12 Image Pattern Classification

李东晓

lidx@zju.edu.cn

浙江大学信息与电子工程学院
College of Information Science & Electronic Engineering, Zhejiang University

Contents

Neural Networks

- Training = Learning
- Training set: set of training patterns
- Perceptron (感知器)

e.g. Linear decision function

$$d(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i + w_{n+1}$$

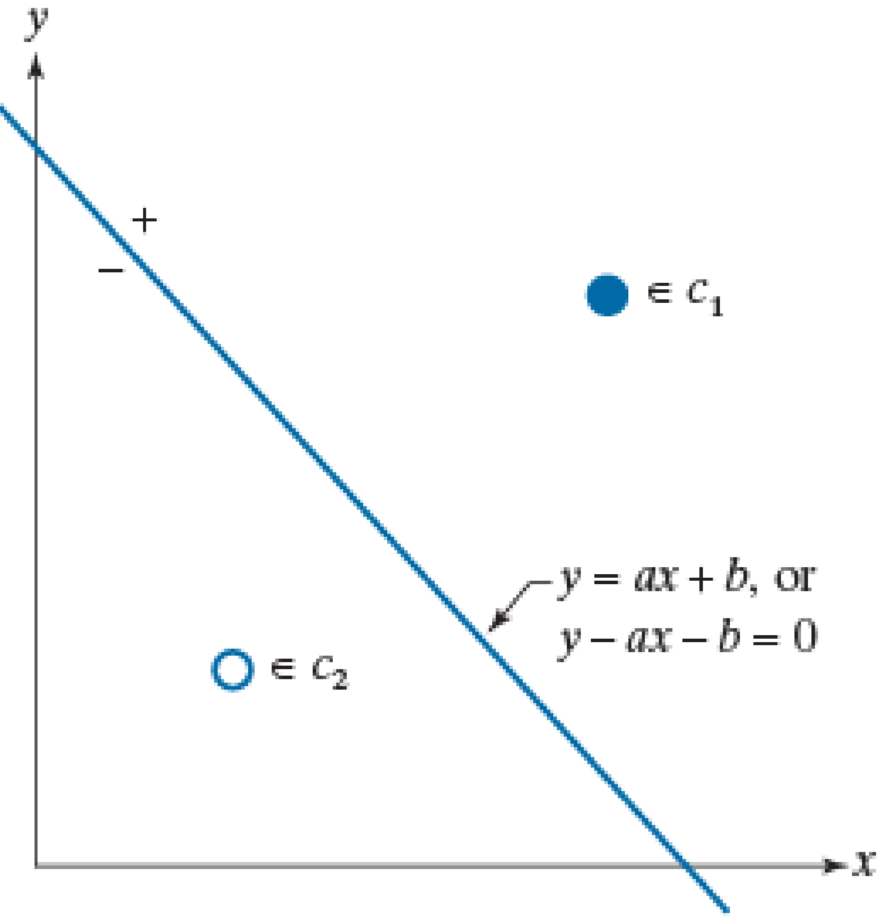
- Augmented patter vector

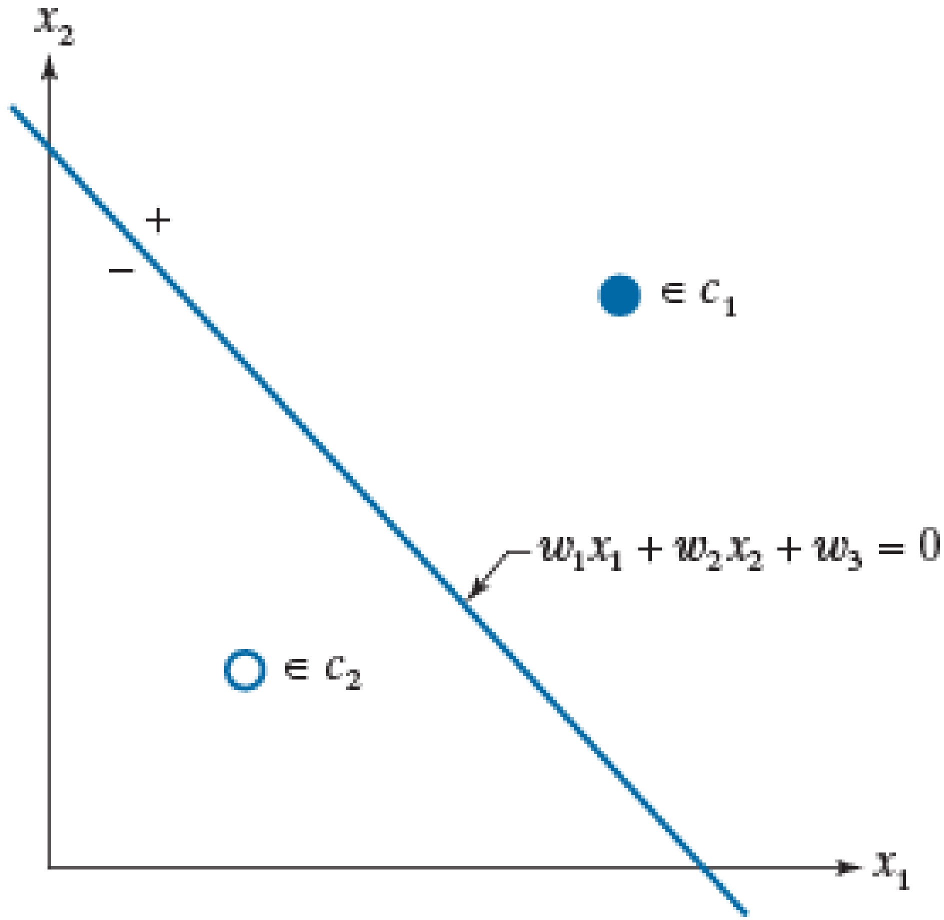
$$y_i = x_i, i = 1, 2, \ldots, n, \; y_{n+1} = 1$$

$$d(\mathbf{y}) = \sum_{i=1}^{n+1} w_i y_i = \mathbf{w}^T \mathbf{y}$$

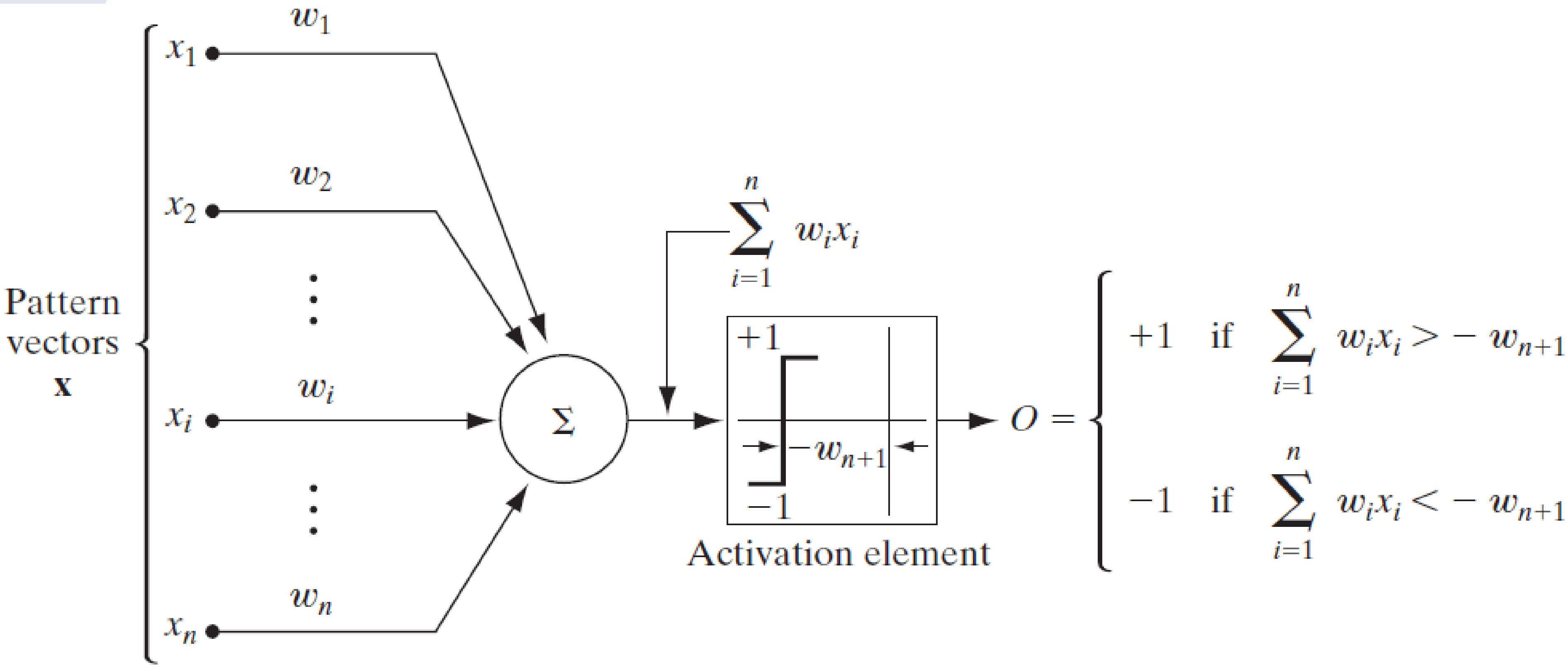
Linear boundary between two classes

$$\boldsymbol{w}^T\mathbf{x} + w_{n+1} = \begin{cases} > 0 & \text{if } \mathbf{x} \in c_1 \\ < 0 & \text{if } \mathbf{x} \in c_2 \end{cases}$$

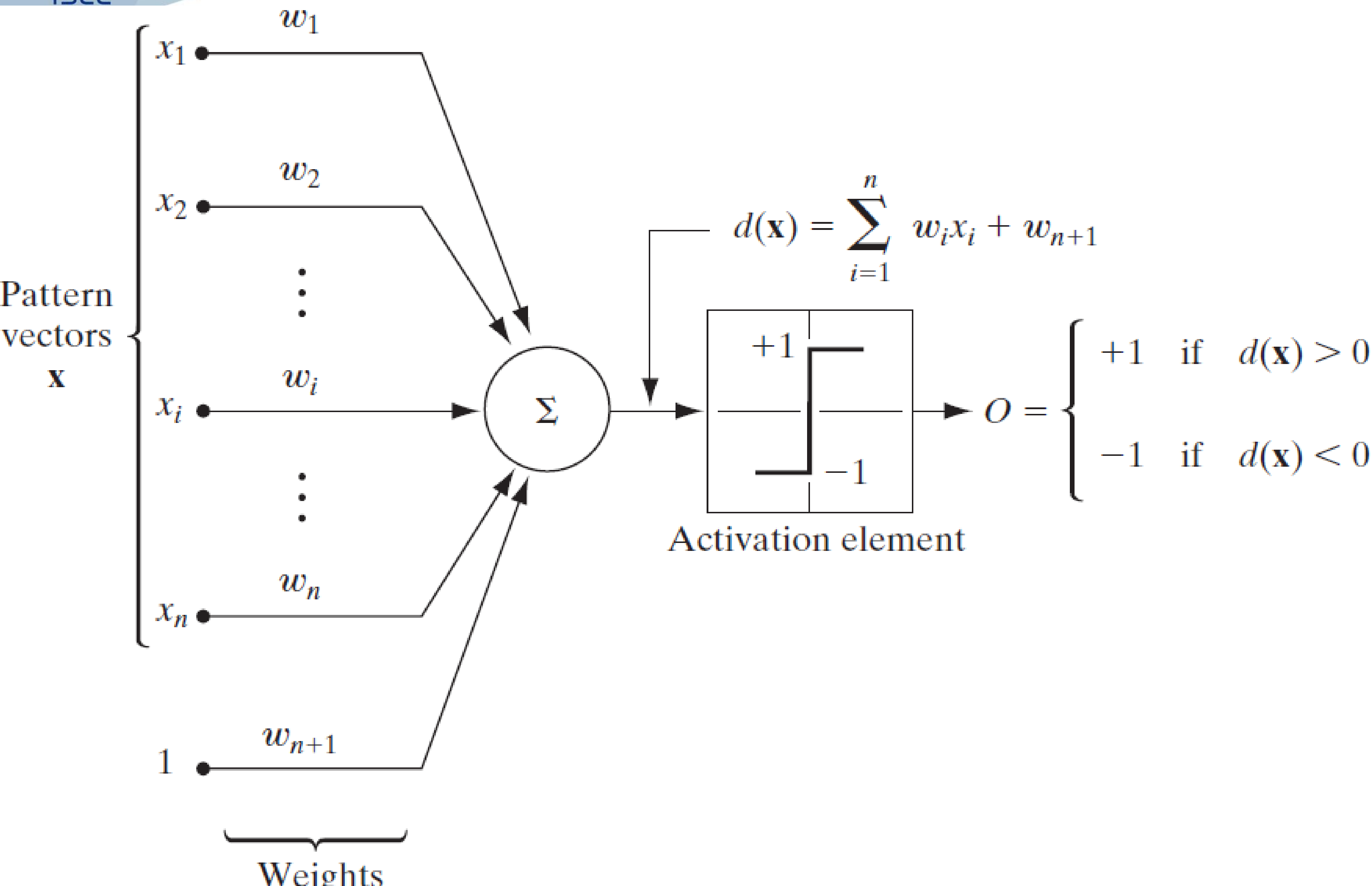




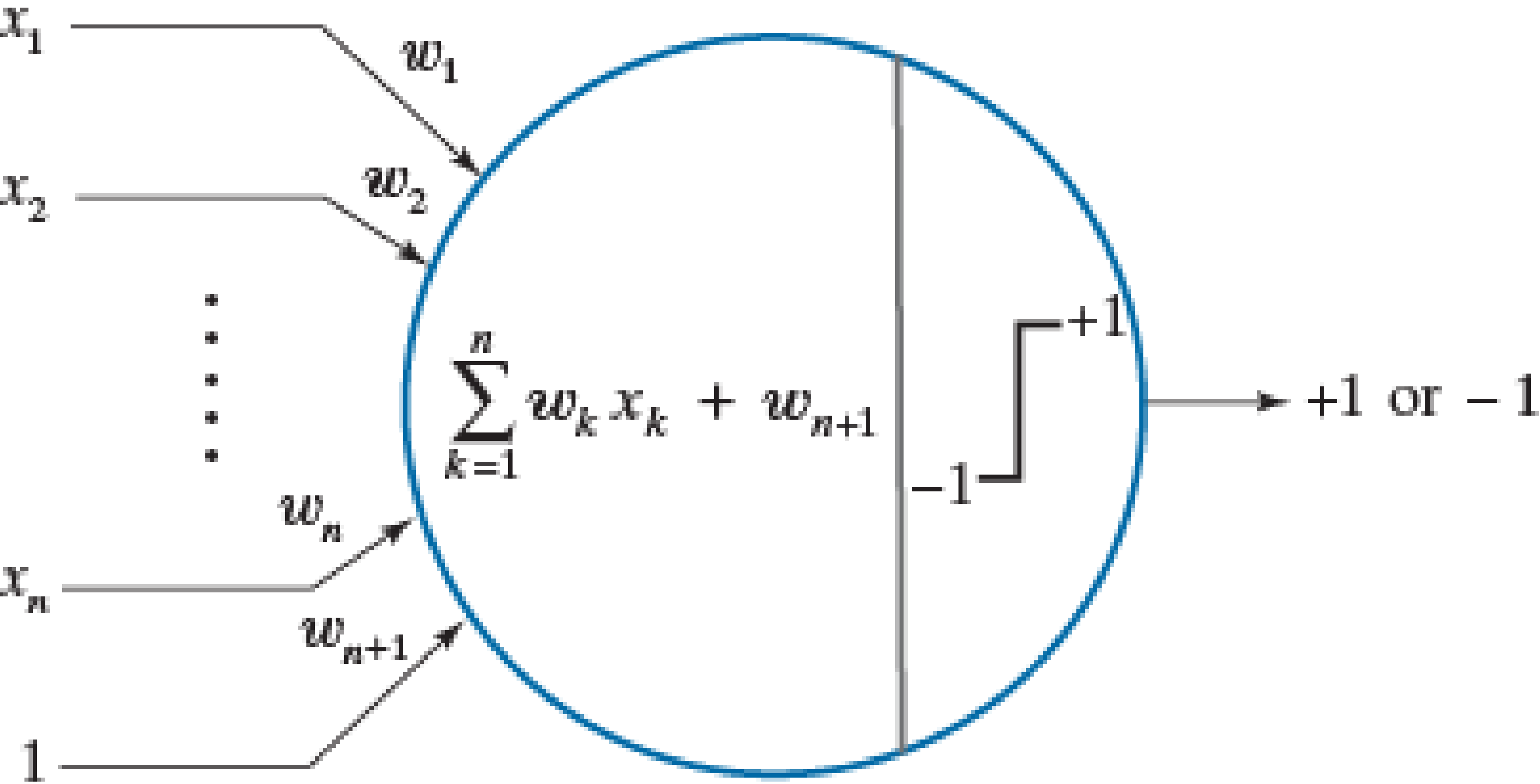
Equivalent perceptron models



Equivalent perceptron models



Equivalent perceptron models



Perceptron training algorithm

Learning increment / rate

1) If $\mathbf{x}(k) \in c_1$ and $\boldsymbol{w}^T(k)\mathbf{x}(k) + w_{n+1}(k) \le 0$, let

