CH. 11: Multilayer Perceptrons

Ups and Downs of NN Study:

- 1958: Perceptron (linear model)
- 1969: Perceptron has limitation
- 1980: Multi-Layer Perceptron -- Do not have significant difference from DNN today
- 1986: Backpropagation -- Efficient for training multi-
- layer neural networks
- 2006: Encoder-Decoder, Restricted Boltzmann Machine -- For pre-training multi-layer neural networks
- 2009: GPU
- 2011: Start to be popular in speech and image processing

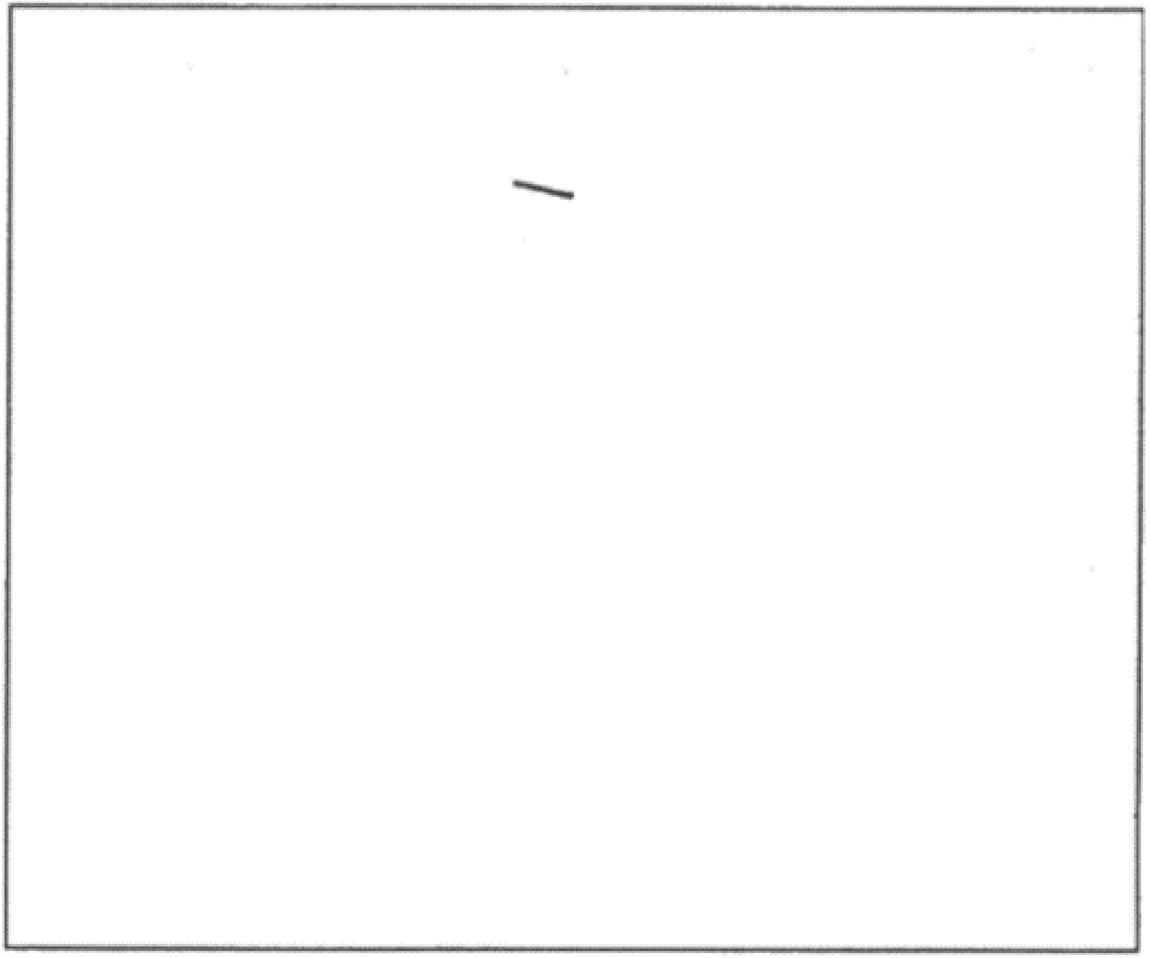
11.1 Introduction

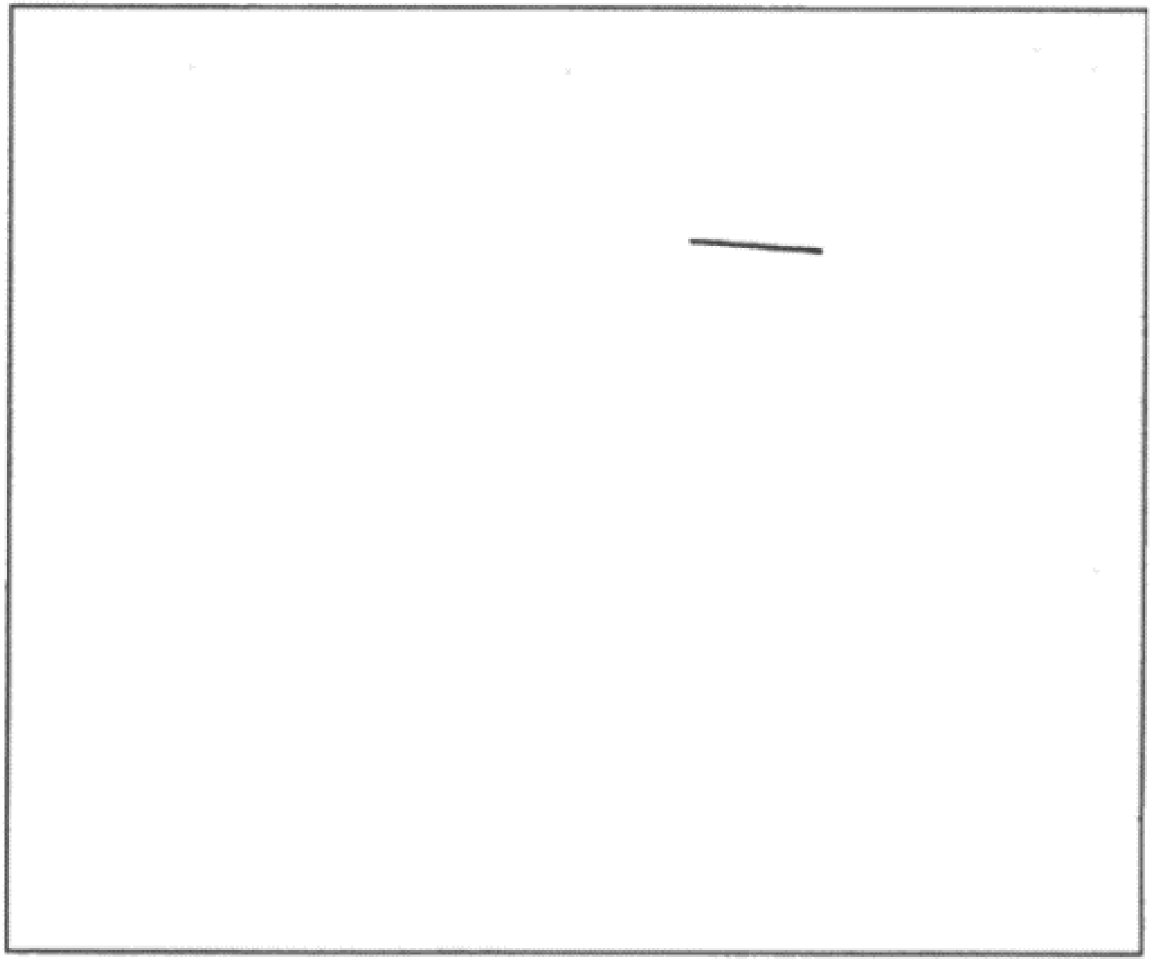
- ANN technology
  - Human brain is the most complex computing device that we have ever known.
  - Conventional computers are good in scientific and mathematical computations, creation and manipulation of databases, control functions, graphics, word processing, …………………
  - How do computers learn, analyze, organize, adapt comprehend, associate, recognize, plan, decide, ……………………………….……..

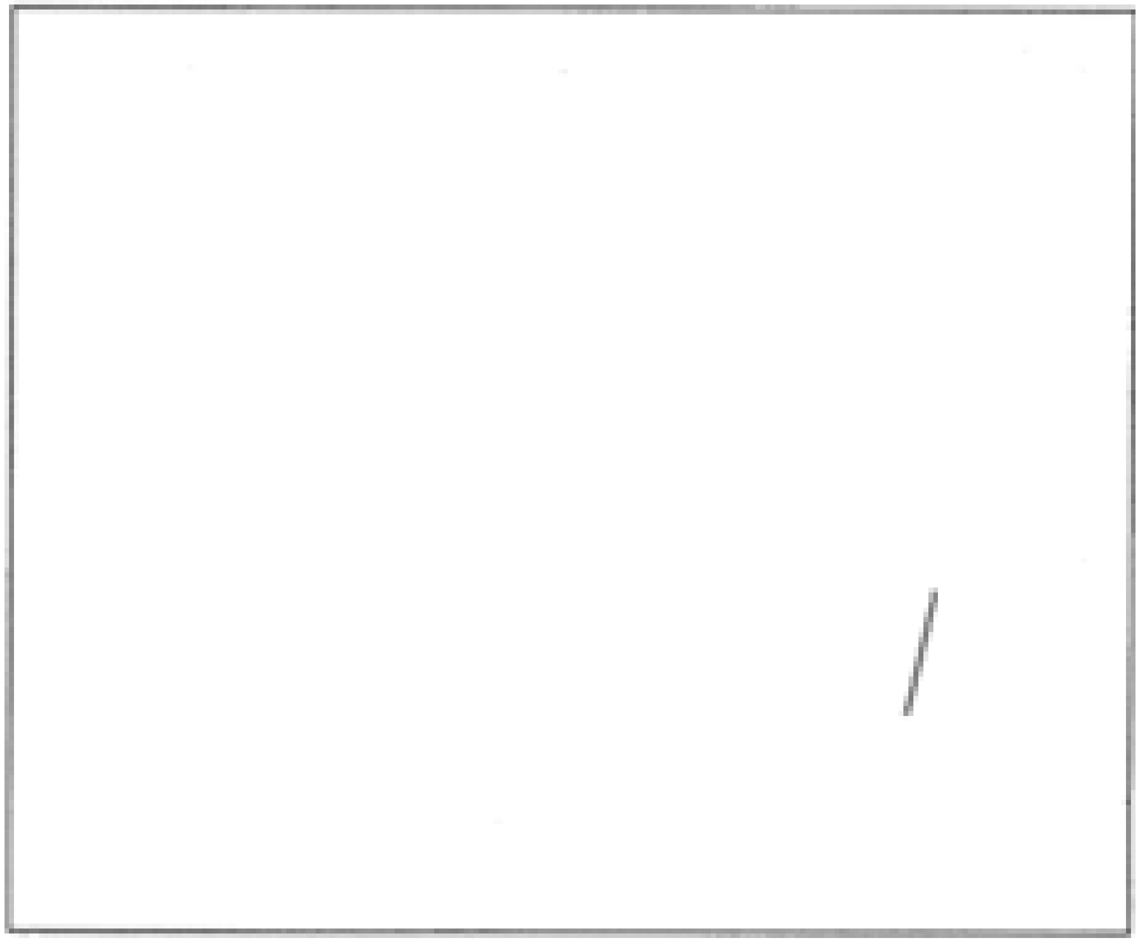
- Many problems are not suitable to be solved by sequential machines and algorithms.

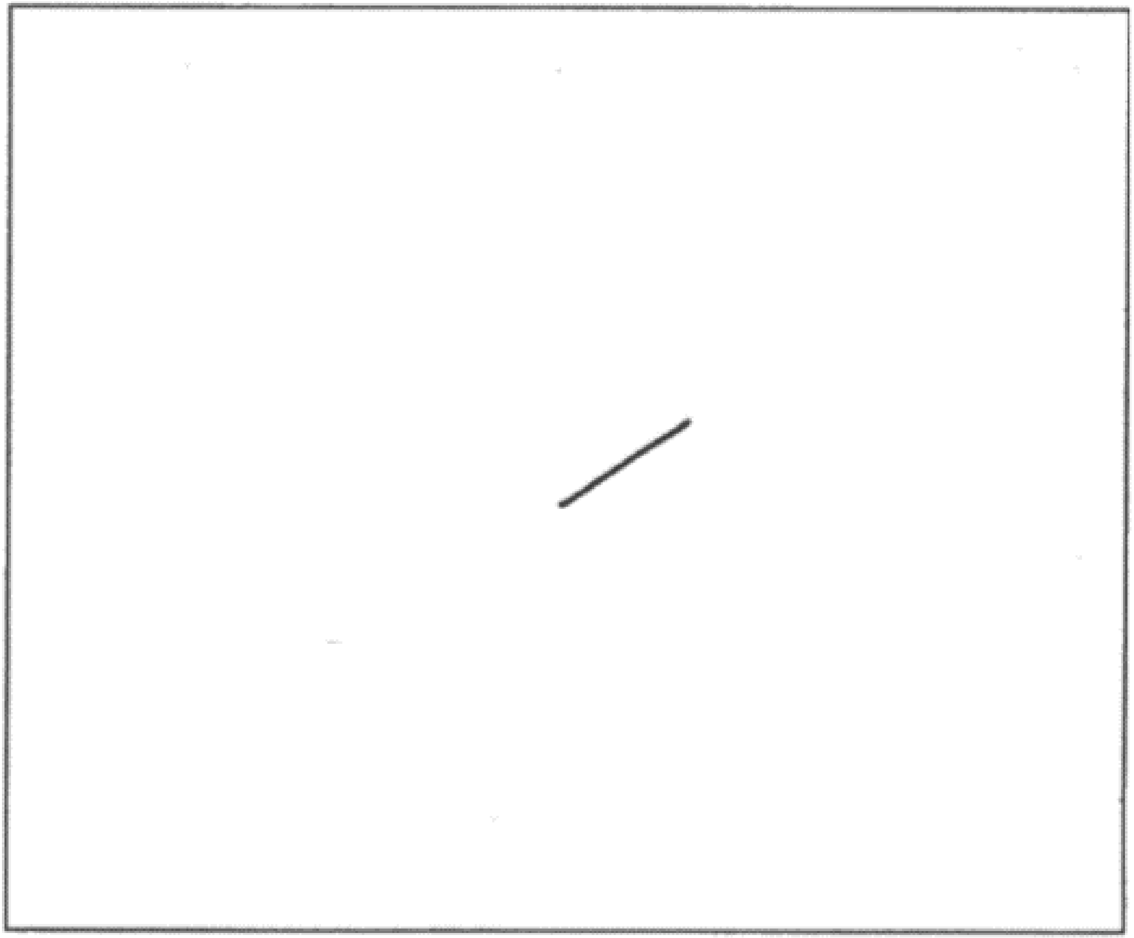
  Parallel-processing architectures may be good tools for solving difficult-to-solve, or unsolved problems.

