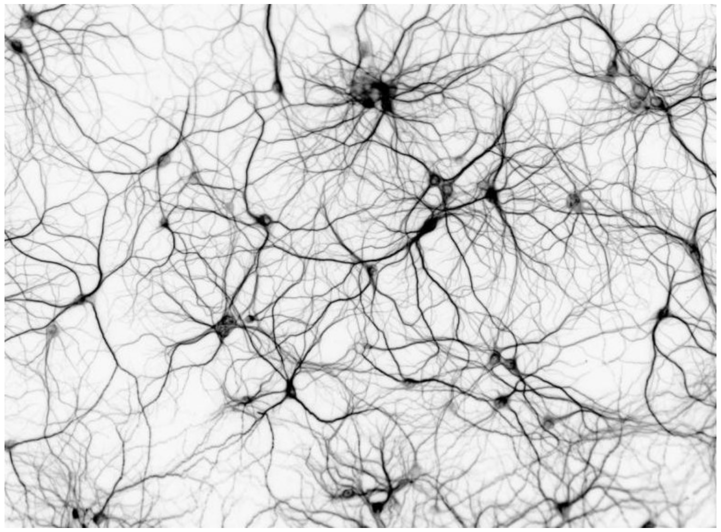
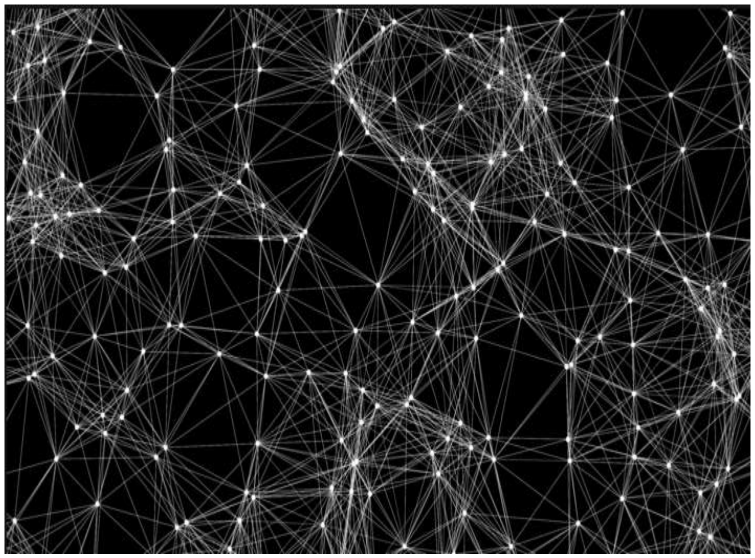
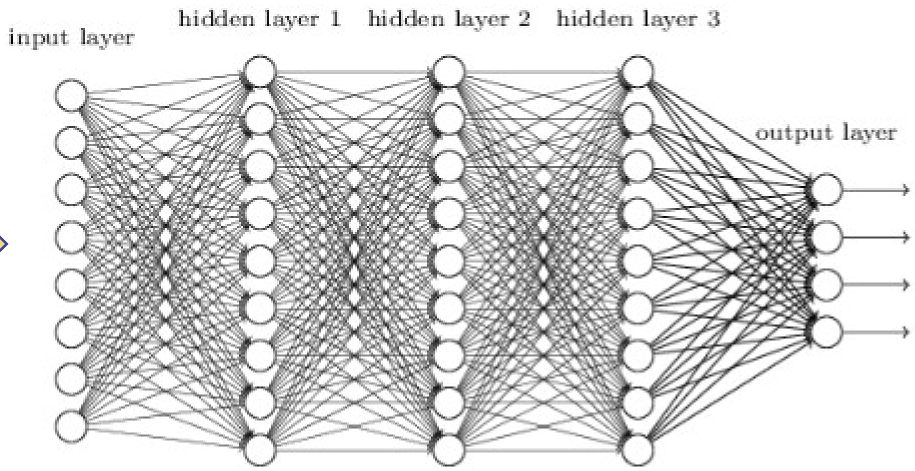
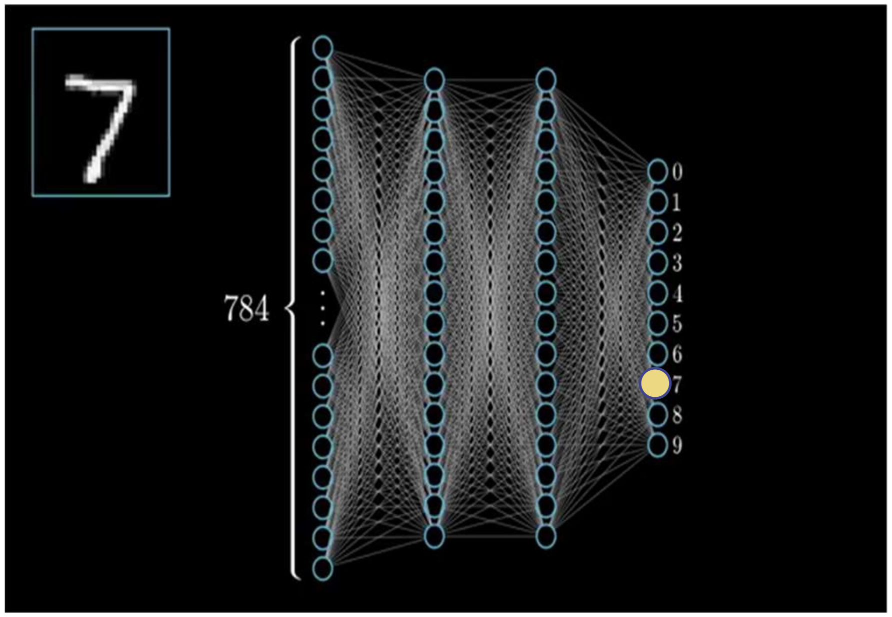
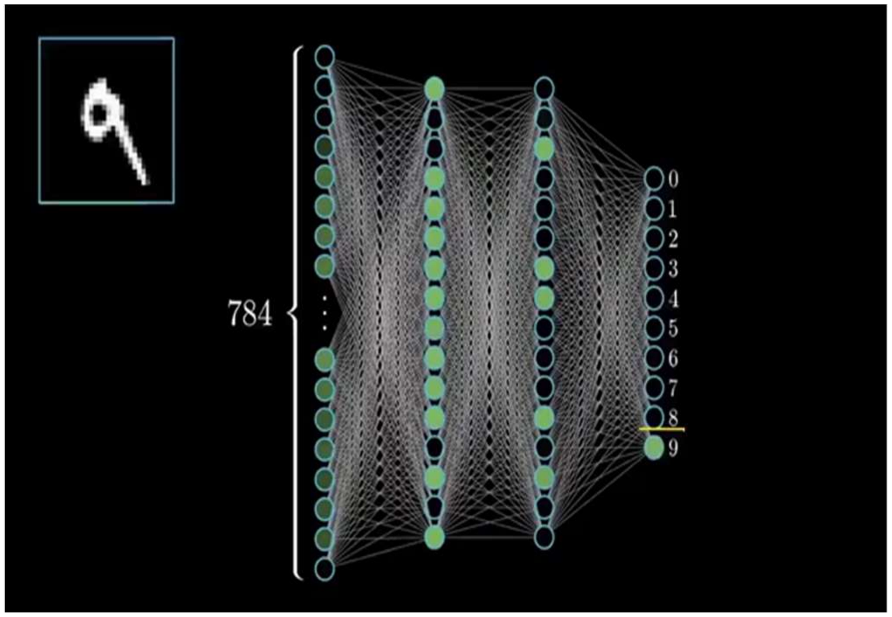





