to calculate the number of hidden units?
1,000,000 hidden units?

3. Convolutional Neural Network

convolution layer

stride, pad, kernel(size), number(output)

$$H_{l+1} = \frac{(H_l + 2 \times pad\_h - kernel\_h)}{stride\_h} + 1 \qquad W_{l+1} = \frac{(W_l + 2 \times pad\_w - kernel\_w)}{stride\_w} + 1$$



$$H_l = W_l = 5$$

$$kernel\_h = kernel\_w = 3$$

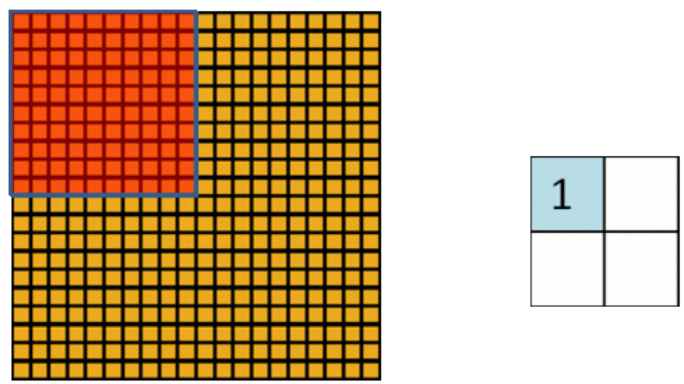
$$pad = 0$$

$$stride = 1$$

3. Convolutional Neural Network

pooling layer

Max Pooling & Avg Pooling

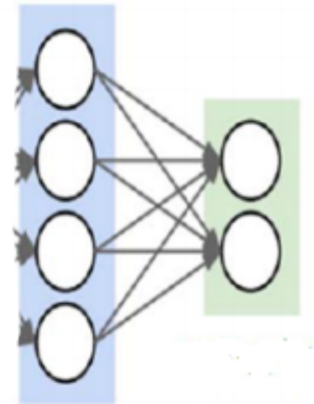


Convolved feature

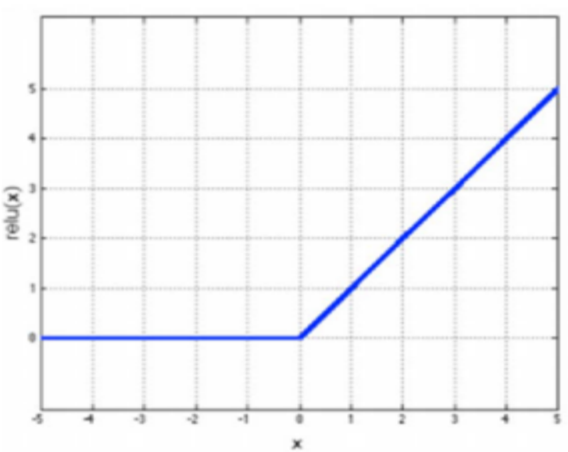
Pooled feature

3. Convolutional Neural Network

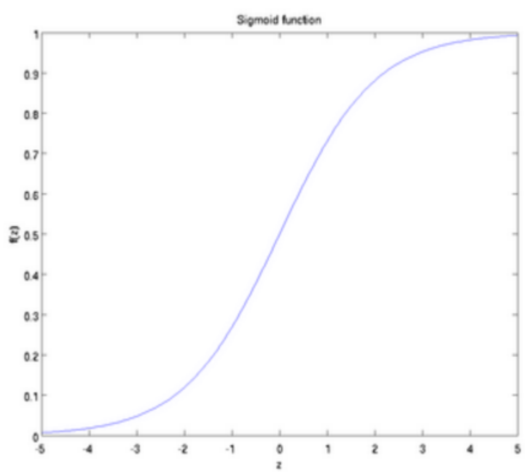
fully connected layer



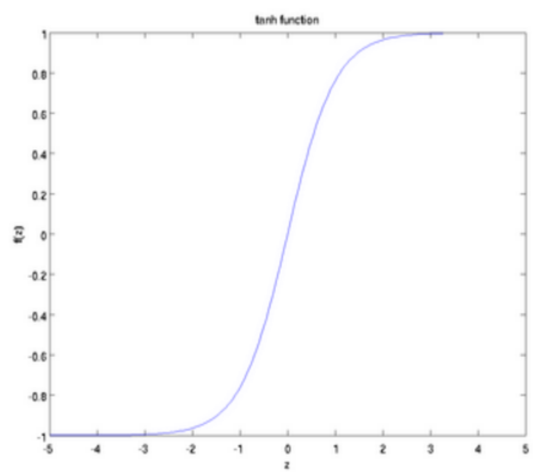