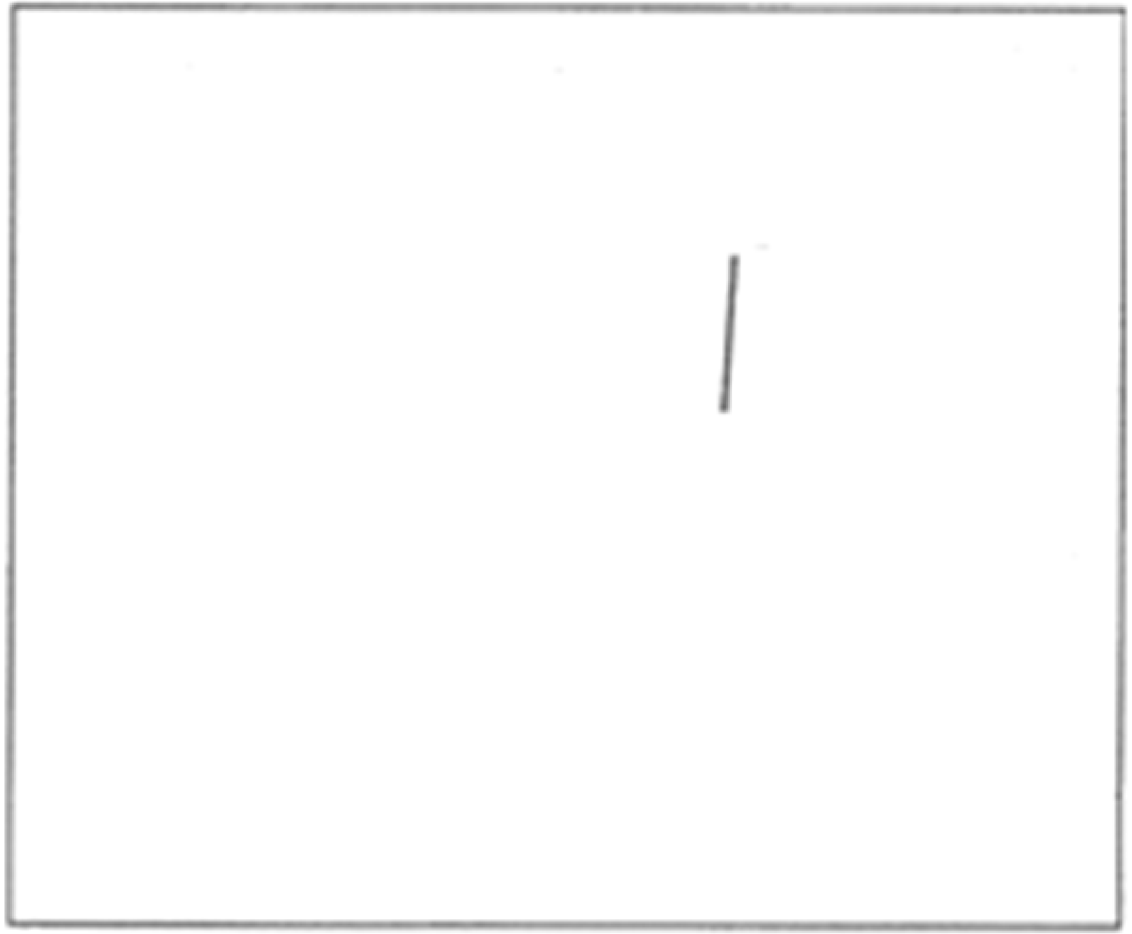
Example 1: Perceptual-Organization Problem (23 line drawings)

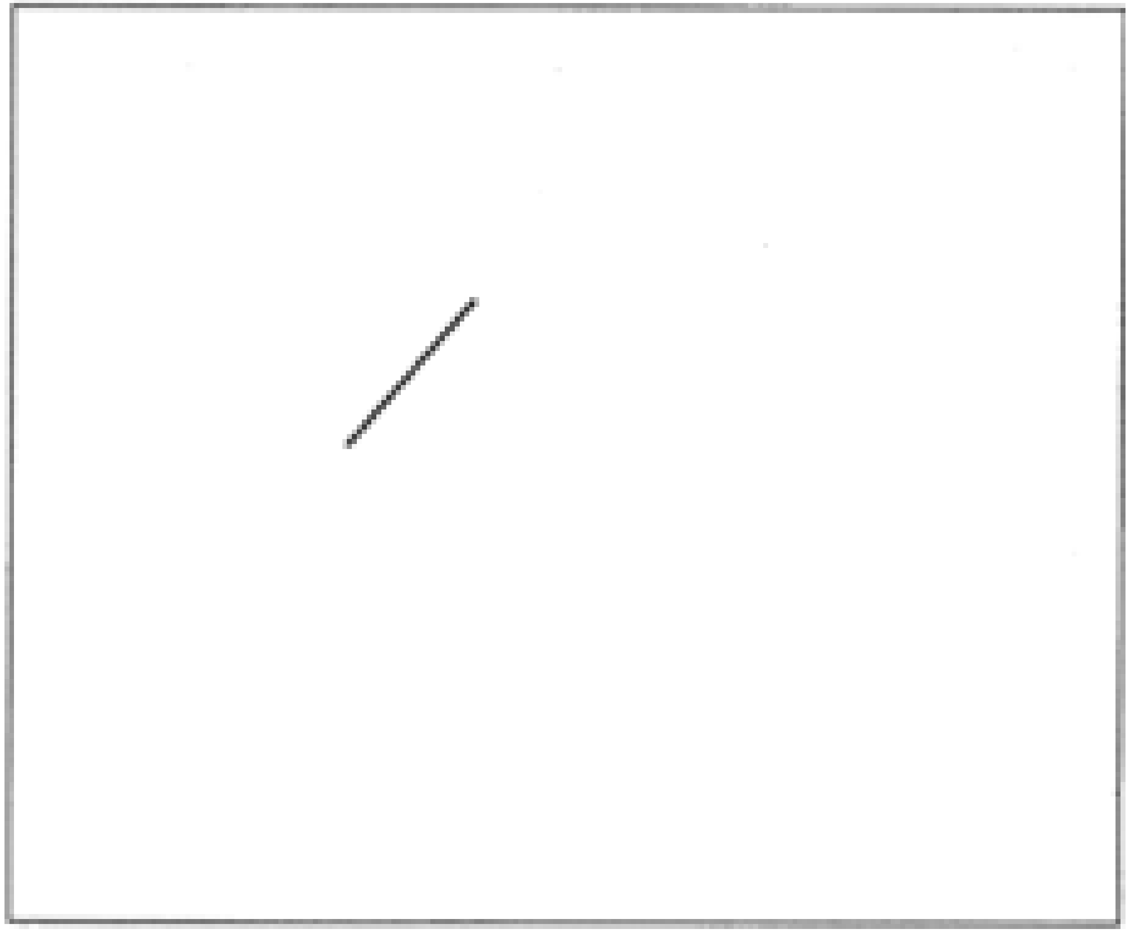




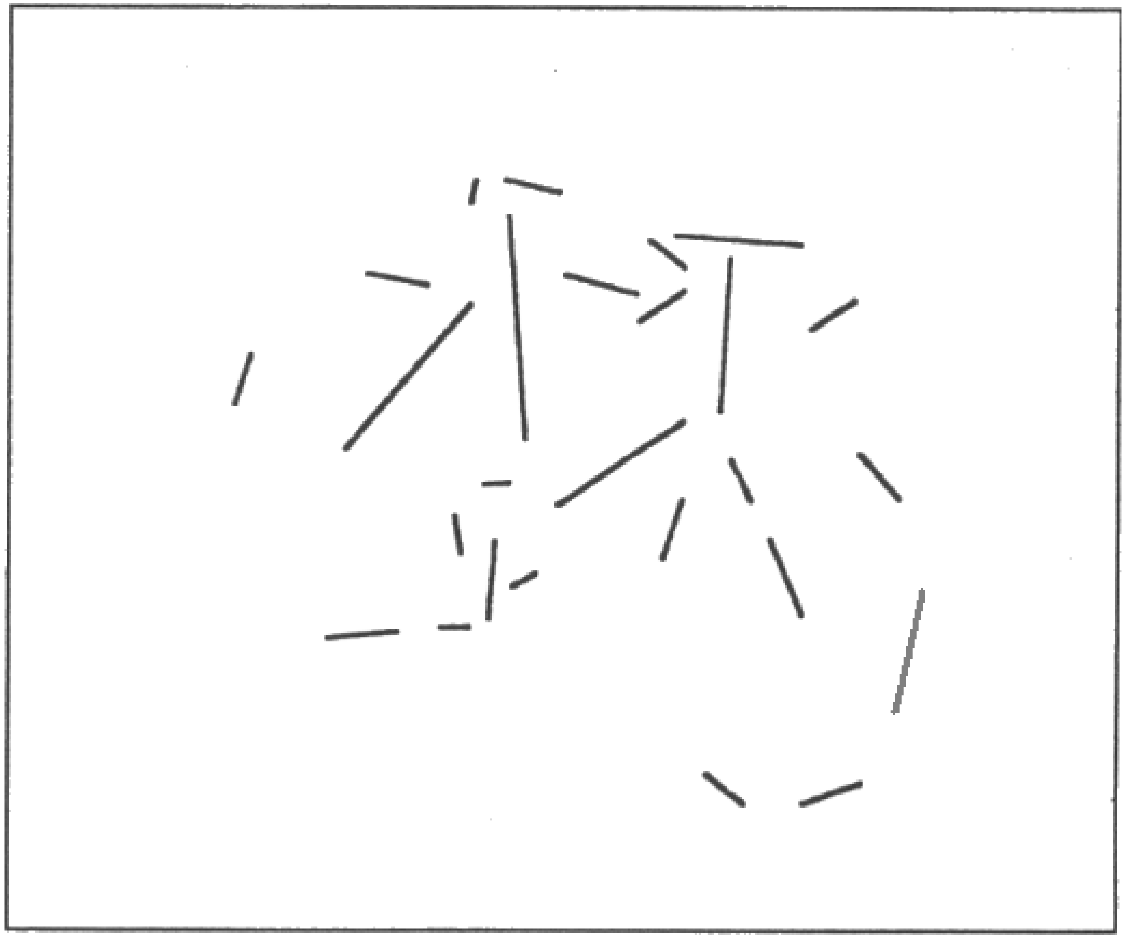




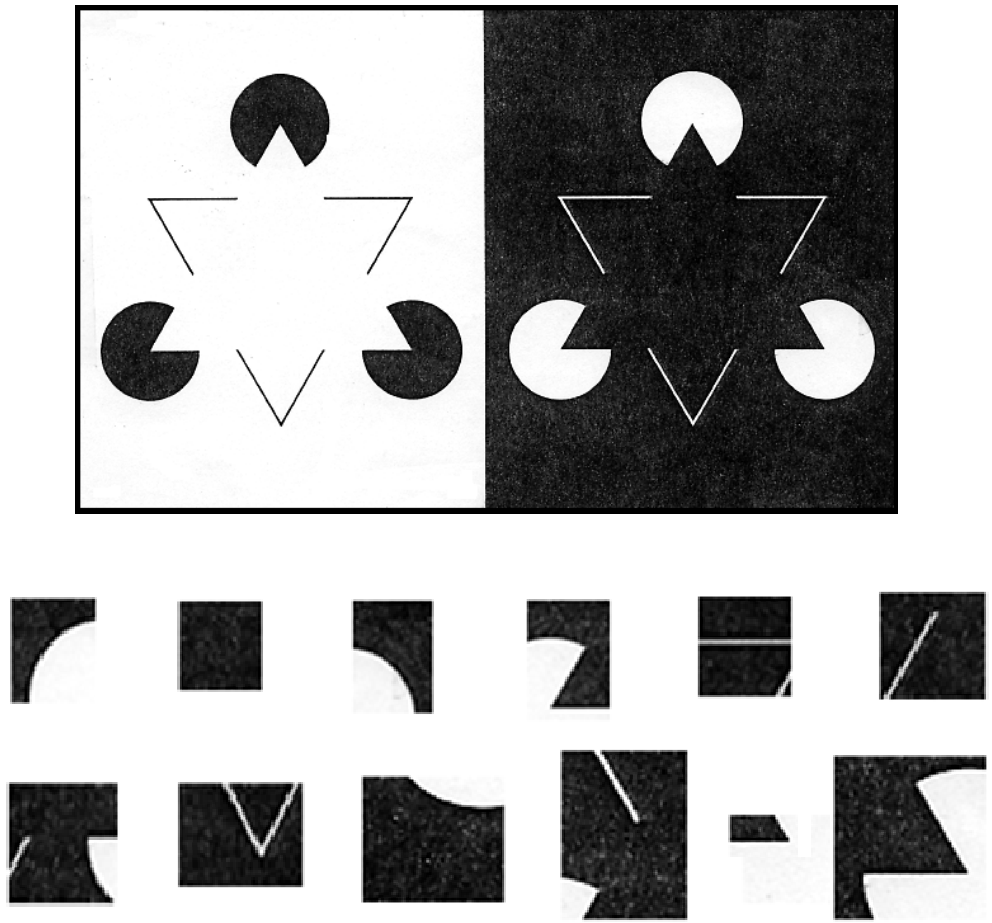




Perceptual organization is well processed in a parallel manner.



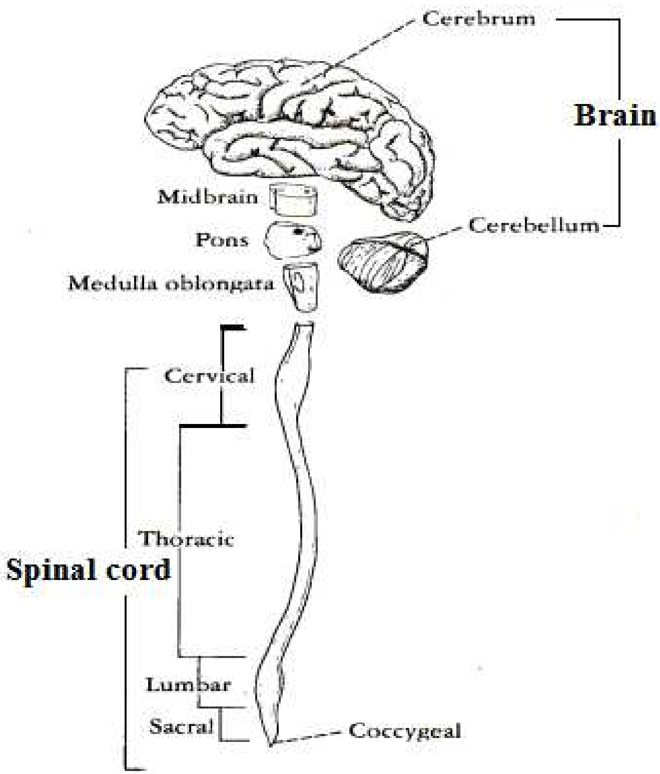
Example 2: Subjective inference (reasoning)

