Tutorials: part I

Fun with Neural Network

程嵩

2016/11

cs:

Network structure

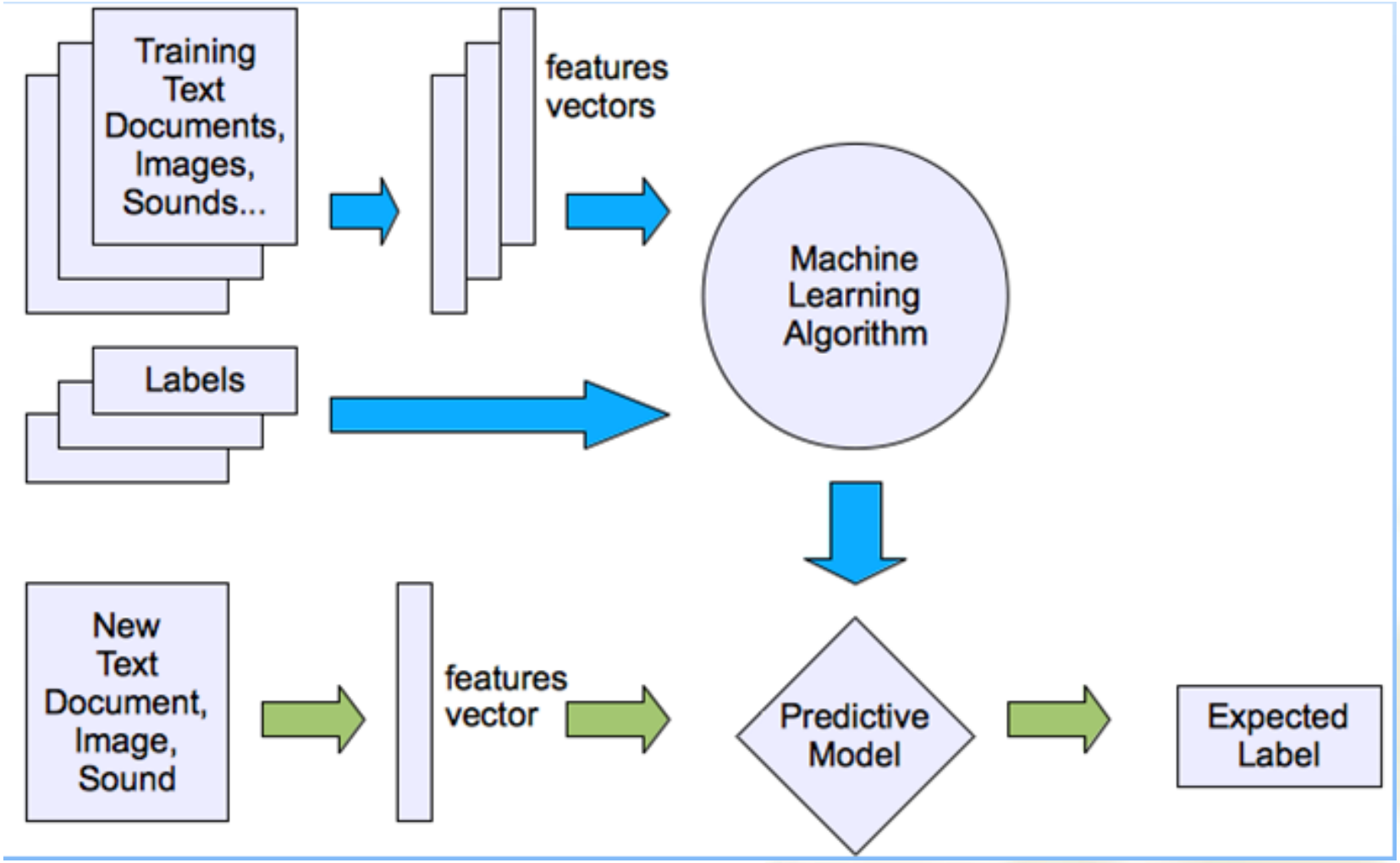
Back propagation algorithm

honghao:

