


# Object Classification

Kenneth R. Castleman and Qiang Wu



## 9.1 Introduction

Classification is the step that tells us what is in the image. Assuming the objects in the image have been segmented and measured, classification identifies them by assigning each of them to one of several previously established types, categories, or classes. There are several mathematical approaches that can be taken to address the classification problem, and a complete coverage is beyond our scope. Here we illustrate the process of classification with the very useful maximum likelihood method. This technique is widely used because it minimizes the probability of making an incorrect assignment. More specifically, we present the minimum Bayes risk classifier, assuming Gaussian statistics, along with several of its interesting special cases. We also address a few other classification strategies.



## 9.2 The Classification Process

When we encounter an unknown object in a microscopic image, we know three things about it. First, we know something about the objects in general. We know the *a priori probability* that an object belongs to each of the classes. For example, if we are attempting to separate abnormal from normal cells, we might know from past experience that 90% of all cells encountered are normal. Thus the *a priori probability* for class 1 (normal) is 0.9, while for class 2 (abnormal) it is 0.1.

$$P(C_1) = 0.9 \quad \text{and} \quad P(C_2) = 0.1 \quad (9.1)$$

This knowledge applies to all of the objects. Quite a large sample of cells might be required to estimate these prior probabilities [1]. Second, we know something about each class. We know the probability density function (pdf) of the features for each of the classes. Third, we know something about each particular object. We know its measured feature values. This is the quantitative data we have that is unique to that particular object. Given these three pieces of information, we seek to make an optimal assignment of that object to a class. For the moment we take the probability of error as the performance criterion, and we seek to minimize it.

### 9.2.1 Bayes' Rule

We now consider how to combine the three things we know about an object to find its most likely class. After an object has been measured, we should be able to use the measurement data and the class-conditional pdfs to improve our knowledge of the object's most likely class membership. The *a posteriori probability* that the object belongs to class  $i$  is given by Bayes' theorem; that is,

$$P(C_i | x) = \frac{P(C_i)p(x | C_i)}{p(x)} \quad (9.2)$$

where  $P(C_i)$  is the a priori probability of class  $i$ ,  $p(x | C_i)$  is the pdf of the feature,  $x$ , for class  $i$ , and

$$p(x) = \sum_{i=1}^N p(x | C_i)P(C_i) \quad (9.3)$$

is the normalization factor required to make the set of a posteriori probabilities sum to unity. Bayes' theorem, then, allows us to combine the a priori probabilities of class membership with the measurement data and the class-specific pdf, to compute, for each class, the probability that the measured object belongs to that class. Given this information, we might choose to assign each object to its most likely class.

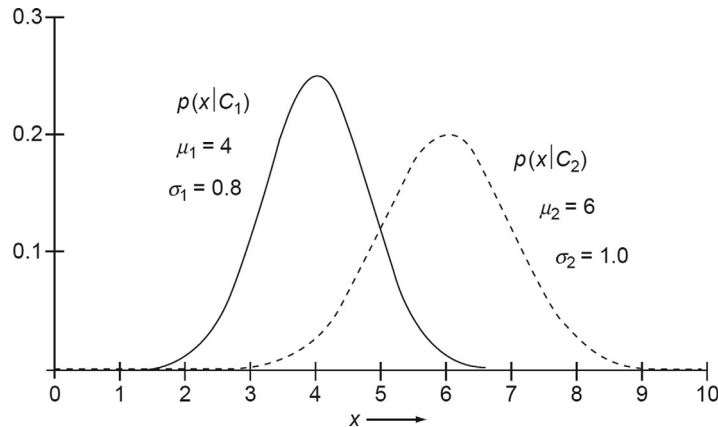


## 9.3 The Single-Feature, Two-Class Case

To illustrate the classification process, we first consider the simple case where two types of objects must be sorted on the basis of a single measurement. For this example, assume we are attempting to separate abnormal from normal cells on the basis of nuclear diameter alone. This means that the cells encountered belong either to class 1 (normal) or to class 2 (abnormal). For each cell, we measure one property, nuclear diameter, and this is the feature we call  $x$ .