input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer



784
0
1
2
3
4
5
6
7
8
9



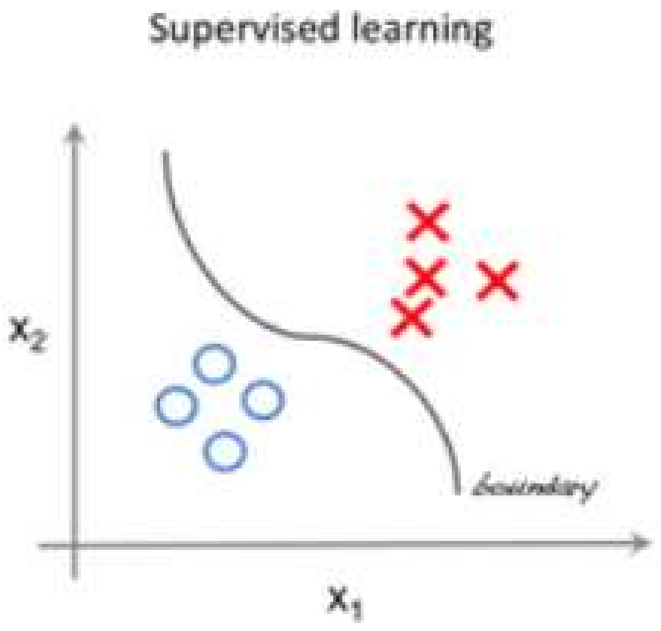
784
0
1
2
3
4
5
6
7
8
9

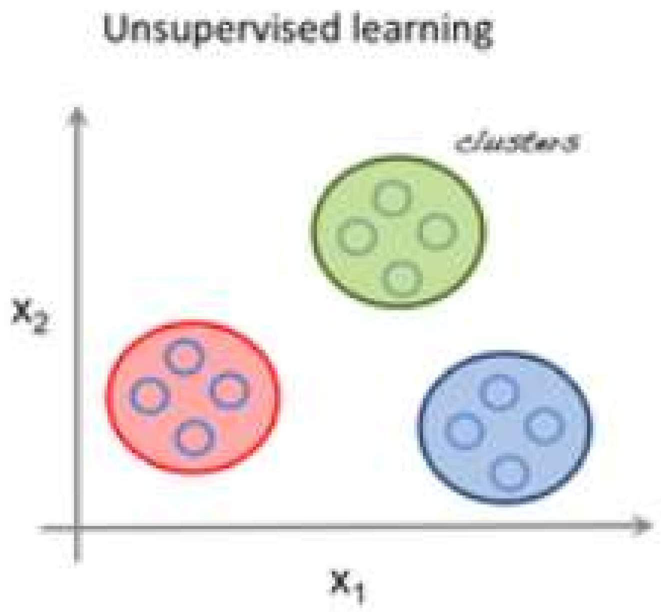
# Chap.2 Learning processes

- Learning is a process by which the $\overset{\text{(weight)}}{\text{free parameters}}$ of a N.N are adapted through a process of stimulation by the environment
- Learning process:
  - (i) The N.N is stimulated by an environment
  - (ii) The N.N undergoes changes in its free parameters as a result of this stimulation
  - (iii) The N.N responds in a new way to the environment

- There is no unique learning Algorithm for the design of N.N
- Basically, learning algorithms differ from each other in the way in which the adjustment to a synaptic weight of a neuron is formulated.

The objective of learning is to find a weight matrix w such that similar input data x will **be grouped or clustered** together.





