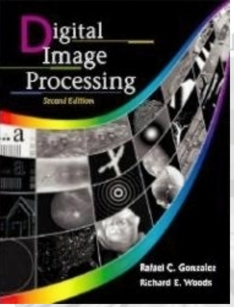


Chapter 12 Object Recognition

- Images regions are treated as objects or patterns
- Object recognition → pattern recognition
- Pattern recognition:

  (a) decision-theoretic: quantitative descriptor, i.e., length, area, texture.

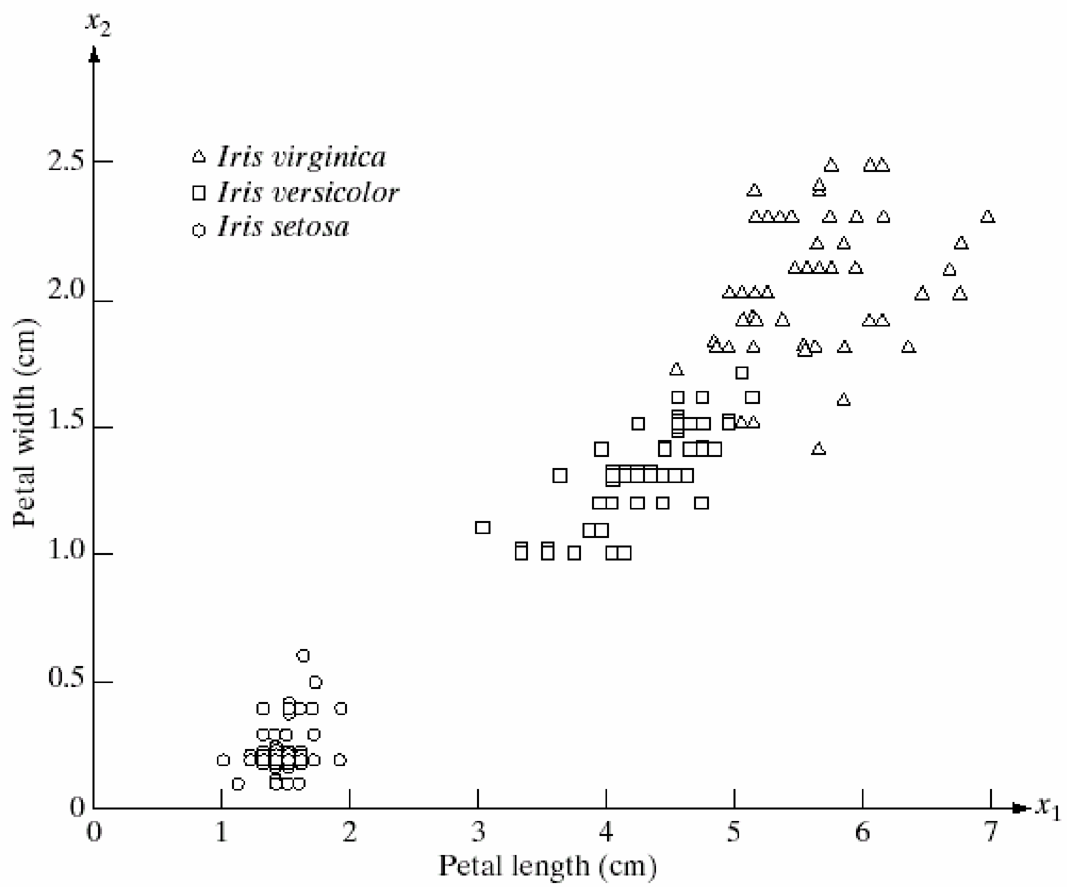
  (b) Structural: qualitative descriptor, i.e., relational descriptor.

2.1 Patterns and Pattern Classes

- A pattern is an arrangement of descriptors.
- Feature is used to denote a descriptor.
- A pattern class is a family of patterns that share some common properties.
- Pattern recognition → assign patterns to their respective classes.
- Three common pattern arrangements: vectors, strings and trees.
- Pattern vectors are $\mathbf{x}=[x_1, x_2, \ldots x_n]$ where each component x_i represent the ith descriptor and n is the total number of descriptors.

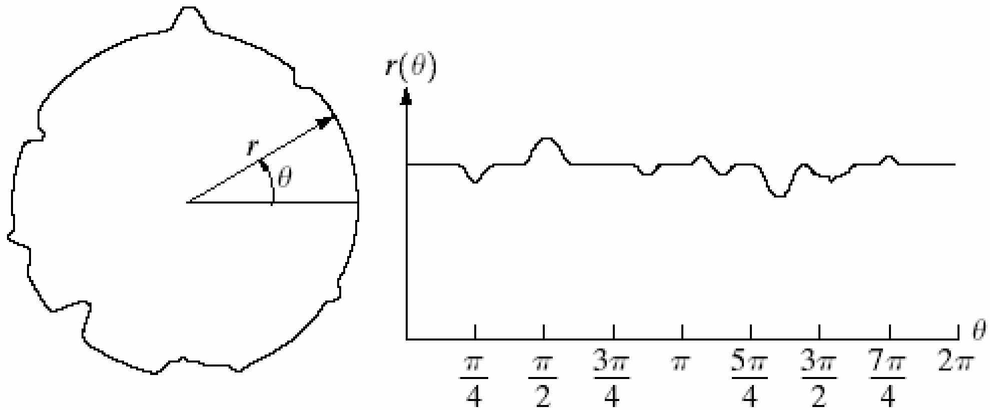
2.1 Patterns and Pattern Classes

FIGURE 12.1
Three types of iris flowers described by two measurements.



The flower is described by two measurement $\mathbf{x}=[x_1, x_2]$ where x_1 and x_2 corresponding to petal length and width

2.1 Patterns and Pattern Classes



a b

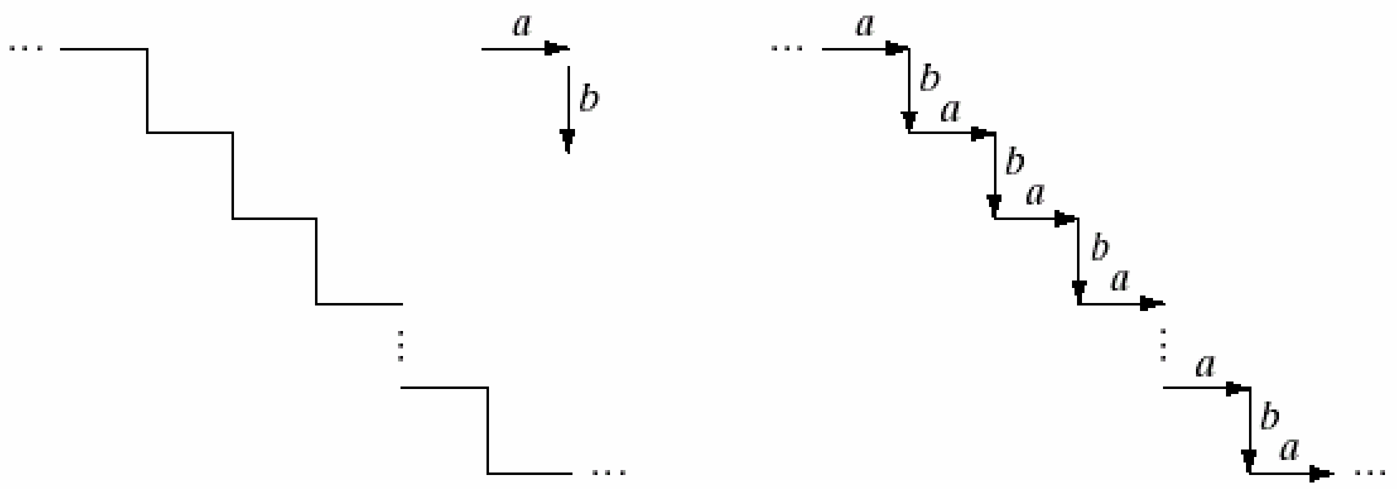
FIGURE 12.2 A noisy object and its corresponding signature.

We represent each object by its signature, and form the pattern vectors by letting x_1=r(θ_1) x_2=r(θ_2) ,……x_n=r(θ_n).

2.1 Patterns and Pattern Classes

- Pattern classes are characterized by ***quantitative Information*** or ***structural relationships***
- ***i.e.***, in *fingerprint recognition***:** interrelationship of ***print features*** called ***minutiae.***
- ***Minutiae*** and their relative size and location are used as primitive components to describe the ridge property, *i.e.*, ending, branching, and merging,…

2.1 Patterns and Pattern Classes



a b

FIGURE 12.3 (a) Staircase structure. (b) Structure coded in terms of the primitives *a* and *b* to yield the string description . . . *ababab* . . . .

2.1 Patterns and Pattern Classes

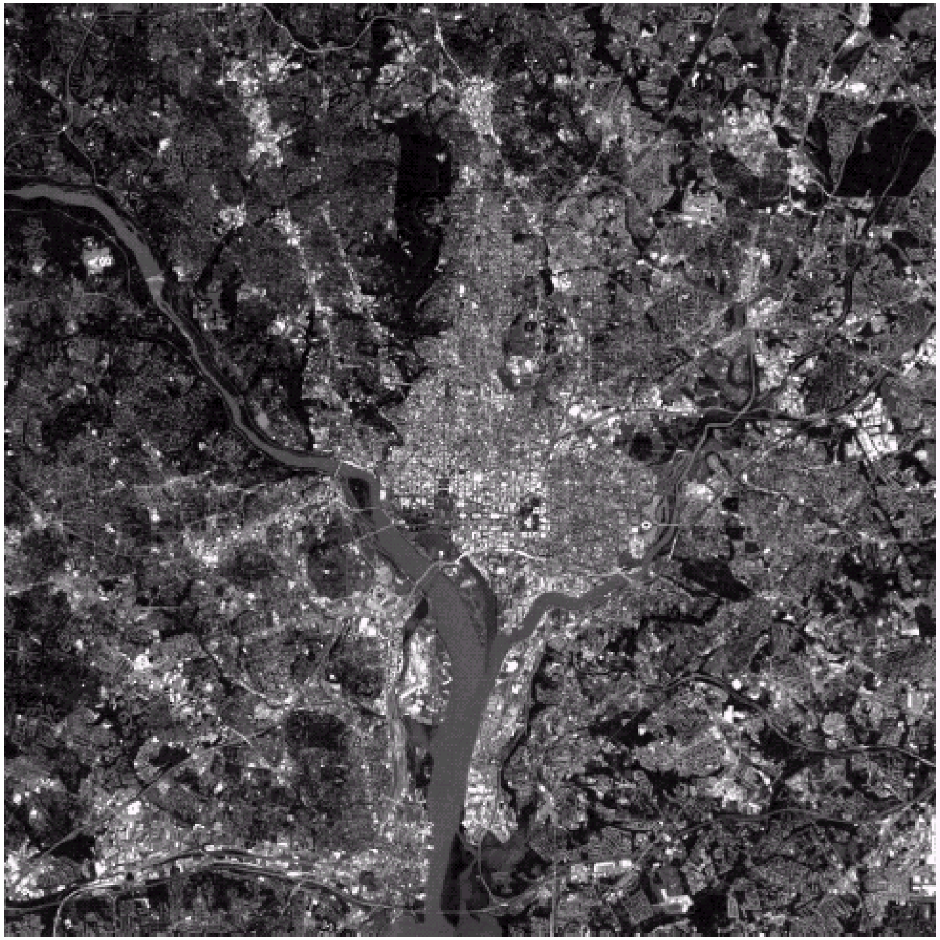
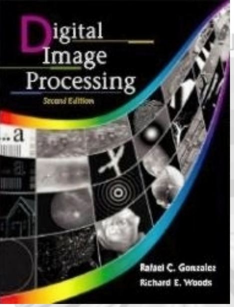


FIGURE 12.4 Satellite image of a heavily built downtown area (Washington, D.C.) and surrounding residential areas. (Courtesy of NASA.)



2.1 Patterns and Pattern Classes

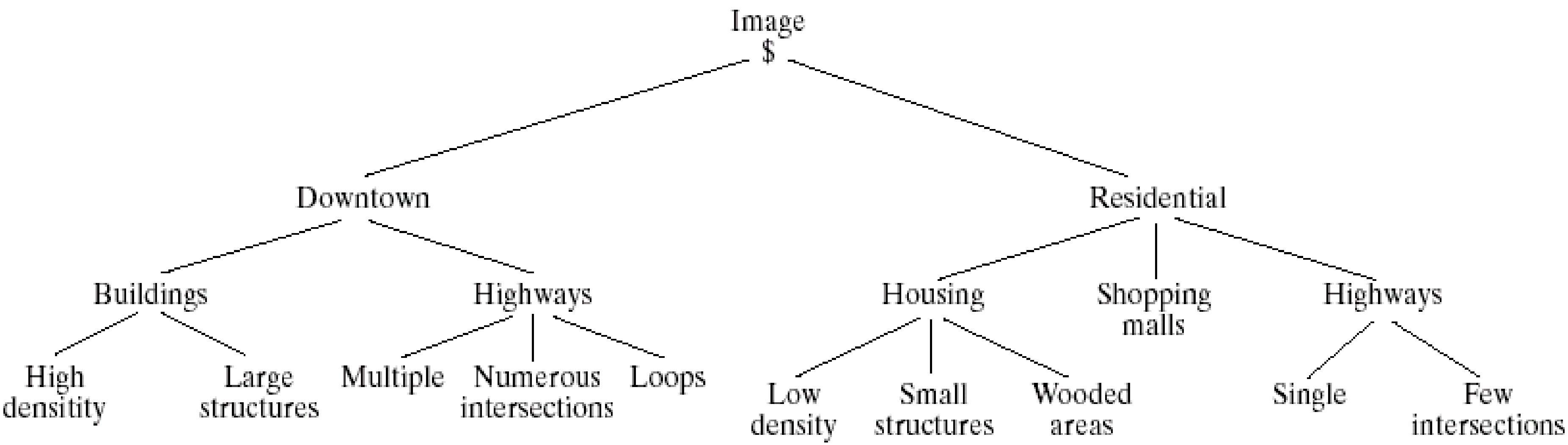


FIGURE 12.5 A tree description of the image in Fig. 12.4.

12.2 Recognition based on decision-theoretic methods

- Decision-theoretic approaches to recognition are based on the use of ***decision*** (or ***discriminant***) ***function***.
- Let $\mathbf{x}=[x_1, x_2, \ldots x_n]$ represent an n-dimensional pattern vector. For W pattern classes $\omega_1, \ldots \omega_W$, find the W decision functions $d_1(\mathbf{x}), d_2(\mathbf{x}), \ldots d_W(\mathbf{x})$ with the property that , if a pattern $\mathbf{x}$ belongs to class $\omega_i, \ldots$ then

$$d_i(\mathbf{x}) > d_j(\mathbf{x}), \ldots \text{for } j=1, 2 \ldots .W;\ j \neq i$$

- The ***decision boundary*** separating class ω_i from ω_j, is given by values of $\mathbf{x}$ for which $d_i(\mathbf{x}) = d_j(\mathbf{x})$.

12.2.1 Matching

- Recognition techniques represent each class by a prototype pattern vector.
- The simple approach is the minimum-distance classifier, which compute the (Euclidean) distance between the unknown and each of the prototype vectors.
- It choose the smallest distance to make a decision

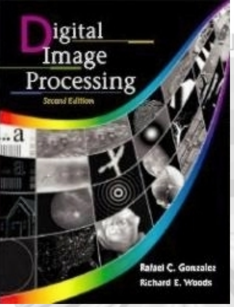
12.2.1 Matching

- ***Minimum distance classifier***
- The prototype of each pattern class to be the mean vector of the patterns of that class: $$\mathbf{m}_j = \frac{1}{N_j}\sum_{\mathbf{x}\in\omega_j}\mathbf{x}$$

  where N_j is the number of patterns from class j.
- Using the Euclidean distance to determine the closeness as the distance measure

  $$D_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{m}_j\| \quad j=1, 2, \ldots. W$$

  where $\|\mathbf{a}\| = (\mathbf{a}^T\cdot\mathbf{a})^{1/2}$ is the Euclidean distance norm.
- We then assign $\mathbf{x}$ to class ω_i if $D_j(\mathbf{x})$ is the smallest distance.