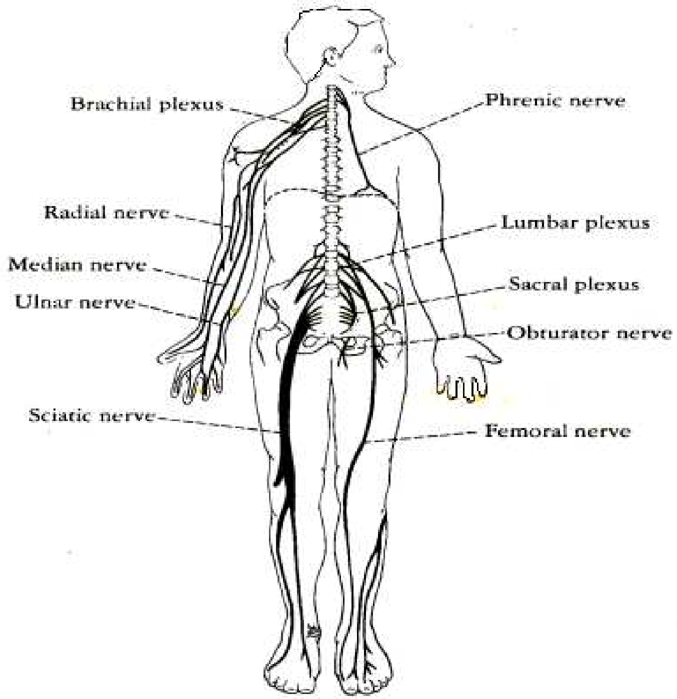


Example 3: Visual Pattern Recognition

11.2 Neurophysiology

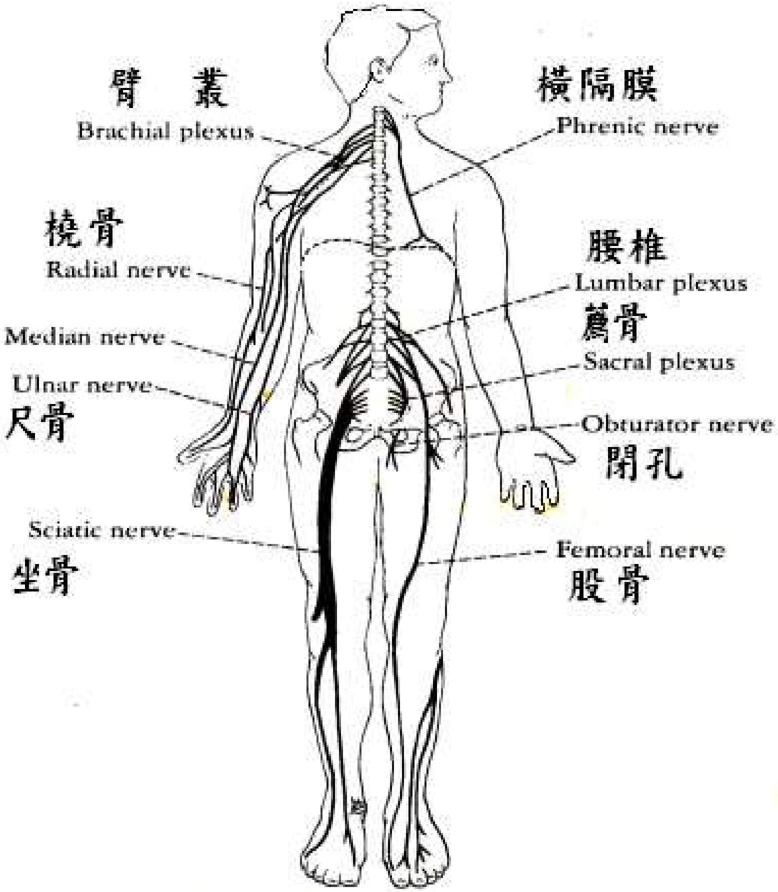
Three major components constructing the human nervous system: brain, spinal cord, periphery



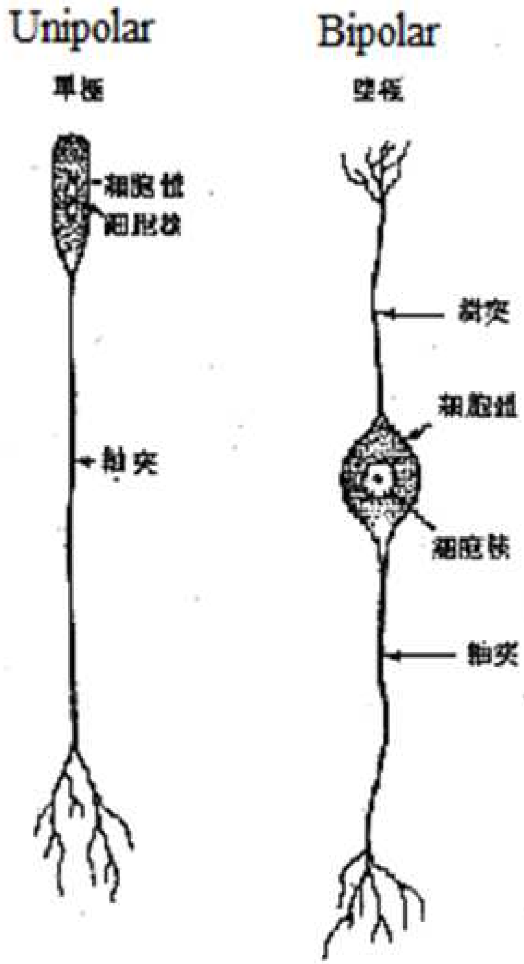


Peripheral nervous system

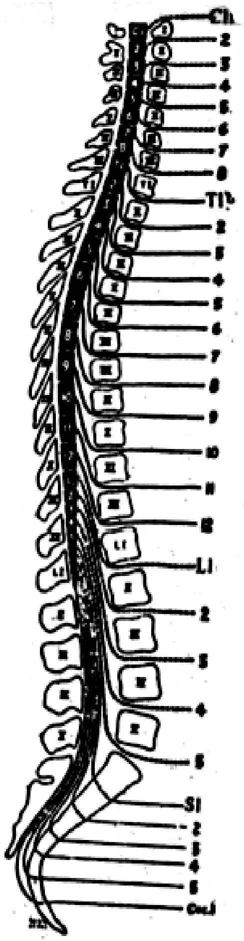
- **Periphery**

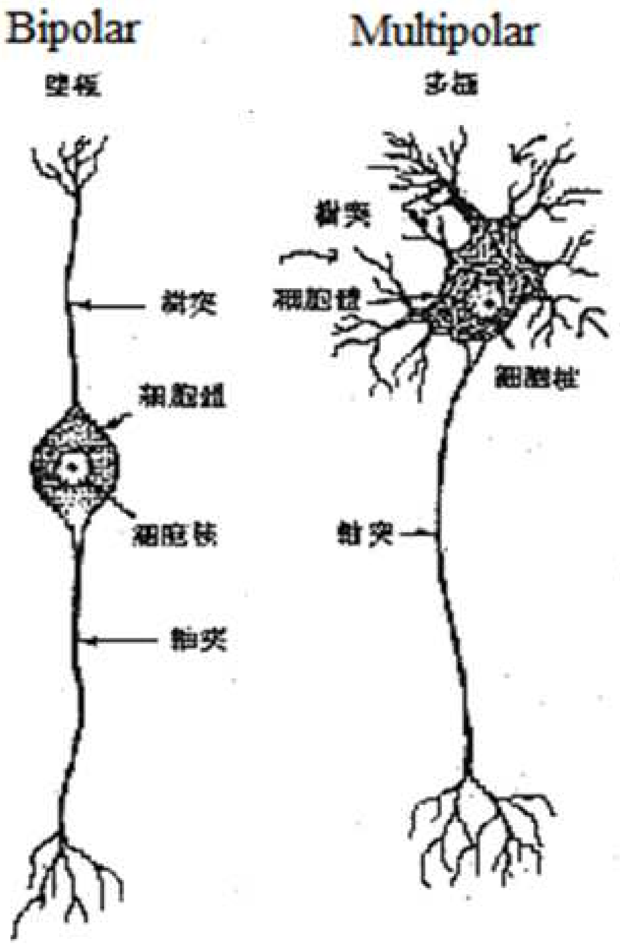


Peripheral nervous system

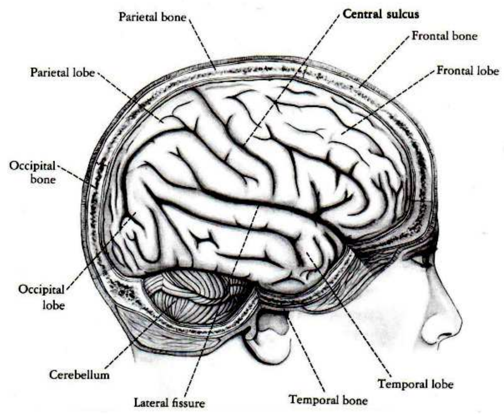


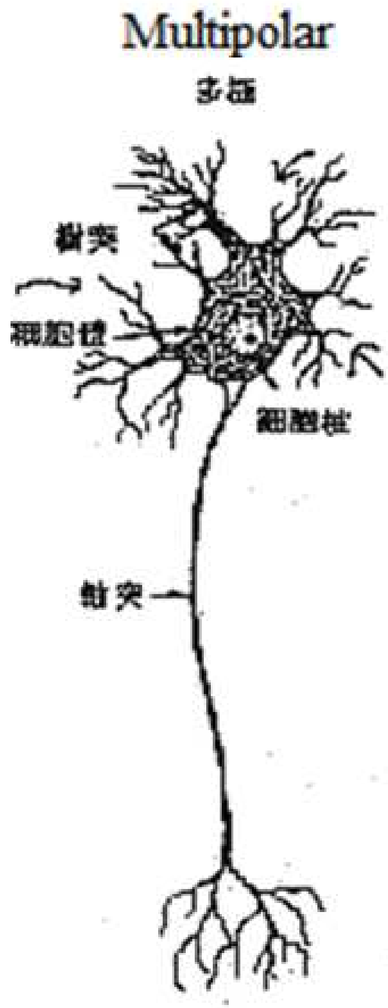
- **Spinal Nerves**



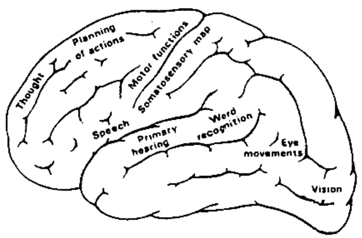


- **Brain**





Cerebral cortex

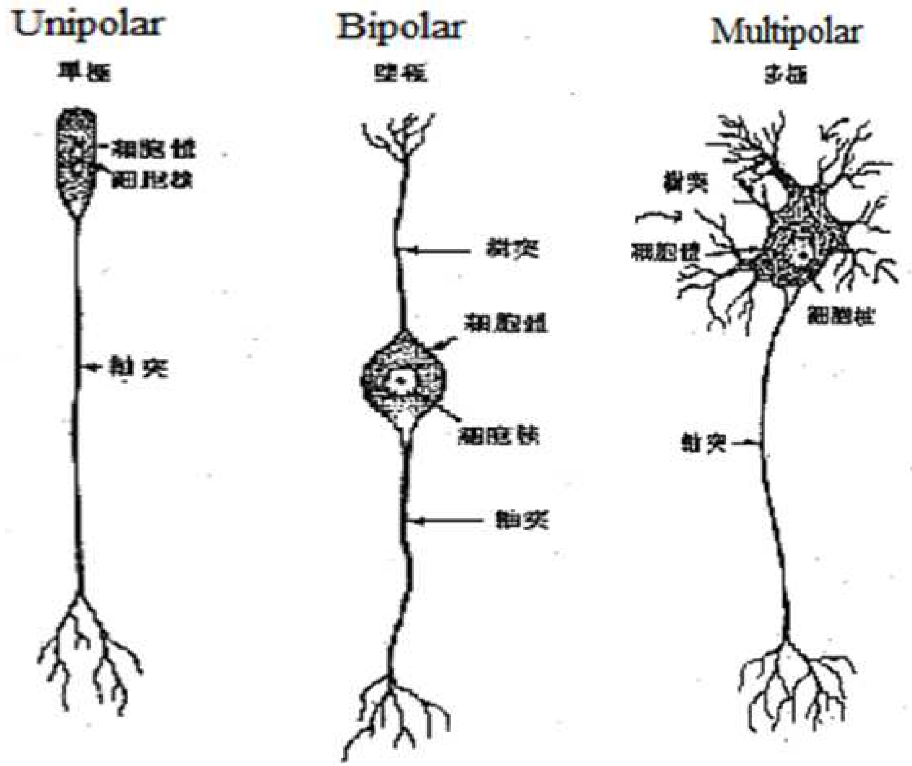


Size: 1 m^2
Thick: 2-4 mm
Layers: 6

◎ Single-Neurons Physiology

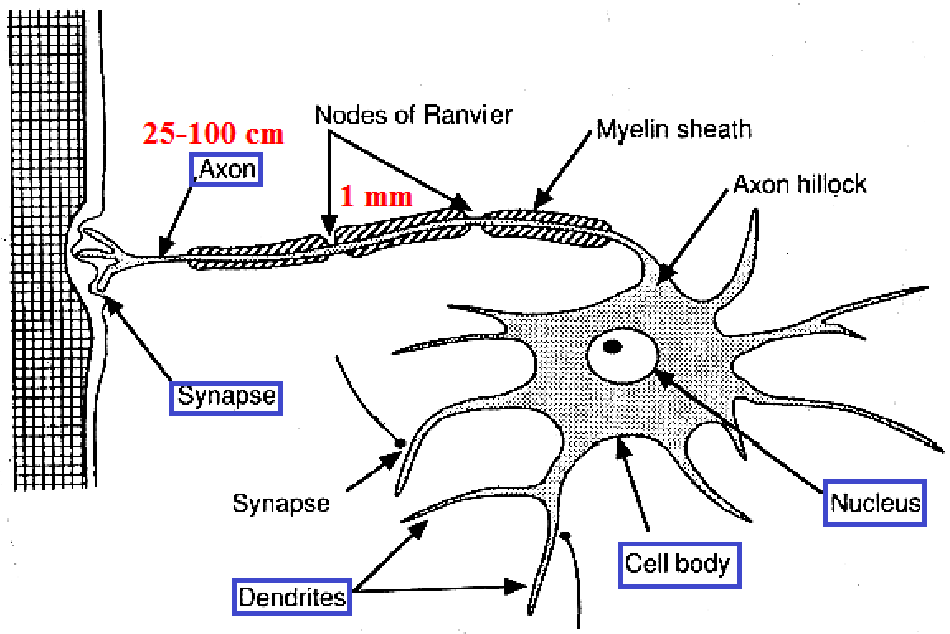
Three types of neurons:

1. unipolar
2. bipolar
3. multipolar



Terminal	Pathway	Central
末梢神經	連絡神經	中樞神經
感覺器官	脊髓	腦

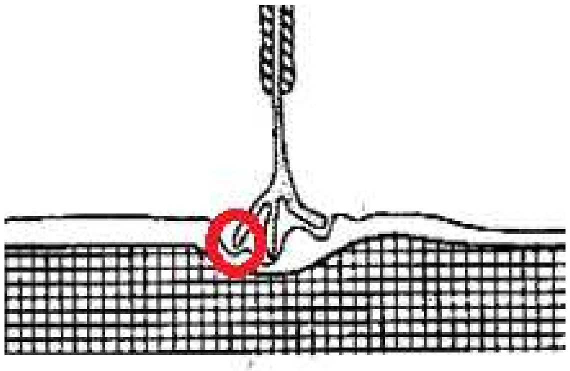
◎ Multi-Polar Neuron

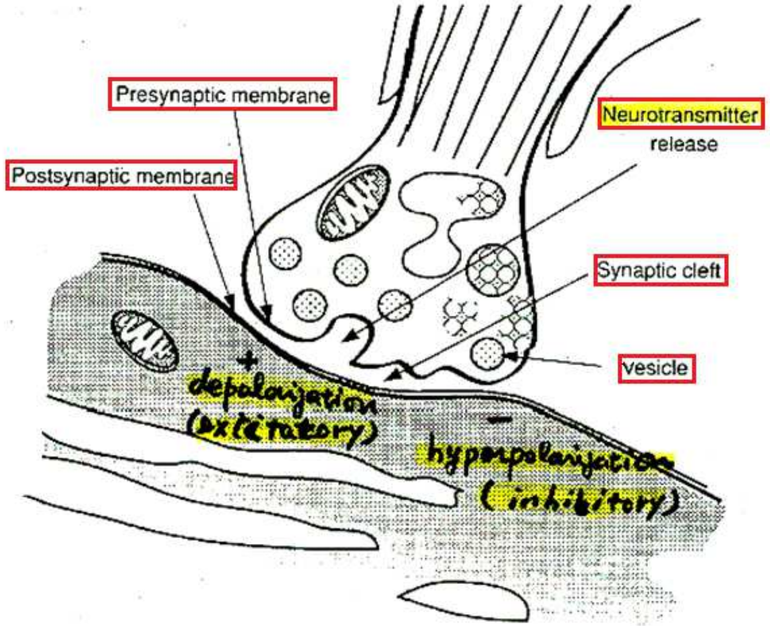


The input impulses can be excitatory or inhibitory and are summed at the axon hillock. If the summed potential is sufficient, an action potential is generated.

