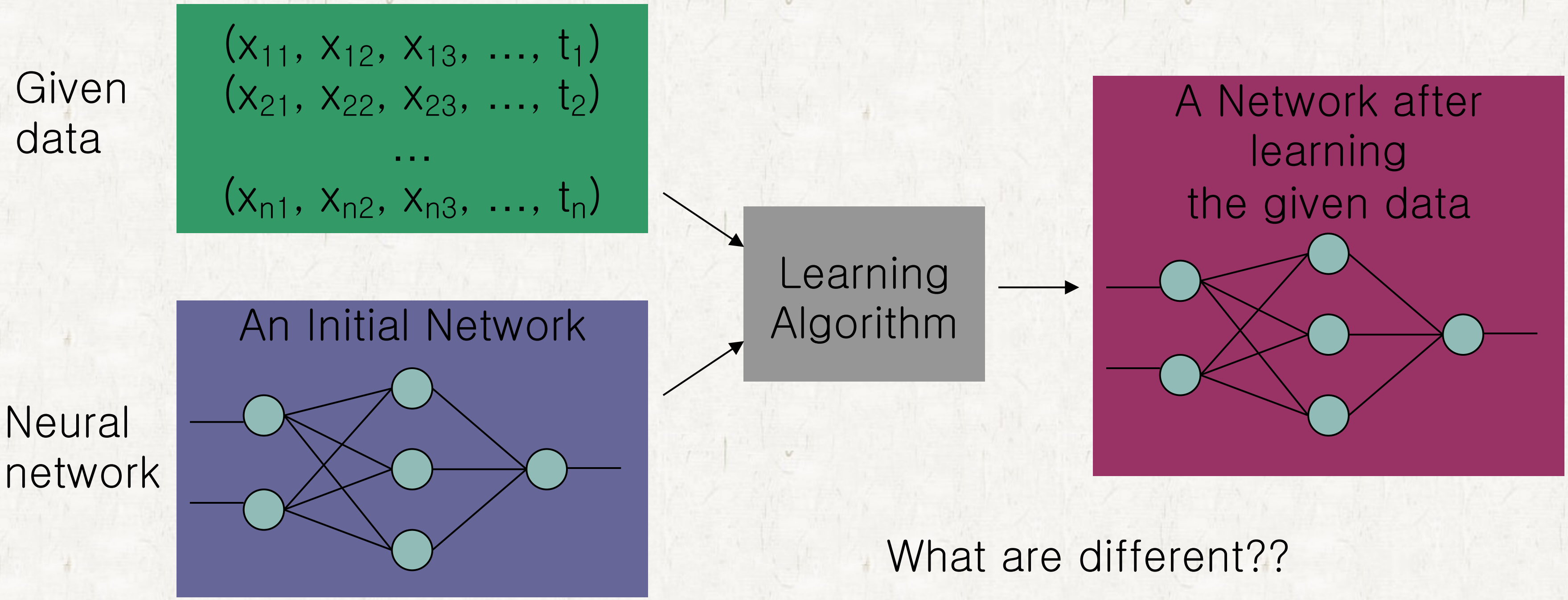
Error Back Propagation

Learning Algorithm (1)

- Preparation for Learning
  - Given input-output data of the target function to learn
  - Given structure of network (# of nodes in hidden layer)
  - Randomly initialized weights



What are different??

Learning Algorithm (2)

- Basic Idea of Learning

Find weights $\boldsymbol{w} = (w_1, w_2, \dots, w_n)$ so that

$NN(\boldsymbol{w}, \boldsymbol{x}) \approx \boldsymbol{t}$ for all $(\boldsymbol{x}, t)$

$NN(\boldsymbol{w}, \boldsymbol{x}) \approx \boldsymbol{t}$ for all $(\boldsymbol{x}, t)$

$$\Leftrightarrow 0 \approx \sum_{(\boldsymbol{x},t)\in Data} |t - NN(\boldsymbol{w}, \boldsymbol{x})|$$

$$\Leftrightarrow 0 \approx \sum_{(\boldsymbol{x},t)\in Data} \left(t - NN(\boldsymbol{w}, \boldsymbol{x})\right)^2$$

$\Leftrightarrow$

$$E(\boldsymbol{w}) = \sum_{(\boldsymbol{x},t)\in Data} \left(t - NN(\boldsymbol{w}, \boldsymbol{x})\right)^2$$

is minimized

Learning Algorithm (3)

- Basic Idea of Learning

Find weights $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$ which minimize

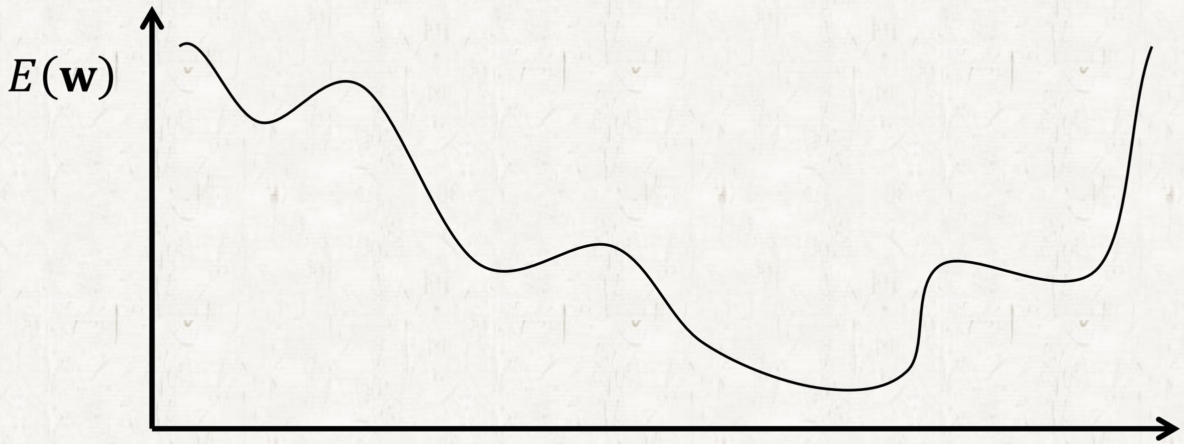
$$E(\boldsymbol{w}) = \sum_{(\boldsymbol{x},t)\in Data} \left(t - NN(\boldsymbol{w}, \boldsymbol{x})\right)^2 \qquad \boldsymbol{w} = (w_1, w_2, \ldots, w_n)$$

Gradient Descent Method (1)

- How?

