Digital Image Processing

Fourth Edition

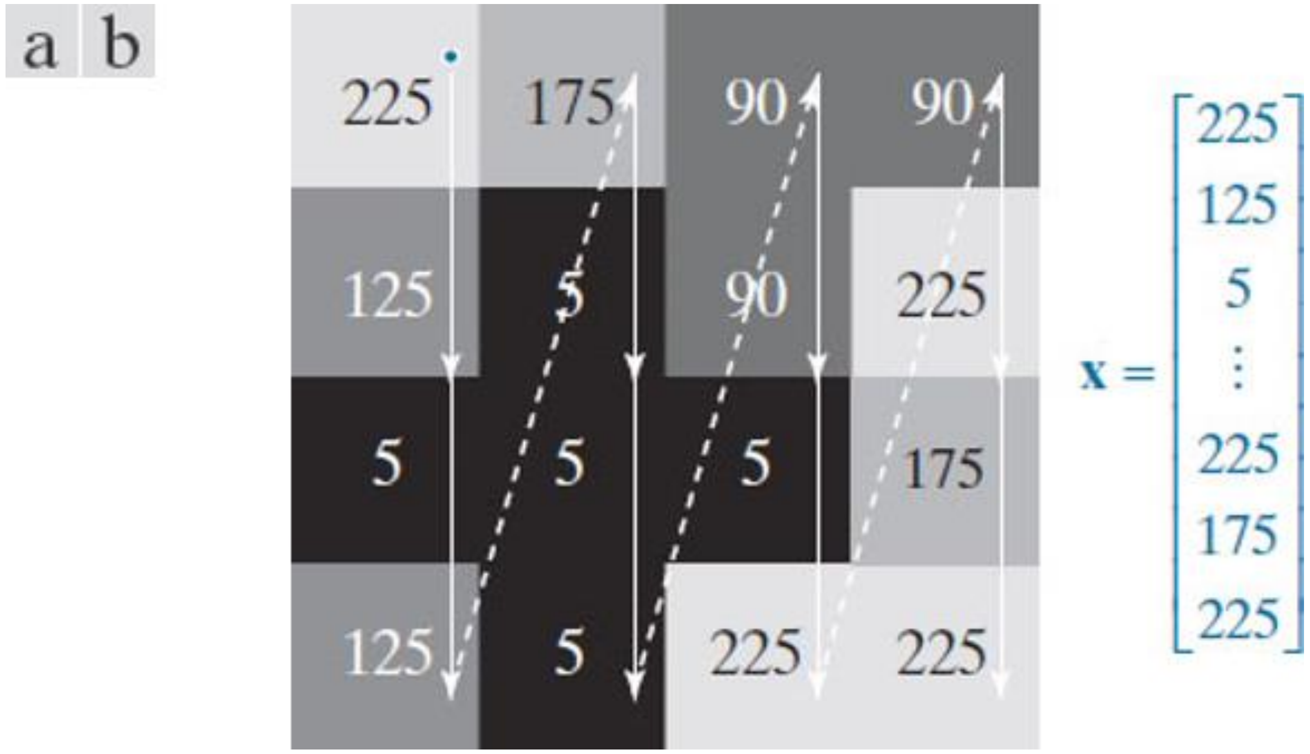
Chapter 13

Image Pattern Classification

Pearson

Figure 13.1

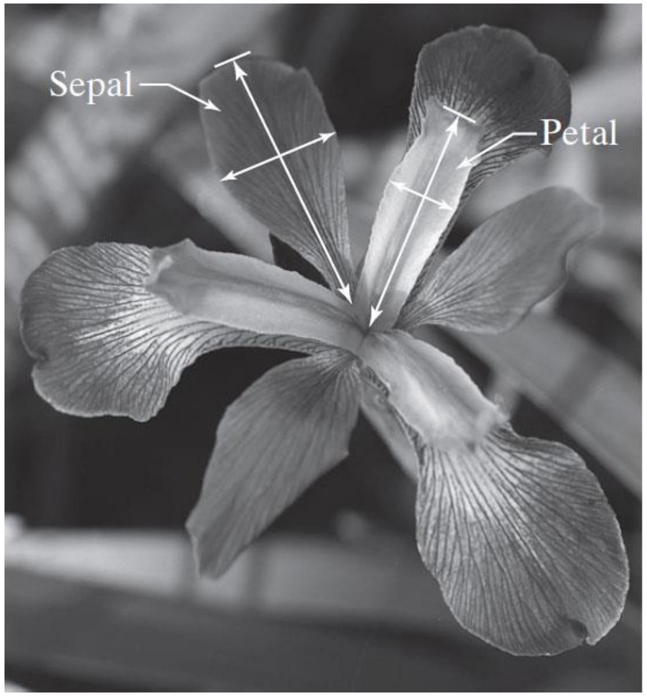
Using linear indexing to vectorize a grayscale image.



$$\mathbf{x} = \begin{bmatrix} 225 \\ 125 \\ 5 \\ \vdots \\ 225 \\ 175 \\ 225 \end{bmatrix}$$

Figure 13.2

Petal and sepal width and length measurements (see arrows) performed on iris flowers for the purpose of data classification. The image shown is of the Iris virginica gender. (Image courtesy of USDA.)

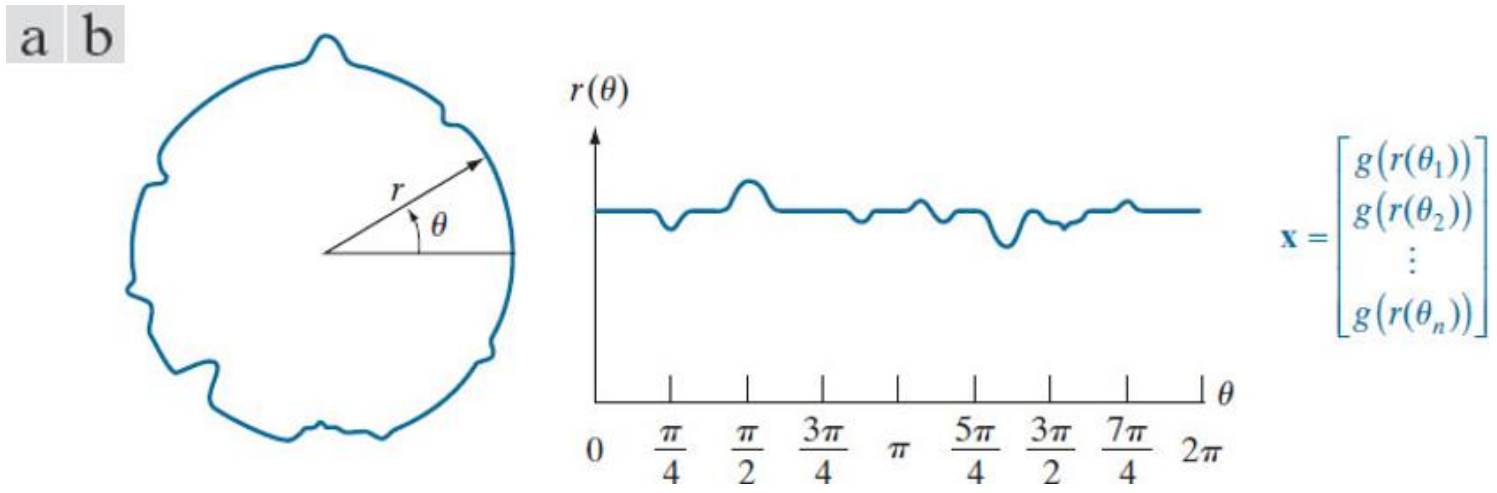


$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

x_1 = Petal width
x_2 = Petal length
x_3 = Sepal width
x_4 = Sepal length

Figure 13.3

(a) A noisy object boundary, and (b) its corresponding signature.

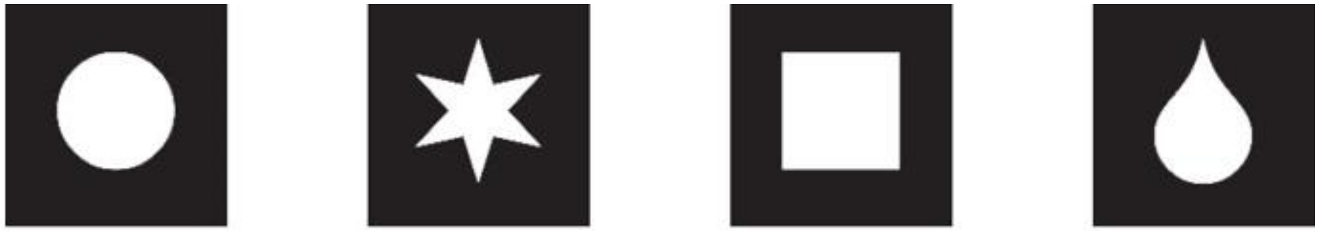


$$\mathbf{x} = \begin{bmatrix} g(r(\theta_1)) \\ g(r(\theta_2)) \\ \vdots \\ g(r(\theta_n)) \end{bmatrix}$$

Figure 13.4

Pattern vectors whose components capture both boundary and regional characteristics.

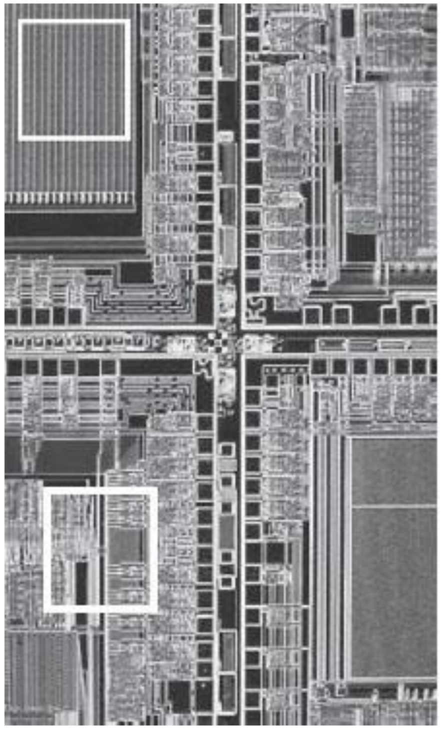
a b c d



$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \begin{aligned} x_1 &= \text{compactness} \\ x_2 &= \text{circularity} \\ x_3 &= \text{eccentricity} \end{aligned}$$

Figure 13.5

An example of pattern vectors based on properties of subimages. See Table 12.3 for an explanation of the components of **x**.

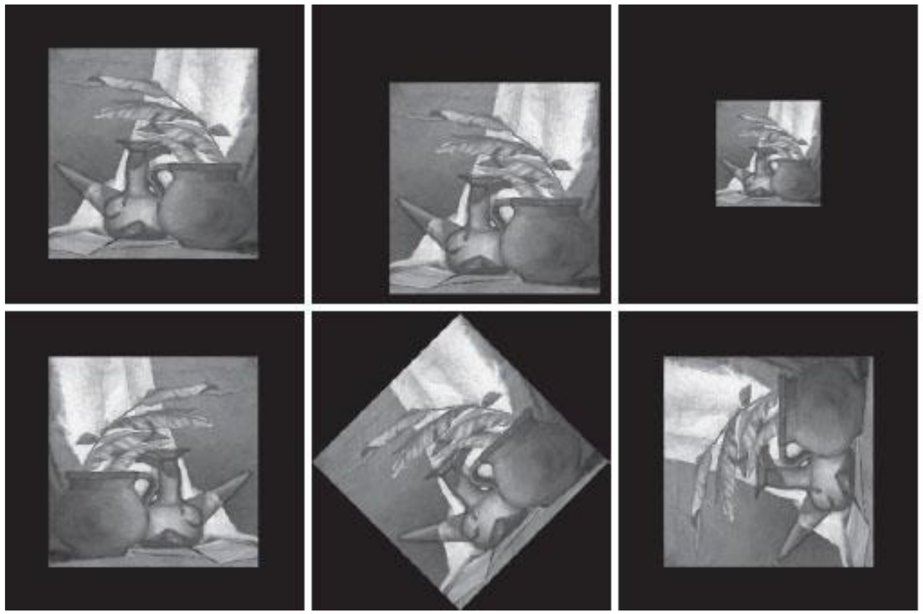


$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}$$

x_1 = max probability
x_2 = correlation
x_3 = contrast
x_4 = uniformity
x_5 = homogeneity
x_6 = entropy

Figure 13.6

Feature vectors with components that are invariant to transformations such as rotation, scaling, and translation. The vector components are moment invariants.

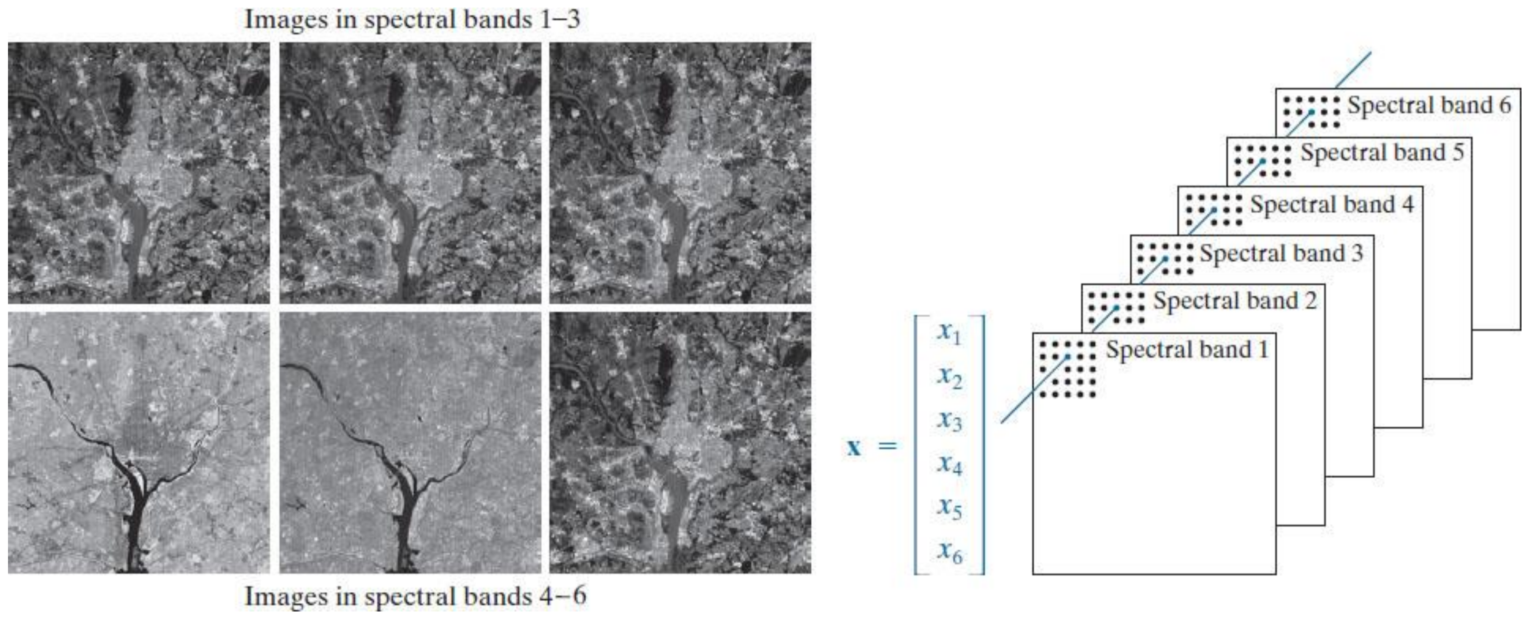


$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \\ \phi_7 \end{bmatrix}$$

The ϕ's are moment invariants

Figure 13.7

Pattern (feature) vectors formed by concatenating corresponding pixels from a set of registered images. (Original images courtesy of NASA.)



$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}$$

Figure 13.8

Symbol string generated from a polygonal approximation of the boundaries of medicine bottles.

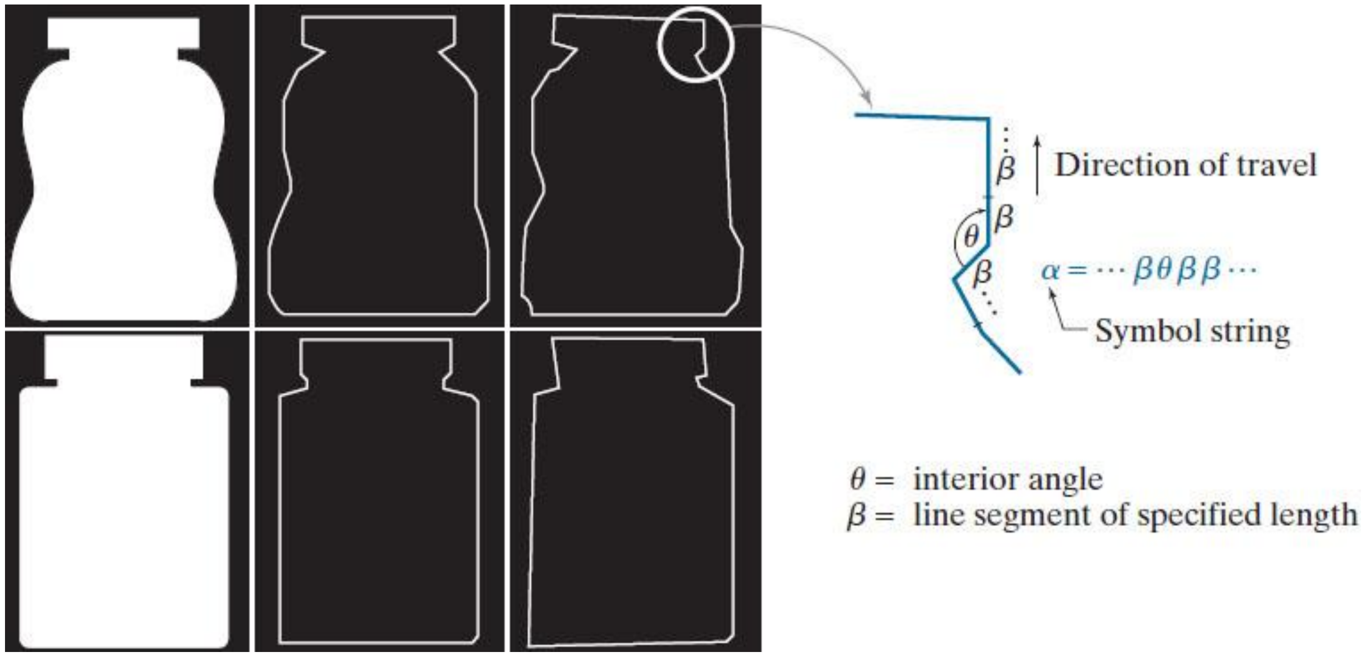
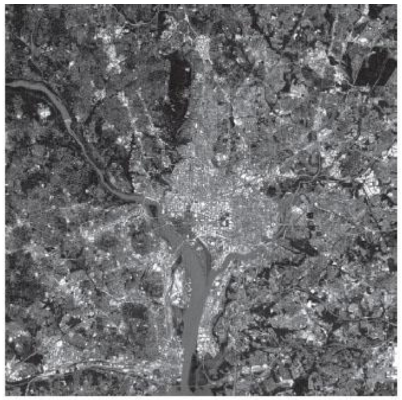

Direction of travel
β
β
θ
β
$\alpha = \cdots \beta\theta\beta\beta \cdots$
Symbol string
$\theta =$ interior angle
$\beta =$ line segment of specified length

Figure 13.9

Tree representation of a satellite image showing a heavily built downtown area (Washington, D.C.) and surrounding residential areas. (Original image courtesy of NASA.)



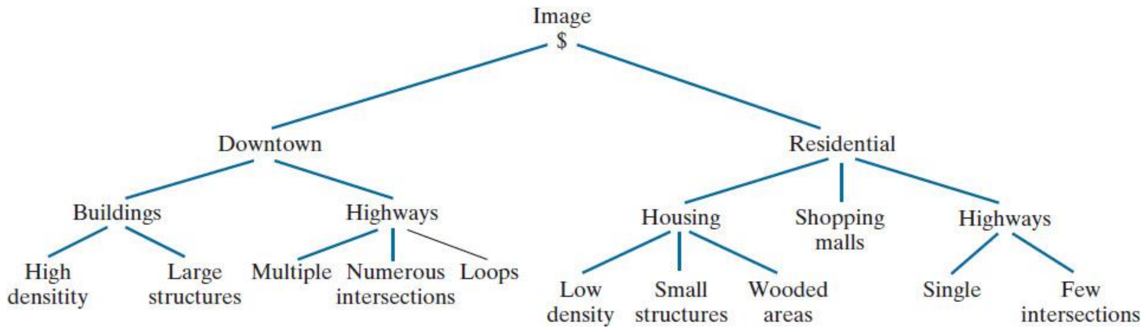


Figure 13.10

Decision boundary of a minimum distance classifier (based on two measurements) for the classes of Iris versicolor and Iris setosa. The dark dot and square are the means of the two classes.

