Tensorflow for NNs:
http://playground.tensorflow.org/

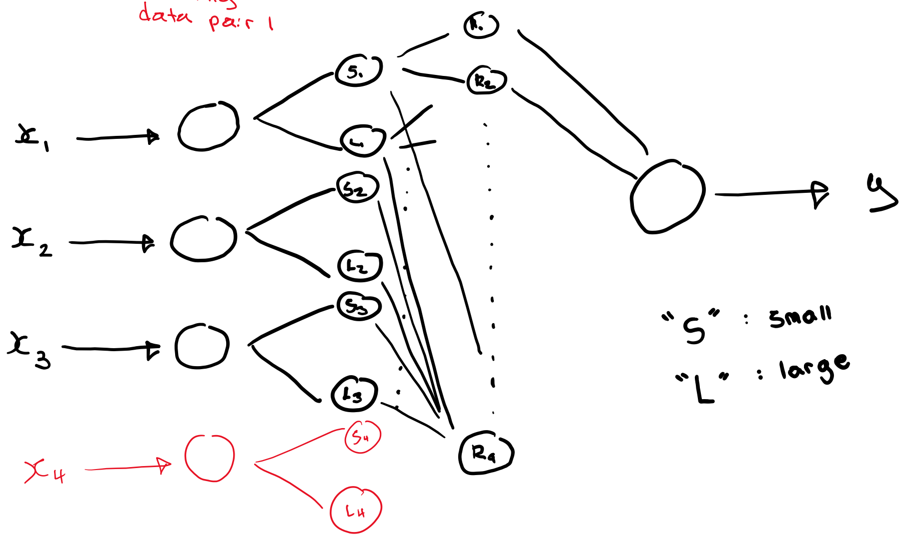
Mackey-Glass data:
https://www.mathworks.com/matlabcentral/fileexchange/24390-mackey-glass-time-series-generator

training data pair 2

For example:

If we have x_1 , x_2 , x_3 , x_4 , x_5 , x_6 , x_7 , x_8 , …

training data pair 1



One -step-ahead prediction:

$$\{(x_1, x_2, x_3); x_4\}$$
$$\{(x_2, x_3, x_4); x_5\}$$
$$\{(x_3, x_4, x_5); x_6\}$$
$$\vdots$$

TSK-1:

If $(x_1 \text{ is } s_1)$ and $(x_2 \text{ is } s_2)$ and $(x_3 \text{ is } L_3)$ then
$y_1 = a_1 x_1 + b_1 x_2 + c_1 x_3 + d_1$

Then each rule has 4 linear parameters to be updated, and we have 9 rules.

Then in total, we have $4 \text{ x } 9 = 36$ linear parameters to be updated.

Non-linear parameters are related to the ‘small’ and ‘large’ membership function parameters (assume each is a sigmoid function, and has two parameters):
$2 \text{ x } 6 = 12$

In total, there are 48 training data pairs.

Training data pairs is at least 5 times the number of non-linear parameters:

$$5 \text{ x } 48 = 240$$

If you have 16 rules with 4 inputs, each having 2 membership functions:
16 rules: linear

R_1: if (x_1 is s_1) and (x_2 is s_2) and (x_3 is L_3) and (x_4 is s_4)

Then: $y_1 = a_1x_1 + b_1x_2 + c_1x_3 + d_1x_4 + g_1$

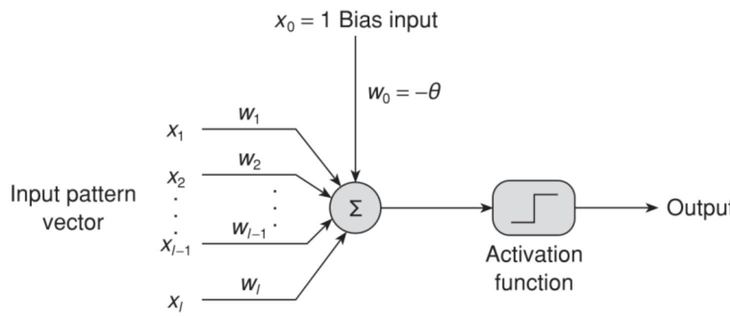
How many linear parameters for each rule?
$5 \text{ x } 16 = 80$