◎ Synaptic Function

Neurotransmitters held in vesicles and are released near presynaptic membrane into synaptic cleft and absorbed by postsynaptic membrane.





Signal : frequency of pulses

Learning: adjusting synaptic gaps

Memory : strength of synaptic connections

Knowledge is acquired through a learning process.

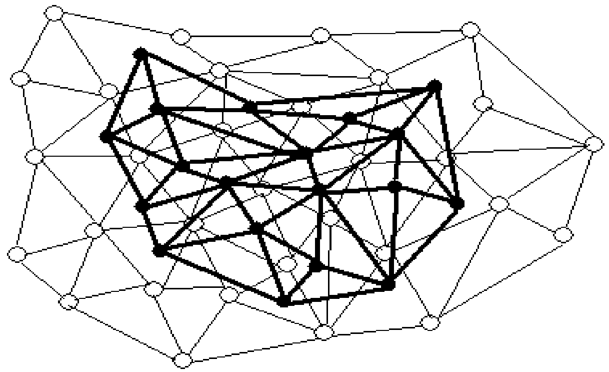
Acquired knowledge is stored in interneuron links in terms of strengths.

◎ Comparison between nervous and computer systems

	Neurons	Processing units
Switch time	10e-3 sec	10e-9 sec
Synapses	10e+3	10
Number	10e+11	~

11.3 Artificial Neural Networks

- A collection of parallel processing units (neurons) interconnected together

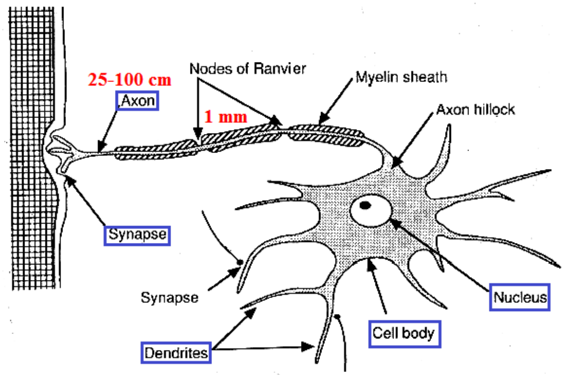


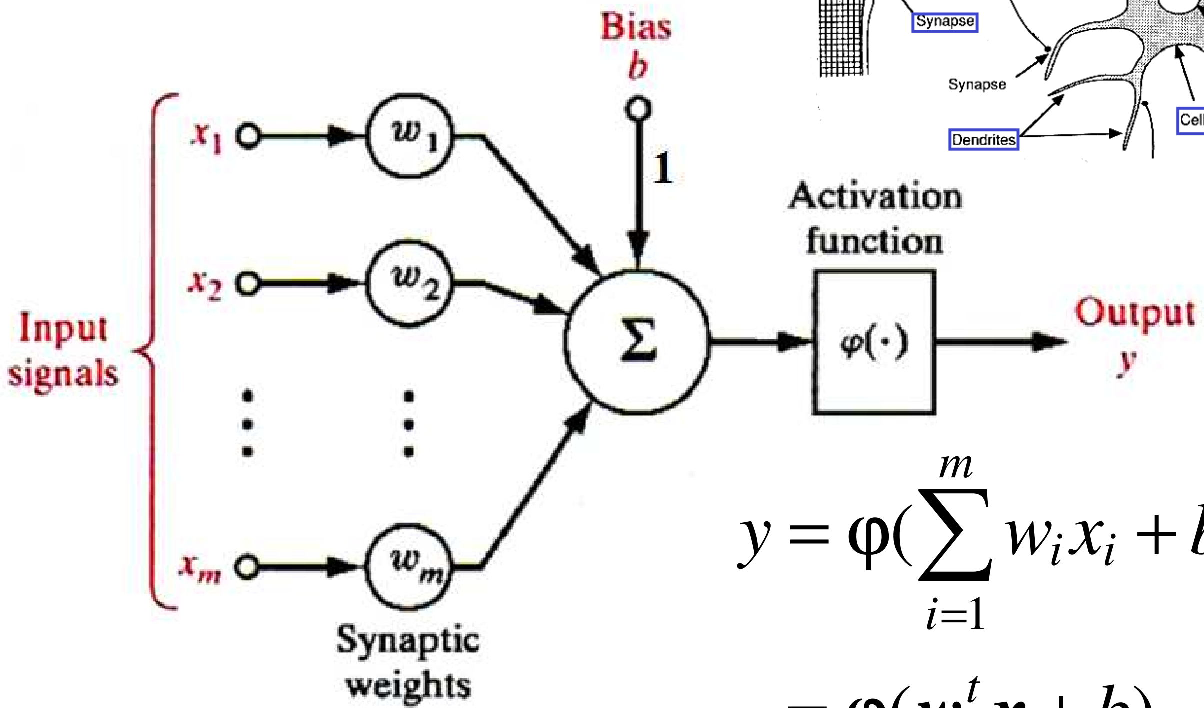
- Various neural systems with different characteristics and functions result from
  - i) different functions of neurons
  - ii) different ways of connections
  - iii) different flows of information

- Characteristics: nonlinearity, non-locality, non-algorithm, dynamics, adaptivity, fault-tolerance, input-output mapping, evidential response, self-organization

- Functions: learn, analyze, associate, organize, comprehend, recognize, plan, decide

◎ Neuron Model





$$y = \varphi\left(\sum_{i=1}^{m} w_i x_i + b\right)$$

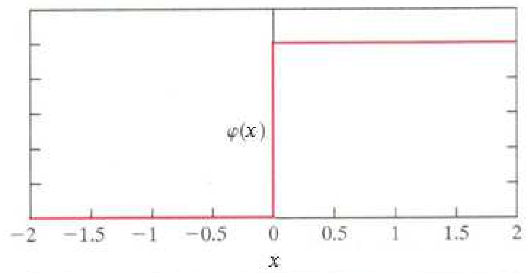
$$= \varphi(\boldsymbol{w}^t \boldsymbol{x} + b)$$

Bias term *b*: feedback, error, gain, adjustment

Types of activation function $\varphi(\cdot)$:

1. Threshold function (Heaviside function)

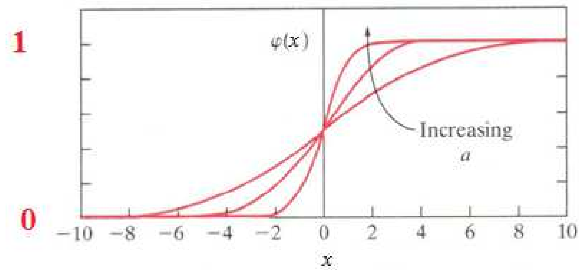
$$\varphi(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$



2. Sigmoid function

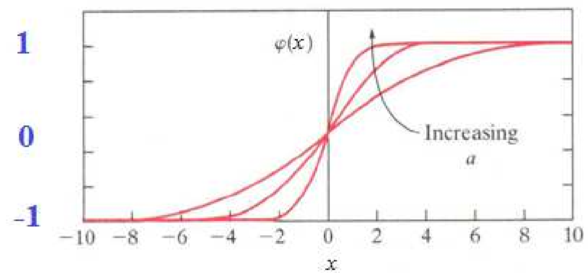
Logistic function

$$\varphi(x) = \frac{1}{1+\exp(-ax)}$$



Hyperbolic function

$$\varphi(x) = \tanh(ax)$$



3. Softsign function

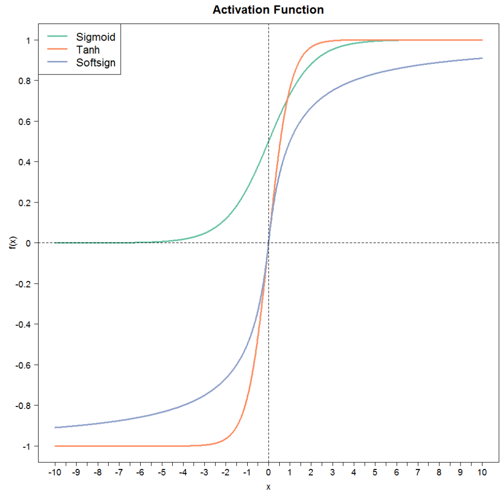
$$\varphi(x) = \frac{x}{1+|x|}$$

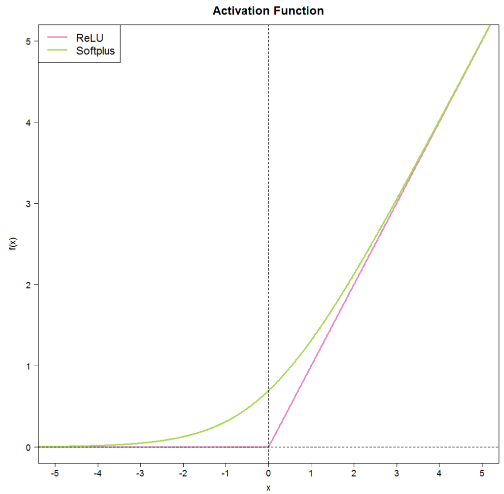
4. Softplus function

$$\varphi(x) = \ln(1+e^{x})$$

5. ReLU function

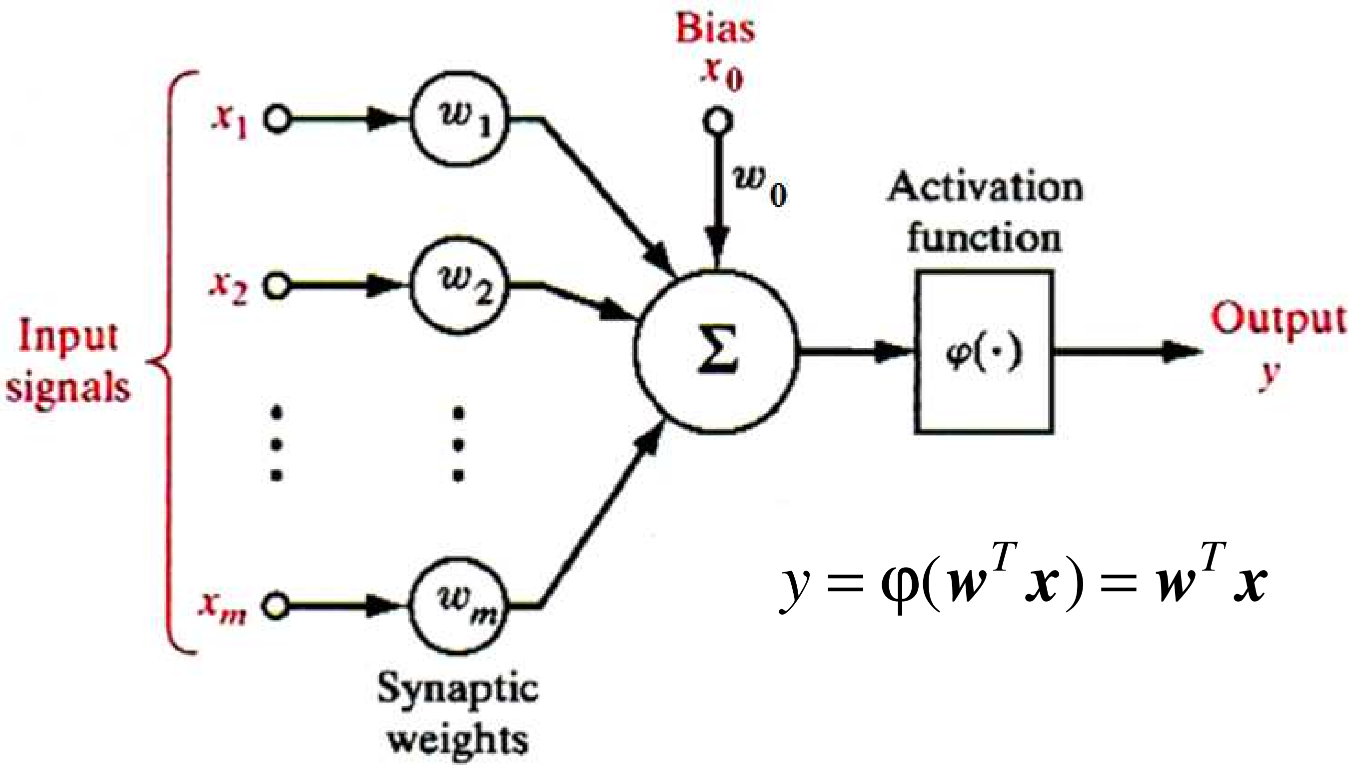
$$\varphi(x) = \max(\theta, x)$$





Perceptron: Neuron model with linear active function

$$\varphi(x) = x \quad \therefore \; y = \varphi(\boldsymbol{w}^T \boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$$



$$y = \varphi(\boldsymbol{w}^T \boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$$

11.4 Training a Perceptron

11.4.1 Least Mean Square (LMS) Learning

◎ Input vectors : $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_L\}$

Ideal outputs : $\{d_1, d_2, \cdots, d_L\}$

Actual outputs : $\{y_1, y_2, \cdots, y_L\}$

Mean square error:

$$\left\langle \varepsilon_k^2 \right\rangle = \frac{1}{L}\sum_{k=1}^{L} \varepsilon_k^2 = \left\langle \left(d_k - y_k\right)^2 \right\rangle = \left\langle \left(d_k - \boldsymbol{w}^T \boldsymbol{x}_k\right)^2 \right\rangle$$

$$= \left\langle d_k{}^2 \right\rangle + \boldsymbol{w}^T \left\langle \boldsymbol{x}_k \boldsymbol{x}_k^T \right\rangle \boldsymbol{w} - 2\left\langle d_k \boldsymbol{x}_k^T \right\rangle \boldsymbol{w} \quad \text{-- (2.4)}$$

Let $\xi = \left\langle \varepsilon_k^2 \right\rangle$, $\boldsymbol{p} = \left\langle d_k \boldsymbol{x}_k \right\rangle$, $R = \left\langle \boldsymbol{x}_k \boldsymbol{x}_k^T \right\rangle$: correlation matrix

$$(2.4) \;\Rightarrow\; \xi = \left\langle d_k^2 \right\rangle + \boldsymbol{w}^T R \boldsymbol{w} - 2\boldsymbol{p}^T \boldsymbol{w}$$