Find weights $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$ which minimize

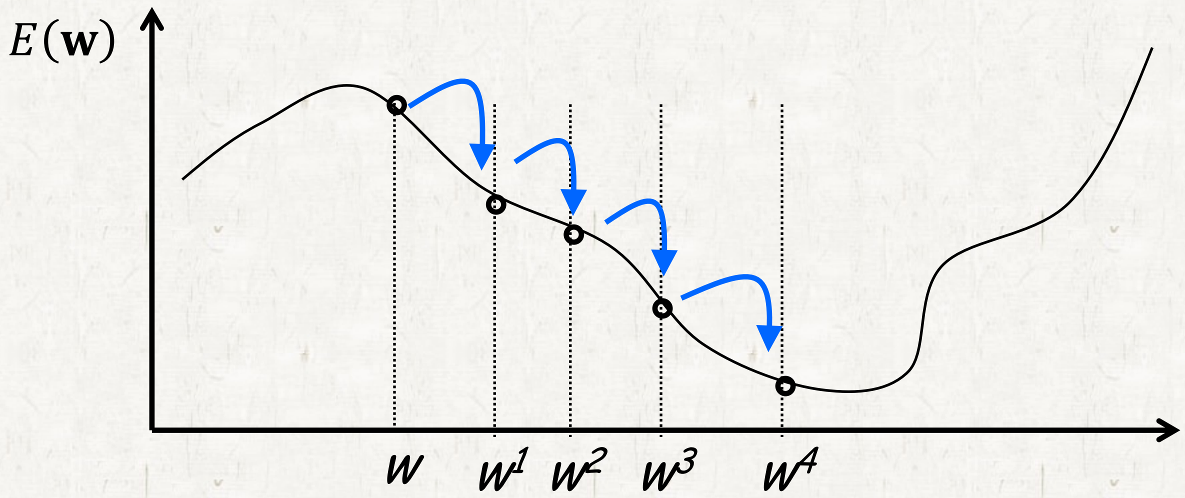
$$E(\mathbf{w}) = \sum_{(\mathbf{x},t)\in Data} \left(y - NN(\mathbf{x};\mathbf{w})\right)^2$$



Gradient Descent Method (2)

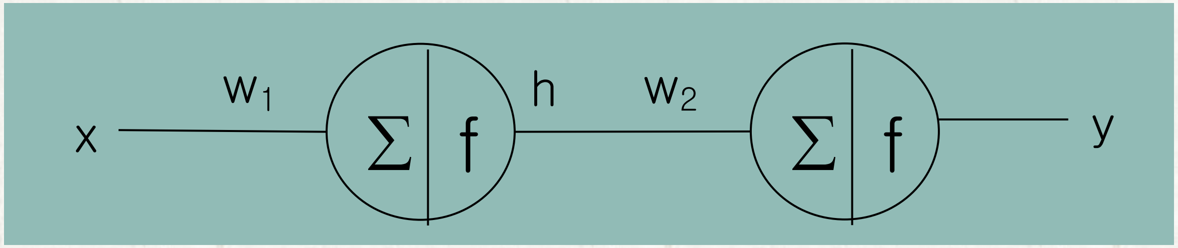
4. Repeat until the gradient is zero

$$w^{t+1} = w^t - \eta \left.\frac{\partial E}{\partial w}\right|_{w=w^t}$$



Gradient Descent Method (3)

- Training of a Simple Neural Network
  - Let's assume that there is one training data (x_t, y_t)



$$net_1 = x_t \cdot w_1$$

$$h = sigmoid(net_1)$$

$$net_2 = h \cdot w_2$$

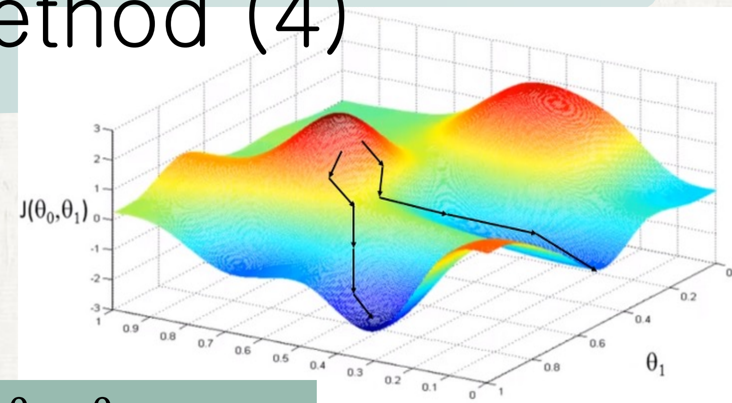
$$y = sigmoid(net_2)$$

$$E = \frac{1}{2}(y_t - y)^2$$

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial net_2}\frac{\partial net_2}{\partial w_2}$$

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial net_2}\frac{\partial net_2}{\partial h}\frac{\partial h}{\partial net_1}\frac{\partial net_1}{\partial w_1}$$

Gradient Descent Method (4)



- Multi-variable case

Randomly choose an initial solution, w_0^0 w_1^0

Repeat

$$w_0^{t+1} = w_0^t - \eta \left.\frac{\partial E}{\partial w_0}\right|_{w_0=w_0^t, w_1=w_1^t}$$

$$w_1^{t+1} = w_1^t - \eta \left.\frac{\partial E}{\partial w_1}\right|_{w_0=w_0^t, w_1=w_1^t}$$

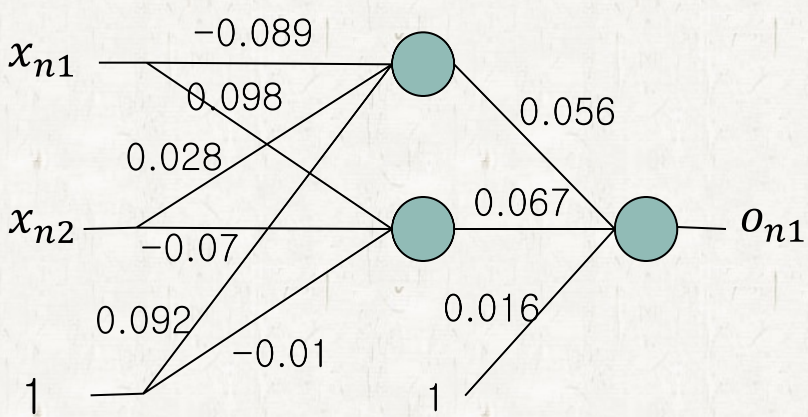
Until stopping condition is satisfied

Example of Error Back Propagation (1)

