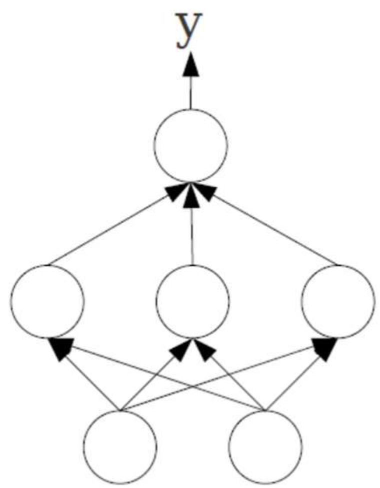
Training of Multi-Layer NN: Back-Propagation

How to Train Multi-Layer NN



Define sum-squared error:

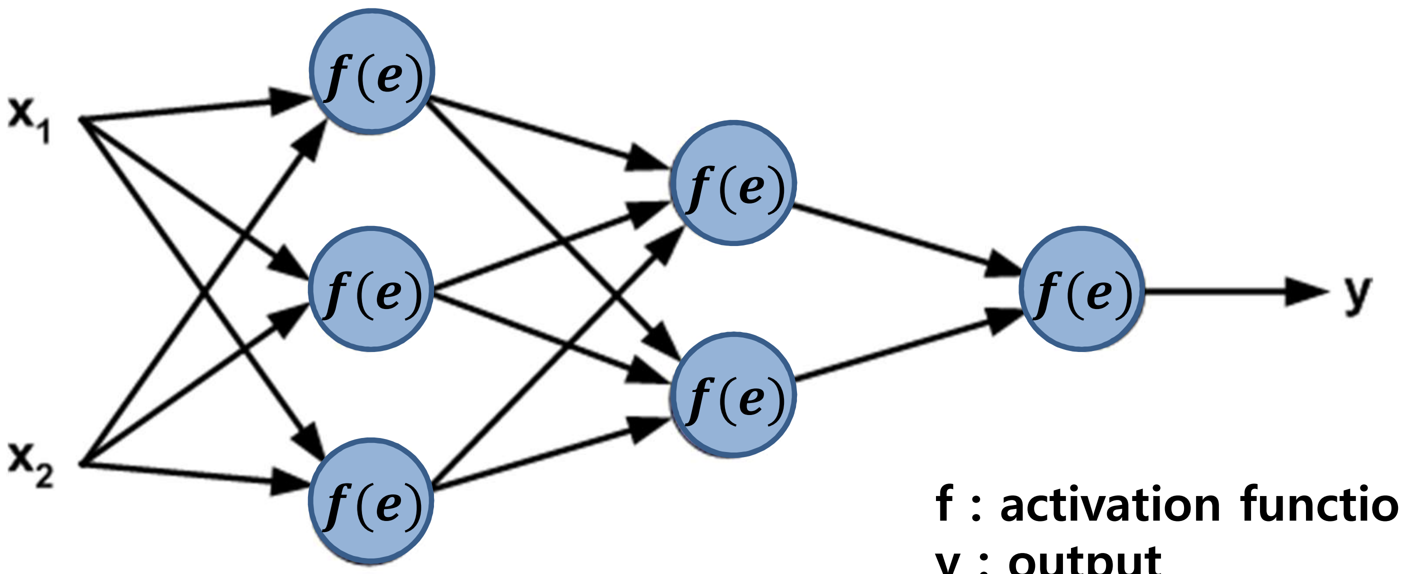
$$E = \frac{1}{2} \sum_{p} (d^p - y^p)^2$$

Use gradient descent error minimization:

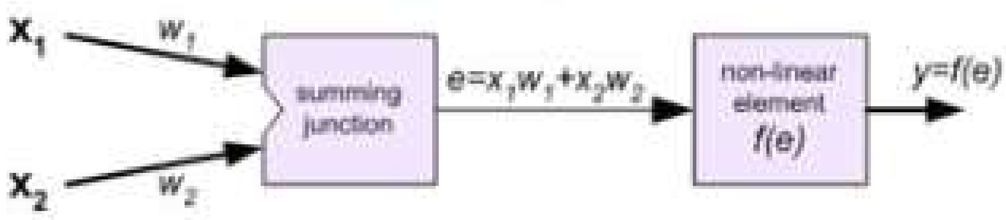
$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$

Works if the nonlinear transfer function is differentiable.