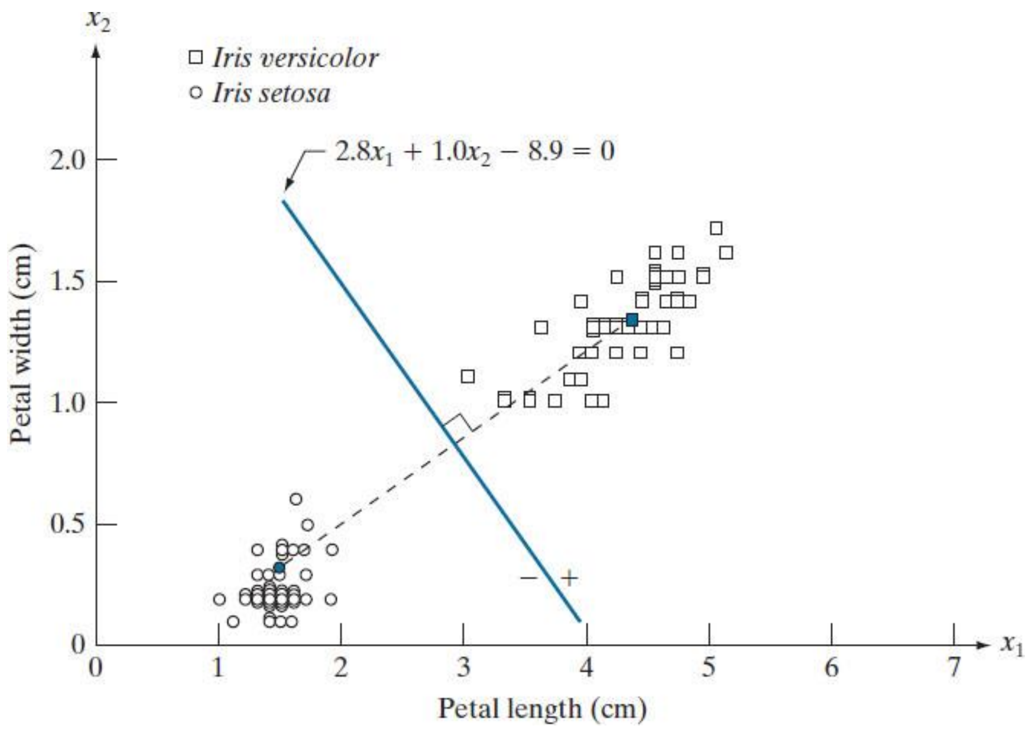


Figure 13.11

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

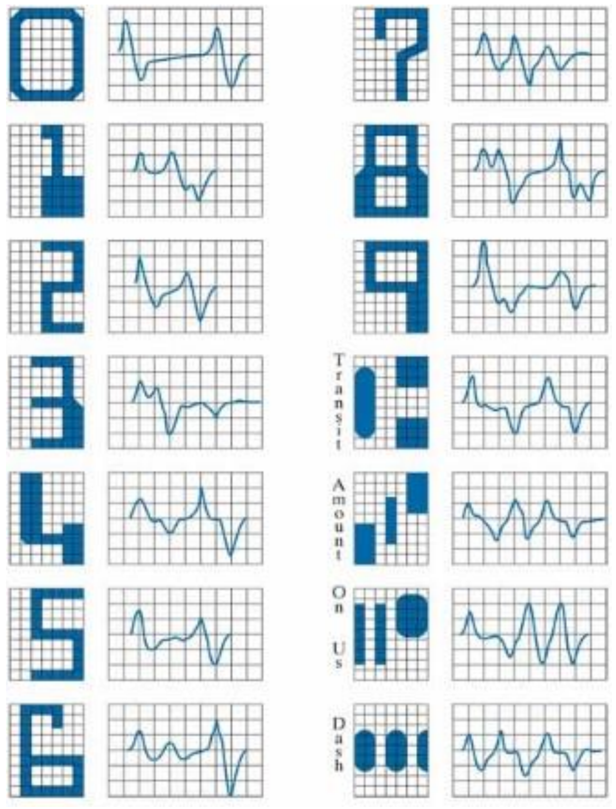


Figure 13.12

The mechanics of template matching.

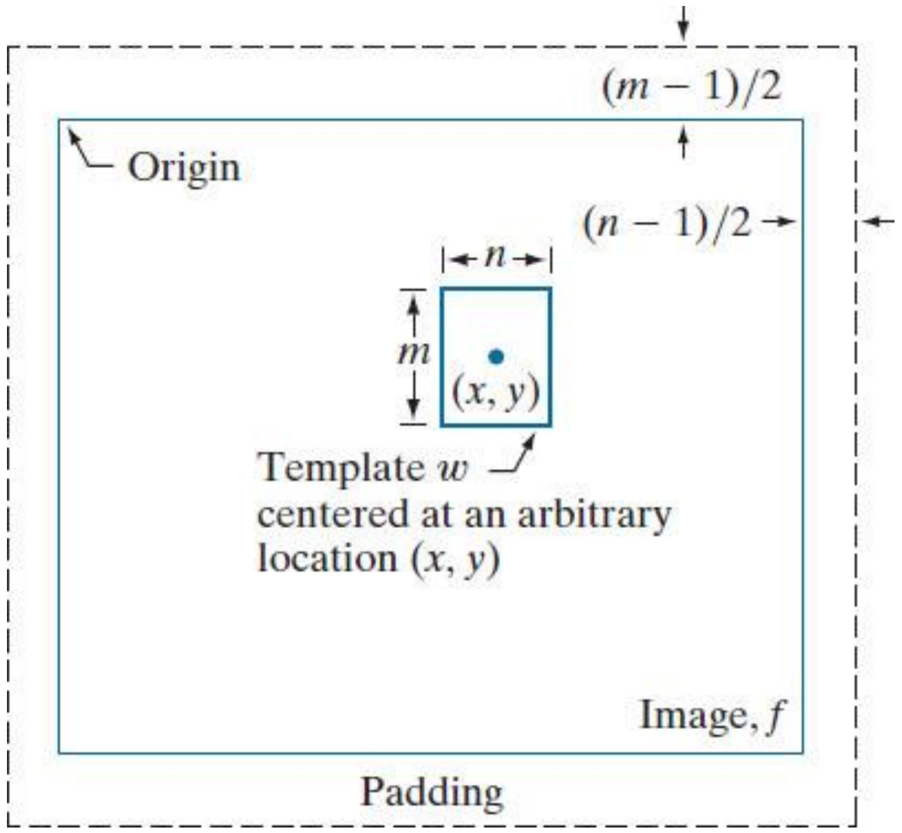


Figure 13.13

(a) 913×913 satellite image of Hurricane Andrew. (b) 31×31 template of the eye of the storm. (c) Correlation coefficient shown as an image (note the brightest point, indicated by an arrow). (d) Location of the best match (identified by the arrow). This point is a single pixel, but its size was enlarged to make it easier to see. (Original image courtesy of NOAA.)

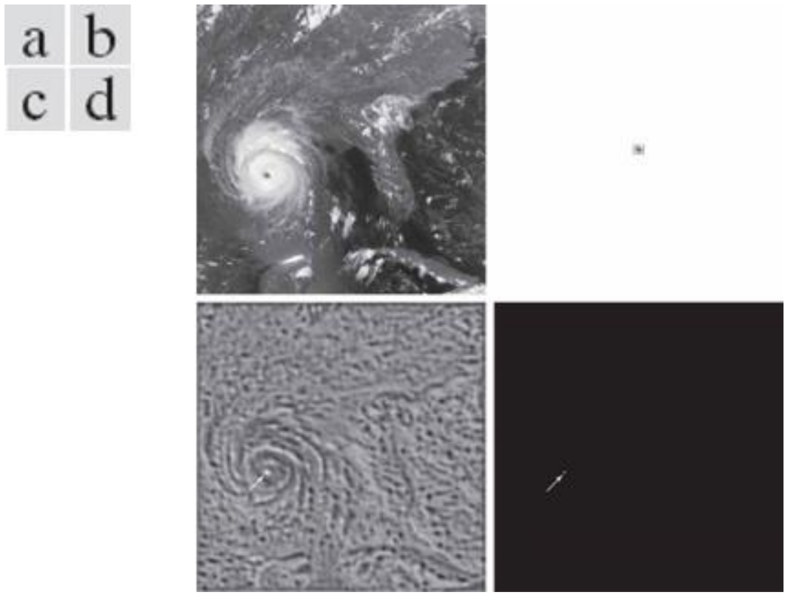
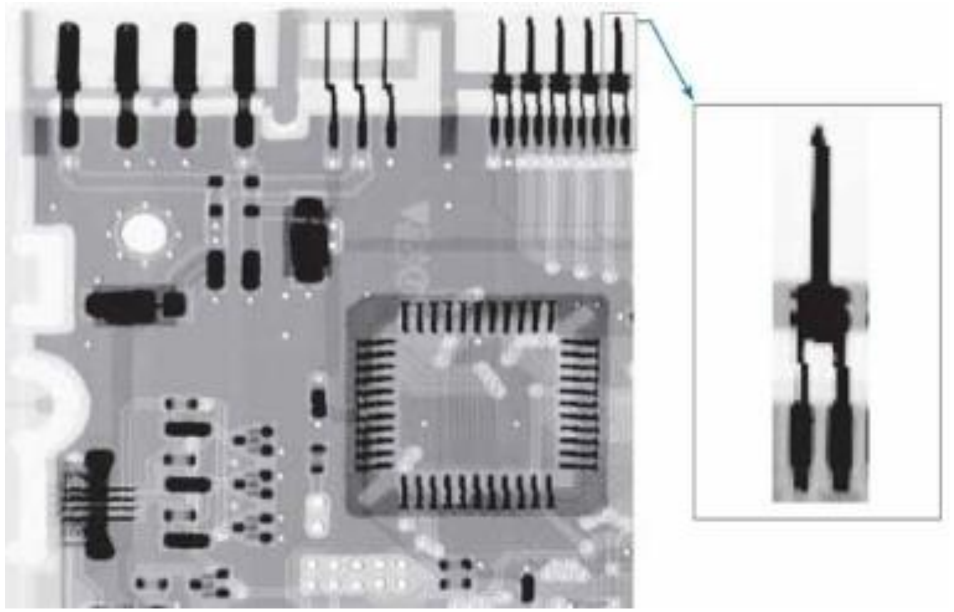


Figure 13.14

Circuit board image of size 948×948 pixels, and a subimage of one of the connectors. The subimage is of size 212×212 pixels, shown zoomed on the right for clarity. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)



Pearson

Figure 13.15

Keypoints found by SIFT. The large image has 2714 keypoints (visible as faint gray lines). The subimage has 35 keypoints. This is a separate image, and SIFT found its keypoints independently of the large image. The zoomed section is shown for clarity.

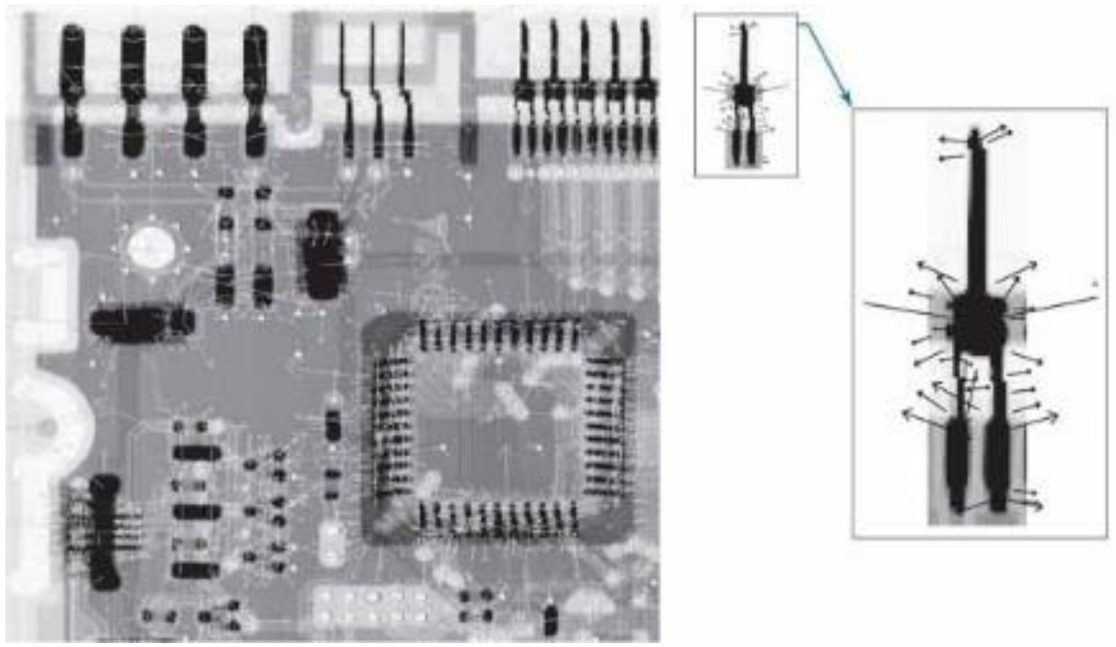


Figure 13.16

Matches found by SIFT between the large and small images. A total of 41 matching pairs were found. They are shown connected by straight lines. Only three of the matches were “real” errors (labeled “Errors” in the figure).

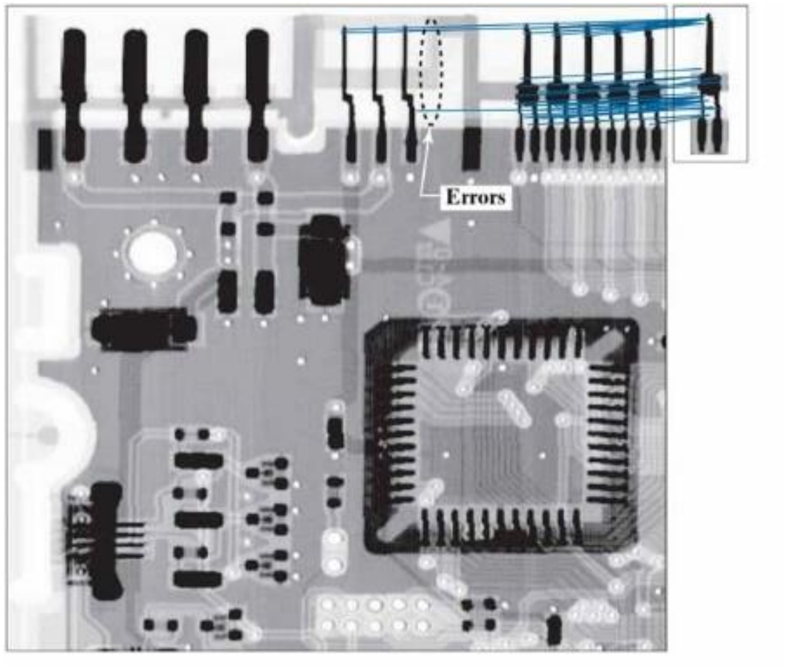
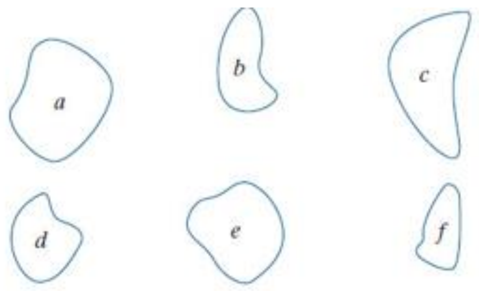


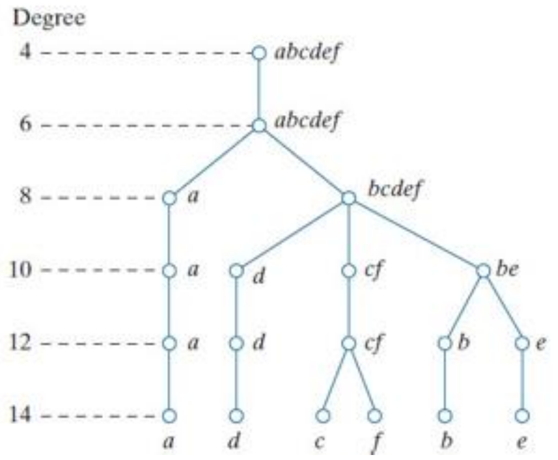
Figure 13.17

(a) Shapes. (b) Similarity tree. (c) Similarity matrix. (Bribiesca and Guzman.)

a

b c





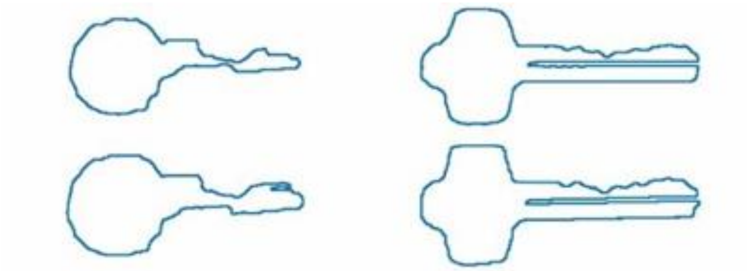
	a	b	c	d	e	f
a	∞	6	6	6	6	6
b		∞	8	8	10	8
c			∞	8	8	12
d				∞	8	8
e					∞	8
f						∞

Pearson

Figure 13.18

(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.)

a	b
c	d
e	f
g	



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

Figure 13.19

Probability density functions for two 1-D pattern classes. Point x_0 (at the intersection of the two curves) is the Bayes decision boundary if the two classes are equally likely to occur.

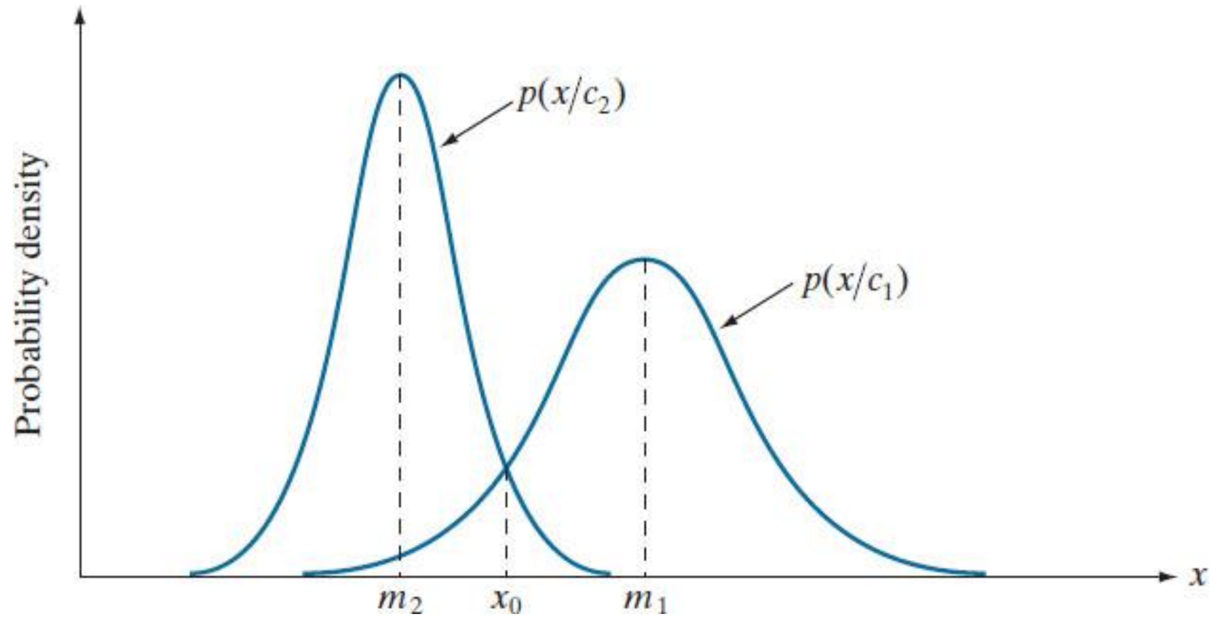


Figure 13.20

Two simple pattern classes and the portion of their Bayes decision boundary (shaded) that intersects the cube.