Idea: $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \xi(\boldsymbol{w})$

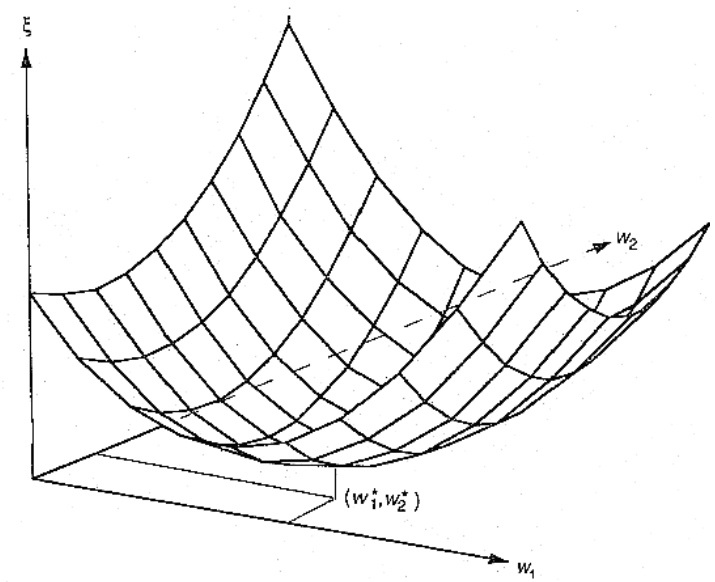
Let $\dfrac{d\xi(\boldsymbol{w})}{d\boldsymbol{w}} = 2R\boldsymbol{w} - 2\boldsymbol{p} = 0.$ Obtain $\boldsymbol{w}^* = R^{-1}\boldsymbol{p}.$

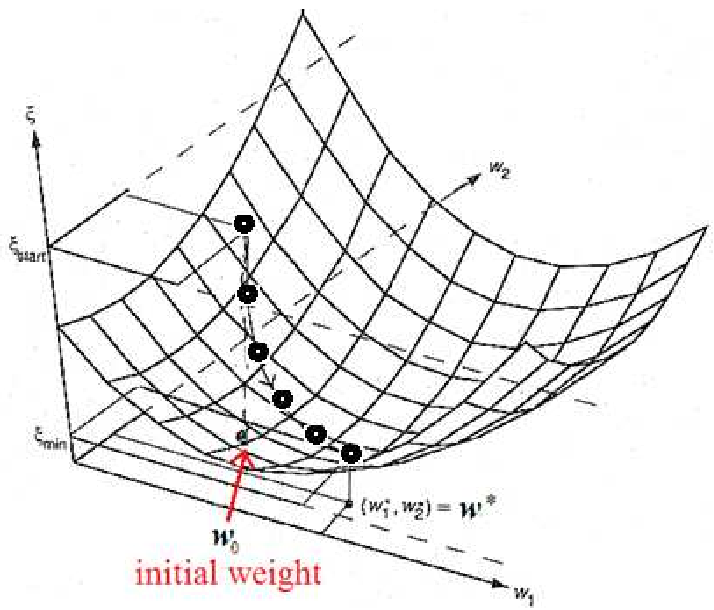
Practical difficulties of analytical formula :

1. Large dimensions – R^{-1} difficult to calculate
2. Need to calculate the expected values of $\boldsymbol{p} = \langle d_k \boldsymbol{x}_k \rangle,\ R = \left\langle \boldsymbol{x}_k \boldsymbol{x}_k^T \right\rangle$ – lack of the knowledge of probabilities

11.4.2 Gradient Descent (GD) Learning

The graph of $\xi(\boldsymbol{w}) = \left\langle d_k^2 \right\rangle + \boldsymbol{w}^T R \boldsymbol{w} - 2\boldsymbol{p}^T \boldsymbol{w}$ is a paraboloid.





Steps: 1. Initialize weight values $\boldsymbol{w}(t_0)$

2. Determine the steepest descent direction

$$-\nabla_{\boldsymbol{w}} \xi(\boldsymbol{w}(t)) = -\frac{d\xi(\boldsymbol{w}(t))}{d\boldsymbol{w}(t)} = 2(\boldsymbol{p} - R\boldsymbol{w}(t))$$

Let $\Delta\boldsymbol{w}(t) = -\nabla_{\boldsymbol{w}} \xi(\boldsymbol{w}(t)) = 2(\boldsymbol{p} - R\boldsymbol{w}(t))$

3. Modify weight values

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \mu\Delta\boldsymbol{w}(t), \quad \mu: \text{ step size}$$

4. Repeat 2~3.

- No calculation of R^{-1}

Drawbacks:

i) Need to calculate $\boldsymbol{p}$ and R,

ii) Steepest descent is a batch training method.

11.4.3 Stochastic Gradient Descent (SGD) Learning

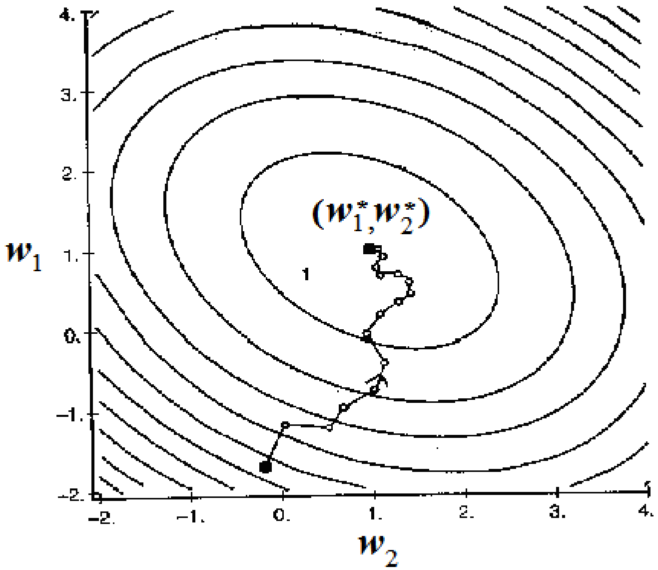
Approximate $-\nabla_{\boldsymbol{w}} \xi(\boldsymbol{w}(t)) = 2(\boldsymbol{p} - R\boldsymbol{w}(t))$ by randomly selecting one training example at a time

1. Apply an input vector $\boldsymbol{x}_k$
2. $\varepsilon_k^2(t) = (d_k - y_k)^2 = (d_k - \boldsymbol{w}^T(t) \cdot \boldsymbol{x}_k)^2$
3. $-\nabla_{\boldsymbol{w}} \xi(\boldsymbol{w}(t)) = -\nabla_{\boldsymbol{w}} \left\langle \varepsilon_k^2(t) \right\rangle \approx -\nabla_{\boldsymbol{w}} \varepsilon_k^2(t)$

$$= 2(d_k - \boldsymbol{w}^T(t) \cdot \boldsymbol{x}_k)\boldsymbol{x}_k = 2\varepsilon_k(t)\boldsymbol{x}_k$$

4. $\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \mu\Delta\boldsymbol{w}(t) = \boldsymbol{w}(t) + 2\mu\varepsilon_k(t)\boldsymbol{x}_k$
5. Repeat 1~4 with the next input vector

- No calculation of $\boldsymbol{p}$ and R



Drawback: time consuming.

Improvement: mini-batch training method.

- Practical Considerations:

  (a) No. of training vectors, (b) Stopping criteria

  (c) Initial weights, (d) Step size

11.4.4 Conjugate Gradient Descent (CGD) Learning

-- Drawback: can only minimize quadratic functions,

e.g., $f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T A\boldsymbol{w} - \boldsymbol{b}^T\boldsymbol{w} + c$ $\quad A = 2R$

- Adequate for our error function $\quad \boldsymbol{b}^T = 2\boldsymbol{p}^T$

$$\xi(\boldsymbol{w}) = \boldsymbol{w}^T R\boldsymbol{w} - 2\boldsymbol{p}^T\boldsymbol{w} + \left\langle d_k^2 \right\rangle \qquad c = \left\langle d_k^2 \right\rangle$$

Advantage: guarantees to find the optimum solution in at most n iterations, where n is the size of matrix A.

- The size of correlation matrix R is the dimension of input vectors $\boldsymbol{x}$.

A-Conjugate Vectors:

Let $A_{n\times n}$: square, symmetric, positive-definite matrix.

$\{\boldsymbol{s}(0),\ \boldsymbol{s}(1),\ \cdots,\ \boldsymbol{s}(n-1)\}$ are A-conjugate vectors if

$$\boldsymbol{s}^T(i)A\boldsymbol{s}(j)=0,\ \forall i\neq j$$

* If $A = I$ (identity matrix), conjugacy = orthogonality.

- CGD finding the $\boldsymbol{w}$ to minimize $f(\boldsymbol{w})$ is through

  $\boldsymbol{w}(i+1)=\boldsymbol{w}(i)+\eta(i)\boldsymbol{s}(i)$, where $i=0,\cdots,n-1$,

  (GD: $\boldsymbol{w}(i+1)=\boldsymbol{w}(i)-\eta(i)\nabla_{\boldsymbol{w}} f(\boldsymbol{w}(i))$)

  $\eta(i)$ is determined by $\min_{\eta} f(\boldsymbol{w}(i)+\eta\boldsymbol{s}(i))$.

$$\left(\because f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T A\boldsymbol{w} - \boldsymbol{b}^T\boldsymbol{w} + c \right.$$

$$f(\boldsymbol{w}(i) + \eta\boldsymbol{s}(i)) = \frac{1}{2}(\boldsymbol{w}(i) + \eta\boldsymbol{s}(i))^T A(\boldsymbol{w}(i) + \eta\boldsymbol{s}(i))$$

$$- \boldsymbol{b}^T(\boldsymbol{w}(i) + \eta\boldsymbol{s}(i)) + c$$

$$\text{Let } \frac{d}{d\eta}\left[f(\boldsymbol{w}(i) + \eta\boldsymbol{s}(i))\right] = 0$$

$$\frac{d}{d\eta}\left[f(\boldsymbol{w}(i) + \eta\boldsymbol{s}(i))\right] = \frac{d}{d\eta}\Big[\boldsymbol{w}(i)^T A\boldsymbol{w}(i) + \eta\boldsymbol{s}(i)^T A\boldsymbol{w}(i)$$

$$+\eta\boldsymbol{w}(i)^T A\boldsymbol{s}(i) + \eta^2\boldsymbol{s}(i)^T A\boldsymbol{s}(i) - \boldsymbol{b}^T\boldsymbol{w}(i) - \eta\boldsymbol{b}^T\boldsymbol{s}(i)\Big]$$

$$= \boldsymbol{s}(i)^T A\boldsymbol{w}(i) + \boldsymbol{w}(i)^T A\boldsymbol{s}(i) + 2\eta\boldsymbol{s}(i)^T A\boldsymbol{s}(i) - \boldsymbol{b}^T\boldsymbol{s}(i) = 0$$

$$\left. \eta = \frac{\boldsymbol{b}^T\boldsymbol{s}(i) - (\boldsymbol{s}(i)^T A\boldsymbol{w}(i) + \boldsymbol{w}(i)^T A\boldsymbol{s}(i))}{2\boldsymbol{s}(i)^T A\boldsymbol{s}(i)} \right) \text{ ---- (A)}$$

How to determine $\boldsymbol{s}(i)$?

$$f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T A\boldsymbol{w} - \boldsymbol{b}^T\boldsymbol{w} + c \;\Rightarrow\; -\nabla_{\boldsymbol{w}} f(\boldsymbol{w}) = \boldsymbol{b} - A\boldsymbol{w}$$

Define $\boldsymbol{r}(i) = -\nabla_{\boldsymbol{w}} f(\boldsymbol{w}) = \boldsymbol{b} - A\boldsymbol{w}(i)$ --- (B)

Let $\boldsymbol{s}(0) = \boldsymbol{r}(0) = \boldsymbol{b} - A\boldsymbol{w}(0)$, $\boldsymbol{w}(0)$: arbitrary vector

$\boldsymbol{s}(i) = \boldsymbol{r}(i) + \alpha(i)\boldsymbol{s}(i-1),\ i = 1,\cdots,n-1$ --- (C)

To determine $\alpha(i)$, multiply (C) by $\boldsymbol{s}(i\text{-}1)A$,

$\boldsymbol{s}^T(i-1)A\boldsymbol{s}(i) = \boldsymbol{s}^T(i-1)A(\boldsymbol{r}(i) + \alpha(i)\boldsymbol{s}(i-1))$ --- (D)

In order to be A-conjugate: $\boldsymbol{s}^T(i)A\boldsymbol{s}(j) = 0,\ \forall i \neq j$

(D) $\Rightarrow$ $0 = \boldsymbol{s}^T(i-1)A\boldsymbol{r}(i) + \alpha(i)\boldsymbol{s}^T(i-1)A\boldsymbol{s}(i-1).$

$$\alpha(i) = -\frac{\boldsymbol{s}^T(i-1)A\boldsymbol{r}(i)}{\boldsymbol{s}^T(i-1)A\boldsymbol{s}(i-1)} \quad \text{--- (D)}$$

Consider the error function:

$$\xi(\boldsymbol{w}) = \boldsymbol{w}^T R\boldsymbol{w} - 2\boldsymbol{p}^T\boldsymbol{w} + \left\langle d_k^2 \right\rangle$$

Comparing with $f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T A\boldsymbol{w} - \boldsymbol{b}^T\boldsymbol{w} + c,$

$A = 2R,\ \boldsymbol{b}^T = 2\boldsymbol{p}^T,\ c = \left\langle d_k^2 \right\rangle$

From (A) $\eta(i) = \dfrac{\boldsymbol{b}^T\boldsymbol{s}(i) - (\boldsymbol{s}(i)^T A\boldsymbol{w}(i) + \boldsymbol{w}(i)^T A\boldsymbol{s}(i))}{2\boldsymbol{s}(i)^T A\boldsymbol{s}(i)},$

$$\eta(i) = \frac{\boldsymbol{p}^T\boldsymbol{s}(i) - (\boldsymbol{s}(i)^T R\boldsymbol{w}(i) + \boldsymbol{w}(i)^T R\boldsymbol{s}(i))}{2\boldsymbol{s}(i)^T R\boldsymbol{s}(i)}.$$

From (B) $\boldsymbol{r}(i) = \boldsymbol{b} - A\boldsymbol{w}(i),$

$$\boldsymbol{r}(i) = 2(\boldsymbol{p} - R\boldsymbol{w}(i)).$$

From (C) $\boldsymbol{s}(0) = \boldsymbol{r}(0) = 2(\boldsymbol{p} - R\boldsymbol{w}(0)),$

$$\boldsymbol{s}(i) = \boldsymbol{r}(i) + \alpha(i)\boldsymbol{s}(i-1).$$

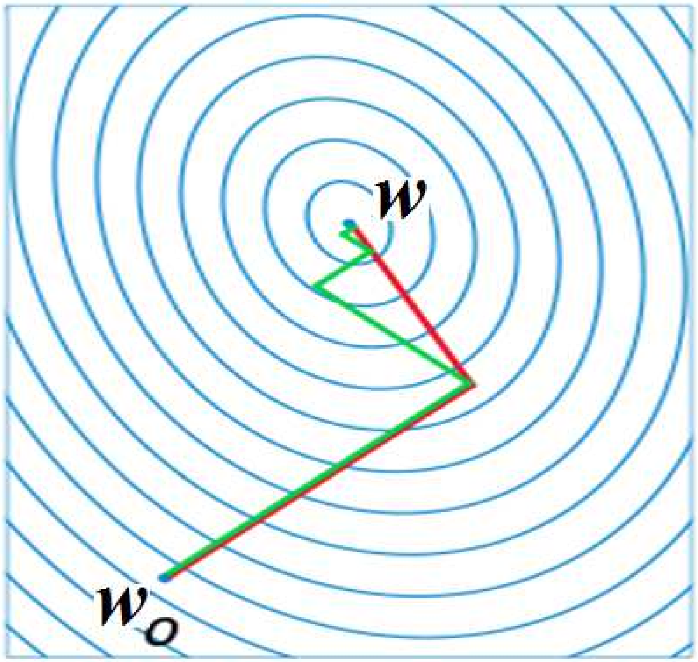
From (D) $\alpha(i) = -\dfrac{\boldsymbol{s}^{\boldsymbol{T}}(i-1)A\boldsymbol{r}(i)}{\boldsymbol{s}^{\boldsymbol{T}}(i-1)A\boldsymbol{s}(i-1)},$

$$\alpha(i) = -\frac{\boldsymbol{s}^{\boldsymbol{T}}(i-1)R\boldsymbol{r}(i)}{\boldsymbol{s}^{\boldsymbol{T}}(i-1)R\boldsymbol{s}(i-1)}.$$