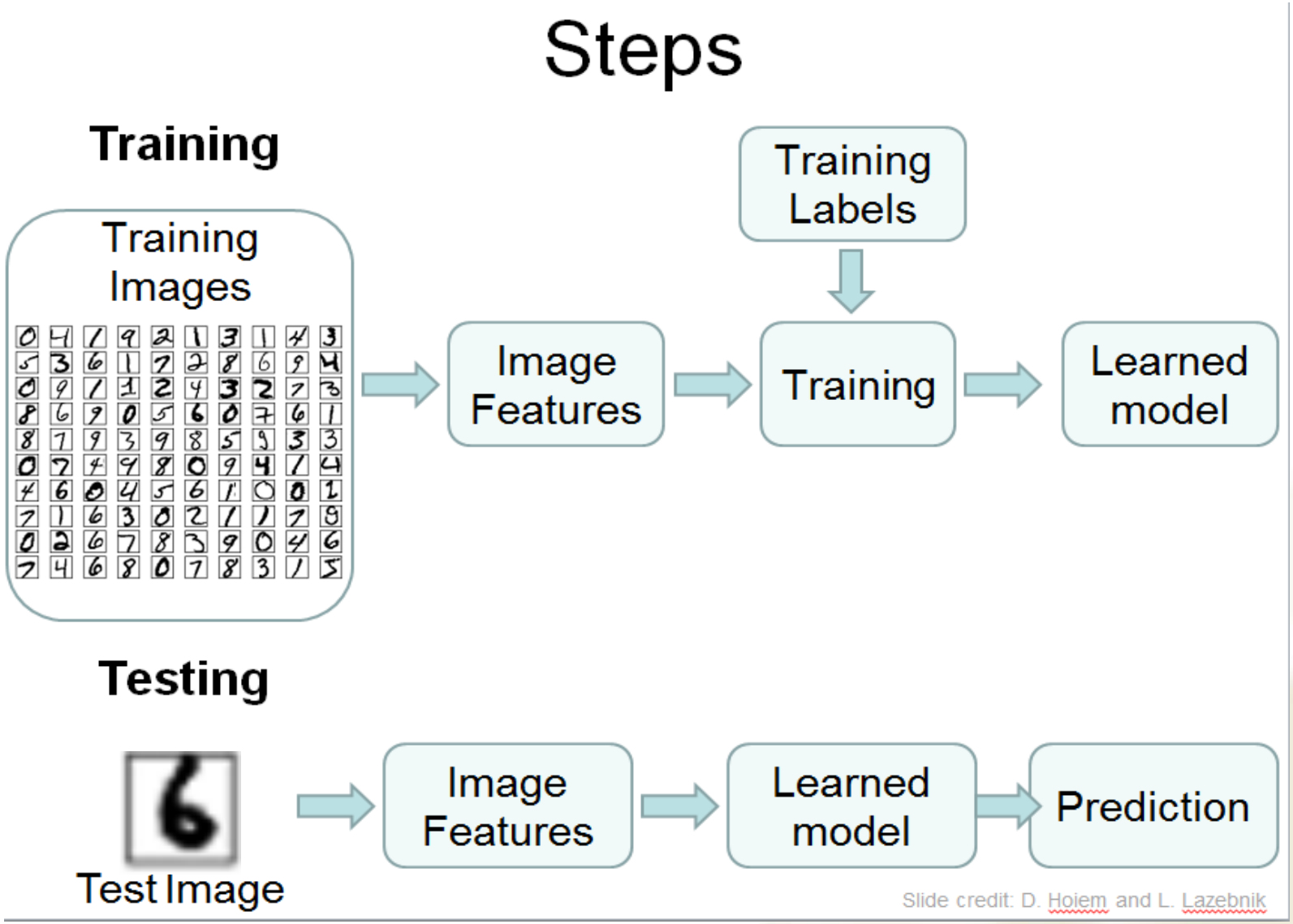
Tensorflow

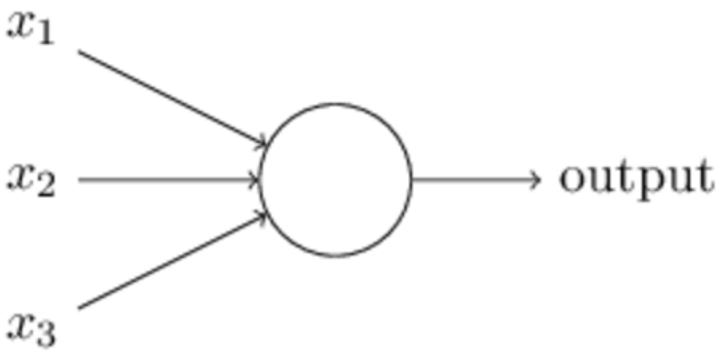
Tensorboard

Supervised learning

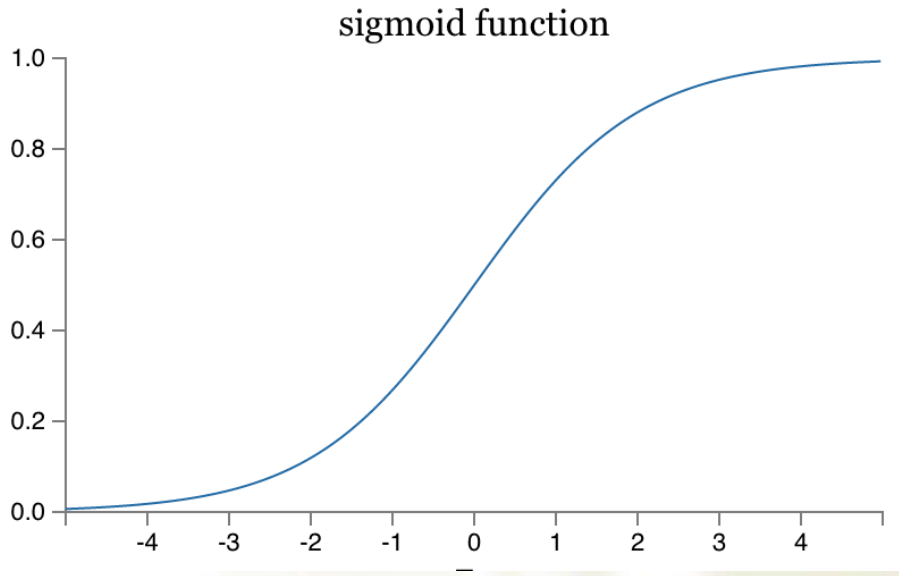


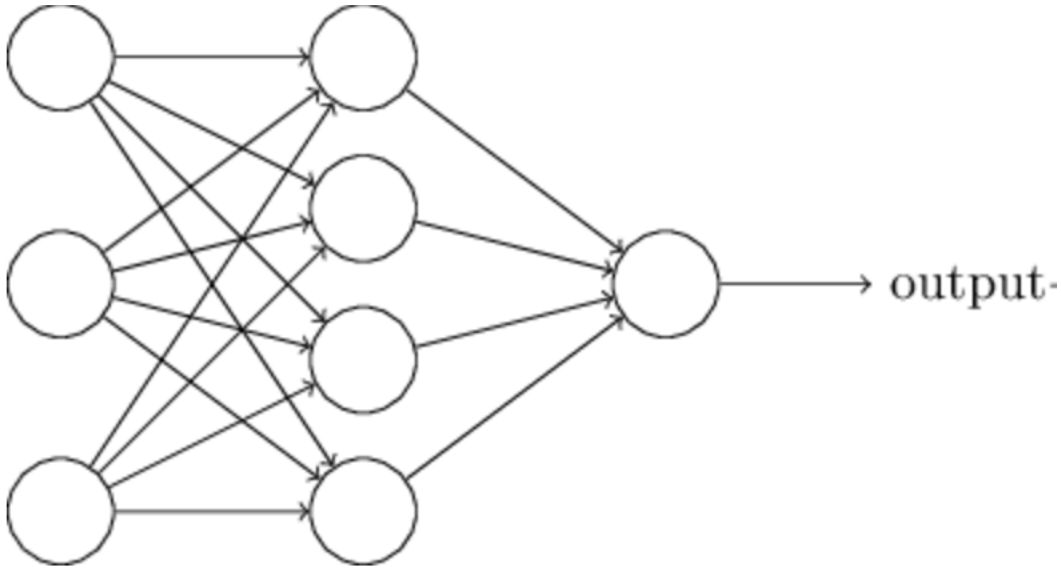
Steps


Training
Training Images
Training Labels
Image Features
Training
Learned model
Testing
Test Image
Image Features
Learned model
Prediction
Slide credit: D. Hoiem and L. Lazebnik