12.2.1 Matching

- Selecting the smallest distance is equivalent to evaluating the functions $$d_j(\mathbf{x}) = \mathbf{x}^T\mathbf{m}_j - \frac{1}{2}\mathbf{m}_j^T\mathbf{m}_j$$ and assigning **x** to class ω_i if $d_j(\mathbf{x})$ yields the largest numerical value.
- The decision boundary between ω_i and ω_j for a minimum distance classifier is

$$d_{ij}(\mathbf{x}) = d_i(\mathbf{x}) - d_j(\mathbf{x}) = \mathbf{x}^T(\mathbf{m}_i - \mathbf{m}_j) - \frac{1}{2}(\mathbf{m}_i - \mathbf{m}_j)^T(\mathbf{m}_i - \mathbf{m}_j)$$

- It is a surface indicating a perpendicular bisector (a line or a surface) of the line segment joining $\mathbf{m}_i$ and $\mathbf{m}_j$.

12.2.1 Matching

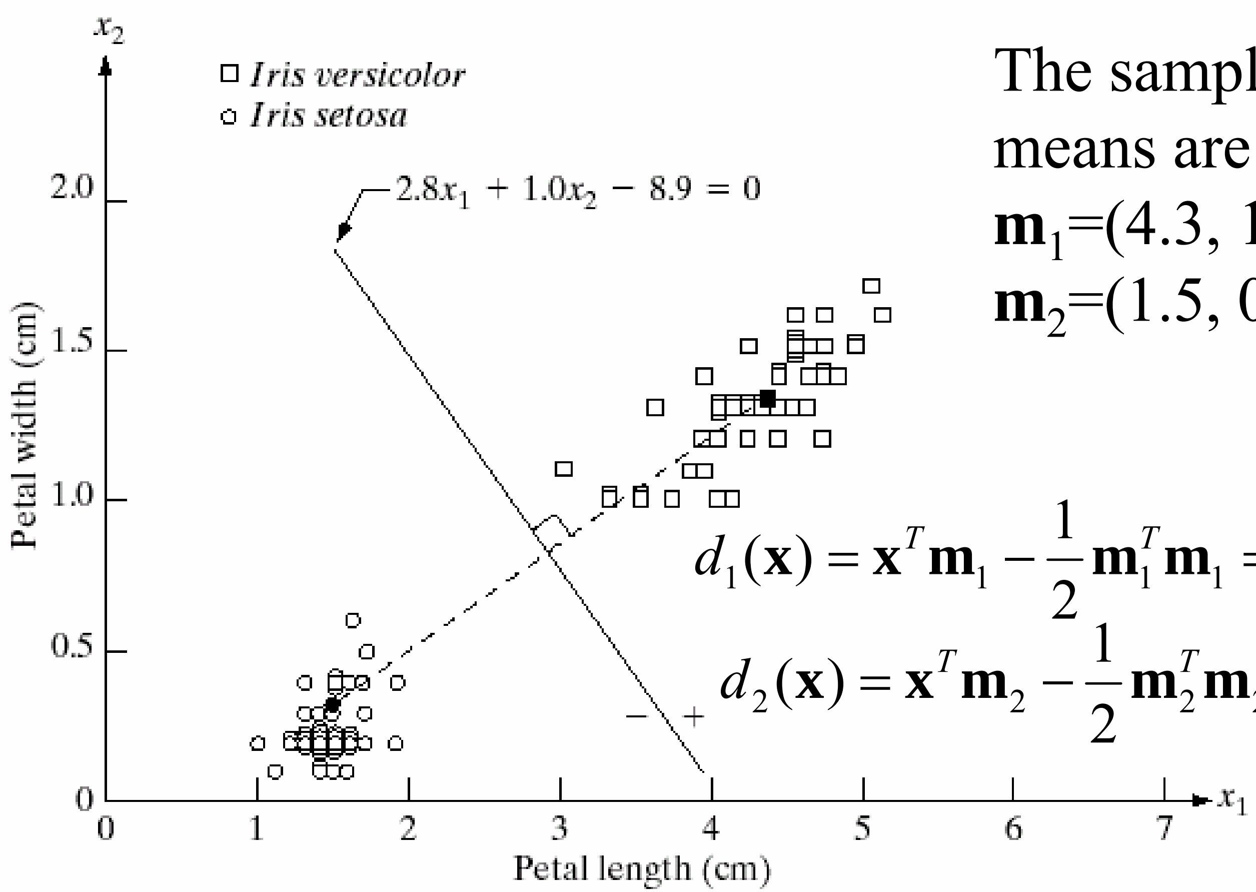


FIGURE 12.6 Decision boundary of minimum distance classifier for the classes of *Iris versicolor* and *Iris setosa*. The dark dot and square are the means.

The sample means are

$\mathbf{m}_1=(4.3, 1.3)^T$

$\mathbf{m}_2=(1.5, 0.3)^T$

$$d_1(\mathbf{x}) = \mathbf{x}^T\mathbf{m}_1 - \frac{1}{2}\mathbf{m}_1^T\mathbf{m}_1 = 4.3x_1 + 1.3x_2 - 10.1$$

$$d_2(\mathbf{x}) = \mathbf{x}^T\mathbf{m}_2 - \frac{1}{2}\mathbf{m}_2^T\mathbf{m}_2 = 1.5x_1 + 0.3x_2 - 1.17$$

Equ. of boundary: $d_{12}(\mathbf{x}) = d_1(\mathbf{x}) - d_2(\mathbf{x}) = 2.8x_1 + 1.0x_2 - 8.9 = 0$

12.2.1 Matching

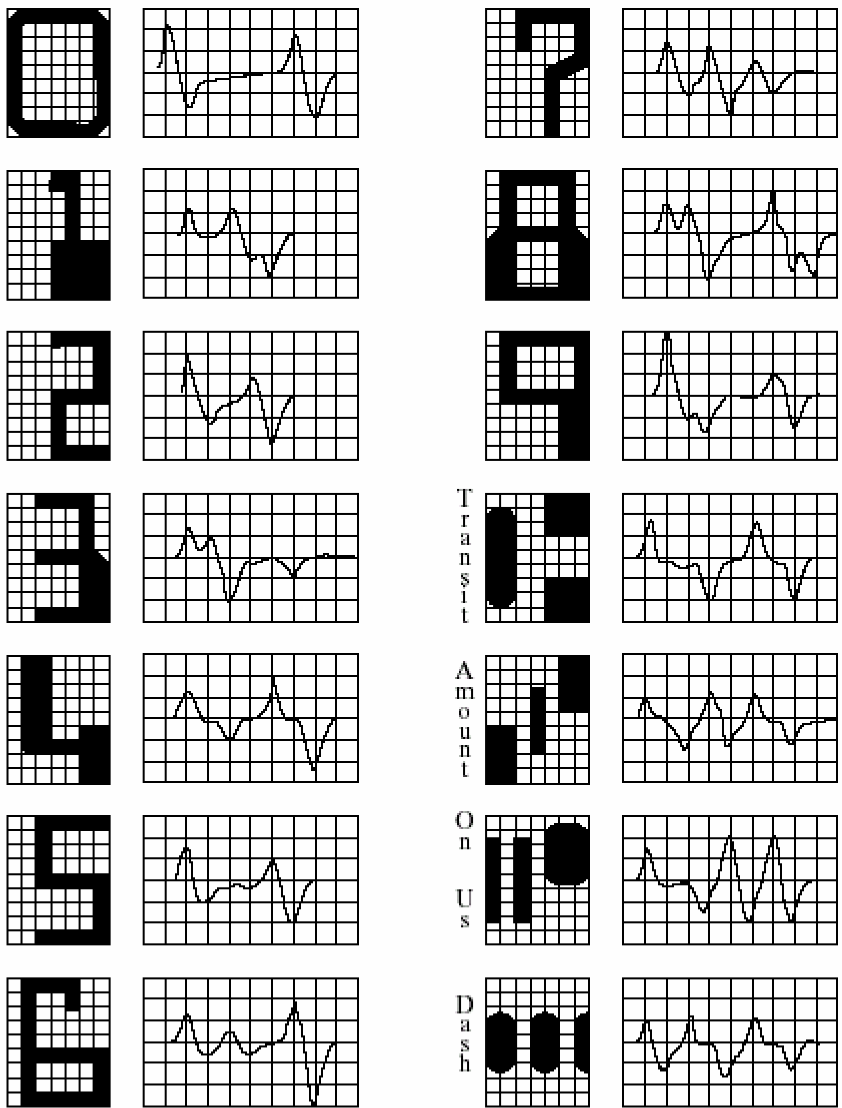


FIGURE 12.7 American Bankers Association E-13B font character set and corresponding waveforms.

12.2.1 Matching

- ***Matching by correlation***
- The correlation between $f(x, y)$ and $w(x, y)$ is

  $c(x, y)=\Sigma\Sigma f(s, t)w(x+s, y+t)$

  for x=0, 1, 2,….M-1 and y=0, 1, 2,…N-1.
- The correlation $c(x, y)$ is sensitive to the changes in the amplitude of f and w, a normalization is applied on the $c(x, y)$ as

$$\gamma(x,y)=\frac{\sum_{s}\sum_{t}\left[f(s,t)-\overline{f}(s,t)\right]\left[w(x+s,y+t)-\overline{w}\right]}{\left\{\sum_{s}\sum_{t}\left[f(s,t)-\overline{f}(s,t)\right]^{2}\left[w(x+s,y+t)-\overline{w}\right]^{2}\right\}^{1/2}}$$

12.2.1 Matching

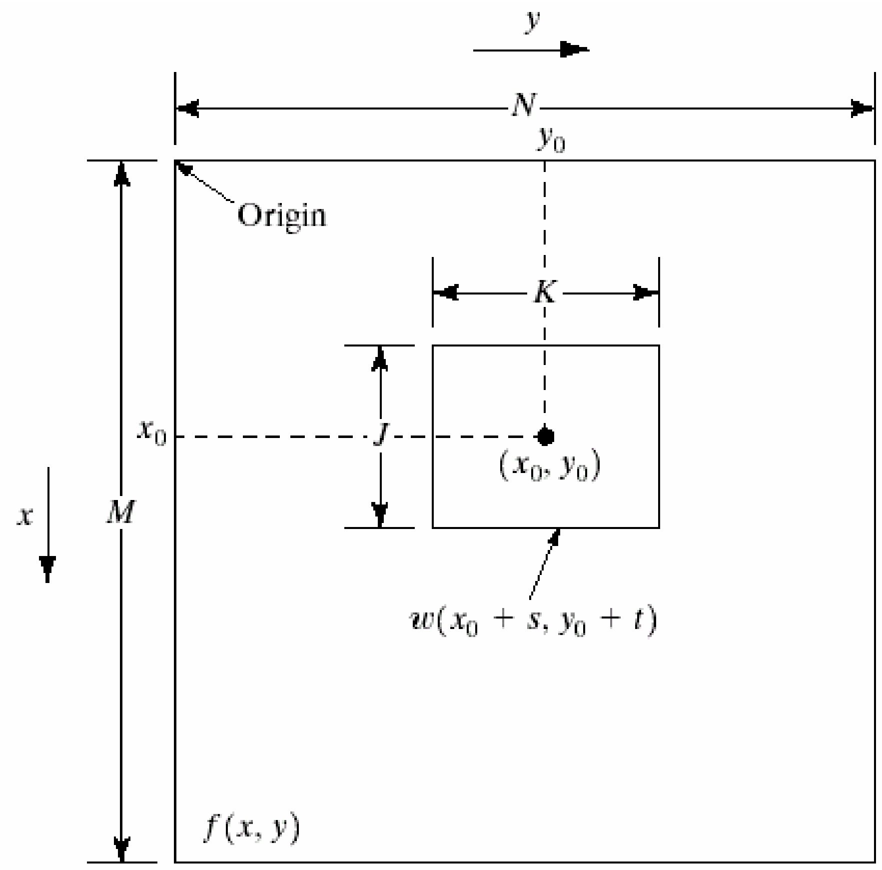
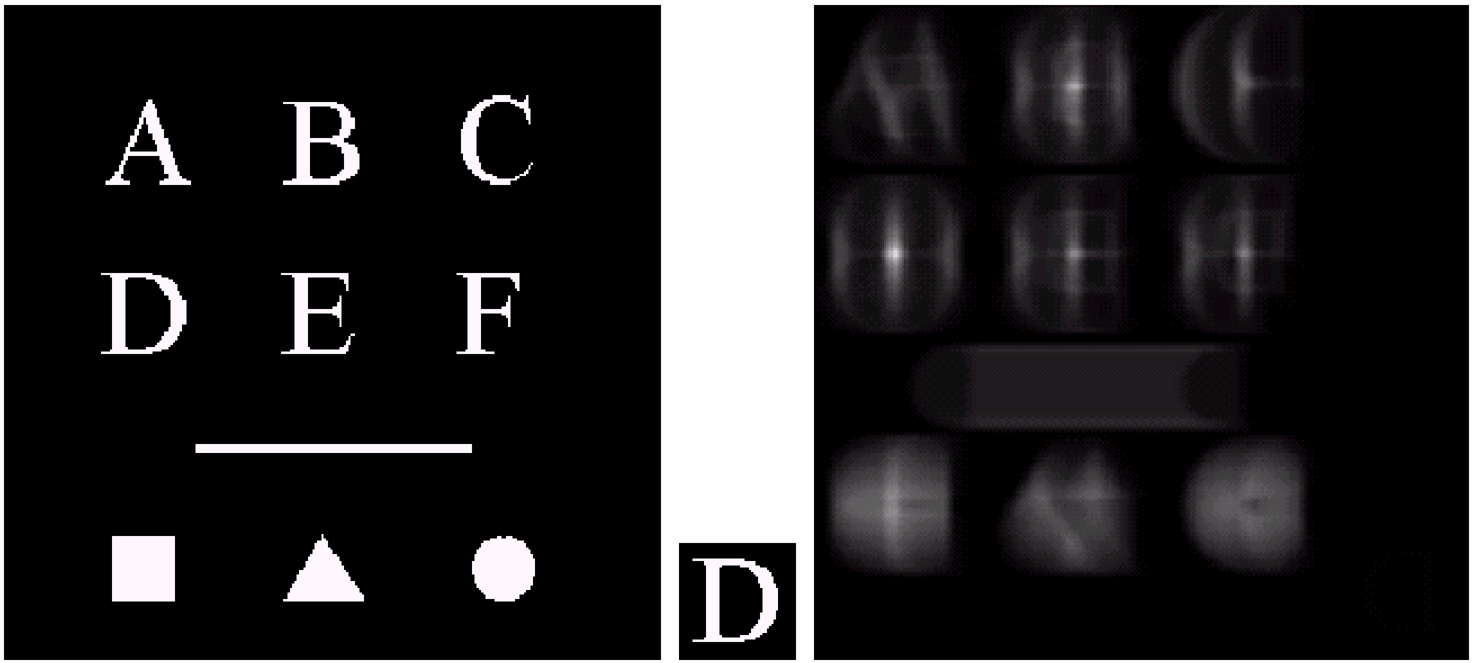


FIGURE 12.8 Arrangement for obtaining the correlation of f and w at point (x_0, y_0).

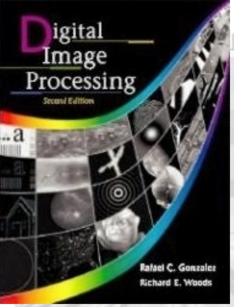
12.2.1 Matching



a b c

FIGURE 12.9 (a) Image. (b) Subimage. (c) Correlation coefficient of (a) and (b). Note that the highest (brighter) point in (c) occurs when subimage (b) is coincident with the letter "D" in (a).

