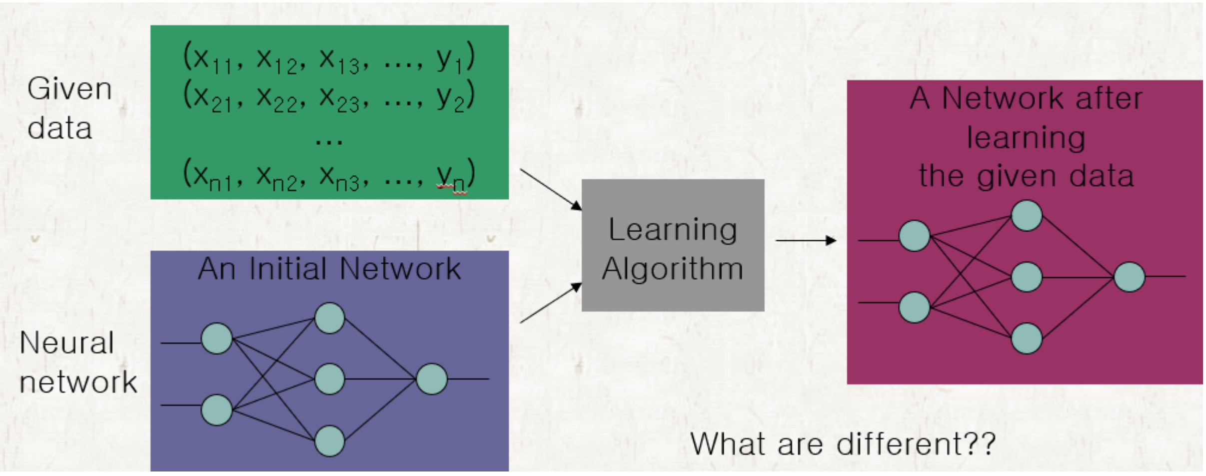
Neural Networks-Learning Algorithm(Back Propagation)

Learning Algorithm(1)

Preparation for Learning

- Given input -output data of the target function to learn
- Given structure of network(#of nodes in hidden layers)
- **Randomly initialized weights**



학습할때마다 w값은 다르다!!

Learning Algorithm(2)

Basic Idea of Learning

- Find weights w=(w1,w2,...,wn) so that

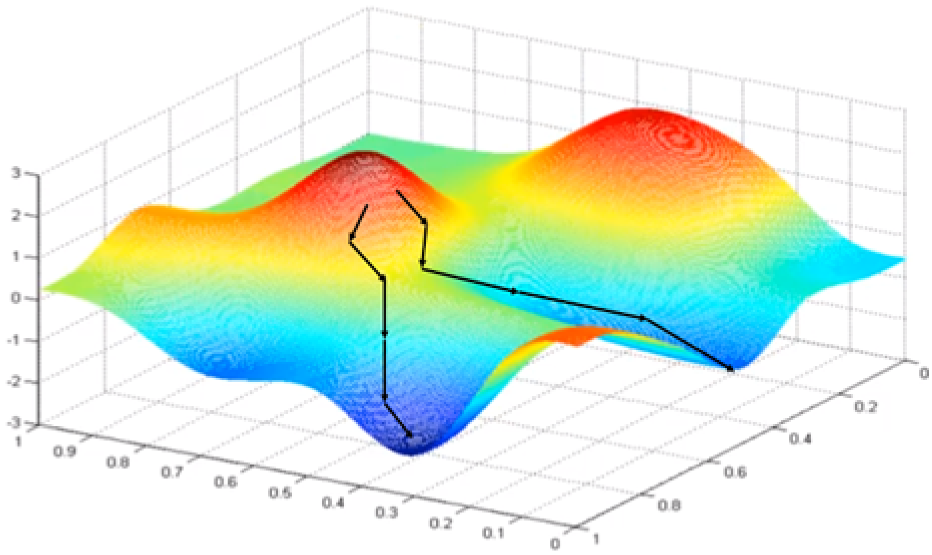
  NN(w,t)=t for all (x,t)

- Find weights w=(w1,w2,...,wn) which minimize Error function

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

Learning Algorithm(3)