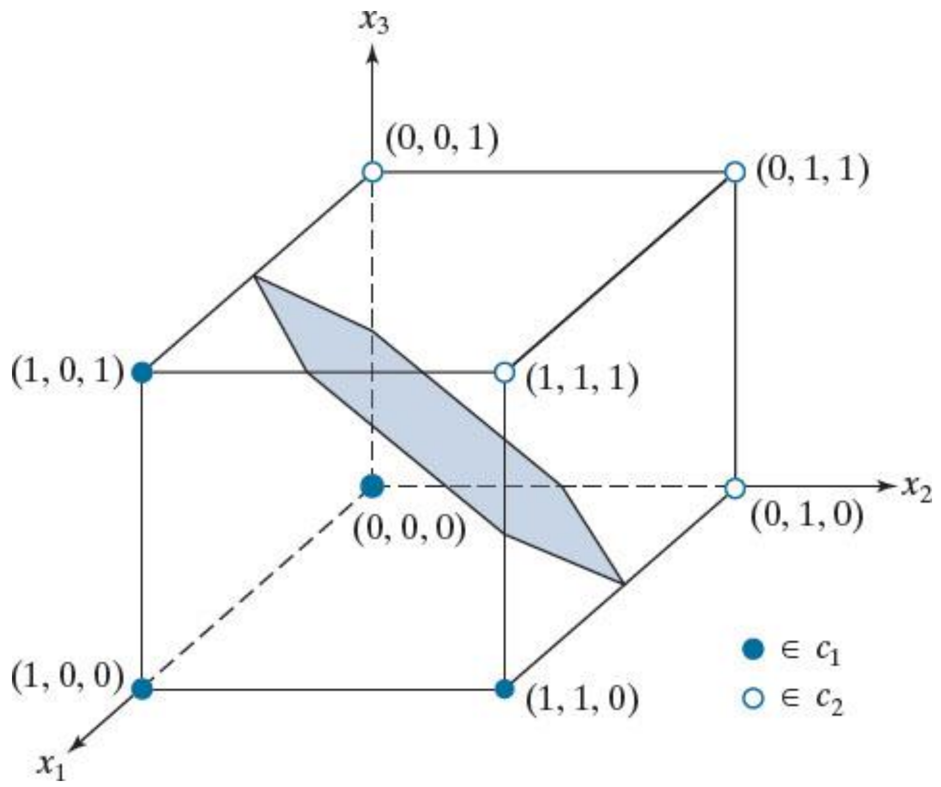


Figure 13.21

Bayes classification of multispectral data. (a)–(d) Images in the visible blue, visible green, visible red, and near infrared wavelength bands. (e) Masks for regions of water (labeled 1), urban development (labeled 2), and vegetation (labeled 3). (f) Results of classification; the black dots denote points classified incorrectly. The other (white) points were classified correctly. (g) All image pixels classified as water (in white). (h) All image pixels classified as urban development (in white). (i) All image pixels classified as vegetation (in white).

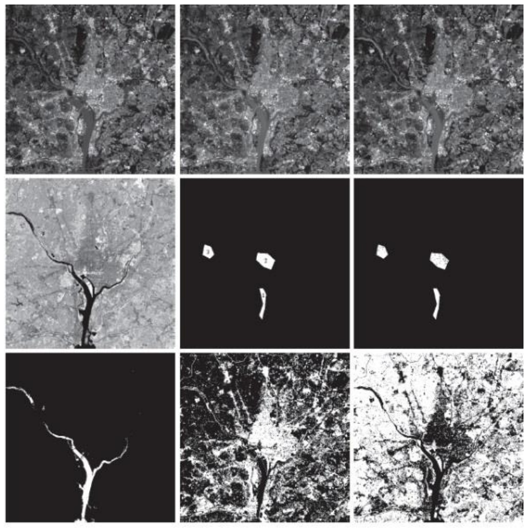
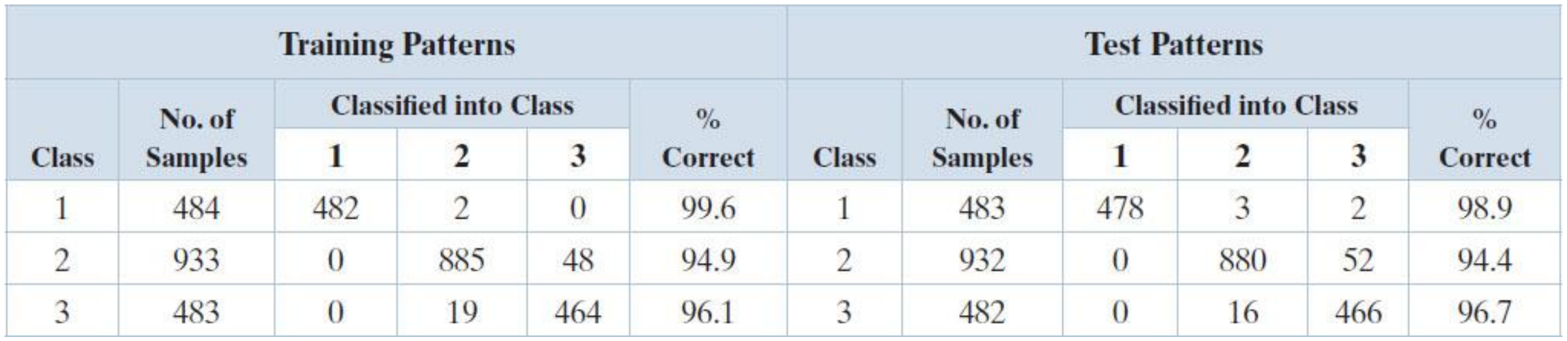


Table 13.1

Bayes classification of multispectral image data. Classes 1, 2, and 3 are water, urban, and vegetation, respectively.



Training Patterns						Test Patterns					
	No. of	Classified into Class			%		No. of	Classified into Class			%
Class	Samples	1	2	3	Correct	Class	Samples	1	2	3	Correct
1	484	482	2	0	99.6	1	483	478	3	2	98.9
2	933	0	885	48	94.9	2	932	0	880	52	94.4
3	483	0	19	464	96.1	3	482	0	16	466	96.7

Pearson

Figure 13.22

(a) The simplest two-class example in 2-D, showing one possible decision boundary out of an infinite number of such boundaries. (b) Same as (a), but with the decision boundary expressed using more general notation.

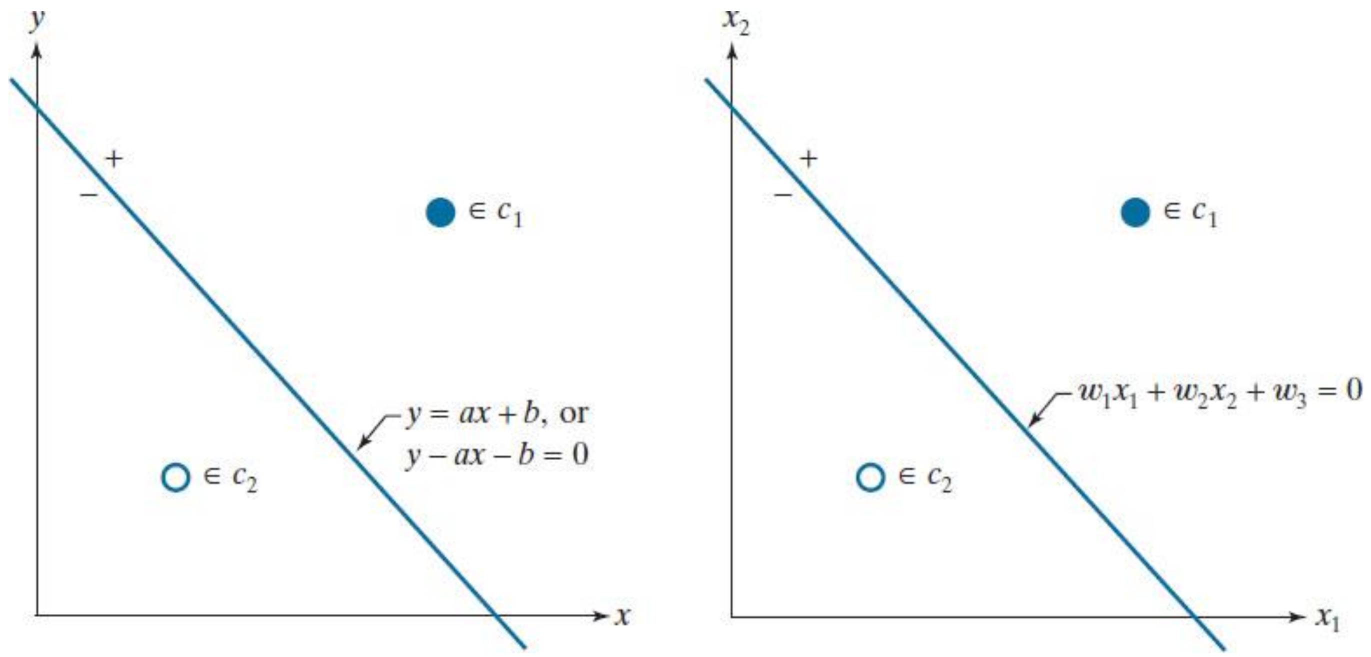


Figure 13.23

Schematic of a perceptron, showing the operations it performs.

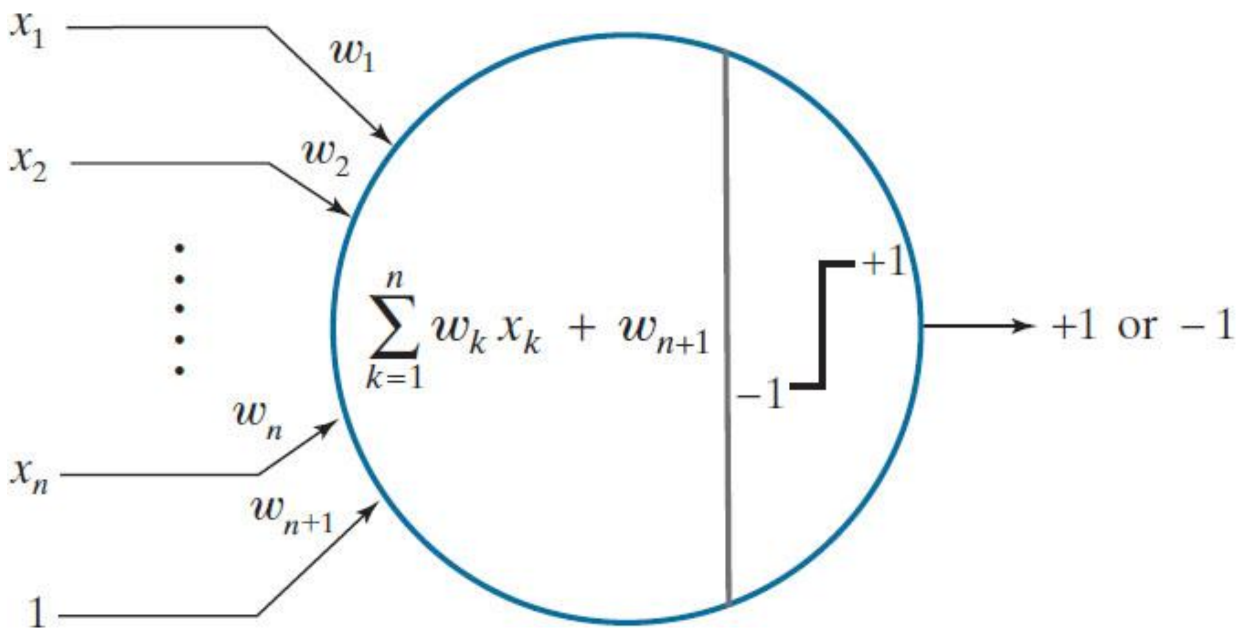


Figure 13.24

(a) Segment of the decision boundary learned by the perceptron algorithm. (b) Section of the decision surface. The decision boundary is the intersection of the decision surface with the x_1x_2 plane.

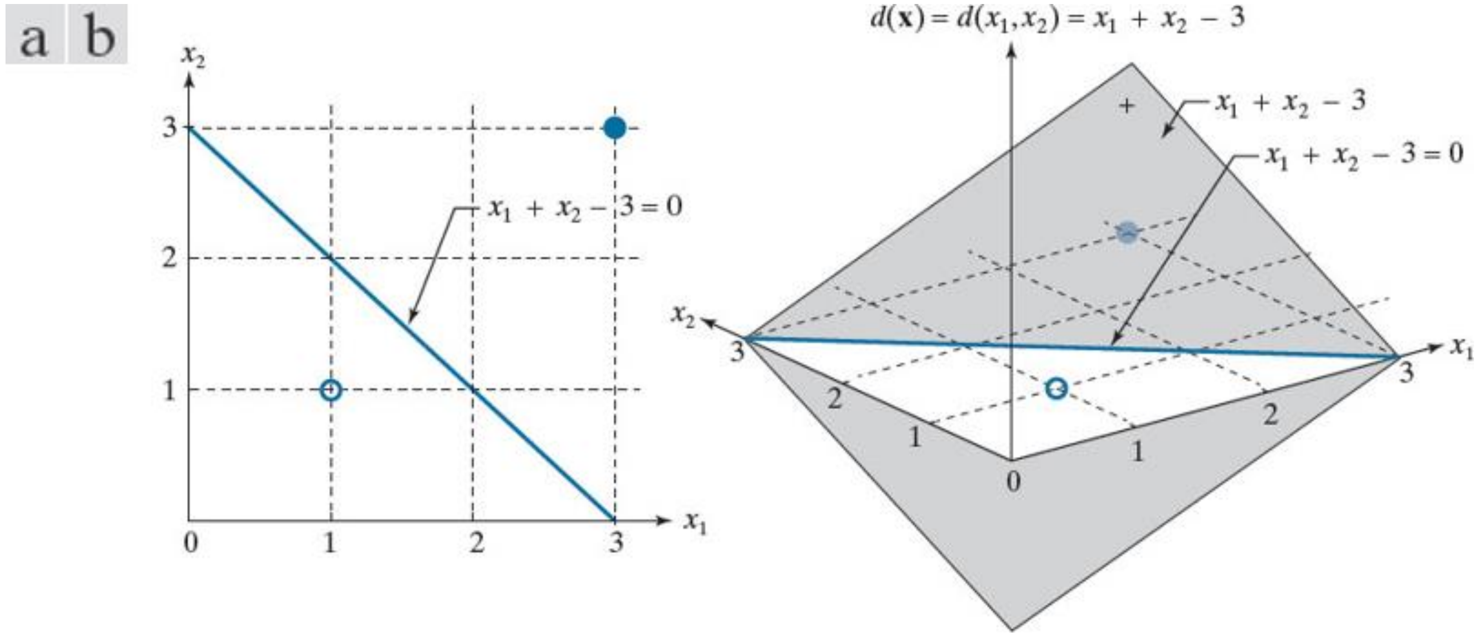


Figure 13.25

Plots of *E* as a function of w*x* for *r* = 1. (a) A value of a that is too small can slow down convergence. (b) If a is too large, large oscillations or divergence may occur. (c) Shape of the error function in 2-D.

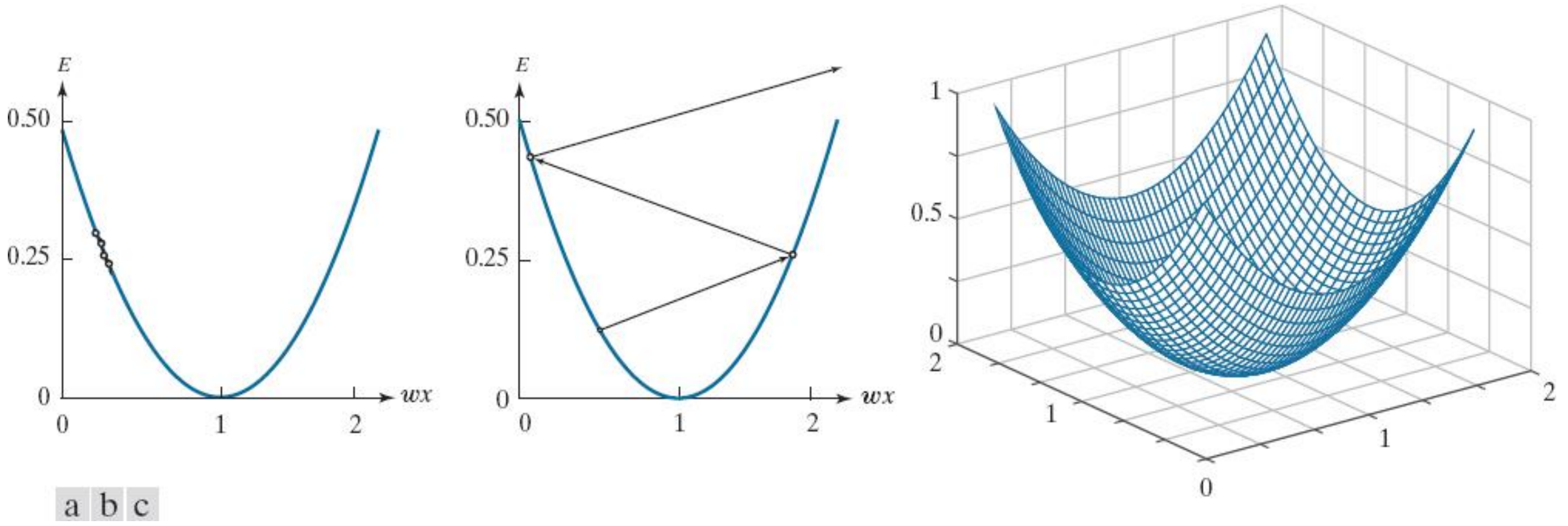


Figure 13.26

MSE as a function of epoch for: (a) the linearly separable Iris classes (setosa and versicolor); and (b) the linearly nonseparable Iris classes (versicolor and virginica).