Multilayer Neural Network



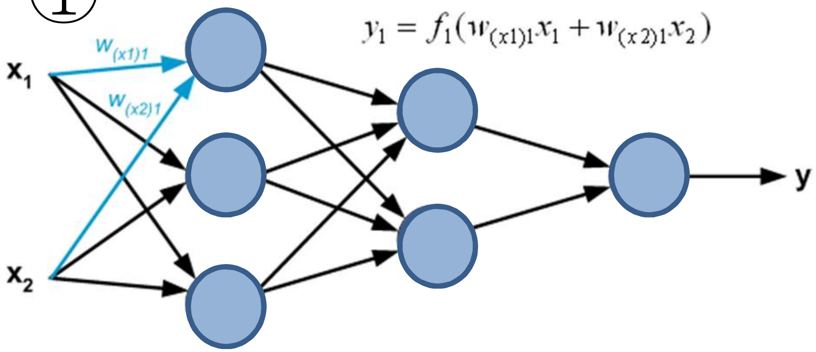
f : activation function
y : output
x1, x2 : input



Forward-Propagation

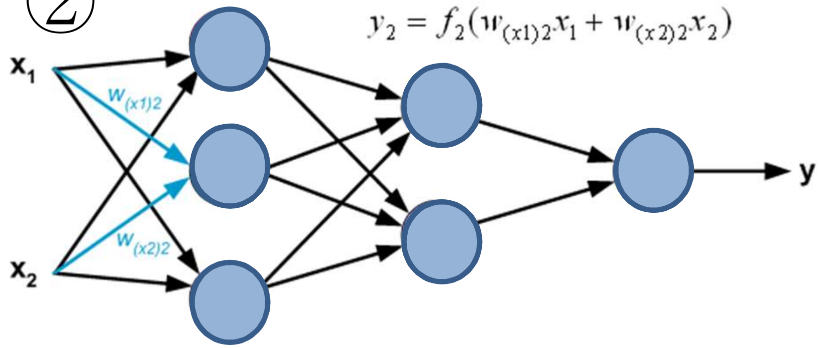
①

$y_1 = f_1(w_{(x1)1}x_1 + w_{(x2)1}x_2)$



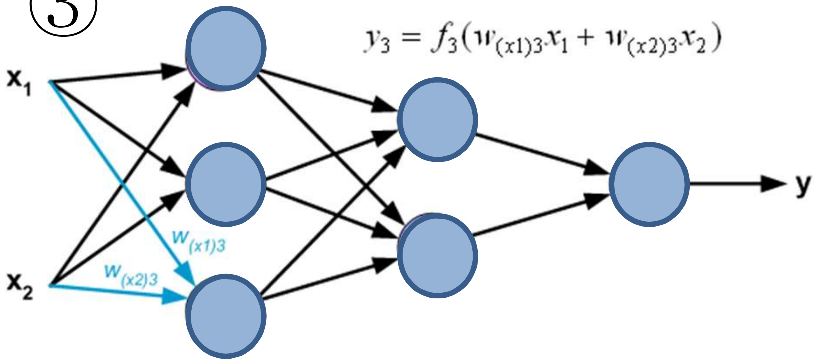