Update rule: $\boldsymbol{w}(i+1) = \boldsymbol{w}(i) + \eta(i)\boldsymbol{s}(i),\ i = 0,1,\cdots,n-1$

$\boldsymbol{w}(0)$: arbitrary starting vector

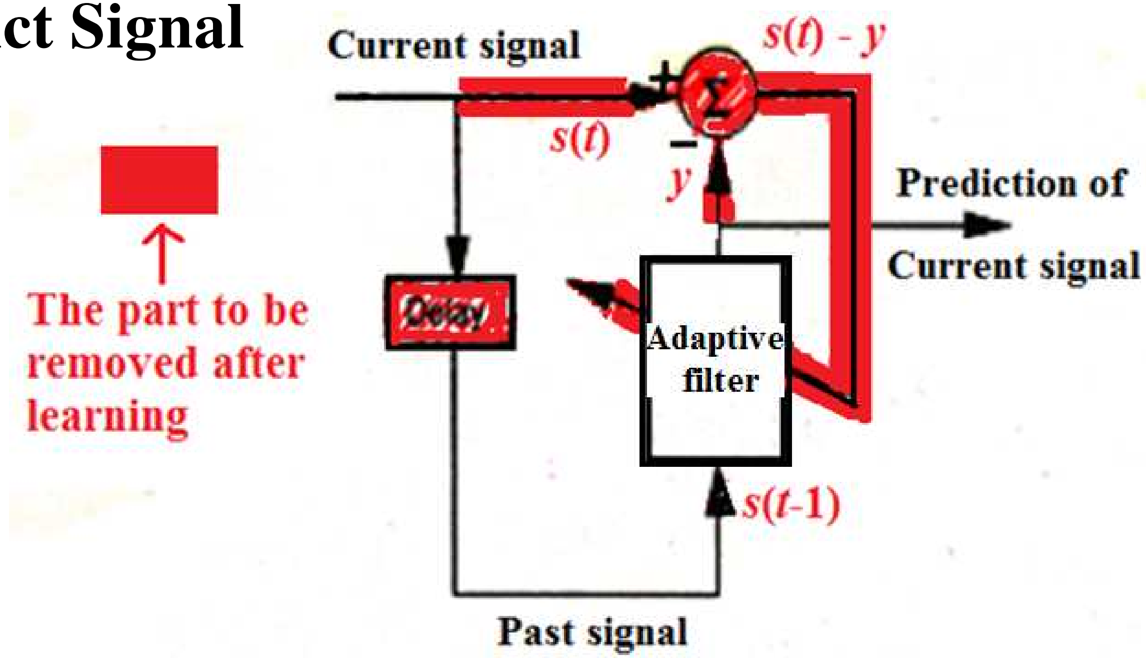
Example: A comparison of the convergences of gradient descent (green) and conjugate gradient (red) for minimizing a quadratic function.



Conjugate gradient converges in at most *n* steps where *n* is the size of the matrix of the system (here *n*=2).

11.5. Applications

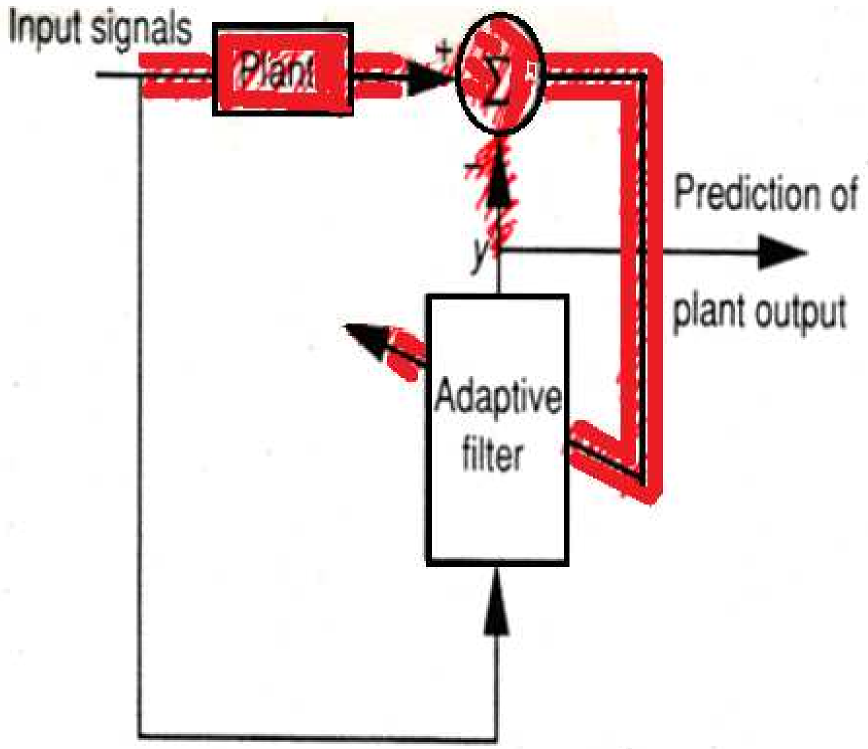
11.5.1 Predict Signal



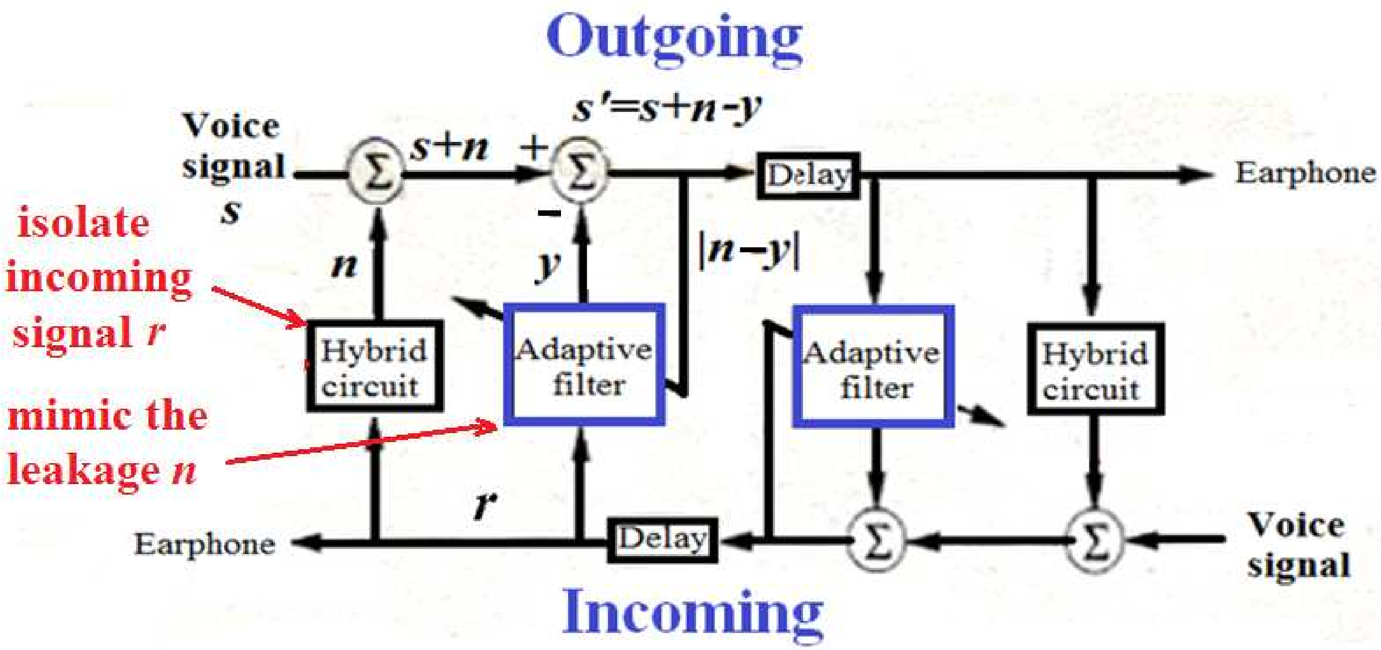
An adaptive filter is trained to predict signal. The Signal used to train the filter is a delayed actual signal. The expected output is the current signal.

11.5.2 Reproduce Signal

An adaptive filter is used to model a plant. Inputs to the filter are the same as those to the plant. The filter adjusts its weights based on the difference between its output and the output of the plant.



11.5.3. Echo Cancellation in Telephone Circuits



s : outgoing voice, r : incoming voice

n : noise (leakage of r)

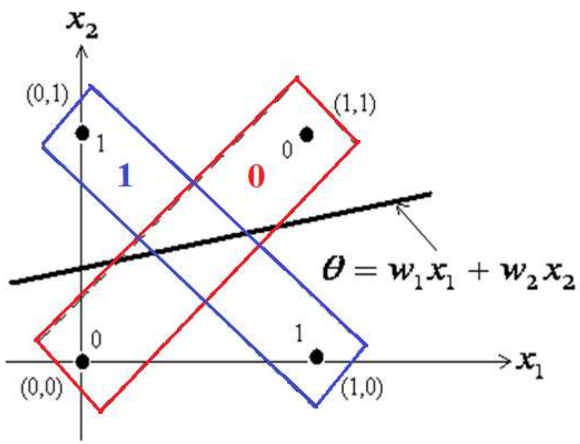
y : the output of the filter mimics n

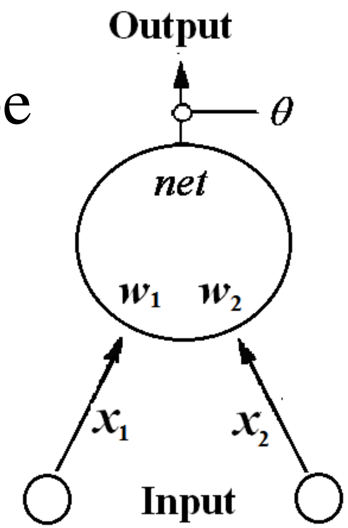
11.5 Multilayer Perceptrons (MLP)

○ XOR function

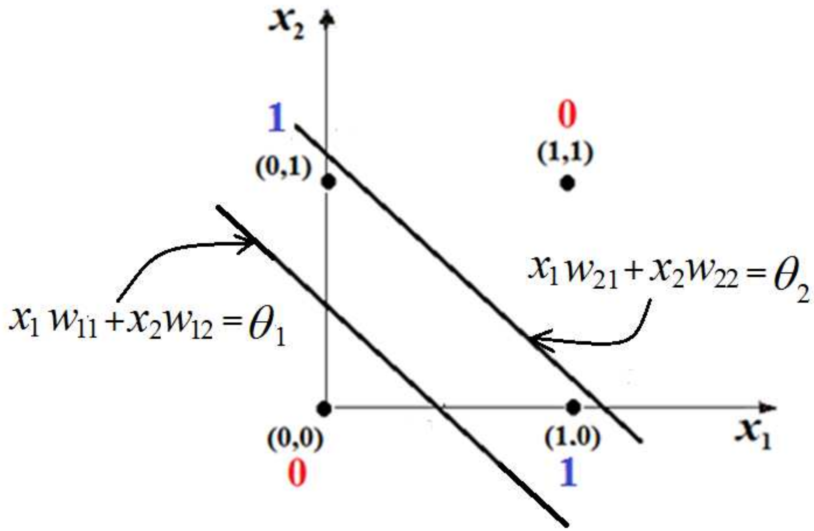
x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

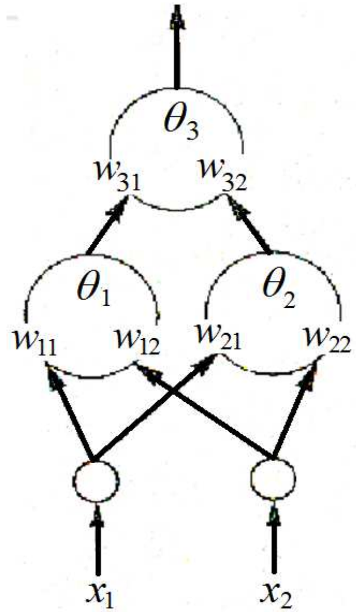
This problem cannot be solved by an adaline.





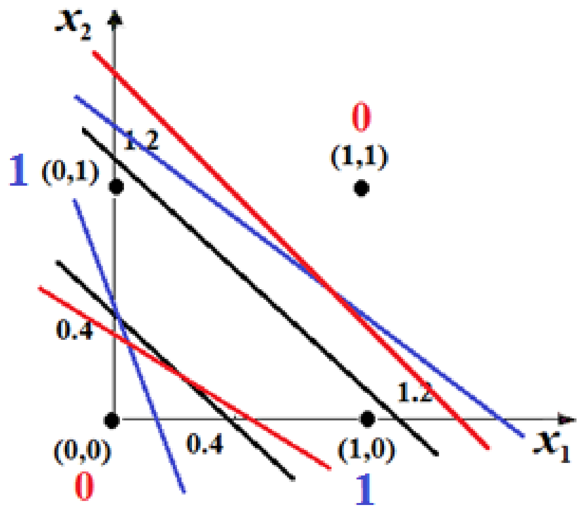
Reason: $w_1x_1 + w_2x_2 = \theta$ specifies is a line in the (x_1, x_2) plane.



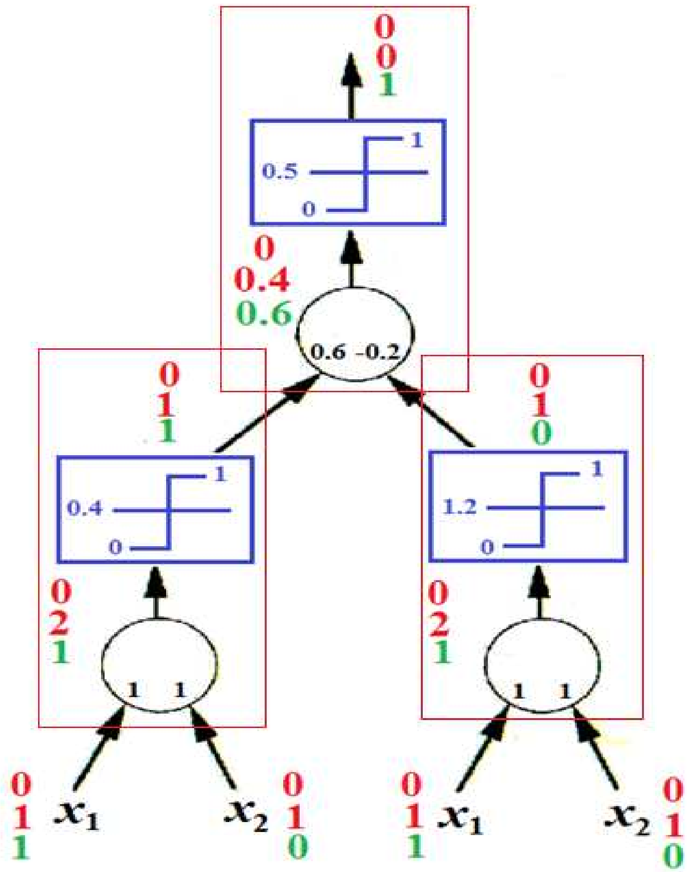


The two neurons in the hidden layer provides two lines that can separate the plane into three regions. The two regions containing (0,0) and (1,1) are associated with the network output of 0. The central region is associated with the network output of 1.