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \alpha\mathbf{x}(k)$$

$$\omega_{n+1}(k+1) = \omega_{n+1}(k) + \alpha$$

2) If $\mathbf{x}(k) \in c_2$ and $\boldsymbol{w}^T(k)\mathbf{x}(k) + w_{n+1}(k) \ge 0$, let

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \alpha\mathbf{x}(k)$$

$$\omega_{n+1}(k+1) = \omega_{n+1}(k) - \alpha$$

3) Otherwise, let

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k)$$

$$\omega_{n+1}(k+1) = \omega_{n+1}(k)$$

Perceptron training algorithm

Using augmented pattern

1′) If $\mathbf{x}(k) \in c_1$ and $\boldsymbol{w}^T(k)\mathbf{x}(k) \le 0$, let

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \alpha\mathbf{x}(k)$$

2′) If $\mathbf{x}(k) \in c_2$ and $\boldsymbol{w}^T(k)\mathbf{x}(k) \ge 0$, let

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \alpha\mathbf{x}(k)$$

3′) Otherwise, let

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k)$$

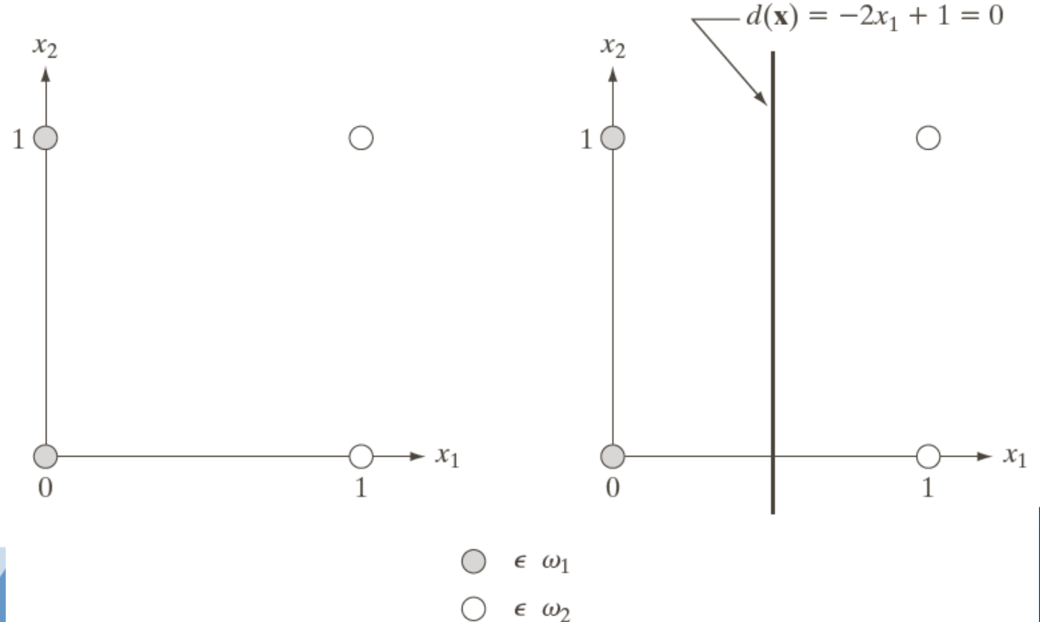
Example for linearly separable classes

$$\mathbf{w}^T(1)\mathbf{y}(1) = [0,0,0]\begin{bmatrix}0\\0\\1\end{bmatrix} = 0 \qquad \mathbf{w}(2) = \mathbf{w}(1) + \mathbf{y}(1) = \begin{bmatrix}0\\0\\1\end{bmatrix}$$

$$\mathbf{w}^T(2)\mathbf{y}(2) = [0,0,1]\begin{bmatrix}0\\1\\1\end{bmatrix} = 1 \qquad \mathbf{w}(3) = \mathbf{w}(2) = \begin{bmatrix}0\\0\\1\end{bmatrix}$$

$$\mathbf{w}^T(3)\mathbf{y}(3) = [0,0,1]\begin{bmatrix}1\\0\\1\end{bmatrix} = 1 \qquad \mathbf{w}(4) = \mathbf{w}(3) - \mathbf{y}(3) = \begin{bmatrix}-1\\0\\0\end{bmatrix}$$

$$\mathbf{w}^T(4)\mathbf{y}(4) = [-1,0,0]\begin{bmatrix}1\\1\\1\end{bmatrix} = -1 \qquad \mathbf{w}(5) = \mathbf{w}(4) = \begin{bmatrix}-1\\0\\0\end{bmatrix}$$

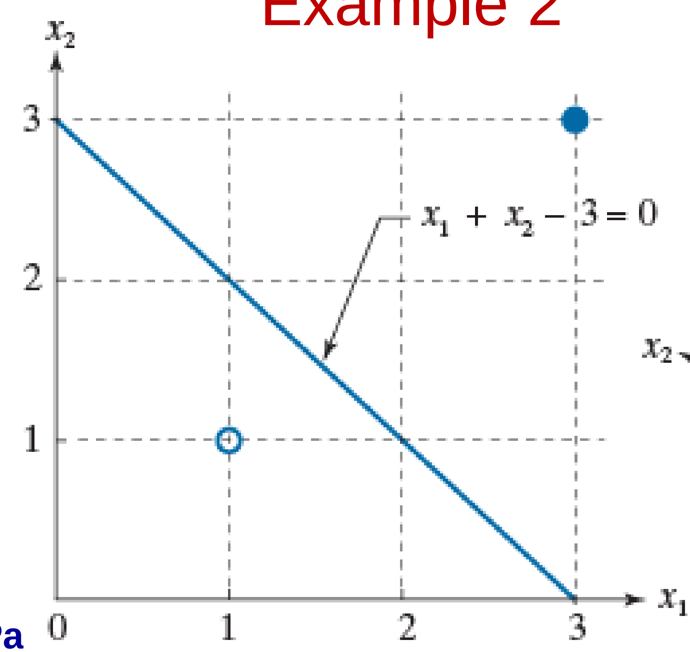


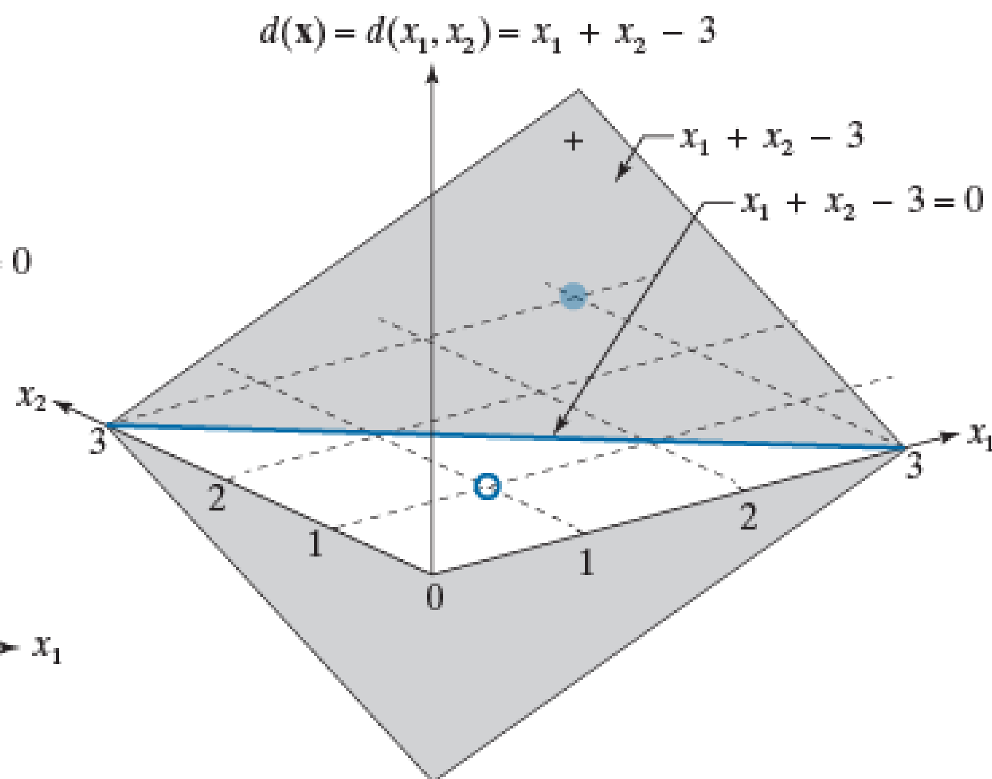
$$w^T(1)\mathbf{x}(1) = \begin{bmatrix}0 & 0 & 0\end{bmatrix}\begin{bmatrix}3\\3\\1\end{bmatrix} = 0 \qquad w(2) = w(1) + \alpha\mathbf{x}(1) = \begin{bmatrix}0\\0\\0\end{bmatrix} + (1)\begin{bmatrix}3\\3\\1\end{bmatrix} = \begin{bmatrix}3\\3\\1\end{bmatrix}$$

$$w^T(2)\mathbf{x}(2) = \begin{bmatrix}3 & 3 & 1\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix} = 7 \qquad w(3) = w(2) - \alpha\mathbf{x}(2) = \begin{bmatrix}3\\3\\1\end{bmatrix} - (1)\begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}2\\2\\0\end{bmatrix}$$

$$w(4) = w(3) = \begin{bmatrix}2\\2\\0\end{bmatrix} \qquad w(5) = w(4) - \alpha\mathbf{x}(4) = \begin{bmatrix}2\\2\\0\end{bmatrix} - (1)\begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}1\\1\\-1\end{bmatrix} \quad \ldots \quad w = w(12) = \begin{bmatrix}1\\1\\-3\end{bmatrix}$$

Example 2





Nonseparable classes

- E.g. criterion function $E(\boldsymbol{w}) = \frac{1}{2}\left(r - \boldsymbol{w}^T \mathbf{x}\right)^2$

r is the desired response

- Gradient descent

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \alpha \left[\frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}}\right]_{\boldsymbol{w}=\boldsymbol{w}(k)} \qquad \alpha > 0$$

$$\frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}} = -\left(r - \boldsymbol{w}^T \mathbf{x}\right)\mathbf{x}$$

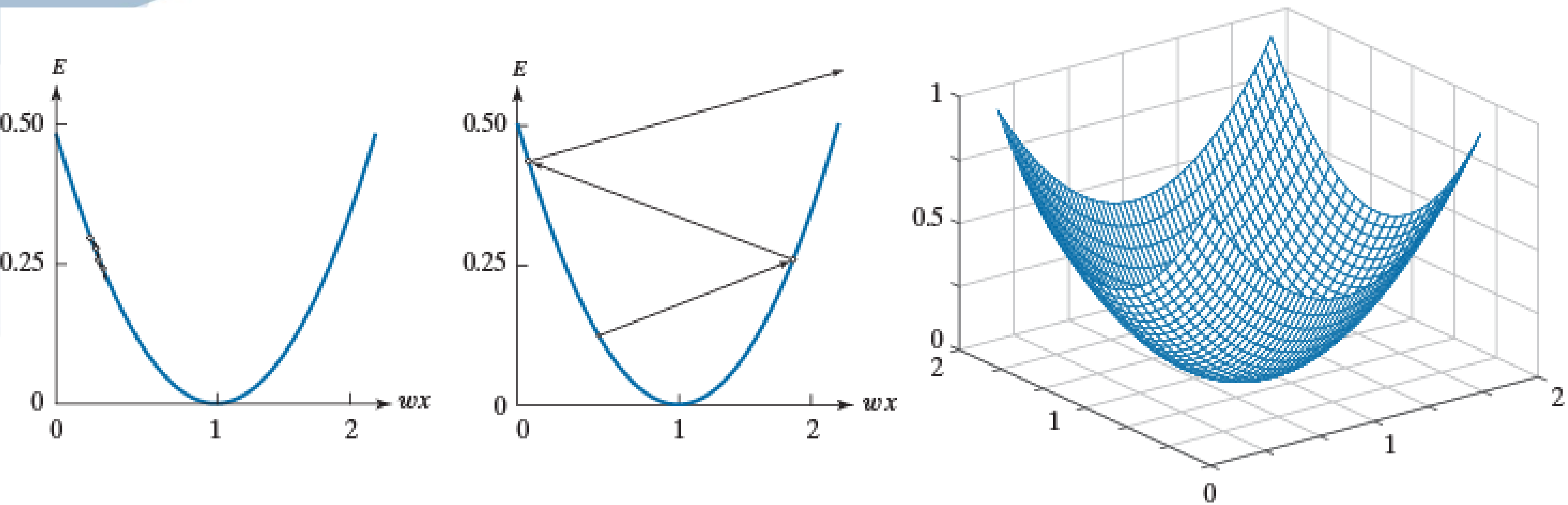
$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \alpha\left[r(k) - \boldsymbol{w}^T(k)\mathbf{x}(k)\right]\mathbf{x}(k)$$

$$\Delta \mathbf{w} = \alpha e(k)\mathbf{x}(k)$$

delta correction algorithm.

typical range for α is $0.1 < \alpha < 1.0$

Error Function



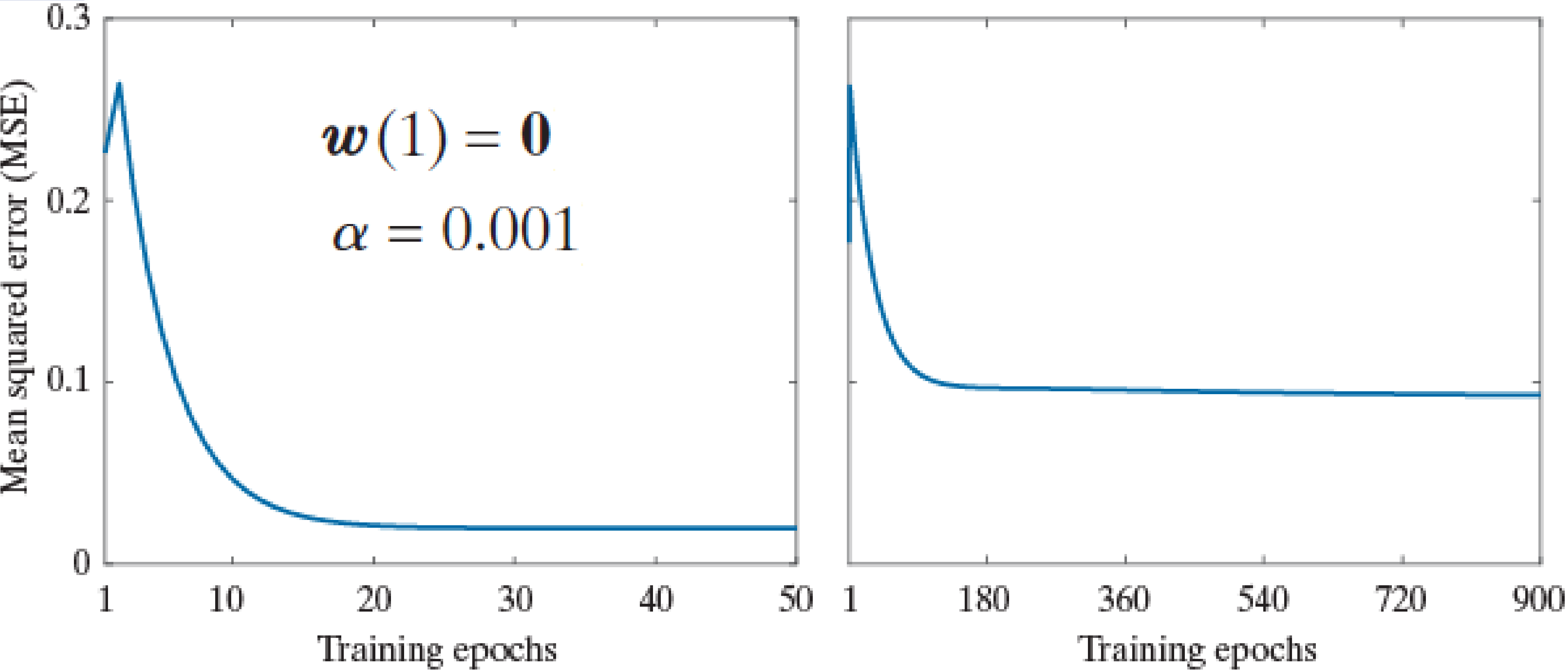
a b c

FIGURE 12.25 Plots of E as a function of wx for $r = 1$. (a) A value of α that is too small can slow down convergence. (b) If α is too large, large oscillations or divergence may occur. (c) Shape of the error function in 2-D.