Learning with Gradient Descend Method



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

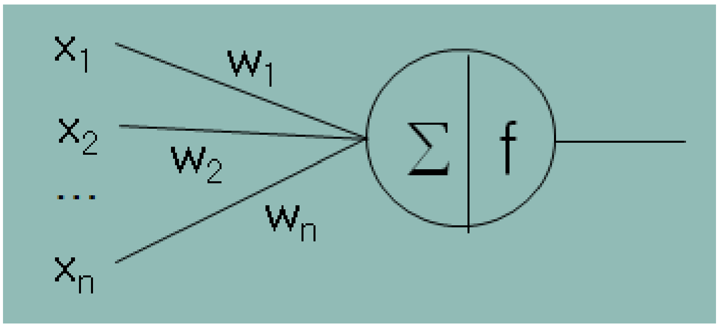
#Gradient descent method는 추가적으로 다룰 예정

Error Back Propagation(1)

Error Back Propagation

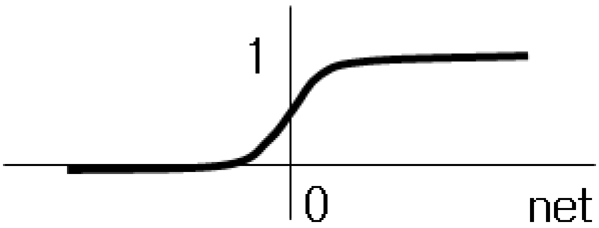
- ANN learning algorithm based on gradient descent method
- Using derivatives to change the weights so that the error is minimized

- Hard limit(step function) is not differentiable→ Sigmoid

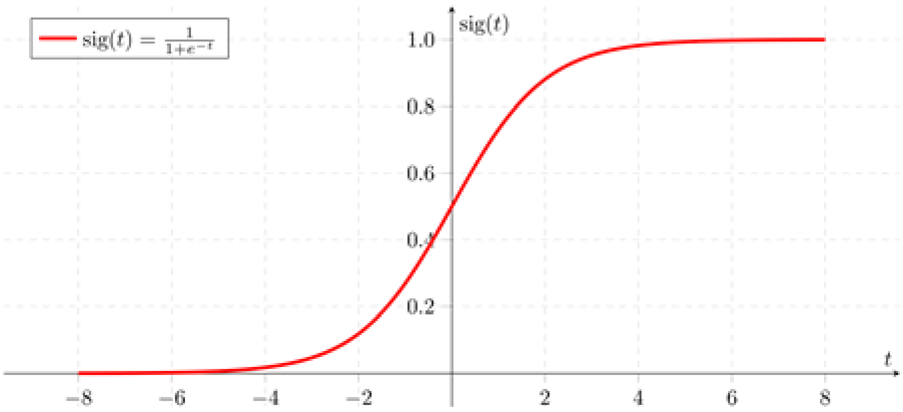


Gradient descent method를 사용하여 w값을 update하는건데 미분해서 0이나오면 update의미가 없다.

net=x1w1+x2w2+x3w3+...+xnwn



Sigmoid function



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

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

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

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

$$= y(1-y)$$

Error Back Propagation(3)

Basic Idea

- Given input-target pairs and output of NN

$D_1=(x_{11}, x_{12}, \dots, x_{1d}, t_{11}, t_{12}, \dots, t_{1m})$ $(o_{11}, o_{12}, \dots, o_{1m})$

$D_2=(x_{21}, x_{22}, \dots, x_{2d}, t_{21}, t_{22}, \dots, t_{2m})$ $(o_{21}, o_{22}, \dots, o_{2m})$

…

$D_N=(\underbrace{x_{N1}, x_{N2}, \dots, x_{Nd}}_{\text{inputs}}, \underbrace{t_{N1}, t_{N2}, \dots, t_{Nm}}_{\text{targets}})$ $\underbrace{(o_{N1}, o_{N2}, \dots, o_{Nm})}_{\text{Outputs of NN}}$

- Minimize the error

$$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w}) \quad \text{where} \quad E_n(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{m}(t_{nk} - o_{nk})^2$$

Error func에있는 아래첨자 n은 데이터개수다.