There are many solution pairs of lines.

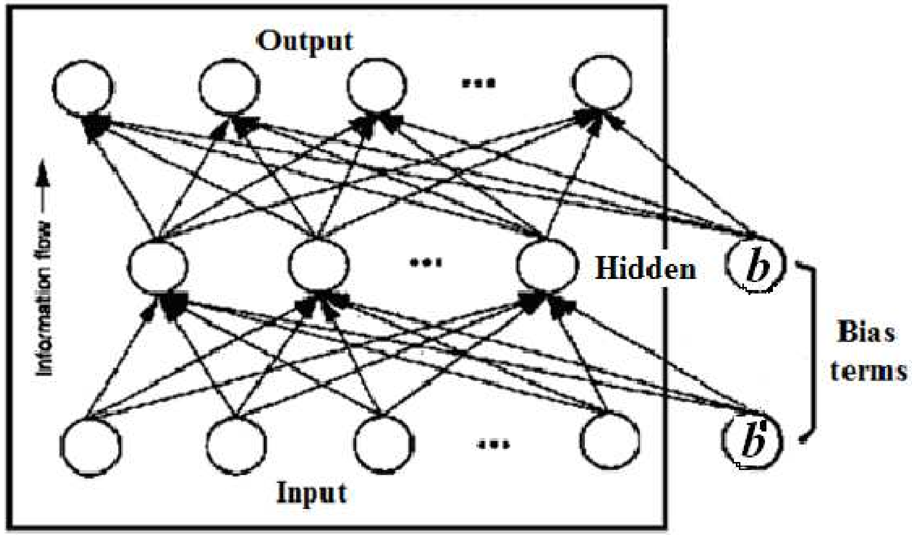


One solution:



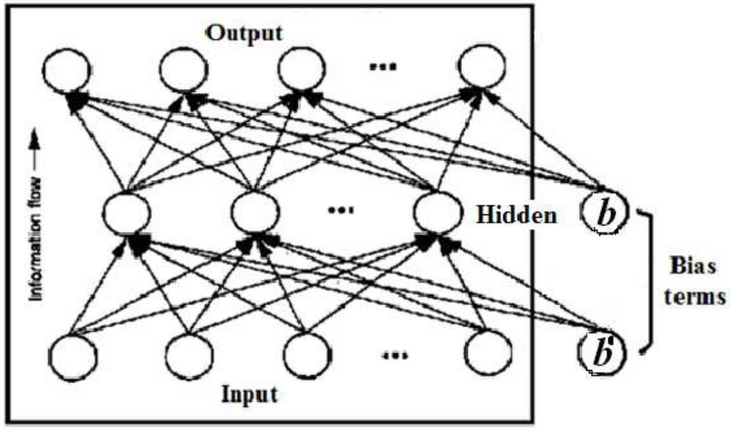
11.6 Train MLP by Backpropagation

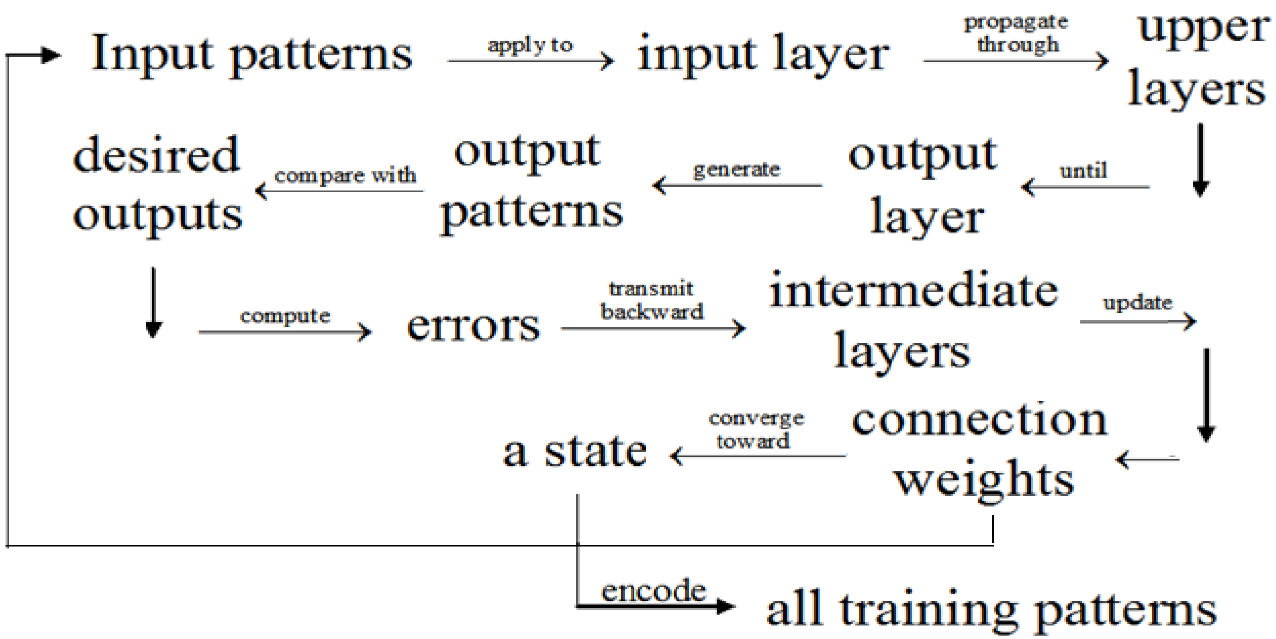
- **Architecture of MLP**



During training, self-organization of nodes on the intermediate layers s.t. different nodes recognize different features or their relationships. Noisy and incomplete patterns can thus be handled.

- **Learning cycle**





Given training examples: $(\boldsymbol{x}_1, \boldsymbol{y}_1), \cdots, (\boldsymbol{x}_p, \boldsymbol{y}_p), \cdots, (\boldsymbol{x}_P, \boldsymbol{y}_P)$

where $\boldsymbol{y}_p = \Phi(\boldsymbol{x}_p)$, find an $\mathrm{NN}(\boldsymbol{\theta})$ approximating Φ

where $\boldsymbol{\theta} = (w_{ij}^l, b_i^l)_{i,j,l}$. Consider input $\boldsymbol{x} = (x_1, x_2, \cdots, x_N)$

Net input to the

*j*th hidden unit:

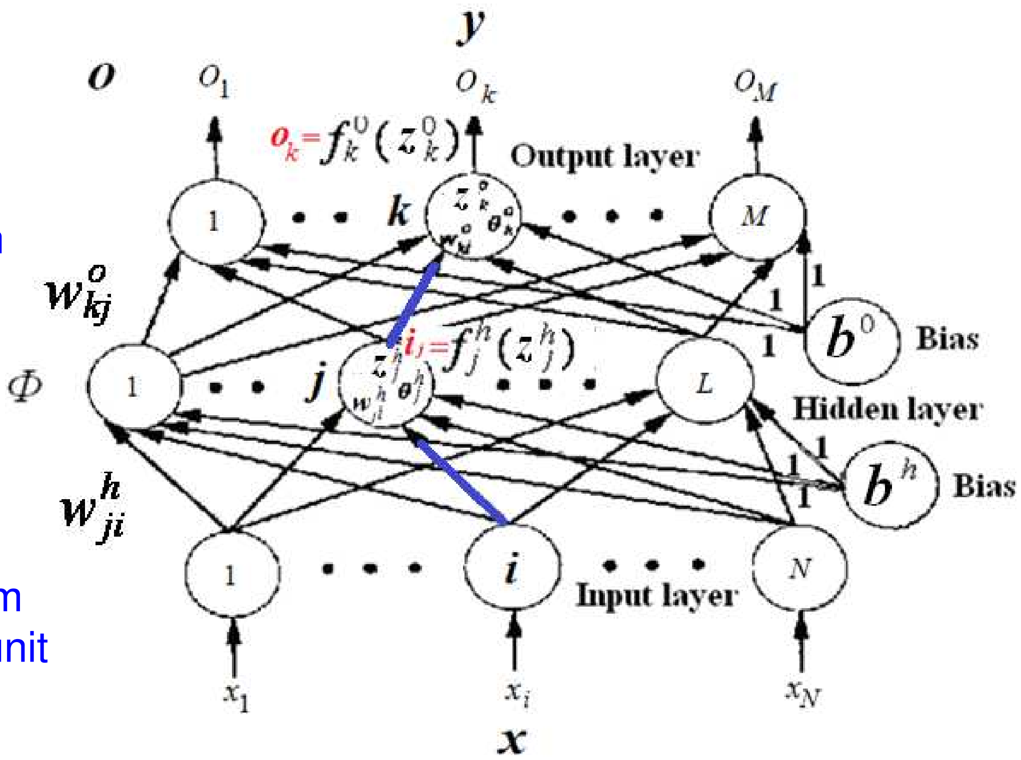
$$z_j^h = \sum_{i=1}^{N} w_{ji}^h x_i + b_j^h$$

hidden layer

*j*th hidden unit

*i*th input unit

bias term with *j*th unit



Output of the *j*th hidden unit: $i_j = f_j^h(z_j^h)$

transfer function

Net input to the *k*th output unit:

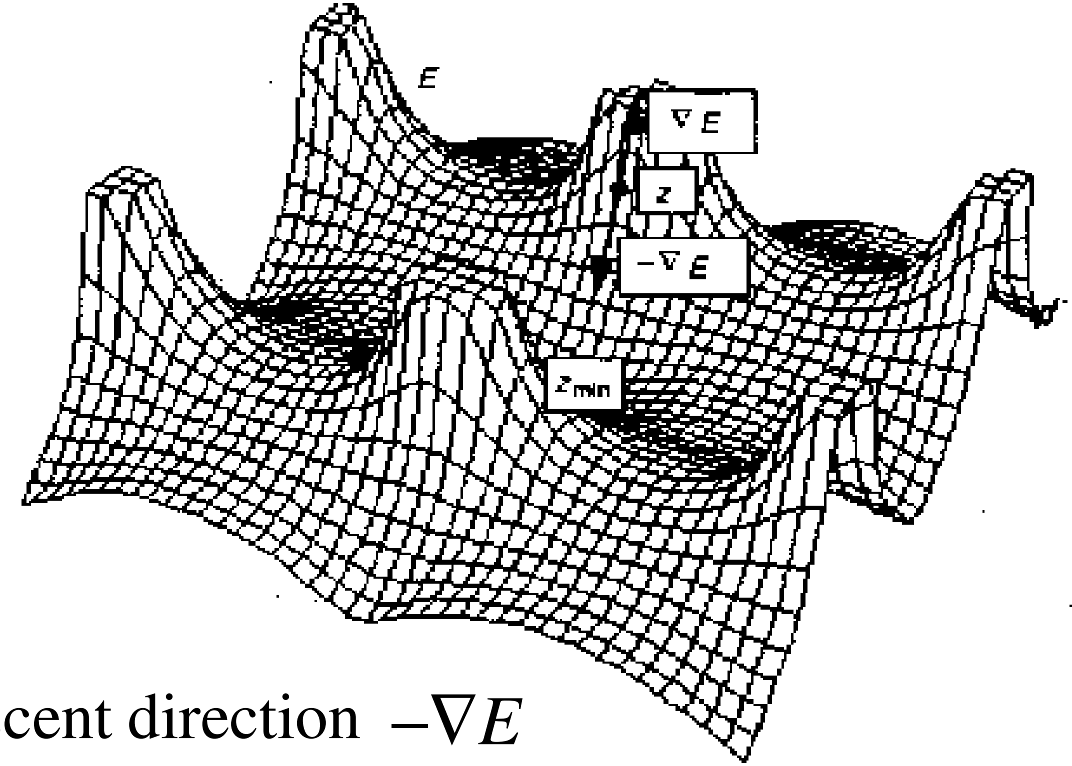
$$z_k^o = \sum_{j=1}^{L} w_{kj}^o i_j + b_k^o$$

Output of the *k*th output unit: $o_k = f_k^o(z_k^o)$

- Update of output layer weights W^o

The error to be minimized:

$$E = \frac{1}{2}\sum_{k=1}^{M}(y_k - o_k)^2,$$ where M: # output units



The descent direction $-\nabla E$

The learning rule:

$$W^{o}(t+1) = W^{o}(t) + \Delta W^{o} = W^{o}(t) - \eta \nabla E$$

where: η learning rate

- Chain Rule: $\dfrac{\partial f(g(x))}{\partial x} = \dfrac{\partial f(g)}{\partial g}\dfrac{\partial g(x)}{\partial x} = f'g'$

$$\frac{\partial f(g(h(x)))}{\partial x} = \frac{\partial f(g)}{\partial g}\frac{\partial g(h)}{\partial h}\frac{\partial h(x)}{\partial x} = f'g'h'$$

- Determine ∇E

$$E = \frac{1}{2}\sum_{k=1}^{M}(y_k - o_k)^2,\ \ \frac{\partial E}{\partial w_{kj}^o} = -(y_k - o_k)\frac{\partial o_k}{\partial w_{kj}^o},\ o_k = f_k^o(z_k^o)$$

$$\frac{\partial o_k}{\partial w_{kj}^o} = \frac{\partial f_k^o(z_k^o)}{\partial w_{kj}^o} = \frac{\partial f_k^o(z_k^o)}{\partial z_k^o}\frac{\partial z_k^o}{\partial w_{kj}^o} = f_k^{o\prime}(z_k^o)\frac{\partial z_k^o}{\partial w_{kj}^o}$$

$$\frac{\partial z_k^o}{\partial w_{kj}^o} = \frac{\partial}{\partial w_{kj}^o}\left(\sum_{j=1}^{L} w_{kj}^o i_j + b_j^o\right) = i_j \quad \therefore\ -\frac{\partial E}{\partial w_{kj}^o} = (y_k - o_k)f_k^{o\prime} i_j$$

The weights on the output layer are updated as

$$w_{kj}^{o}(t+1) = w_{kj}^{o}(t) + \Delta w_{kj}^{o}(t) = w_{kj}^{o}(t) - \eta \frac{\partial E}{\partial w_{kj}^{o}}$$

$$= w_{kj}^{o}(t) + \eta (y_k - o_k) f_k^{o\prime} i_j \quad \text{--- (A)}$$

- Consider $f_k^{o\prime}$

  Two forms for the output functions f_k^{o}

  i) Linear $\quad f_k^{o}(x) = x$

  ii) Sigmoid $\quad f_k^{o}(x) = (1 + e^{-\lambda x})^{-1}$ or

  $$f_k^{o}(x) = \frac{1}{2}[1 - \tanh(\lambda x)]$$

- Sigmoid:

$$f(x) = \frac{1}{1+e^{-x}} \quad \Rightarrow \quad \frac{df}{dx} = \frac{e^{-x}}{(1+e^{-x})^2}$$