- Example : XOR

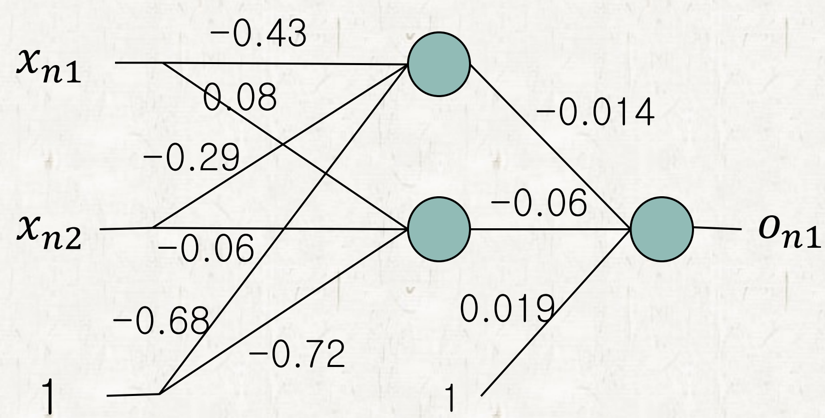
Iteration : 0

x_{n1}	x_{n2}	t_{n1}	o_{n1}
1	1	0	0.52
1	0	1	0.50
0	1	1	0.52
0	0	0	0.55



Iteration : 1000

x_{n1}	x_{n2}	t_{n1}	o_{n1}
1	1	0	0.50
1	0	1	0.48
0	1	1	0.50
0	0	0	0.52



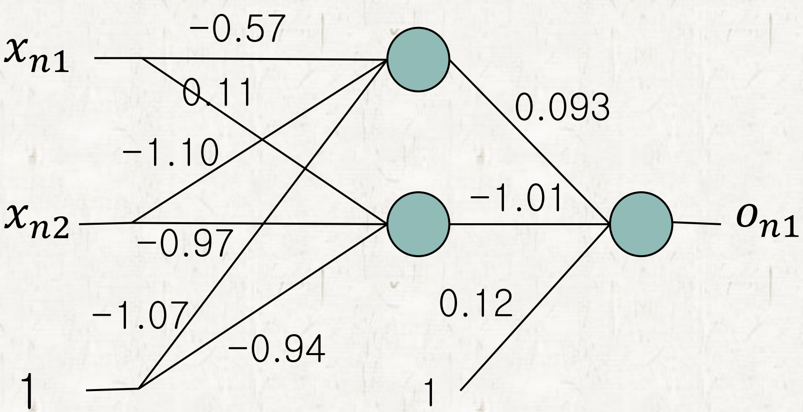
J.H. Lee, Dept. of Software, Sungkyunkwan Univ.

Example of Error Back Propagation (2)

- Example : XOR

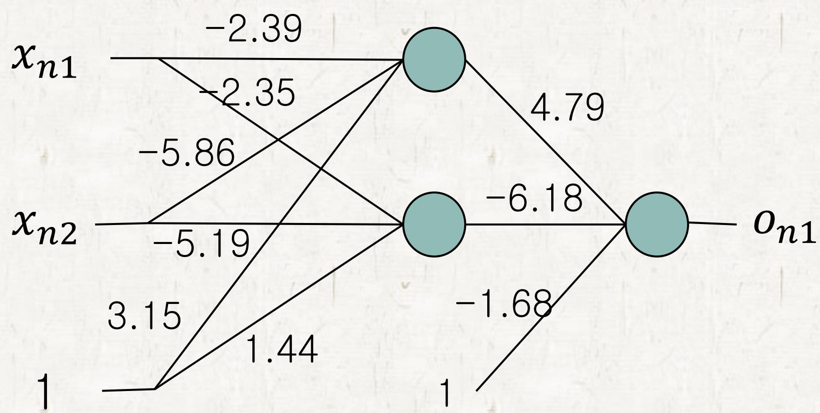
Iteration : 2000

x_{n1}	x_{n2}	t_{n1}	o_{n1}
1	1	0	0.53
1	0	1	0.48
0	1	1	0.50
0	0	0	0.48



Iteration : 3000

x_{n1}	x_{n2}	t_{n1}	o_{n1}
1	1	0	0.30
1	0	1	0.81
0	1	1	0.81
0	0	0	0.11

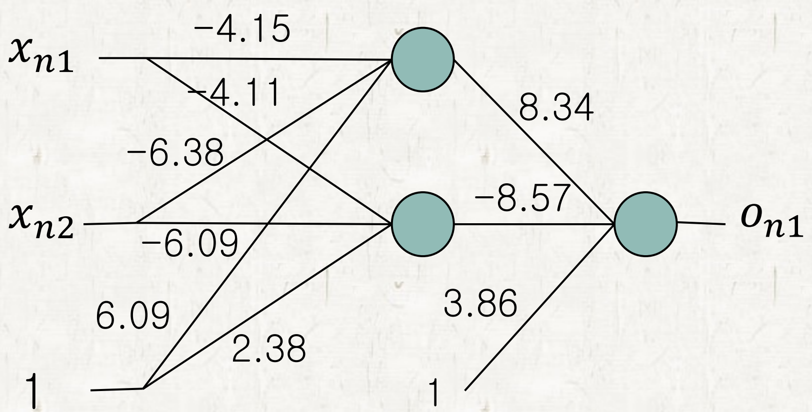


Example of Error Back Propagation (3)