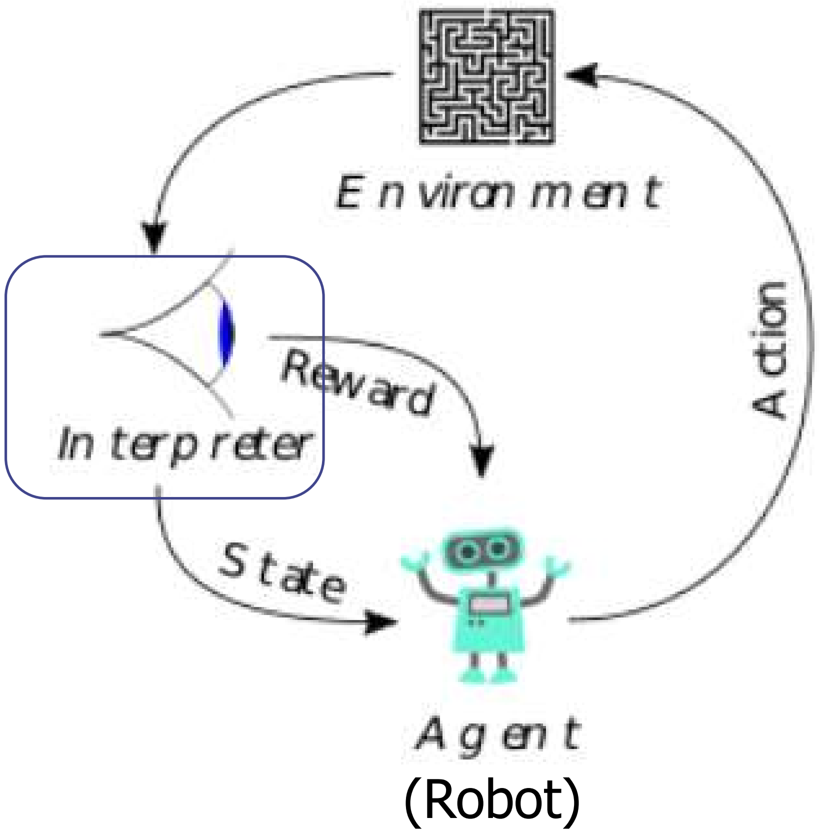
## Learning paradiam

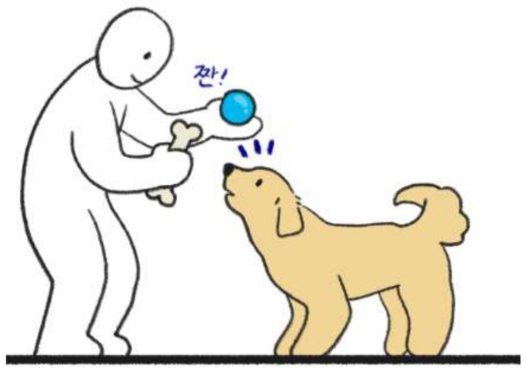
(i) Learning with Teacher: supervised learning with label(answer)
(ii) Learning w/o teacher : unsupervised learning
(iii) Reinforcement learning: unsupervised learning with reward

## [ Reinforcement Learning ]

- The typical framing of a Reinforcement Learning (RL) scenario: an agent takes actions in an environment, which is interpreted into a reward/penalty and a representation of the state, which are fed back into the agent.

- RL is the ML that tries to maximize some notion of cumulative reward.





## 2.1 Error Correction Learning (delta rule, Widrow-Hoff rule)

- Consider a neuron *k*

- Find an error signal, $e_k(n)$ ,such that minimize a cost function($\varepsilon(n)$), or index of performance:

$$e_k(n) = d_k(n) - y_k(n)$$

$$\varepsilon(n) = \frac{1}{2} e_k^2(n) \longleftarrow$$

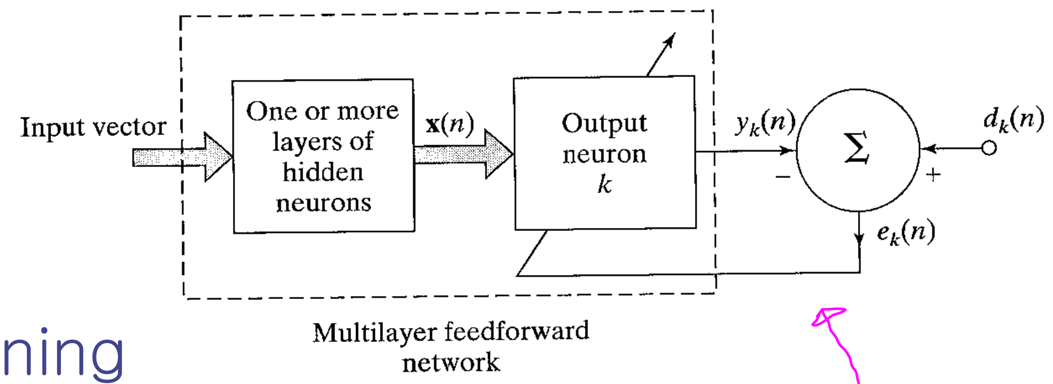
instantaneous value of error energy

,where $n$ denotes time step

- The step-by-step adjustment are continued until the system reaches steady state, I.e., synaptic weights are Stabilized.
  The $e_k(n)$ is used for producing a sequence of corrective adjustments to the synaptic weight of neuron