How many non-linear function parameters?
$8 \text{ x } 2 = 16$

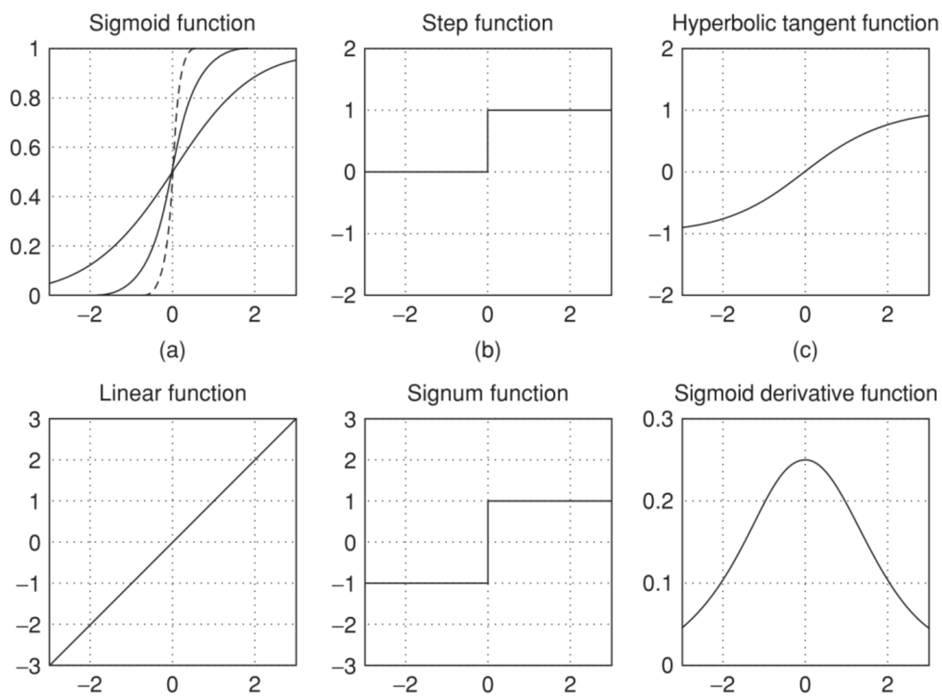
4.4 Connectionist Modeling

- MP Modeling

$$O = f(x_1 w_1 + x_2 w_2 + \cdots + x_l w_l - \theta) = f\left(\sum_{i=1}^{l} x_i w_i - \theta\right)$$



Consider the general activation functions:



(a) (b) (c)
(d) (e) (f)

- Perceptron

Training algorithm (DIRECTLY FROM TEXTBOOK):

1. Initialize weights and thresholds to small random values.
2. Choose an input-output pattern $\left(x^{(k)}, t^{(k)}\right)$ from the training data.
3. Compute the network's actual output $o^{(k)} = f\left(\sum_{i=1}^{l} w_i x_i^{(k)} - \theta\right)$.
4. Adjust the weights and bias according to the Perceptron learning rule:
   $$\Delta w_i = \eta\left[t^{(k)} - o^{(k)}\right] x_i^{(k)}$$
   And:
   $$\Delta\theta = -\eta\left[t^{(k)} - o^{(k)}\right]$$
   Where $\eta \in [0, 1]$ is the Perceptron's learning rate.
   If f is the signum function, this becomes equivalent to:
   $$\Delta w_i = \begin{cases} 2\eta t^{(k)} x_i^{(k)} & ;\ if\ t^{(k)} \neq o^{(k)} \\ 0 & ;\ \text{otherwise} \end{cases}$$
   And:
   $$\Delta\theta = \begin{cases} -2\eta t^{(k)} & ;\ if\ t^{(k)} \neq o^{(k)} \\ 0 & ;\ \text{otherwise} \end{cases}$$
5. If a whole epoch is complete, then pass to the following step; otherwise go to Step 2.
6. If the weights (and bias) reached steady state ($\Delta w_i \approx 0$) through the whole epoch, then stop the learning; otherwise go through one more epoch starting from Step 2.