②

$y_2 = f_2(w_{(x1)2}x_1 + w_{(x2)2}x_2)$



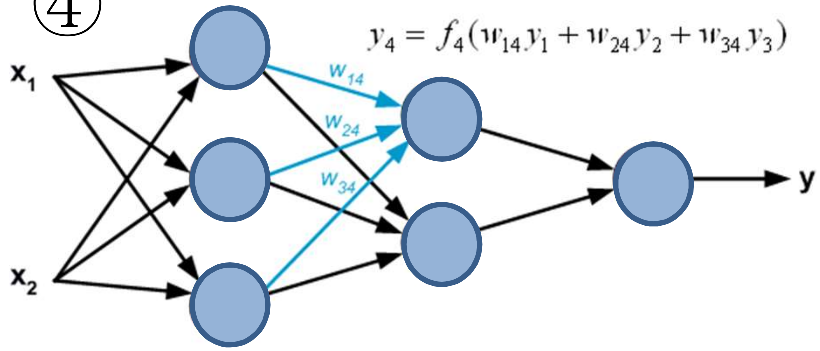
③

$y_3 = f_3(w_{(x1)3}x_1 + w_{(x2)3}x_2)$



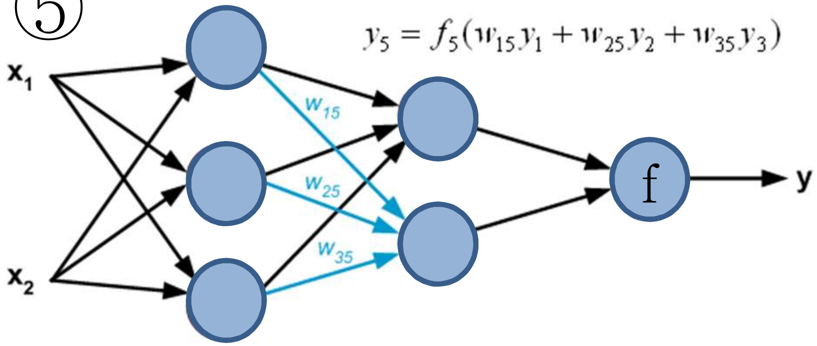
④

$y_4 = f_4(w_{14}y_1 + w_{24}y_2 + w_{34}y_3)$



