Error Back Propagation

Introduction

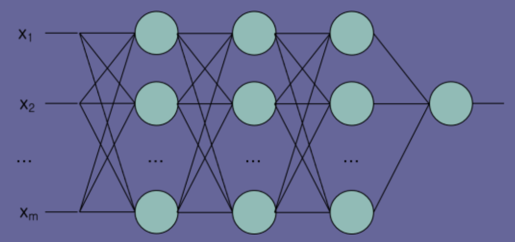
- Preparation for Learning
  - Given input-output data of the target function to learn
  - Given structure of network (# of nodes in hidden layer)
  - Randomly initialized weights

Given data

$(x_{11}, x_{12}, x_{13}, \ldots, t_1)$
$(x_{21}, x_{22}, x_{23}, \ldots, t_2)$
$\ldots$
$(x_{n1}, x_{n2}, x_{n3}, \ldots, t_n)$

Neural network

An Initial Network



Introduction

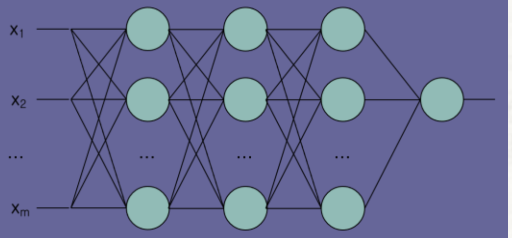
- Learning Algorithm
  - Update connection weights

Given data

(x_{11}, x_{12}, x_{13}, …, t_1)
(x_{21}, x_{22}, x_{23}, …, t_2)
…
(x_{n1}, x_{n2}, x_{n3}, …, t_n)

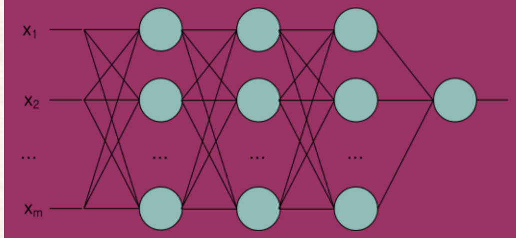
Neural network

An Initial Network



Learning Algorithm

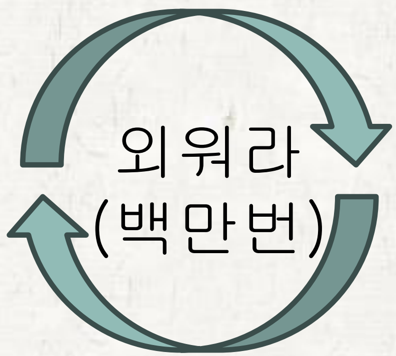
A Network after learning the given data

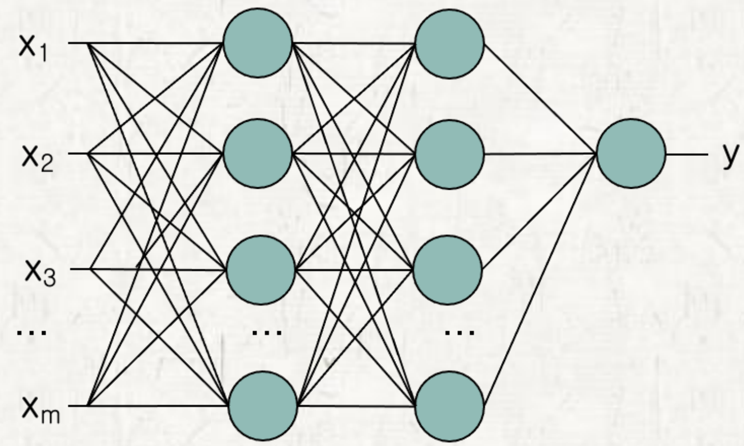


What are different??