MSE vs. epoch

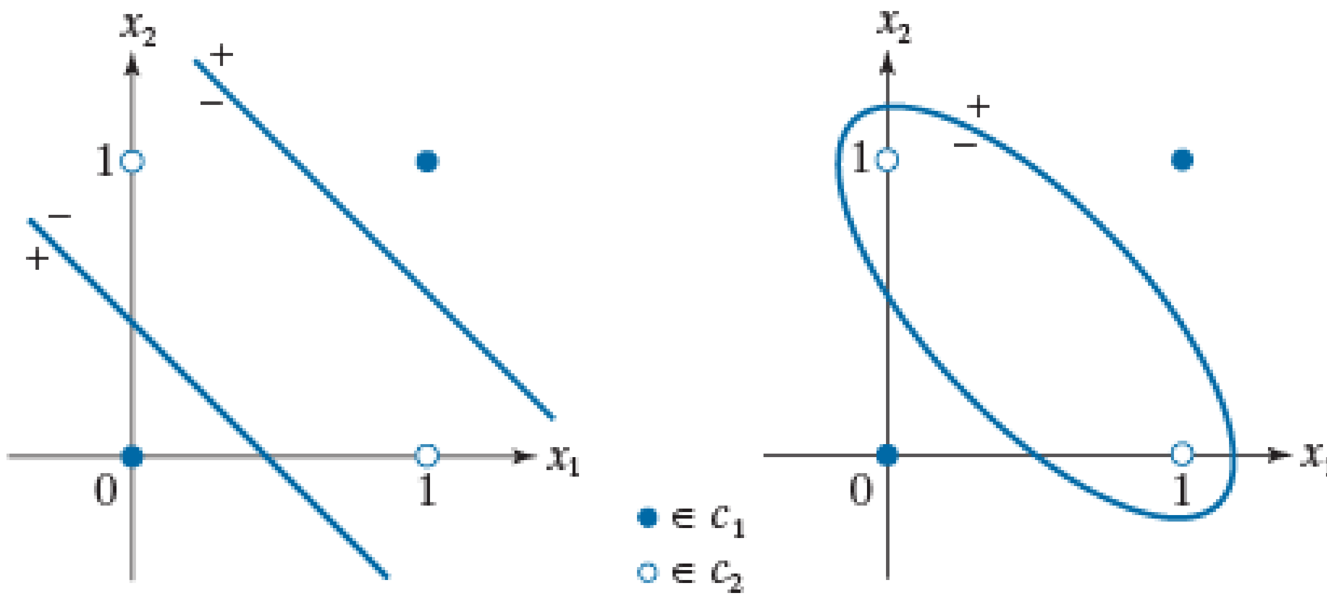
Linearly separable Iris classes

Linearly nonseparable Iris classes



XOR classification problem

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

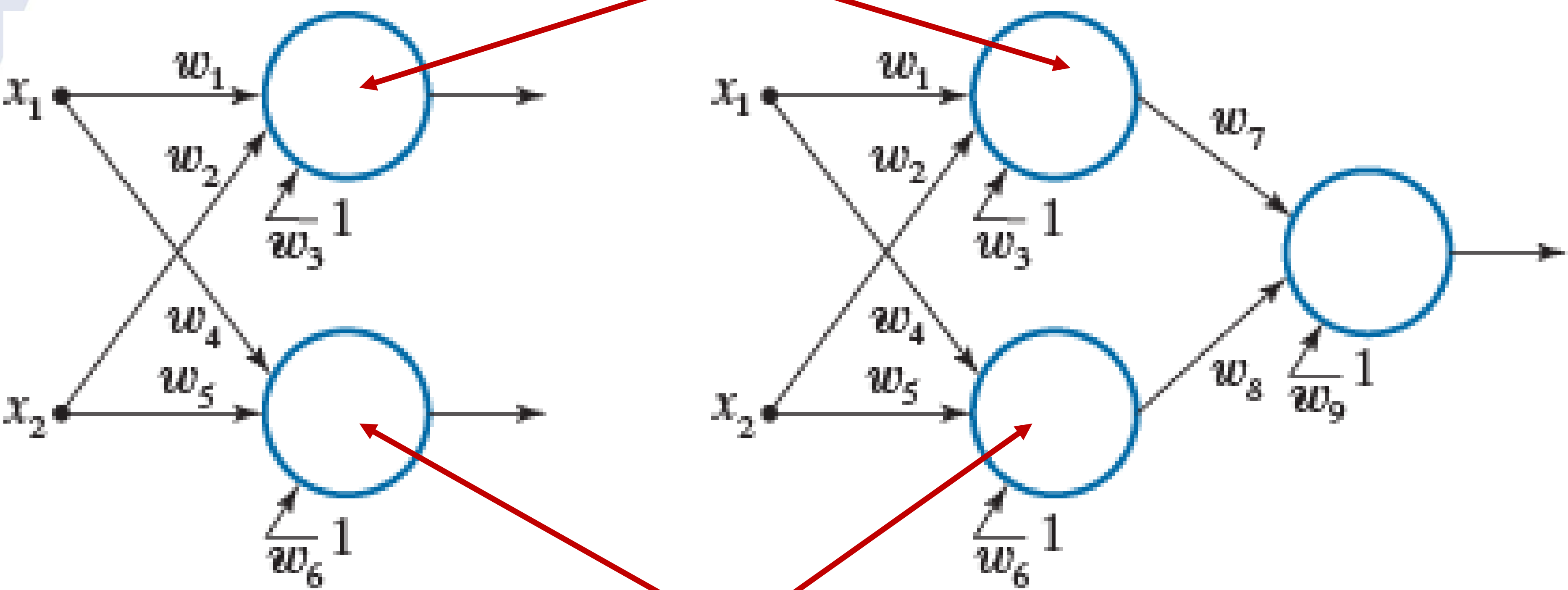


a b c

FIGURE 12.27 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.

How to solve XOR classification problem?

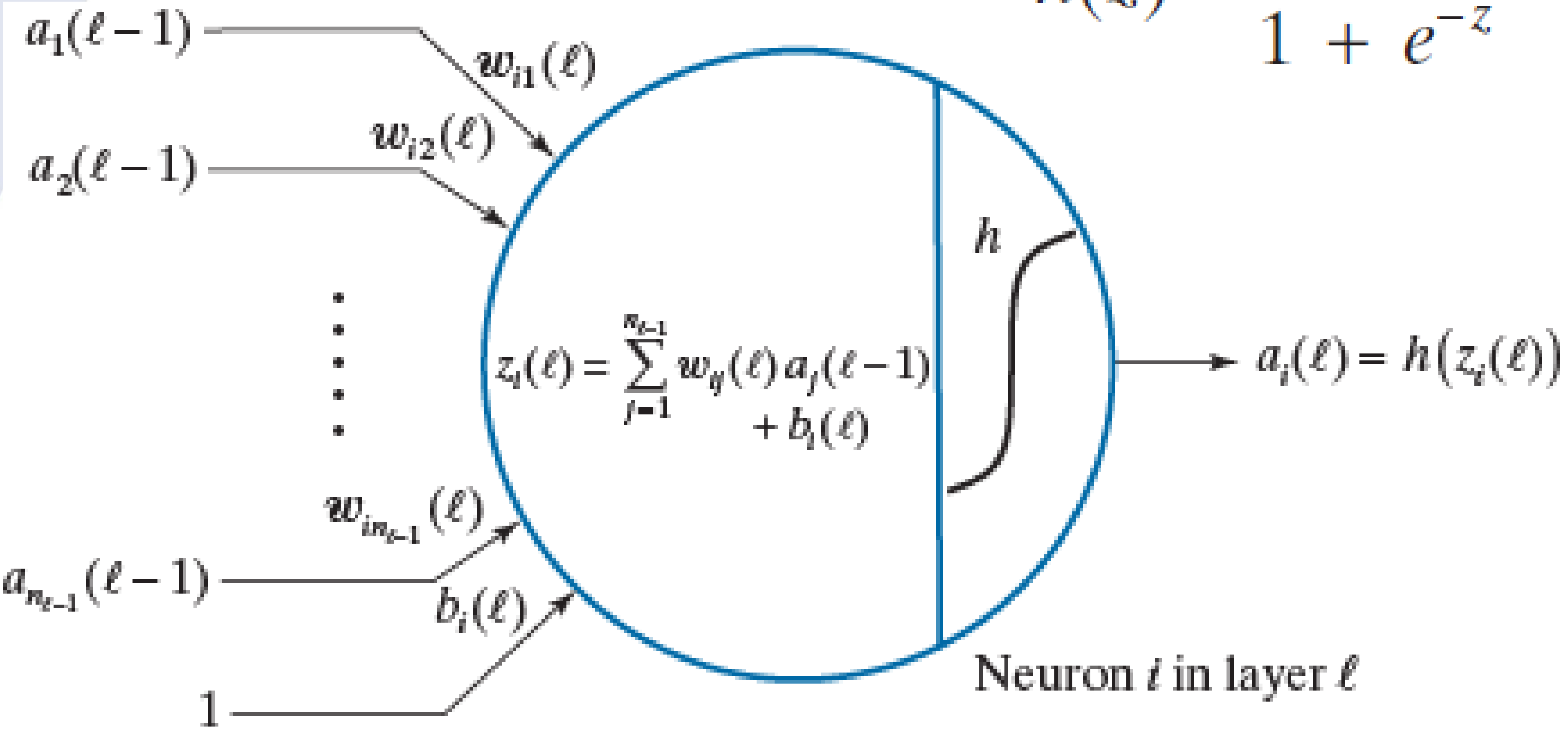
Maps any input from one class into 1



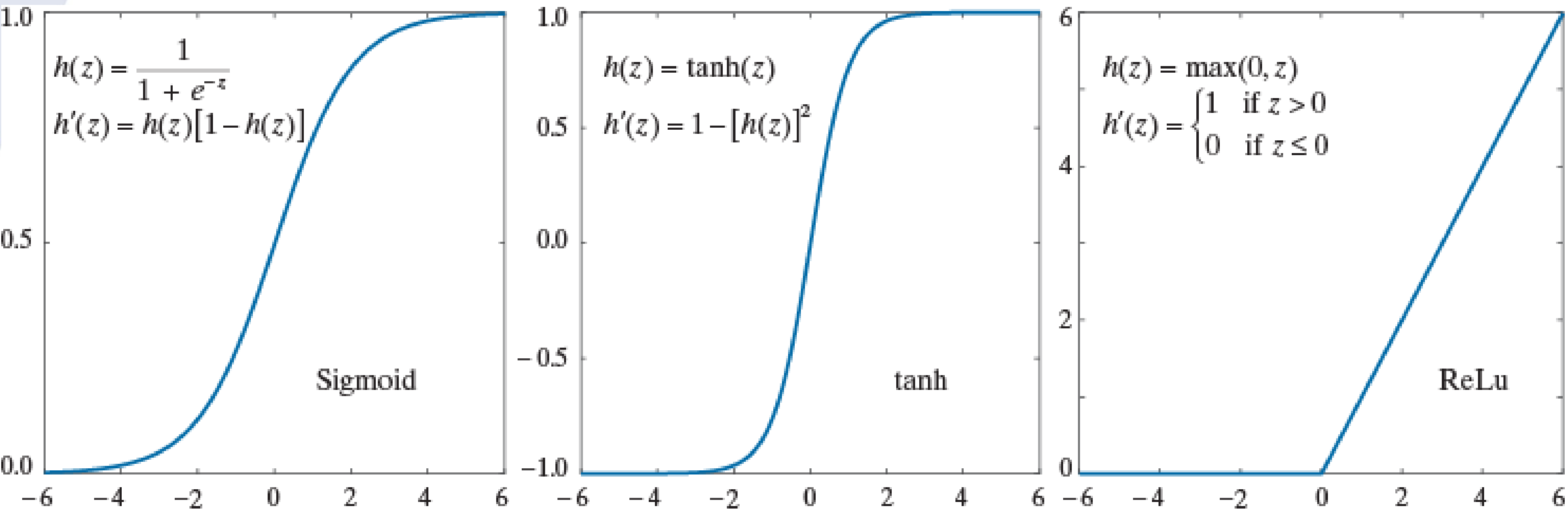
Maps any input from the other class into 0