output = $\dfrac{1}{1 + \exp(-\sum_j w_j x_j - b)}$



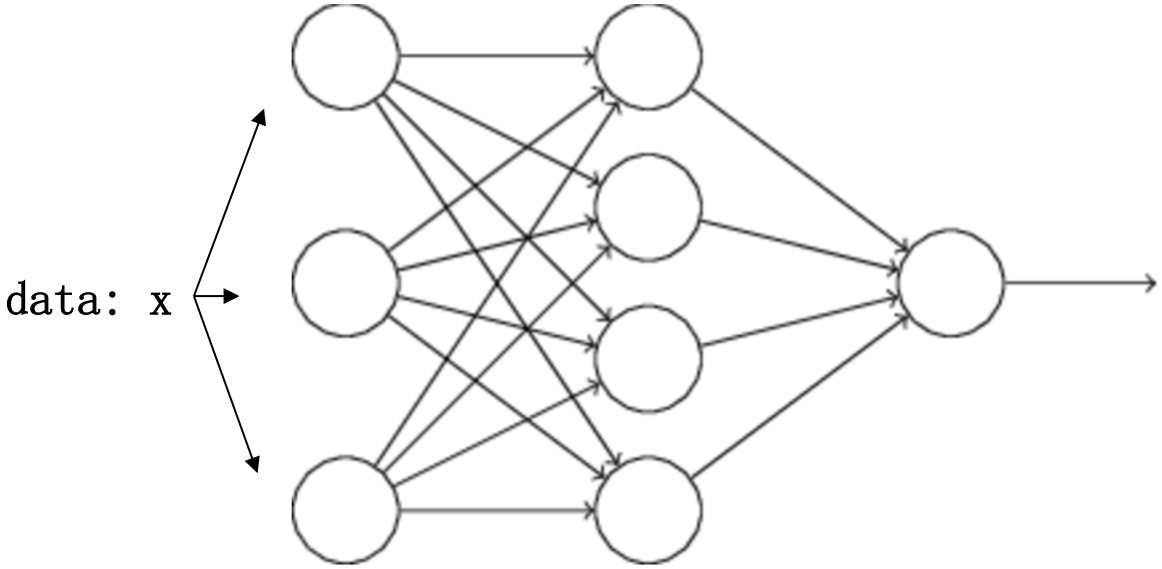


$z = w\,x + b$

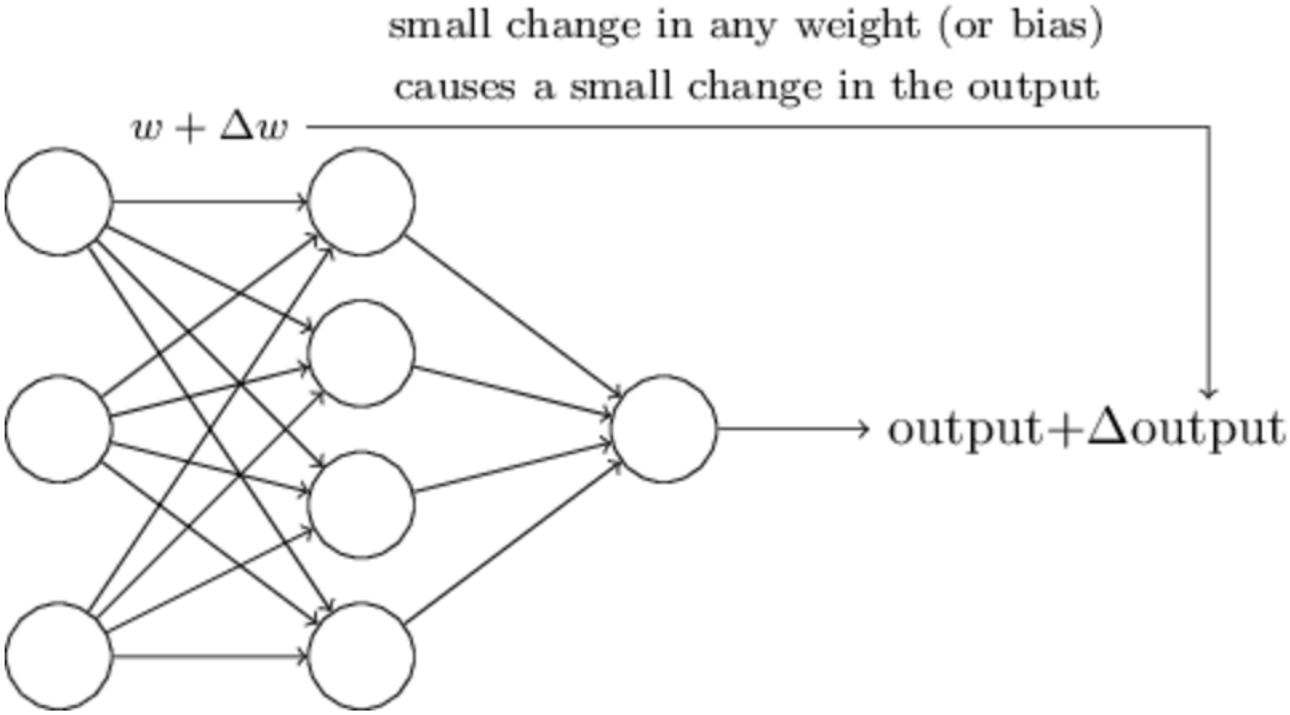
$a = \sigma(z) = 1/1 + e\uparrow z$

$[\blacksquare z\downarrow 1\ @z\downarrow 2\ @z\downarrow 3\ @z\downarrow 4\ ] = [\blacksquare w\downarrow 11\ \&w\downarrow 12\ \&w\downarrow 13\ @w\downarrow 21\ \&w\downarrow 22\ \&w\downarrow 23\ @w$

data_label: y(x)