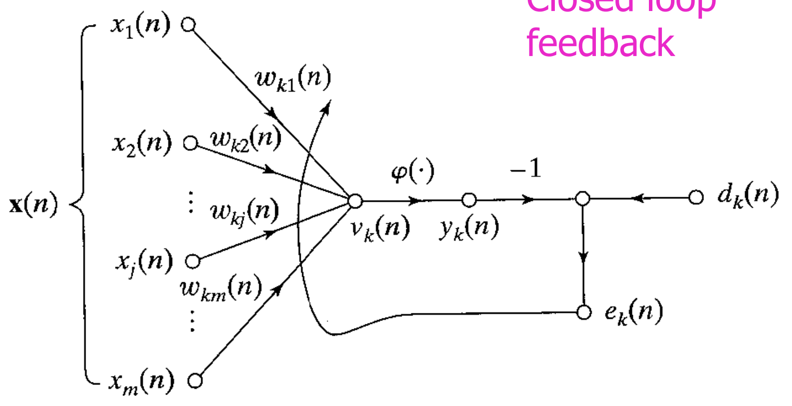
$$\Delta w_{kj}(n) = \eta \cdot e_k(n) \cdot x_j(n)$$

where η : the rate of learning



(a) Block diagram of a neural network, highlighting the only neuron in the output layer