Model of an Artificial Neuron

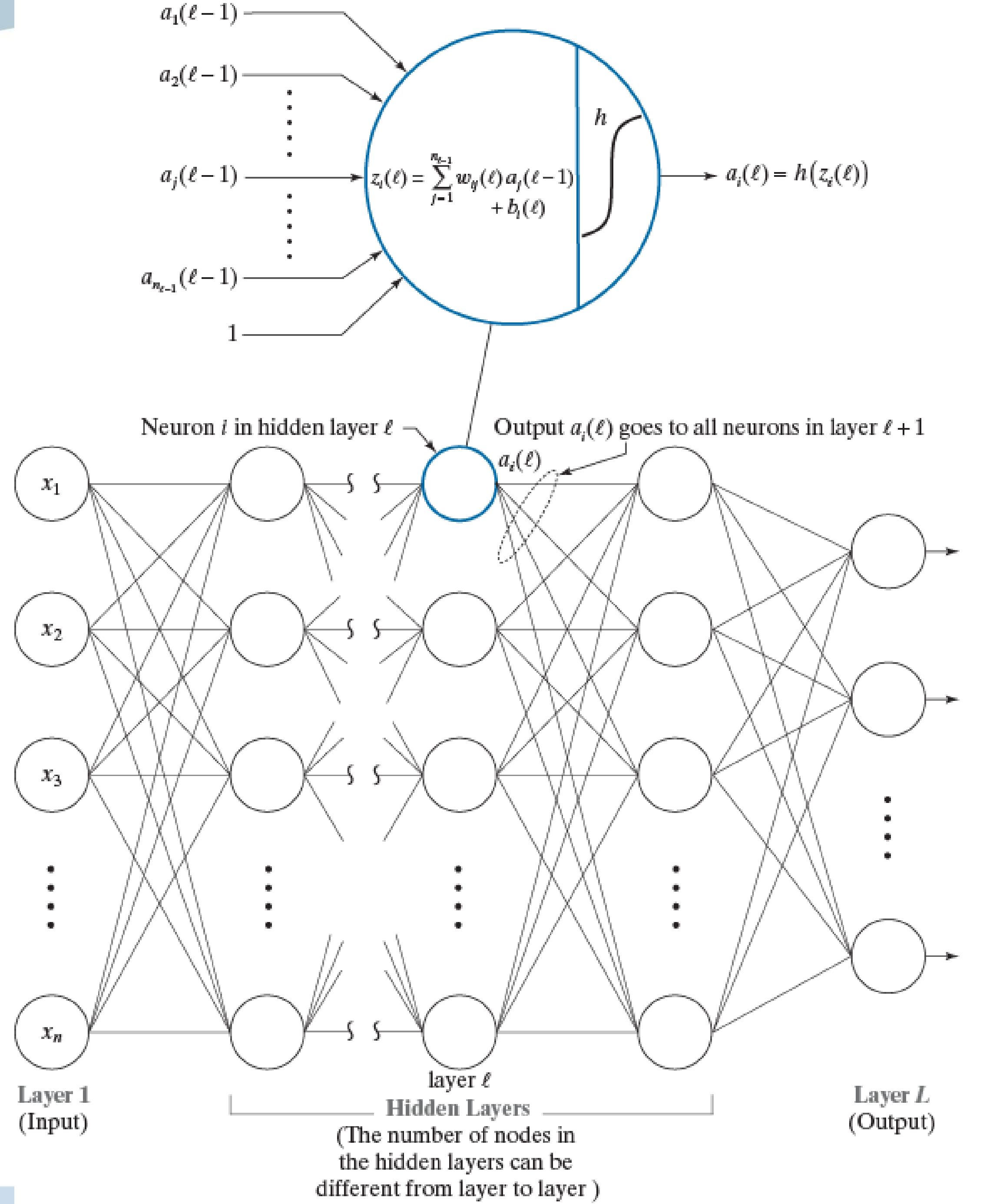
$$h(z) = \frac{1}{1 + e^{-z}}$$



Various activation functions



Multilayer fully-connected feedforward neural network



- Shallow neural network
- Deep neural network

  ≥ 2 hidden layers

Forward Pass Through Network

$$a_j(1) = x_j \qquad j = 1, 2, \ldots, n_1$$

Input

$$z_i(\ell) = \sum_{j=1}^{n_{\ell-1}} w_{ij}(\ell) a_j(\ell - 1) + b_i(\ell)$$

Layer ℓ

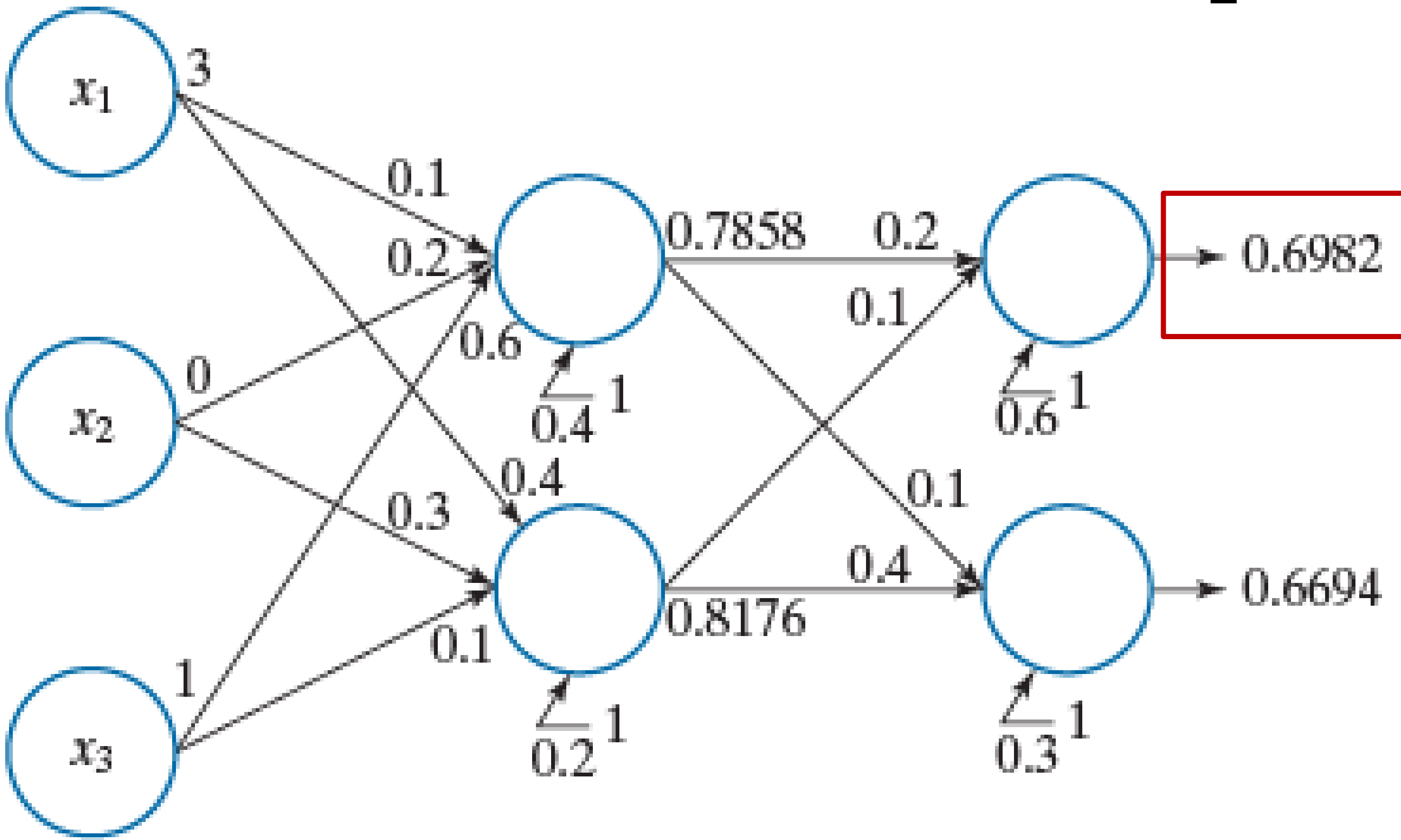
$$a_i(\ell) = h\left(z_i(\ell)\right) \qquad i = 1, 2, \ldots, n_\ell$$

$$a_i(L) = h\left(z_i(L)\right) \qquad i = 1, 2, \ldots, n_L$$

Output

Illustration of a forward pass

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 1 \end{bmatrix}$$



$$z_1(2) = \sum_{j=1}^{3} w_{1j}(2) a_j(1) + b_1(2) = (0.1)(3) + (0.2)(0) + (0.6)(1) + 0.4 = 1.3$$

$$a_1(2) = h\left(z_1(2)\right) = \frac{1}{1+e^{-1.3}} = 0.7858$$

Matrix Formulation

$$\mathbf{a}(1) = \mathbf{x}$$

$$\mathbf{W}(\ell) = \begin{bmatrix} w_{11}(\ell) & w_{12}(\ell) & \cdots & w_{1n_{\ell-1}}(\ell) \\ w_{21}(\ell) & w_{22}(\ell) & \cdots & w_{2n_{\ell-1}}(\ell) \\ \vdots & \vdots & \cdots & \\ w_{n_\ell 1}(\ell) & w_{n_\ell 2}(\ell) & \cdots & w_{n_\ell n_{\ell-1}}(\ell) \end{bmatrix}$$

$$\mathbf{z}(\ell) = \mathbf{W}(\ell)\mathbf{a}(\ell-1) + \mathbf{b}(\ell) \qquad \ell = 2, 3, \ldots, L$$

$$\mathbf{a}(\ell) = h[\mathbf{z}(\ell)] = \begin{bmatrix} h(z_1(\ell)) \\ h(z_2(\ell)) \\ \vdots \\ h(z_{n_\ell}(\ell)) \end{bmatrix}$$

Processing n_p patterns in a single forward pass

$$\mathbf{A}(1) = \mathbf{X} \qquad n \times n_p \text{ matrix}$$

$$\mathbf{Z}(\ell) = \mathbf{W}(\ell)\mathbf{A}(\ell - 1) + \mathbf{B}(\ell)$$

$$\mathbf{A}(\ell) = h\left[\mathbf{Z}(\ell)\right]$$

Use GPU for parallel processing

Training by Back Propagation

- Output layer

$$E = \sum_{j=1}^{n_L} E_j = \frac{1}{2}\sum_{j=1}^{n_L}\left(r_j - a_j(L)\right)^2 = \frac{1}{2}\|\mathbf{r} - \mathbf{a}(L)\|^2$$

$$\delta_j(L) = \frac{\partial E}{\partial z_j(L)} = \frac{\partial E}{\partial a_j(L)}\frac{\partial a_j(L)}{\partial z_j(L)} = \frac{\partial E}{\partial a_j(L)}\frac{\partial h\left(z_j(L)\right)}{\partial z_j(L)}$$

$$= \frac{\partial E}{\partial a_j(L)} h'\left(z_j(L)\right)$$

$$= h\left(z_j(L)\right)\left[1 - h\left(z_j(L)\right)\right]\left[a_j(L) - r_j\right]$$

Training by Back Propagation

- Hidden internal layer

$$\delta_j(\ell) = \frac{\partial E}{\partial z_j(\ell)} = \sum_i \frac{\partial E}{\partial z_i(\ell+1)} \frac{\partial z_i(\ell+1)}{\partial a_j(\ell)} \frac{\partial a_j(\ell)}{\partial z_j(\ell)}$$

$$= \sum_i \delta_i(\ell+1) \frac{\partial z_i(\ell+1)}{\partial a_j(\ell)} h'\left(z_j(\ell)\right)$$

$$= h'\left(z_j(\ell)\right) \sum_i w_{ij}(\ell+1)\delta_i(\ell+1)$$

Training by Back Propagation

- Update network parameters using gradient descent

$$\frac{\partial E}{\partial w_{ij}(\ell)} = \frac{\partial E}{\partial z_i(\ell)} \frac{\partial z_i(\ell)}{\partial w_{ij}(\ell)}$$

$$= \delta_i(\ell) \frac{\partial z_i(\ell)}{\partial w_{ij}(\ell)}$$

$$= a_j(\ell-1)\delta_i(\ell)$$

$$\frac{\partial E}{\partial b_i(\ell)} = \delta_i(\ell)$$

Learning rate

$$w_{ij}(\ell) = w_{ij}(\ell) - \alpha \frac{\partial E(\ell)}{\partial w_{ij}(\ell)}$$

$$= w_{ij}(\ell) - \alpha \delta_i(\ell) a_j(\ell-1)$$

$$b_i(\ell) = b_i(\ell) - \alpha \frac{\partial E}{\partial b_i(\ell)}$$

$$= b_i(\ell) - \alpha \delta_i(\ell)$$

Matrix Formulation

TABLE 13.3

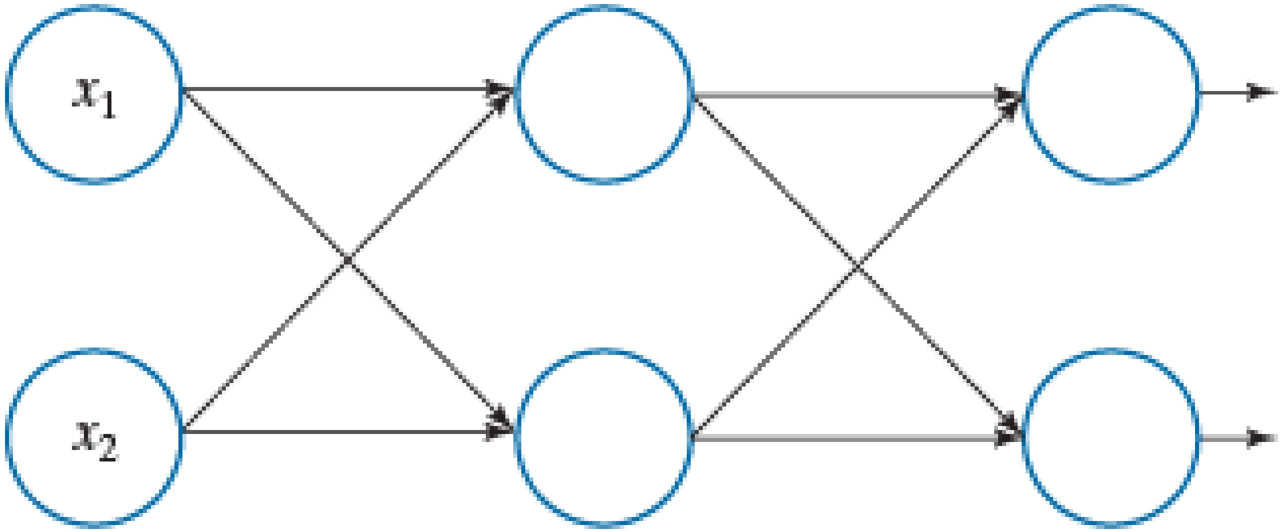
Matrix formulation for training a feedforward, fully connected multilayer neural network using backpropagation. Steps 1–4 are for one epoch of training. **X**, **R**, and the learning rate parameter α, are provided to the network for training. The network is initialized by specifying weights, $\mathbf{W}(1)$, and biases, $\mathbf{B}(1)$, 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: $\mathbf{Z}(\ell) = \mathbf{W}(\ell)\mathbf{A}(\ell-1) + \mathbf{B}(\ell)$; $\mathbf{A}(\ell) = h\big(\mathbf{Z}(\ell)\big)$; $h'\big(\mathbf{Z}(\ell)\big)$; and $\mathbf{D}(L) = \big(\mathbf{A}(L) - \mathbf{R}\big) \odot h'\big(\mathbf{Z}(L)\big)$
Step 3	Backpropagation	For $\ell = L-1, L-2, \ldots, 2$, compute $\mathbf{D}(\ell) = \Big(\mathbf{W}^T(\ell+1)\mathbf{D}(\ell+1)\Big) \odot h'\big(\mathbf{Z}(\ell)\big)$
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 $\mathbf{D}(\ell)$

Fully Connected Net for XOR Problem

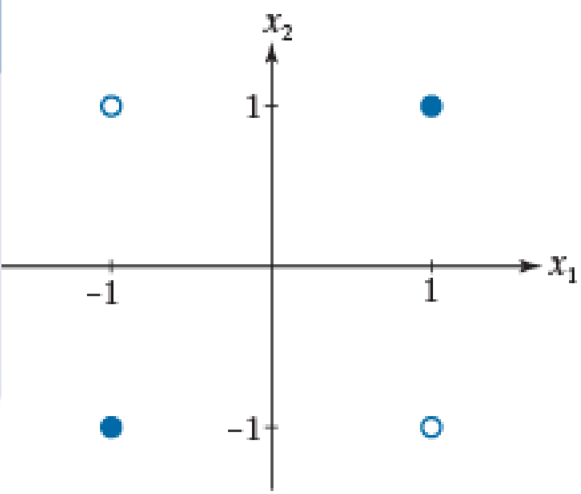
$$\mathbf{X} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}; \quad \mathbf{R} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