activation function



ReLU



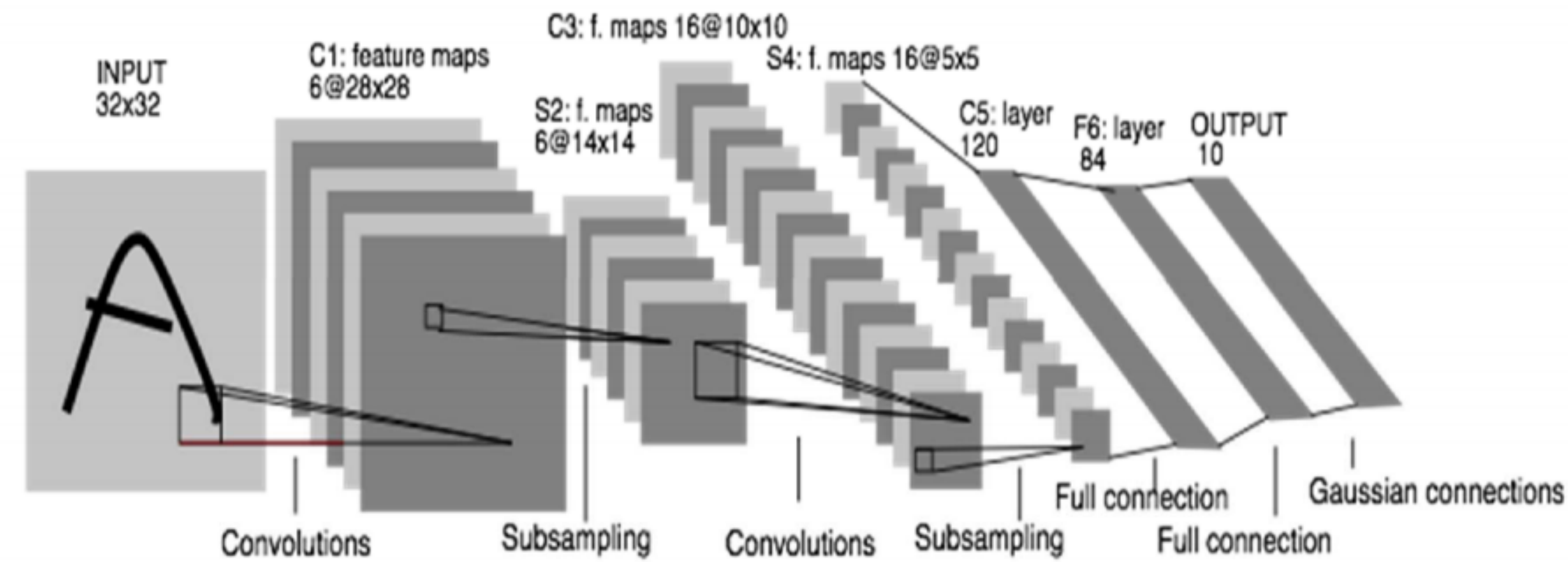
Sigmoid



tanh

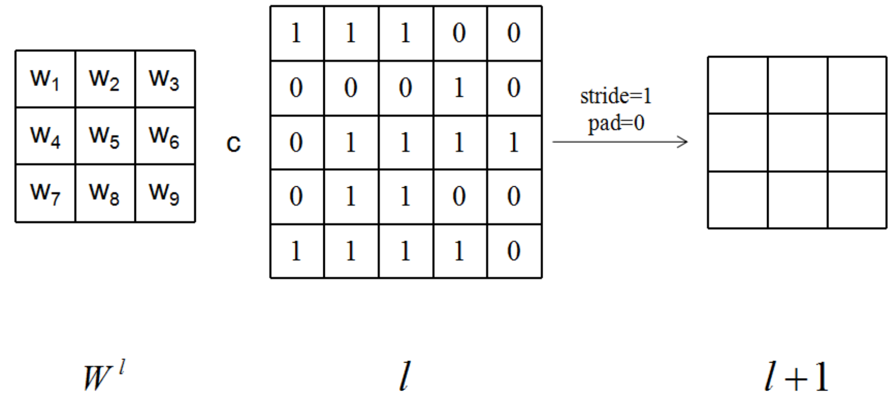
3. Convolutional Neural Network

MNIST



3. Convolutional Neural Network

convolution layer



a. compute the input size

b. setup stride, pad, kernel(size), number(output), λ, α

c. compute the size of filter and output

d. *initialize weights (the first iteration)

e. convolution

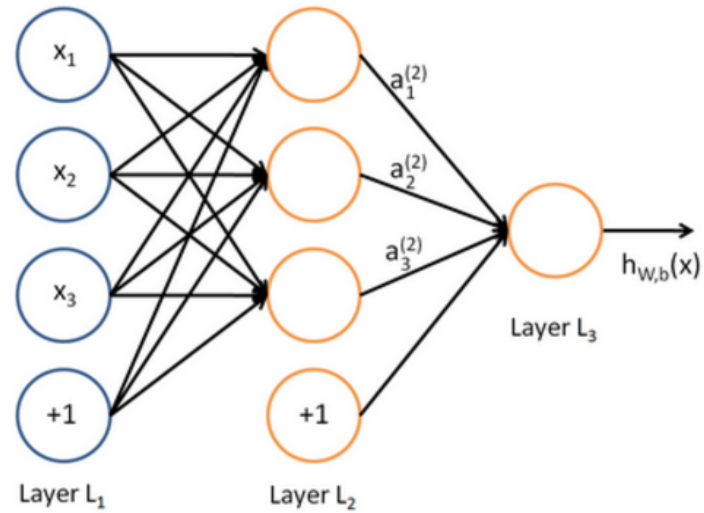
Content

4. Backpropagation on CNN

- Backpropagation on pooling layer
- Backpropagation on convolution layer

4. Backpropagation on CNN

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x,y) = a_j^{(l)} \delta_i^{(l+1)}$$
$$\frac{\partial}{\partial b_i^{(l)}} J(W,b;x,y) = \delta_i^{(l+1)}.$$