Introduction

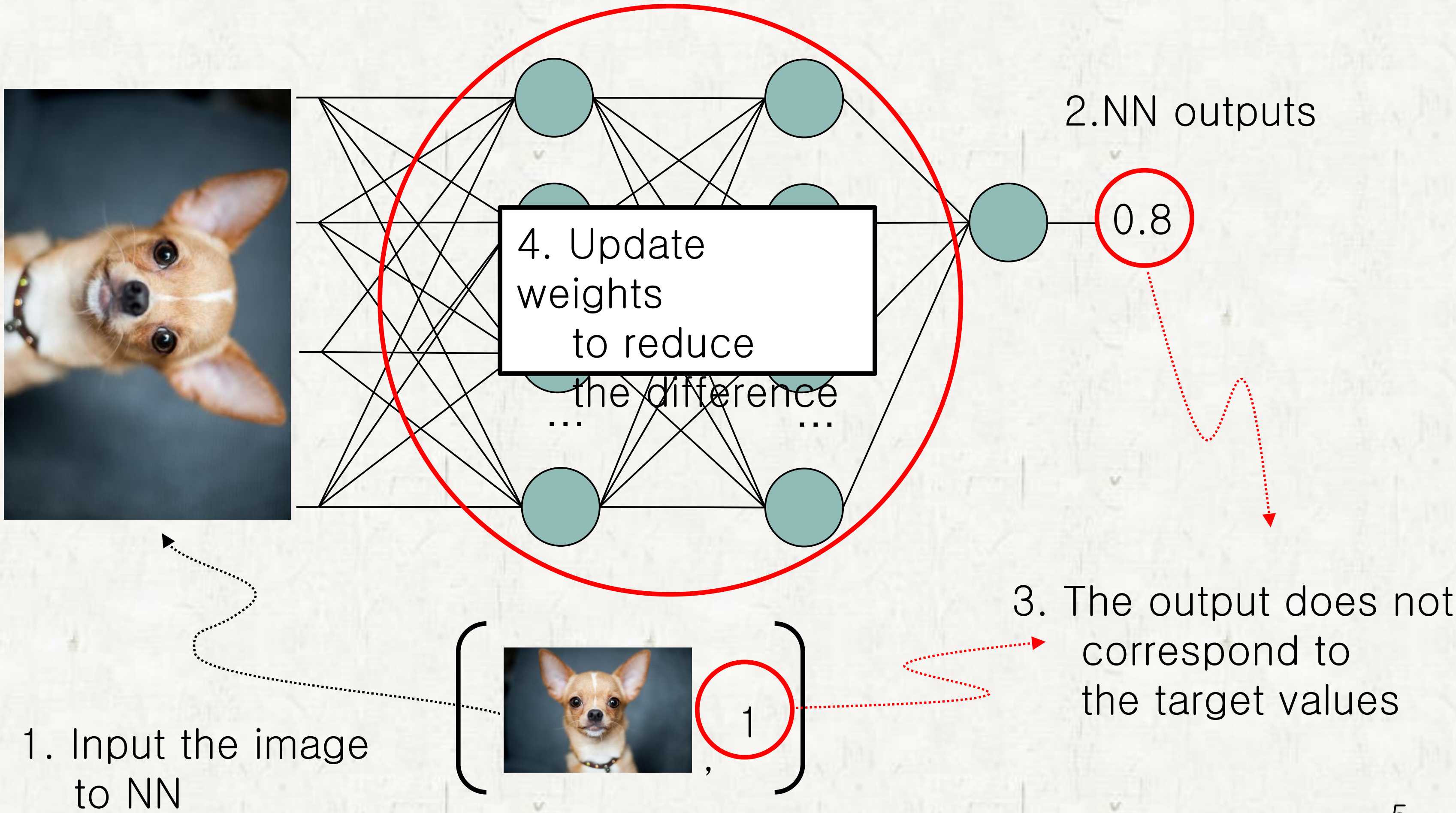
- Learning Algorithm





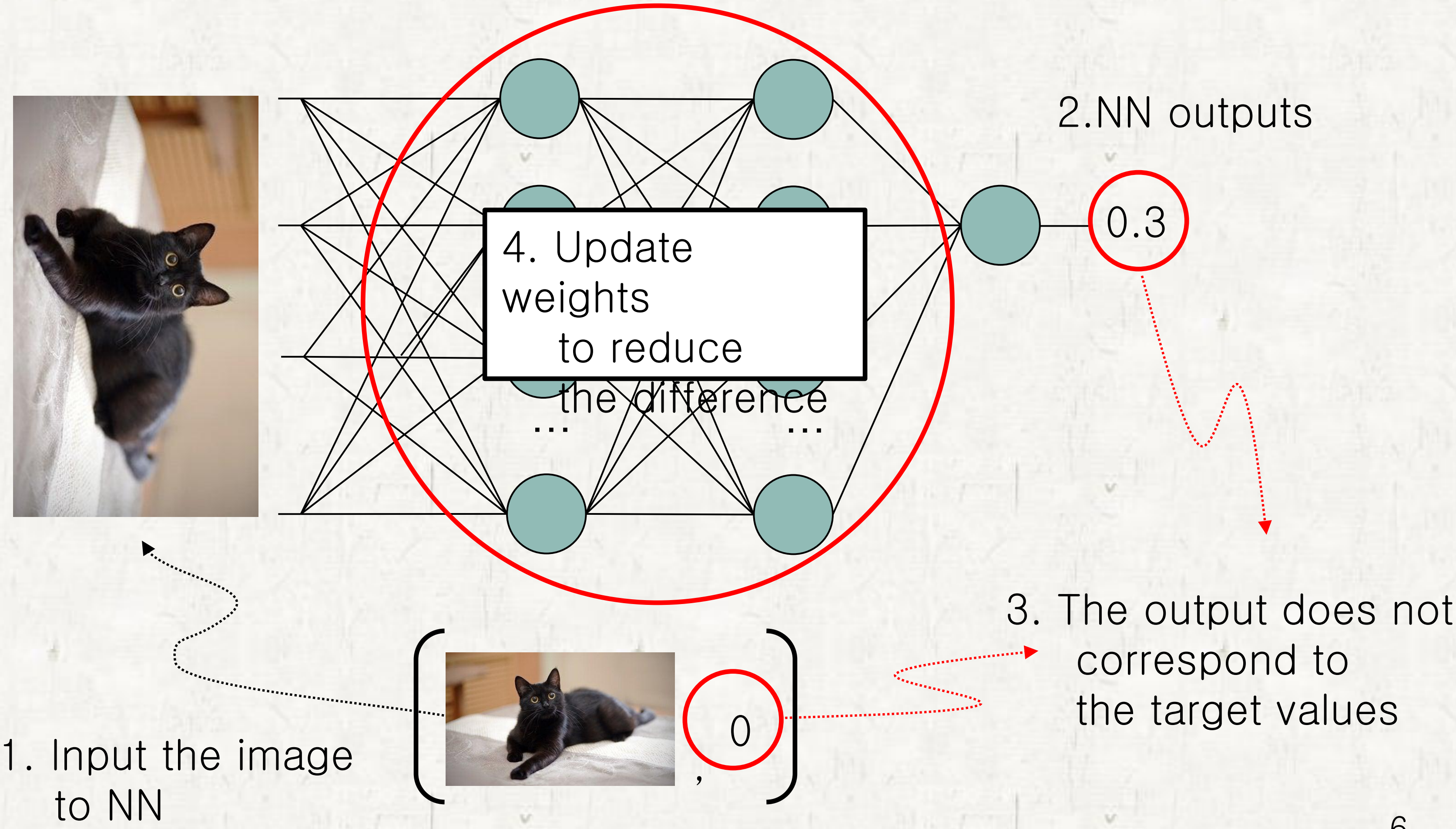
Introduction

- Learning Algorithm



Introduction

- Learning Algorithm

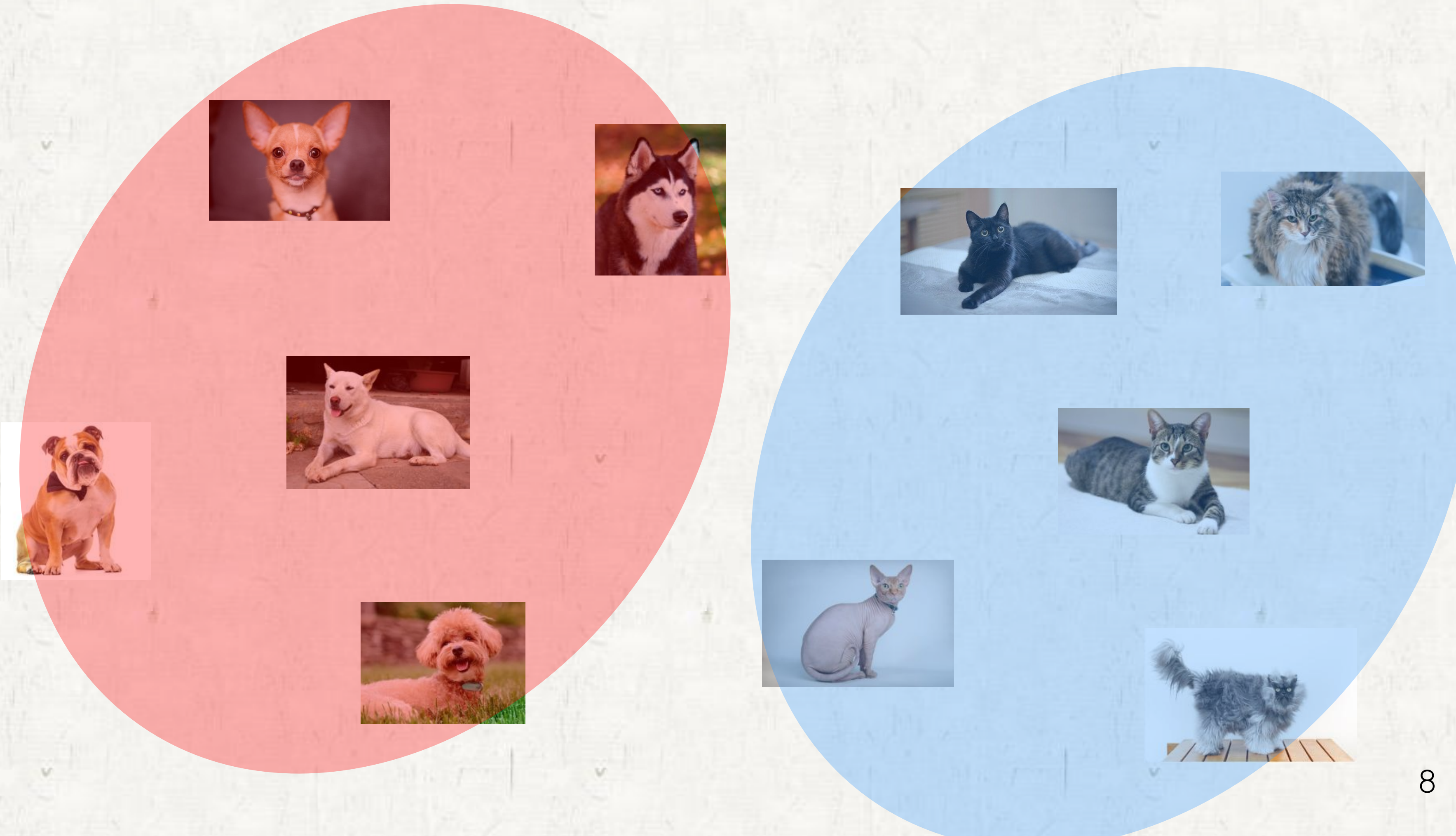


Introduction

- Learning Algorithm: 배운 것을 외우게 됨

Introduction

- Learning Algorithm: 배운 것을 일반화하게 됨



Learning Algorithm

- 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

- Basic Idea of Learning

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

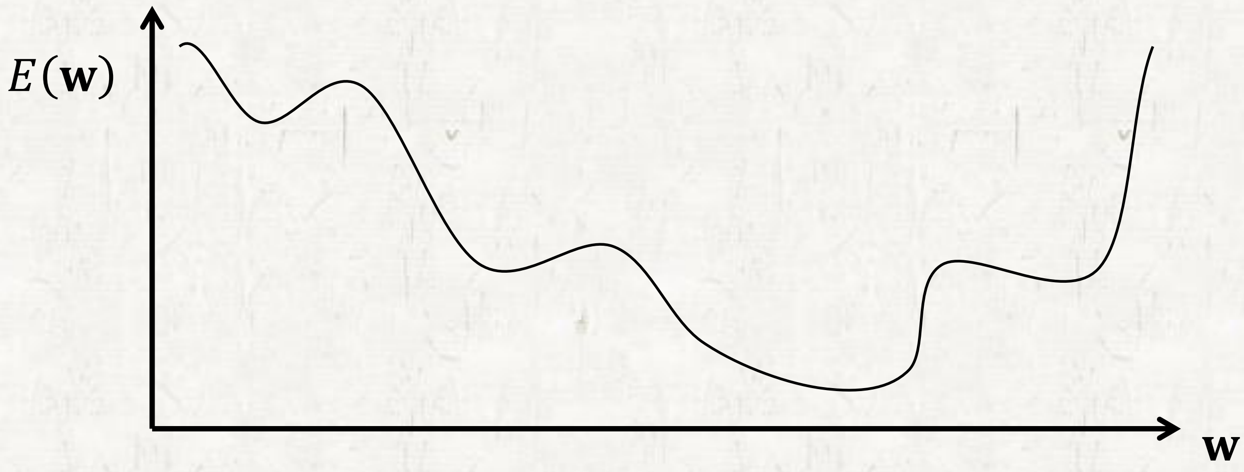
$$E(\boldsymbol{w}) = \sum_{(\boldsymbol{x},t)\in Data} \big(t - NN(\boldsymbol{w}, \boldsymbol{x})\big)^2 \qquad \boldsymbol{w} = (w_1, w_2, \dots, w_n)$$

Gradient Descent Method

- How?

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

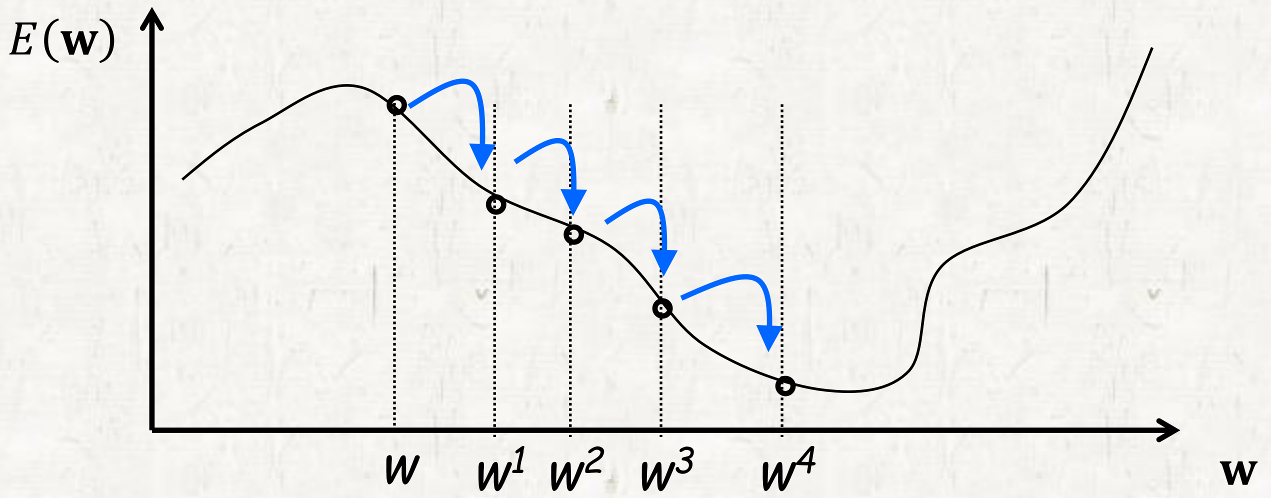
$$E(\mathbf{w}) = \sum_{(\mathbf{x},t)\in Data} \left(y - NN(\mathbf{x};\mathbf{w})\right)^2$$



Gradient Descent Method

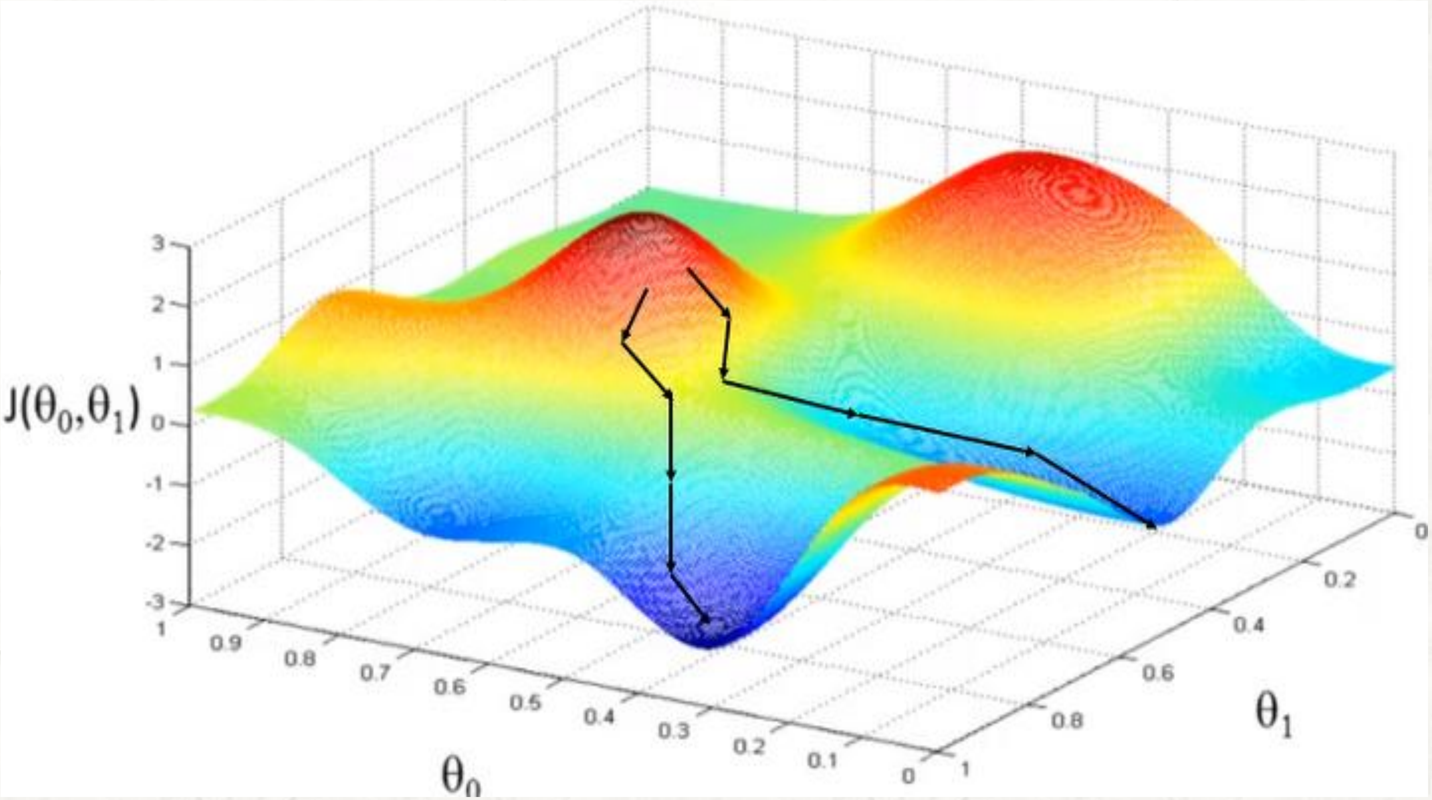
- How?