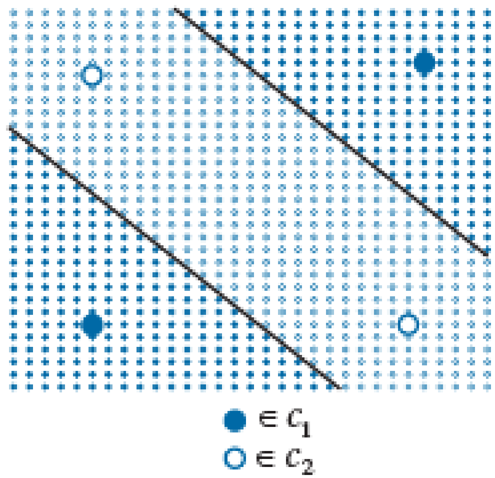
$$\mathbf{W}(2) = \begin{bmatrix} 4.792 & 4.792 \\ 4.486 & 4.486 \end{bmatrix} \quad \mathbf{W}(3) = \begin{bmatrix} -9.180 & 9.429 \\ 9.178 & -9.427 \end{bmatrix}$$

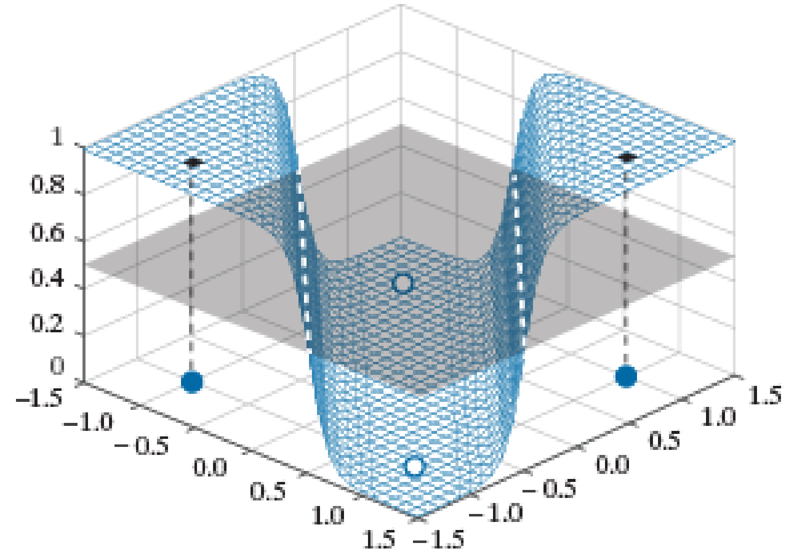


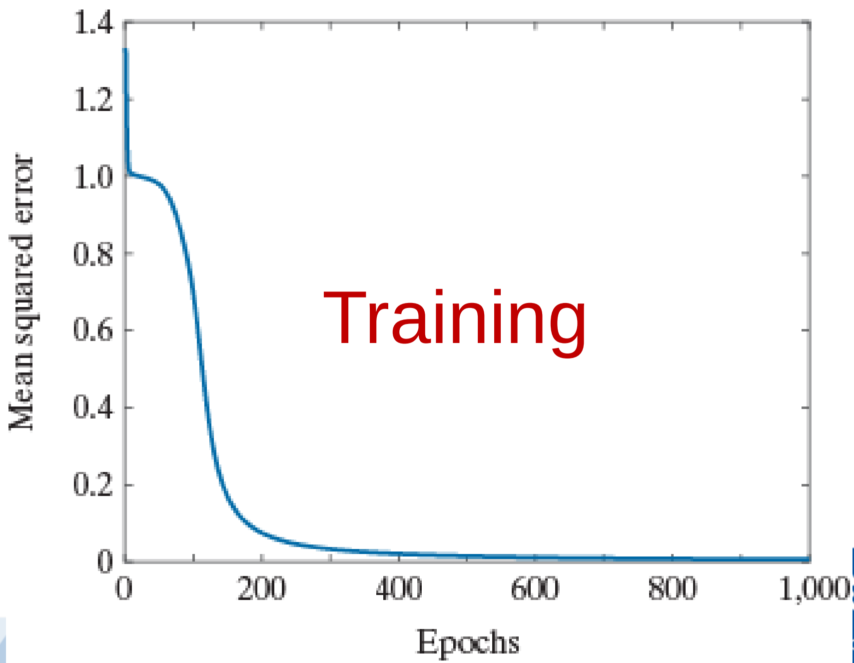
$$\mathbf{b}(2) = \begin{bmatrix} 4.590 \\ -4.486 \end{bmatrix} \quad \mathbf{b}(3) = \begin{bmatrix} 4.420 \\ -4.419 \end{bmatrix}$$

Fully Connected Net for XOR Problem

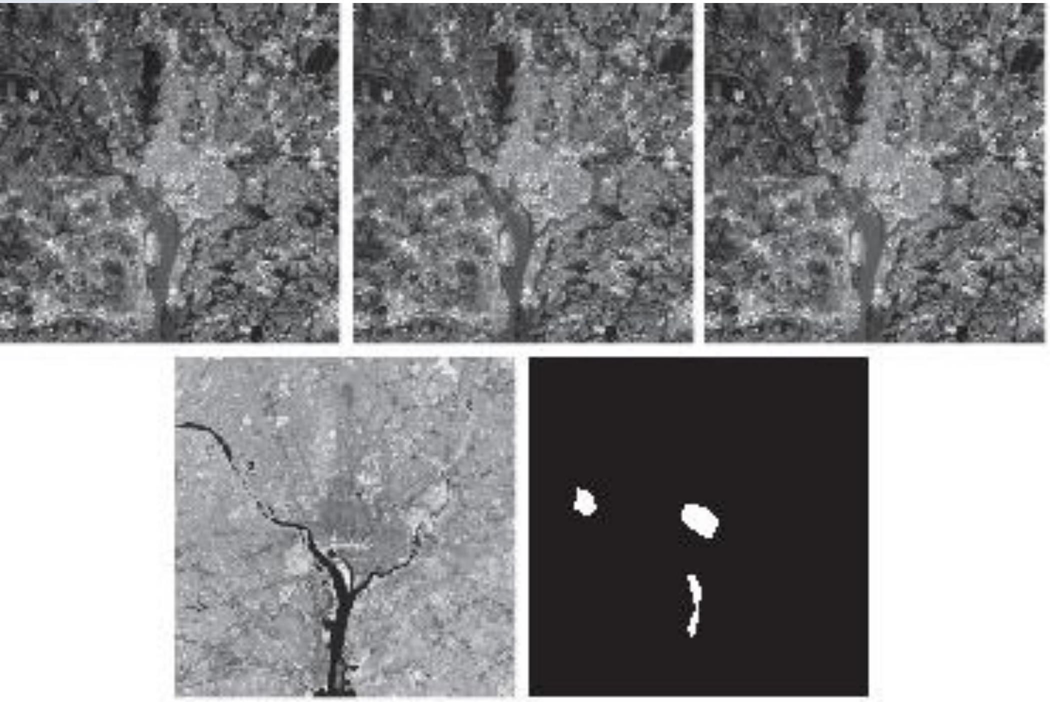




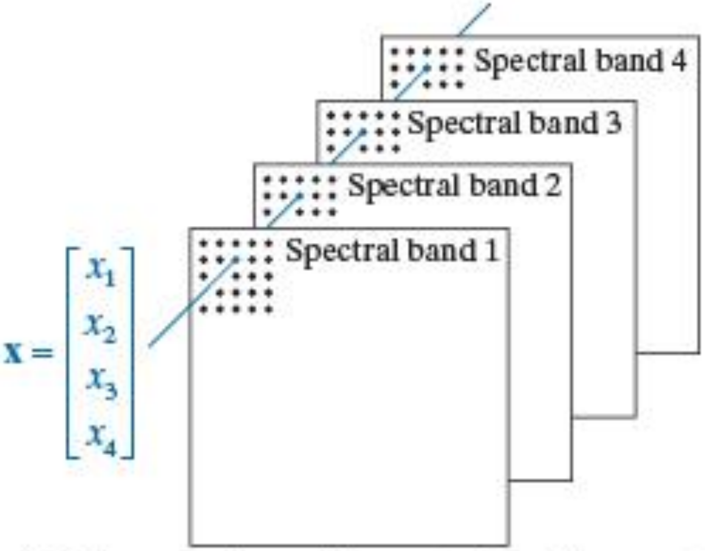




Classify Multispectral Image Data



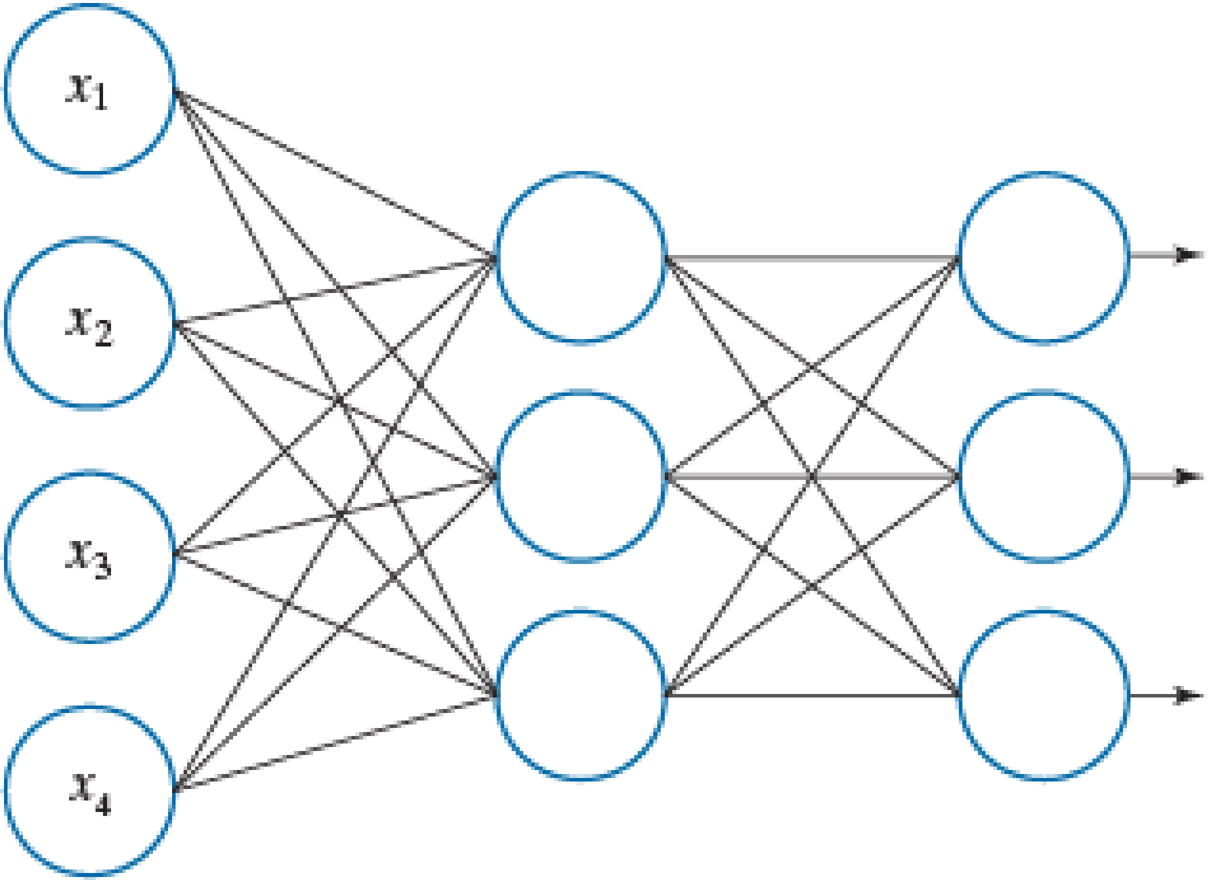
(a) Images in spectral bands 1–4 and binary mask used to extract training samples



(b) Approach used to extract pattern vectors

Classify Multispectral Image Data

FIGURE 12.38
Neural net architecture used to classify the multispectral image data in Fig. 12.37 into three classes: water, urban, and vegetation. The parameters shown were obtained in 50,000 epochs of training using $\alpha = 0.001$.



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

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

Classify Multispectral Image Data

FIGURE 12.39
MSE for the network architecture in Fig. 12.38 as a function of the number of training epochs. The learning rate parameter was $\alpha = 0.001$ in all cases.

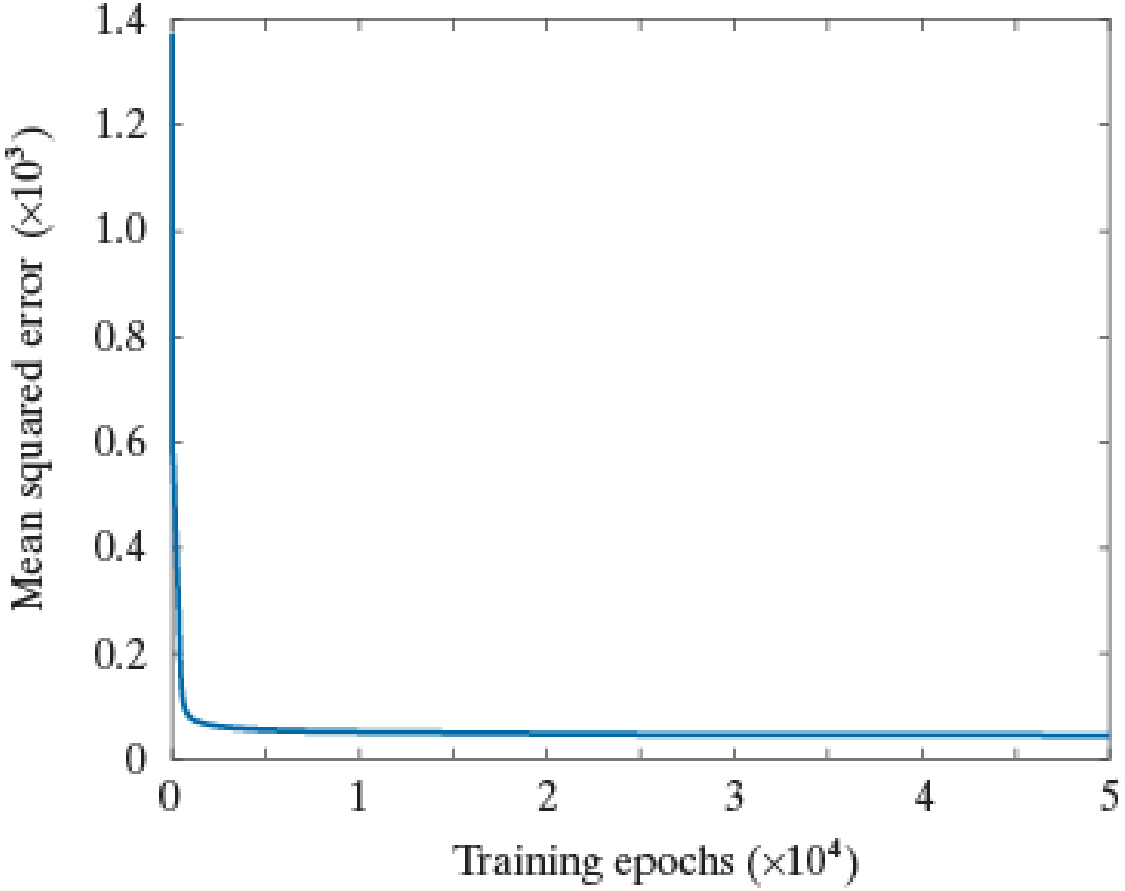
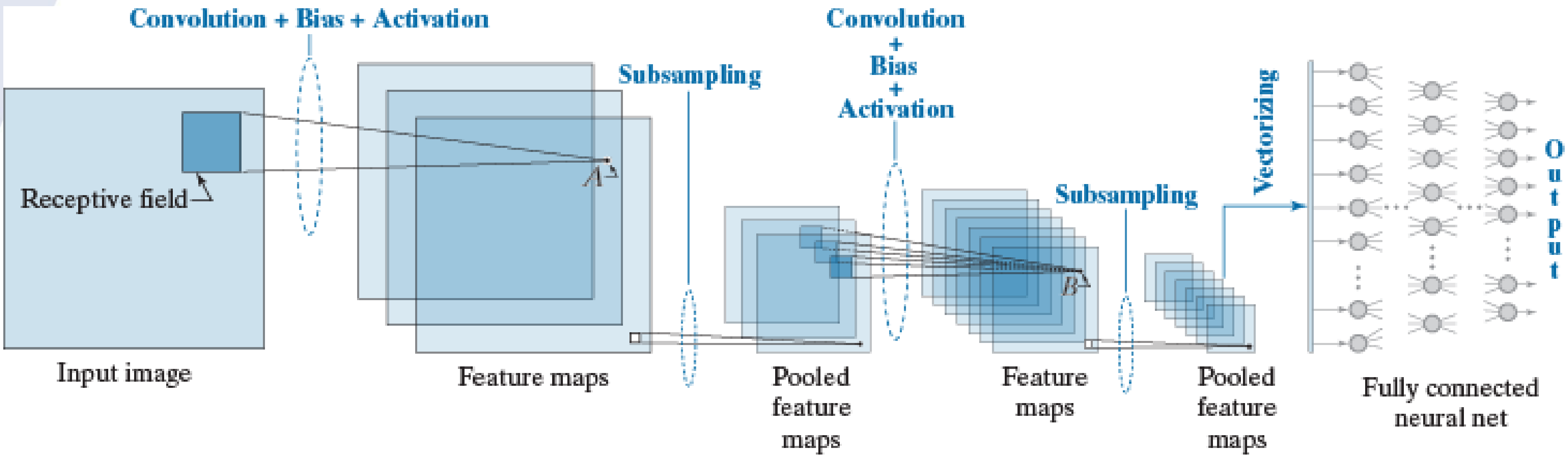


TABLE 13.5
Recognition performance on the training set as a function of training epochs. The learning rate constant was $\alpha = 0.001$ in all cases.