$f(x, y)$ $w(x, y)$ $\gamma(x, y)$



12.2.2 Optimal statistical classifier

- Assume the probability that a particular pattern **x** comes from class ω_i is denoted as $p(\omega_i|\mathbf{x})$.
- If the pattern classifier decides that **x** came from ω_j and when it actually came from ω_i and it incur a loss, denoted as L_{ij}.
- A pattern **x** may be assigned to any class, and the average loss incurred is

$$r_j(\mathbf{x}) = \sum_{k=1}^{W} L_{kj}\, p(\omega_k|\mathbf{x})$$

12.2.2 Optimal statistical classifier

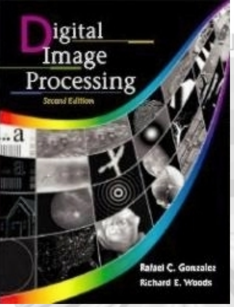
- Under the Bayesian rule: p(A|B)=p(A)p(B|A)/p(B), we have

$$r_j(\mathbf{x}) = \frac{1}{p(\mathbf{x})} \sum_{k=1}^{W} L_{kj}\, p(\mathbf{x}|\omega_k)\, p(\omega_k)$$

- Since $p(\mathbf{x})$ is common to all the $r_j(\mathbf{x})$, it can be dropped as

$$r_j(\mathbf{x}) = \sum_{k=1}^{W} L_{kj}\, p(\mathbf{x}|\omega_k)\, p(\omega_k)$$

- The classifier minimizes the total average loss is called the Bayes classifier.



12.2.2 Optimal statistical classifier

- The Bayes classifier assigns a unknown pattern **x** to class ω_i if $r_i(\mathbf{x}) < r_j(\mathbf{x})$ for $j=1, 2, \ldots W$, and $j \neq i$

$$\sum_{k=1}^{W} L_{ki}\, p(\mathbf{x}|\omega_k)\, p(\omega_k) < \sum_{q=1}^{W} L_{qj}\, p(\mathbf{x}|\omega_q)\, p(\omega_q)$$

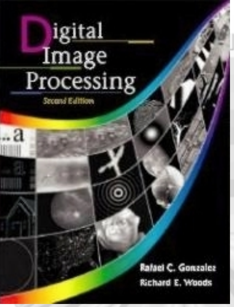
- Assume the lose function $L_{ij}=1-\delta_{ij}$ then we have

$$r_j(\mathbf{x}) = \sum_{k=1}^{W} (1-\delta_{kj})\, p(\mathbf{x}|\omega_k)\, p(\omega_k) = p(\mathbf{x}) - p(\mathbf{x}|\omega_j)\, p(\omega_j)$$

- The Bayes classifier assign a pattern to class ω_i if

$$p(\mathbf{x}) - p(\mathbf{x}|\omega_j)\, p(\omega_j) > p(\mathbf{x}) - p(\mathbf{x}|\omega_i)\, p(\omega_i)$$

- or $$d_j(\mathbf{x}) = p(\mathbf{x}|\omega_j)\, p(\omega_j) < p(\mathbf{x}|\omega_i)\, p(\omega_i) = d_i(\mathbf{x})$$



12.2.2 Optimal statistical classifier

Bayes classifier for Gaussian classes

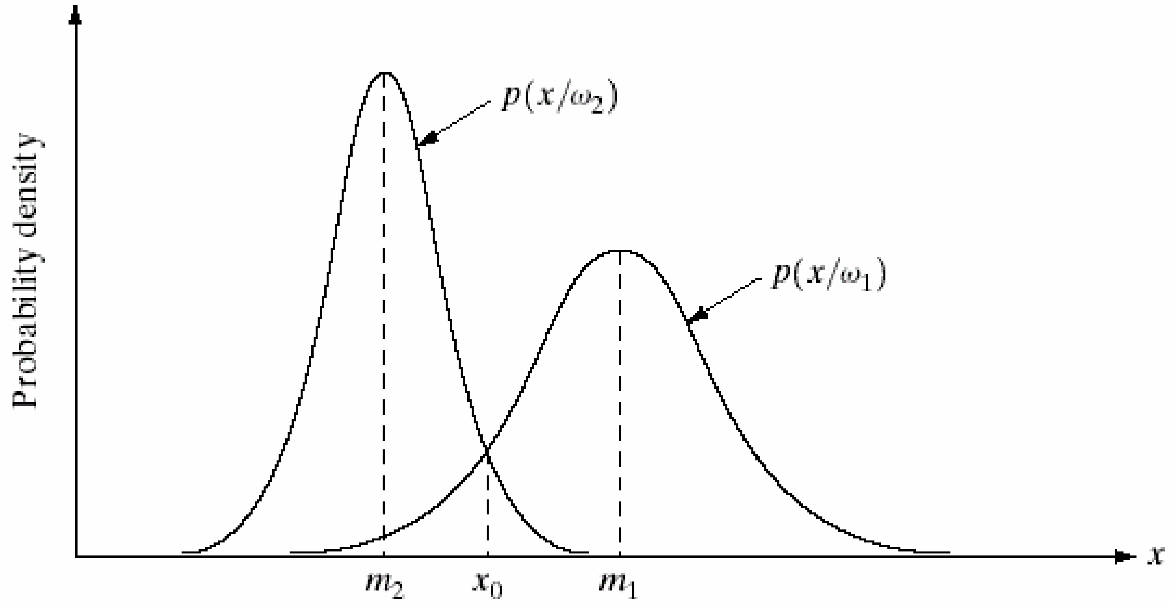
- Consider 1-D problem with two classes W=2.
- The Bayes decision function is

$$d_j(x) = p(x|\omega_j)p(\omega_j) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(x-m_j)^2}{2\sigma_j^2}} p(\omega_j)$$

- The boundary between two classes is $x=x_0$ such that $d_1(x_0)= d_2(x_0)$
- For equal-likely case $p(\omega_1)=p(\omega_2)=1/2$, then $p(x_0|\omega_1) = p(x_0|\omega_1)$, e.g., boundary (at $x= x_0$), is shown in Fig. 12.10.
- For non-equal-likely case $p(\omega_1)\neq p(\omega_2)$, if ω_2 is more likely, then x_0 move to the right, else it moves to the left

12.2.2 Optimal statistical classifier

FIGURE 12.10 Probability density functions for two 1-D pattern classes. The point x_0 shown is the decision boundary if the two classes are equally likely to occur.



12.2.2 Optimal statistical classifier

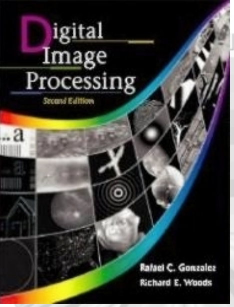
- In n-dimensional case, the Gaussian density of vector is the jth pattern class has the form as

$$p(\mathbf{x}|\omega_j) = \frac{1}{(2\pi)^{n/2}|\mathbf{C}_j|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{m}_j)^T \mathbf{C}_j^{-1}(\mathbf{x}-\mathbf{m}_j)}$$

where the mean vector is $\mathbf{m}_j = E_j\{\mathbf{x}\}$ and covariance matrix $\mathbf{C}_j = E_j\{(\mathbf{x}\text{-}\mathbf{m}^j)(\mathbf{x}\text{-}\mathbf{m}^j)^T\}$

- Approximating the mean by the averaging

$$\mathbf{m}_j = \frac{1}{N_j}\sum_{x\in\omega_j}\mathbf{x} \quad \text{and} \quad \mathbf{C}_j = \frac{1}{N_j}\sum_{x\in\omega_j}\mathbf{x}\mathbf{x}^T - \mathbf{m}_j\mathbf{m}_j^T$$



12.2.2 Optimal statistical classifier

- The decision function can also be written as

$$d_j(\mathbf{x}) = \ln[p(\mathbf{x}|\omega_j)p(\omega_j)] = \ln p(\mathbf{x}|\omega_j) + \ln p(\omega_j)$$

- For n-dimensional Gaussian density function, we have

$$d_j(\mathbf{x}) = \ln p(\omega_j) - \frac{n}{2}(2\pi)^{n/2} - \frac{1}{2}\ln|\mathbf{C}_j| - \frac{1}{2}[(\mathbf{x}-\mathbf{m}_j)^T \mathbf{C}_j^{-1}(\mathbf{x}-\mathbf{m}_j)]$$

- Simplified as $d_j(\mathbf{x}) = \ln p(\omega_j) - \frac{1}{2}\ln|\mathbf{C}_j| - \frac{1}{2}[(\mathbf{x}-\mathbf{m}_j)^T \mathbf{C}_j^{-1}(\mathbf{x}-\mathbf{m}_j)]$

- If all covariance matrix are equal $\mathbf{C}_j=\mathbf{C}$ for all j then

$$d_j(\mathbf{x}) = \ln p(\omega_j) + \mathbf{x}^T \mathbf{C}_j^{-1}\mathbf{m}_j - \frac{1}{2}(\mathbf{m}_j^T \mathbf{C}_j^{-1}\mathbf{m}_j)$$

- If $\mathbf{C}=\mathbf{I}$ then $d_j(\mathbf{x}) = \mathbf{x}^T \mathbf{C}^{-1}\mathbf{m}_j - \frac{1}{2}\mathbf{m}_j^T\mathbf{m}_j$

12.2.2 Optimal statistical classifier

- ***Example***

$$\mathbf{m}_1 = \frac{1}{4}\begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix} \qquad \mathbf{m}_2 = \frac{1}{4}\begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} \qquad \mathbf{C}_1 = \mathbf{C}_1 = \frac{1}{16}\begin{bmatrix} 3 & 1 & 1 \\ 1 & 3 & -1 \\ 1 & -1 & 3 \end{bmatrix}$$

We assume equal-likely case $p(\omega_1)=p(\omega_2)=1/2$ then

$$d_j(\mathbf{x}) = \mathbf{x}^T \mathbf{C}^{-1} \mathbf{m}_j - \frac{1}{2}\mathbf{m}_j^T \mathbf{m}_j$$

where $\mathbf{C}^{-1} = \begin{bmatrix} 8 & -4 & -4 \\ -4 & 8 & 4 \\ -4 & 4 & 8 \end{bmatrix}$

The *decision functions*: $d_1(\mathbf{x}) = 4x_1 - 15$ and $d_2(\mathbf{x}) = -4x_1 + 8x_2 + 8x_3 - 5.5$

Decision surface: $d(\mathbf{x}) = d_1(\mathbf{x}) - d_2(\mathbf{x}) = -4x_1 + 8x_2 + 8x_3 - 5.5$

12.2.2 Optimal statistical classifier

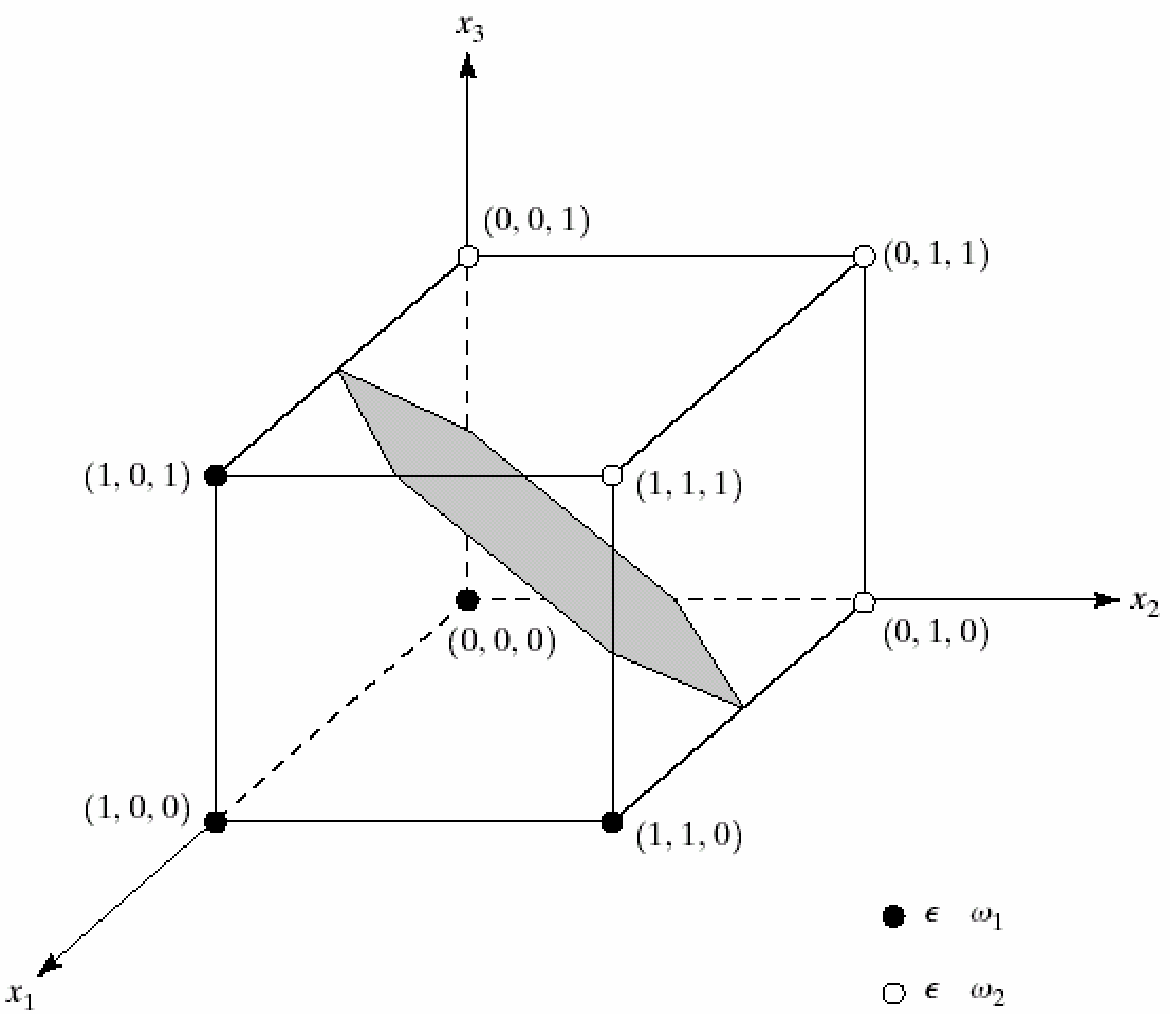
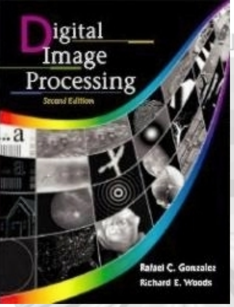


FIGURE 12.11 Two simple pattern classes and their Bayes decision boundary (shown shaded).