Training algorithm (CLASS NOTES):

$$\vec{x}^{(k)} = \left\{x_1^{(k)}, \quad x_2^{(k)}, \dots, \quad x_l^{(k)}\right\}^T$$

Where:
$t^{(k)} = \text{target, desired output}$

$$O^{(k)} = f\left(\sum_{i=1}^{l} w_i^{(k)} x_i^{(k)} - \theta\right)$$

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} \underbrace{+ \Delta\vec{w}}_{\text{update}}$$

$$\Delta\vec{w} = \eta\left(t^{(k)} - O^{(k)}\right)\vec{x}^{(k)}$$

If $f(\cdot) \sim signum\ fxn \begin{cases} 1 & ; \quad \text{input} > 0 \\ 0 & ; \quad \text{otherwise} \\ -1 & ; \quad \text{input} < 0 \end{cases}$

If $t^{(k)} = O^{(k)}$ (then we don't need to update anything)

$$\Delta\vec{w} = \begin{cases} 0 & ; \quad t^{(k)} = O^{(k)} \\ 2\eta t^{(k)}\vec{x}^{(k)} & ; \quad \text{otherwise} \end{cases}$$

Otherwise, if $t^{(k)} \neq O^{(k)}$

If $t^{(k)} = +1$, then $O^{(k)} = -1 = -t^{(k)}$
If $t^{(k)} = -1$, then $O^{(k)} = +1 = -t^{(k)}$

Similarly:

$$\theta^{(k+1)} = \theta^{(k)} + \Delta\theta$$

$$\Delta\theta = -\eta(t^{(k)} - O^{(k)})$$

If AF is $signum\ fxn$
If $t^{(k)} = O^{(k)}$
Otherwise $t^{(k)} \neq O^{(k)}$, $O^{(k)} = -t^{(k)}$

$$\Delta\theta = \begin{cases} 0 & ; \quad t^{(k)} = O^{(k)} \\ 2\eta t^{(k)} & ; \quad \text{otherwise} \end{cases}$$

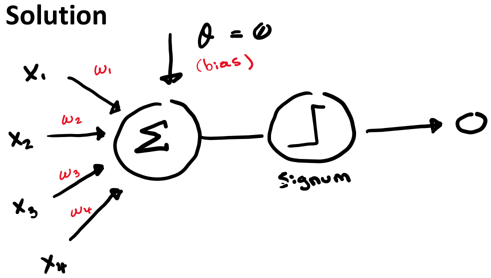
Example 4.1 (Book 1)

Train a network using the following set of input and desired output training vectors:

$$x^{(1)} = [1, -2, 0, 1]^T; \;\; t^{(1)} = -1$$
$$x^{(2)} = [0, 1.5, -0.5, -1]^T; \;\; t^{(2)} = -1$$
$$x^{(3)} = [-1, 1, 0.5, -1]^T; \;\; t^{(3)} = +1$$

With initial weight vector $w^{(1)} = [1, -1, 0, 0.5]^T$, learn $\eta = 0.1$

Solution



$\eta = 0.1$
$w^{(1)} = [1, -1, 0, 0.5]^T$

$O = f(x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4 - \theta) = f(\vec{w}^T \vec{x} - \theta)$
(but θ is 0 here)