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%

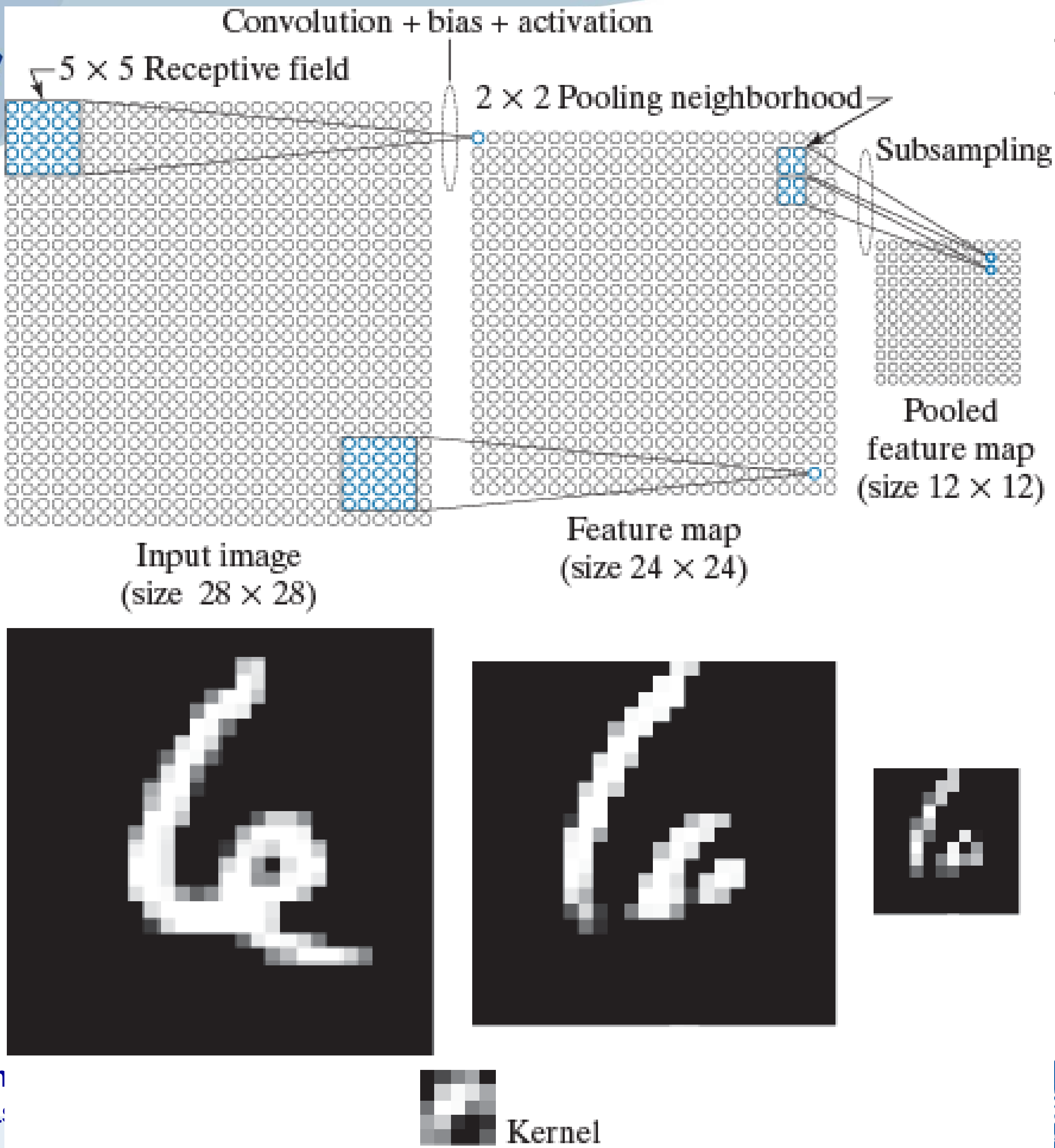
Contents

Convolutional Neural Network (CNN)

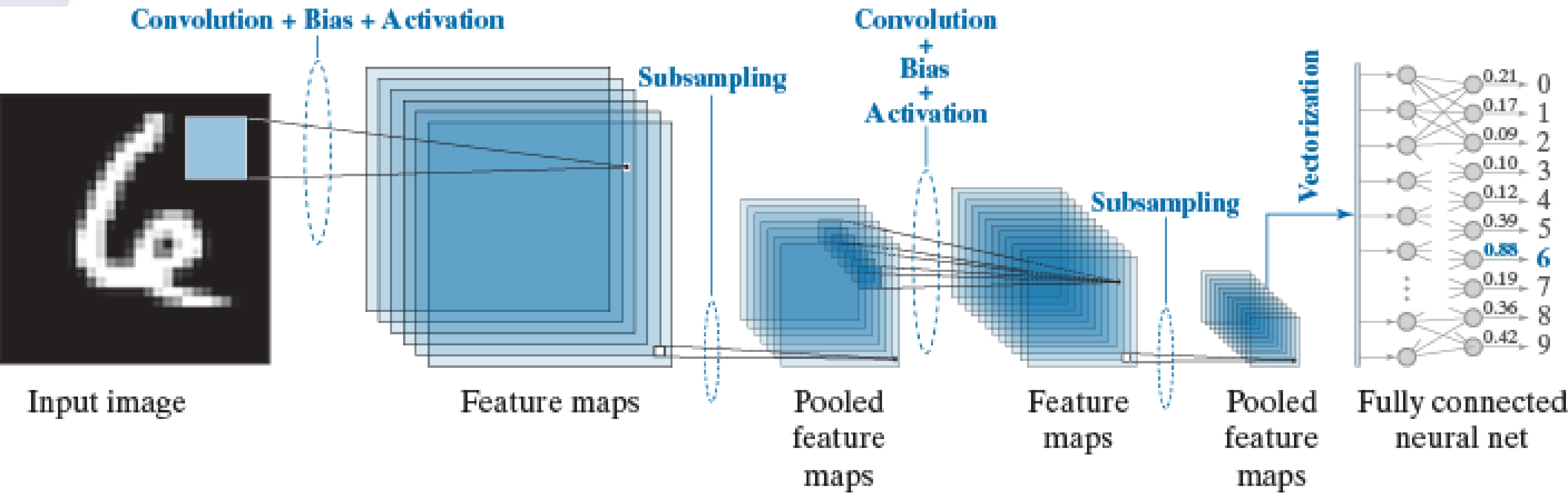


LeNet

Example


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

Example


Convolution + Bias + Activation
Subsampling
Convolution
+
Bias
+
Activation
Subsampling
Vectorization
0.21
0
0.17
1
0.09
2
0.10
3
0.12
4
0.39
5
0.88
6
0.19
7
0.36
8
0.42
9
Input image
Feature maps
Pooled feature maps
Feature maps
Pooled feature maps
Fully connected neural net

Weights

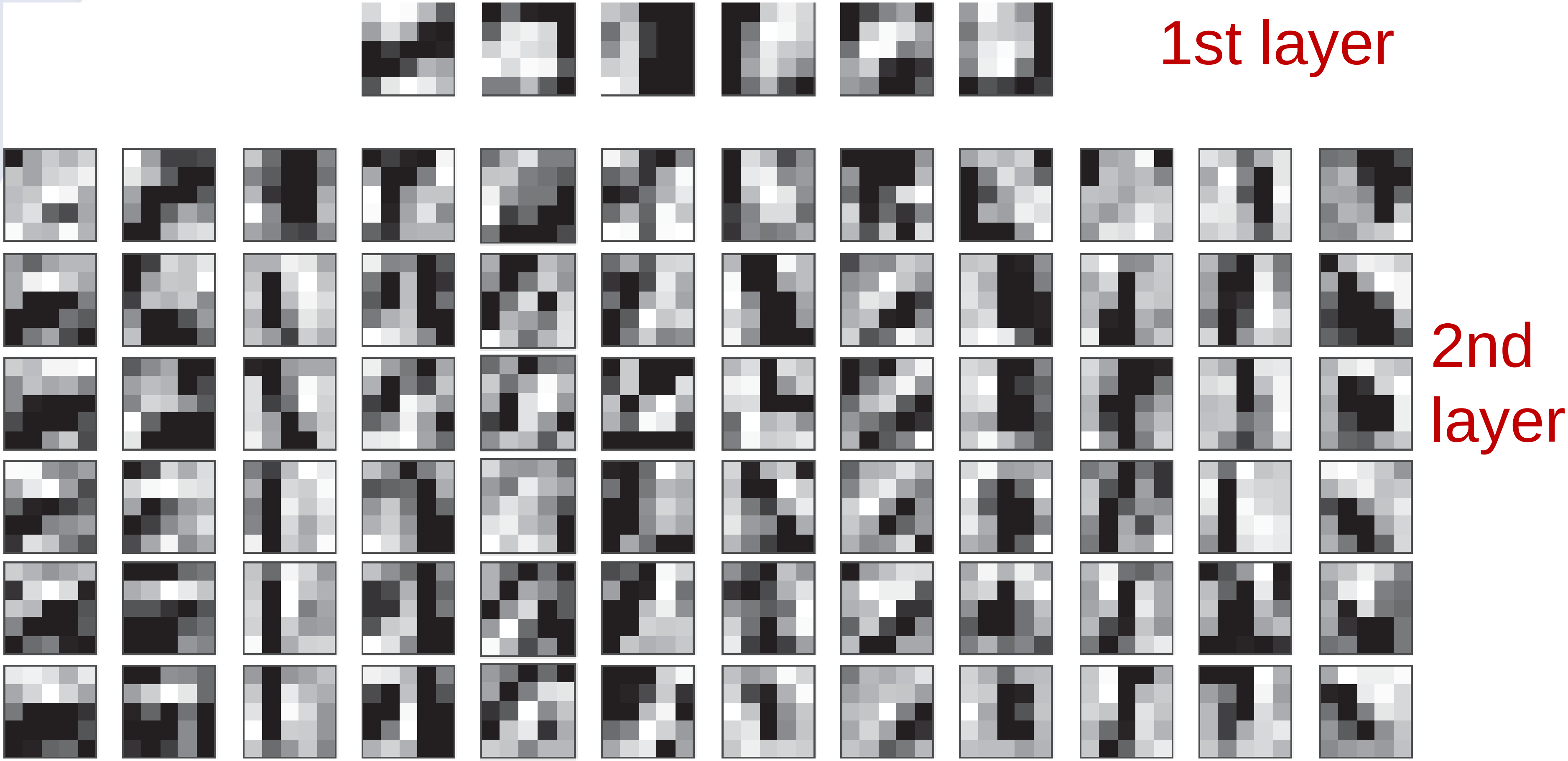
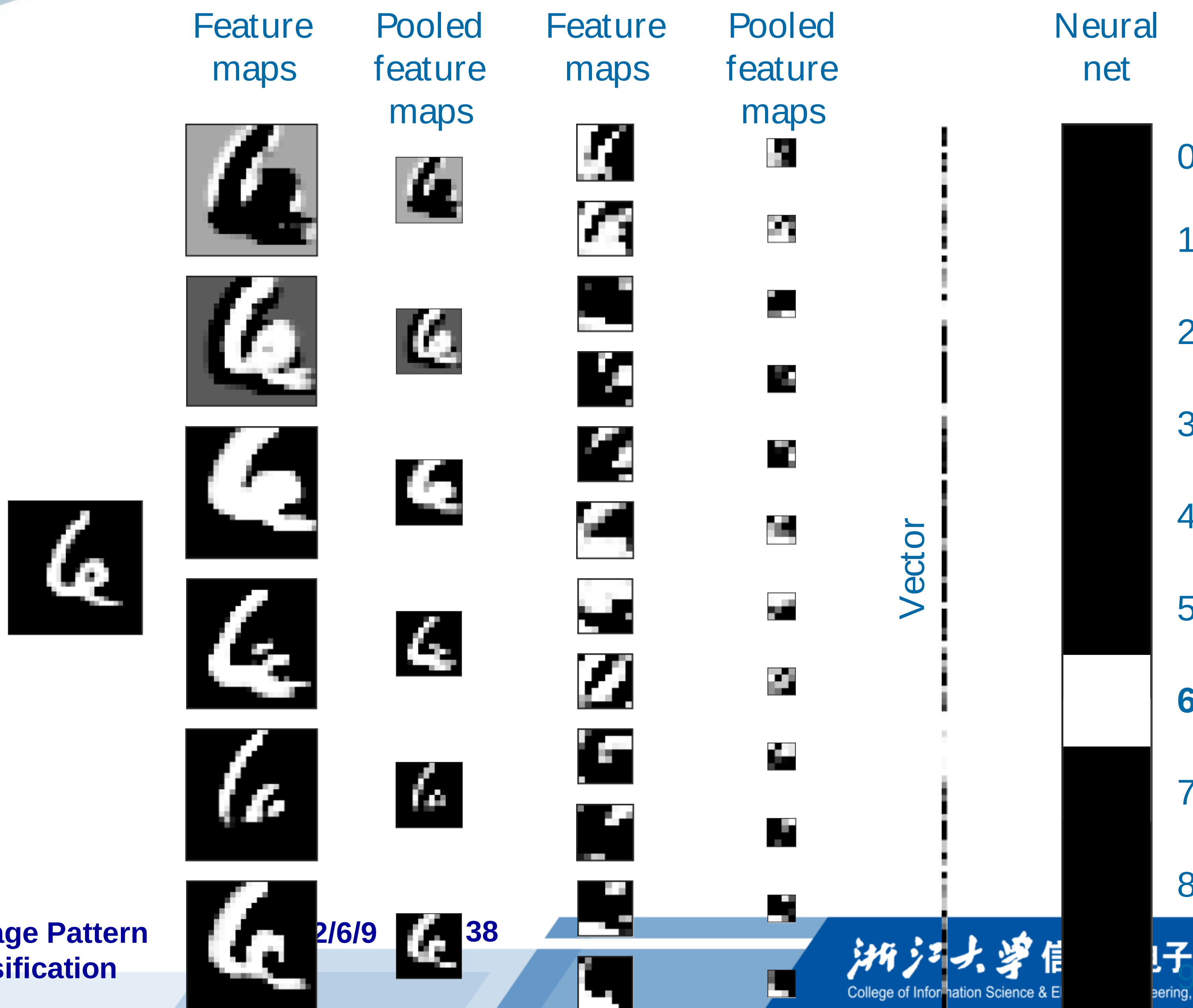


FIGURE 12.43 Top: The weights (shown as images of size 5×5) corresponding to the six feature maps in the first layer of the CNN in Fig. 12.42. Bottom: The weights corresponding to the twelve feature maps in the second layer.