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



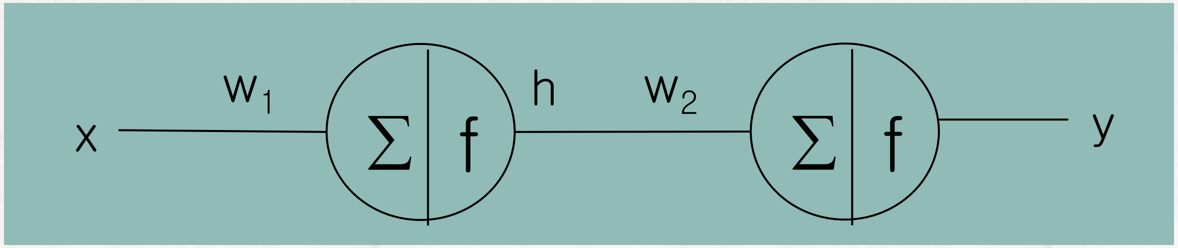
Gradient Descent Method

- How?



Simple Examples

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



$$s_1 = x_t \cdot w_1$$

$$h = sigmoid(s_1)$$

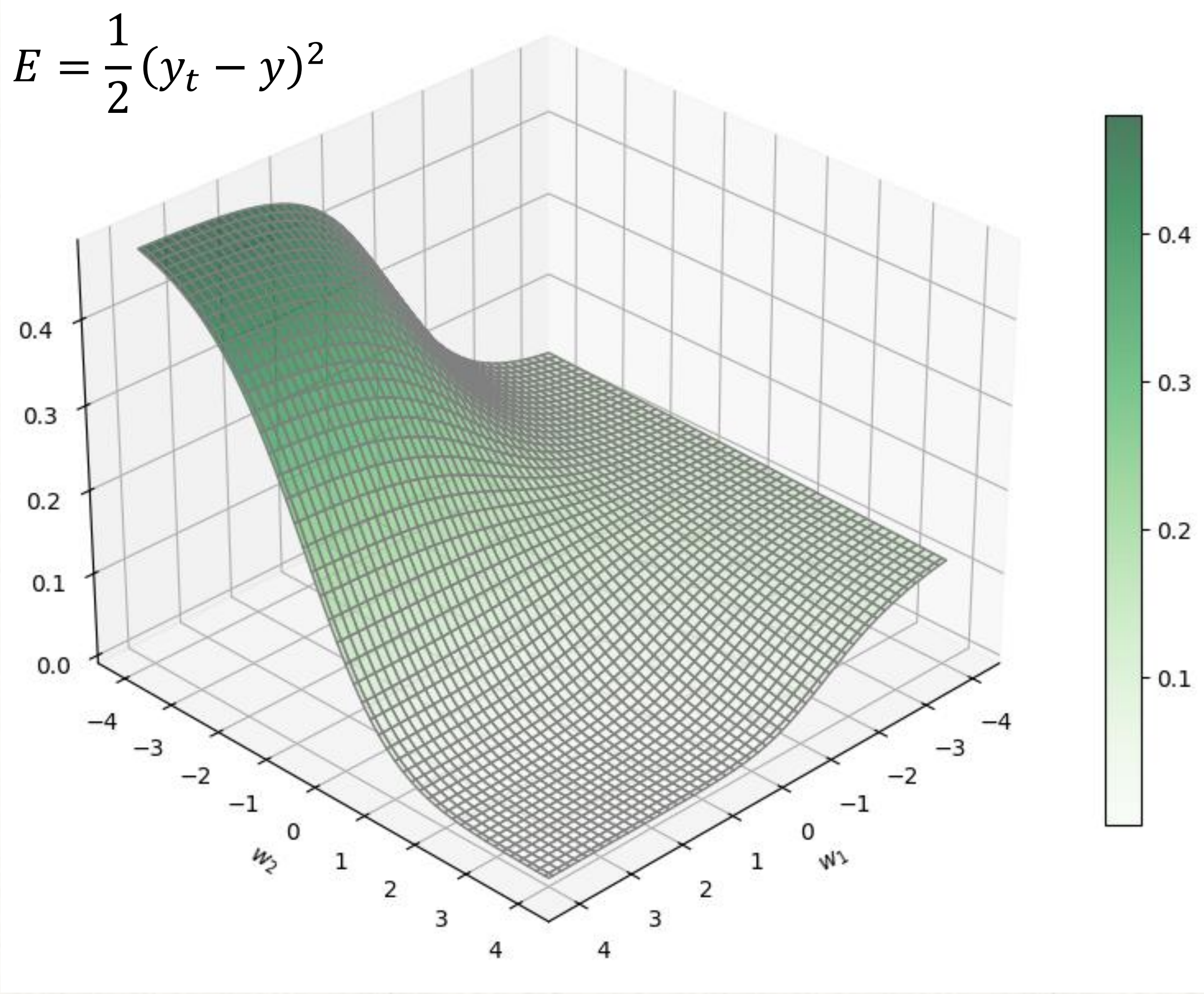
$$s_2 = h \cdot w_2$$

$$y = sigmoid(s_2)$$

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

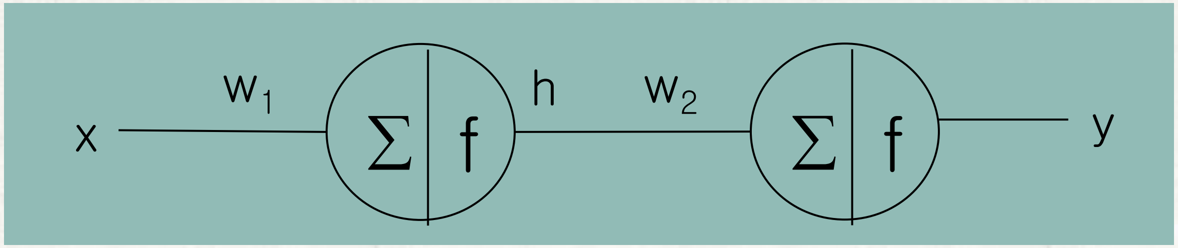
Simple Examples

- Training of a Simple Neural Network



Simple Examples

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



$$s_1 = x_t \cdot w_1$$

$$h = sigmoid(s_1)$$

$$s_2 = h \cdot w_2$$

$$y = sigmoid(s_2)$$

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

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial s_2}\frac{\partial s_2}{\partial w_2}$$

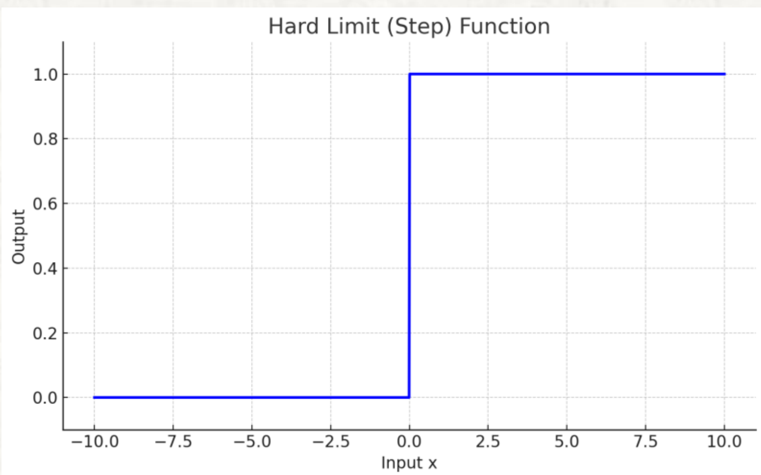
$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial s_2}\frac{\partial s_2}{\partial h}\frac{\partial h}{\partial s_1}\frac{\partial s_1}{\partial w_1}$$

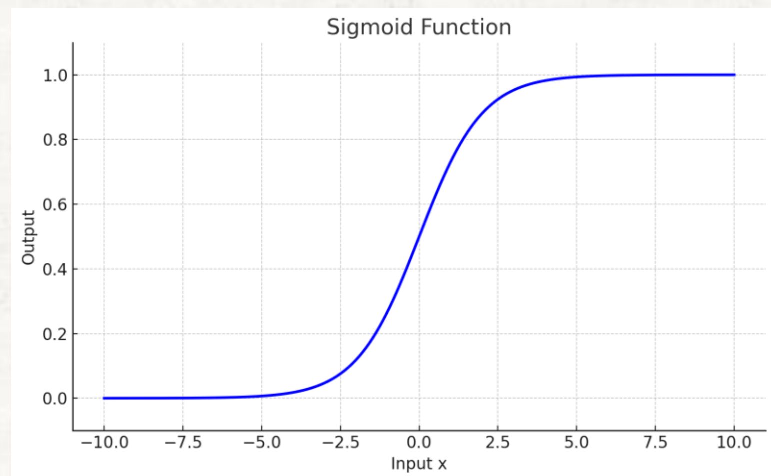
Simple Examples

- Sigmoid function

$$y = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

$$y = \frac{1}{1 + e^{-x}}$$





Simple Examples

- Training of a Simple Neural Network

$$s_1 = x_t \cdot w_1$$

$$h = sigmoid(s_1)$$

$$s_2 = h \cdot w_2$$

$$y = sigmoid(s_2)$$

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

$$\frac{\partial s_1}{\partial w_1} = x_t$$

$$\frac{\partial h}{\partial s_1} = h(1 - h)$$

$$\frac{\partial s_2}{\partial w_2} = h \qquad \frac{\partial s_2}{\partial h} = w_2$$

$$\frac{\partial y}{\partial s_2} = y(1 - y)$$

$$\frac{\partial E}{\partial y} = -(y_t - y)$$

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial s_2}\frac{\partial s_2}{\partial w_2}$$

$$= -(y_t - y)y(1 - y)h$$

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial s_2}\frac{\partial s_2}{\partial h}\frac{\partial h}{\partial s_1}\frac{\partial s_1}{\partial w_1}$$