- Example : XOR

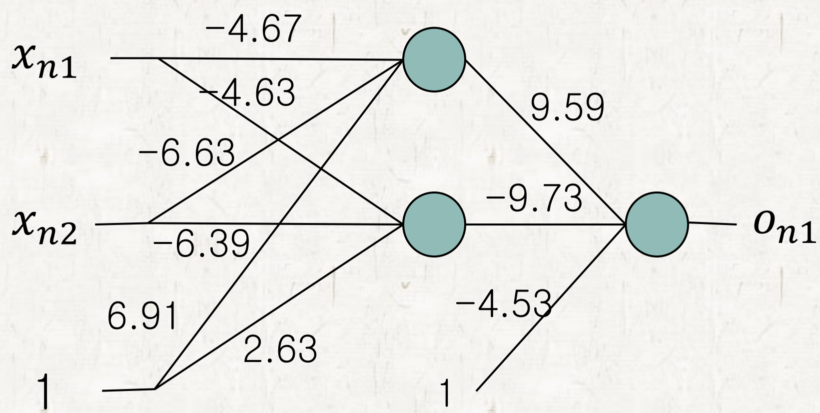
Iteration : 5000

x_{n1}	x_{n2}	t_{n1}	o_{n1}
1	1	0	0.05
1	0	1	0.96
0	1	1	0.96
0	0	0	0.03



Iteration : 10000

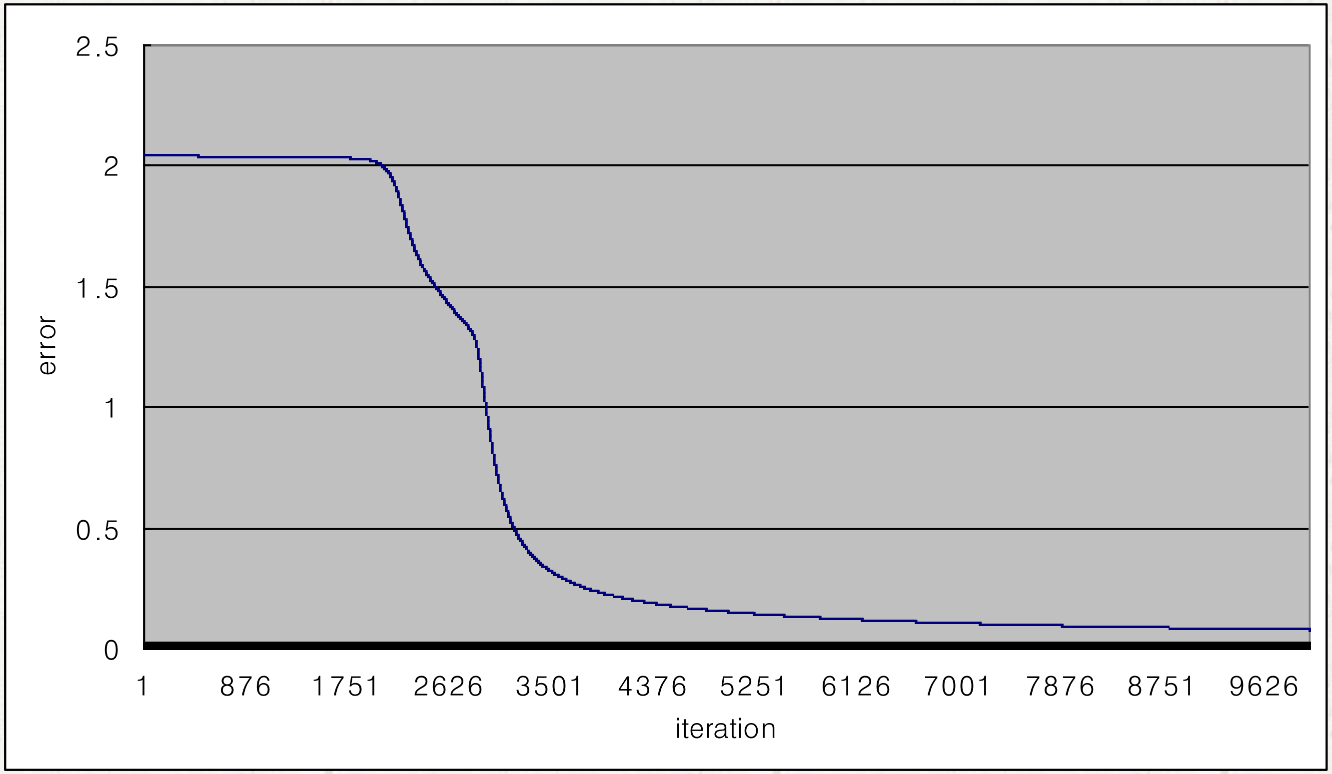
x_{n1}	x_{n2}	t_{n1}	o_{n1}
1	1	0	0.02
1	0	1	0.98
0	1	1	0.98
0	0	0	0.02



J.H. Lee, Dept. of Software, Sungkyunkwan Univ.

Example of Error Back Propagation (4)

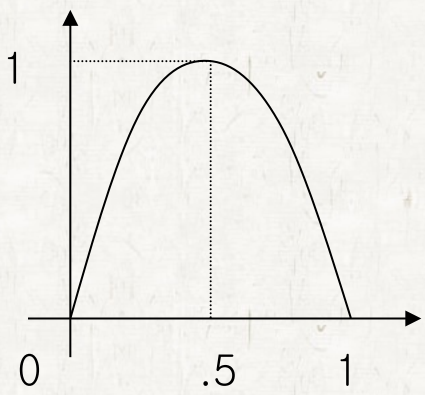
- Example : XOR
  - Error graph

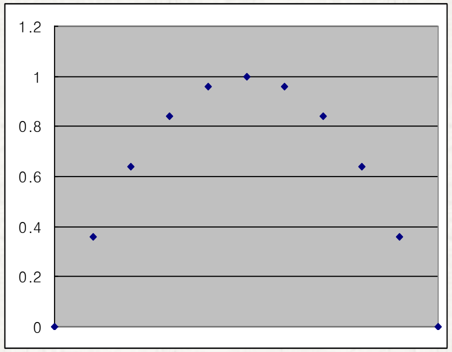


Example of Error Back Propagation (5)

- Example2 :
  - Hidden nodes : 4
  - Iteration : 500,000
  - Learning rate : 0.7

f(x) = 4x*(1−x)