- Remember

$$\frac{\partial E}{\partial w} = \frac{\partial}{\partial w}\sum_{d=1}^{N} E_d = \sum_{d=1}^{N}\frac{\partial}{\partial w}E_d$$

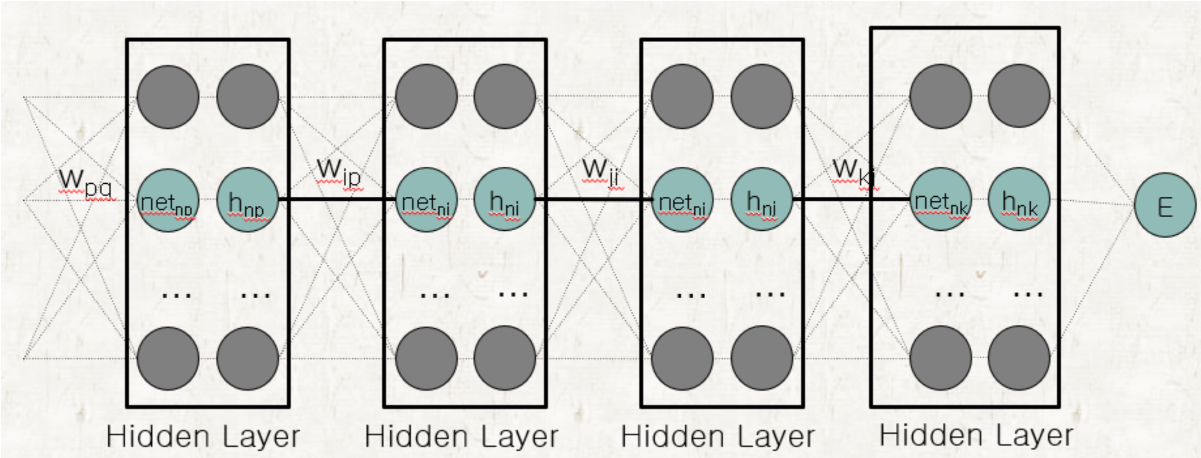
- So, we need to evaluate :

$$\frac{\partial}{\partial w}E_d$$

Algorithm(1)

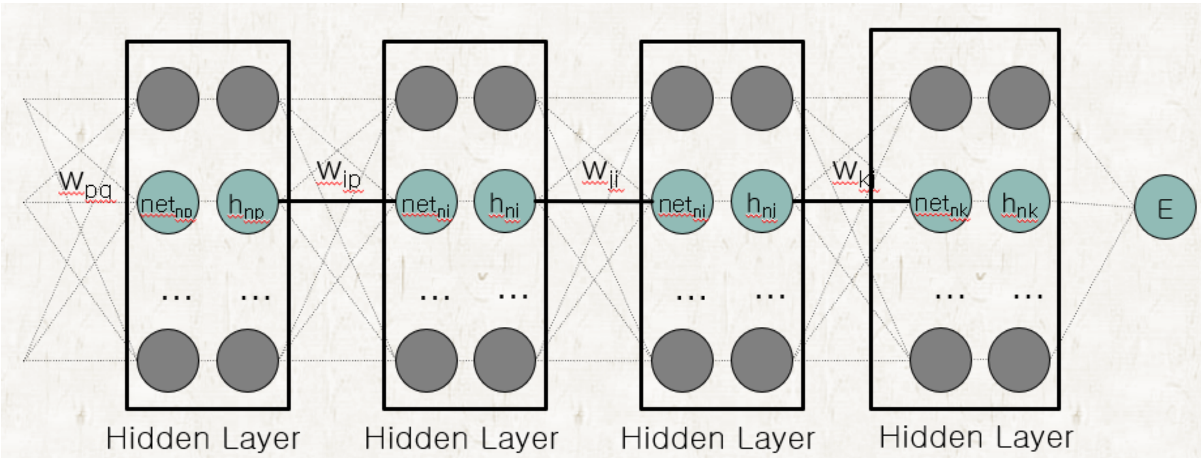
Weights between deep layers

- For Dn=(xn1,xn2,...xnd,tn1,tn2,...,tnm)



Algorithm(2)