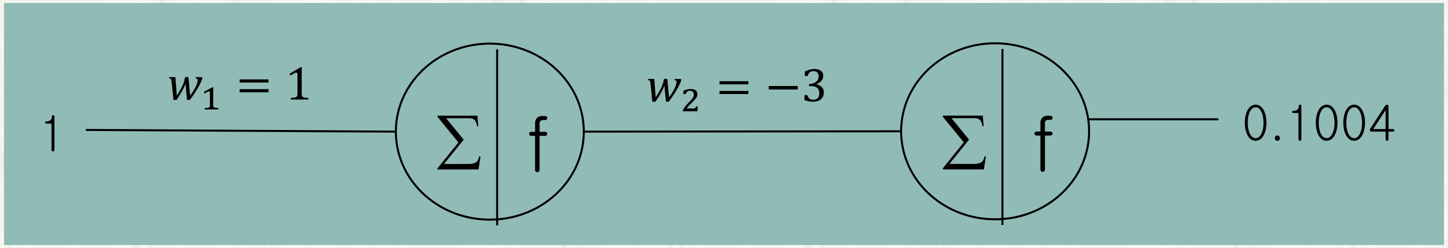
$$= -(y_t - y)y(1 - y)w_2 h(1 - h)x_t$$

Simple Examples

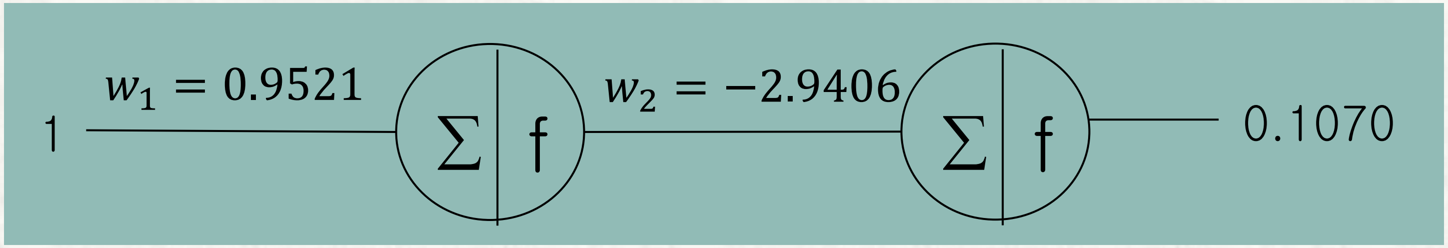
- Training of a Simple Neural Network

$$(x_t, y_t) = (1,1), w_1 = 1, w_2 = -3, \eta = 1$$

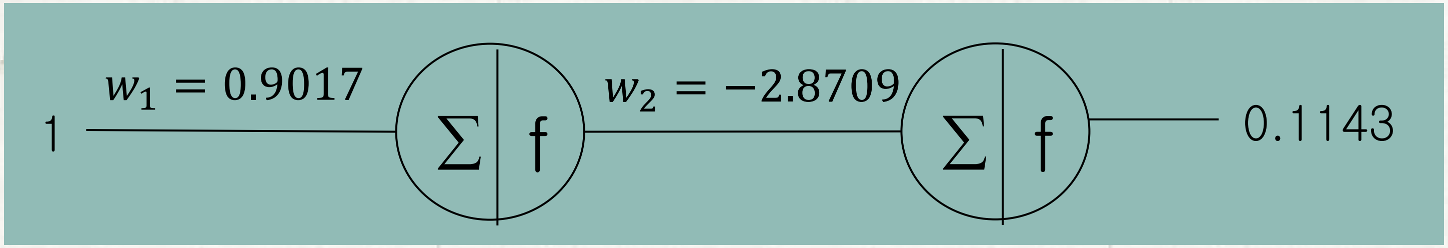
1



2



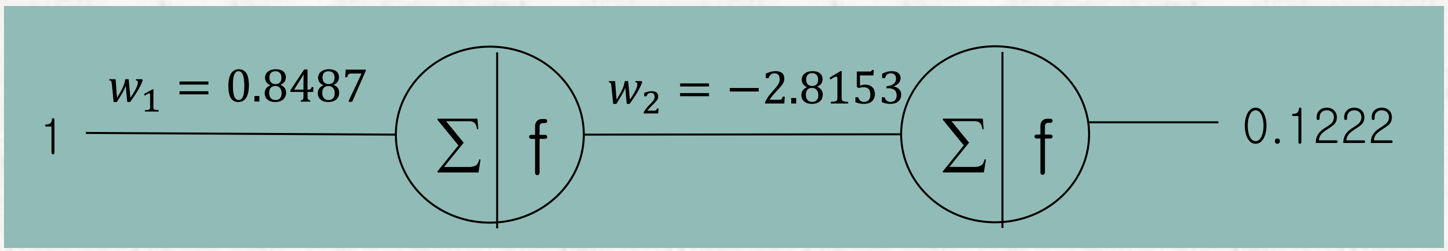
3



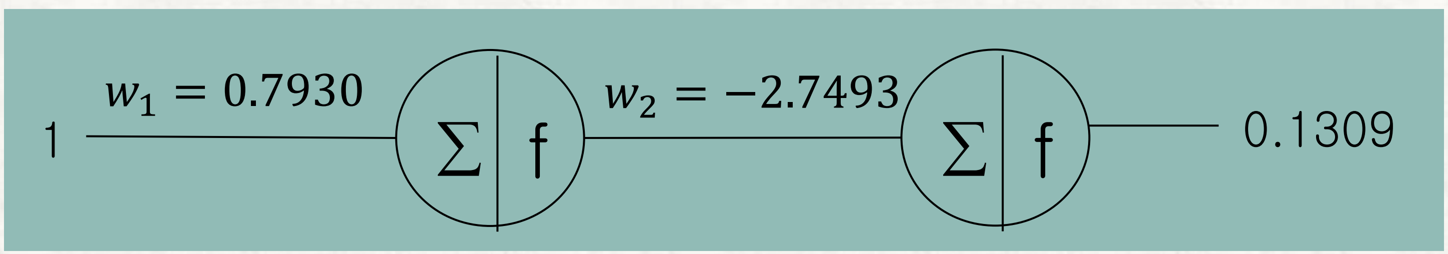
Simple Examples

- Training of a Simple Neural Network

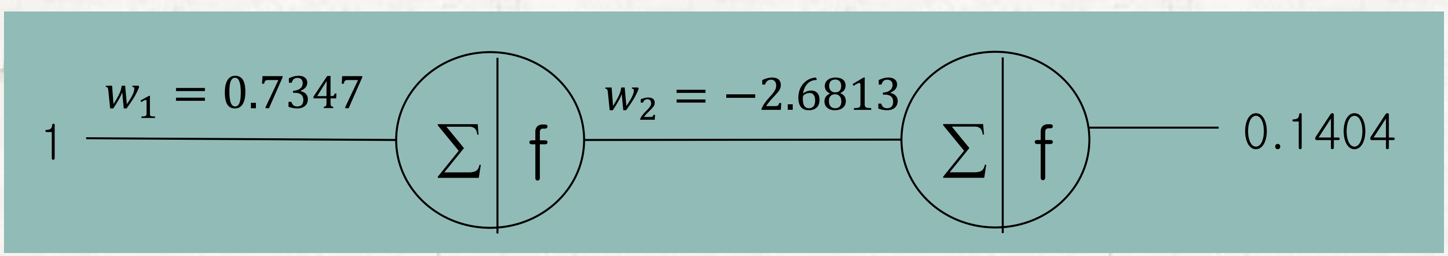
3



5

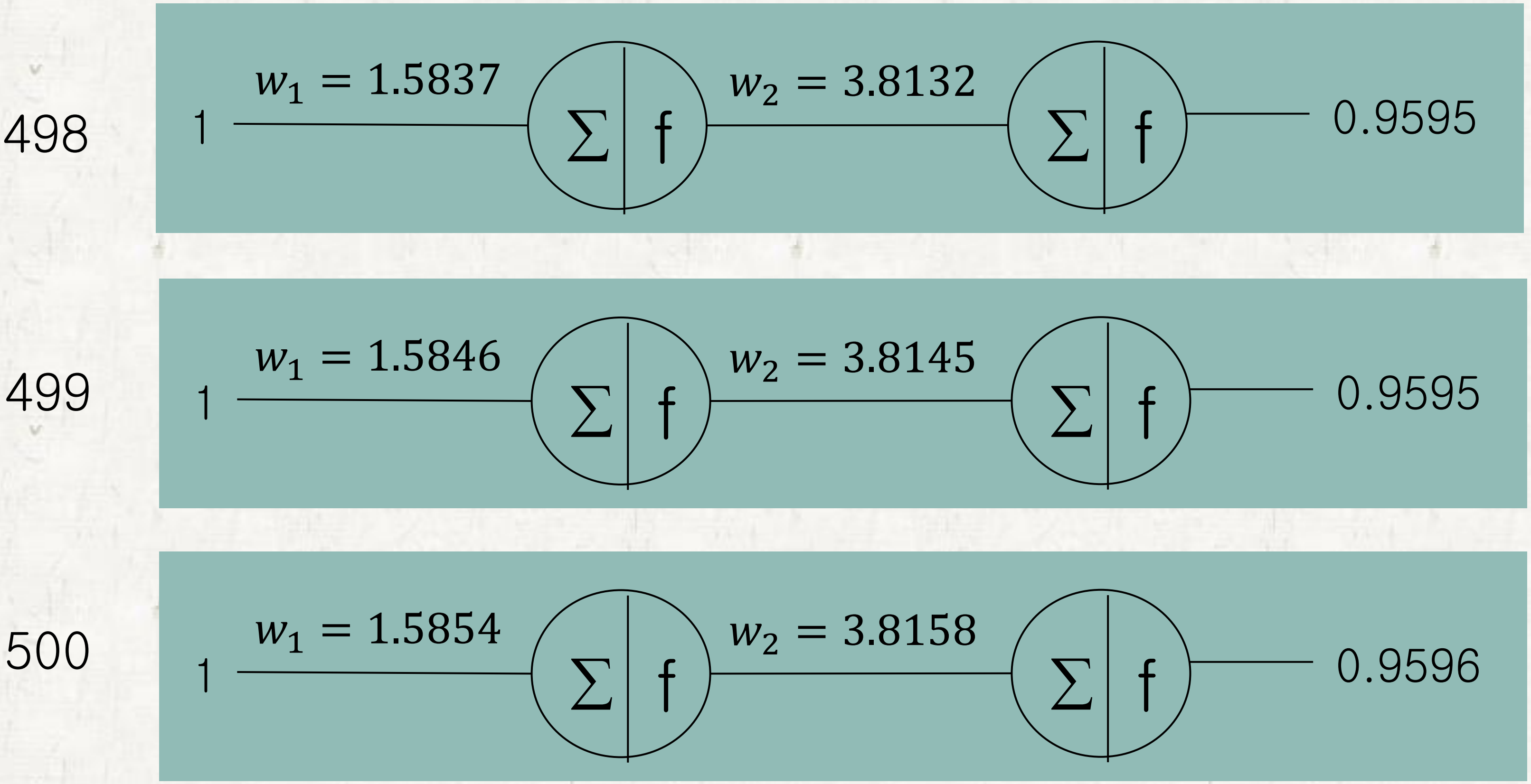


6



Simple Examples

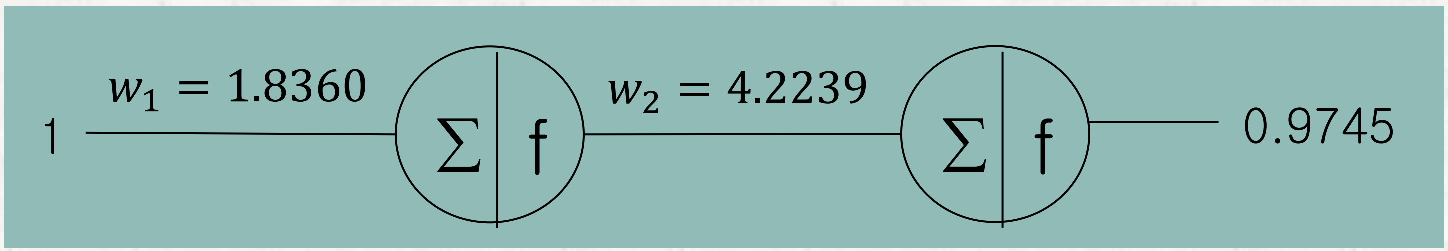
- Training of a Simple Neural Network