12.2.2 Optimal statistical classifier

- ***Example*** Multi-spectral scanner response to selected wavelength bands: 0.40~0.44 microns (violet), 0.58~0.62 microns (green), 0.66~072 microns (red), 0.80~1.00 microns (infrared). Every point in the ground is represented by 4-element pattern vector as $\mathbf{x}=[x_1, x_2, x_3, x_4]$

12.2.2 Optimal statistical classifier

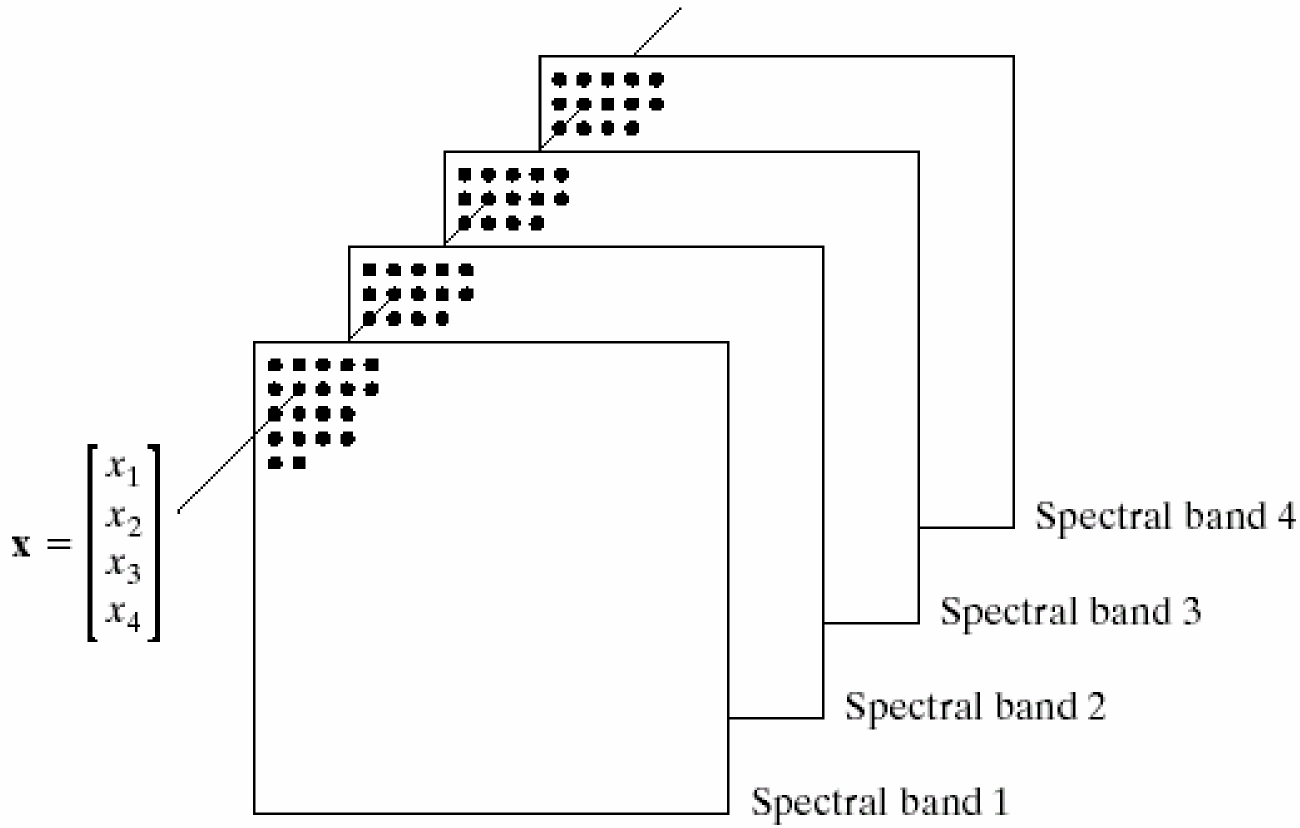
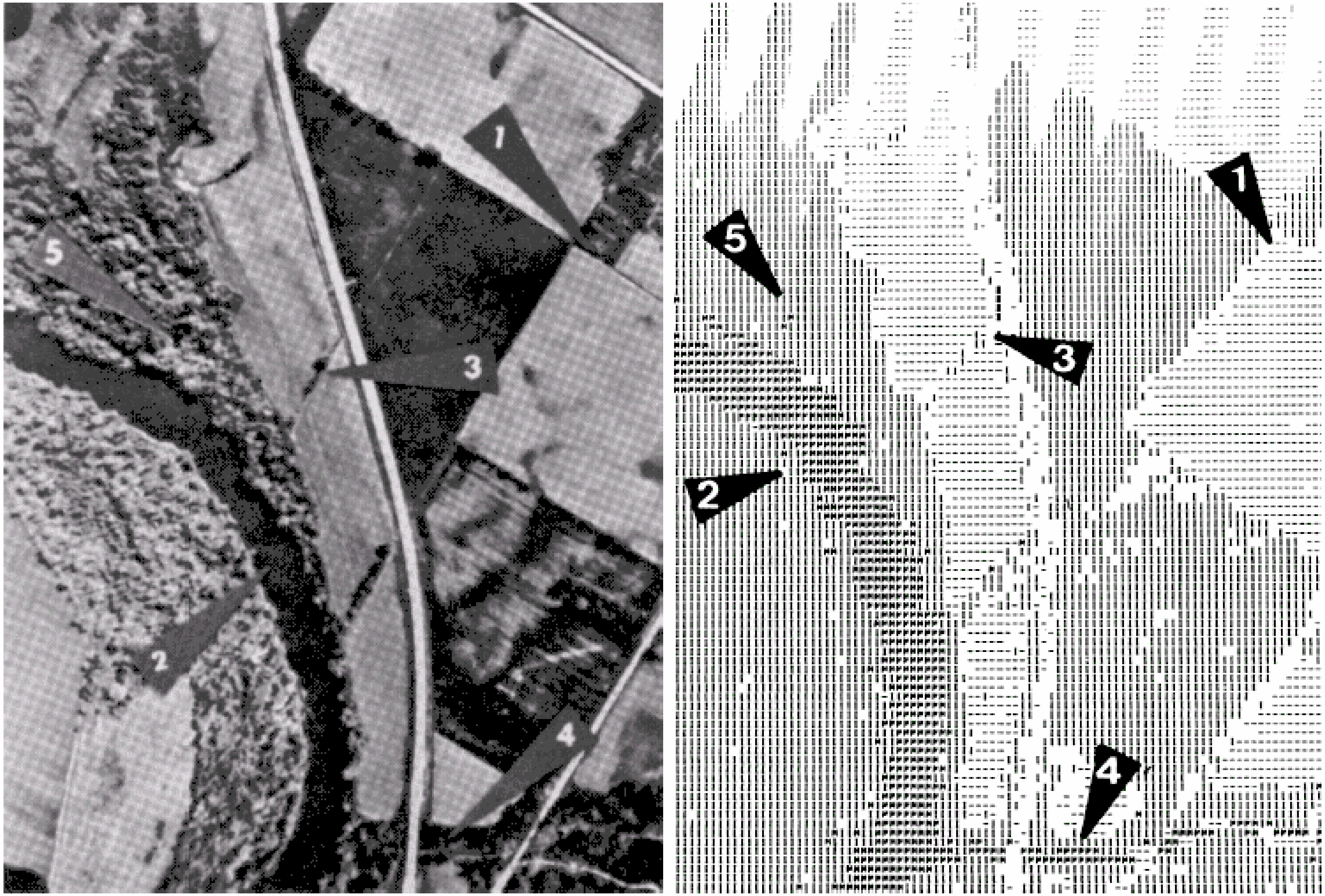


FIGURE 12.12 Formation of a pattern vector from registered pixels of four digital images generated by a multispectral scanner.

12.2.2 Optimal statistical classifier

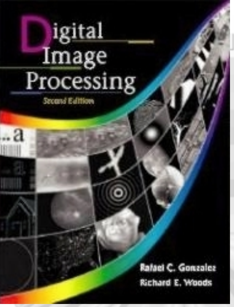


a b

FIGURE 12.13 (a) Multispectral image. (b) Printout of machine classification results using a Bayes classifier. (Courtesy of the Laboratory for Applications of Remote Sensing, Purdue University.)

12.2.3 Neural Networks

- The patterns used to estimate the parameters (mean and covariance of each class) are the training patterns, or training set
- The process by which a training set is used to obtain the decision function is called learning or training.
- The statistical properties of pattern classes in a problem often are unknown or cannot be estimated
- *Solution*: ***neural networks***



12.2.3 Neural Networks

- Non-linear computing elements organized as a network are believed to be similar to the neurons in the brain called ***neural network, neurocomputers, parallel distribution model*** (***PDP***) etc.
- Interest in neural networks dated back to 1940s
- During 1950~1960, learning machine such as ***perceptron*** is proposed by Rosenblatt.
- 1969, Minsky and Papert discouraged the perceptron-like machine.
- 1986, Rumelheart, Hinto and Willams, dealing with the developement of multi-layer perceptrons

12.2.3 Neural Networks

- Perceptron of two pattern classes (Fig. 12.14)
- The response of the basic devise is based on a weighted sum of its inputs as $d(\mathbf{x})=\Sigma_i w_i x_i + w_{n+1}$.
- It is a ***linear decision function*** with respect to a pattern vectors. The coefficient w_i $i=1,\ldots n,\ n+1$, are weights.
- The function that maps the output of summation to the output of the device is called the activation function.
- When $d(\mathbf{x})>0$ the threshold element causes the output of the perceptron to be +1, indicating $\mathbf{x}$ belonging to ω_1. When $d(\mathbf{x})<0$ indicating the other case.
- The ***decision boundary*** is $d(\mathbf{x})= \Sigma_i w_i x_i + w_{n+1} = 0$

12.2.3 Neural Networks

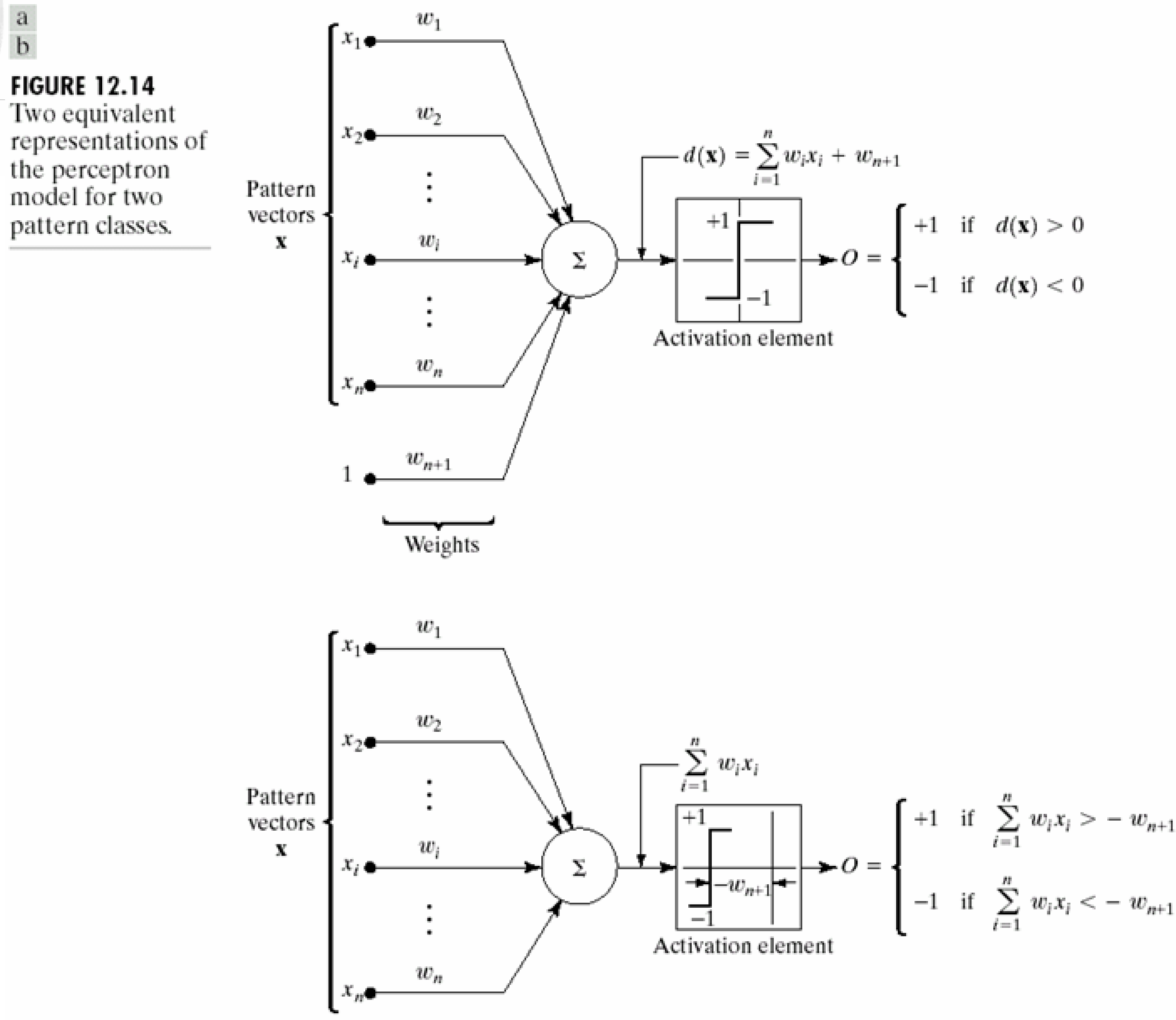
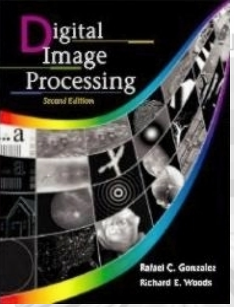


FIGURE 12.14 Two equivalent representations of the perceptron model for two pattern classes.

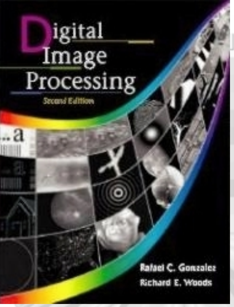


12.2.3 Neural Networks

- Another formulation is to augments the pattern vectors by appending an additional $(n+1)$st element, which is always equal to 1.
- An argument pattern vector y is created from a pattern vector x by letting $y_i = x_i$ and $y_{n+1} = 1$.
- The decision function becomes

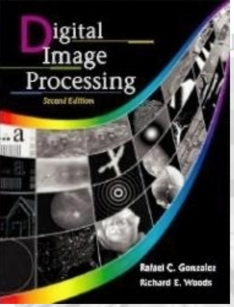
$$d(\mathbf{y}) = \Sigma_i \, w_i y_i$$

where $\mathbf{y} = (y_1, y_2 \ldots \ldots y_n, 1)^T$ is an *argument pattern vector*.



12.2.3 Neural Networks

- ***Training Algorithms* $\rightarrow$ *linearly separable case***
- For two training sets belonging to pattern classes ω_1 and ω_2 . Let $\mathbf{w}(1)$ be the initial weight vector chosen arbitrarily.
- At the *k*th iterative step, if $\mathbf{y}(k) \in \omega_1$ and $\mathbf{w}^T(k)\mathbf{y}(k) \leq 0$, replace $\mathbf{w}(k)$ by $\mathbf{w}(k+1) = \mathbf{w}(k) + c\mathbf{y}(k)$, where c is a positive correction number.
- Conversely, if $\mathbf{y}(k) \in \omega_2$ and $\mathbf{w}^T(k)\mathbf{y}(k) \geq 0$, replace $\mathbf{w}(k)$ by $\mathbf{w}(k+1) = \mathbf{w}(k) - c\mathbf{y}(k)$.
- Otherwise leave $\mathbf{w}(k)$ unchanged $\mathbf{w}(k+1) = \mathbf{w}(k)$
- The algorithm is referred as ***fixed increment correction rule***. It converges if two training pattern sets are ***linearly separable***,



12.2.3 Neural Networks

- ***Example*** - Consider two training sets (Fig. 12.15)
- $\{(0,0,1), (0,1,1)\} \in \omega_1$, $\{(1,0,1), (1,1,1)\} \in \omega_2$
- Let c=1 and **w**(1)=**0**. representing the patterns in the order of sequence as

$$\mathbf{w}^T(1)\mathbf{y}(1) = \begin{bmatrix}0 & 0 & 0\end{bmatrix} \cdot \begin{bmatrix}0\\0\\1\end{bmatrix} = 0$$

$$\mathbf{w}(2) = \mathbf{w}(1) + \mathbf{y}(1) = \begin{bmatrix}0\\0\\1\end{bmatrix}$$

$$\mathbf{w}^T(2)\mathbf{y}(2) = \begin{bmatrix}0 & 0 & 1\end{bmatrix} \cdot \begin{bmatrix}0\\1\\1\end{bmatrix} = 1$$

w(3)=**w**(2)

$$\mathbf{w}^T(3)\mathbf{y}(3) = \begin{bmatrix}0 & 0 & 1\end{bmatrix} \cdot \begin{bmatrix}1\\0\\1\end{bmatrix} = 1$$

$$\mathbf{w}(4) = \mathbf{w}(3) - \mathbf{y}(3) = \begin{bmatrix}-1\\0\\0\end{bmatrix}$$