Closed loop feedback

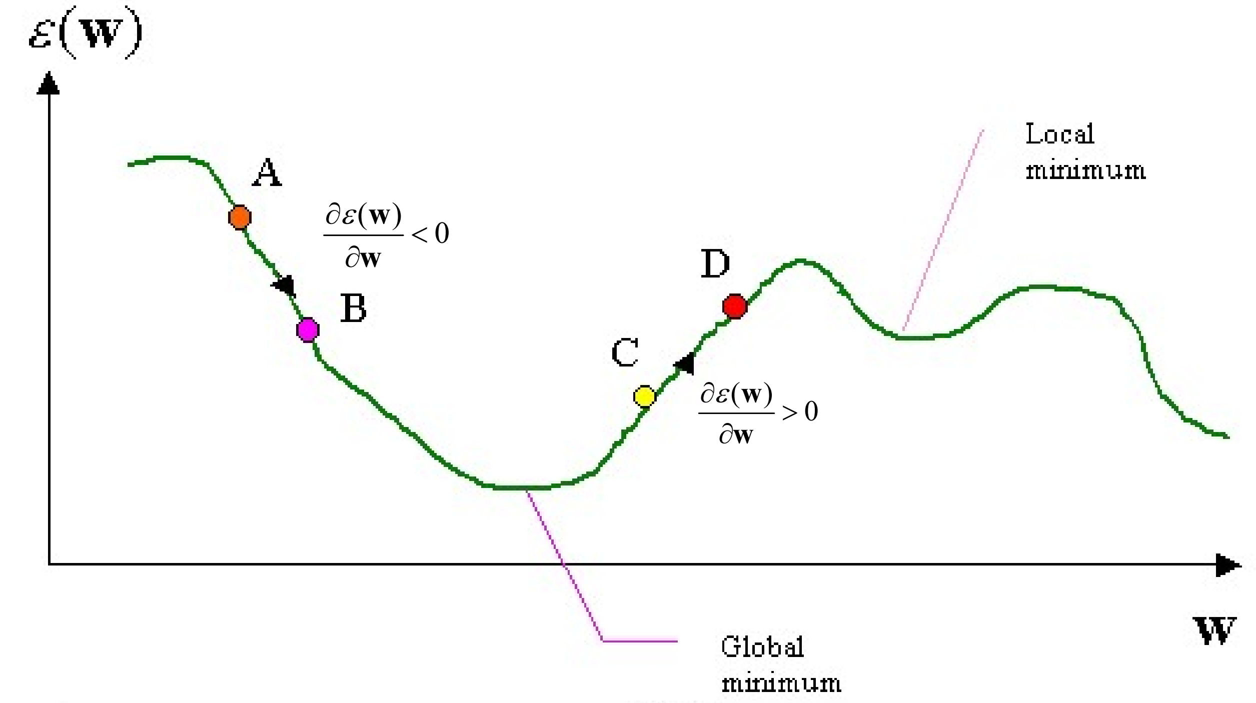


(b) Signal-flow graph of output neuron

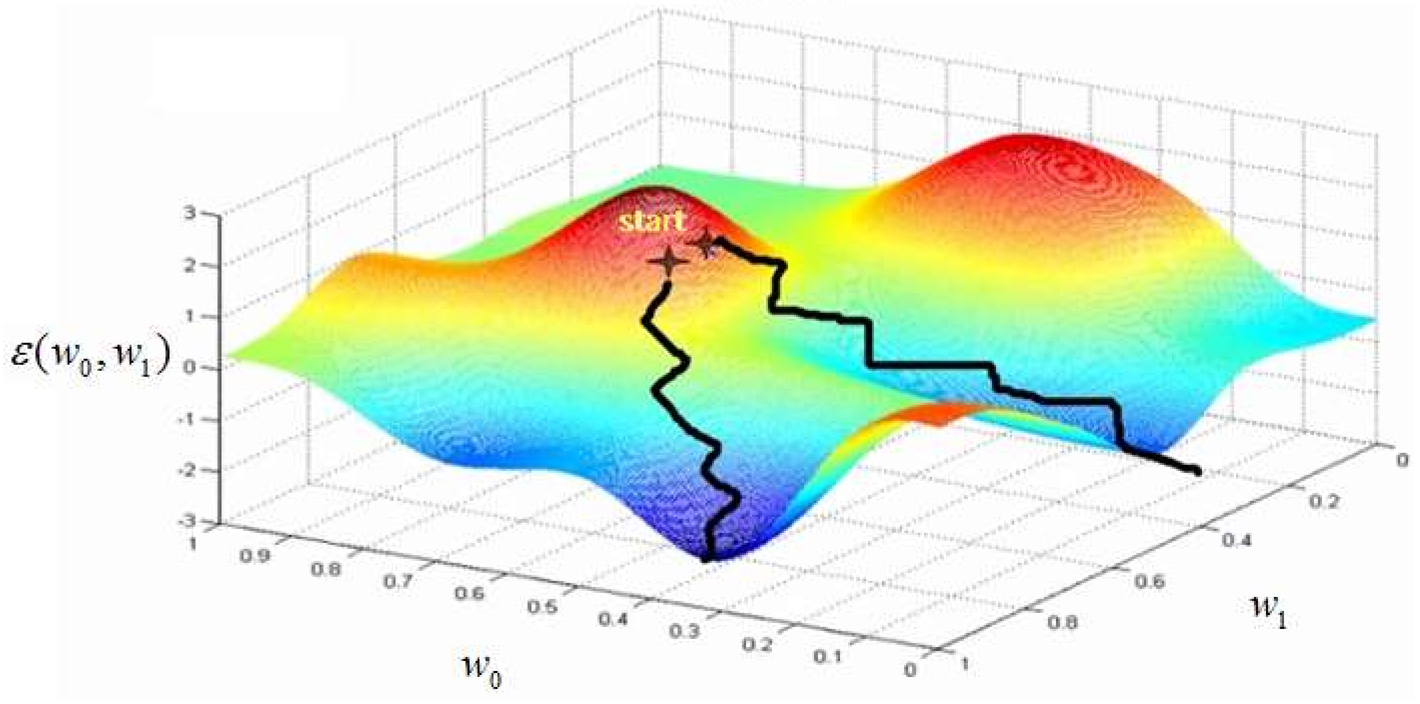
**FIGURE 2.1** Illustrating error-correction learning.

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$$

The η is carefully selected to ensure that the stability or convergence of the iterative learning process is achieved (∵ η is one of parameters that constitute feedback loop)



$\varepsilon(\mathbf{w})$
A
$\frac{\partial \varepsilon(\mathbf{w})}{\partial \mathbf{w}} < 0$
B
C
D
$\frac{\partial \varepsilon(\mathbf{w})}{\partial \mathbf{w}} > 0$
Local minimum
Global minimum
$\mathbf{w}$



start
$\varepsilon(w_0, w_1)$
$w_0$
$w_1$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1

# Gradient (Steepest) Descent Algorithm

✓ the successive adjustments applied to w are in the direction of steepest descent, that is, the choice of direction is where $\varepsilon(\mathbf{w})$ decreases most quickly, which is in the direction opposite to the gradient vector $\nabla\varepsilon(\mathbf{w})$

For any cost func $\varepsilon(\mathbf{w})$

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta \cdot \nabla\varepsilon(\mathbf{w})$$

$\mathbf{g}(n)$ :1'st derivative for cost function !!!

⟵ find $\mathbf{w}$ such that minimize the cost function $\varepsilon(\mathbf{w})$

*where* $\eta(>0)$ : *step size or learning rate*

$$\mathbf{g}(n) = \nabla\varepsilon(\mathbf{w}) = \left[\frac{\partial\varepsilon}{\partial w_1}, \frac{\partial\varepsilon}{\partial w_2}, \ldots, \frac{\partial\varepsilon}{\partial w_m}\right]^T$$

(m x 1) gradient matrix

$$\therefore \Delta\mathbf{w}(n) = \mathbf{w}(n+1) - \mathbf{w}(n) = -\eta \cdot \mathbf{g}(n) \qquad (1)^*$$

convergence

*By considering* 1'*st order taylor series* (assuming $\eta \approx 0$)

$$\varepsilon(\mathbf{w}(n+1)) \approx \varepsilon(\mathbf{w}(n)) + \mathbf{g}(n)^T \cdot \underbrace{\boxed{\Delta\mathbf{w}(n)}}_{-\eta\cdot\mathbf{g}(n)} \qquad (2)$$

*Therefore,*

*Substituing* (1)* *into* (2) *yields* :

$$\varepsilon(\mathbf{w}(n+1)) = \varepsilon(\mathbf{w}(n)) - \eta \cdot \mathbf{g}^T(n) \cdot \mathbf{g}(n)$$

$$= \varepsilon(\mathbf{w}(n)) - \eta \cdot \|g(n)\|^2$$

$$\Rightarrow \varepsilon(\mathbf{w}(n+1)) < \varepsilon(\mathbf{w}(n)) \quad \textit{only if } \eta \approx 0$$

$\varepsilon(\mathbf{w}(n+1))$ $\varepsilon(\mathbf{w}(n))$ $\Delta\mathbf{w}(n)$

*Taylor series* :

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n-1)}(a)}{(n-1)!}(x-a)^{n-1} + \frac{f^{(n)}(a)}{n!}(x-a)^n + \cdots$$

if $\eta$ is not small enough to ignore:

$$\varepsilon(\mathbf{w}(n+1)) = \varepsilon(\mathbf{w}(n)) + \boldsymbol{g}^{\boldsymbol{T}}(n)\cdot\Delta\mathbf{w}(n) + \frac{1}{2}\cdot\Delta\mathbf{w}^{T}(n)\cdot\boldsymbol{H}(n)\cdot\Delta\mathbf{w}(n) \qquad (3)$$