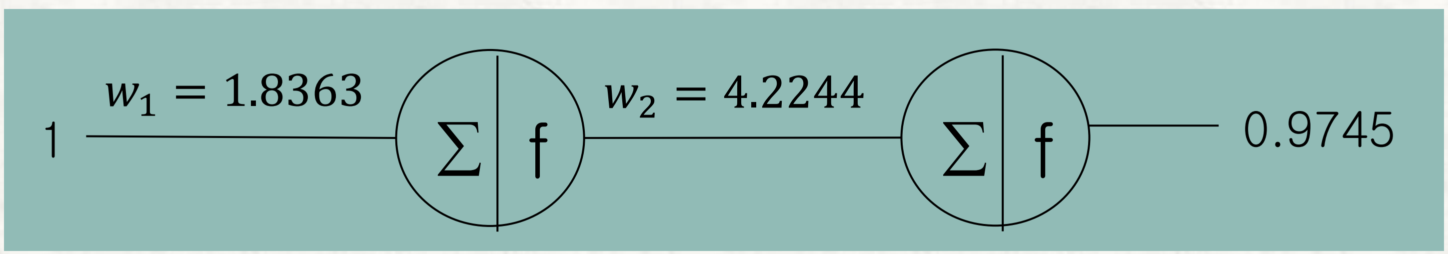
Simple Examples

- Training of a Simple Neural Network

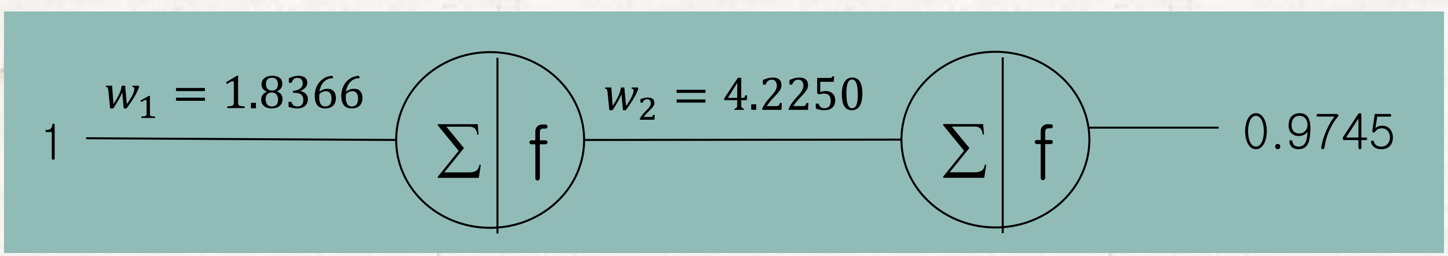
998



999

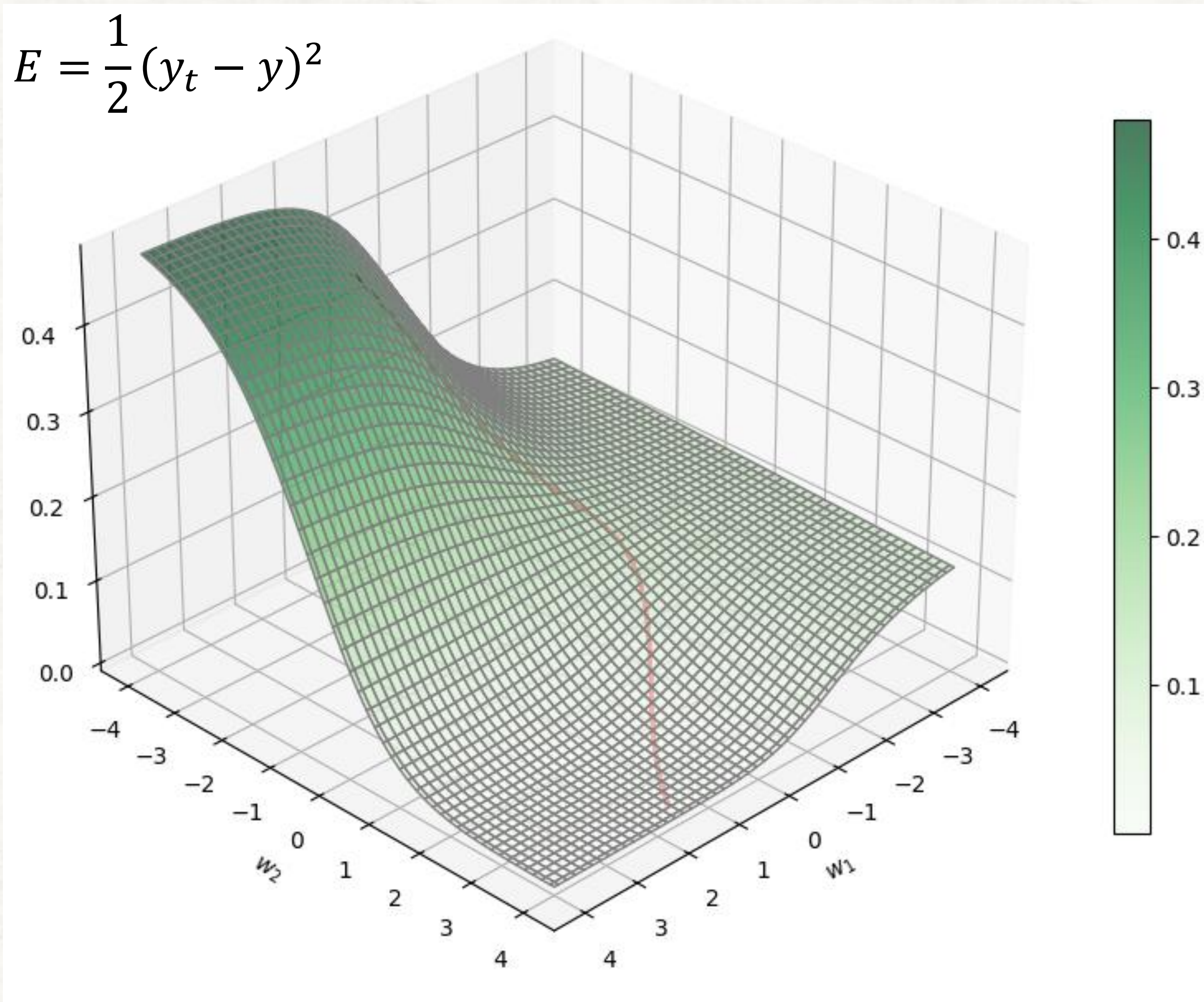


1000



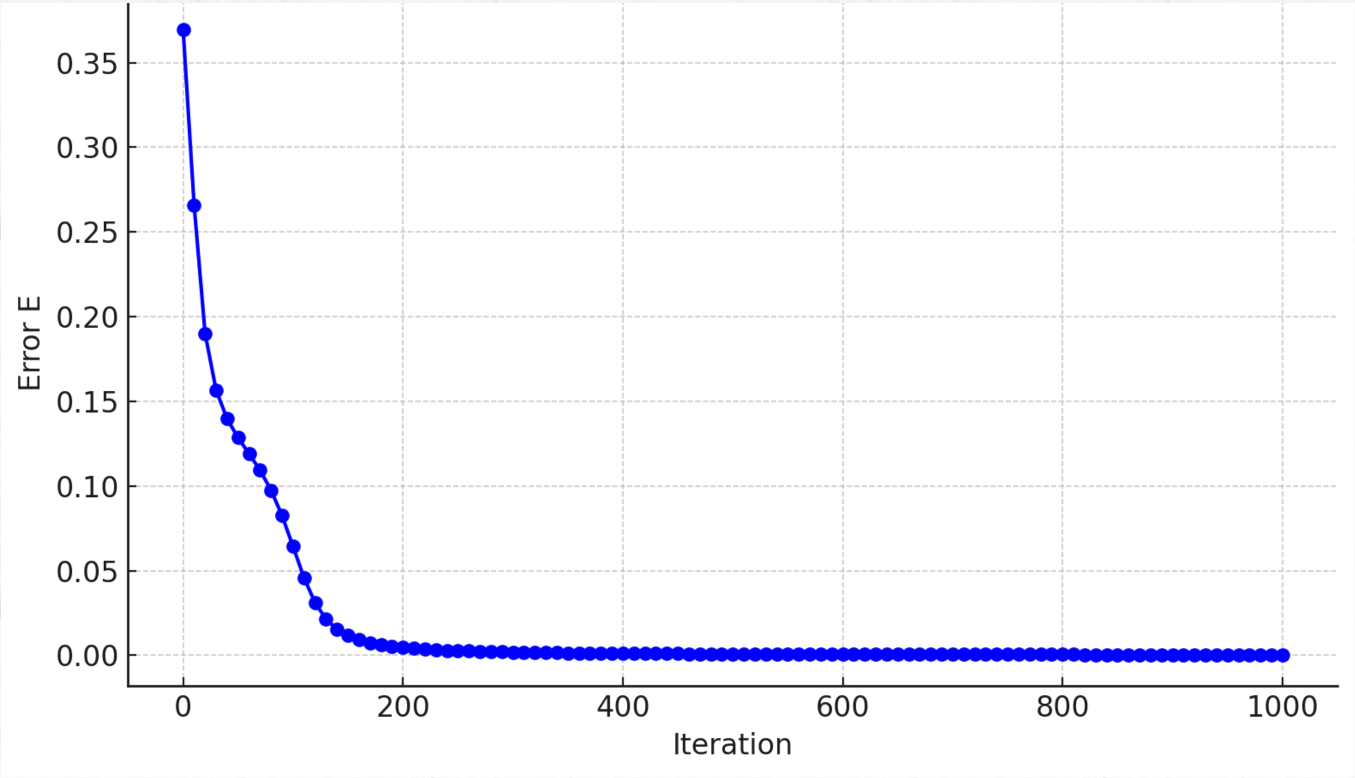
Simple Examples

- Training of a Simple Neural Network



Simple Examples

- Training of a Simple Neural Network
  - Learning Curve

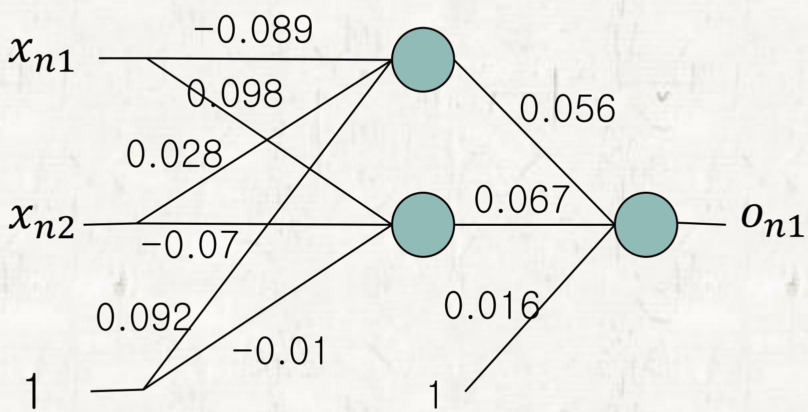


Example of Error Back Propagation

- Example : XOR

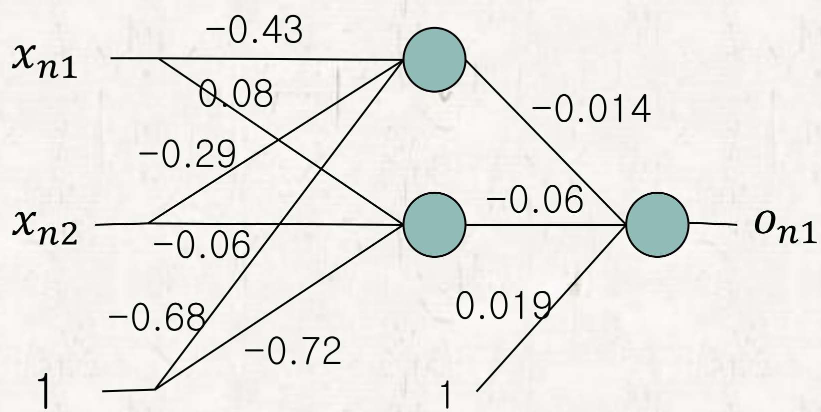
Iteration : 0

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