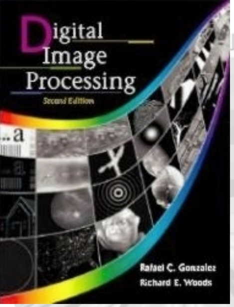
$$\mathbf{w}^T(4)\mathbf{y}(4) = \begin{bmatrix}-1 & 0 & 0\end{bmatrix} \cdot \begin{bmatrix}1\\1\\1\end{bmatrix} = -1$$

w(5)=**w**(4)

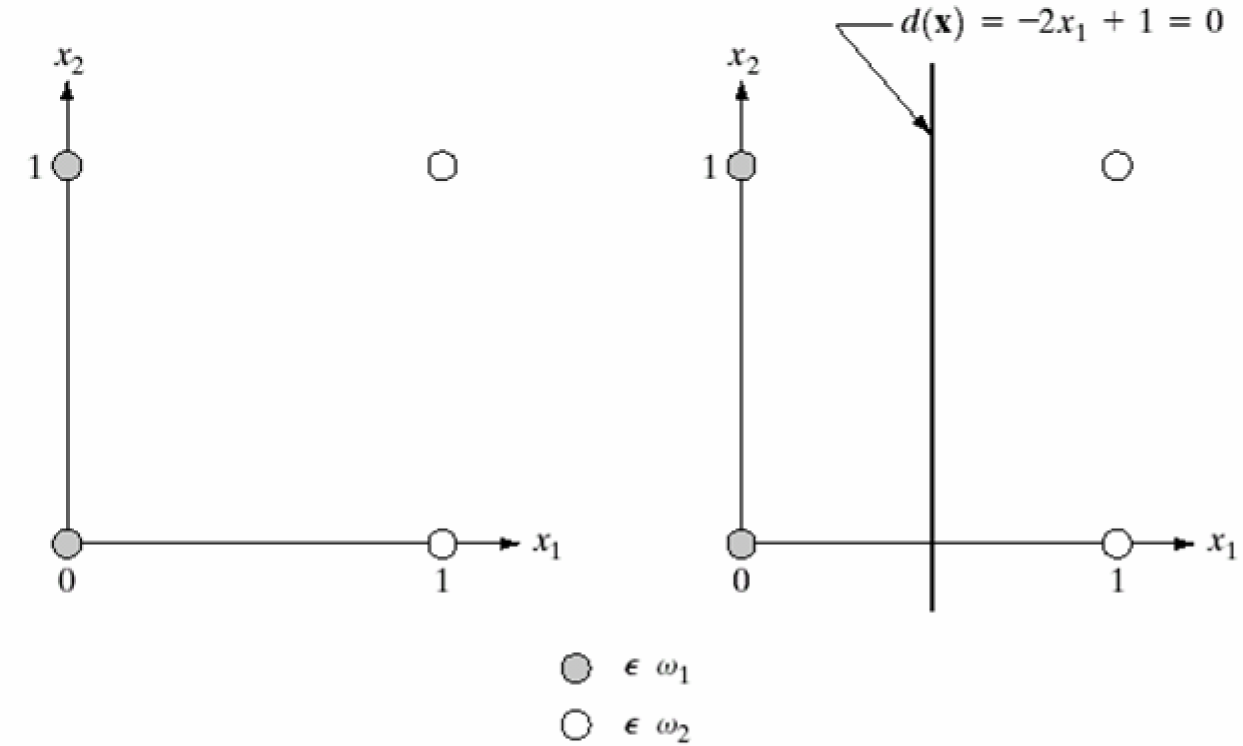
12.2.3 Neural Networks

- ***Example (continued)***
- A solution is obtained only when the algorithm yields a complete error-free iteration through all training patterns
- Convergence is achieved at k=14, yield the solution weight vector $\mathbf{w}(14)=(-2, 0, 1)^T$.
- The corresponding decision function is

$$d(\mathbf{y})=-2y_1+1 \text{ and } d(\mathbf{x})=-2x_1+1.$$



12.2.3 Neural Networks



a b

FIGURE 12.15
(a) Patterns belonging to two classes.
(b) Decision boundary determined by training.

12.2.3 Neural Networks

- Non-separable classes (non-linear case) $\rightarrow$ Least-mean square (***LMS) data rule***
- Consider *iteration function* $J(\mathbf{w})=\frac{1}{2}(r-\mathbf{w}^T\mathbf{y})^2$

  where r is the desired response (*i.e.*, r=+1, $\mathbf{y}\in\omega_2$, and r=-1 $\mathbf{y}\in\omega_1$)
- The task of LMS data rule is to adjust w incrementally in the direction of negative gradient of $J(\mathbf{w})$ in order to minimize $J(\mathbf{w})$ which occurs when $r=\mathbf{w}^T\mathbf{y}$
- After the k iteration step, the $\mathbf{w}(k)$ is updated as

$$\mathbf{w}(k+1)=\mathbf{w}(k)-\alpha\left[\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}\right]_{\mathbf{w}=\mathbf{w}(k)}$$

12.2.3 Neural Networks

- From $J(\mathbf{w})=½(\mathrm{r}–\mathbf{w}^{\mathrm{T}}\mathbf{y})^2$ we have

$$\mathbf{w}(k+1)=\mathbf{w}(k)-\alpha\left[r(k)-\mathbf{w}^{T}(k)\mathbf{y}(k)\right]\mathbf{y}(k)$$

or $\mathbf{w}(k+1)= \mathbf{w}(k)+\alpha\, e(k)\mathbf{y}(k)$, and $e(k)=\mathrm{r}(k)-\mathbf{w}^{\mathrm{T}}(k)\mathbf{y}(k)$

- If we change $\mathbf{w}(k)$ to $\mathbf{w}(k+1)$ but leave the pattern the same, the error becomes $\mathrm{e}(k) = \mathrm{r}(k)-\mathbf{w}^{\mathrm{T}}(k+1)\mathbf{y}(k)$.
- So $\Delta e(k)= [\mathbf{w}^{\mathrm{T}}(k+1) - \mathbf{w}^{\mathrm{T}}(k)]\mathbf{y}(k)$

$$= -\alpha\, e(k)\mathbf{y}^{\mathrm{T}}(k)\mathbf{y}(\mathrm{k})= -\alpha\, e(k)\,\|\,\mathbf{y}(\mathrm{k})\,\|^{2}$$

12.2.3 Neural Networks

- ***Multilayer feedforward neural networks (fig. 12.14)***
- Each neuron has the same form as the ***perceptron*** model except that the hard-limiting activation function has been replaced by a soft-limiting "***sigmoid***" function which has the necessary differentiability as

$$h_j(I_j) = \frac{1}{1 + e^{-(I_j + \theta_j)/\theta_0}}$$

where I_j, $j=1,2,\ldots N_J$, is the input to the ***activation element*** of each node in layer J, θ_{j} is an offset, and θ_{j} control the shape of the sigmoid function.

The sigmoid function is plotted in fig. 12.17.

12.2.3 Neural Networks

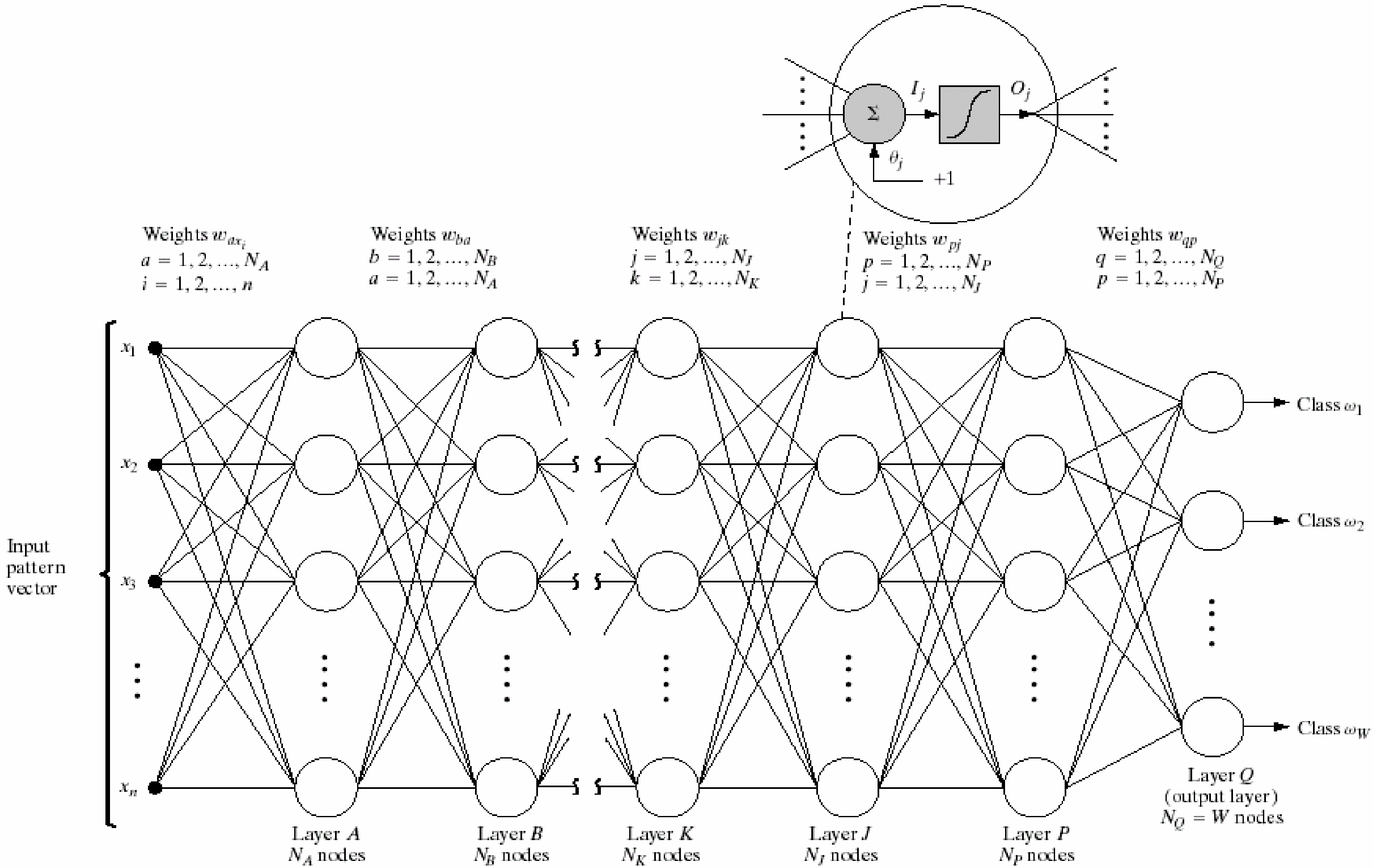


FIGURE 12.16 Multilayer feedforward neural network model. The blowup shows the basic structure of each neuron element throughout the network. The offset, θ_j, is treated as just another weight.

12.2.3 Neural Networks

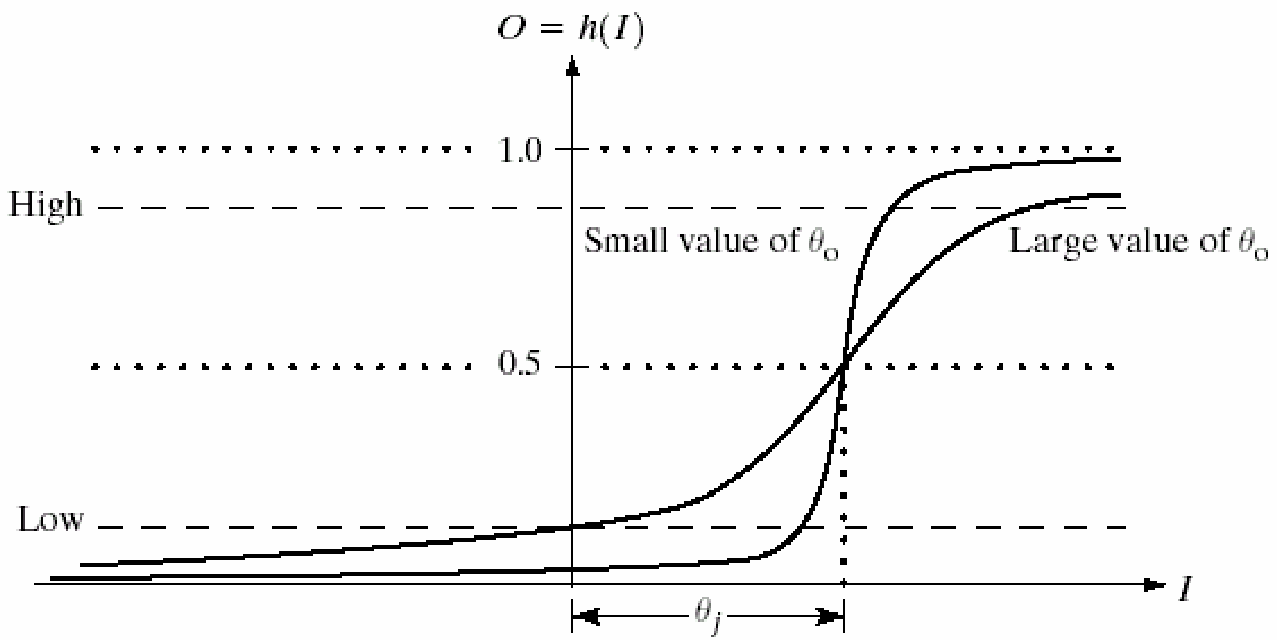


FIGURE 12.17 The sigmoidal activation function of Eq. (12.2-47).

12.2.3 Neural Networks

- From fig. 12,17, the offset θ_j is analog to the weight w_{i+1} in ***perceptron.***
- In fig. 12.16, the input to node in any layer is the weighted sum of the output from previous layer.
- Let layer *K* preceding layer *J* gives the input to the activation element of each node in layer *J*, denoted as I_j as

$$I_j = \sum_{k=1}^{N_k} w_{jk} O_k$$

where N_j or N_k is the number of nodes in layer *J* or *K*

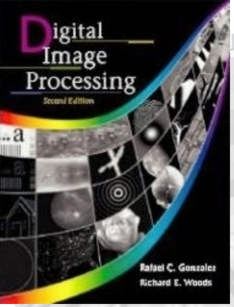
Then output of layer *K* are $O_k = h_k(I_k)$, $k=1,\ldots N_k$.

12.2.3 Neural Networks

- Every node in layer J, but each individual input can be weighted differently, as w_{ik} and w_{2k} for $k=1,2,\ldots N_k$ are the weights on the inputs to the 1st and 2nd nodes

$$h_j(I_j)=\frac{1}{1+e^{-(\sum_{k=1}^{N_k} w_{jk}O_k+\theta_j)/\theta_0}}$$

- The main problem in training a multilayer network lies in adjusting the weights in the so-called hidden layers.