<u>Epoch 1:</u>

$$\vec{x}^{(1)} = [1, -2, 0, -1]^T, \qquad t^{(1)} = -1$$

$$O^{(1)} = sgn\left({\vec{w}^{(1)}}^T \vec{x}\right)$$
$$= sgn\left([1 \quad -1 \quad 0 \quad 0.5] \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix} \right)$$
$$= sgn(1 + 2 + 0 - 0.5) = sgn(2.5)$$

$\vec{w}^{(2)} = \vec{w}^{(1)} + \Delta\vec{w}$

$$\vec{w}^{(2)} = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix} + 2(0.1)(-1) \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}$$

$$\vec{w}^{(2)} = \begin{bmatrix} 0.8 \\ -0.6 \\ 0 \\ -0.7 \end{bmatrix}$$

Input the 2nd training data pair $\vec{x}^{(2)}$:

$$O^{(2)} = f\left(\vec{w}^{(2)^T}\vec{x}^{(2)}\right)$$

$$= sgn\left(\begin{bmatrix}0.8 & -0.6 & 0 & 0.7\end{bmatrix}\begin{bmatrix}0\\1.5\\-0.5\\-1\end{bmatrix}\right)$$

$$= sgn(0 - 0.9 + 0 - 0.7)$$
$$= sgn(-1.6)$$
$$= -1 = t^{(2)} = -1$$

$$\vec{w}^{(3)} = \vec{w}^{(2)} + \Delta\vec{w} = \vec{w}^{(2)} = \begin{bmatrix}0.8\\-0.6\\0\\0.7\end{bmatrix}$$

Input the 3rd training data pair $\vec{x}^{(3)}$:

$$\vec{x}^{(3)} = [-1, 1, 0.5, -1]^T, \qquad t^{(3)} = 1$$

$$O^{(3)} = f\left(\vec{w}^{(3)^T}\vec{x}^{(3)}\right)$$

$$= sgn\left(\begin{bmatrix}0.8 & -0.6 & 0 & 0.7\end{bmatrix}\begin{bmatrix}-1\\1\\0.5\\-1\end{bmatrix}\right)$$

$$= sgn(0.8 - 0.6 + 0 - 0.7)$$
$$= sgn(-2.1)$$
$$= -1 \neq t^{(3)} = 1$$

$$\vec{w}^{(4)} = \vec{w}^{(3)} + \Delta\vec{w}$$

$$\vec{w}^{(4)} = \begin{bmatrix}0.8\\-0.6\\0\\-0.7\end{bmatrix} + \underbrace{2(0.1)(1)\begin{bmatrix}-1\\1\\0.5\\-1\end{bmatrix}}_{\Delta W}$$

$$\vec{w}^{(4)} = \begin{bmatrix}0.6\\-0.4\\0.1\\0.5\end{bmatrix}$$

<u>End of epoch 1</u>

Since $\Delta w \neq 0$, proceed to epoch 2…

$$\begin{array}{ccc} \left(\vec{x}^{(1)},\ t^{(1)}\right) & \left(\vec{x}^{(2)},\ t^{(2)}\right) & \left(\vec{x}^{(3)},\ t^{(3)}\right) \\ \downarrow & \downarrow & \downarrow \\ \left(\vec{x}^{(4)},\ t^{(4)}\right) & \left(\vec{x}^{(5)},\ t^{(5)}\right) & \left(\vec{x}^{(6)},\ t^{(6)}\right) \end{array}$$

$\vec{w}^{(4)} = [0.6, -0.4, 0.1, 0.5]^T$

$O^{(4)} = f\left({\vec{w}^{(4)}}^T \vec{x}^{(4)}\right)$

$$= sgn\left(\begin{bmatrix} 0.6 & -0.4 & 0.1 & 0.5 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}\right)$$