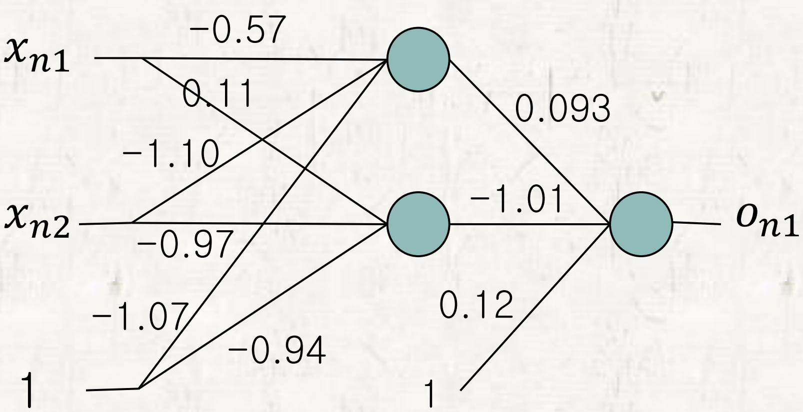
J.H. Lee, Dept. of Software, Sungkyunkwan Univ.

Example of Error Back Propagation

- Example : XOR

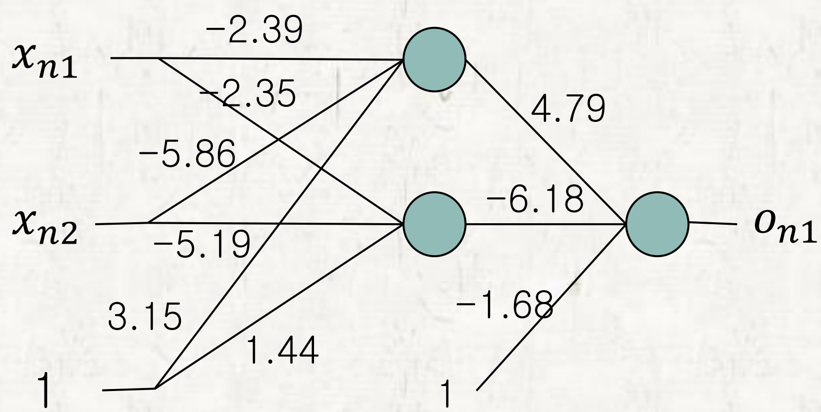
Iteration : 2000

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



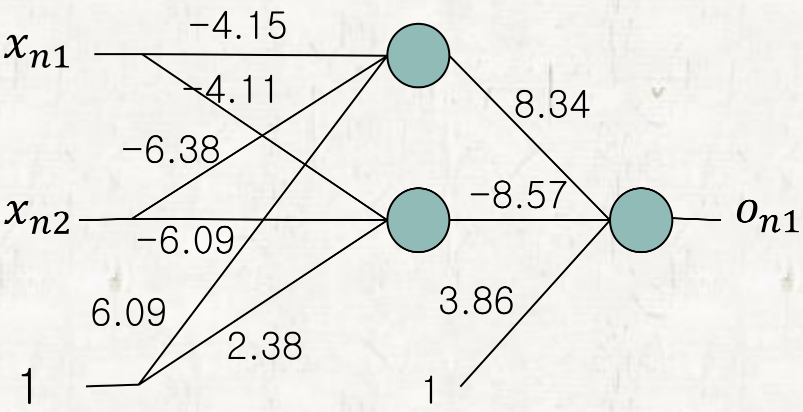
J.H. Lee, Dept. of Software, Sungkyunkwan Univ.

Example of Error Back Propagation

- Example : XOR

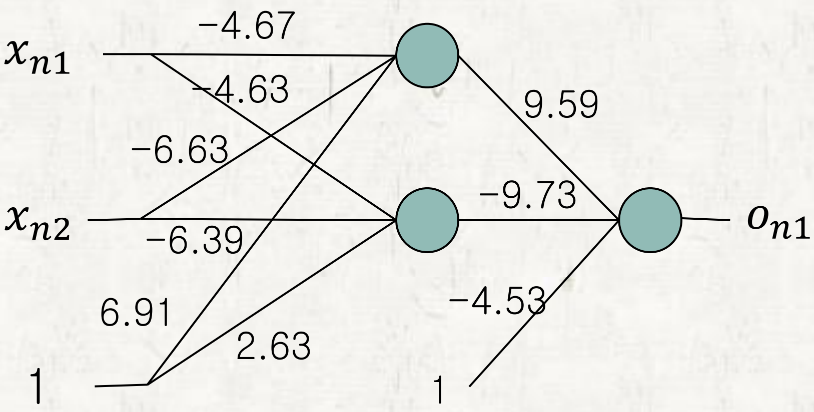
Iteration : 5000

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

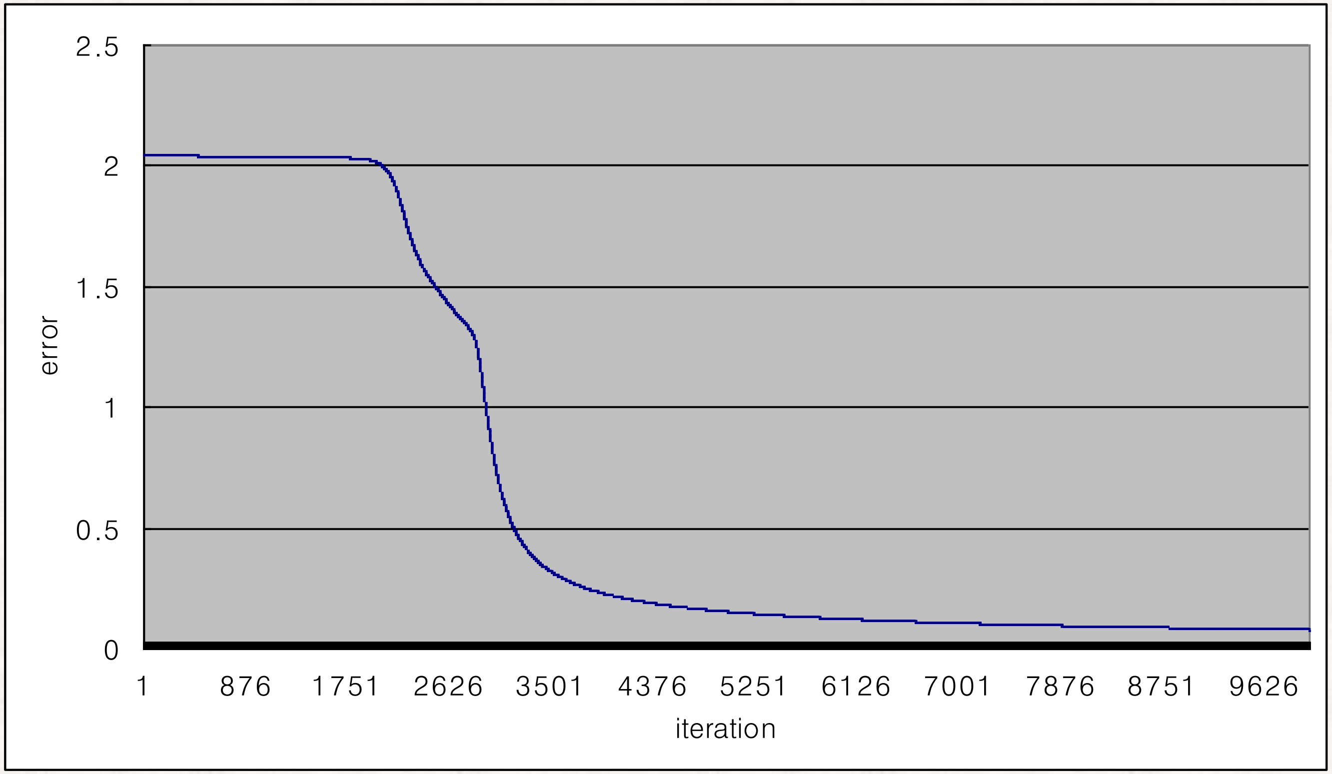
x_{n1}	x_{n2}	t_{n1}	o_{n1}
1	1	0	0.02
1	0	1	0.98
0	1	1	0.98
0	0	0	0.02