12.2.3 Neural Networks

- ***Training by back propagation***
- Begin from the ***output layer***, the total square error between the desired output r_q and actual output O_q of nodes in layer Q is

$$E_q = \frac{1}{2}\sum_{q=1}^{N_Q}(r_q - O_q)^2$$

- Similar to delta rule, the training rule adjust the weights in each layer in a way that seeks a minimum of an error function, i.e.,

$$\Delta w_{qp} = -\alpha \frac{\partial E_Q}{\partial w_{qp}}$$

- Using the chain rule, we have

$$\frac{\partial E_Q}{\partial w_{qp}} = \frac{\partial E_Q}{\partial I_q}\frac{\partial I_q}{\partial w_{qp}} = \frac{\partial E_Q}{\partial I_q} O_p$$

12.2.3 Neural Networks

- Therefore, $\Delta w_{qp} = -\alpha(\partial E_Q/\partial I_q)O_p = \alpha\, \delta_q\, O_p$

  where $\delta_q = -(\partial E_Q/\partial I_q)$
- From chain rule, we have $\delta_q = -(\partial E_Q/\partial O_q)(\partial O_Q/\partial I_q)$

  where $\partial E_Q/\partial O_q = -(r_q - O_q)$

  and $\partial O_Q/\partial I_q = \partial[h_q(I_q)]/\partial I_q = h'_q(I_q)$.
- Finally, we have $\delta_q = (\mathrm{r_q} - \mathrm{O_q})\, h'_q(I_q)$
- So, $\Delta w_{qp} = \alpha\delta_q O_p = \alpha\, (\mathrm{r_q} - \mathrm{O_q})\, h'_q(I_q)\, O_p$

12.2.3 Neural Networks

- Now considering layer P, preceeding in the same manner as above as $\Delta w_{pj} = \alpha\, \delta_q O_j = \alpha\, (\mathrm{r_p} - \mathrm{O_p})\, h'_q(I_q)\, O_j$
- We have similar error terms, *i.e.*,

$$\delta_p = -(\partial E_p/\partial I_p) = -(\partial E_p/\partial O_p)(\partial O_p/\partial I_p)$$

$$= -(\partial E_p/\partial O_p) h'_{\mathrm{p}}(I_p)$$

- The term $(\partial E_p/\partial O_p)$ does not produce r_p, but is expressed as

$$-\frac{\partial E_p}{\partial O_p} = -\sum_{q=1}^{N_q} \frac{\partial E_p}{\partial I_q}\frac{\partial I_q}{\partial O_p} = \sum_{q=1}^{N_q}\left(-\frac{\partial E_p}{\partial I_q}\right) w_{pq} = \sum_{q=1}^{N_q} \delta_q w_{pq}$$

$$\longrightarrow \boxed{\text{Layer J}} \longrightarrow \boxed{\text{Layer P}} \longrightarrow \boxed{\text{Layer Q}}$$

12.2.3 Neural Networks

- The parameter δ_p *is* $\delta_p = h'_p(I_p)\, \Sigma_q\, \delta_p\, w_{qp}$
- After the error term and weights have been computed for layer *P*, these quantities can be used to compute the error and weights for the layer preceding layer *P*.

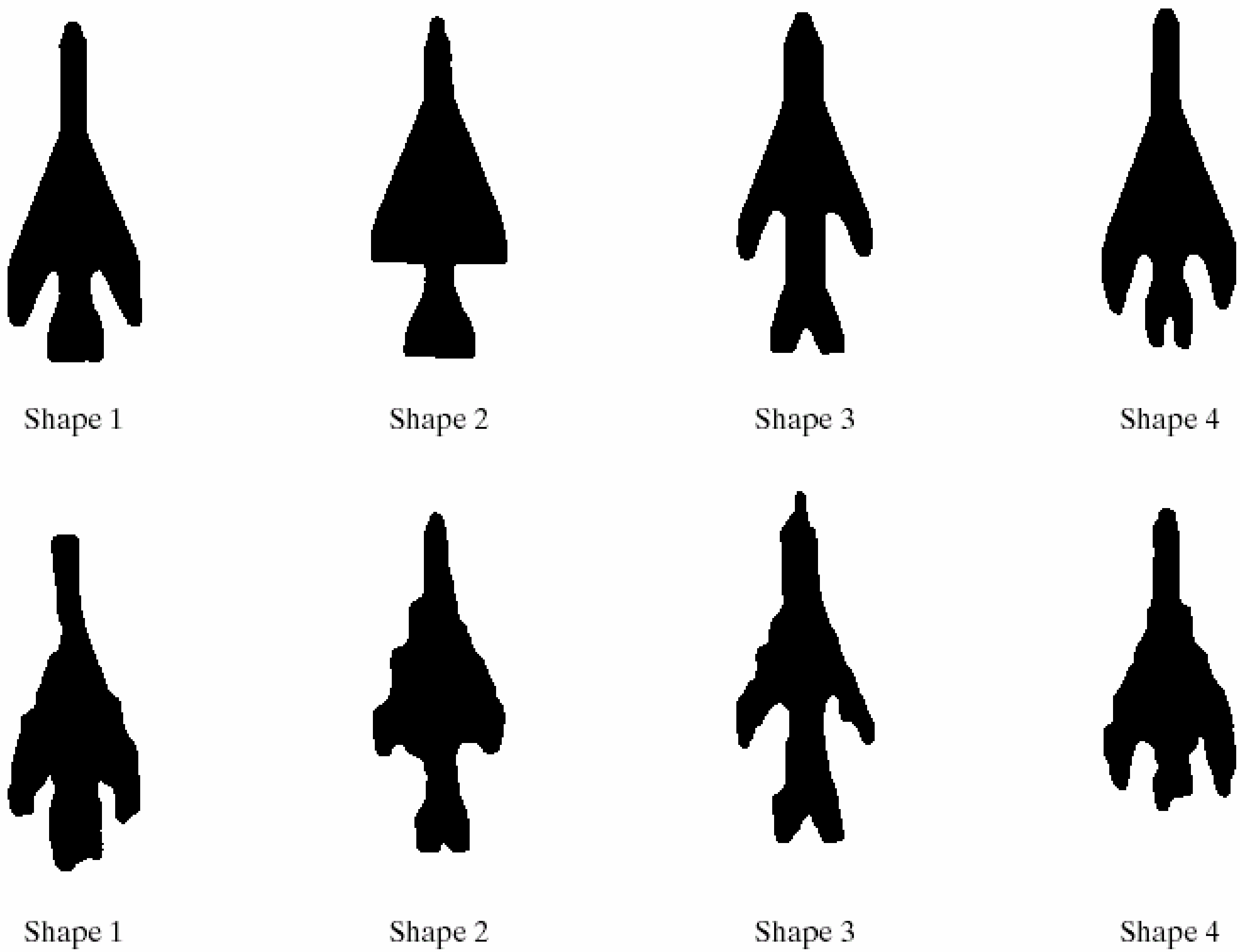
Summary: for any layers *K* and *J*, where *K* precedes *J*.

1. Computer the weights w_{jk}, which modify the connections between these two layers, by $\Delta w_{jk} = \alpha \delta_j O_k$.
2. If *J* is the output layer, $\delta_j = (r_j - O_j)\, h'_j(I_j)$
3. If *J* is the internal layer $\delta_j = h'_j(I_j)\, \Sigma_p\, \delta_p\, w_{jp}$ for $j=1,2,\ldots N_j$.

12.2.3 Neural Networks

a
b

FIGURE 12.18 (a) Reference shapes and (b) typical noisy shapes used in training the neural network of Fig. 12.19. (Courtesy of Dr. Lalit Gupta, ECE Department, Southern Illinois University.)



12.2.3 Neural Networks

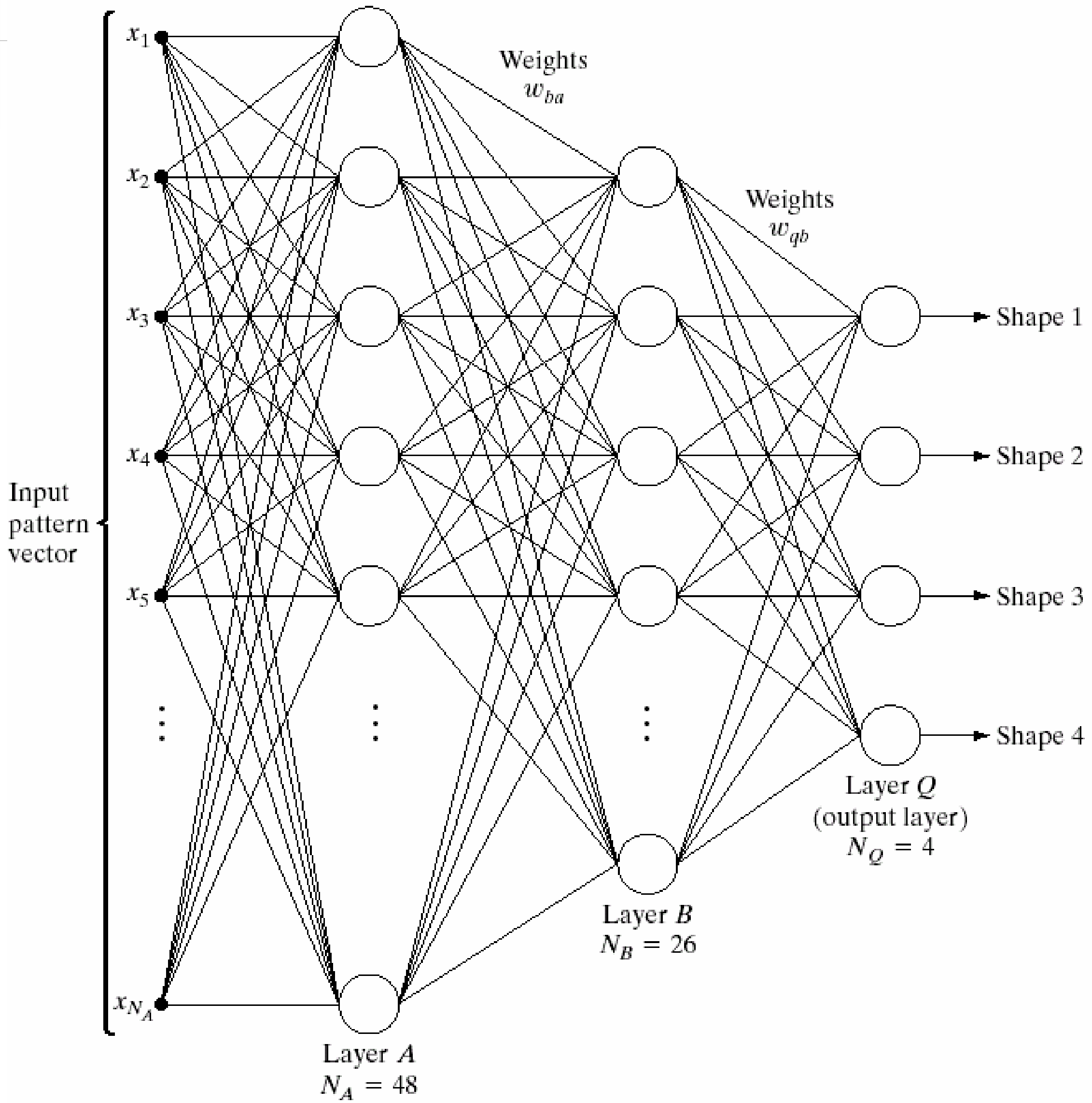


FIGURE 12.19 Three-layer neural network used to recognize the shapes in Fig. 12.18. (Courtesy of Dr. Lalit Gupta, ECE Department, Southern Illinois University.)

12.2.3 Neural Networks

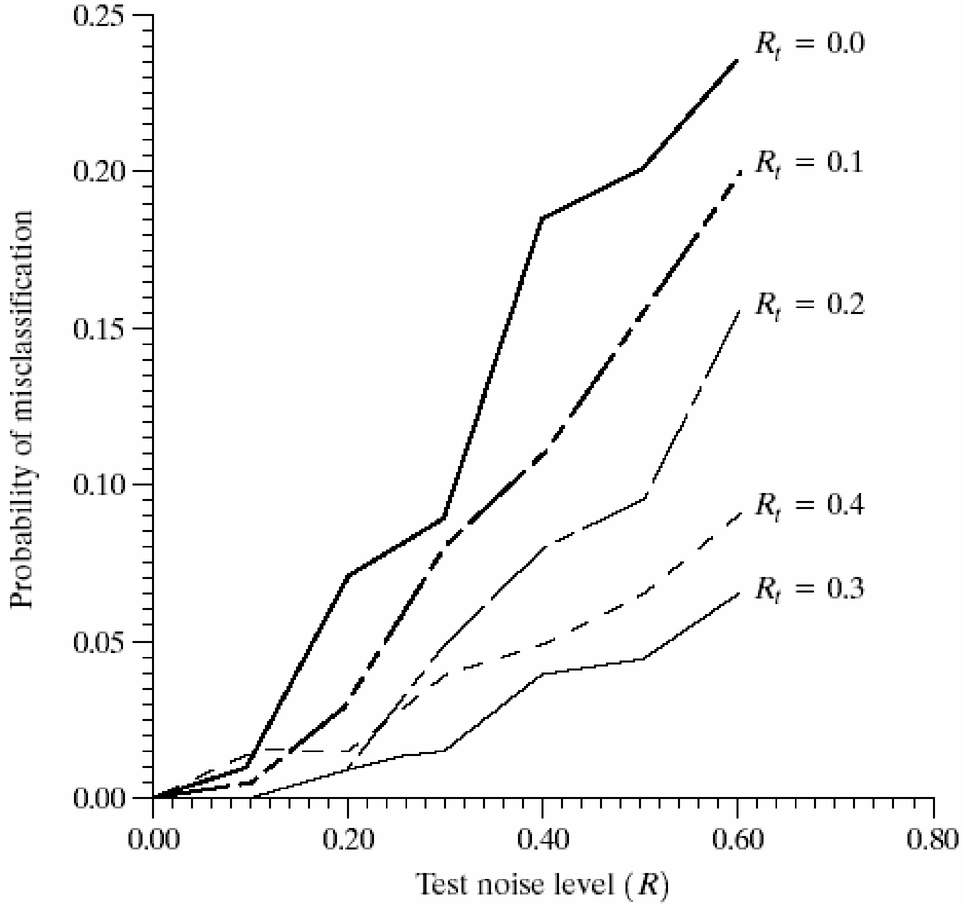


FIGURE 12.20 Performance of the neural network as a function of noise level. (Courtesy of Dr. Lalit Gupta, ECE Department, Southern Illinois University.)

12.2.3 Neural Networks

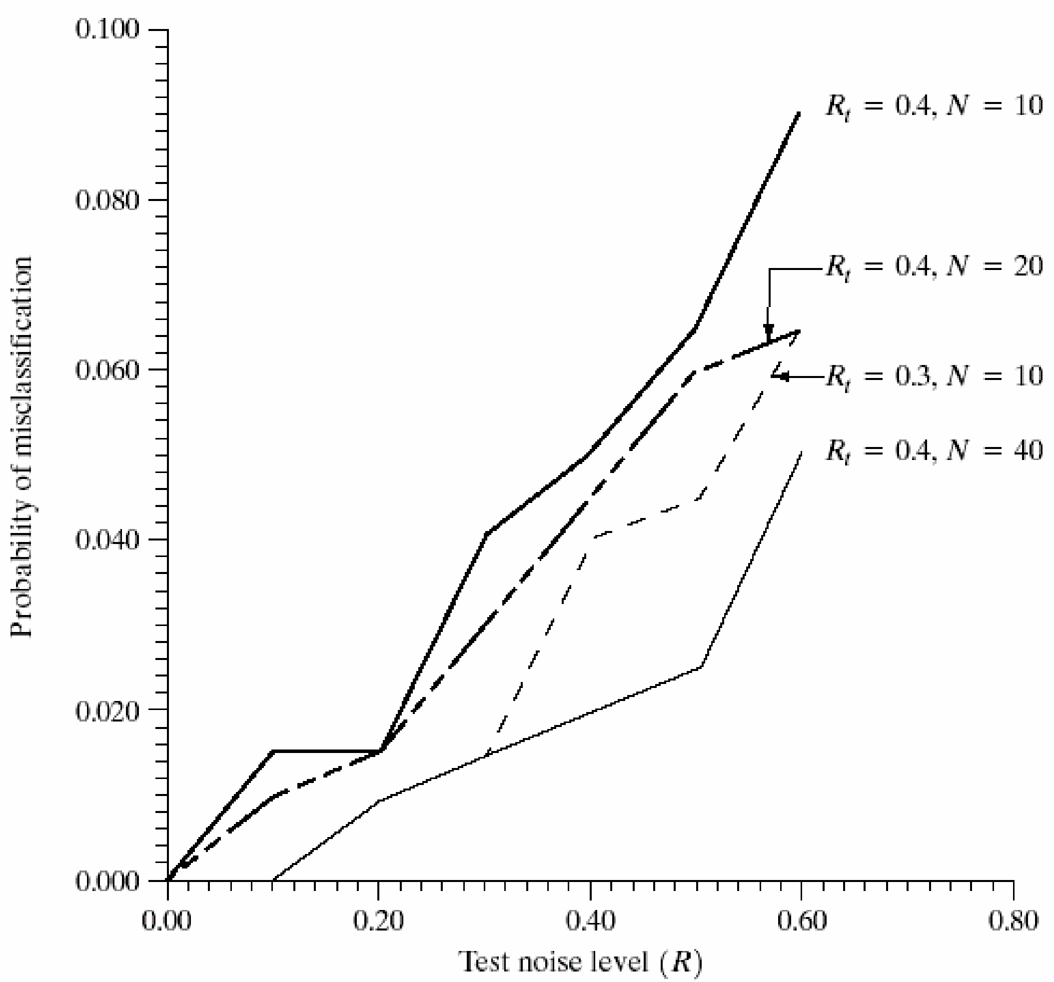


FIGURE 12.21 Improvement in performance for $R_t = 0.4$ by increasing the number of training patterns (the curve for $R_t = 0.3$ is shown for reference). (Courtesy of Dr. Lalit Gupta, ECE Department, Southern Illinois University.)