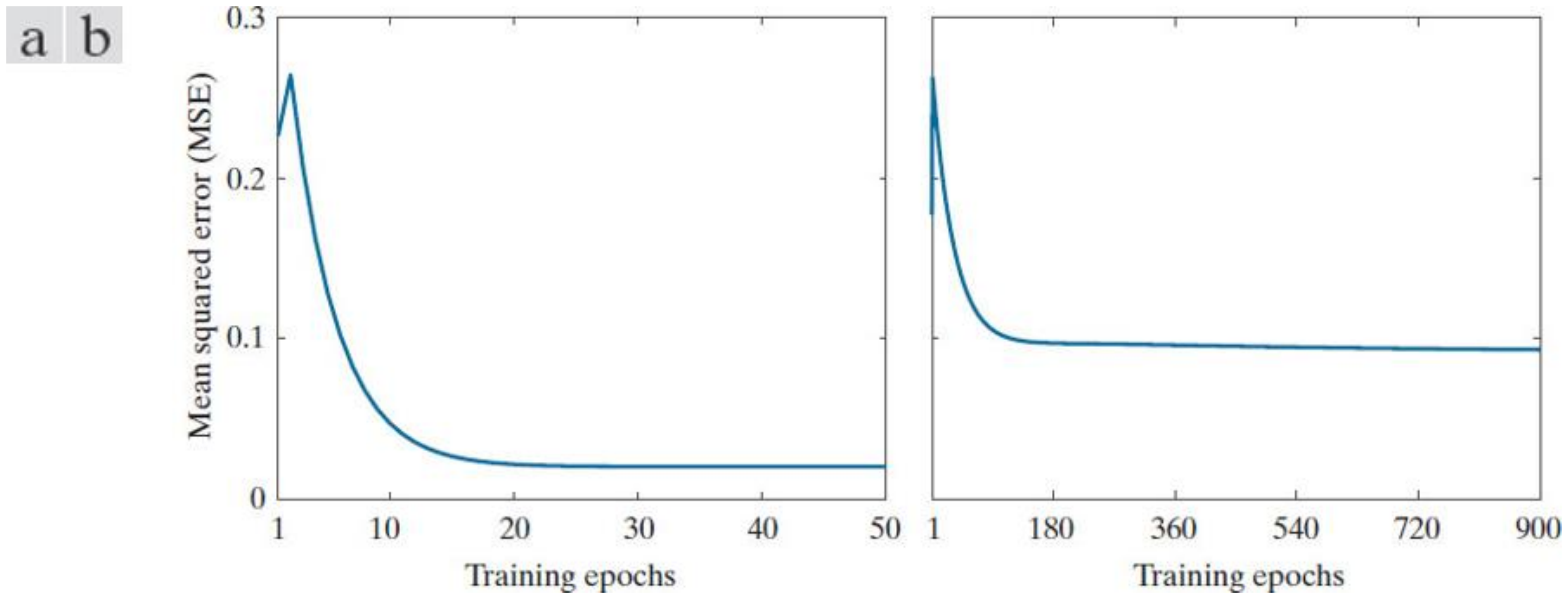
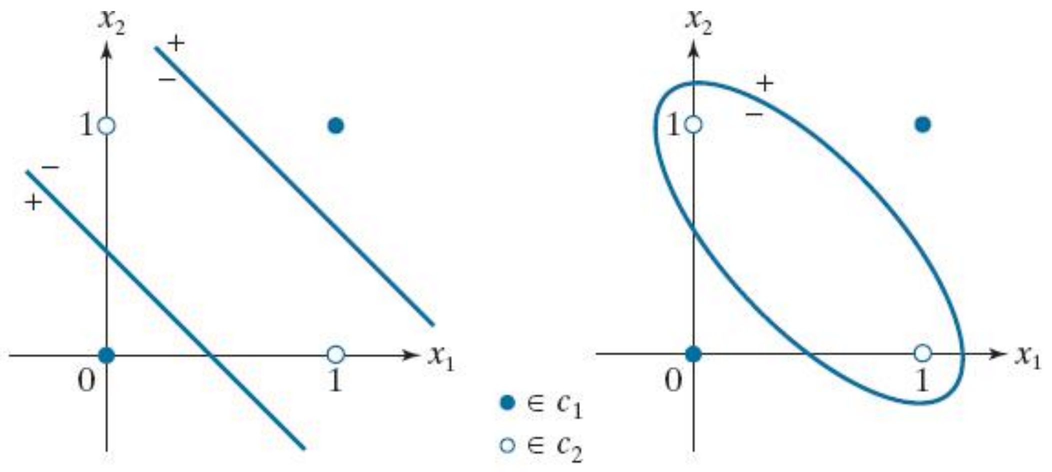


Figure 13.27

The XOR classification problem in 2-D. (a) Truth table definition of the XOR operator. (b) 2-D pattern classes formed by assigning the XOR truth values (1) to one pattern class, and false values (0) to another. The simplest decision boundary between the two classes consists of two straight lines. (c) Nonlinear (quadratic) boundary separating the two classes.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0



a b c

Figure 13.28

(a) Minimum Percheron solution to the XOR problem in 2-D. (b) A solution that implements the XOR truth table in Fig. 13.27(a).

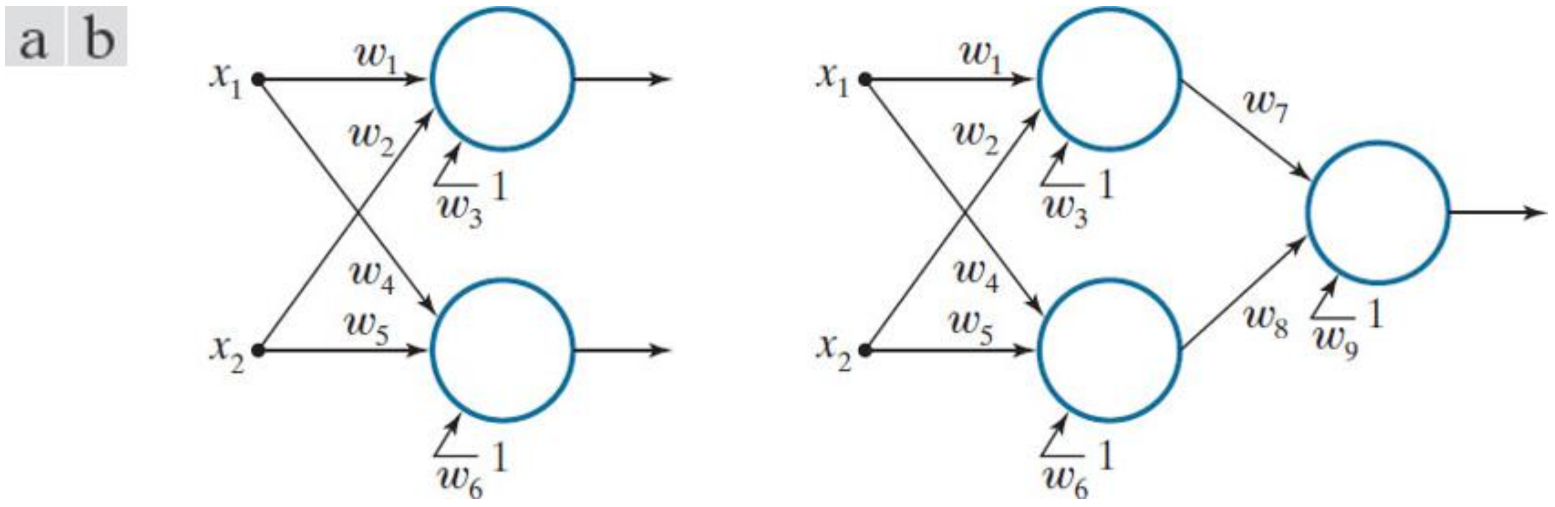
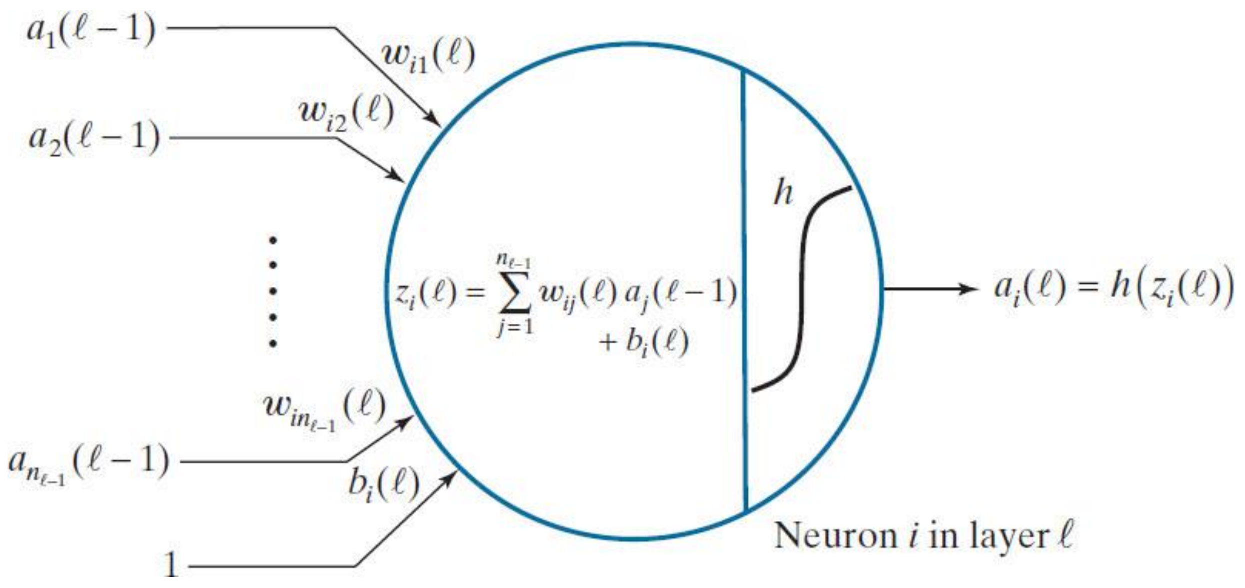


Figure 13.29

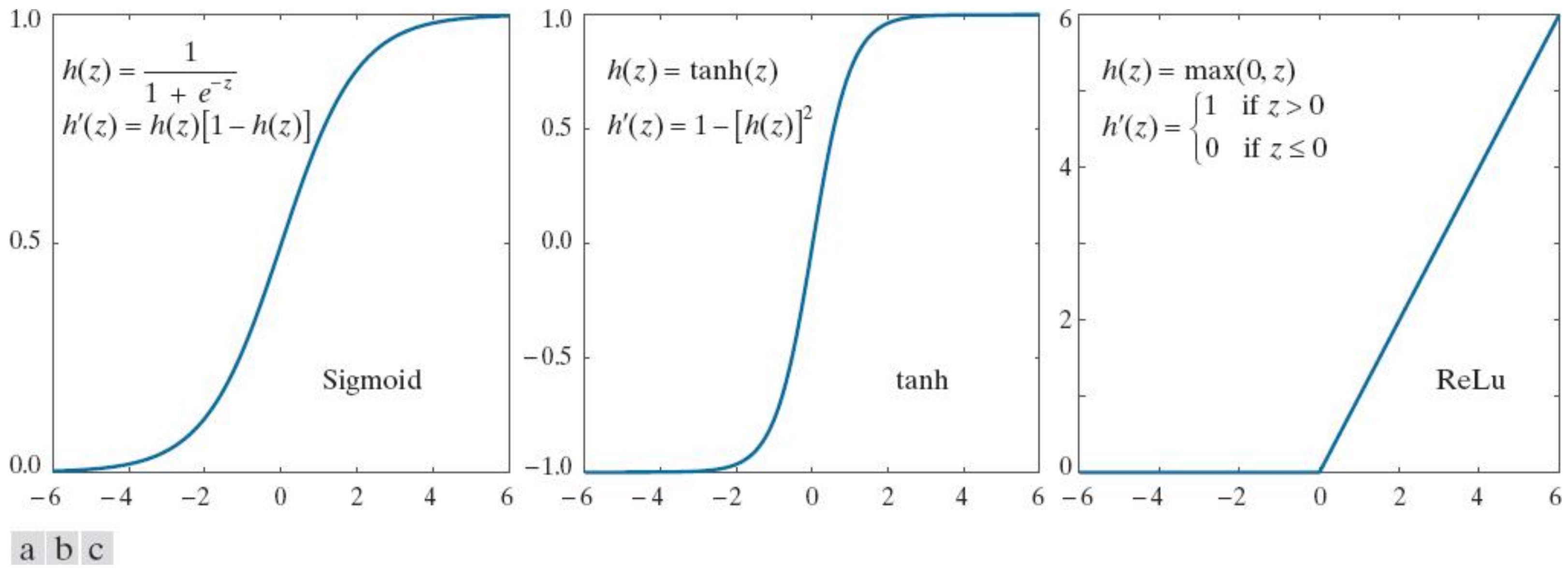
Model of an artificial neuron, showing all the operations it performs. The “l” is used to denote a particular layer in a layered network.



$a_1(\ell-1)$
$w_{i1}(\ell)$
$a_2(\ell-1)$
$w_{i2}(\ell)$
$w_{in_{\ell-1}}(\ell)$
$a_{n_{\ell-1}}(\ell-1)$
$b_i(\ell)$
1
$z_i(\ell)=\sum_{j=1}^{n_{\ell-1}} w_{ij}(\ell)\,a_j(\ell-1)+b_i(\ell)$
h
$a_i(\ell)=h\big(z_i(\ell)\big)$
Neuron i in layer ℓ

Figure 13.30

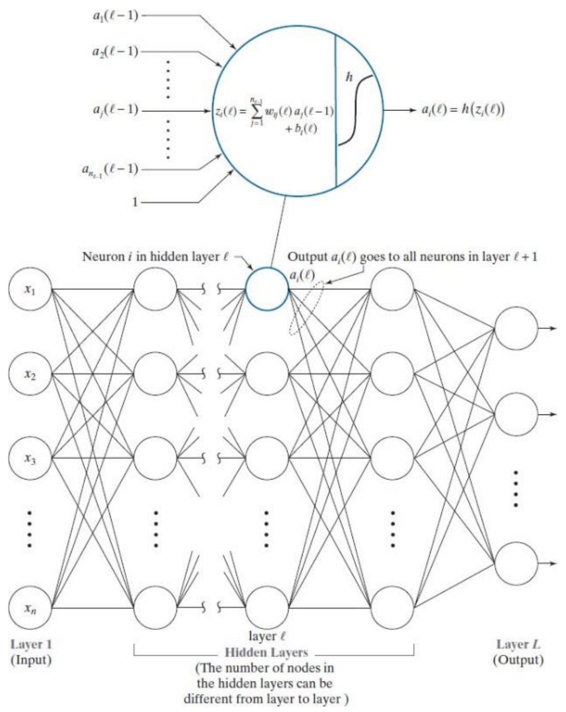
Various activation functions. (a) Sigmoid. (b) Hyperbolic tangent (also has a sigmoid shape, but it is centered about 0 in both dimensions). (c) Rectifier linear unit (ReLU).



Pearson

Figure 13.31

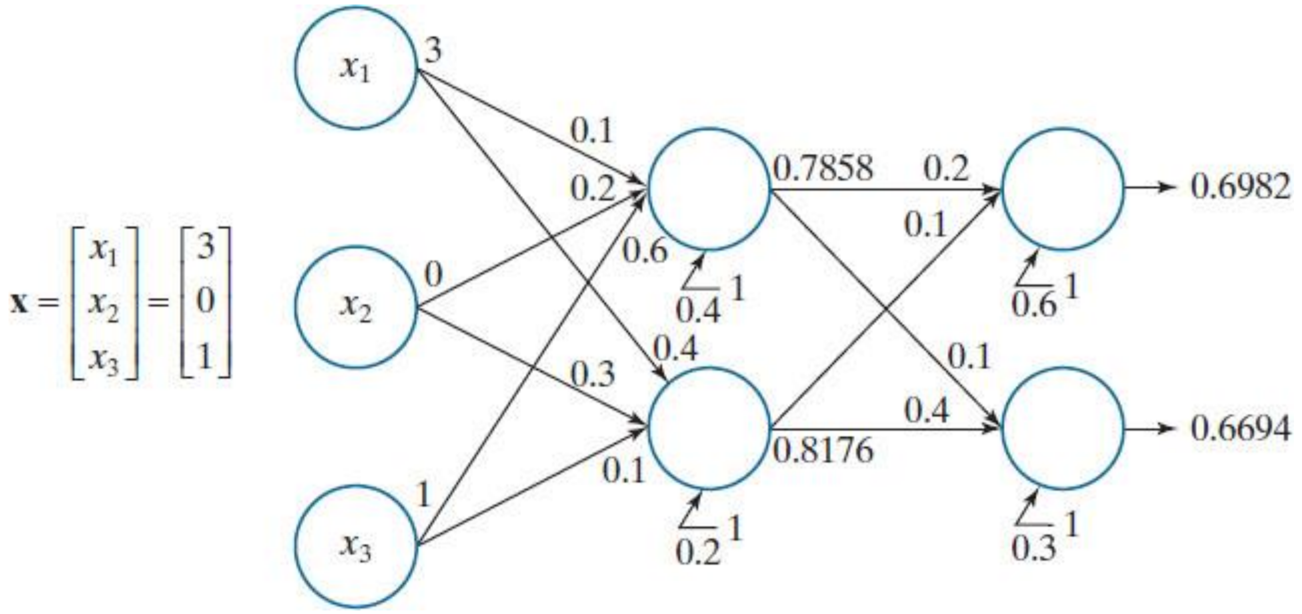
General model of a feedforward, fully connected neural net. The neuron is the same as in Fig. 13.29. Note how the output of each neuron goes to the input of all neurons in the following layer, hence the name fully connected for this type of architecture.



$a_1(\ell-1)$
$a_2(\ell-1)$
$a_j(\ell-1)$
$a_{n_{\ell-1}}(\ell-1)$
1
$z_i(\ell)=\sum_{j=1}^{n_{\ell-1}} w_{ij}(\ell)\,a_j(\ell-1)+b_i(\ell)$
h
$a_i(\ell)=h(z_i(\ell))$
Neuron i in hidden layer ℓ
Output $a_i(\ell)$ goes to all neurons in layer $\ell+1$
$a_i(\ell)$
x_1
x_2
x_3
x_n
Layer 1 (Input)
layer ℓ
Hidden Layers (The number of nodes in the hidden layers can be different from layer to layer)
Layer L (Output)

Figure 13.32

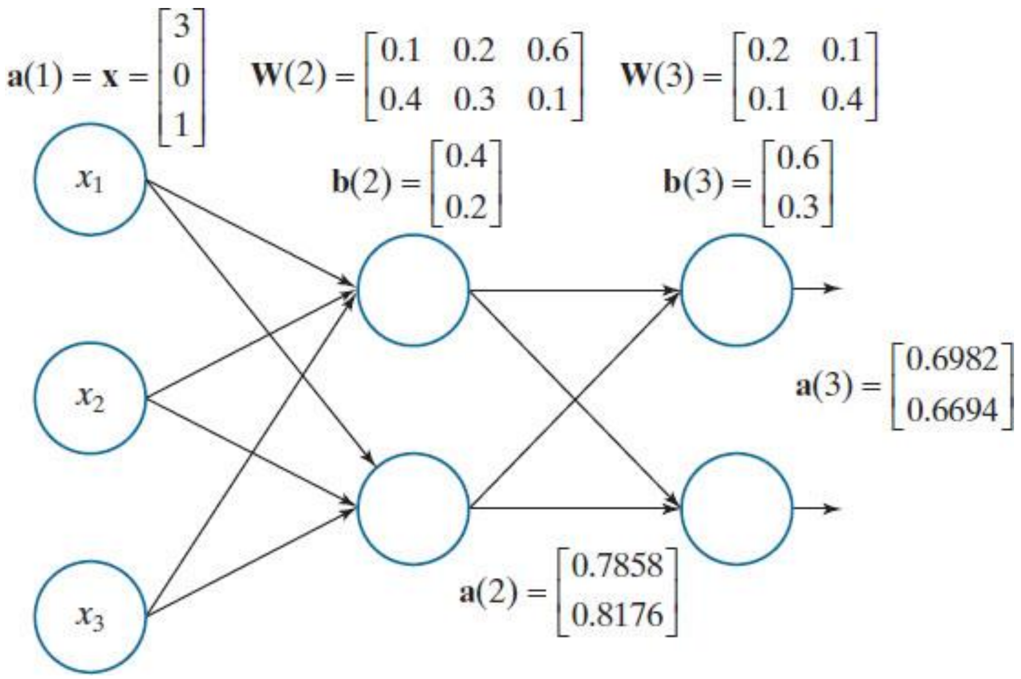
A small, fully connected, feedforward net with labeled weights, biases, and outputs. The activation function is a sigmoid.



$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 1 \end{bmatrix}$
x_1
x_2
x_3
3
0
1
0.1
0.2
0.6
0.4
0.3
0.1
0.4
1
0.2
1
0.7858
0.8176
0.2
0.1
0.1
0.4
0.6
1
0.3
1
0.6982
0.6694

Figure 13.33

Same as Fig.13.32, but using matrix labeling.