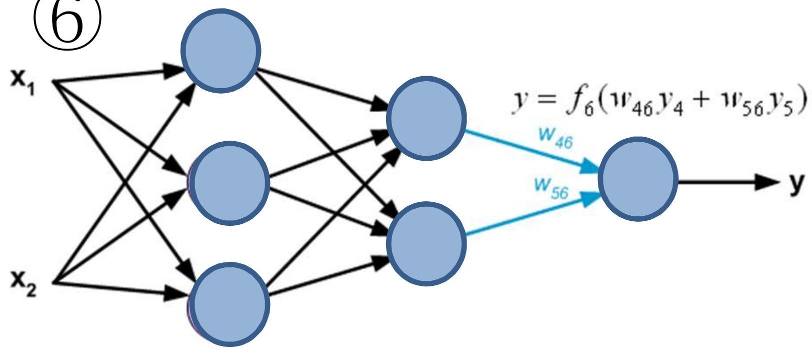
⑤

$y_5 = f_5(w_{15}y_1 + w_{25}y_2 + w_{35}y_3)$



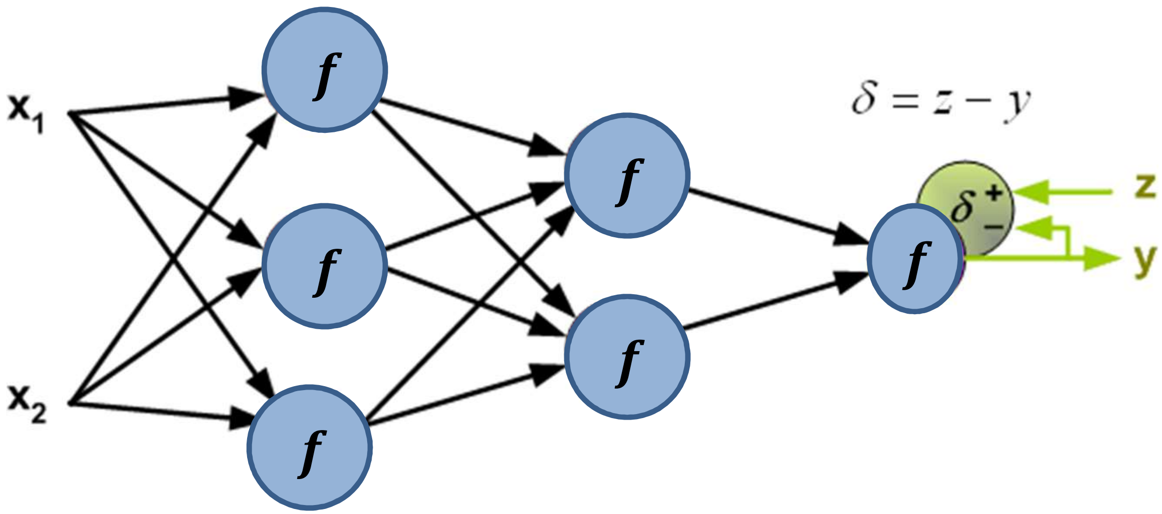
⑥

$y = f_6(w_{46}y_4 + w_{56}y_5)$



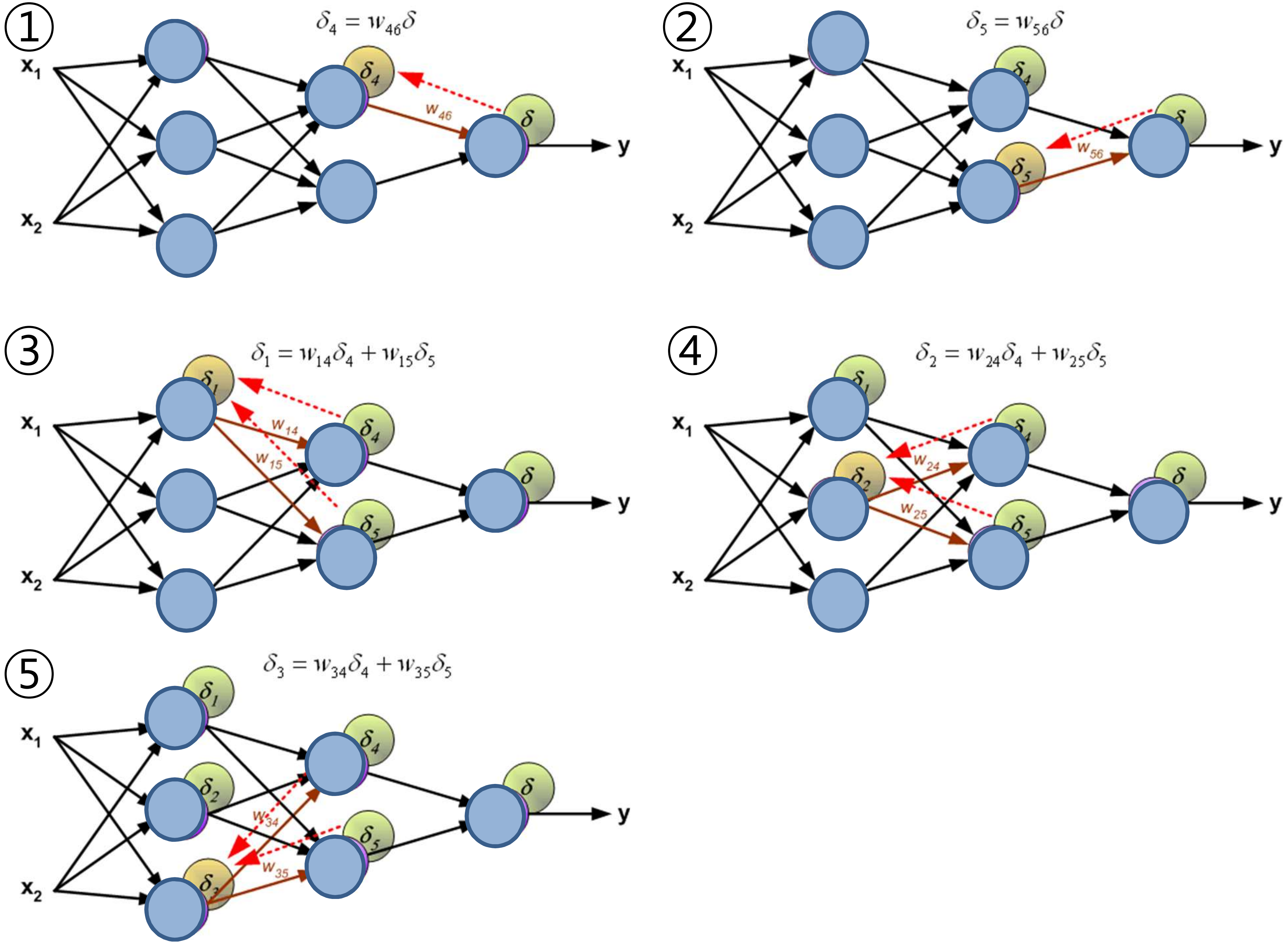
Back-Propagation

Minimize error(δ) by finding the weights (w)