12.2.3 Neural Networks

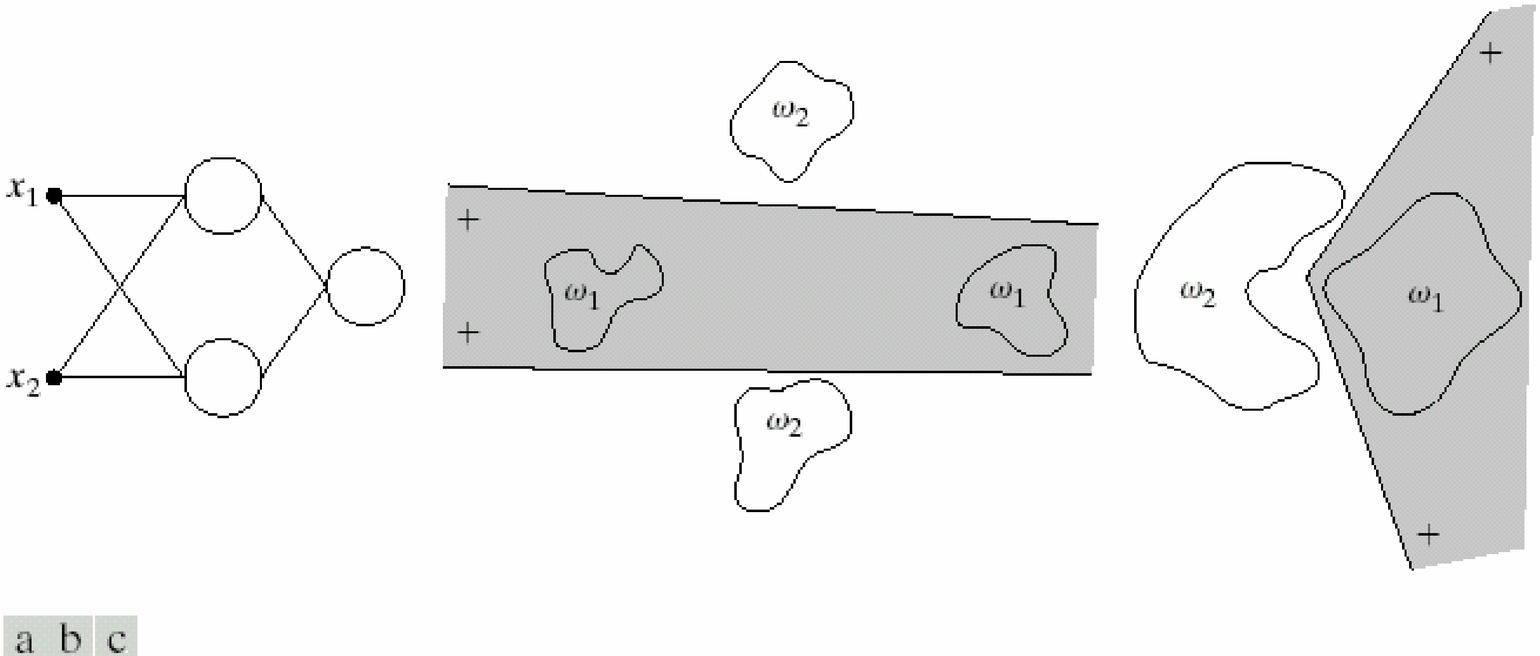


FIGURE 12.22 (a) A two-input, two-layer, feedforward neural network. (b) and (c) Examples of decision boundaries that can be implemented with this network.

12.2.3 Neural Networks

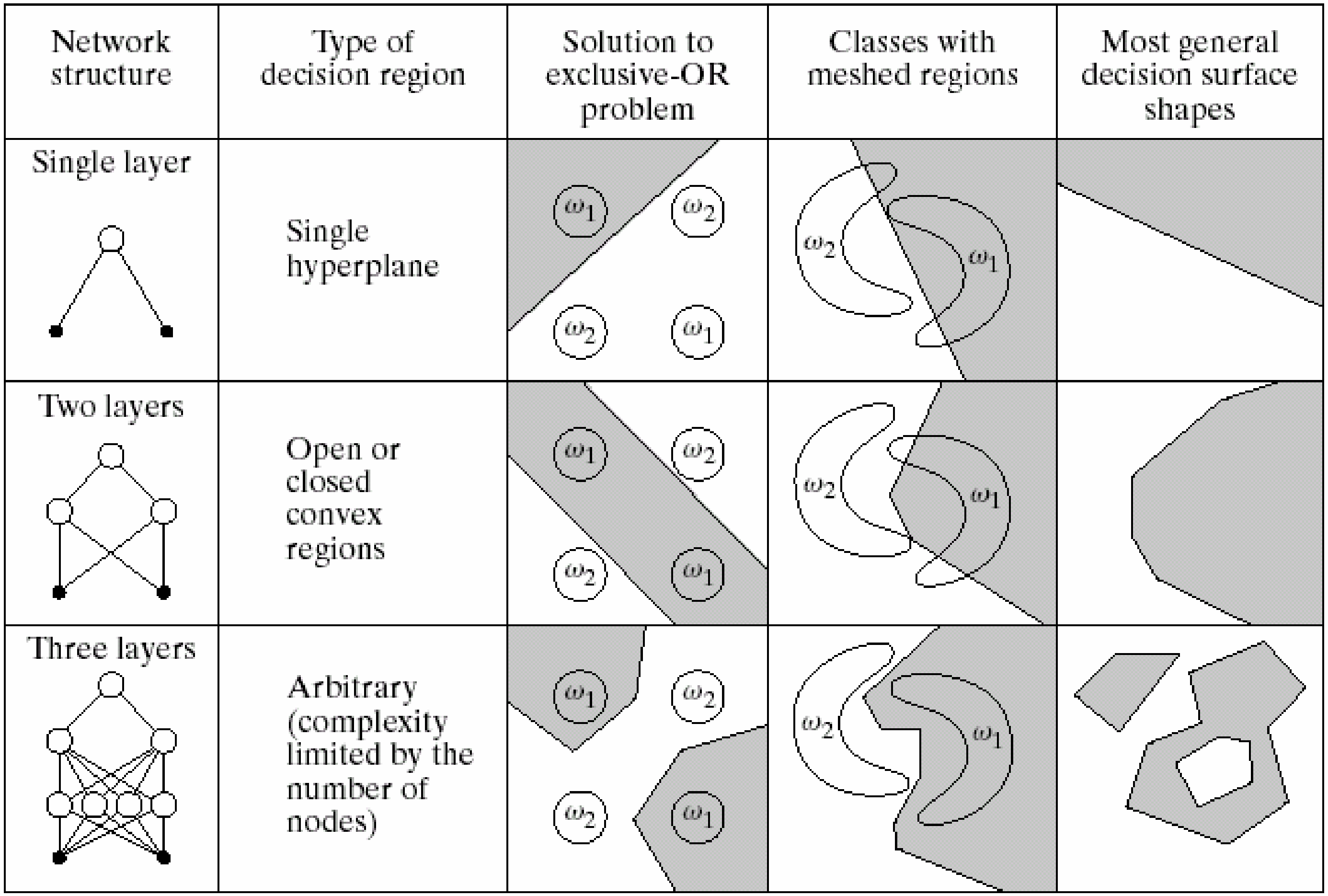
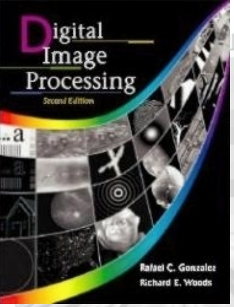


FIGURE 12.23 Types of decision regions that can be formed by single- and multilayer feed-forward networks with one and two layers of hidden units and two inputs. (Lippman)



12.3 Structural Methods

- Structural relationships inherent in a pattern's shape.
- 12.3.1 Matching Shape Numbers
- 12.3.2 String Matching
- 12.3.3 Syntactic Recognition of Strings

12.3.1 Matching Shape numbers

- Refer to 11.2.2, the degree of similarity, *k*, between two region boundaries (shapes) is defined as the largest order for which their shape numbers still coincide.
- For example, let *a* and *b* denote shape numbers of closed boundaries represented by 4-directional chain codes. These two shapes have a degree of similarity *k* if

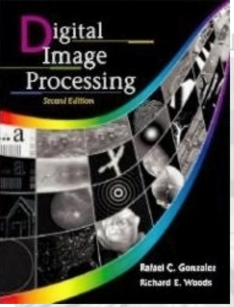
  $$s_j(a)=s_j(b) \text{ for } j=4, 6, 8, \ldots k.$$

  $$s_j(a)\neq s_j(b) \text{ for } j=k+2, k+4, \ldots.$$

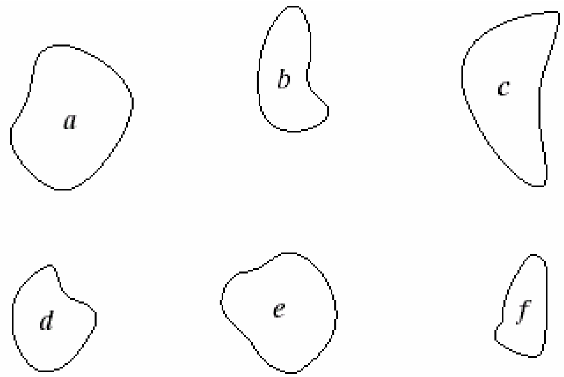
  where *s* indicates shape number and the subscript indicates order.
- The distance between two shapes *a* and *b* is defined as the inverse of their degree of similarity, *i.e.*, $D(a, b)=1/k$

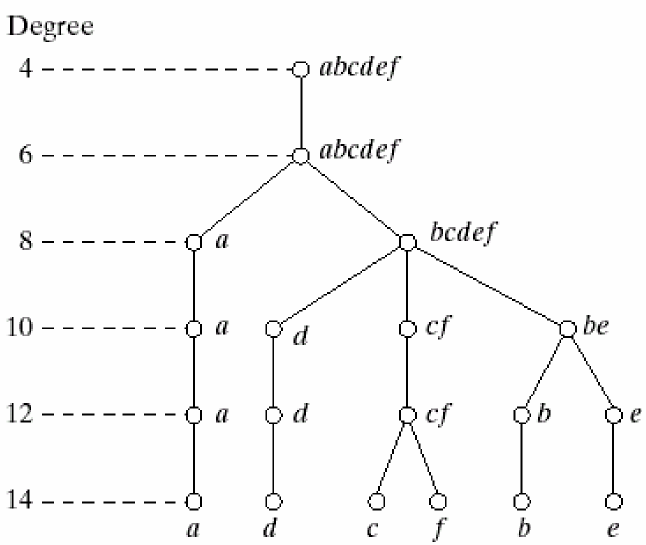
12.3.1 Matching Shape numbers

- The distance satisfy the following properties:
  - $D(a, b) \geq 0$
  - $D(a, b) = 0$ if $a = b$
  - $D(a, b) \leq max[D(a, b), D(b, c)]$
- ***Example***. Find the closest match between the give shape *f* and the other five shapes (*a*~*e*) as shown in Fig. 12.24



12.3.1 Matching Shape numbers





	a	b	c	d	e	f
a	∞	6	6	6	6	6
b		∞	8	8	10	8
c			∞	8	8	12
d				∞	8	8
e					∞	8
f						∞

a
b c

FIGURE 12.24
(a) Shapes.
(b) Hypothetical similarity tree.
(c) Similarity matrix. (Bribiesca and Guzman.)

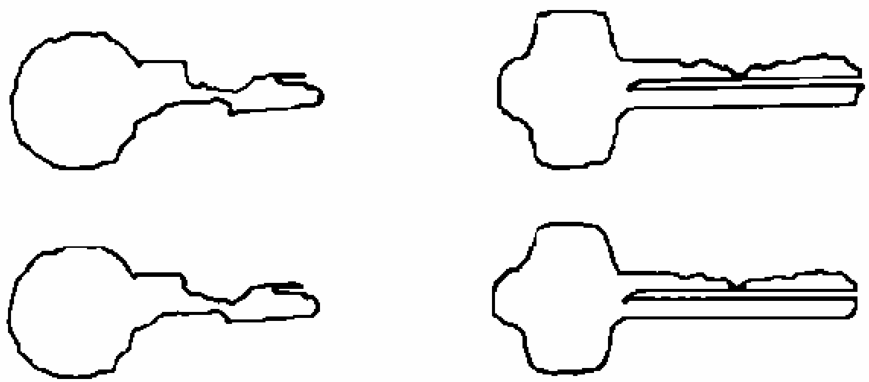
12.3.2 String Matching

- Suppose two regions *a* and *b* are coded into two strings as $a_1 a_2, \ldots a_n$, and $b_1 b_2, \ldots b_m$, respectively.
- Let α represent the number of matches between the two strings, the number of symbols that do not match is

$$\beta = \max(|a|, |b|) - \alpha$$

- where $|a|$, is the length of symbol *a*, $\beta = 0$ if *a* and *b* are identical.
- The measurement of similarity between *a* and *b* is the ratio $R = \alpha / \beta$
- Because matching is done symbol by symbol, the starting point on each boundary is important.
- Example 12.25(a) and (b) show sample boundaries of two objects, 12.25(c) and (d) show the polygonal approximations. Strings are formed from the polygon by computing the interior angle θ between segments as each polygon was traversed clockwise.

12.3.2 String Matching



a b
c d
e f
g

FIGURE 12.25 (a) and (b) Sample boundaries of two different object classes; (c) and (d) their corresponding polygonal approximations; (e)–(g) tabulations of *R*. (Sze and Yang.)

R	1.a	1.b	1.c	1.d	1.e	1.f
1.a	∞					
1.b	16.0	∞				
1.c	9.6	26.3	∞			
1.d	5.1	8.1	10.3	∞		
1.e	4.7	7.2	10.3	14.2	∞	
1.f	4.7	7.2	10.3	8.4	23.7	∞

R	2.a	2.b	2.c	2.d	2.e	2.f
2.a	∞					
2.b	33.5	∞				
2.c	4.8	5.8	∞			
2.d	3.6	4.2	19.3	∞		
2.e	2.8	3.3	9.2	18.3	∞	
2.f	2.6	3.0	7.7	13.5	27.0	∞

R	1.a	1.b	1.c	1.d	1.e	1.f
2.a	1.24	1.50	1.32	1.47	1.55	1.48
2.b	1.18	1.43	1.32	1.47	1.55	1.48
2.c	1.02	1.18	1.19	1.32	1.39	1.48
2.d	1.02	1.18	1.19	1.32	1.29	1.40
2.e	0.93	1.07	1.08	1.19	1.24	1.25
2.f	0.89	1.02	1.02	1.24	1.22	1.18

Angels are coded into one of eight possible symbols corresponding to 45^{o} increments.