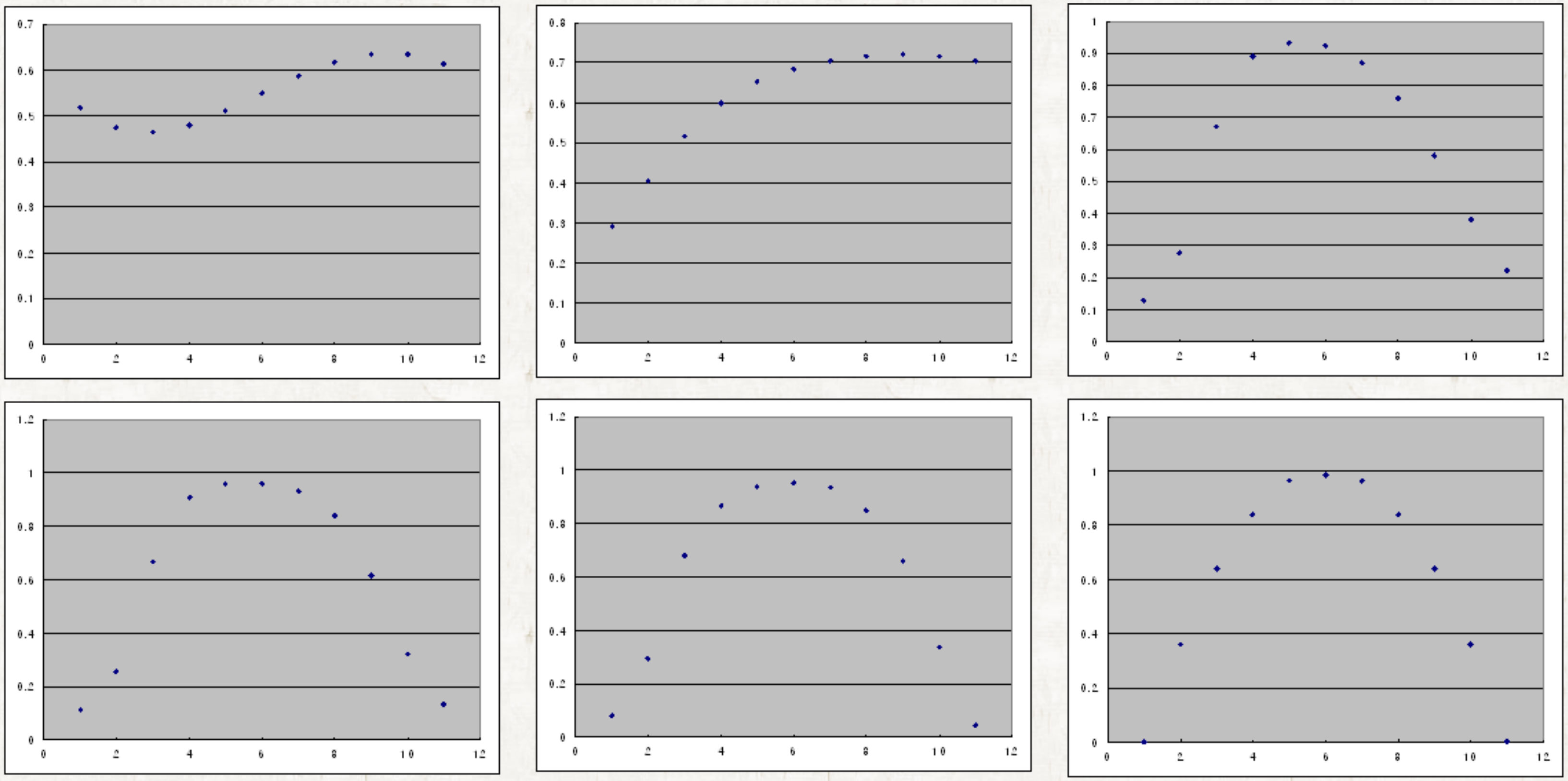




Input	Output
0.00	0.00
0.10	0.36
0.20	0.64
0.30	0.84
0.40	0.96
0.50	1.00
0.60	0.96
0.70	0.84
0.80	0.64
0.90	0.36
1.00	0.00

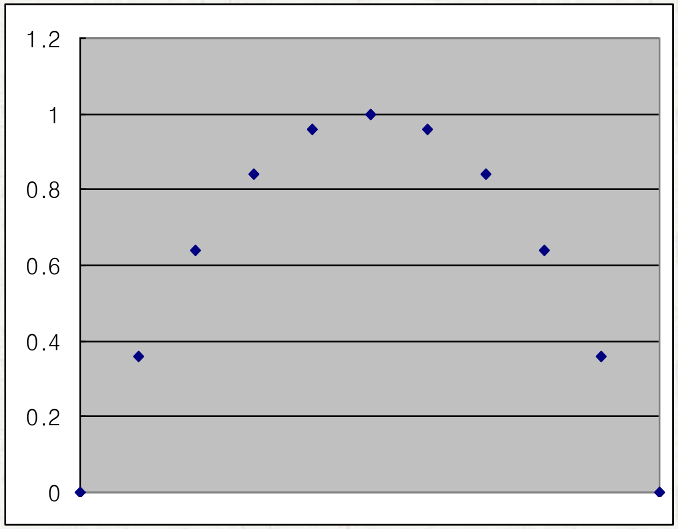
Example of Error Back Propagation (6)

- Example2

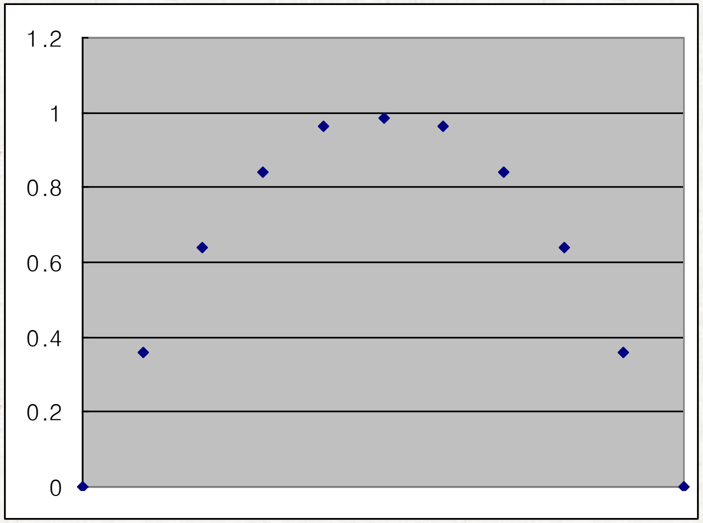


Generalization and Overfitting (1)