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

Example of Error Back Propagation

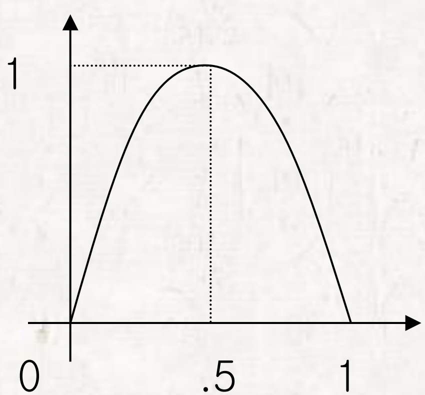
- Example : XOR
  - Error graph

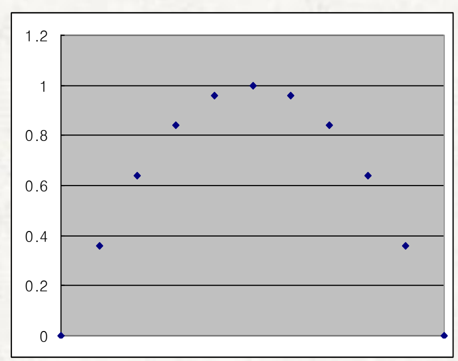


Example of Error Back Propagation

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

$f(x) = 4x*(1-x)$

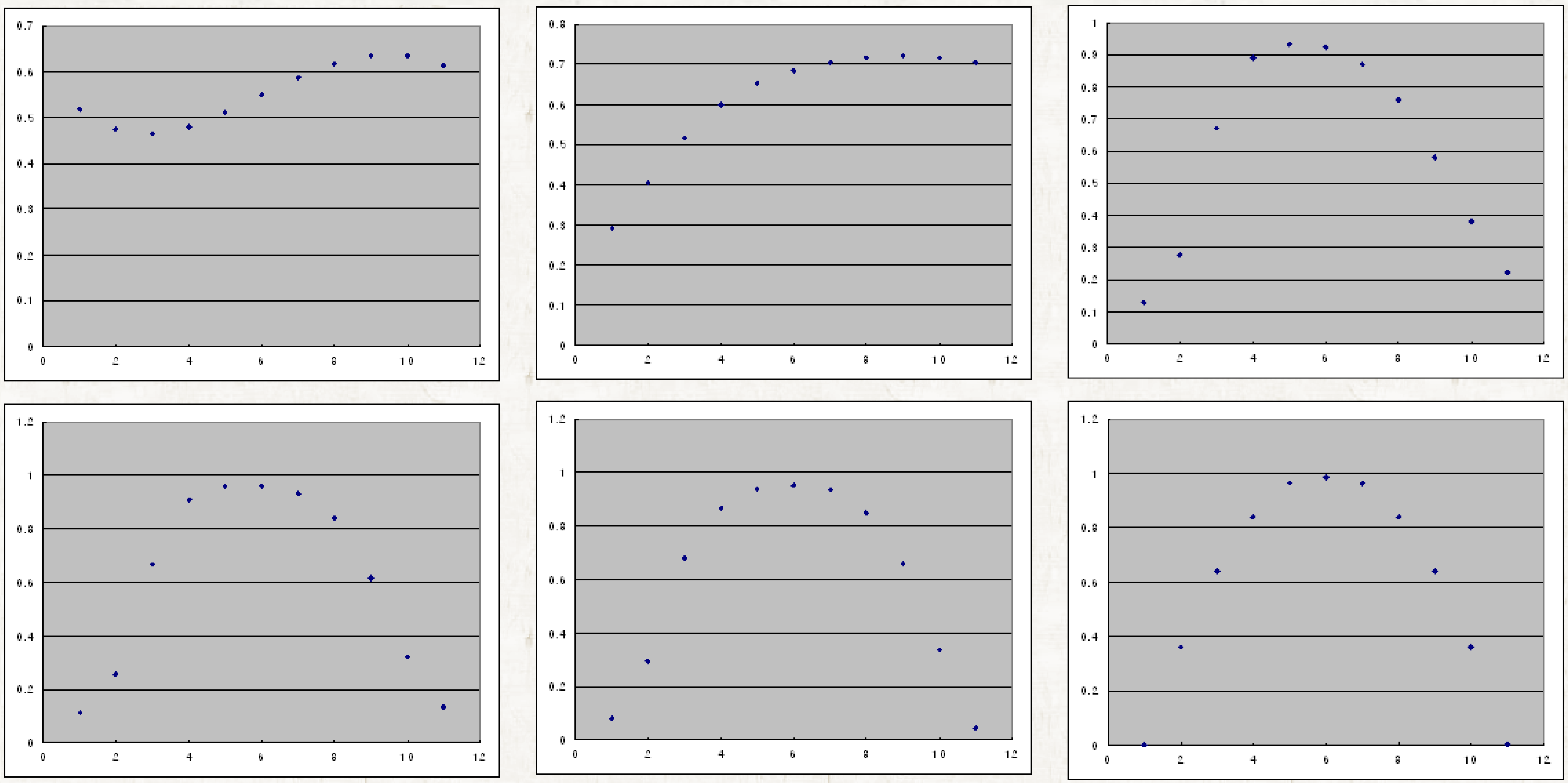




Input	Output
0.00	0.00
0.10	0.36
0.20	0.64
0.30	0.84
0.40	0.96
0.50	1.00
0.60	0.96
0.70	0.84
0.80	0.64
0.90	0.36
1.00	0.00

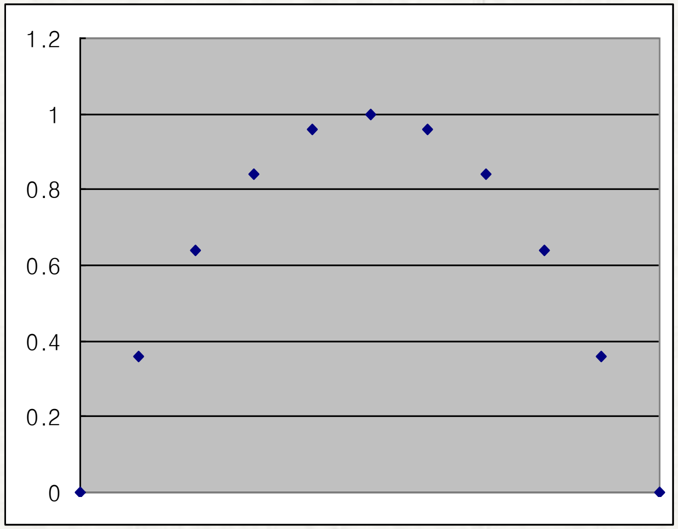
Example of Error Back Propagation

- Example2

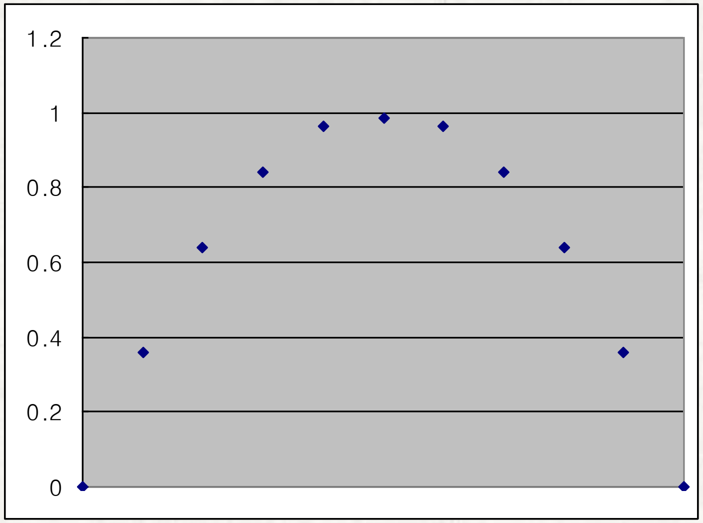


Generalization and Overfitting (1)

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



Training data

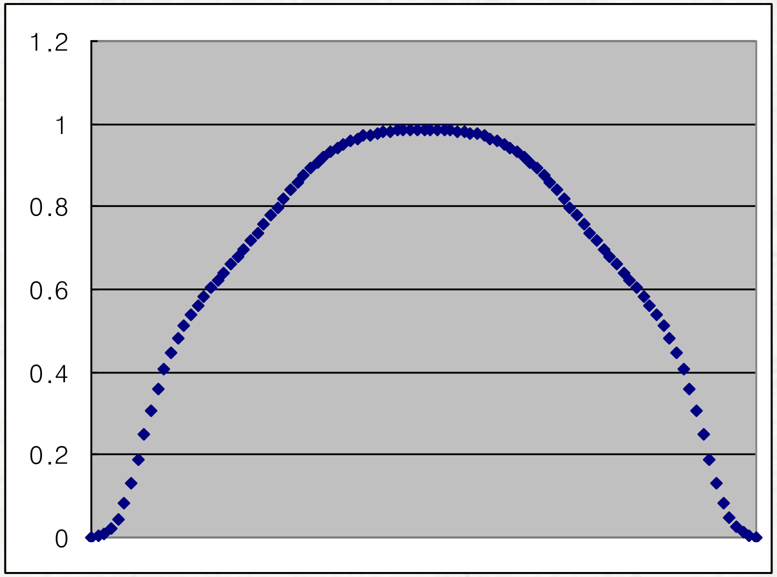


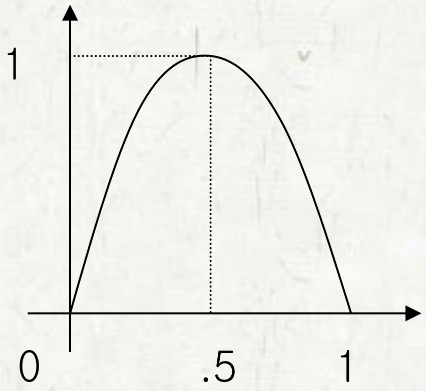
Training result

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

Generalization and Overfitting (2)

- Yes, NNs generalize what they have learned

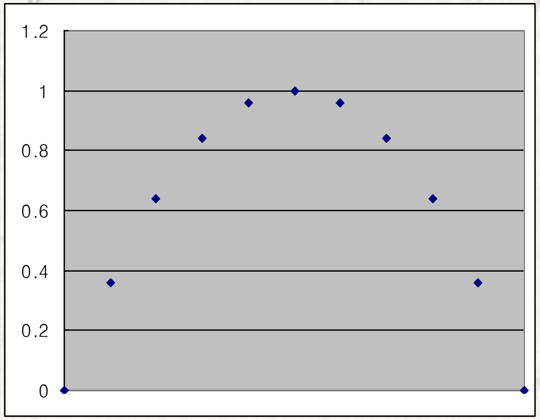




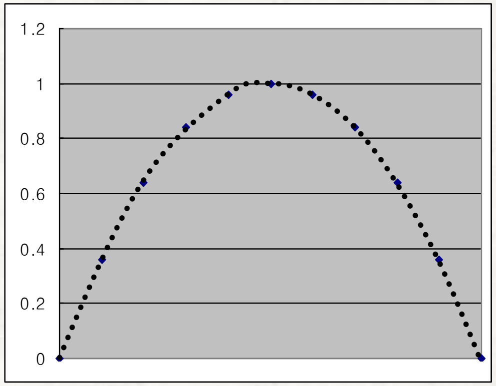
J.H. Lee, Dept. of Software, Sungkyunkwan Univ.

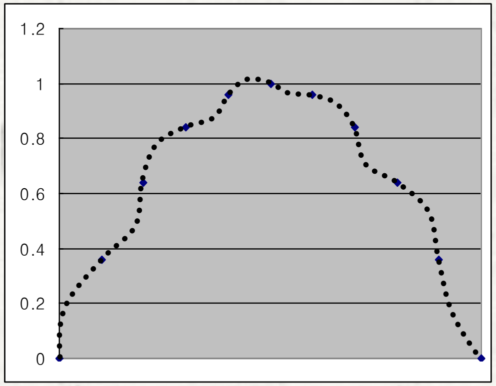
Generalization and Overfitting (3)

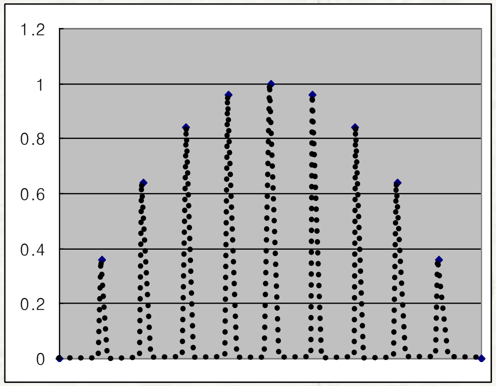
- Which one is better?