$\mathbf{a}(1) = \mathbf{x} = \begin{bmatrix} 3 \\ 0 \\ 1 \end{bmatrix}$
$\mathbf{W}(2) = \begin{bmatrix} 0.1 & 0.2 & 0.6 \\ 0.4 & 0.3 & 0.1 \end{bmatrix}$
$\mathbf{W}(3) = \begin{bmatrix} 0.2 & 0.1 \\ 0.1 & 0.4 \end{bmatrix}$
$\mathbf{b}(2) = \begin{bmatrix} 0.4 \\ 0.2 \end{bmatrix}$
$\mathbf{b}(3) = \begin{bmatrix} 0.6 \\ 0.3 \end{bmatrix}$
x_1
x_2
x_3
$\mathbf{a}(3) = \begin{bmatrix} 0.6982 \\ 0.6694 \end{bmatrix}$
$\mathbf{a}(2) = \begin{bmatrix} 0.7858 \\ 0.8176 \end{bmatrix}$

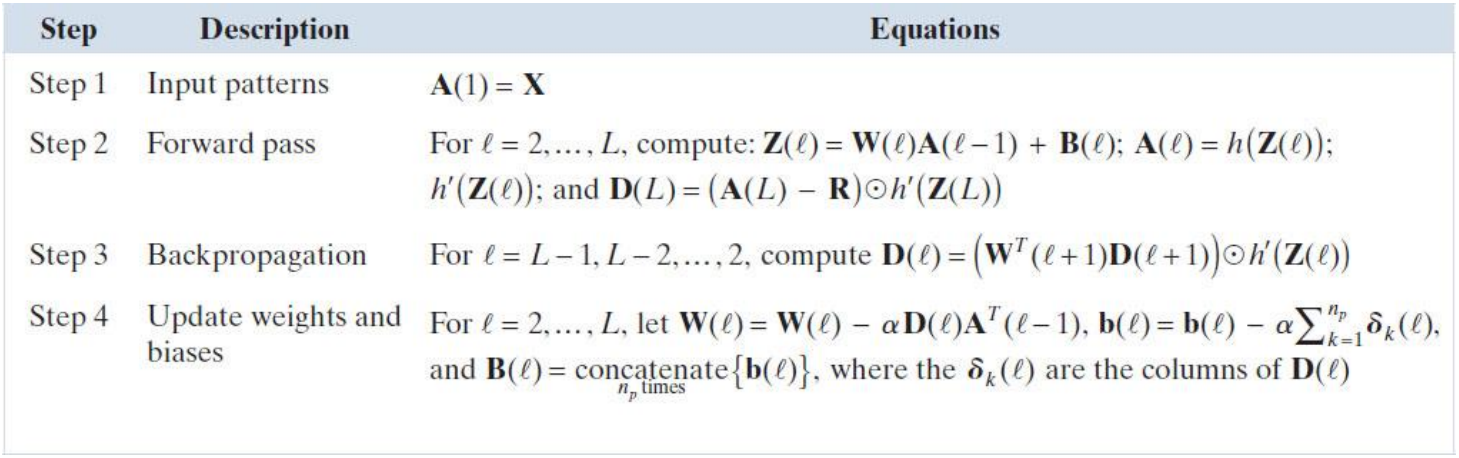
Table 13.2

Steps in the matrix computation of a forward pass through a fully connected, feedforward multilayer neural net.

Step	Description	Equations
Step 1	Input patterns	$\mathbf{A}(1) = \mathbf{X}$
Step 2	Feedforward	For $\ell = 2, \ldots, L$, compute $\mathbf{Z}(\ell) = \mathbf{W}(\ell)\mathbf{A}(\ell - 1) + \mathbf{B}(\ell)$ and $\mathbf{A}(\ell) = h(\mathbf{Z}(\ell))$
Step 3	Output	$\mathbf{A}(L) = h(\mathbf{Z}(L))$

Table 13.3

Matrix formulation for training a feedforward, fully connected multilayer neural network using backpropagation. Steps 1–4 are for one epoch of training. **X**, **R**, and the learning rate parameter are provided to the network for training. The network is α, initialized by specifying weights, **W**(1), and biases, **B**(1), as small random numbers.



Step	Description	Equations
Step 1	Input patterns	$\mathbf{A}(1) = \mathbf{X}$
Step 2	Forward pass	For $\ell = 2, \ldots, L$, compute: $\mathbf{Z}(\ell) = \mathbf{W}(\ell)\mathbf{A}(\ell - 1) + \mathbf{B}(\ell)$; $\mathbf{A}(\ell) = h(\mathbf{Z}(\ell))$; $h'(\mathbf{Z}(\ell))$; and $\mathbf{D}(L) = (\mathbf{A}(L) - \mathbf{R}) \odot h'(\mathbf{Z}(L))$
Step 3	Backpropagation	For $\ell = L-1, L-2, \ldots, 2$, compute $\mathbf{D}(\ell) = \left(\mathbf{W}^T(\ell+1)\mathbf{D}(\ell+1)\right) \odot h'(\mathbf{Z}(\ell))$
Step 4	Update weights and biases	For $\ell = 2, \ldots, L$, let $\mathbf{W}(\ell) = \mathbf{W}(\ell) - \alpha \mathbf{D}(\ell)\mathbf{A}^T(\ell - 1)$, $\mathbf{b}(\ell) = \mathbf{b}(\ell) - \alpha \sum_{k=1}^{n_p} \boldsymbol{\delta}_k(\ell)$, and $\mathbf{B}(\ell) = \underset{n_p \text{ times}}{\text{concatenate}}\{\mathbf{b}(\ell)\}$, where the $\boldsymbol{\delta}_k(\ell)$ are the columns of $\mathbf{D}(\ell)$

Figure 13.34

Neural net solution to the XOR problem. (a) Four patterns in an XOR arrangement. (b) Results of classifying additional points in the range −1.5 to 1.5 in increments of 0.1. All solid points were classified as belonging to class c_1 and all open circles were classified as belonging to class c_2. Together, the two lines separating the regions constitute the decision boundary [compare with Fig. 13.27(b)]. (c) Decision surface, shown as a mesh. The decision boundary is the pair of dashed, white lines in the intersection of the surface and a plane perpendicular to the vertical axis, intersecting that axis at 0.5. (Figure (c) is shown in a different perspective than (b) in order to make all four patterns visible.)

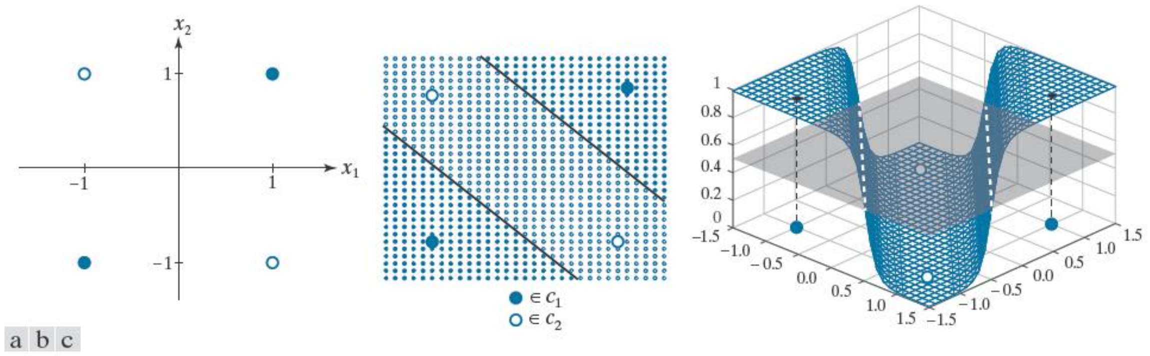


Figure 13.35

Neural net used to solve the XOR problem, showing the weights and biases learned via training using the equations in Table 13.3.

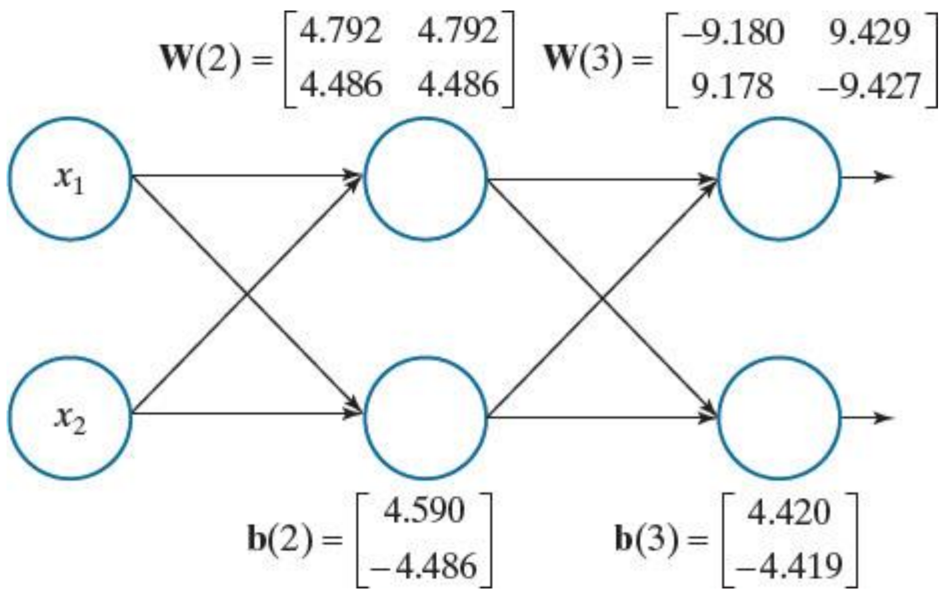

$\mathbf{W}(2) = \begin{bmatrix} 4.792 & 4.792 \\ 4.486 & 4.486 \end{bmatrix}$
$\mathbf{W}(3) = \begin{bmatrix} -9.180 & 9.429 \\ 9.178 & -9.427 \end{bmatrix}$
x_1
x_2
$\mathbf{b}(2) = \begin{bmatrix} 4.590 \\ -4.486 \end{bmatrix}$
$\mathbf{b}(3) = \begin{bmatrix} 4.420 \\ -4.419 \end{bmatrix}$

Figure 13.36

MSE as a function of training epochs for the XOR pattern arrangement.

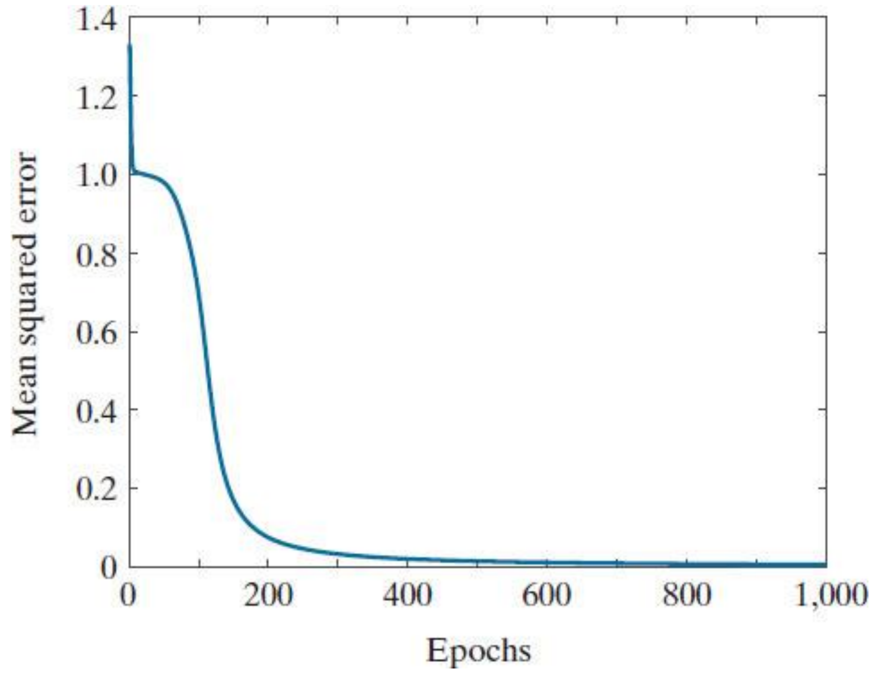
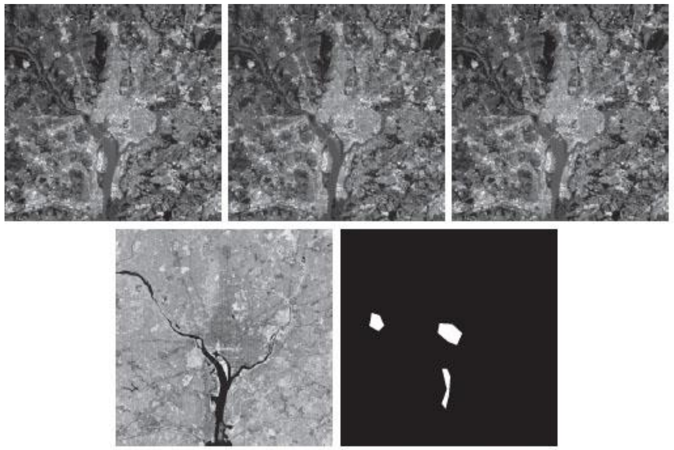
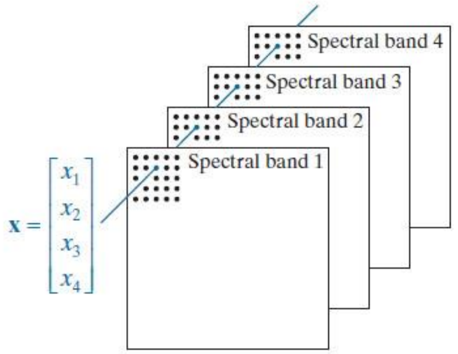


Figure 13.37

(a) Starting with the leftmost image: blue, green, red, near infrared, and binary mask images. In the mask, the lower region is for water, the center region is for the urban area, and the left mask corresponds to vegetation. All images are of size 512×512 pixels. (b) Approach used for generating 4-D pattern vectors from a stack of the four multispectral images. (Multispectral images courtesy of NASA.)



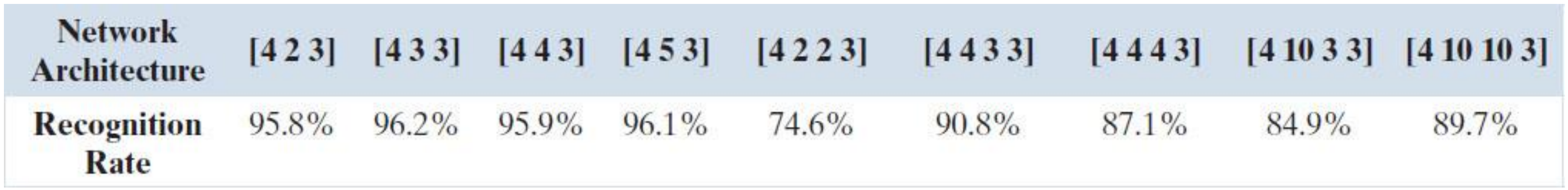
(a) Images in spectral bands 1–4 and binary mask used to extract training samples


Spectral band 4
Spectral band 3
Spectral band 2
Spectral band 1
$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$

(b) Approach used to extract pattern vectors

Table 13.4

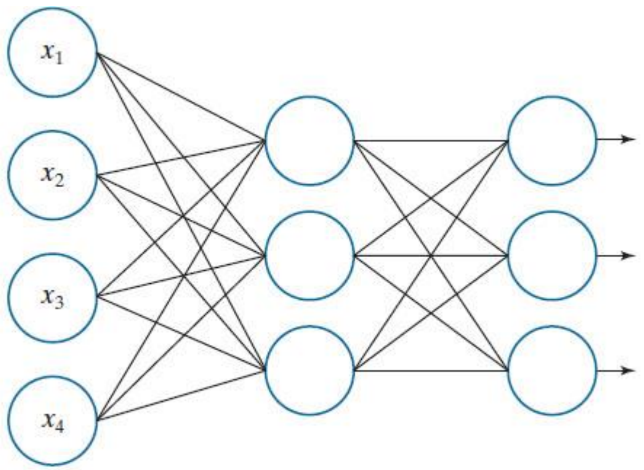
Recognition rate as a function of neural net architecture for $\alpha = 0.001$ and 1,000 training epochs. The network architecture is defined by the numbers in brackets. The first and last number inside each bracket refer to the number of input and output nodes, respectively. The inner entries give the number of nodes in each hidden layer.



Network Architecture	[4 2 3]	[4 3 3]	[4 4 3]	[4 5 3]	[4 2 2 3]	[4 4 3 3]	[4 4 4 3]	[4 10 3 3]	[4 10 10 3]
Recognition Rate	95.8%	96.2%	95.9%	96.1%	74.6%	90.8%	87.1%	84.9%	89.7%

Figure 13.38

Neural net architecture used to classify the multispectral image data in Fig. 13.37 into three classes: water, urban, and vegetation. The parameters shown were obtained in 50,000 epochs of training using $\alpha = 0.001$.

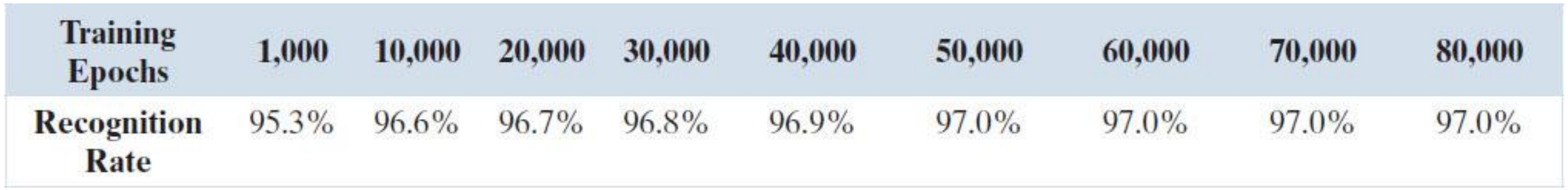


$$\mathbf{W}(2) = \begin{bmatrix} 2.393 & 1.020 & 1.249 & -15.965 \\ 6.599 & -2.705 & -0.912 & 14.928 \\ 8.745 & 0.270 & 3.358 & 1.249 \end{bmatrix} \qquad \mathbf{W}(3) = \begin{bmatrix} 4.093 & -10.563 & -3.245 \\ 7.045 & 9.662 & 6.436 \\ -7.447 & 3.931 & -6.619 \end{bmatrix}$$

$$\mathbf{b}(2) = [4.920 \quad -2.002 \quad -3.485]^T \qquad \mathbf{b}(3) = [3.277 \quad -14.982 \quad 1.582]^T$$

Table 13.5

Recognition performance on the training set as a function of training epochs. The learning rate constant was $\alpha = 0.001$ in all cases.



Training Epochs	1,000	10,000	20,000	30,000	40,000	50,000	60,000	70,000	80,000
Recognition Rate	95.3%	96.6%	96.7%	96.8%	96.9%	97.0%	97.0%	97.0%	97.0%

Figure 13.39

MSE for the network architecture in Fig. 13.38 as a function of the number of training epochs. The learning rate parameter was $\alpha = 0.001$ in all cases.

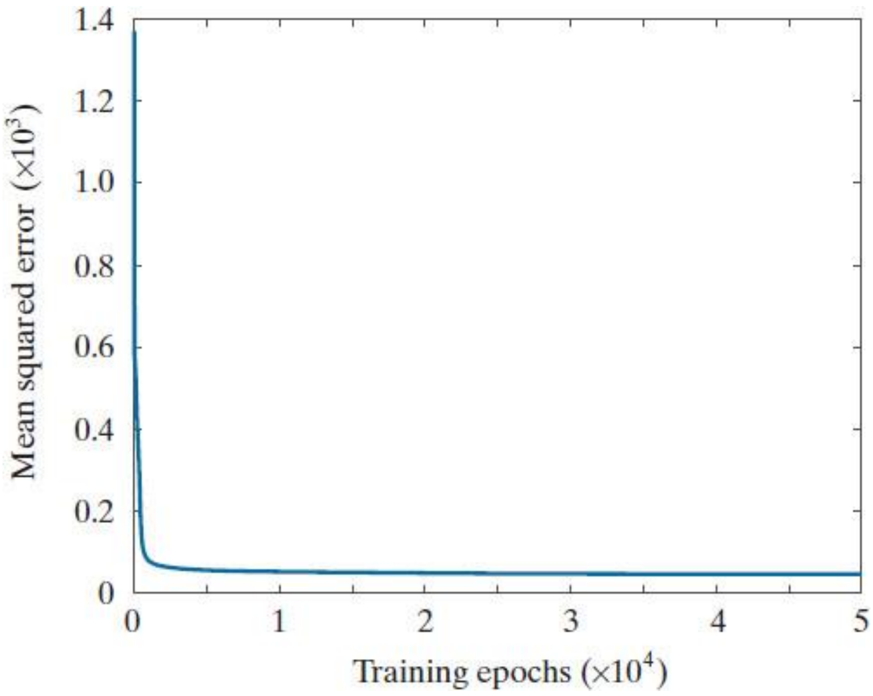


Figure 13.40

A CNN containing all the basic elements of a LeNet architecture. Points **A** and **B** are specific values to be addressed later in this section. The last pooled feature maps are vectorized and serve as the input to a fully connected neural network. The class to which the input image belongs is determined by the output neuron with the highest value.