*where* $\Delta\mathbf{w}(n) = \mathbf{w}(n+1) - \mathbf{w}(n)$

$\boldsymbol{H}(n)$: *Hessian matrix of* $\varepsilon(\mathbf{w}(n))$

*Therefore,*

*Substituing* (1)* *into* (3) *yields*:

$$\varepsilon(\mathbf{w}(n+1)) = \varepsilon(\mathbf{w}(n)) - \eta\cdot\boldsymbol{g}^{\boldsymbol{T}}(n)\cdot\boldsymbol{g}(n) + \frac{1}{2}\cdot\eta^{2}\cdot\boldsymbol{g}^{T}(n)\cdot\boldsymbol{H}(n)\cdot\boldsymbol{g}(n)$$

$$= \varepsilon(\mathbf{w}(n)) - \eta\cdot\left\|g(n)\right\|^{2} + \frac{1}{2}\cdot\eta^{2}\cdot\boldsymbol{g}^{T}(n)\cdot\boldsymbol{H}(n)\cdot\boldsymbol{g}(n)$$

$\Rightarrow\ \varepsilon(\mathbf{w}(n+1))\ ?\ \varepsilon(\mathbf{w}(n))\ \Rightarrow$ We can't guarantee it's covergence

## Hessian

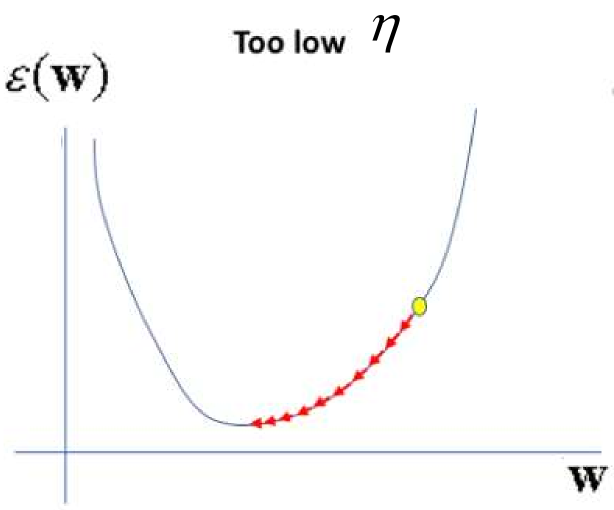
## Gradient

$$\nabla \varepsilon(\mathbf{w}) = \left[\frac{\partial \varepsilon}{\partial w_1}, \frac{\partial \varepsilon}{\partial w_2}, \ldots, \frac{\partial \varepsilon}{\partial w_m}\right]^T$$

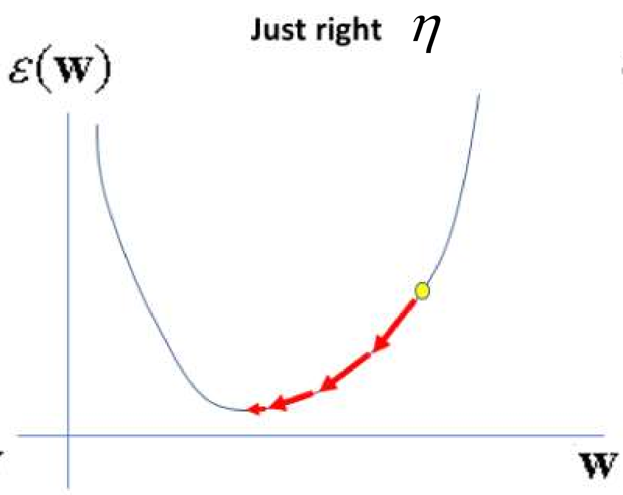
$\nabla^2 \varepsilon(\mathbf{w})$

Laplacian

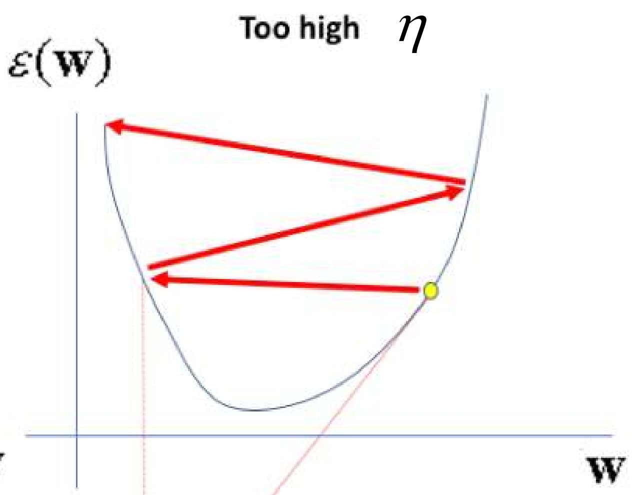
$$H[\varepsilon(\mathbf{w})] = \begin{bmatrix} \frac{\partial^2 \varepsilon}{\partial w_1^2}, & \frac{\partial^2 \varepsilon}{\partial w_1 w_2} & ,\ldots, & \frac{\partial^2 \varepsilon}{\partial w_1 w_m} \\ \frac{\partial^2 \varepsilon}{\partial w_2 w_1}, & \frac{\partial^2 \varepsilon}{\partial w_2^2} & ,\ldots, & \frac{\partial^2 \varepsilon}{\partial w_2 w_m} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 \varepsilon}{\partial w_m w_1}, & \frac{\partial^2 \varepsilon}{\partial w_m w_2} & ,\ldots, & \frac{\partial^2 \varepsilon}{\partial w_m^2} \end{bmatrix}^T$$