δ = error
z = desired output
y = actual output

Back-Propagation

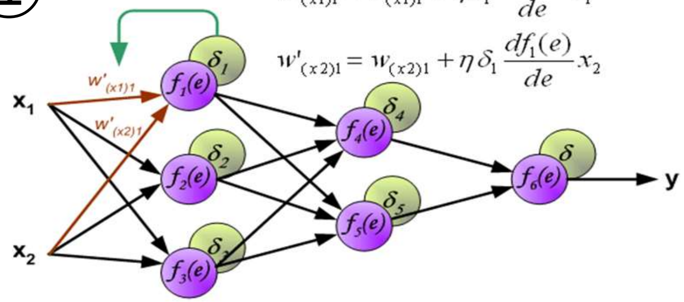


Update the weights, w

①

$$w'_{(x1)1} = w_{(x1)1} + \eta\delta_1 \frac{df_1(e)}{de} x_1$$

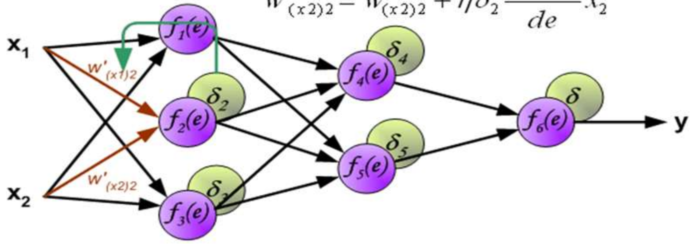
$$w'_{(x2)1} = w_{(x2)1} + \eta\delta_1 \frac{df_1(e)}{de} x_2$$



②

$$w'_{(x1)2} = w_{(x1)2} + \eta\delta_2 \frac{df_2(e)}{de} x_1$$

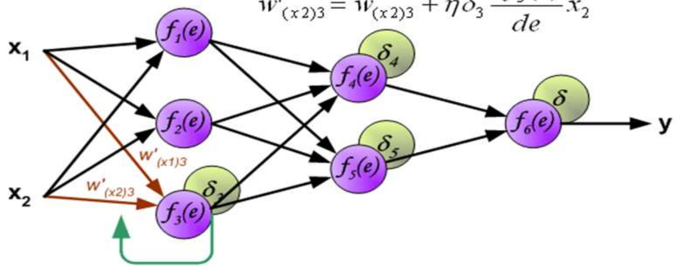
$$w'_{(x2)2} = w_{(x2)2} + \eta\delta_2 \frac{df_2(e)}{de} x_2$$



③

$$w'_{(x1)3} = w_{(x1)3} + \eta\delta_3 \frac{df_3(e)}{de} x_1$$

$$w'_{(x2)3} = w_{(x2)3} + \eta\delta_3 \frac{df_3(e)}{de} x_2$$

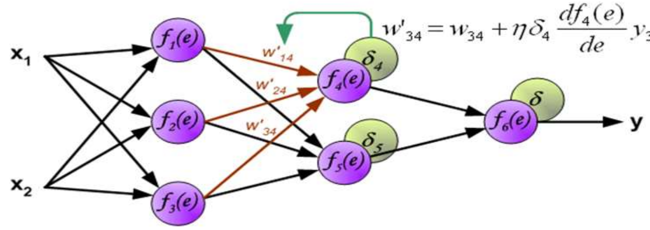


④

$$w'_{14} = w_{14} + \eta\delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta\delta_4 \frac{df_4(e)}{de} y_2$$

$$w'_{34} = w_{34} + \eta\delta_4 \frac{df_4(e)}{de} y_3$$

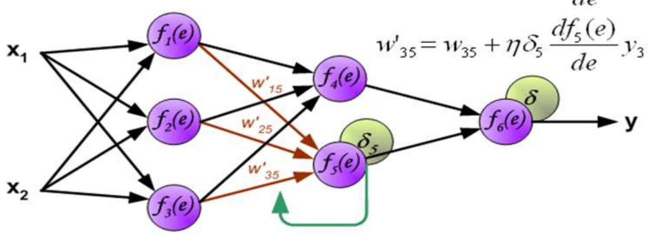


⑤

$$w'_{15} = w_{15} + \eta\delta_5 \frac{df_5(e)}{de} y_1$$

$$w'_{25} = w_{25} + \eta\delta_5 \frac{df_5(e)}{de} y_2$$

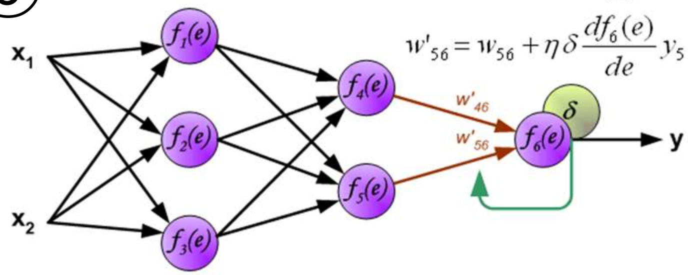
$$w'_{35} = w_{35} + \eta\delta_5 \frac{df_5(e)}{de} y_3$$