An input image goes through the CNN



Neural Computations in a CNN

- Convolution $$w \star a_{x,y} = \sum_l \sum_k w_{l,k} a_{x-l,y-k}$$
- Add a bias $$z = w \star a_{x,y} + b$$
- Activation $$a = h(z)$$

Forward pass through a CNN

$$z_{x,y}(\ell) = \sum_{l}\sum_{k} w_{l,k}(\ell)a_{x-l,y-k}(\ell-1) + b(\ell)$$
$$= w(\ell)\star a_{x,y}(\ell-1) + b(\ell)$$

$$a_{x,y}(\ell) = h\left(z_{x,y}(\ell)\right)$$

Backpropagation for training

$$\delta_{x,y}(\ell) = \frac{\partial E}{\partial z_{x,y}(\ell)} = \sum_u \sum_v \frac{\partial E}{\partial z_{u,v}(\ell+1)} \frac{\partial z_{u,v}(\ell+1)}{\partial z_{x,y}(\ell)}$$

$$= h'\left(z_{x,y}(\ell)\right)\left[\delta_{x,y}(\ell+1) \star \mathrm{rot}180\left(w(\ell+1)\right)\right]$$

$$\frac{\partial E}{\partial w_{l,k}} = \delta_{l,k}(\ell) \star \mathrm{rot}180\left(a(\ell-1)\right)$$

$$\frac{\partial E}{\partial b(\ell)} = \sum_x \sum_y \delta_{x,y}(\ell)$$

Backpropagation for training

$$w_{l,k}(\ell) = w_{l,k}(\ell) - \alpha \frac{\partial E}{\partial w_{l,k}}$$

$$= w_{l,k}(\ell) - \alpha \delta_{l,k}(\ell) \star \mathrm{rot}180\big(a(\ell - 1)\big)$$

$$b(\ell) = b(\ell) - \alpha \frac{\partial E}{\partial b(\ell)}$$

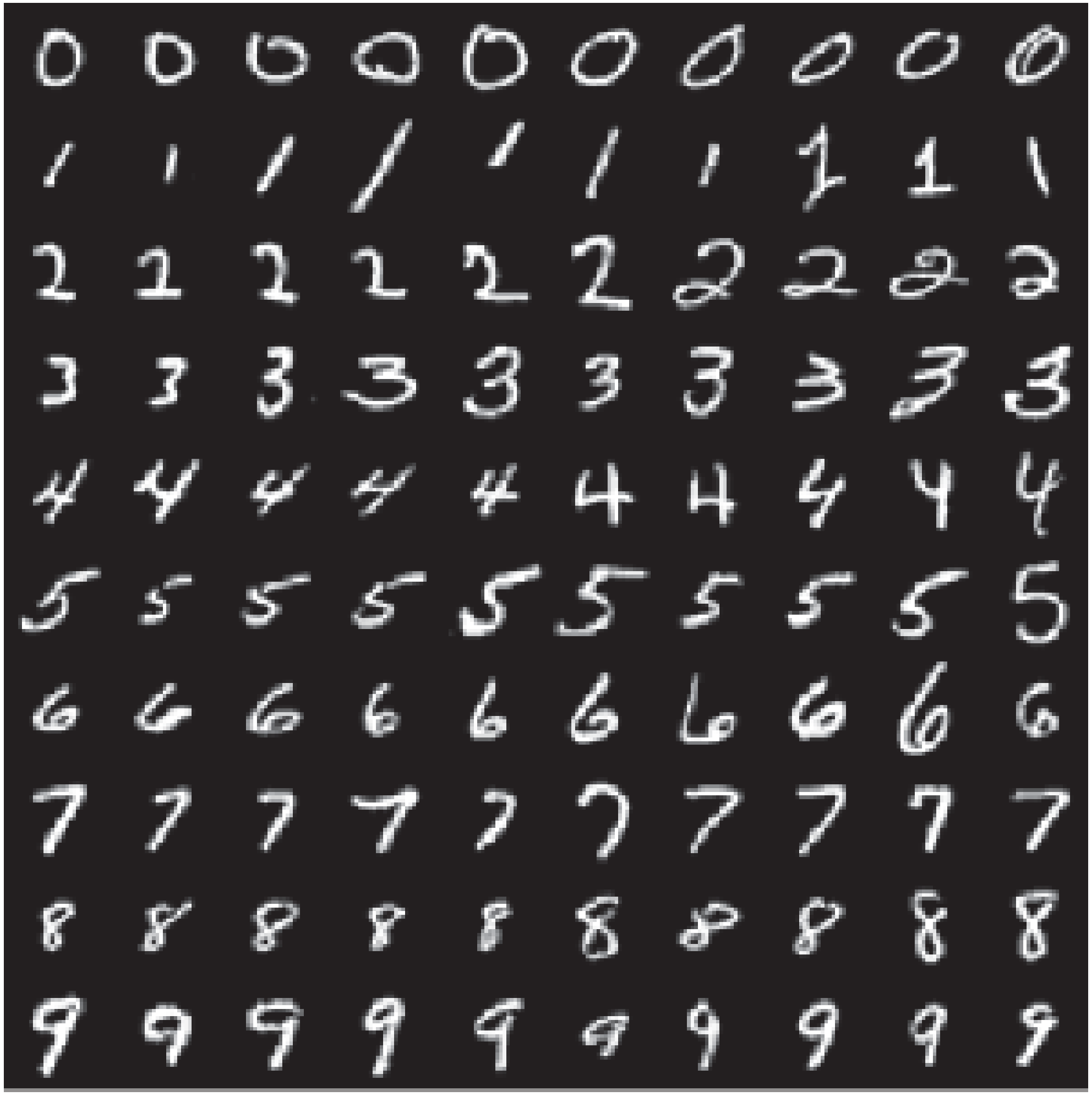
$$= b(\ell) - \alpha \sum_x \sum_y \delta_{x,y}(\ell)$$

Training Steps

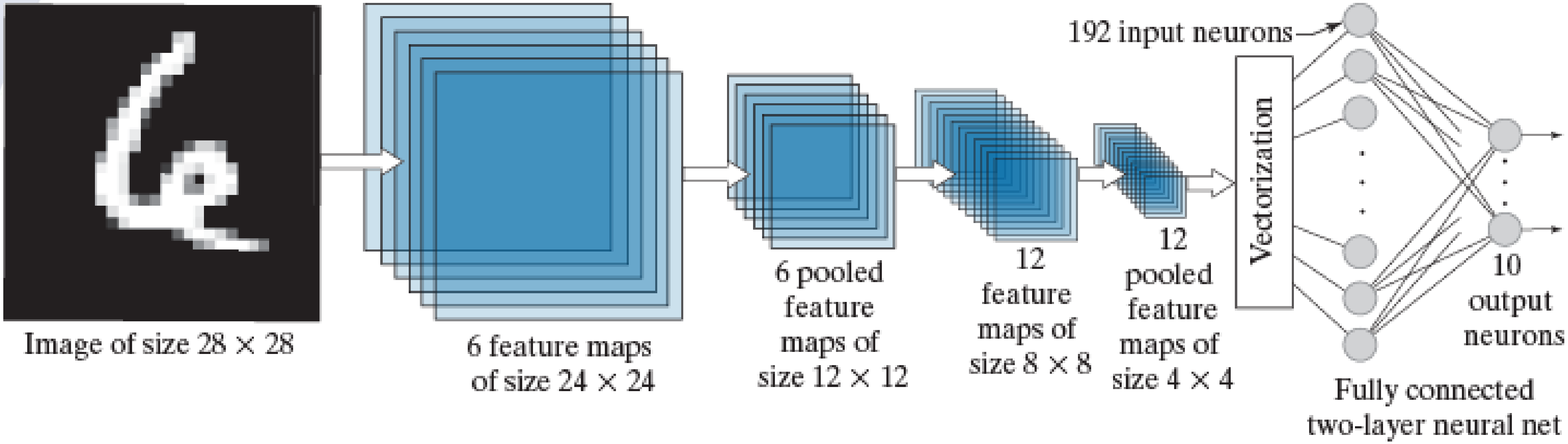
TABLE 13.6
The principal steps used to train a CNN. The network is initialized with a set of small random weights and biases. In backpropagation, a vector arriving (from the fully connected net) at the output pooling layer must be converted to 2-D arrays of the same size as the pooled feature maps in 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: $\delta_{x,y}(\ell) = h'\left(z_{x,y}(\ell)\right)\left[\delta_{x,y}(\ell+1) \star \text{rot}180\left(w(\ell+1)\right)\right]$; $\ell = L_c - 1, L_c - 2, \ldots, 1$
Step 4	Update parameters	Update the weights and bias for each feature map using $w_{l,k}(\ell) = w_{l,k}(\ell) - \alpha\, \delta_{l,k}(\ell) \star \text{rot}180\left(a(\ell - 1)\right)$ and $b(\ell) = b(\ell) - \alpha \sum_x \sum_y \delta_{x,y}(\ell)$; $\ell = 1, 2, \ldots, L_c$

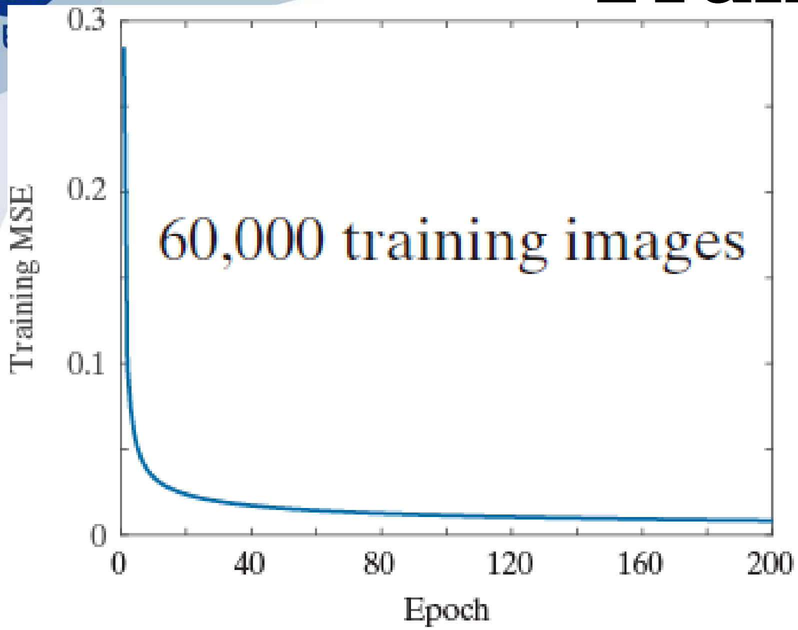
Handwritten Numeral recognition

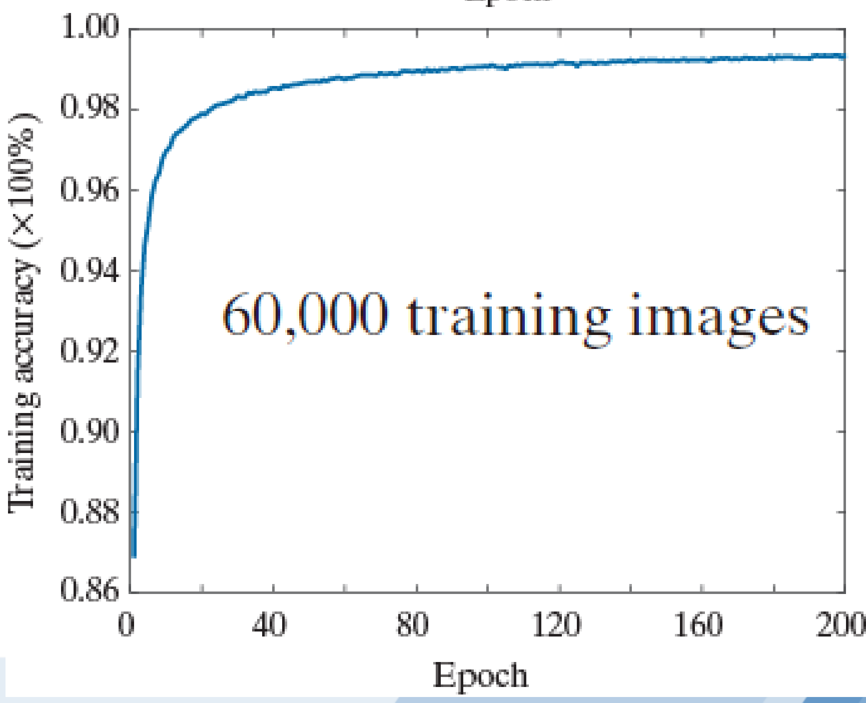


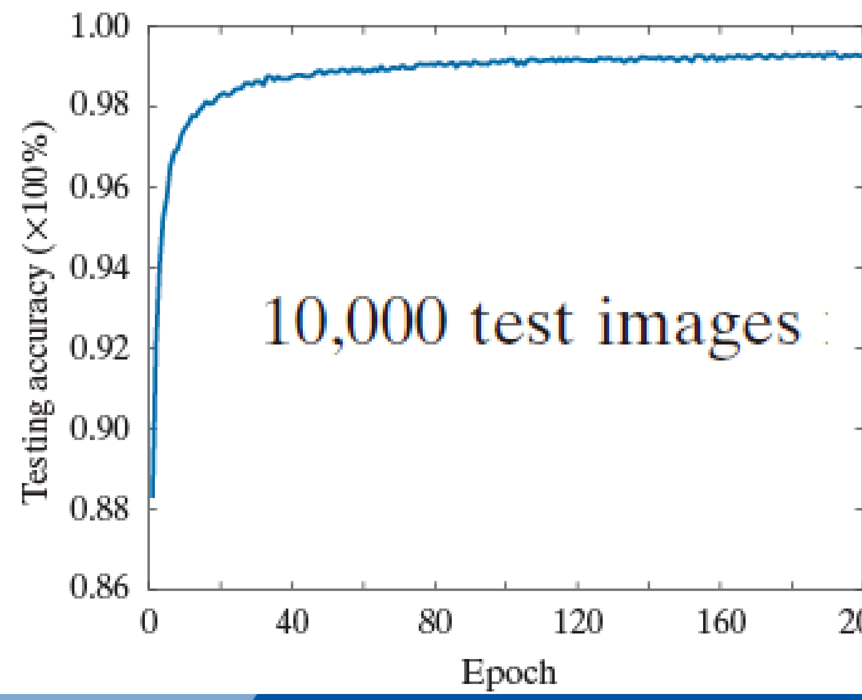
Handwritten Numeral recognition



Training MSE & accuracy







Kernels

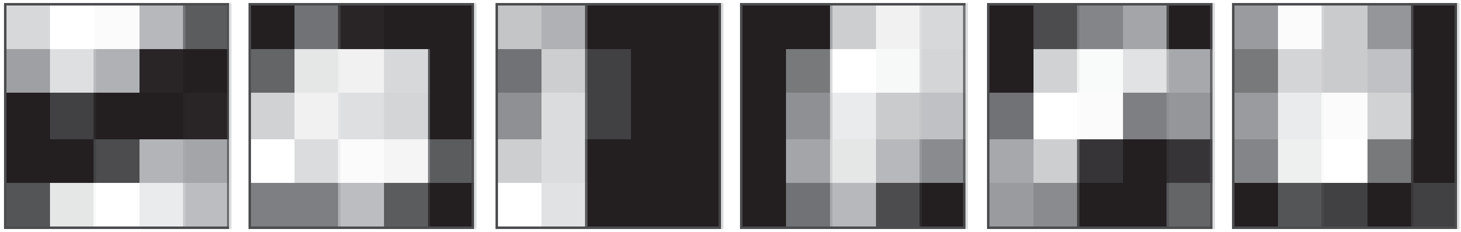


FIGURE 12.53 Kernels of the first layer after 200 epochs of training, shown as images.