Weights between deep layers

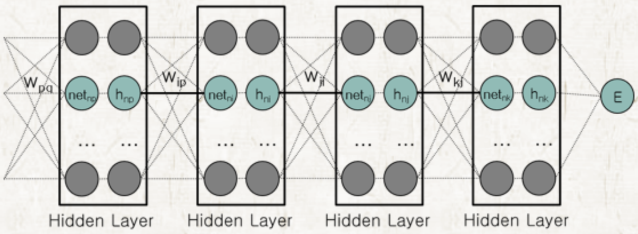


$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial net_{nk}} \frac{\partial net_{nk}}{\partial w_{kj}} = \delta_{nk} h_{nj} \qquad \delta_{nk} = \frac{\partial E}{\partial net_{nk}}$$

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial net_{nj}} \frac{\partial net_{nj}}{\partial w_{ji}} = \delta_{nj} h_{ni} \qquad \delta_{nj} = \frac{\partial E}{\partial net_{nj}}$$

$$\frac{\partial E}{\partial w_{ip}} = \frac{\partial E}{\partial net_{ni}} \frac{\partial net_{ni}}{\partial w_{ip}} = \delta_{ni} h_{np} \qquad \delta_{ni} = \frac{\partial E}{\partial net_{ni}}$$

Algorithm(3)

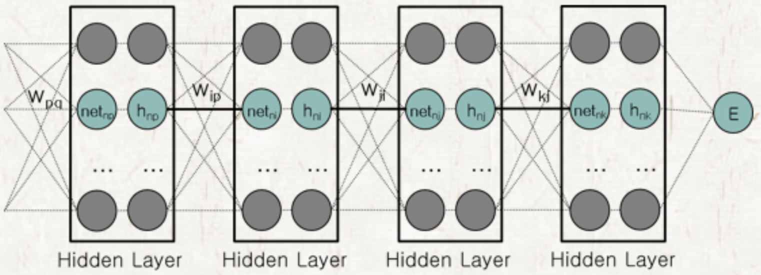


$$\delta_{nk} = \frac{\partial E}{\partial net_{nk}} = \frac{\partial E}{\partial h_{nk}} \frac{\partial h_{nk}}{\partial net_{nk}}$$

$$\delta_{nj} = \frac{\partial E}{\partial net_{nj}} = \boxed{\frac{\partial E}{\partial h_{nj}}} \frac{\partial h_{nj}}{\partial net_{nj}} = \boxed{\left(\sum_{k=1}^{K} \frac{\partial E}{\partial net_{nk}} \frac{\partial net_{nk}}{\partial h_{nj}} \right)} \frac{\partial h_{nj}}{\partial net_{nj}} = \left(\sum_{k=1}^{K} \delta_{nk} w_{kj} \right) \frac{\partial h_{nj}}{\partial net_{nj}}$$

$$\delta_{ni} = \frac{\partial E}{\partial net_{ni}} = \boxed{\frac{\partial E}{\partial h_{ni}}} \frac{\partial h_{ni}}{\partial net_{ni}} = \boxed{\left(\sum_{j=1}^{J} \frac{\partial E}{\partial net_{nj}} \frac{\partial net_{nj}}{\partial h_{ni}} \right)} \frac{\partial h_{ni}}{\partial net_{ni}} = \left(\sum_{j=1}^{J} \delta_{nj} w_{ji} \right) \frac{\partial h_{ni}}{\partial net_{ni}}$$

Algoritm(4)

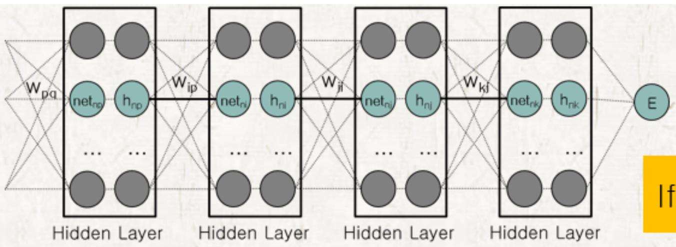


$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial net_{nk}}\frac{\partial net_{nk}}{\partial w_{kj}} = \delta_{nk} h_{nj} \qquad \delta_{nk} = \frac{\partial E}{\partial h_{nk}}\frac{\partial h_{nk}}{\partial net_{nk}}$$