A small learning rate requires many updates before reaching the minimum point

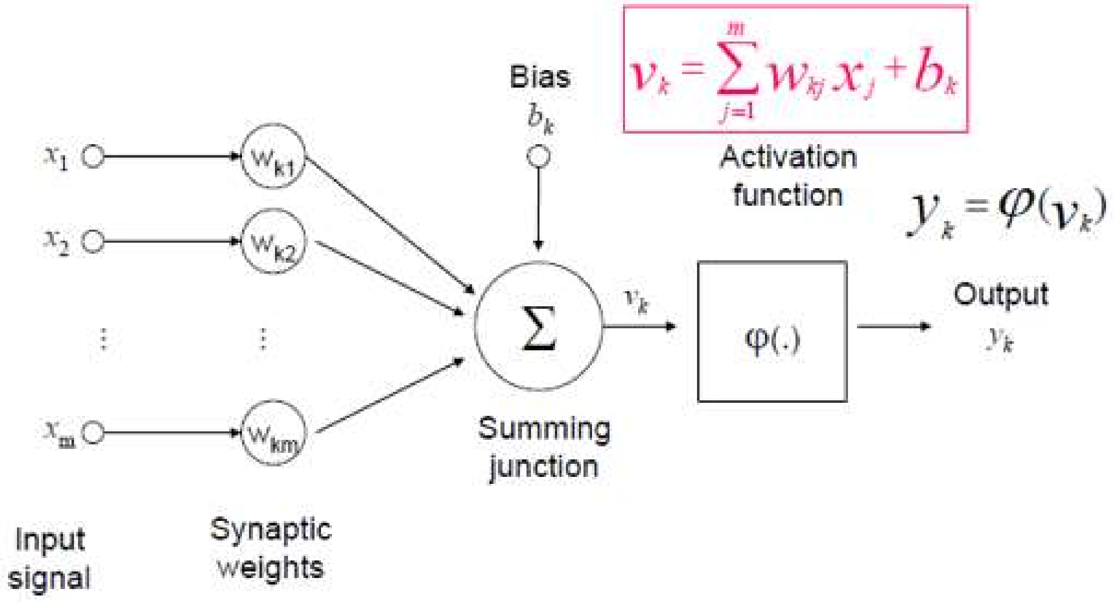


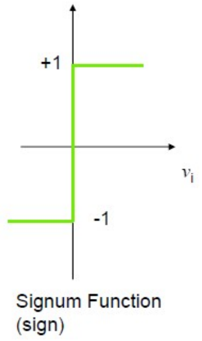
The optimal learning rate swiftly reaches the minimum point

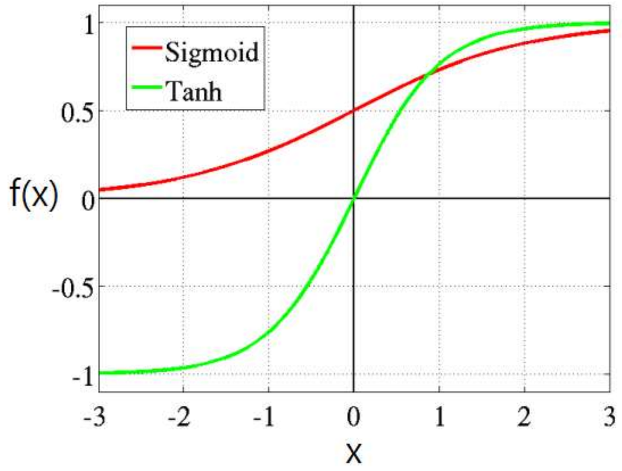


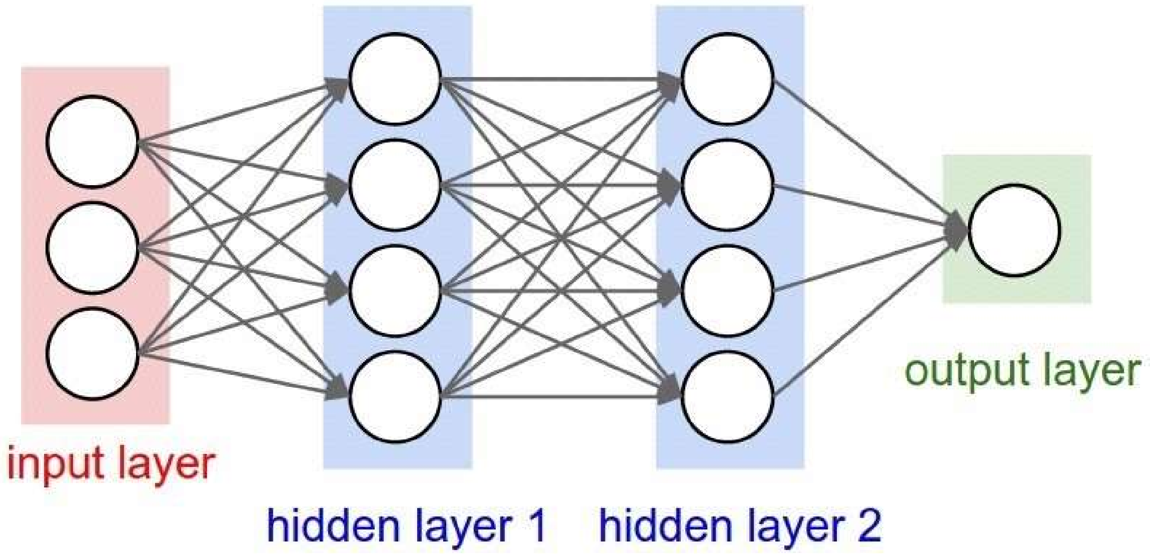
Too large of a learning rate causes drastic updates which lead to divergent behaviors