"error term"

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \|y - h_{W,b}(x)\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)})$$

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)})$$

结论：一个点（神经元）的残差=前向的时候下一层中使用过该点的神经单元的残差按权重求和

4. Backpropagation on CNN

Backpropagation on pooling layer

Forward

1	2	3	4
5	6	7	8
9	2	3	5
6	5	1	1

max pooling

kernel_size=2
stride=2
pad=0

6	8
9	5

Forward

0	0	0	0
0	1	0	2
3	0	0	4
0	0	0	0

$\delta^{(l+1)} \rightarrow \delta^{(l)}$

1	2
3	4

4. Backpropagation on CNN

Backpropagation on pooling layer

Forward

1	2	3	4
5	6	7	8
9	2	3	5
6	5	1	1

avg pooling

kernel_size=2
stride=2
pad=0

7	11
11	5

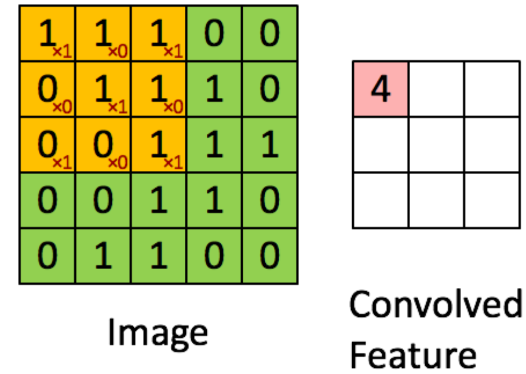
Backward

0.25	0.25	0.5	0.5
0.25	0.25	0.5	0.5
0.75	0.75	1	1
0.75	0.75	1	1

$\delta^{(l+1)} \to \delta^{(l)}$

1	2
3	4

4. Backpropagation on CNN



Backpropagation on convolution layer

$$\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} \quad c \quad \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} \xrightarrow[\text{pad}=0]{\text{stride}=1} \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$$

W^l $\qquad$ l $\qquad$ $l+1$

4. Backpropagation on CNN

Backpropagation on convolution layer

$$W_{l+1} = \frac{(W_l + 2 \times pad - \mathrm{ker}\,nel)}{stride} + 1$$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

W^l

c

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	2	3	0	0
0	0	4	5	6	0	0
0	0	7	8	9	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$\delta^{l+1}(+\,pad)$

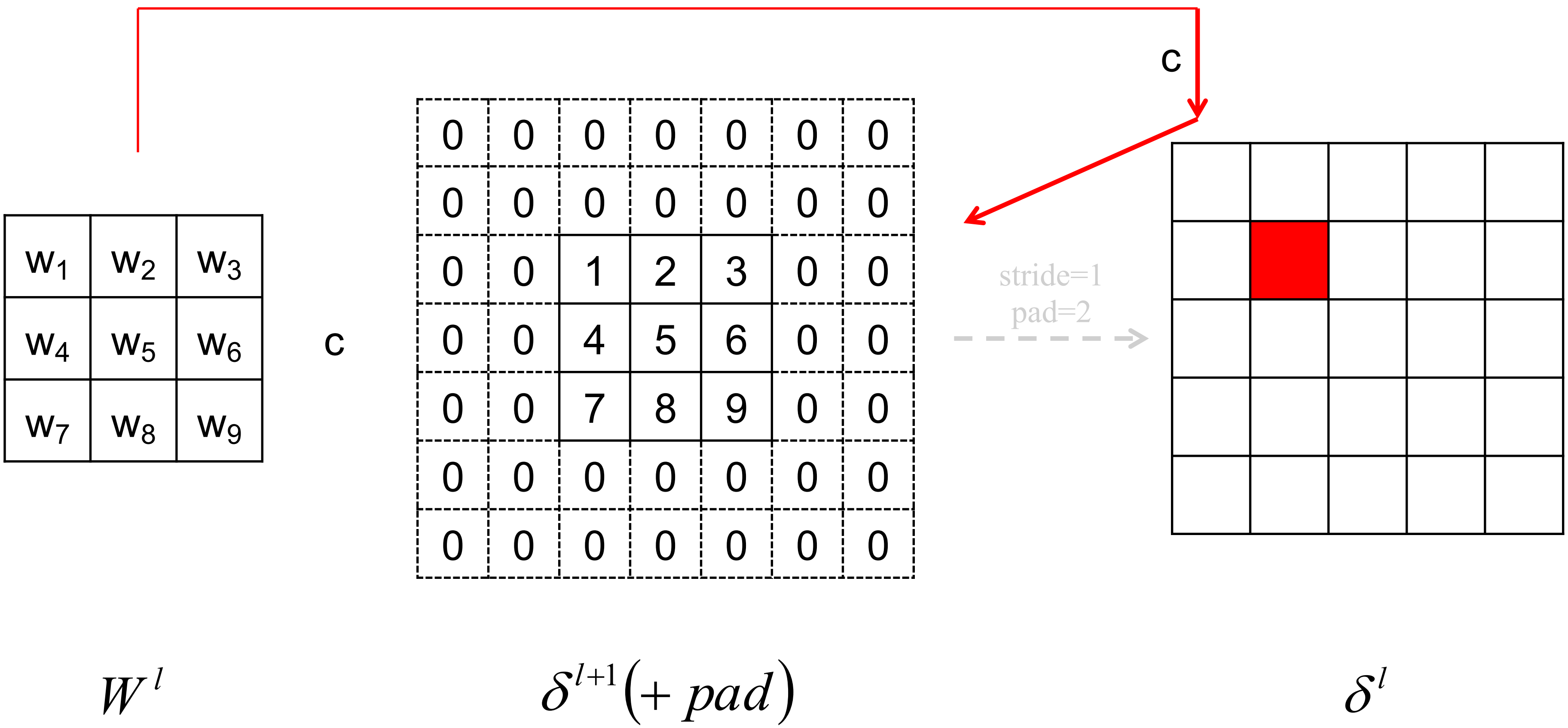
$\xrightarrow[\text{pad=?}]{\text{stride=1}}$