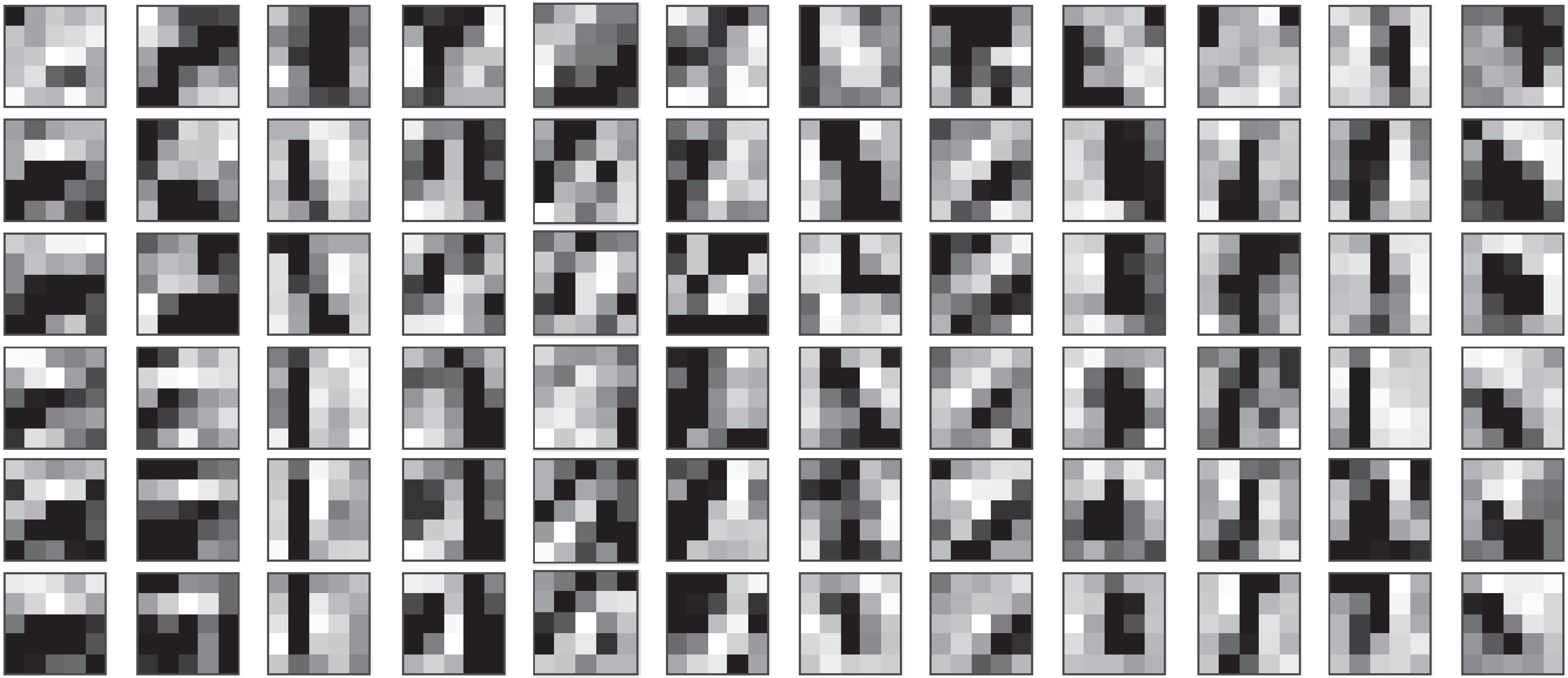


FIGURE 12.54 Kernels of the second layer after 200 epochs of training, displayed as images of size 5×5. There are six inputs (pooled feature maps) into the second layer. Because there are twelve feature maps in the second layer, the CNN learned the weights of $6 \times 12 = 72$ kernels.

Forward pass for one image

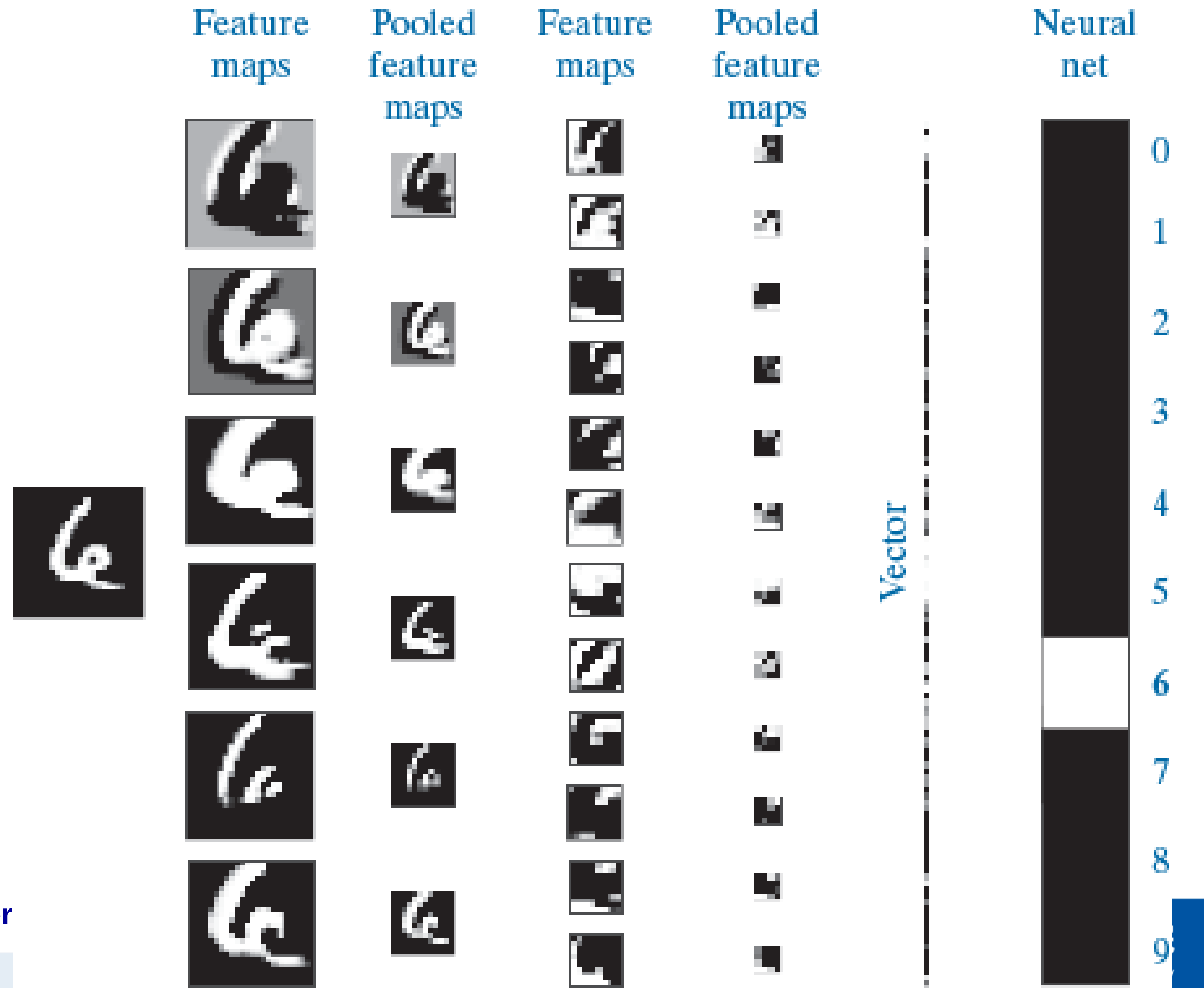
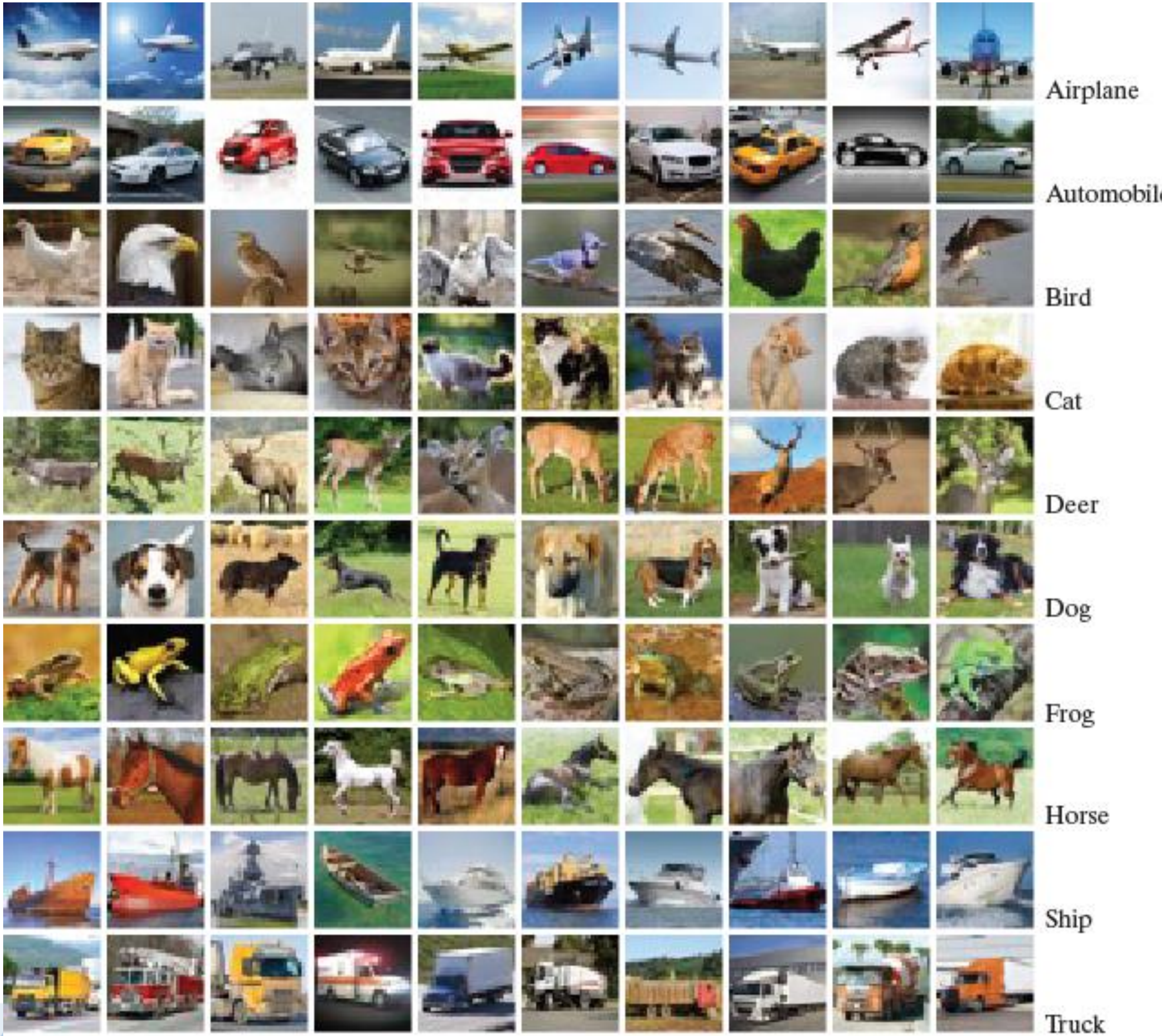


FIGURE 12.55 Results of a forward pass for one digit image through the CNN in Fig. 12.49 after training. The feature maps were generated using the kernels from Figs. 12.53 and 12.54, followed by pooling. The neural net is the two-layer neural network from Fig. 12.49. The output high value (in white) indicates that the CNN recognized the input properly. (This figure is the same as Fig. 12.44.)

CIFAR-10

FIGURE 12.56
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.)



Forward pass for one image

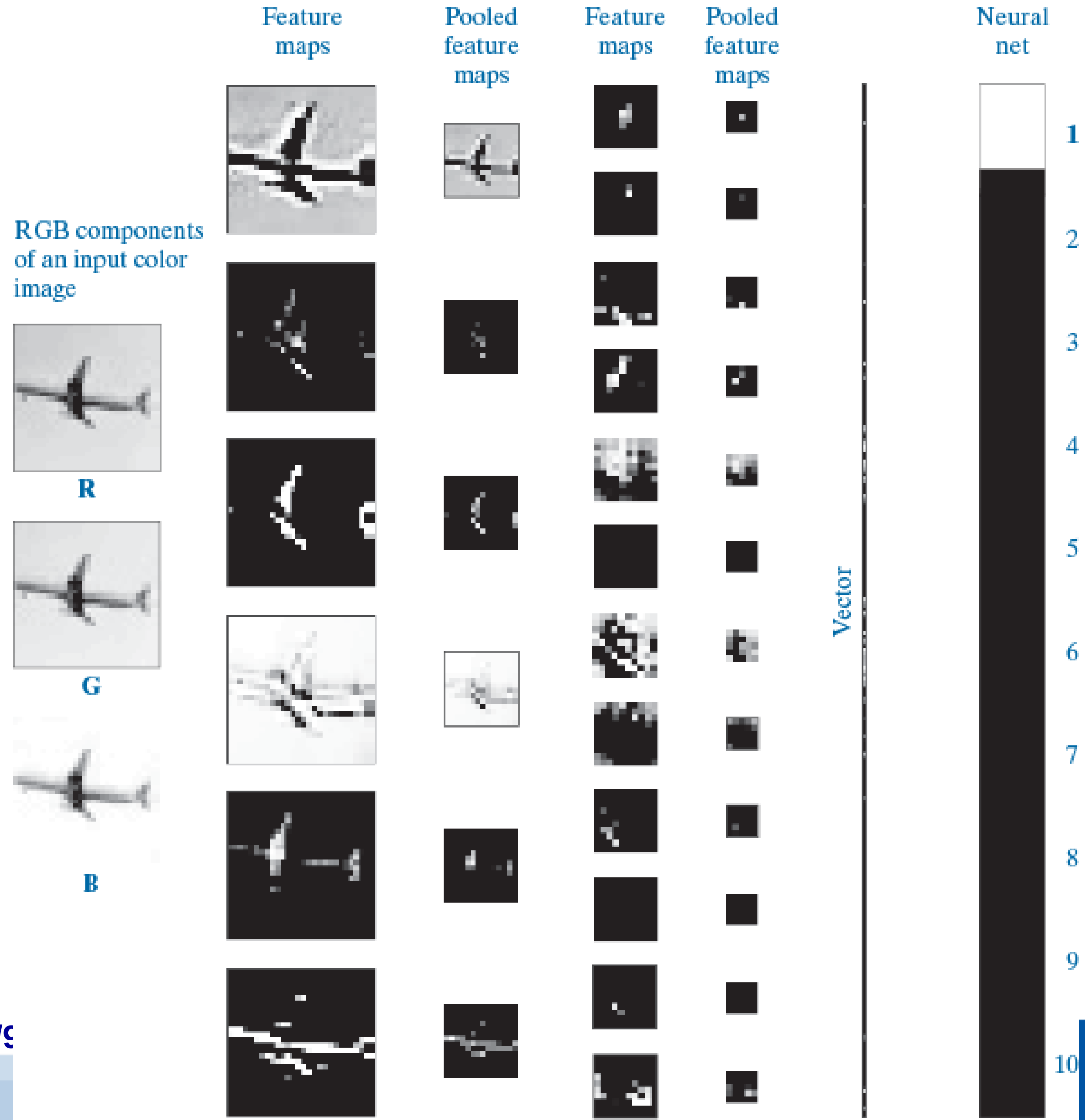


FIGURE 12.62 Graphical illustration of a forward pass through the trained CNN. The purpose was to recognize one input image from the set in Fig. 12.56. As the output shows, the image was recognized correctly as belonging to class 1, the class of airplanes. (Original image courtesy of Pearson Education.)