- We gave only 11 points
  - A NN learned only that 11 points



Training data

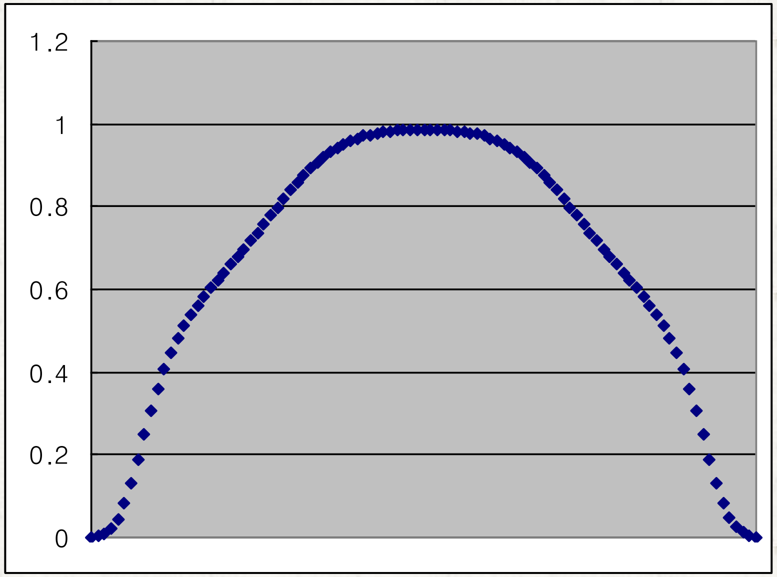


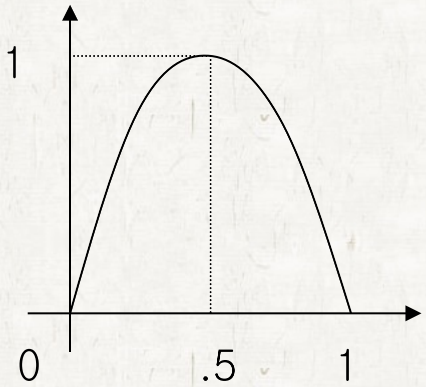
Training result

  - Can the NN answer to the un-learned points?

Generalization and Overfitting (2)

- Yes, NNs generalize what they have learned

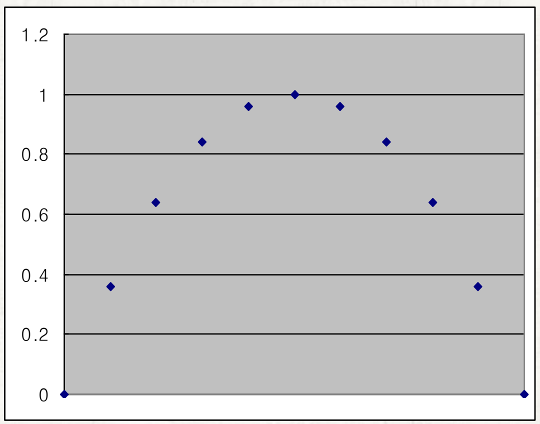




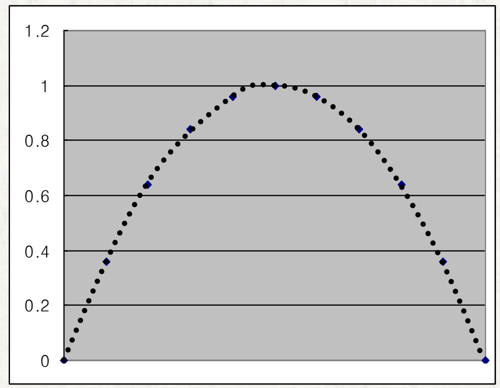
J.H. Lee, Dept. of Software, Sungkyunkwan Univ.

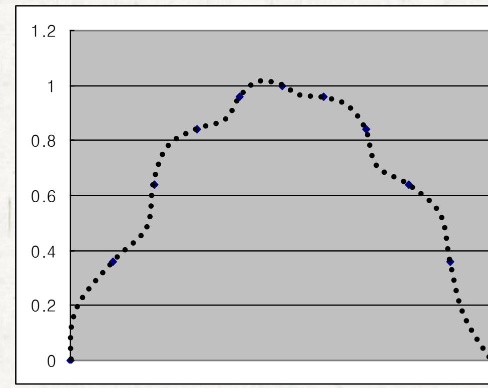
Generalization and Overfitting (3)

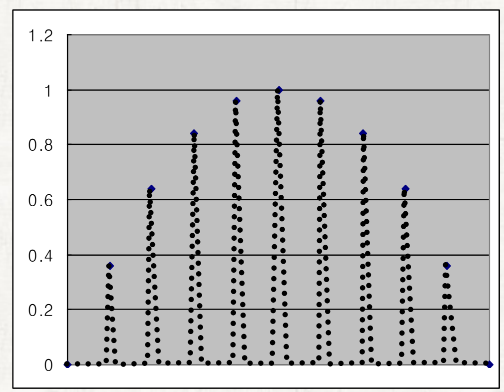
- Which one is better?