δ^l

4. Backpropagation on CNN

$$W_{l+1} = \frac{(W_l + 2 \times pad - kernel)}{stride} + 1$$

Backpropagation on convolution layer

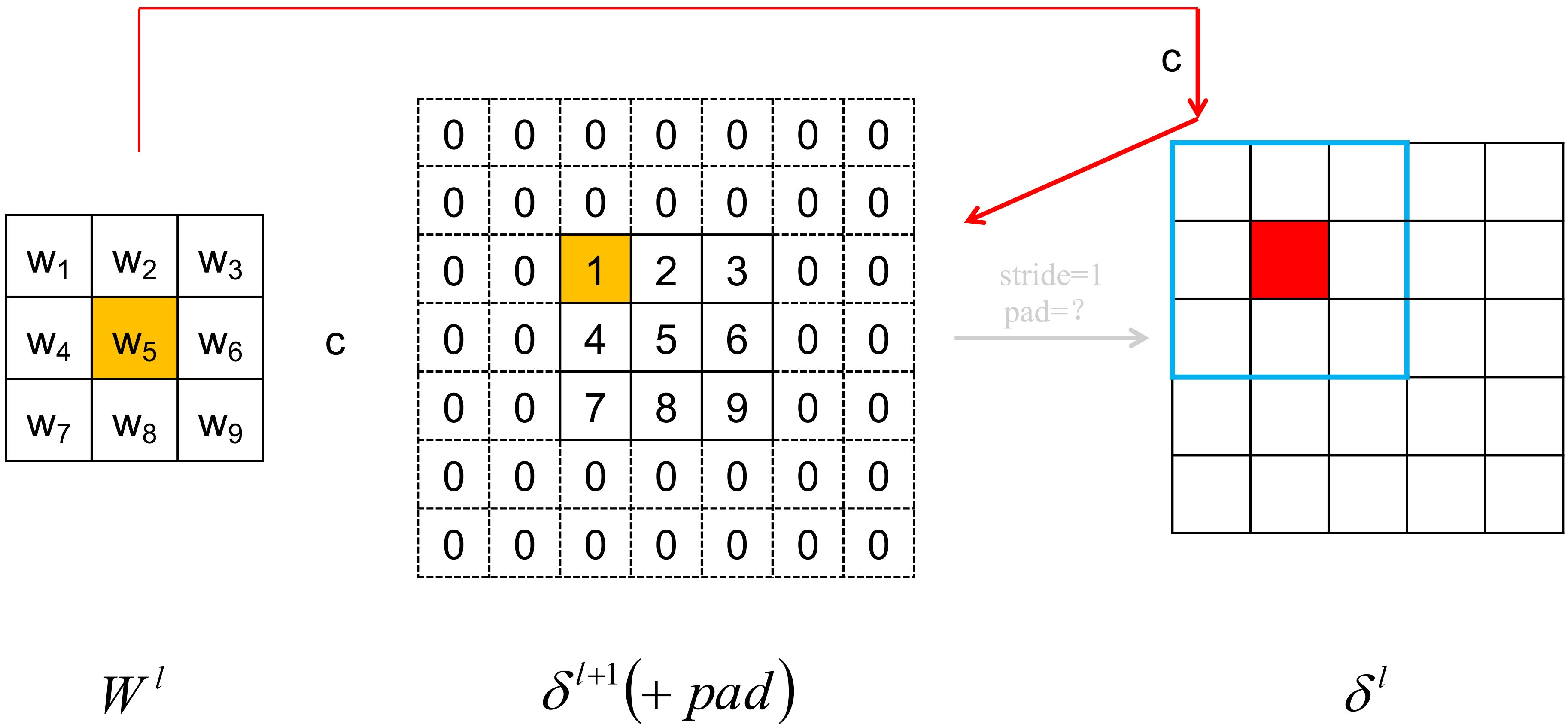


结论：一个点（神经元）的残差=前向的时候下一层中使用过该点的神经单元的残差按权重求和

4. Backpropagation on CNN

$$W_{l+1} = \frac{(W_l + 2\times pad - \ker nel)}{stride} + 1$$

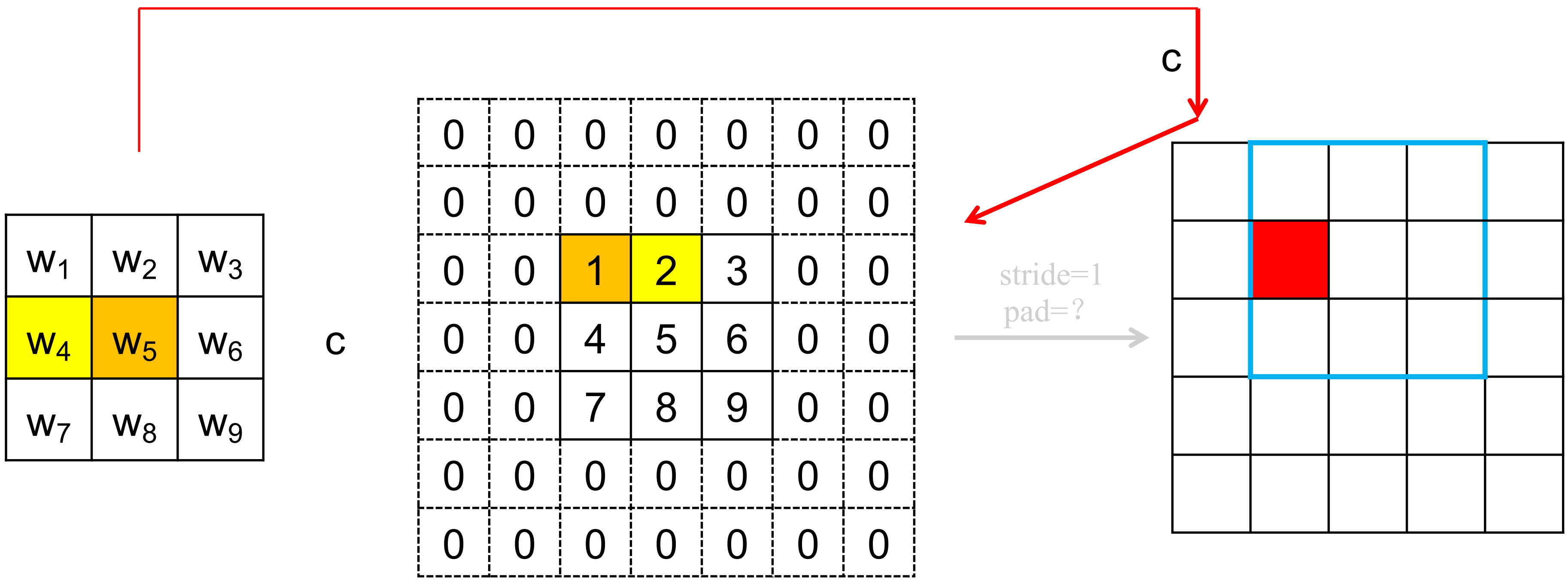