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial net_{nj}}\frac{\partial net_{nj}}{\partial w_{ji}} = \delta_{nj} h_{ni} \qquad \delta_{nj} = \left(\sum_{k=1}^{K} \delta_{nk} w_{kj}\right)\frac{\partial h_{nj}}{\partial net_{nj}}$$

$$\frac{\partial E}{\partial w_{ip}} = \frac{\partial E}{\partial net_{ni}}\frac{\partial net_{ni}}{\partial w_{ip}} = \delta_{ni} h_{np} \qquad \delta_{ni} = \left(\sum_{j=1}^{J} \delta_{nj} w_{ji}\right)\frac{\partial h_{ni}}{\partial net_{ni}}$$

Algoritm(5)



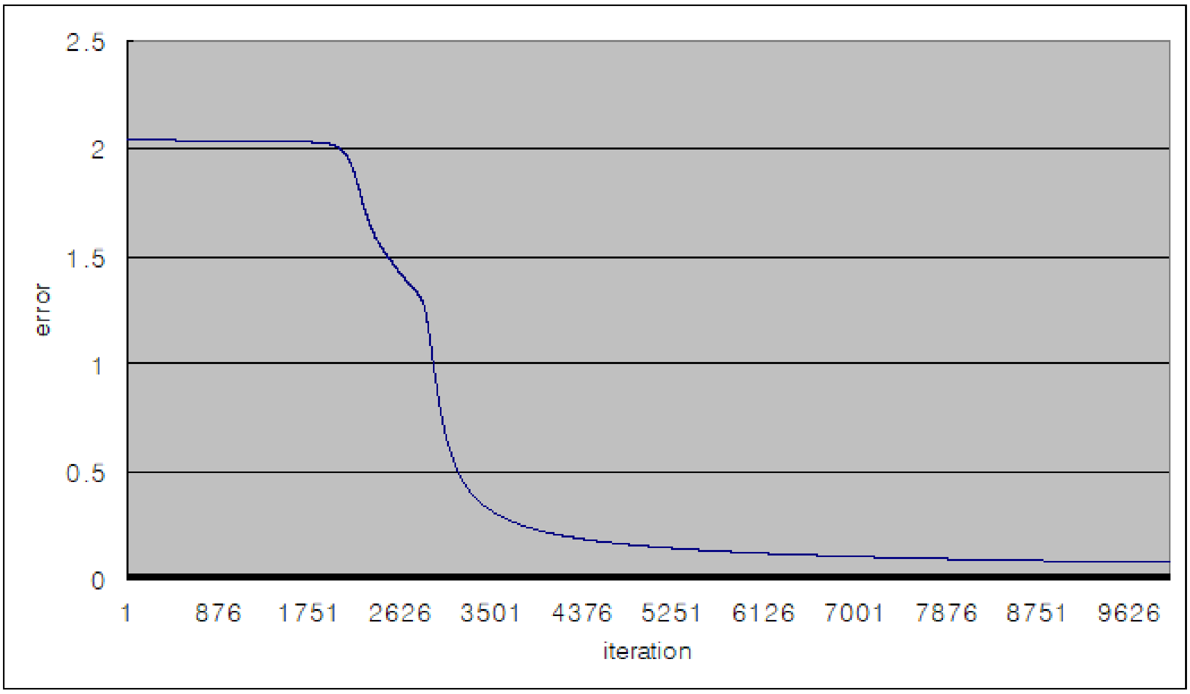
If $h = Sigmoid(net)$

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial net_{nk}}\frac{\partial net_{nk}}{\partial w_{kj}} = \delta_{nk} h_{nj} \qquad \delta_{nk} = \frac{\partial E}{\partial h_{nk}}\frac{\partial h_{nk}}{\partial net_{nk}} = -(t_n - h_{nk})h_{nk}(1 - h_{nk})$$