⑥

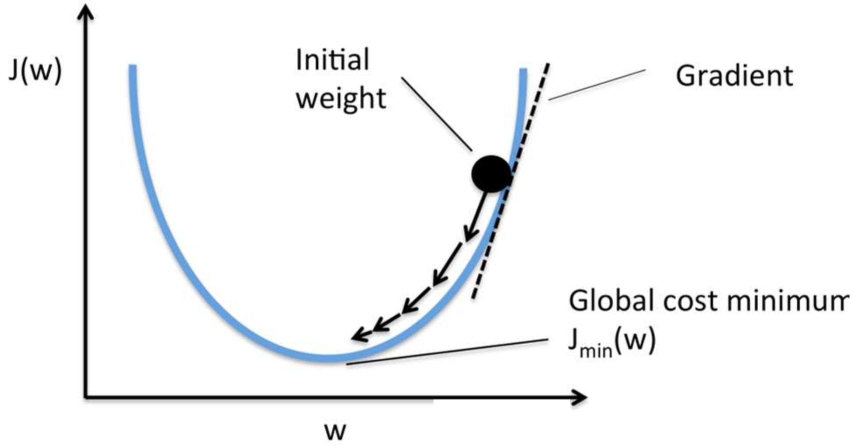
$$w'_{46} = w_{46} + \eta\delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} + \eta\delta \frac{df_6(e)}{de} y_5$$



Gradient Descent

The cost function *J(·)*, the sum of squared errors (SSE), can be written as:



$$J(\mathbf{w}) = \frac{1}{2}\sum_i (\text{target}^{(i)} - \text{output}^{(i)})^2$$

$$SSE = \sum_i (\text{target}^{(i)} - \text{output}^{(i)})^2$$

$$MSE = \frac{1}{n} \times SSE$$

The magnitude and direction of the weight update is computed by taking a step in the opposite direction of the cost gradient

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j},$$

where η is the learning rate. The weights are then updated after each epoch via the following update rule:

$$\mathbf{w} := \mathbf{w} + \Delta\mathbf{w},$$

where **Δw** is a vector that contains the weight updates of each weight coefficient *w*, which are computed as follows:

$$\begin{aligned}\Delta w_j &= -\eta \frac{\partial J}{\partial w_j} \\ &= -\eta \sum_i (\text{target}^{(i)} - \text{output}^{(i)})(-x_j^{(i)}) \\ &= \eta \sum_i (\text{target}^{(i)} - \text{output}^{(i)}) x_j^{(i)}.\end{aligned}$$

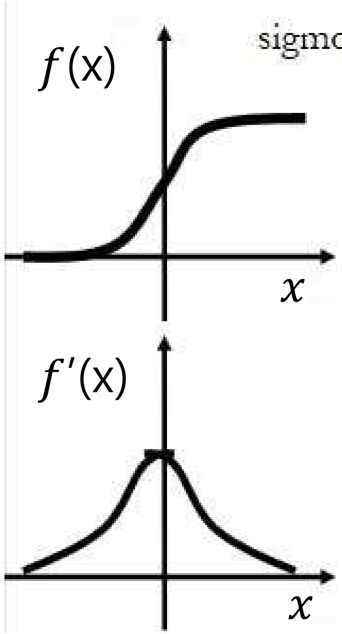
http://sebastianraschka.com/faq/docs/closed-form-vs-gd.html

Example

If activation function is a sigmoid function

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

$$f'(x) = \frac{-e^{-x}}{(1+e^{-x})^2}$$



$$E \; or \; J = \tfrac{1}{2}\sum_{i=1}^{N}(z_i - f(x_i w_i))^2$$

$$\frac{\partial E}{\partial w} = -\sum_{i=1}^{N}(z_i - f(e_j))f'(x_i w_i)x_i$$

$$= -\sum_{i=1}^{N}\left(z_i - f(e_j)\right)f(x_i w_i)(1 - f(x_i w_i))x_i$$

Training rule :

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta(z_i - y_i)f(x_i w_i)(1 - f(x_i w_i))x_i$$

η = learning rate

as in the textbook.