$= sgn(0.6 + 0.8 + 0 - 0.5) =$

$= sgn(0.9) = +1 \neq t^{(4)} = -1$

$$\vec{w}^{(5)} = \vec{w}^{(4)} + 2\eta t^{(4)} \vec{x}^{(4)}$$

$$= \begin{bmatrix} 0.6 \\ -0.4 \\ 0.1 \\ 0.5 \end{bmatrix} + 2(0.1)(-1) \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.4 \\ 0 \\ 0.1 \\ 0.7 \end{bmatrix}$$

$\vec{x}^{(5)} = \vec{x}^{(2)} = [0, 1.5, -0.5, -1]^T$

$$O^{(5)} = f({\vec{w}^{(5)}}^T \vec{x}^{(5)})$$

$$\vdots$$

Epoch 3 (still doesn't meet requirements)

$\vec{w}^{(10)} = [0 \quad 0.4 \quad 0.3 \quad 0.3]^T$

Epoch 4 (still doesn't meet requirements)

$\vec{w}^{(12)} = [-2 \quad 0.3 \quad 0.5 \quad 0.3]^T$

After Epoch 5, we can meet the requirements.

Example 4.2 (Example 2 in Book 1)

Assume $\eta = 0.5$, and there exists two sets of patterns to be classified:

Class 1: Target value -1:

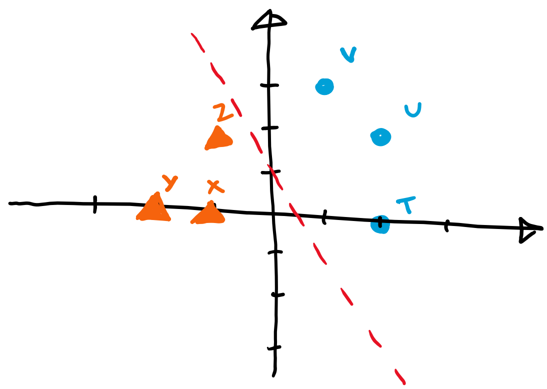
$$T = [2, 0]^T; \quad U = [2, 2]^T; \quad V = [1, 3]^T$$

Class 2: Target value 1:

$$X = [-1, 0]^T; \quad Y = [-2, 0]^T; \quad Z = [-1, 2]^T$$

Solution:

$t^{(1)} = -1$



$$w_1 x_1 + w_2 x_2 - \theta = 0$$

$t^{(2)} = +1$

Assume initial values:

$$w_1 = -1, w_2 = 1, \theta = -1$$
$$(-1)x_1 + (1)x_2 + 1 = 0$$

- Pattern $T = [2, 0]^T$, signum AF

$$0 = sgn(\vec{w}^T \vec{x}) + 1 = sgn\left([-1 \quad 1]\begin{bmatrix}2\\0\end{bmatrix} + 1\right)$$
$$= sgn(-2 + 0) + 1 = -1 = t^{(1)}$$

- Input $\vec{x} = \vec{u} = \begin{bmatrix}2\\2\end{bmatrix}$

$$O = sgn(\vec{w}^T \vec{x} - \theta)$$
$$= sgn\left([-1 \quad 1]\begin{bmatrix}2\\2\end{bmatrix} + 1\right)$$
$$= sgn(-2 + 2 + 1) = +1 \neq t^{(1)}$$
$$t^{(1)} = -1$$

$$\vec{w}^{(2)} = \vec{w}^{(1)} + 2\eta t^{(1)} \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$
$$= \begin{bmatrix} -1 \\ 1 \end{bmatrix} + 2(0.5)(-1) \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ -1 \end{bmatrix}$$

$$\theta^{(2)} = \theta^{(1)} + \Delta\theta$$
$$= -1 + [-2(0.5)(-1)] = 0$$

Boundary function:

$$w_1 x_1 + w_2 x_2 - \theta = 0$$
$$-3x_1 - x_2 = 0$$
$$x_2 = -3x_1$$

- Input $\vec{x} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$

$$O = sgn(\vec{w}^T\ \vec{x} - \theta)$$
$$= sgn\left(\begin{bmatrix} -3 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} - 0 \right)$$
$$= sgn(-3 + 3) = -1 = t^{(1)}$$