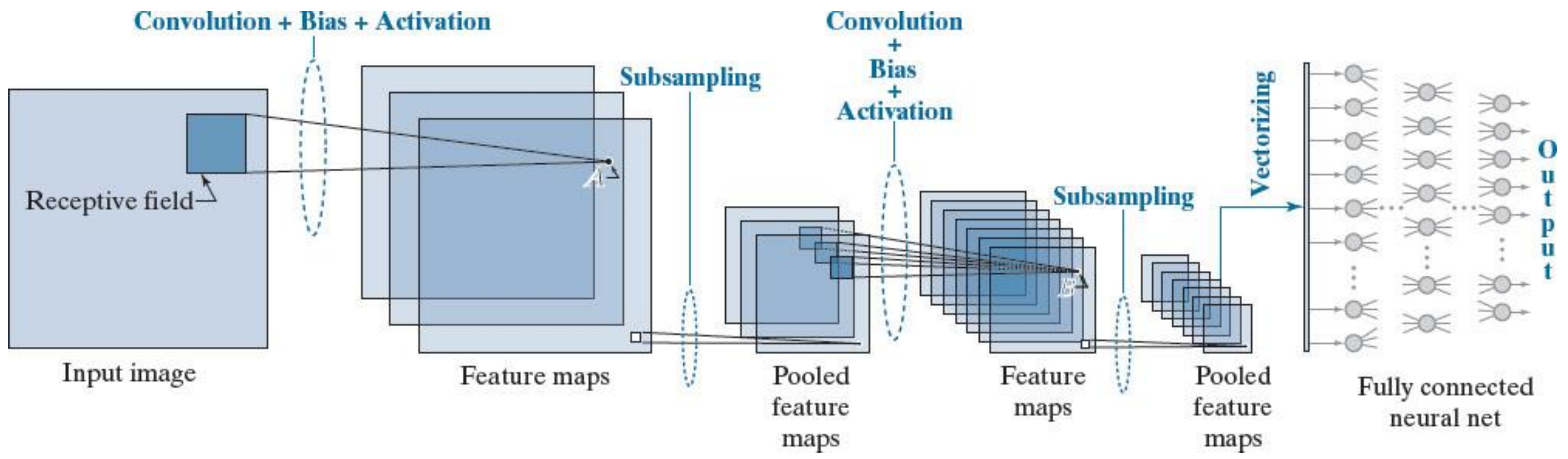
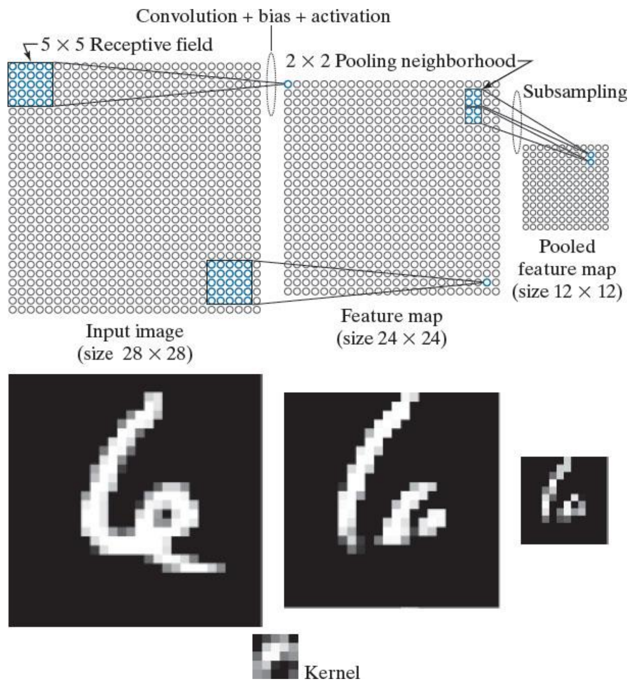


Figure 13.41

Top row: How the sizes of receptive fields and pooling neighborhoods affect the sizes of feature maps and pooled feature maps. Bottom row: An image example. This figure is explained in more detail in Example 13.17. (Image courtesy of NIST.)



Convolution + bias + activation
5 × 5 Receptive field
2 × 2 Pooling neighborhood
Subsampling
Pooled feature map (size 12 × 12)
Feature map (size 24 × 24)
Input image (size 28 × 28)
Kernel

Figure 13.42

Numerical example illustrating the various functions of a CNN, including recognition of an input image. A sigmoid activation function was used throughout.

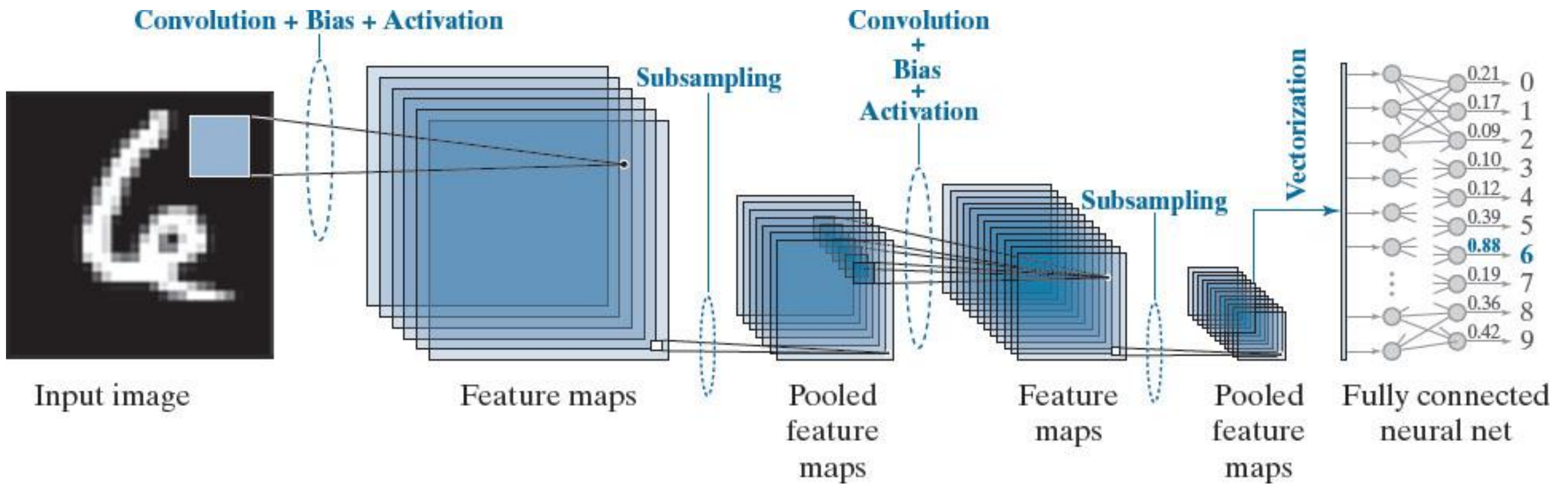


Figure 13.43

Top: The weights (shown as images of size 5×5) corresponding to the six feature maps in the first layer of the CNN in Fig. 13.42. Bottom: The weights corresponding to the twelve feature maps in the second layer.

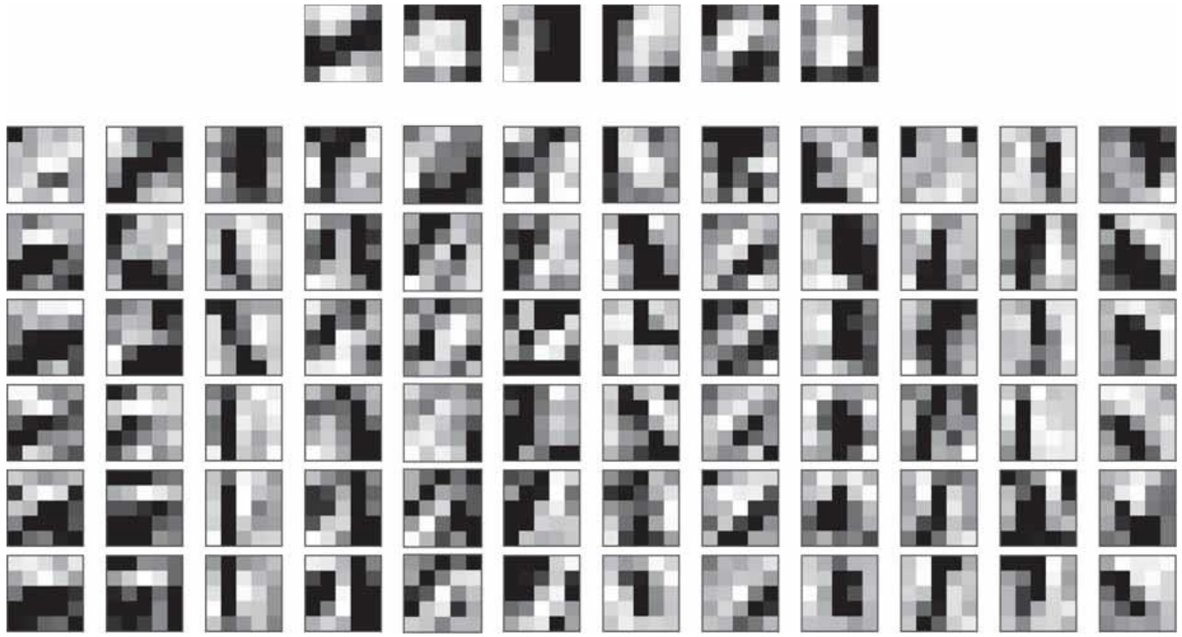


Figure 13.44

Visual summary of an input image propagating through the CNN in Fig. 13.42. Shown as images are all the results of convolution (feature maps) and pooling (pooled feature maps) for both layers of the network. (Example 13.17 contains more details about this figure.)

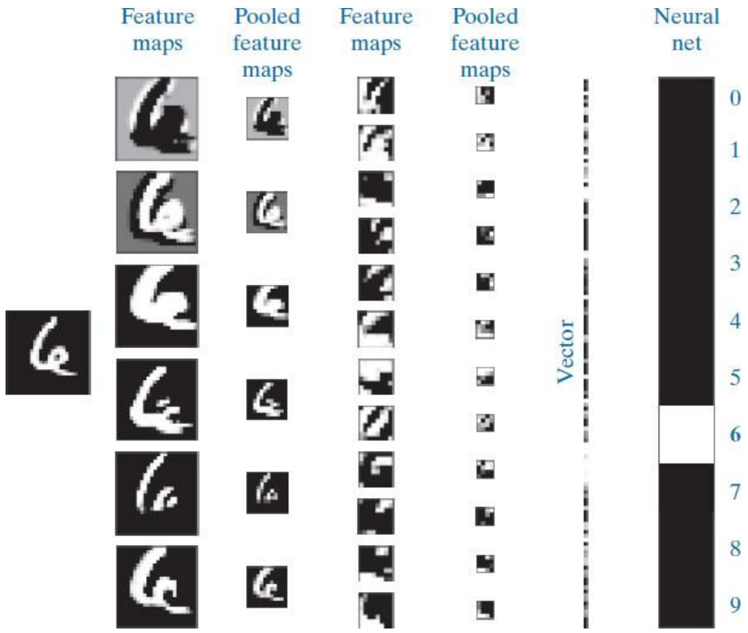
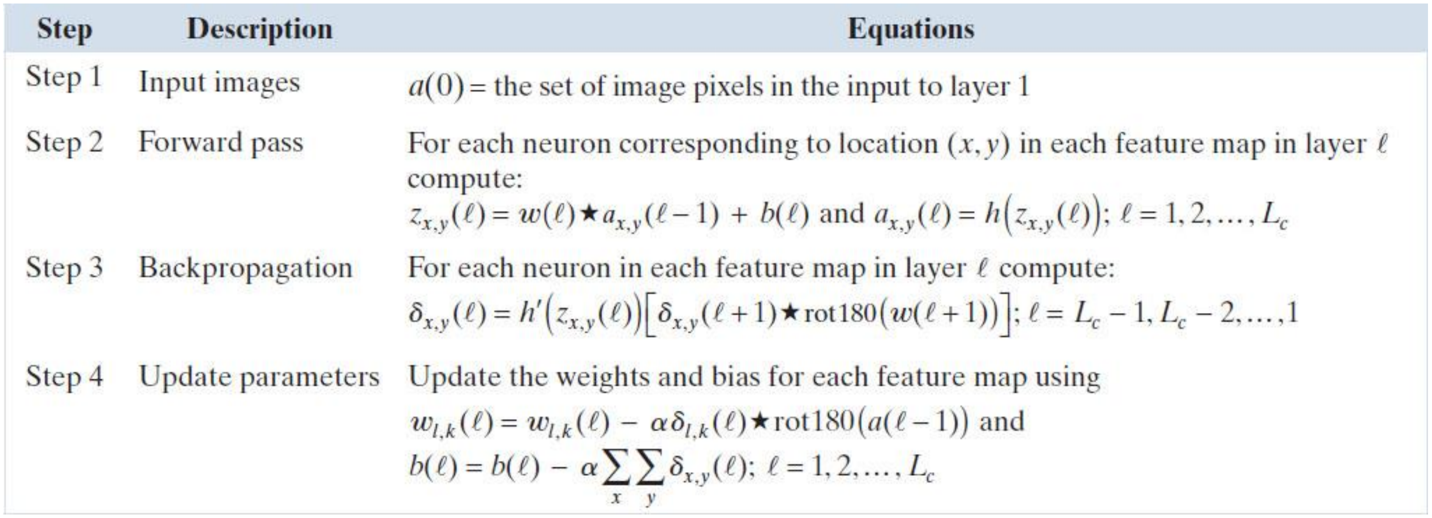


Table 13.6

The principal steps used to train a CNN. The network is initialized with a set of small random weights and biases. In backpropagation, a vector arriving (from the fully connected net) at the output pooling layer must be converted to 2-D arrays of the same size as the pooled feature maps in that layer. Each pooled feature map is upsampled to match the size of its corresponding feature map. The steps in the table are for one epoch of training.



Step	Description	Equations
Step 1	Input images	$a(0) =$ the set of image pixels in the input to layer 1
Step 2	Forward pass	For each neuron corresponding to location (x, y) in each feature map in layer ℓ compute: $z_{x,y}(\ell) = w(\ell) \star a_{x,y}(\ell-1) + b(\ell)$ and $a_{x,y}(\ell) = h\left(z_{x,y}(\ell)\right)$; $\ell = 1, 2, \ldots, L_c$
Step 3	Backpropagation	For each neuron in each feature map in layer ℓ compute: $\delta_{x,y}(\ell) = h'\left(z_{x,y}(\ell)\right)\left[\delta_{x,y}(\ell+1) \star \text{rot180}\left(w(\ell+1)\right)\right]$; $\ell = L_c - 1, L_c - 2, \ldots, 1$
Step 4	Update parameters	Update the weights and bias for each feature map using $w_{l,k}(\ell) = w_{l,k}(\ell) - \alpha\delta_{l,k}(\ell) \star \text{rot180}\left(a(\ell-1)\right)$ and $b(\ell) = b(\ell) - \alpha \sum_x \sum_y \delta_{x,y}(\ell)$; $\ell = 1, 2, \ldots, L_c$

Figure 13.45

CNN with one convolutional layer used to learn to recognize the images in Fig. 13.46.

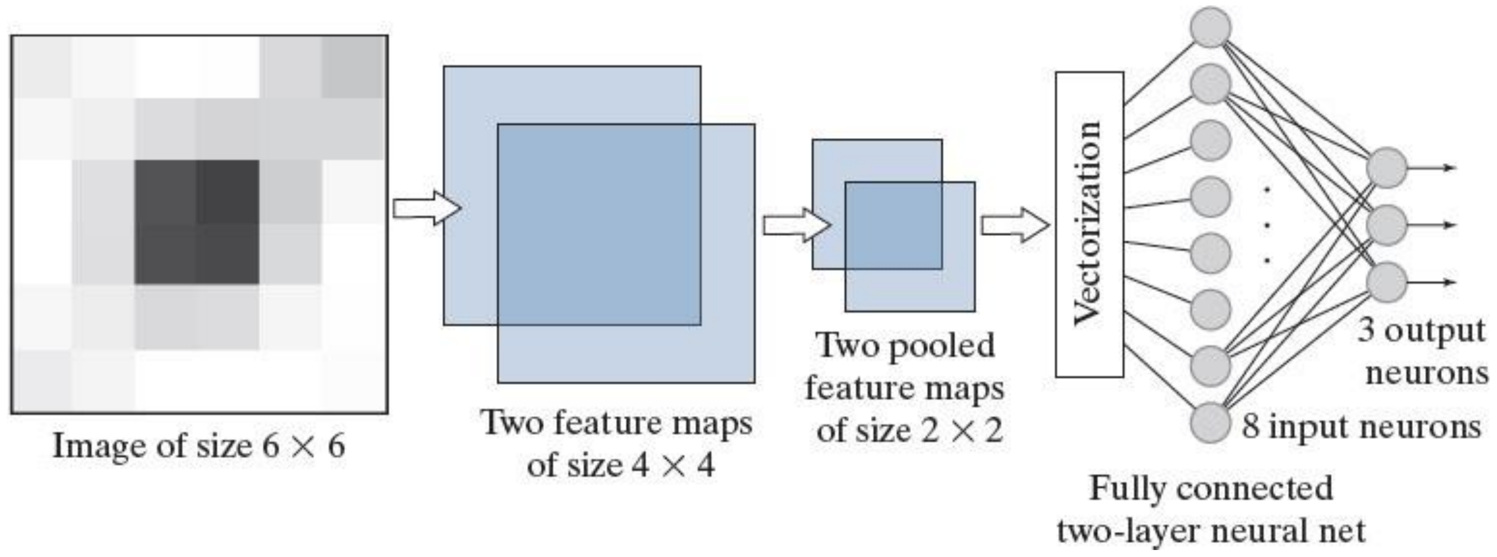


Image of size 6 × 6
Two feature maps of size 4 × 4
Two pooled feature maps of size 2 × 2
Vectorization
3 output neurons
8 input neurons
Fully connected two-layer neural net

Figure 13.46

Left: Training images. Top row: Samples of a dark horizontal stripe. Center row: Samples of a centered dark square. Bottom row: Samples of a dark vertical stripe. Right: Noisy samples of the three categories on the left, created by adding Gaussian noise of zero mean and unit variance to the samples on the left. (All images are 8-bit grayscale images.)

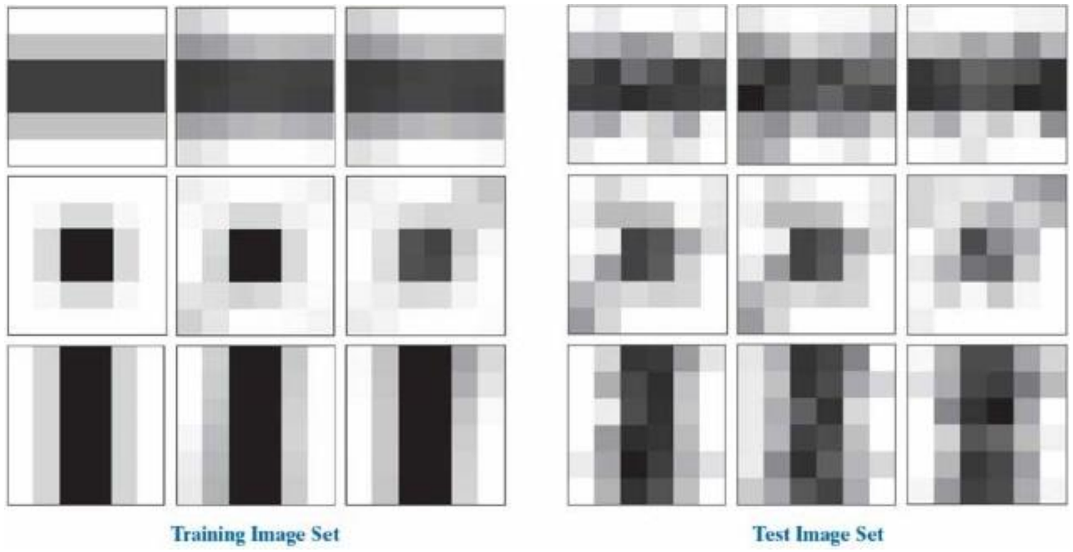


Figure 13.47

Training MSE as a function of epoch for the images in Fig. 13.46. Perfect recognition of the training and test sets was achieved after approximately 100 epochs, despite the fact that the MSE was relatively high there.