12.3.3 Syntactic Recognition of Strings

- Syntactic pattern recognition: (1) a set of pattern primitives; (2) a set of rules (grammar) that governs their interconnection; (3) recognizer (automaton) whose structure is determined by the set of rules in the grammar.
- String Grammar is defined as 4-tuple:

  G=(N, Σ, P, S)

  where

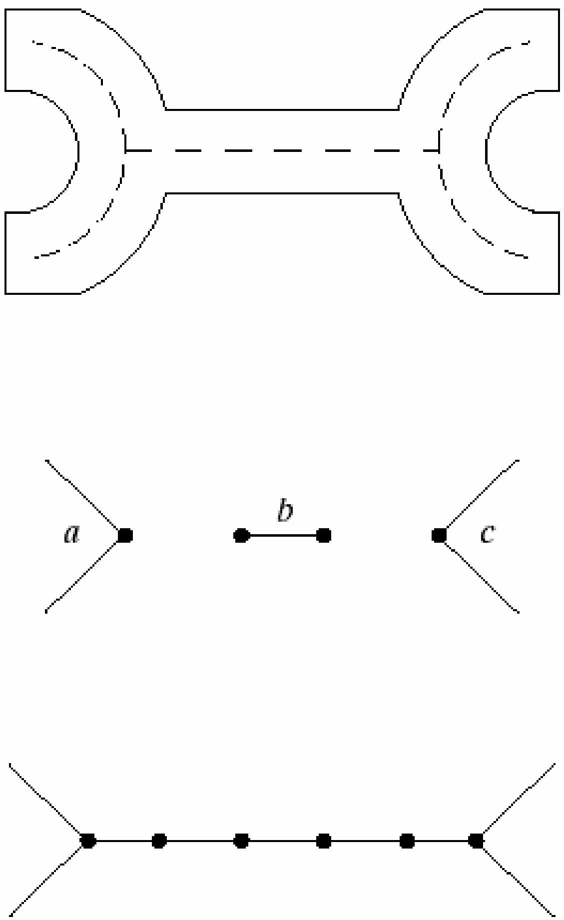
  N is a finite set of variable called non-terminals.

  Σ is a finite set of constants called terminals

  P is a set of rewritting rules called productions

  S in N is calllled starting symbol

12.3.3 Syntactic Recognition of strings



a
b
c

FIGURE 12.26
(a) Object represented by its (pruned) skeleton.
(b) Primitives.
(c) Structure generated by using a regular string grammar.

12.3.3 Syntactic Recognition of strings

TABLE 12.1 Example of semantic information attached to production rules.

Production	Semantic Information
$S \rightarrow aA$	Connections to a are made only at the dot. The direction of a, denoted θ, is given by the direction of the perpendicular bisector of the line joining the end points of the two undotted segments. The line segments are 3 cm each.
$A \rightarrow bA$	Connections to b are made only at the dots. No multiple connections are allowed. The direction of b must be the same as the direction of a. The length of b is 0.25 cm. This production cannot be applied more than 10 times.
$A \rightarrow bB$	The direction of a and b must be the same. Connections must be simple and made only at the dots.
$B \rightarrow c$	The direction of c and a must be the same. Connections must be simple and made only at the dots.

12.3.3 Syntactic Recognition of strings

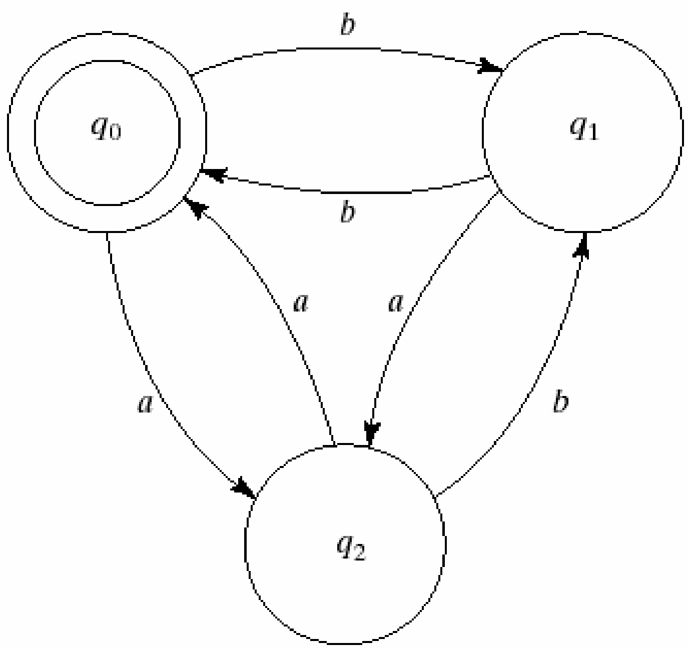
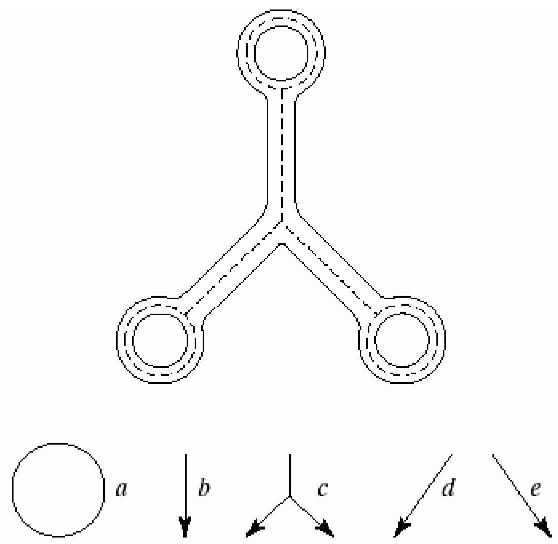


FIGURE 12.27 A finite automaton.

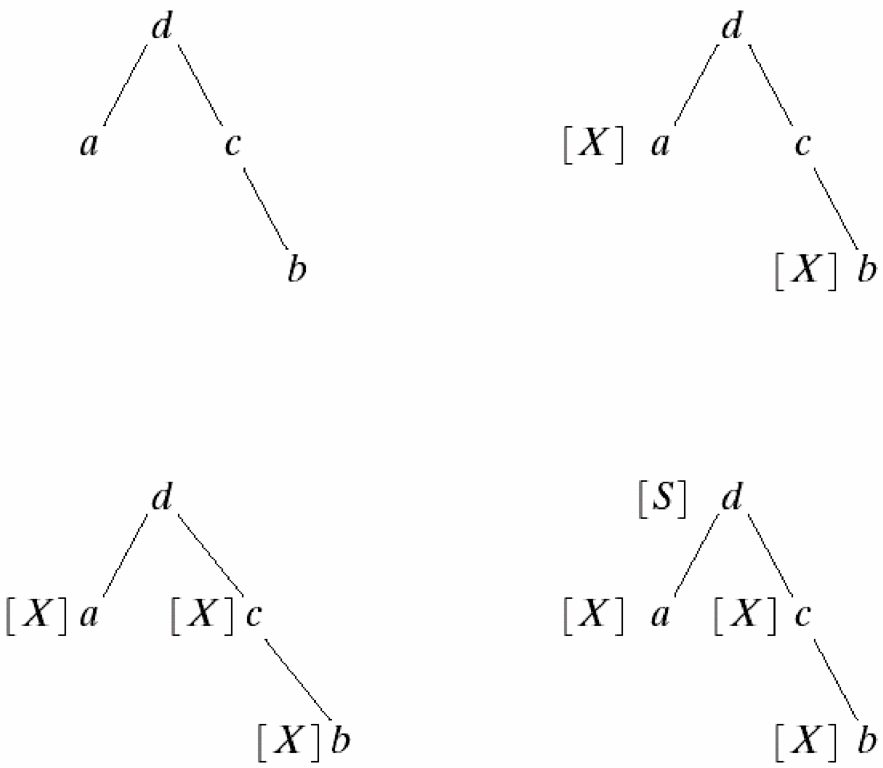
Chapter 12
Object Recognition



a
b

FIGURE 12.28 (a) An object and (b) primitives used for representing the skeleton by means of a tree grammar.

Chapter 12
Object Recognition



a b
c d

FIGURE 12.29 Processing stages of a frontier-to-root tree automaton: (a) Input tree. (b) State assignment to frontier nodes. (c) State assignment to intermediate nodes. (d) State assignment to root node.

Chapter 12
Object Recognition

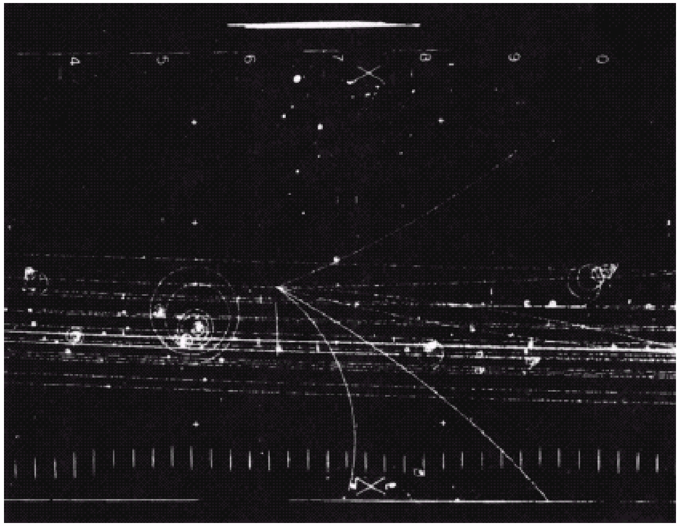
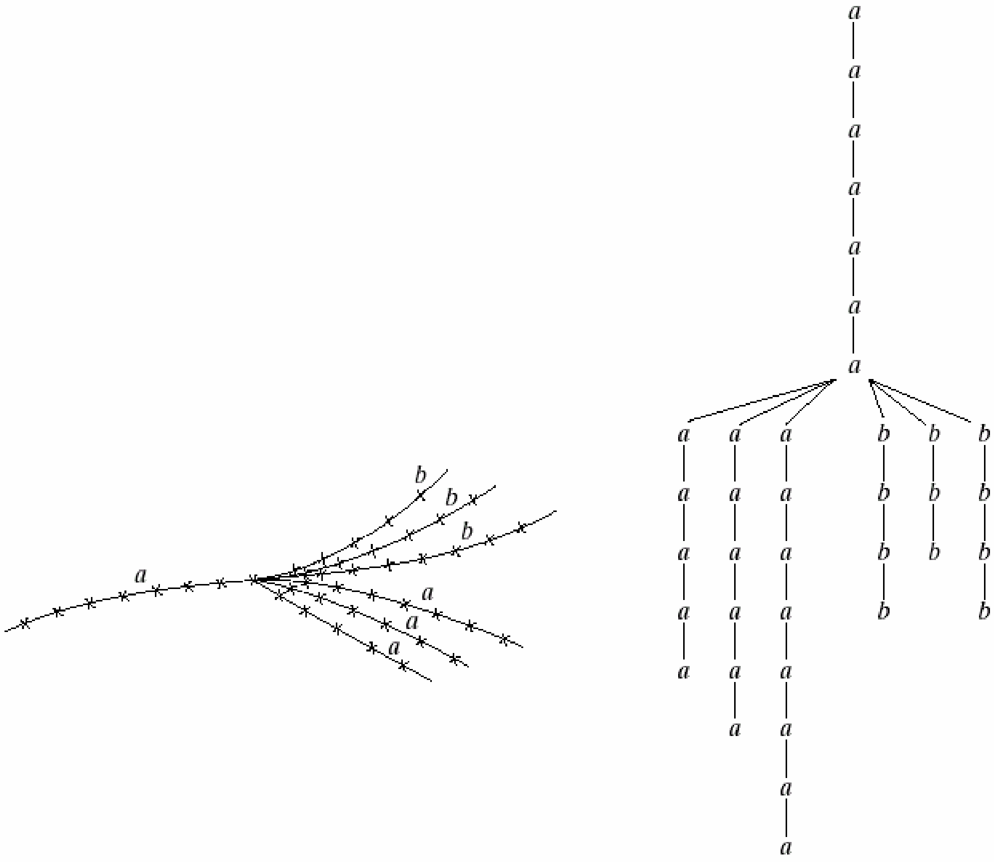


FIGURE 12.30 A bubble chamber photograph. (Fu and Bhargava.)

Chapter 12
Object Recognition



a b

FIGURE 12.31 (a) Coded event from Fig. 12.30. (b) Corresponding tree representation. (Fu and Bhargava.)

Chapter 12
Object Recognition

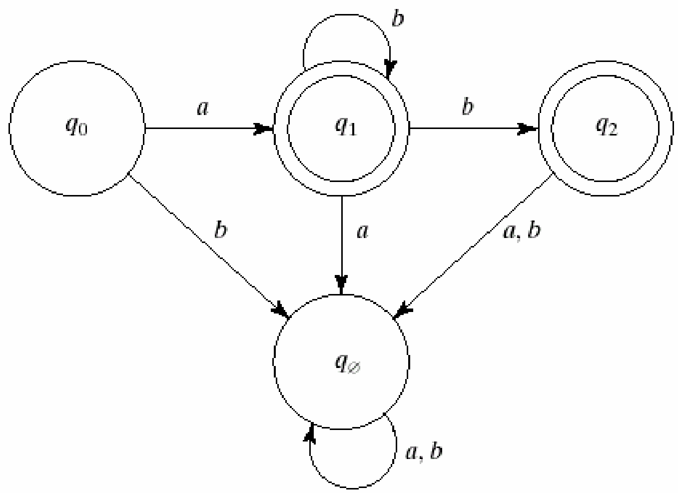


FIGURE 12.32 State diagram for the finite automaton inferred from the sample set $R^+ = \{a, ab, abb\}$.

Chapter 12
Object Recognition

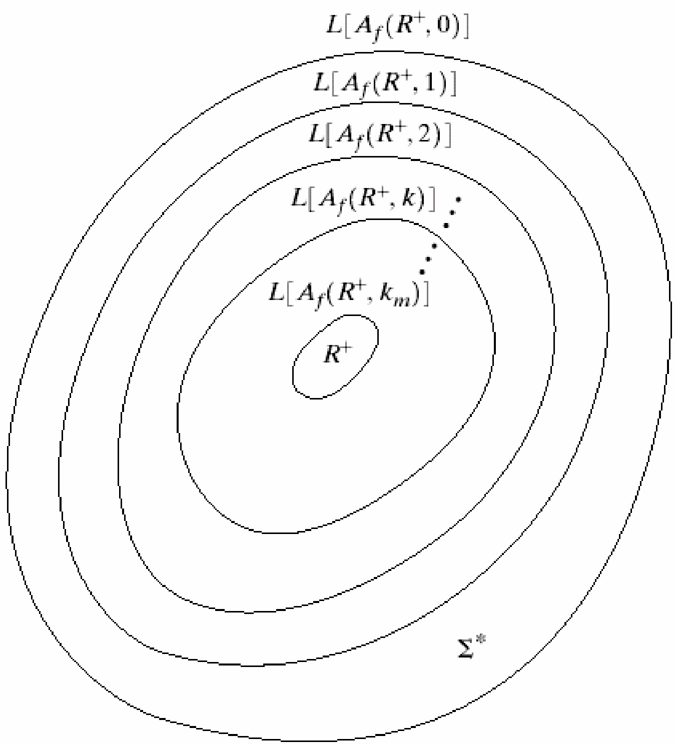


FIGURE 12.33 Relationship between $L[A_f(R^+, k)]$ and k. The value of k_m is such that $k_m \geq$ (length of the longest string in R^+).

Chapter 12
Object Recognition

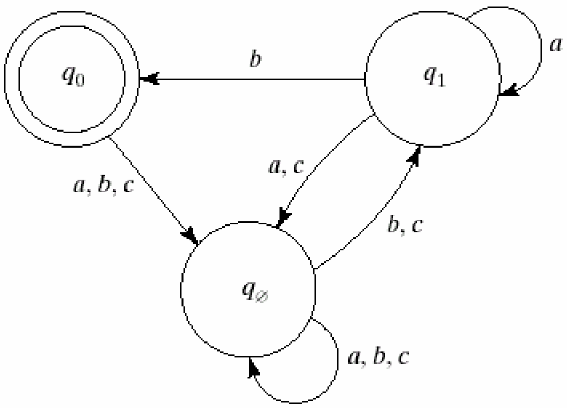


FIGURE 12.34 State diagram for the automaton $A_f(R^+, 1)$ inferred from the sample set $R^+ = \{caaab, bbaab, caab, bbab, cab, bbb, cb\}$.