$$\vec{w}^{(3)} = \begin{bmatrix} -3 \\ -1 \end{bmatrix}$$

$$\theta^{(3)} = 0$$

- Input $\vec{x} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$

$$O = sgn(\vec{w}^T\ \vec{x} - \theta)$$
$$= sgn\left(\begin{bmatrix} -3 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \end{bmatrix} - 0 \right)$$
$$= sgn(3 + 0) = +1 = t^{(2)}$$

$$\vec{w}^{(4)} = \vec{w}^{(3)} = \begin{bmatrix} -3 \\ -1 \end{bmatrix}$$

$$\theta^{(4)} = 0$$

- Input $\vec{x} = \begin{bmatrix} -2 \\ 0 \end{bmatrix}$

$$O = sgn(\vec{w}^T\ \vec{x} - \theta)$$
$$= sgn\left(\begin{bmatrix} -3 & -1 \end{bmatrix} \begin{bmatrix} -2 \\ 0 \end{bmatrix} - 0 \right)$$
$$= sgn(6 + 0 - 0) = +1 = t^{(2)}$$

$$\vec{w}^{(5)} = \begin{bmatrix} -3 \\ -1 \end{bmatrix}$$

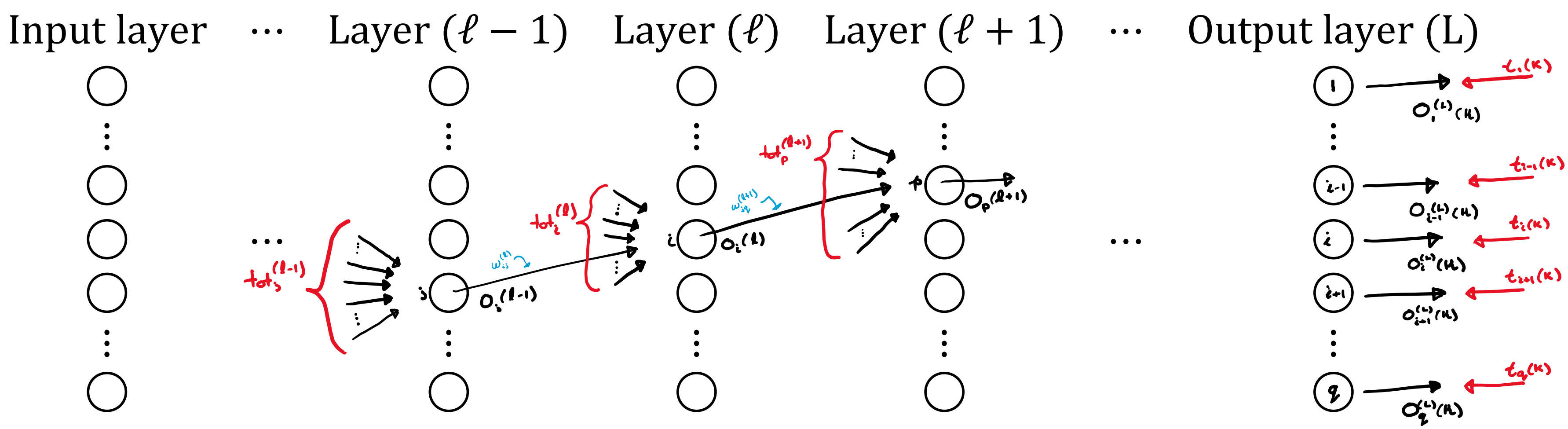
$$\theta^{(5)} = 0$$

- Input $\vec{x} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$

$$O = sgn(\vec{w}^T\ \vec{x} - \theta)$$
$$= sgn\left(\begin{bmatrix} -3 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \end{bmatrix} - 0\right)$$
$$= sgn(3 - 2 - 0) = 1 = t^{(2)}$$

$$\vec{w}^{(6)} = \begin{bmatrix} -3 \\ -1 \end{bmatrix}$$

$$\theta^{(6)} = 0$$



Input layer
Layer (ℓ − 1)
Layer (ℓ)
Layer (ℓ + 1)
Output layer (L)