## <activation function>

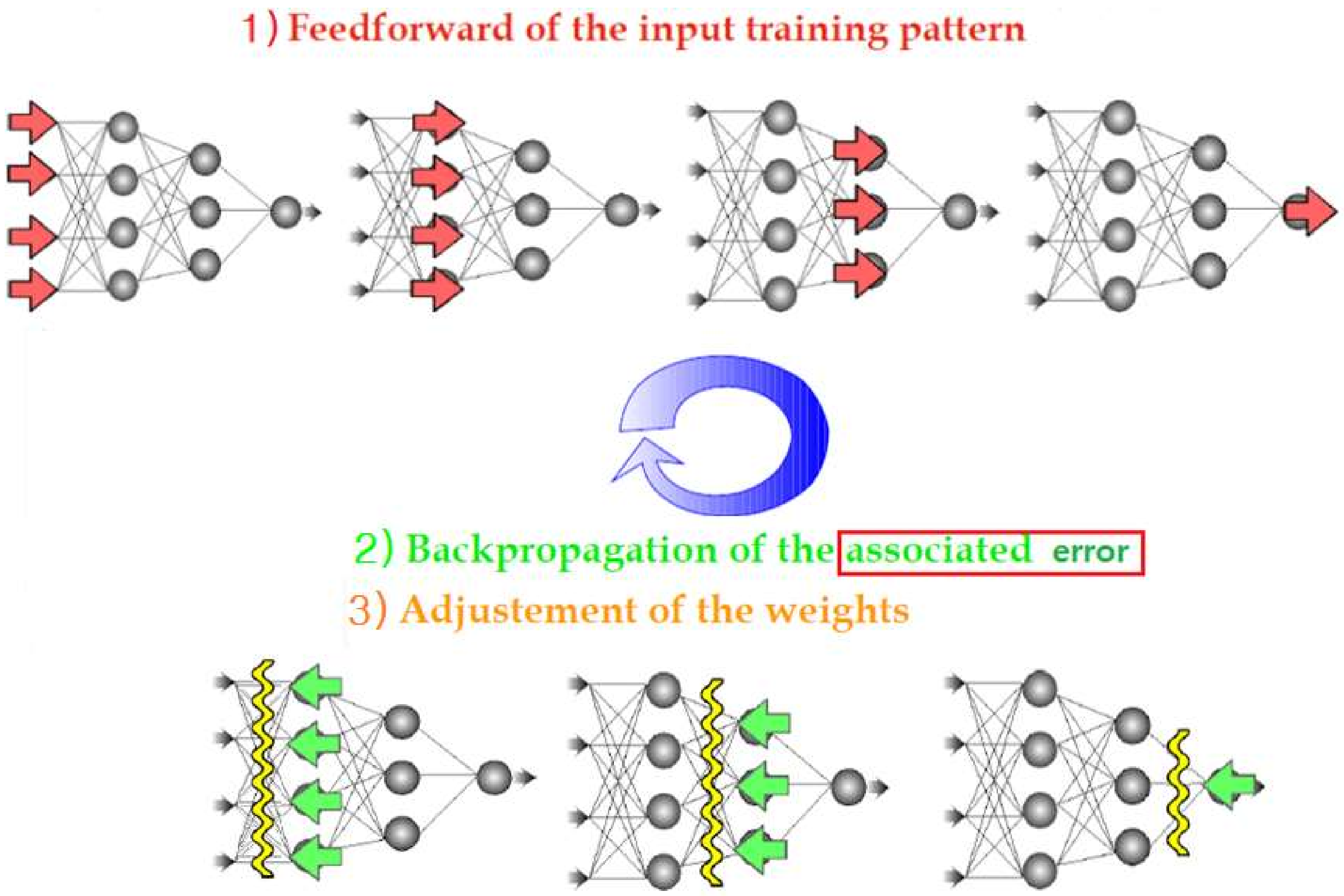

$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k$
Bias
$b_k$
$x_1$
$x_2$
$x_m$
$w_{k1}$
$w_{k2}$
$w_{km}$
$\Sigma$
$v_k$
φ(.)
Activation function
$y_k = \varphi(v_k)$
Output
$y_k$
Summing junction
Input signal
Synaptic weights


+1
-1
$v_i$
Signum Function (sign)


Sigmoid
Tanh
f(x)
X


input layer
hidden layer 1
hidden layer 2
output layer

## Back propagation training cycle


1) Feedforward of the input training pattern
2) Backpropagation of the associated error
3) Adjustement of the weights

## Back-Propagation Algorithm

✓ Error signal for neuron *j* at iteration *n*:

$$e_j(n) = d_j(n) - y_j(n) \quad \text{----①}$$

✓ Total error energy:

$$\varepsilon(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n) \qquad \text{---②}$$

, where the set C includes <u>all the neurons</u> in the output layer

◆ Average squared error energy:

$$\varepsilon_{av} = \frac{1}{N}\sum_{n=1}^{N}\varepsilon(n) \quad :Cost\ \ func \qquad \text{---③}$$

$N$: the total # of patterns in the train set every update (batch size)

# Matlab Tutorial 2

https://www.youtube.com/watch?v=RQ8I6xVEpMU&list=PLnVYEpTNGNtX6FcQm90I0WXdvhoEJPp3p&index=6

https://www.youtube.com/watch?v=TgARZWgXWS4&list=PLnVYEpTNGNtX6FcQm90I0WXdvhoEJPp3p&index=7

https://www.youtube.com/watch?v=kqtPdDaMUEk&list=PLnVYEpTNGNtX6FcQm90I0WXdvhoEJPp3p&index=8

https://www.youtube.com/watch?v=bVu8a-upOBM&list=PLnVYEpTNGNtX6FcQm90I0WXdvhoEJPp3p&index=9

https://www.youtube.com/watch?v=IuI9cONtwFY&list=PLnVYEpTNGNtX6FcQm90I0WXdvhoEJPp3p&index=10