Cost function: $C = \frac{1}{2n} \sum_x \|y(x) - a^L(x)\|^2$



$$\Delta \text{output} \approx \sum_j \frac{\partial\, \text{output}}{\partial w_j} \Delta w_j + \frac{\partial\, \text{output}}{\partial b} \Delta b$$

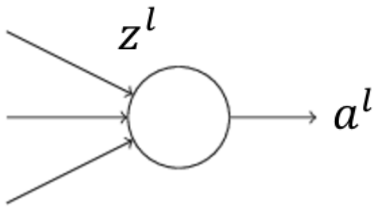
Optimization

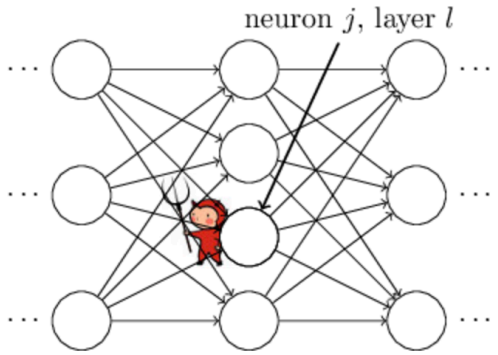
Backpropagation algorithm

Def: $z^l = w^l a^{l-1} + b^l$

Def: $a^l = \sigma(z^l)$

Def: $\delta_j^l = \partial C / \partial Z_j^l$





Def: hadamard product $\odot$: $a=[1,2]$, $b=[3,4]$, $a \odot b = [1*3, 2*4] = [3, 8]$

sum j

$\delta_j^L = \partial C / \partial Z_j^L = \partial C / \partial a_j \; \sigma'(z_j^L)$ $\qquad$ $\delta^L = \nabla_a C \odot \sigma'(z^L)$

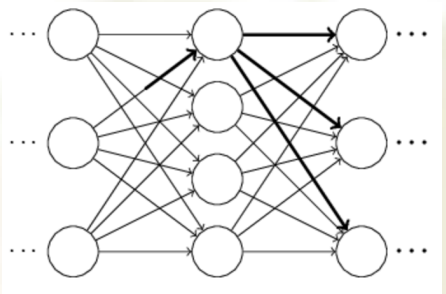
$\delta_j^l = (\sum_i w_{ji}^{l+1} \delta_i^{l+1}) \sigma'(z^l)$

sum j

$\rightarrow \; \delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$$



Optimization

Backpropagation algorithm

1. **Input x:** Set the corresponding activation a^1 for the input layer.

2. **Feedforward:** For each $l = 2, 3, \ldots, L$ compute $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$.

3. **Output error δ^L:** Compute the vector $\delta^L = \nabla_a C \odot \sigma'(z^L)$.

4. **Backpropagate the error:** For each $l = L-1, L-2, \ldots, 2$ compute $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$.

5. **Output:** The gradient of the cost function is given by $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$ and $\frac{\partial C}{\partial b^l_j} = \delta^l_j$.

Stochastic gradient descent

1. **Input a set of training examples**

2. **For each training example** x**:** Set the corresponding input activation $a^{x,1}$, and perform the following steps:

   - **Feedforward:** For each $l = 2, 3, \ldots, L$ compute $z^{x,l} = w^l a^{x,l-1} + b^l$ and $a^{x,l} = \sigma(z^{x,l})$.
   - **Output error** $\delta^{x,L}$**:** Compute the vector $\delta^{x,L} = \nabla_a C_x \odot \sigma'(z^{x,L})$.
   - **Backpropagate the error:** For each $l = L-1, L-2, \ldots, 2$ compute $\delta^{x,l} = ((w^{l+1})^T \delta^{x,l+1}) \odot \sigma'(z^{x,l})$.

3. **Gradient descent:** For each $l = L, L-1, \ldots, 2$ update the weights according to the rule $w^l \to w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T$, and the biases according to the rule $b^l \to b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$.

Andrew Ng 百度首席科学家
https://www.coursera.org/learn/machine-learning

Hinton 是deep learning的大牛
https://www.coursera.org/course/neuralnets

比较详尽的概览
https://www.udacity.com/course/machine-learning--ud262

极好的网络教程(Michael Nielsen)
http://neuralnetworksanddeeplearning.com/index.html

Deep learning 专著
http://www.deeplearningbook.org/

MNIST DEMO:
http://myselph.de/neuralNet.html

Thanks!