If $t(k) = k^{th}$ largest target output of the NN

k^{th} training data pair

$k = 1, 2, \dots, n$
$n =$ total number of training data pairs

Error function:

$$E(k) \sim \left[t_1(k) - O_1^{(L)}(k)\right], \dots, \left[t_i(k) - O_i^{(L)}(k)\right], \dots, \left[t_q(k) - O_q^{(L)}(k)\right]$$

$$E(k) = \frac{1}{2}\left[\left(t_1(k) - O_1^{(L)}(k)\right)^2 + \cdots + \left(t_q(k) - O_q^{(L)}(k)\right)^2\right]$$

$$= \frac{1}{2}\sum_{i=1}^{q}[t_i(k) - O_i(k)]^2$$

online training

(For simplicity, drop the "(L)" from notation)

Overall error function:

$$E_c = \sum_{k=1}^{n} E(k)$$
$$= E(1) + E(2) + \cdots + E(k) + \cdots + E(n)$$

$$E_c = \sum_{k=1}^{n} E(k) = \frac{1}{2}\sum_{k=1}^{n}\sum_{i=1}^{q}[t_i(k) - O_i(k)]^2$$

offline training

$E(k) \sim$ online training
$E_c \sim$ offline training

For online training:

$$\min E(k)$$

$$\vec{w}^{(l)}(k+1) = \vec{w}^{(l)}(k) + \Delta\vec{w}(k)$$
gradient descent method

$$\Delta\vec{w}^{(l)} = \Delta w_{ij}^{(l)}$$

Chain rule:

$$\frac{\partial E(k)}{\partial w_{ij}}$$

$$\Delta \vec{w}^{(\ell)} = \Delta \vec{w}_{ij}^{(\ell)}$$
$$= -\eta \frac{\partial E(k)}{\partial O_i^{(\ell)}} \cdot \frac{\partial O_i^{(\ell)}}{\partial t_o t_i^{(\ell)}} \cdot \frac{\partial tot_i{}^{(\ell)}}{\partial w_{ij}^{(\ell)}}$$

If layer ℓ is the output layer L:

$$E(k) = \frac{1}{2} \sum_{i=1}^{q} [t_i(k) - O_i(k)]^2$$

Omit "k"

$$\frac{\partial E}{\partial W_{ij}^{(L)}} = \frac{\partial E}{\partial O_i^{(L)}} \cdot \frac{\partial O_i^{(L)}}{\partial t_o t_i^{(L)}} \cdot \frac{\partial tot_i^{(L)}}{\partial w_{ij}^{(L)}}$$

$$tot_i^{(L)} = \rightarrow O_1^{(L-1)} w_{i1}^{(L)} + O_2^{(L-1)} w_{i2}^{(L)} + \cdots + O_j^{(L-1)} w_{ij}^{(L)} + \cdots$$

$$O_i^{(L)} = f\left(tot_i^{(L)}\right)$$

$$\frac{\partial E}{\partial w_{ij}^{(L)}} = -(t_i - O_i) f'\left(tot_i^{(L)}\right) O_j^{(L-1)}$$

$$\Delta w_{ij}^{(L)} = -\eta \frac{\partial E}{\partial w_{ij}^{(L)}}$$
$$= \eta \underbrace{\left(t_i - O_i^{(L)}\right) f'\left(tot_i^{(L)}\right)}_{\delta_i^{(L)} = \text{error signal}} O_j^{(L-1)}$$

$$\Delta w_{ij}^{(L)} = \eta \delta_i O_j^{(L-1)}$$