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial net_{nj}}\frac{\partial net_{nj}}{\partial w_{ji}} = \delta_{nj} h_{ni} \qquad \delta_{nj} = \left(\sum_{k=1}^{K} \delta_{nk} w_{kj}\right)\frac{\partial h_{nj}}{\partial net_{nj}} = \left(\sum_{k=1}^{K} \delta_{nk} w_{kj}\right) h_{nj}(1 - h_{nj})$$

$$\frac{\partial E}{\partial w_{ip}} = \frac{\partial E}{\partial net_{ni}}\frac{\partial net_{ni}}{\partial w_{ip}} = \delta_{ni} h_{np} \qquad \delta_{ni} = \left(\sum_{j=1}^{J} \delta_{nj} w_{ji}\right)\frac{\partial h_{ni}}{\partial net_{ni}} = \left(\sum_{j=1}^{J} \delta_{nj} w_{ji}\right) h_{ni}(1 - h_{ni})$$

Example of Error Back Propagation



Neural Network의 사용목적

>unknown data를 잘 맞추기 위해서

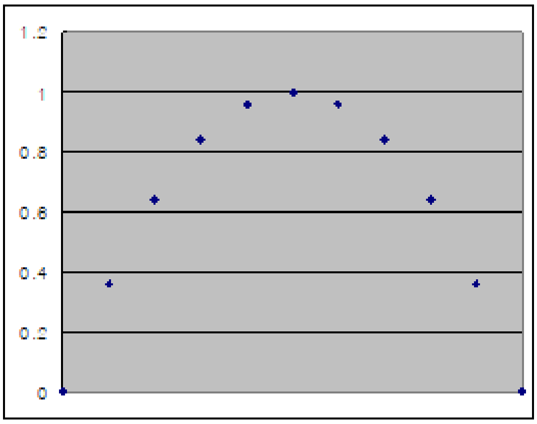
Error가 작다고 무조건 좋은게 아니다(∵overfitting)

overfitting이란? 말그대로 오버피팅 너무 과하게 training data를 학습한것—>training data의 노이즈 까지 학습하는경우

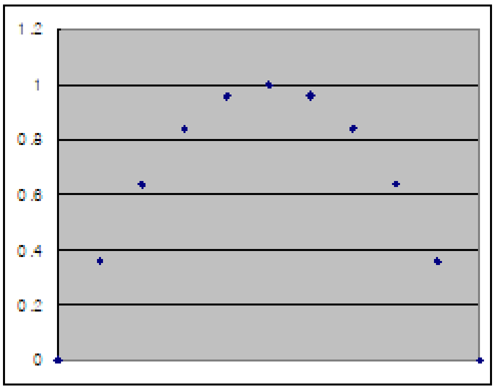
Neural Network의 사용목적은 unknown data를 잘 맞추기 위해서였다. 즉, training data의 패턴을 잘 학습해야 가능하다. —>Generalization

NN은 unknown data를 어떻게 맞추나?

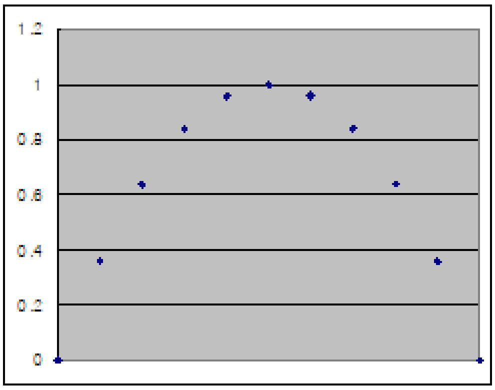
>> interpolation을 통해(non-linear)



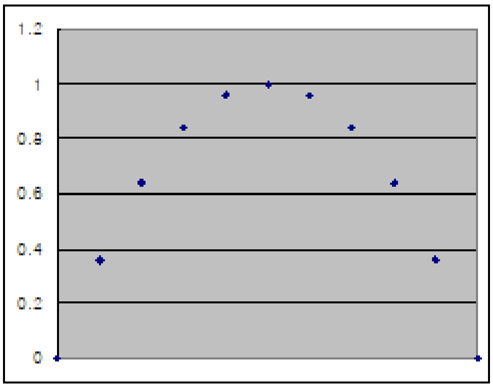
이것은 Training data



1번



2번



Which one is better??

1번이 좋다@@

3번은 overfitting

Generalization and Overfitting

1. Find the optimal number of neurons
2. Find the optimal number of training iterations
3. Use regularization
4. Use more training data>>현실적으로 문제를 해결할 정도로 많은 data를 모으기는 힘들다.