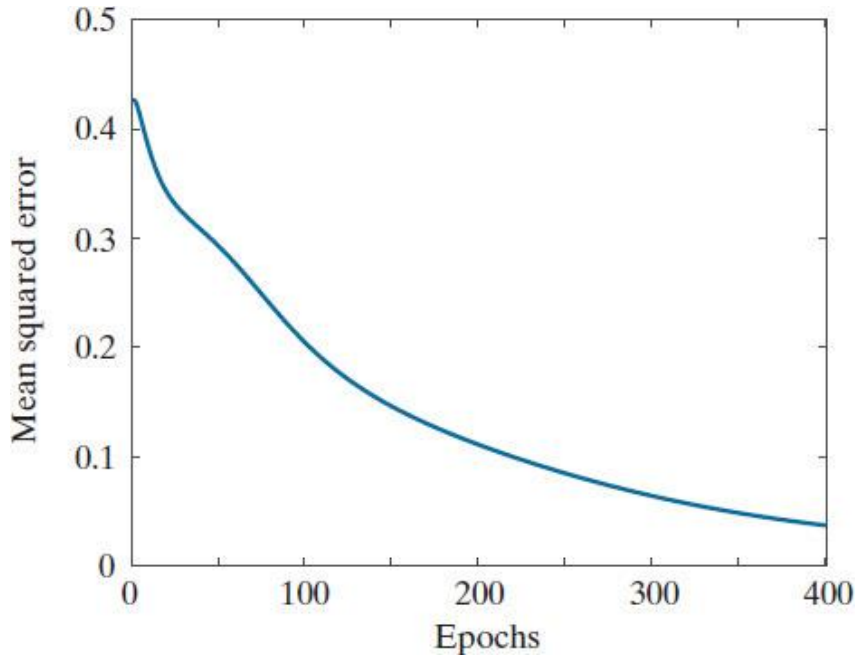


Figure 13.48

Samples similar to those available in the NIST and MNIST databases. Each character subimage is of size 28×28 pixels.(Individual images courtesy of NIST.)

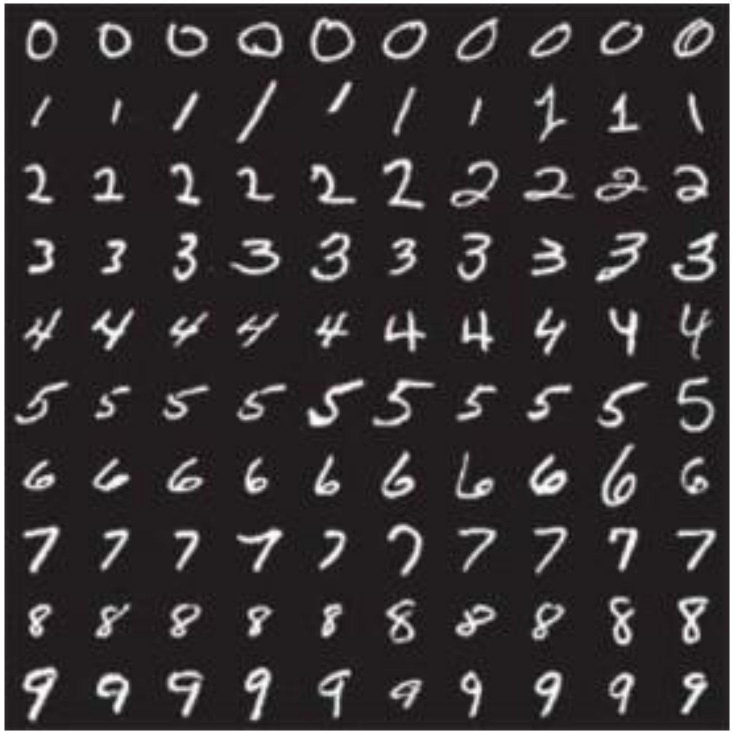


Figure 13.49

CNN used to recognize the ten digits in the MNIST database. The system was trained with 60,000 numerical character images of the same size as the image shown on the left. This architecture is the same as the architecture we used in Fig. 13.42. (Image courtesy of NIST.)

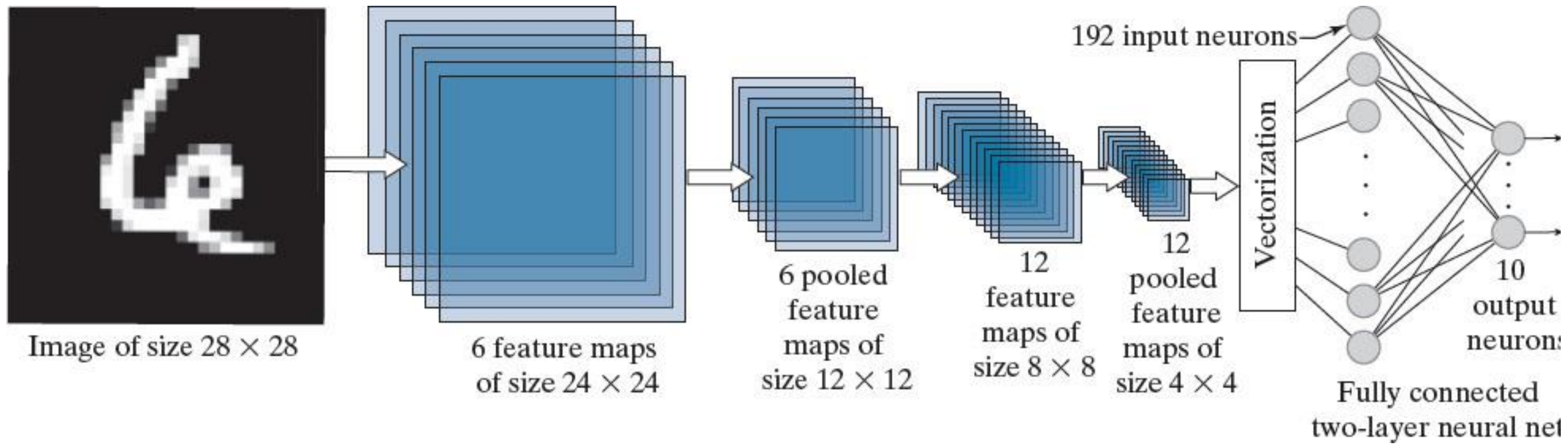


Image of size 28×28
6 feature maps of size 24×24
6 pooled feature maps of size 12×12
12 feature maps of size 8×8
12 pooled feature maps of size 4×4
Vectorization
192 input neurons
10 output neurons
Fully connected two-layer neural net

Figure 13.50

Training mean squared error as a function of epoch for the 60,000 training digit images in the MNIST database.

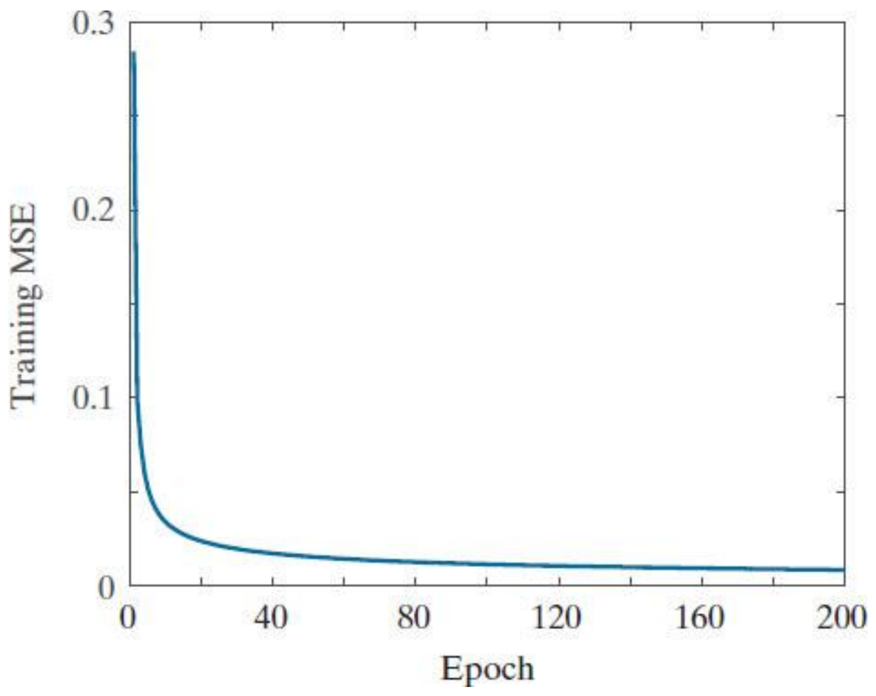


Figure 13.51

(a) Training accuracy (percent correct recognition of the training set) as a function of epoch for the 60,000 training images in the MNIST database. The maximum achieved was 99.36% correct recognition. (b) Accuracy as a function of epoch for the 10,000 test images in the MNIST database. The maximum correct recognition rate was 99.13%.

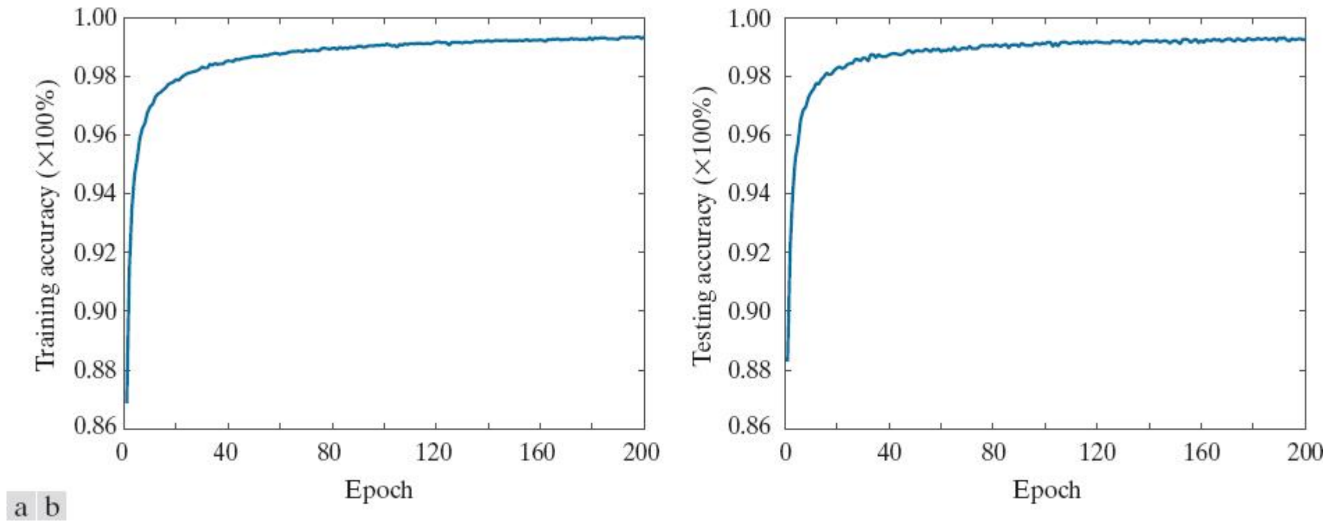


Figure 13.52

(a) Recognition accuracy of training set by image class. Each bar shows a number between 0 and 1. When multiplied by 100%, these numbers give the correct recognition percentage for that class. (b) Recognition results per class in the test set. In both graphs the recognition rate is above 98%.

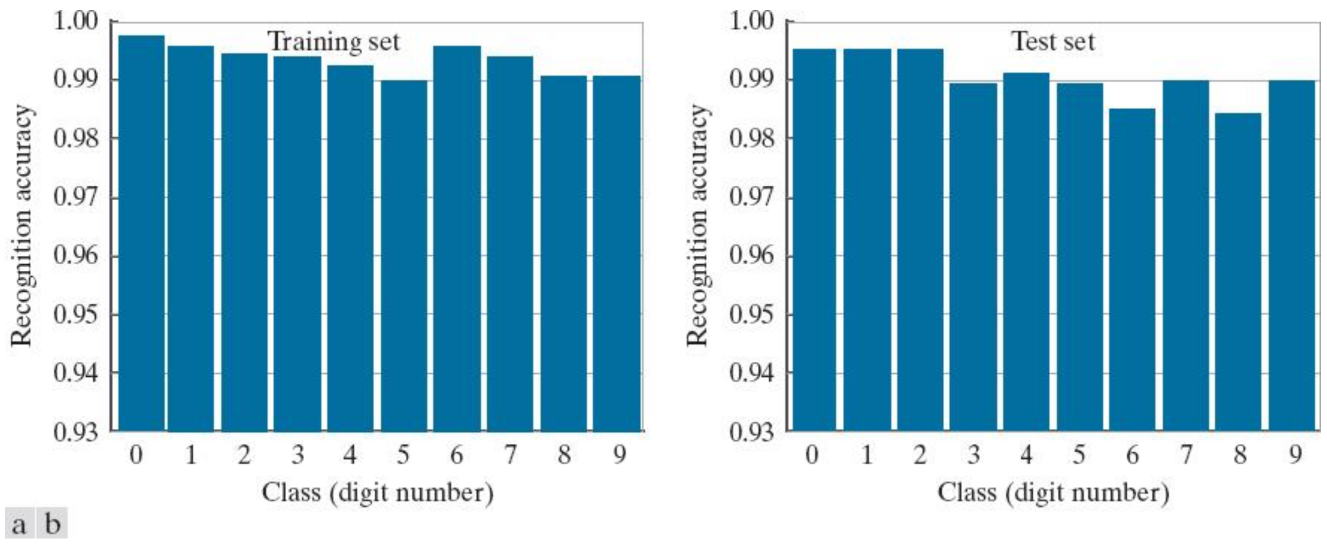


Figure 13.53

Kernels of the first layer after 200 epochs of training, shown as images.

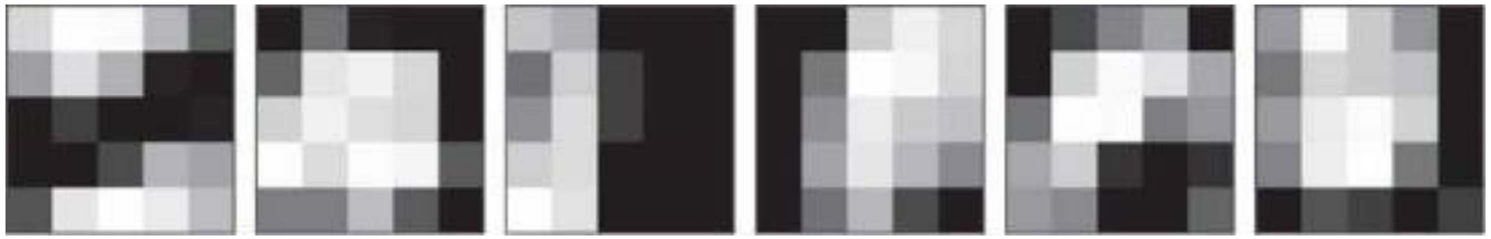


Figure 13.54

Kernels of the second layer after 200 epochs of training, displayed as images of size 5×5. There are six inputs (pooled feature maps) into the second layer. Because there are twelve feature maps in the second layer, the CNN learned the weights of $6 \times 12 = 72$ kernels.

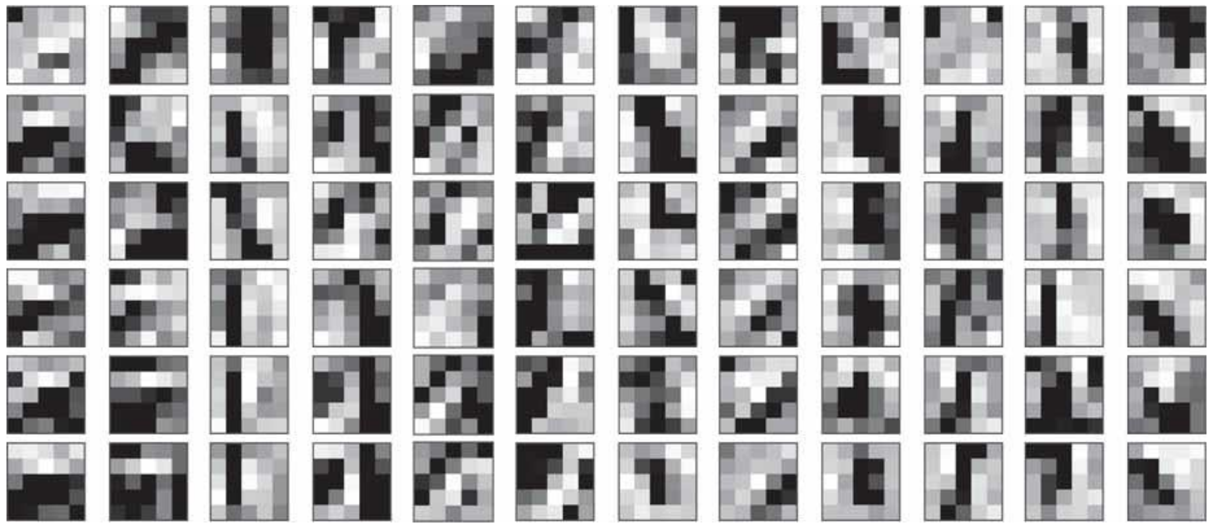


Figure 13.55

Results of a forward pass for one digit image through the CNN in Fig. 13.49 after training. The feature maps were generated using the kernels from Figs. 13.53 and 13.54, followed by pooling. The neural net is the two-layer neural network from Fig. 13.49. The output high value (in white) indicates that the CNN recognized the input properly. (This figure is the same as Fig. 13.44.)

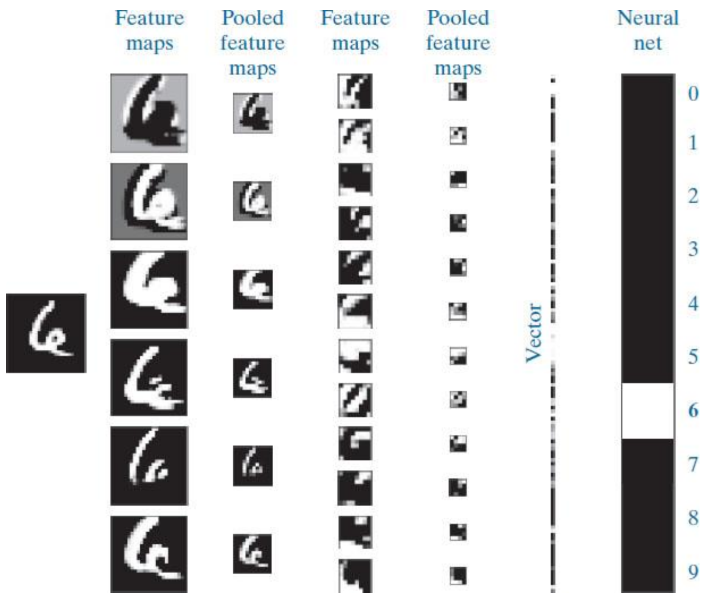


Figure 13.56

Mini images of size 32×32 pixels, representative of the 50,000 training and 10,000 test images in the CIFAR-10 database (the 10 stands for ten classes). The class names are shown on the right. (Images courtesy of Pearson Education.)