- Tanh:

$$f(x) = \frac{1-e^{-2x}}{1+e^{-2x}} \quad \Rightarrow \quad \frac{df}{dx} = 1 - \left(\frac{1-e^{-2x}}{1+e^{-2x}} \right)^2$$

- Softsign:

$$f(x) = \frac{x}{1+|x|} \quad \Rightarrow \quad \frac{df}{dx} = \frac{1}{(1+|x|)^2}$$

i) For linear function $f_k^o(x) = x,\ f_k^{o\prime} = 1$

$$(A) \Rightarrow\ w_{kj}^o(t+1) = w_{kj}^o(t) + \eta(y_k - o_k)i_j$$

ii) For sigmoid function $f_k^o(x) = (1+e^{-\lambda x})^{-1}$,

$$f_k^{o\prime}(z_k^o) = -(1+e^{-\lambda z_k^o})^{-2}\frac{\partial e^{-\lambda z_k^o}}{\partial z_k^o}$$

$$= -(1+e^{-\lambda z_k^o})^{-2} e^{-\lambda z_k^o}(-\lambda) = \lambda(1+e^{-\lambda z_k^o})^{-2} e^{-\lambda z_k^o}$$

$$= \lambda f_k^{o2}(f_k^{o\ -1} - 1) = \lambda f_k^o(1 - f_k^o) = \lambda o_k(1 - o_k)$$

$$(A) \Rightarrow w_{kj}^o(t+1) = w_{kj}^o(t) + \eta\lambda(y_k - o_k)o_k(1 - o_k)i_j$$

◎ Updates of hidden-layer weights

Difficulty: Unknown outputs of the hidden-layer units

Idea: Relate error E to the output of the hidden layer

$$E(w_{ji}^h) = \frac{1}{2}\sum_{k=1}^{M}(y_k - o_k)^2 = \frac{1}{2}\sum_k [y_k - f_k^o(z_k^o)]^2$$

$$= \frac{1}{2}\sum_k [y_k - f_k^o(\sum_{j=1}^{L} w_{kj}^o i_j + b_k^o)]^2$$

$$= \frac{1}{2}\sum_k [y_k - f_k^o(\sum_{j=1}^{L} w_{kj}^o f_j^h(z_j^h) + b_k^o)]^2$$

$$= \frac{1}{2}\sum_k [y_k - f_k^o(\sum_{j=1}^{L} w_{kj}^o f_j^h(\sum_{i=1}^{N} w_{ji}^h x_i + b_j^h) + b_k^o)]^2$$

$$\frac{\partial E}{\partial w_{ji}^h} = \frac{1}{2}\frac{\partial}{\partial w_{ji}^h}[\sum_k (y_k - o_k)^2] = \frac{1}{2}\sum_k [\frac{\partial}{\partial w_{ji}^h}(y_k - o_k)^2]$$

$$= -\sum_k (y_k - o_k)\frac{\partial o_k}{\partial w_{ji}^h}$$

$$\frac{\partial o_k}{\partial w_{ji}^h} = \frac{\partial f_k^o(z_k^0)}{\partial w_{ji}^h} = \frac{\partial f_k^o(z_k^o)}{\partial z_k^o}\frac{\partial z_k^o}{\partial i_j}\frac{\partial i_j}{\partial w_{ji}^h}$$

$$= f_k^{o\prime}(z_k^o)\frac{\partial z_k^o}{\partial i_j}\frac{\partial i_j}{\partial w_{ji}^h}$$

$$\frac{\partial i_j}{\partial w_{ji}^h} = \frac{\partial f_j^h(z_j^h)}{\partial w_{ji}^h} = \frac{\partial f_j^h(z_j^h)}{\partial z_j^h}\frac{\partial z_j^h}{\partial w_{ji}^h} = f_j^{h\prime}(z_j^h)\frac{\partial z_j^h}{\partial w_{ji}^h}$$

$$\frac{\partial o_k}{\partial w_{ji}^h} = f_k^{o\prime}(z_k^o)\frac{\partial z_k^o}{\partial i_j} f_j^{h\prime}(z_j^h)\frac{\partial z_j^h}{\partial w_{ji}^h}$$

$$z_k^o = \sum_j w_{kj}^o i_j + b_k^o, \quad \frac{\partial z_j^o}{\partial i_j} = w_{kj}^o$$

$$z_j^h = \sum_i w_{ji}^h x_i + b_j^h, \quad \frac{\partial z_j^h}{\partial w_{ji}^h} = x_i$$

$$\frac{\partial o_k}{\partial w_{ji}^h} = f_k^{\prime o}(z_k^o) w_{kj}^o f_j^{\prime h}(z_j^h) x_i$$

Consider sigmoid output function $f(x) = (1 + e^{-\lambda x})^{-1}$

$$f_k^{\prime o}(z_k^o) = \lambda o_k (1 - o_k), \;\; f_j^{\prime h}(z_j^h) = \lambda i_j (1 - i_j)$$

$$\frac{\partial o_k}{\partial w_{ji}^h} = \lambda^2 o_k (1 - o_k) w_{kj}^o i_j (1 - i_j) x_i$$

$$\frac{\partial E}{\partial w_{ji}^h} = -\sum_k (y_k - o_k) \frac{\partial o_k}{\partial w_{ji}^h}$$

$$= -\lambda^2 \sum_k (y_k - o_k) o_k (1 - o_k) w_{kj}^o i_j (1 - i_j) x_i$$

$$= -\lambda^2 i_j (1 - i_j) x_i \sum_k (y_k - o_k) o_k (1 - o_k) w_{kj}^o$$

$$\Delta w_{ji}^h = -\eta \frac{\partial E}{\partial w_{ji}^h} = -\eta' i_j (1 - i_j) x_i \sum_k (y_k - o_k) o_k (1 - o_k) w_{kj}^o$$

where $\eta' = \eta \lambda^2$.

The update rule: $w_{ji}^h(t+1) = w_{ji}^h(t) + \Delta w_{ji}^h$

- Summary

1. Apply the input vector, $\boldsymbol{x}_p = (x_{p1}, x_{p2}, \ldots, x_{pN})^t$

   to the input units.

2. Calculate the net-input values to the hidden layer units:

$$z_{pj}^{h} = \sum_{i=1}^{N} w_{ji}^{h} x_{pi} + b_j^{h}$$

3. Calculate the outputs from the hidden layer: $i_{pj} = f_j^{h}(z_{pj}^{h})$

4. Calculate the net-input values to each output unit:

$$z_{pk}^{o} = \sum_{j=1}^{L} w_{kj}^{o} i_{pj} + b_k^{o}$$

5. Calculate the outputs: $o_{pk} = f_k^{o}(z_{pk}^{o})$

6. Calculate the error terms for the output units:

$$\delta^o_{pk} = (y_{pk} - o_{pk}) f^{o\prime}_k$$

7. Calculate the error terms for the hidden units:

$$\delta^h_{pj} = f^{h\prime}_j \sum_k \delta^o_{pk} w^o_{kj}$$

8. Update weights on the output layer:

$$w^o_{kj}(t+1) = w^o_{kj}(t) + \eta \delta^o_{pk} i_{pj}$$

9. Update weights on the hidden layer:

$$w^h_{ji}(t+1) = w^h_{ji}(t) + \eta \delta^h_{pj} x_i$$

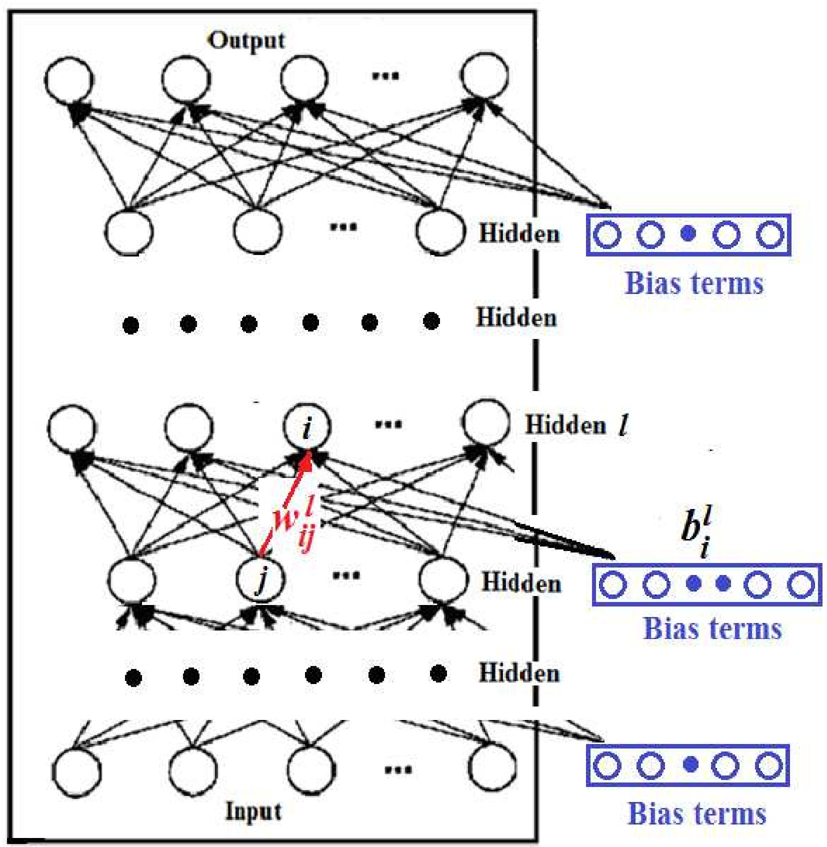
◎ Updates of multiple hidden-layers

$$E = \frac{1}{2}\sum_{k=1}^{M}(y_k - o_k)^2,\ \ o_k = f_k^o(z_k^o),$$

$$z_k^o = \sum_{j=1}^{L} w_{kj}^o i_j + b_k^o,\ \ i_j = f_j^h(z_j^h)$$

$$z_k^o = \sum_{j=1}^{L} w_{kj}^o f_j^h(z_j^h) + b_k^o,$$

$$o_k = f_k^o\left(\sum_{j=1}^{L} w_{kj}^o f_j^h(z_j^h) + b_k^o\right)$$



$$E(w_{kj}^{o}) = \frac{1}{2}\sum_{k=1}^{M}\left(y_k - f_k^{o}\left(\sum_{j=1}^{L} w_{kj}^{o} f_j^{h}(z_j^{h}) + b_k^{o} \right)\right)^2$$

$$\left(z_j^{h} = \sum_{i=1}^{N} w_{ji}^{h} x_i + b_j^{h} \right)$$

$$= \frac{1}{2}\sum_{k}[y_k - f_k^{o}(\sum_{j=1}^{L} w_{kj}^{o} f_j^{h}(\sum_{i=1}^{N} w_{ji}^{h} x_i + b_j^{h}) + b_k^{o})]^2$$

$$E(w_{ji}^{h}) = \frac{1}{2}\sum_{k}[y_k - f_k^{o}(\sum_{j=1}^{L} w_{kj}^{o} f_j^{h}(\sum_{i=1}^{N} w_{ji}^{h} x_i + b_j^{h}) + b_k^{o})]^2$$

$$E(w_{j_1 i}^{h_1}) = \frac{1}{2}\sum_{k}[y_k - f_k^{o}(\sum_{j_1=1}^{L_1} w_{kj_1}^{o} f_{j_1}^{h_1}(\sum_{i=1}^{N} w_{j_1 i}^{h_1} x_i + b_{j_1}^{h_1}) + b_k^{o})]^2$$

$$E(w^{h_2}_{j_2 j_1}) = \frac{1}{2}\sum_k [y_k - f^o_k(\sum_{j_1=1}^{L_1} w^o_{kj_1} f^{h_1}_{j_1}(\sum_{j_2=1}^{L_2} w^{h_1}_{j_1 j_2} f^{h_2}_{j_2}(\sum_{i=1}^{N} w^{h_2}_{j_2 i} x_i + b^{h_2}_{j_2}) + b^{h_1}_{j_1}) + b^o_k)]^2$$

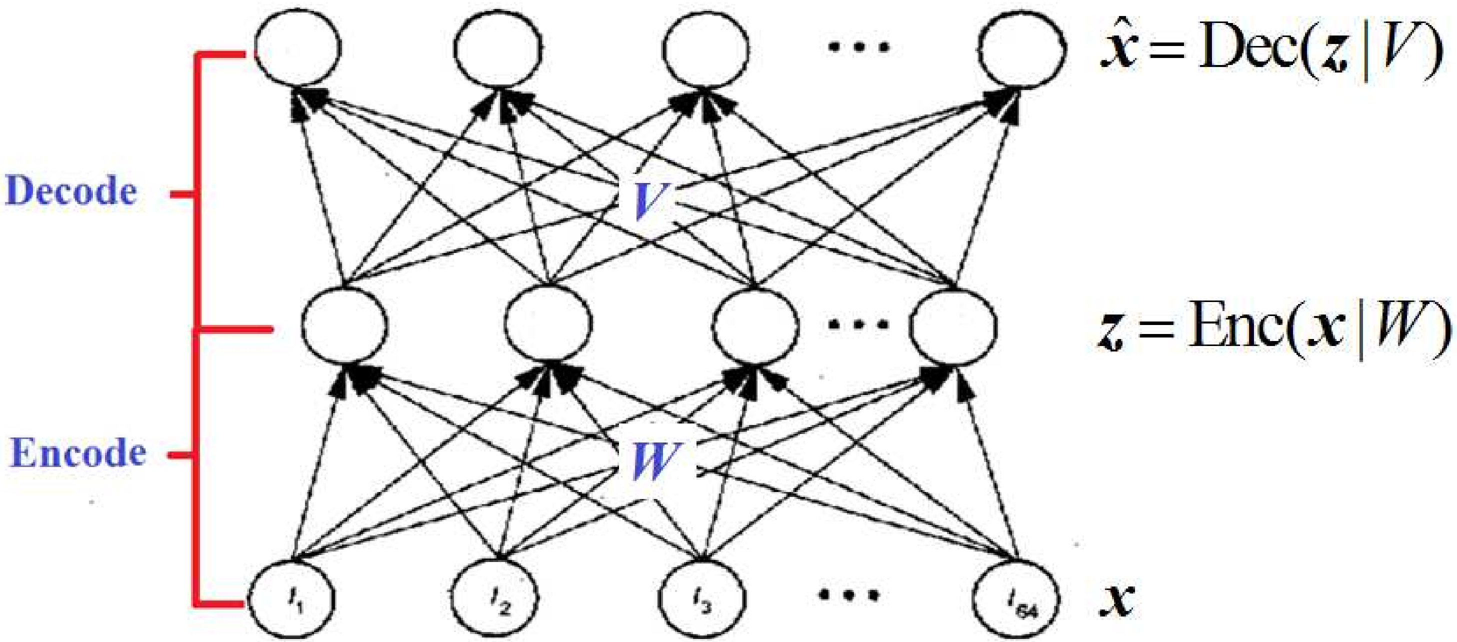
$$\bullet \bullet$$

$$\bullet \bullet$$

$$E(w^{h_n}_{j_n j_{n-1}}) = \frac{1}{2}\sum_k [y_k - \bullet$$

Loss functions: Error function, cross-entropy, differential entropy, negative log-likelihood, information gain

11.7 Autoencoders



Variants: Denoising autoencoder

Sparse autoencoder

11.8 ANNs

Hopfield neural model

Associative memory

Counterpropagation neural networks

Self-organization feature map

Adaptive resonance theory

Neocognitron

Recurrent neural networks

Boltzmann machine

Support vector machine

Kernel machines