Backpropagation on convolution layer



4. Backpropagation on CNN

$$W_{l+1} = \frac{(W_l + 2 \times pad - \text{ker}\,nel)}{stride} + 1$$

Backpropagation on convolution layer

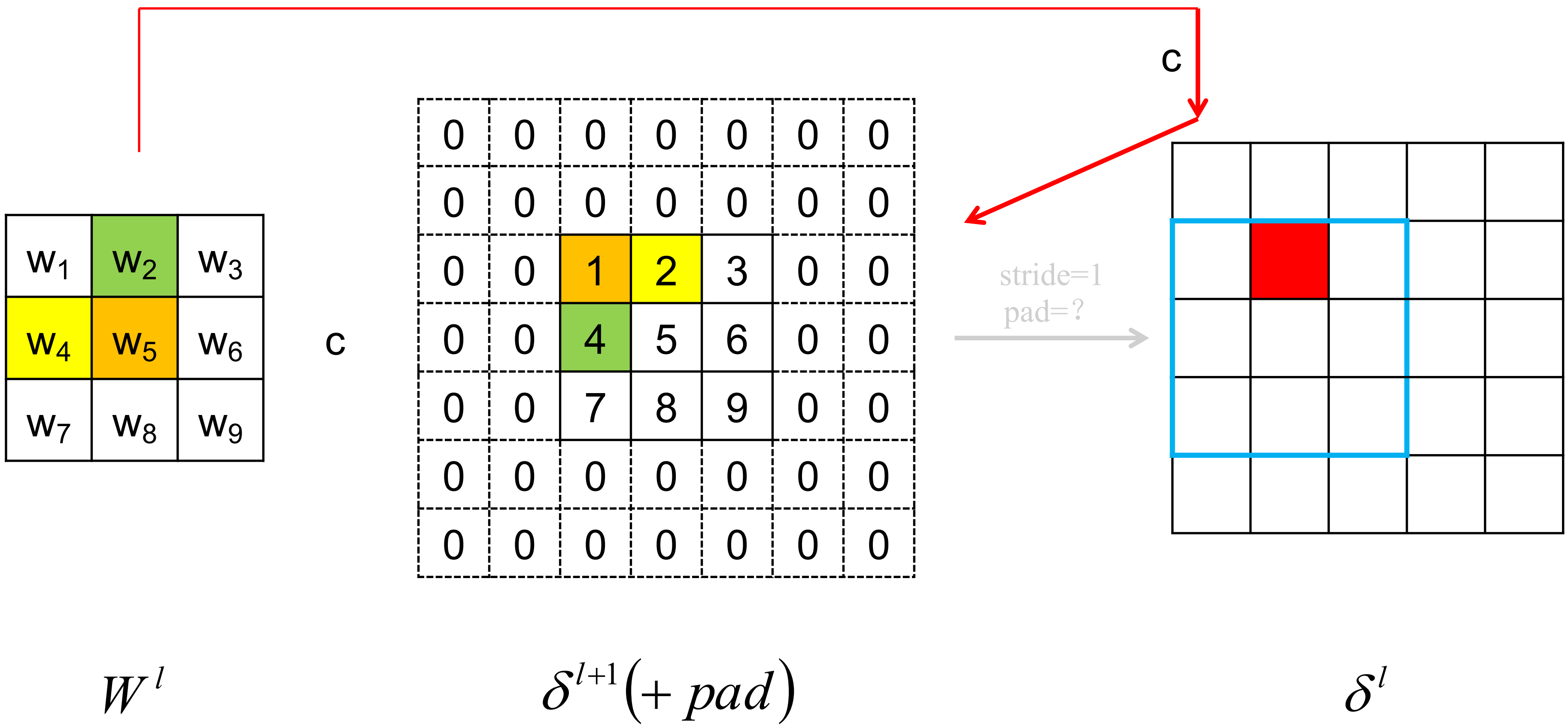


W^l $\delta^{l+1}(+pad)$ δ^l

4. Backpropagation on CNN

$$W_{l+1} = \frac{(W_l + 2\times pad - \mathrm{ker}\,nel)}{stride} + 1$$