Training data

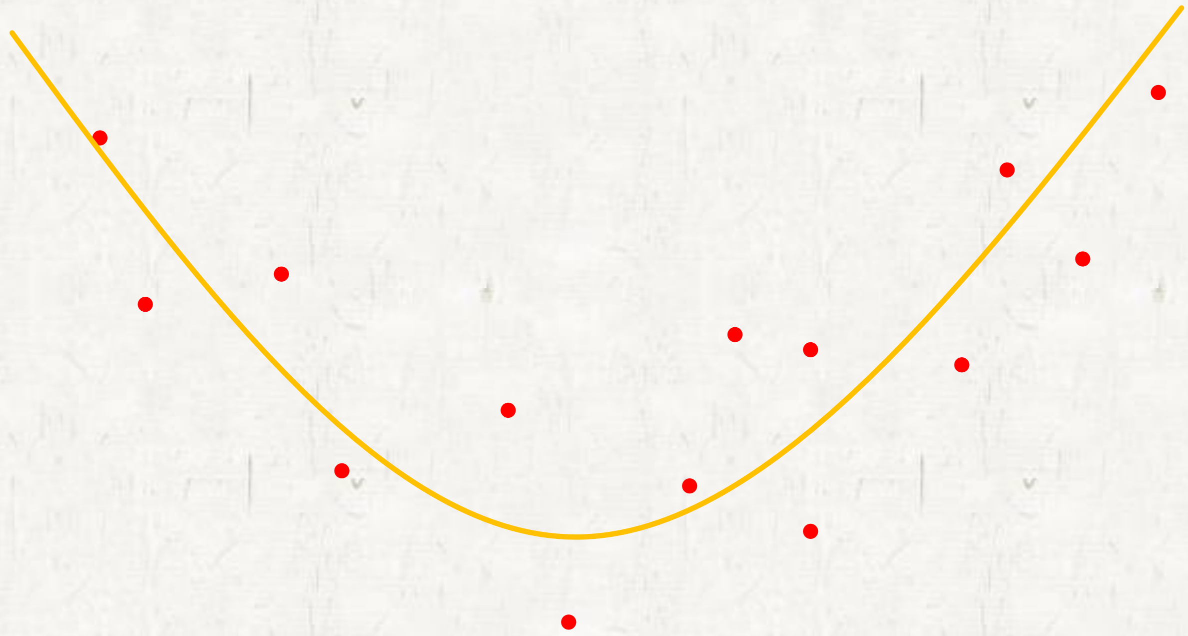






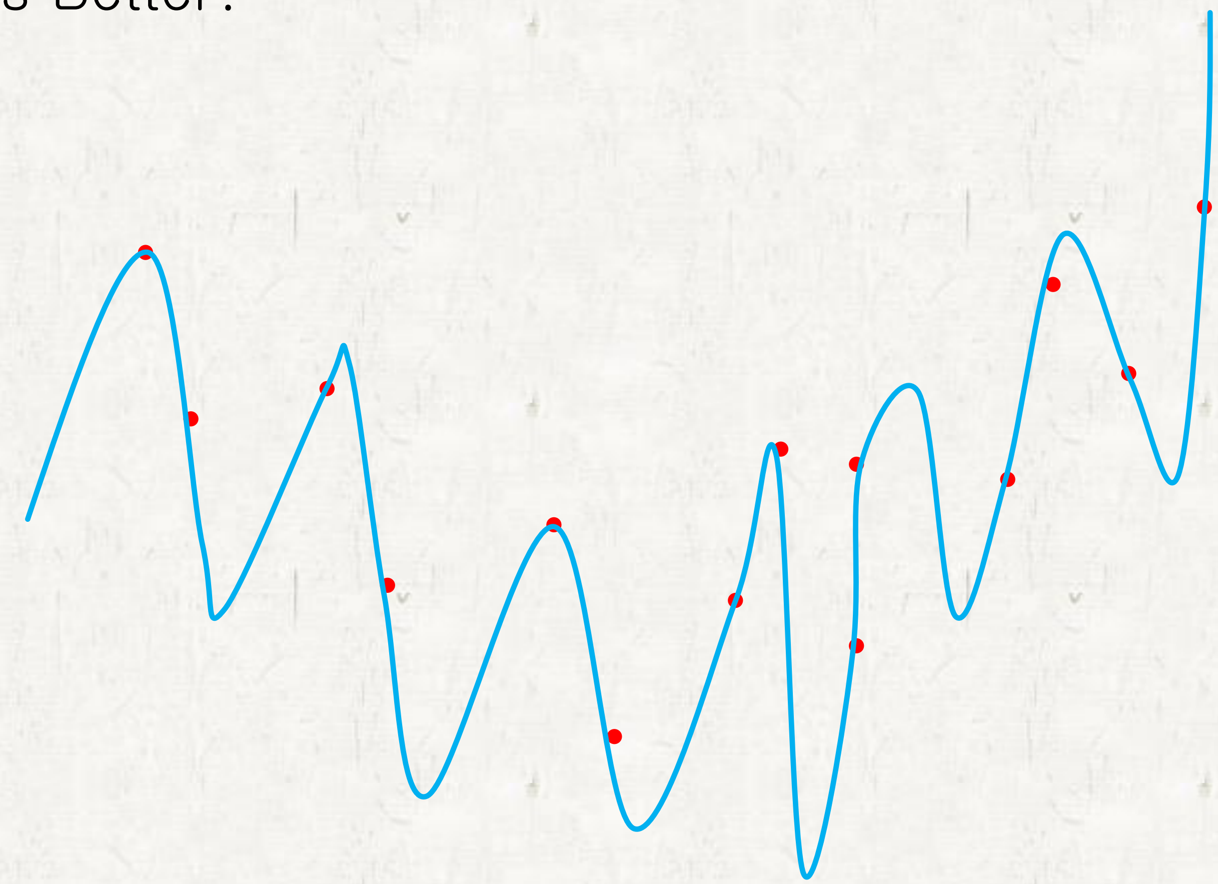
Generalization and Overfitting (4)

- Which is Better?

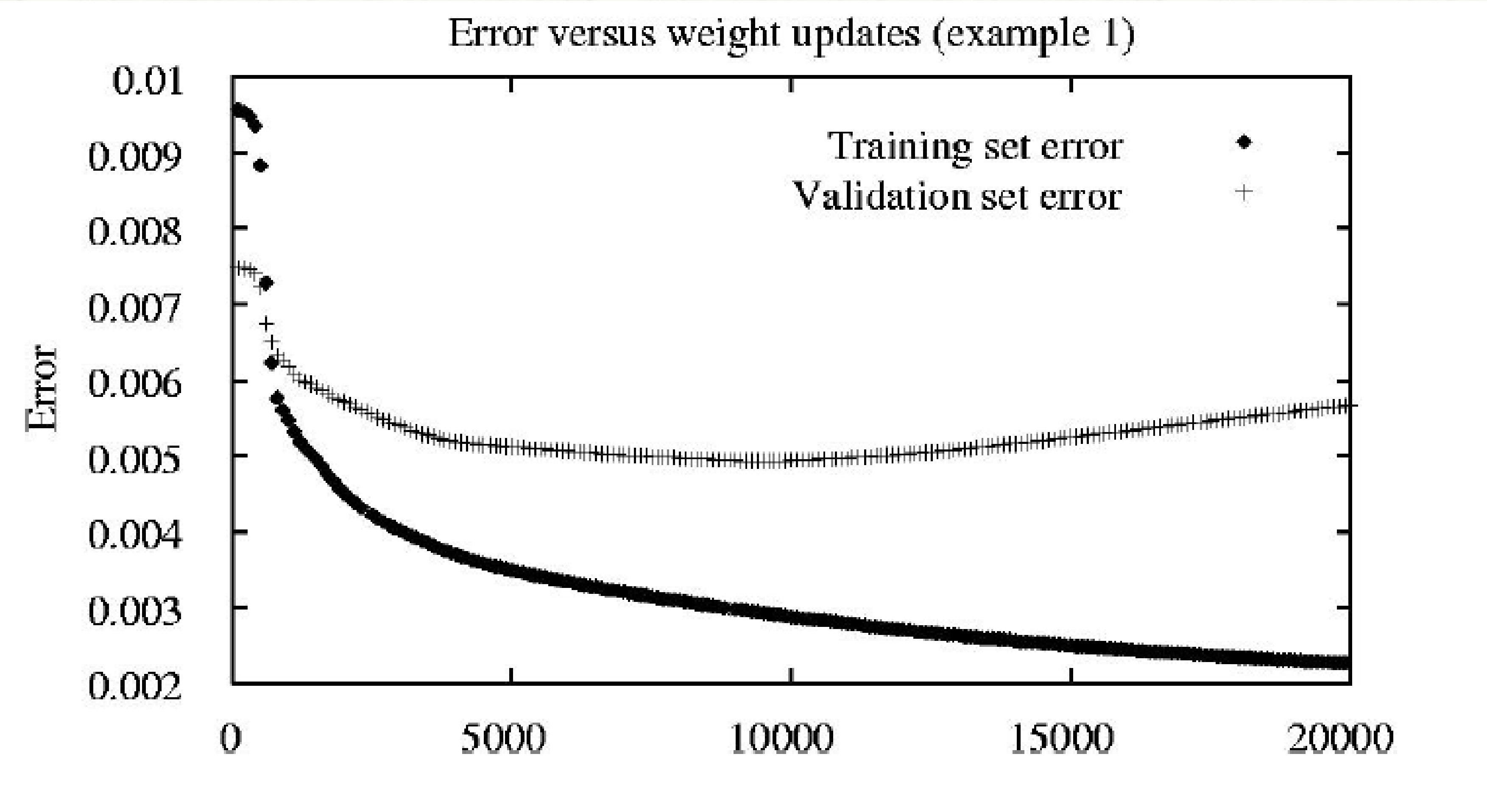


Generalization and Overfitting (5)

- Which is Better?

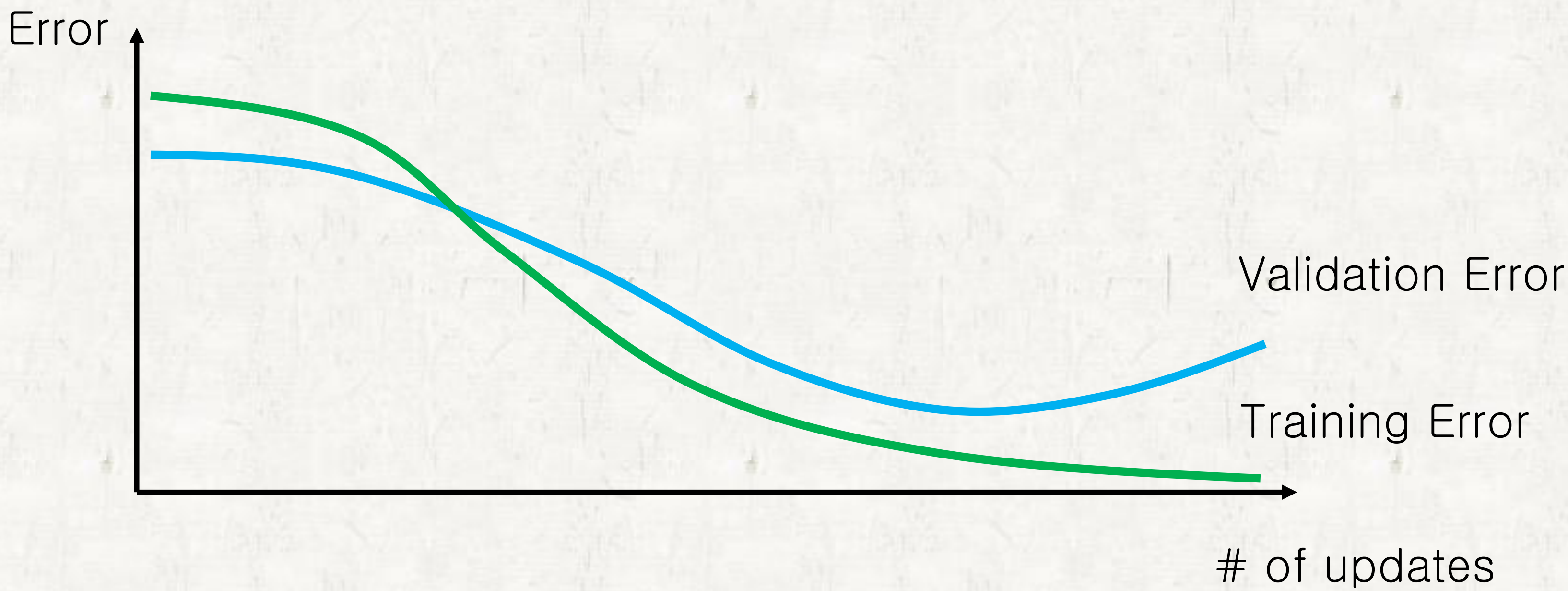


Generalization and Overfitting (6)



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