Training data

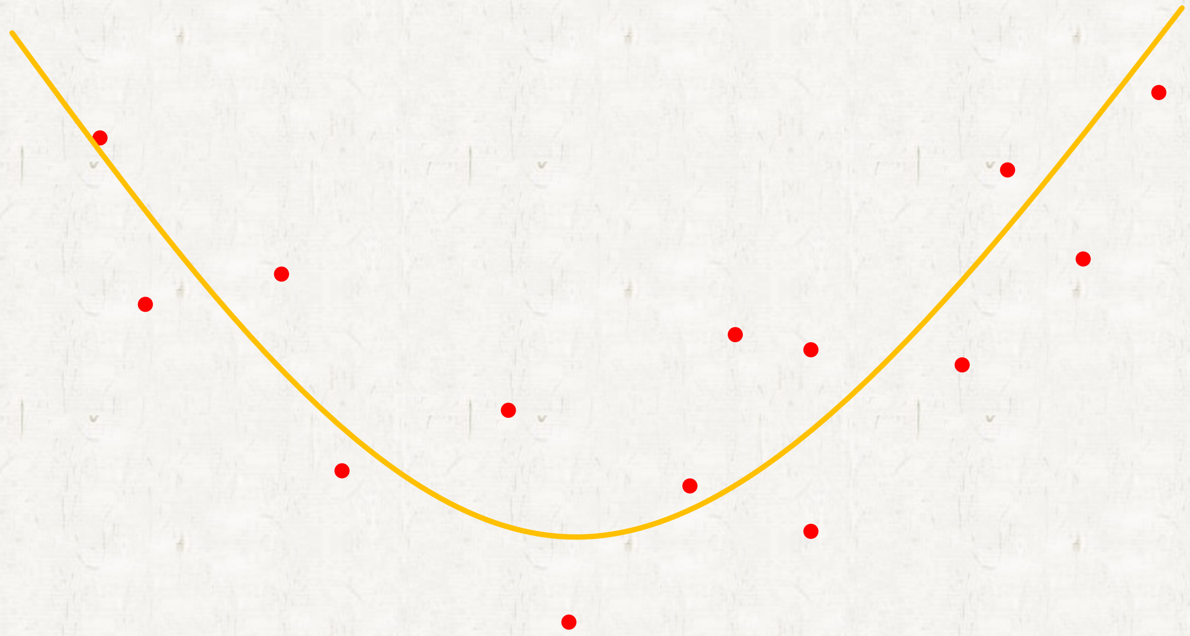






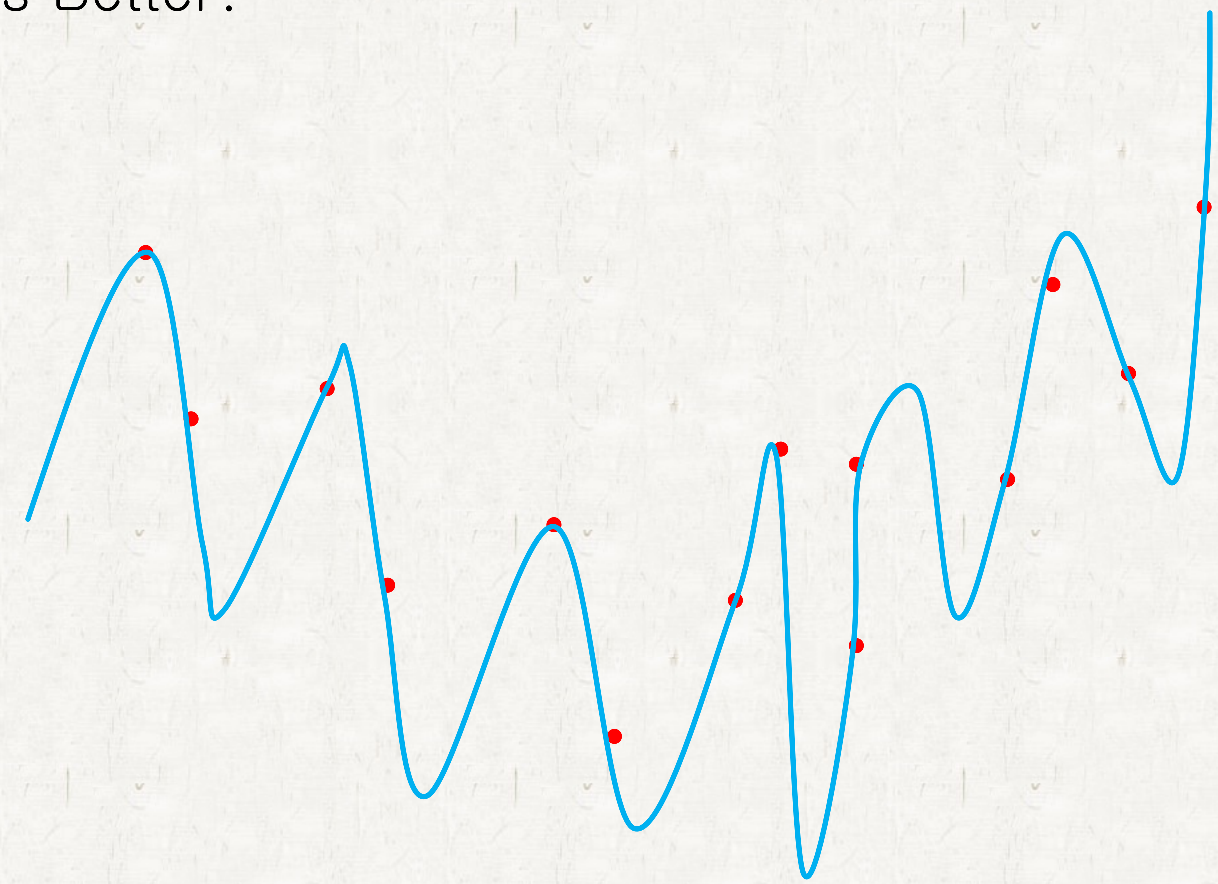
Generalization and Overfitting (4)

- Which is Better?