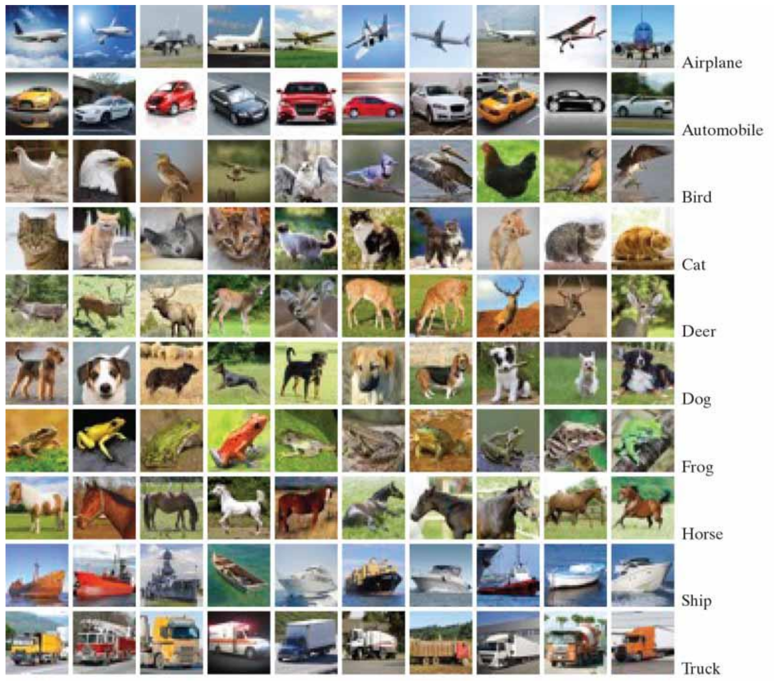


Figure 13.57

Training mean squared error as a function of the number of epochs for a training set of 50,000 CIFAR-10 images.

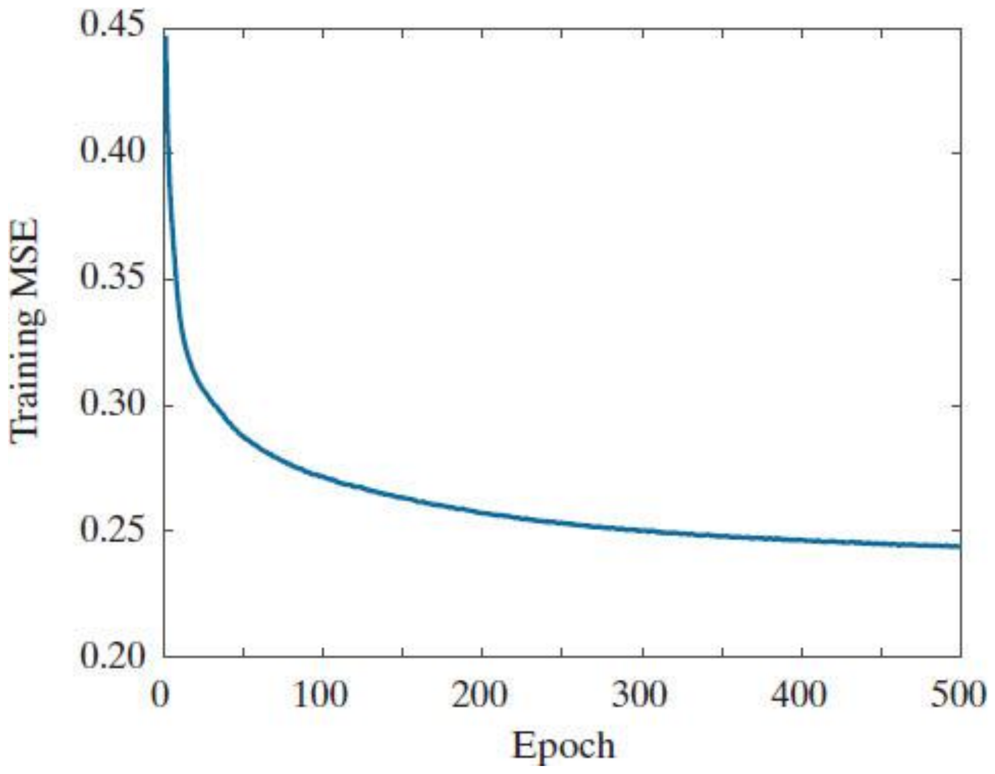


Figure 13.58

(a) Training accuracy (percent correct recognition of the training set) as a function of epoch for the 50,000 training images in the CIFAR-10 database. (b) Accuracy as a function of epoch for the 10,000 CIFAR-10 test images.

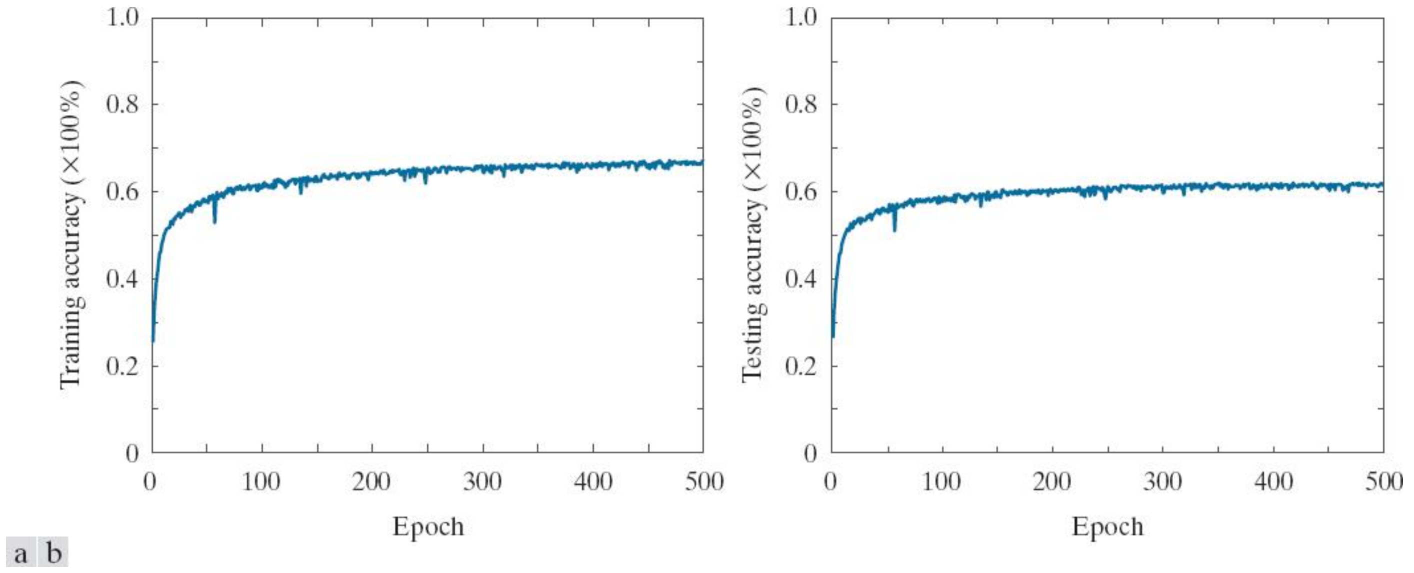


Figure 13.59

(a) CIFAR-10 recognition rate of training set by image class. Each bar shows a number between 0 and 1. When multiplied by 100%, these numbers give the correct recognition percentage for that class. (b) Recognition results per class in the test set.

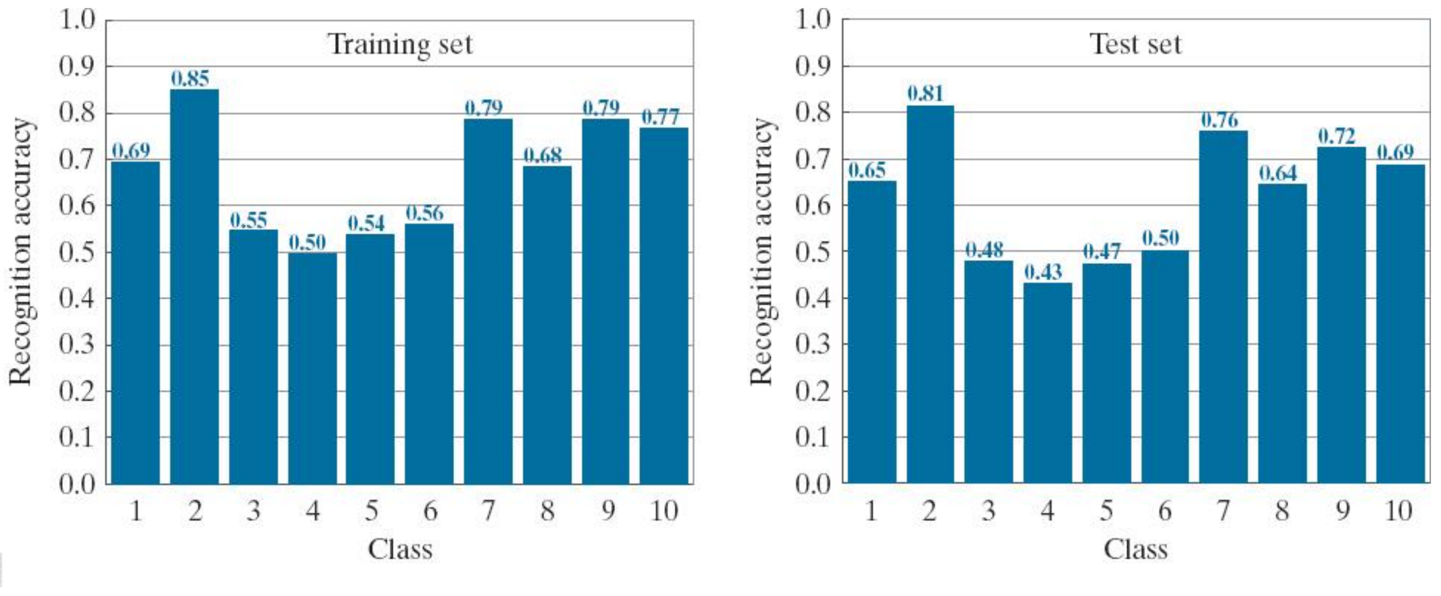


Figure 13.60

Weights of the kernels of the first convolution layer after 500 epochs of training.

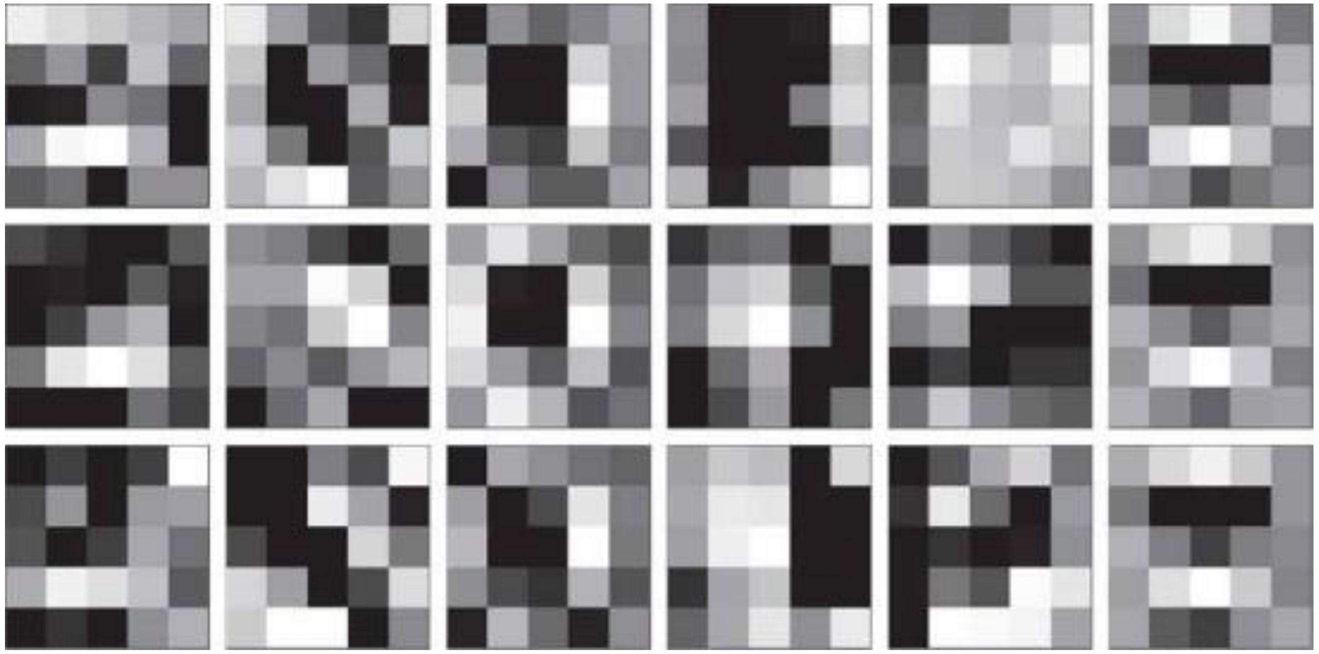


Figure 13.61

Weights of the kernels of the second convolution layer after 500 epochs of training. The interpretation of these kernels is the same as in Fig. 13.54.

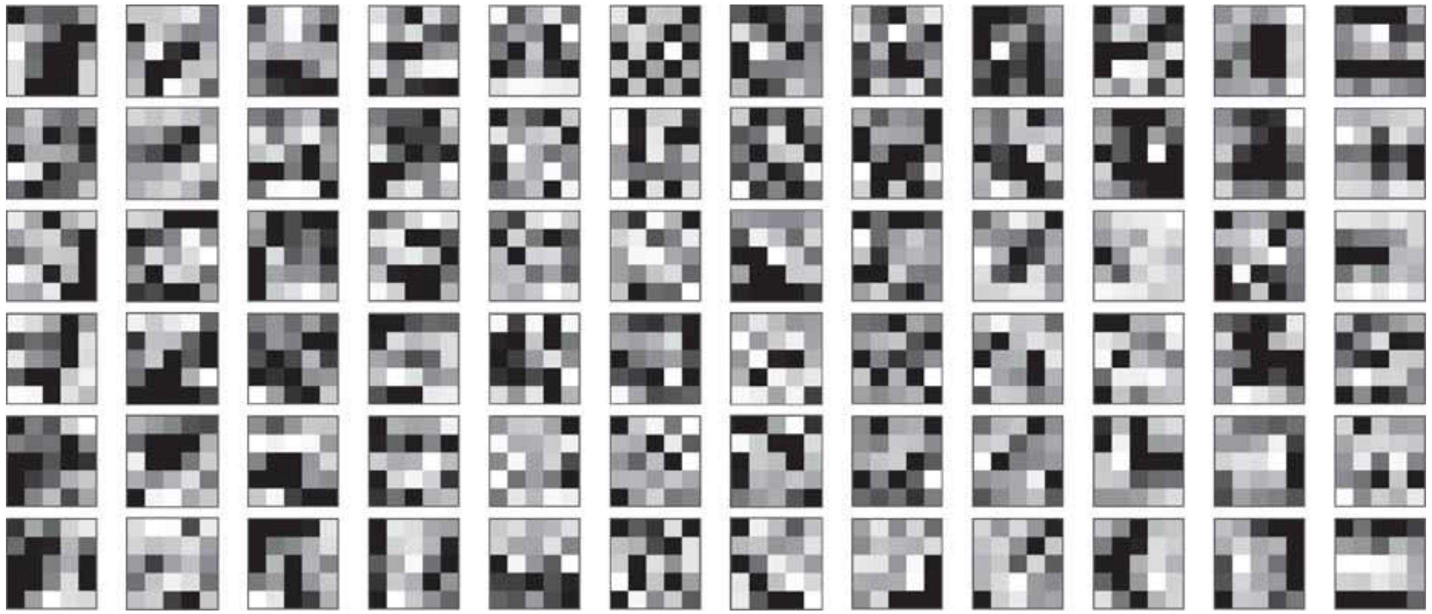
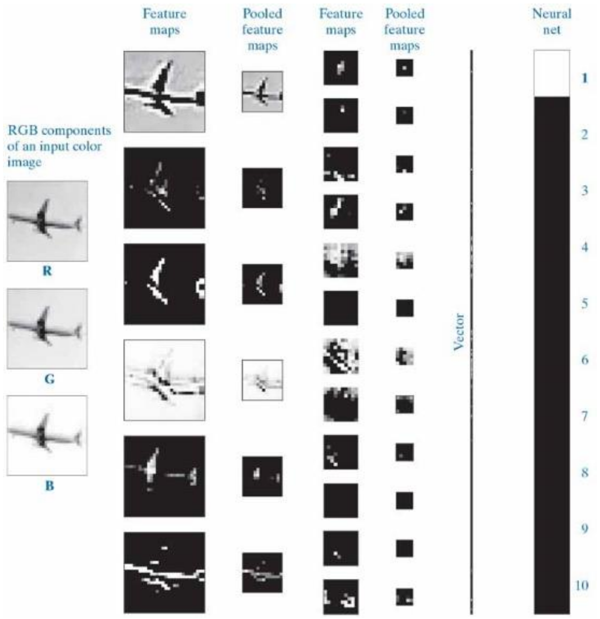
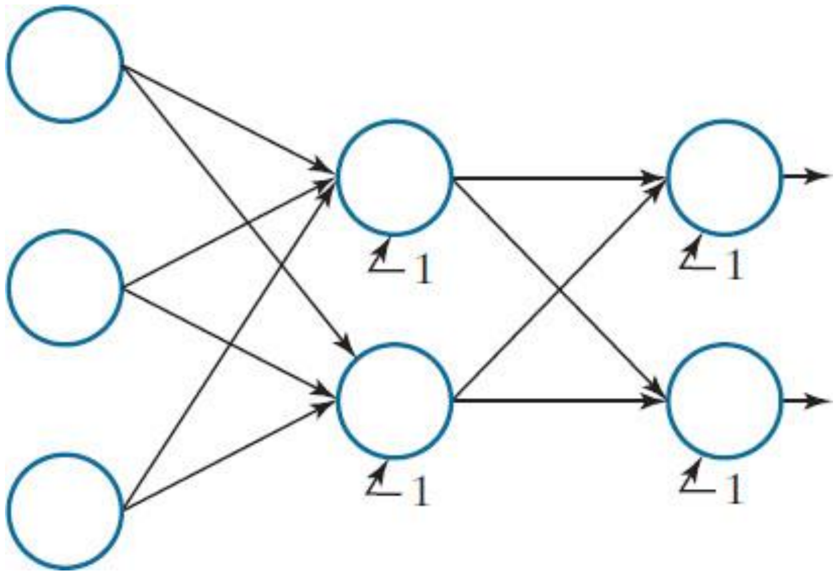


Figure 13.62

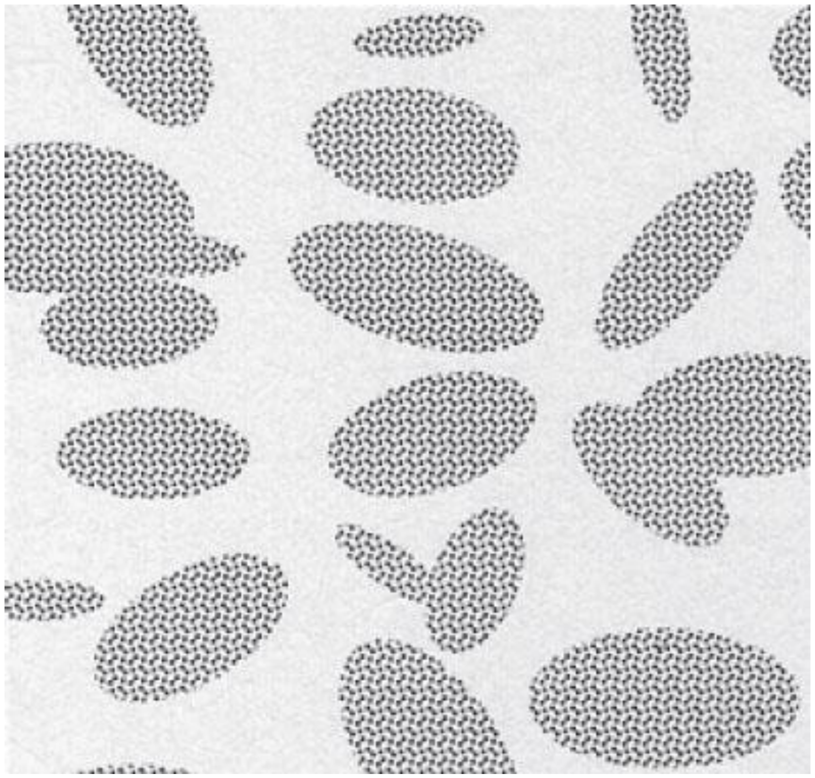
Graphical illustration of a forward pass through the trained CNN. The purpose was to recognize one input image from the set in Fig. 13.56. As the output shows, the image was recognized correctly as belonging to class 1, the class of airplanes. (Original image courtesy of Pearson Education.)



Unnumbered Figure 1



Unnumbered Figure 2



Pearson

Copyright