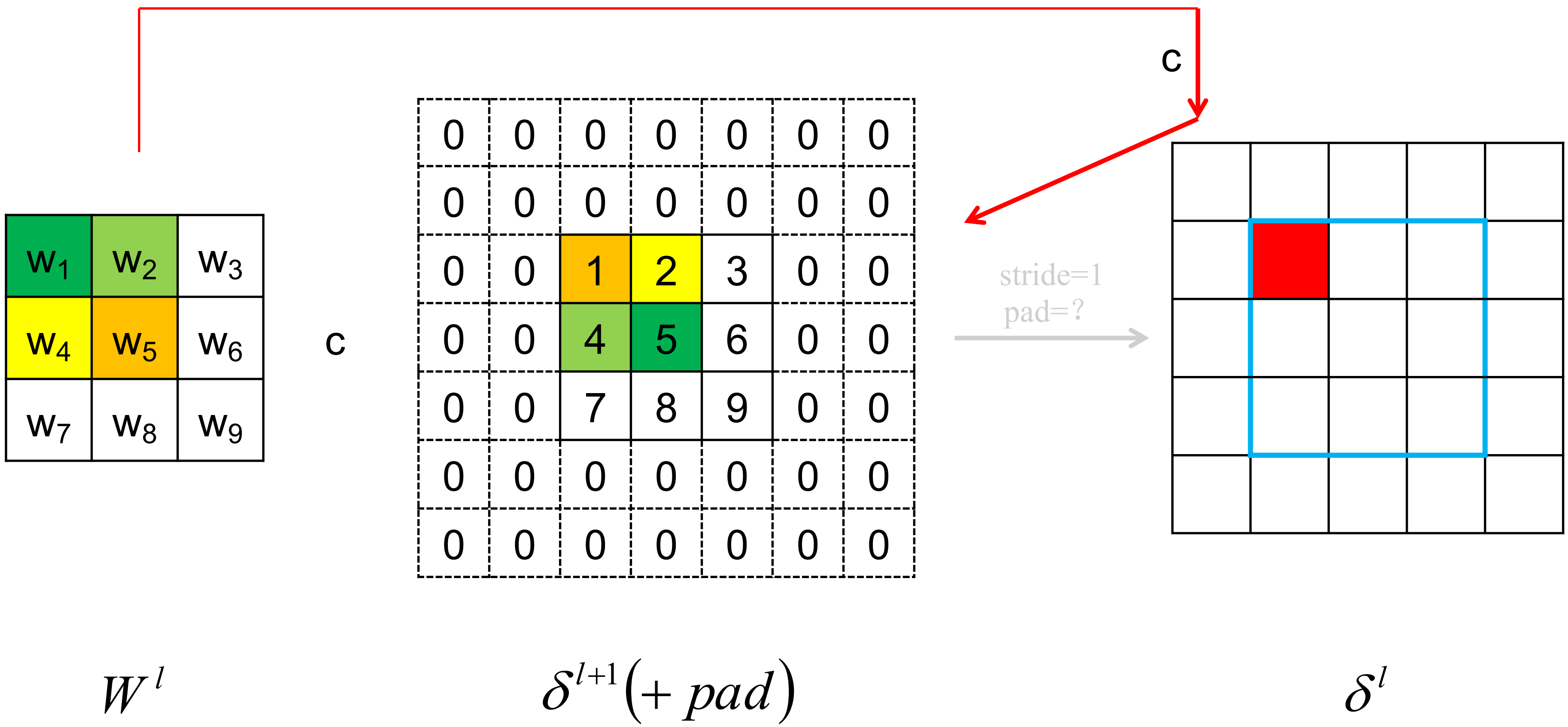
Backpropagation on convolution layer



4. Backpropagation on CNN

$$W_{l+1} = \frac{(W_l + 2\times pad - kernel)}{stride} + 1$$

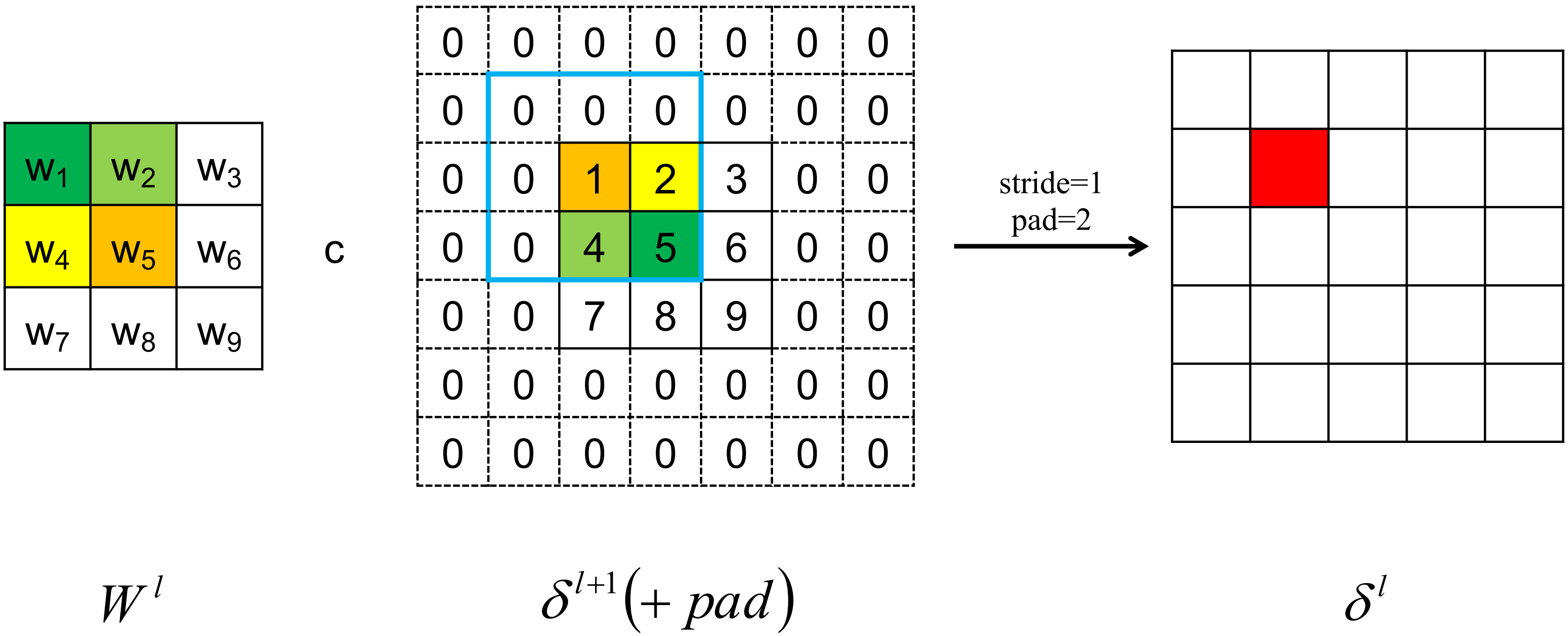
Backpropagation on convolution layer



4. Backpropagation on CNN

$$W_{l+1} = \frac{(W_l + 2 \times pad - \mathrm{ker}\,nel)}{stride} + 1$$