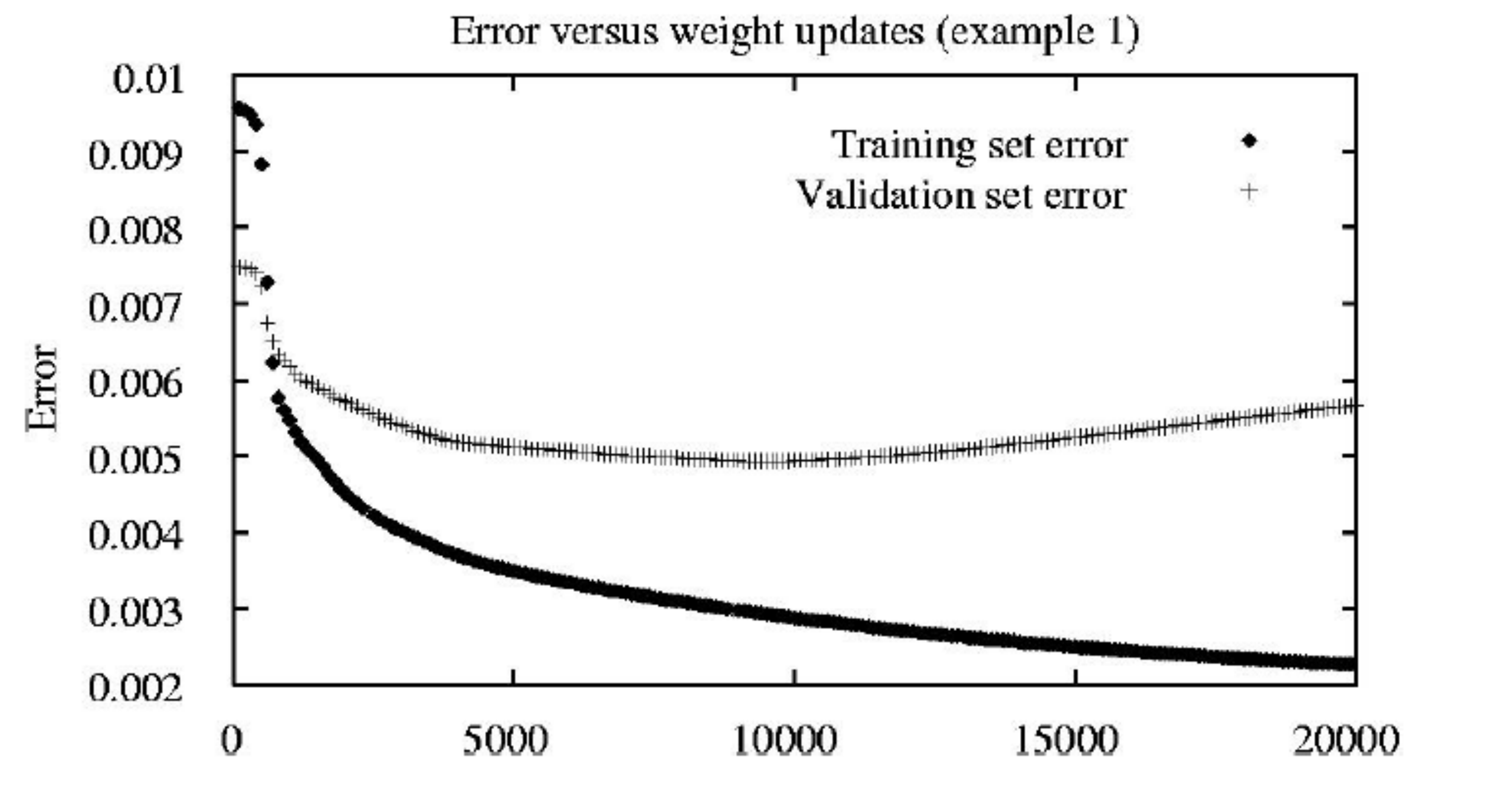


Generalization and Overfitting (5)

- Which is Better?

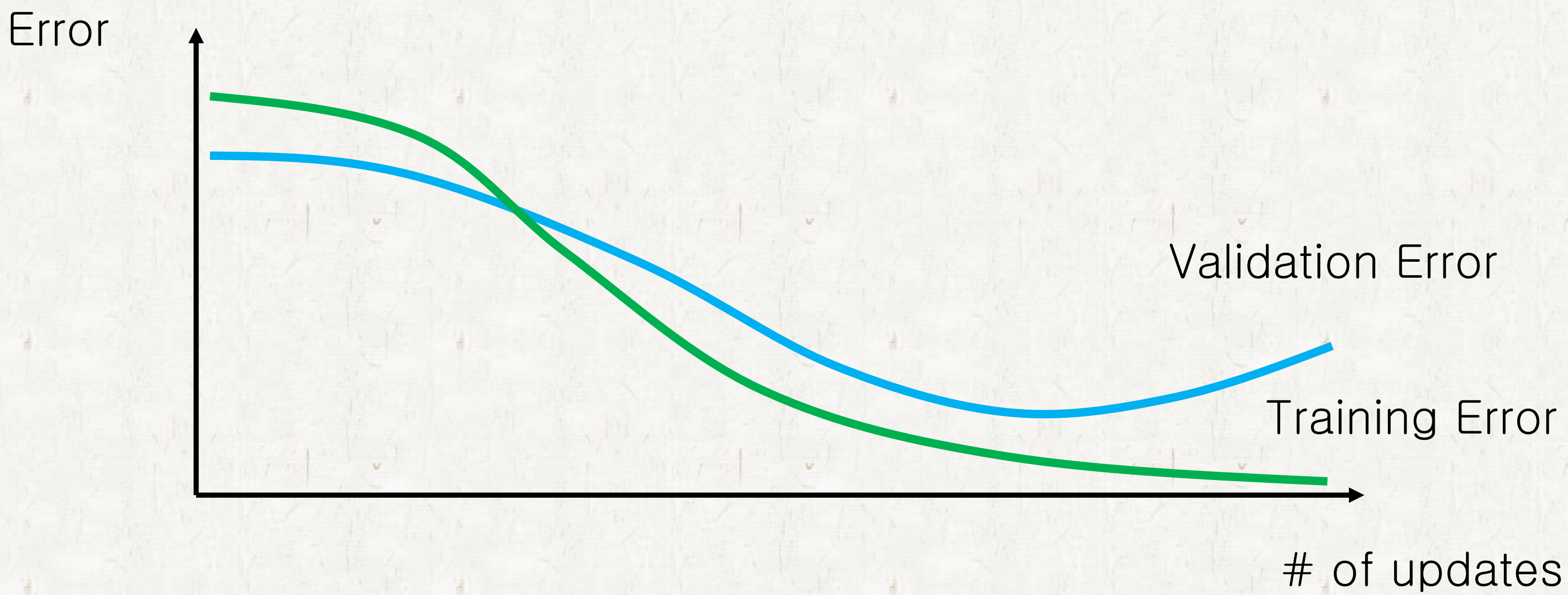


Generalization and Overfitting (6)



Generalization and Overfitting (7)

- Early Stopping



Generalization and Overfitting (8)

- To increase generalization accuracy
  - Find the optimal number of neurons
  - Find the optimal number of training iterations
  - Use regularization
  - Use more training data