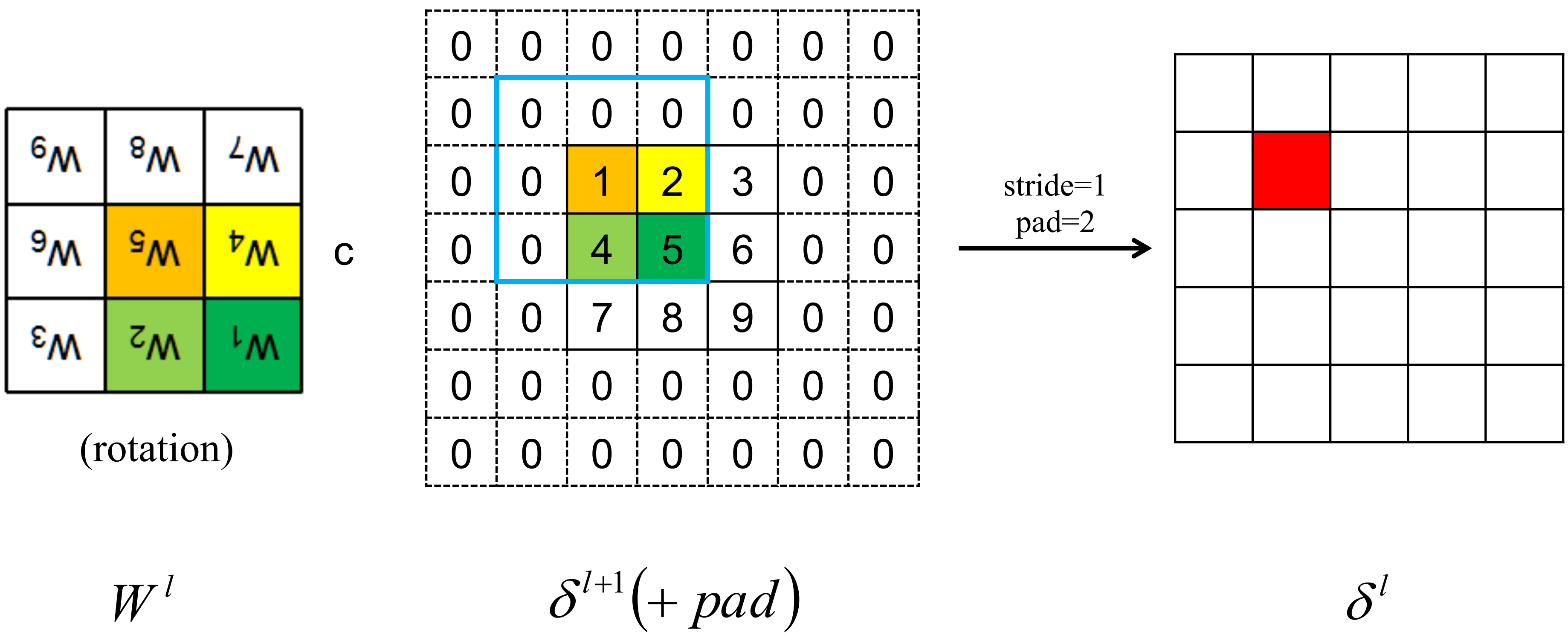
Backpropagation on convolution layer



4. Backpropagation on CNN

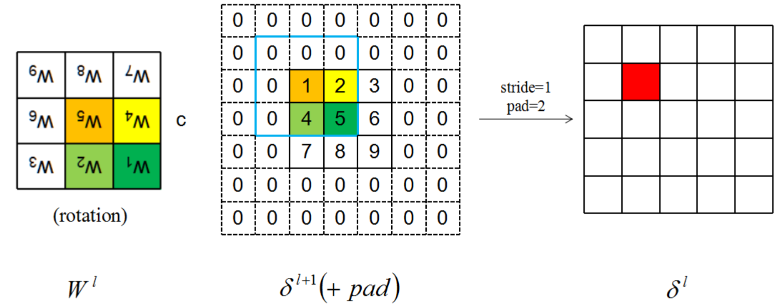
$$W_{l+1} = \frac{(W_l + 2 \times pad - \ker nel)}{stride} + 1$$

Backpropagation on convolution layer



4. Backpropagation on CNN

Backpropagation on convolution layer



a. $\delta^{l+1} + pad$

b. W^l rotate $180°$

c. compute δ^l(convolution)

d. compute gradient $\frac{\partial}{\partial W} J$

e. update weights

Reference

1. UFLDL: http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial
2. Blogs:
   http://www.cnblogs.com/tornadomeet/tag/Deep%20Learning/
   http://blog.csdn.net/zouxy09/article/category/1387932

THANK YOU