It may be that the pdf of the diameter measurement,  $x$ , is already known for one or both classes of cells. If not, we would have to estimate it by measuring a large number of normal and abnormal cells and plotting histograms of their nuclear diameters. After normalization to unit area, and perhaps some smoothing, these histograms can be taken as estimates of the corresponding pdfs. If the histogram fits the Gaussian form, to a reasonable approximation, we can compute the mean,  $\mu$ , and variance,  $\sigma$ , and use the parametric representation for the normal distribution

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (9.4)$$



**Fig. 9.1** Probability density functions for a two-class problem.

There are other standard statistical distributions for the pdf that might fit the histograms if the Gaussian form does not. If we use the Gaussian, then only the mean and variance are required to completely specify the pdf for a class.

### 9.3.1 A Priori Probabilities

The *a priori* probabilities represent our knowledge about an object before it has been measured. In this example, we assume that an unmeasured cell has a nine-to-one chance of being normal (Eq. 9.1).

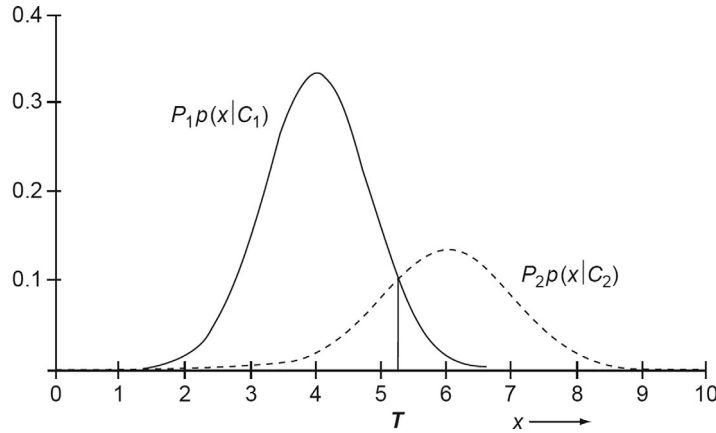
### 9.3.2 Conditional Probabilities

Fig. 9.1 shows what the two pdfs might look like. We denote the conditional pdf for normal cell diameter as  $p(x|C_1)$ , which can be read as “the probability that diameter  $x$  will occur, given that the object belongs to class 1.” Similarly,  $p(x|C_2)$  is the probability of diameter  $x$  occurring, given class 2 (abnormal).

If we scale each of the pdfs by the a priori probability of its class, as in Fig. 9.2, we get a better picture of the error situation. We could establish a decision rule by setting a threshold value,  $T$ , on the nuclear diameter and classifying cells normal if they fall below that value and abnormal if they fall above it. The area under the dotted curve, to the left of the threshold, is proportional to the probability of calling an abnormal cell normal. Similarly, the area under the solid curve, to the right of the threshold, is proportional to the probability of misclassifying a normal cell.

### 9.3.3 Bayes’ Theorem

Before an object has been measured, our knowledge of it consists merely of the a priori probabilities of class membership. After measurement, however, we can combine the



**Fig. 9.2** Probability density functions for a two-class problem, scaled by the prior probabilities.

measurement with the conditional pdfs to improve our knowledge of the object's class membership. After measurement, the a posteriori probability that the object belongs to class  $i$  is given by Bayes' theorem [2–7], that is,

$$P(C_i | x) = \frac{P(C_i)p(x|C_i)}{p(x)} \quad (9.5)$$

where  $P(C_i)$  is the a priori probability of class  $i$ ,  $p(x|C_i)$  is the pdf of the feature,  $x$ , for class  $i$ , and

$$p(x) = \sum_{i=1}^N p(x|C_i)P(C_i) \quad (9.6)$$

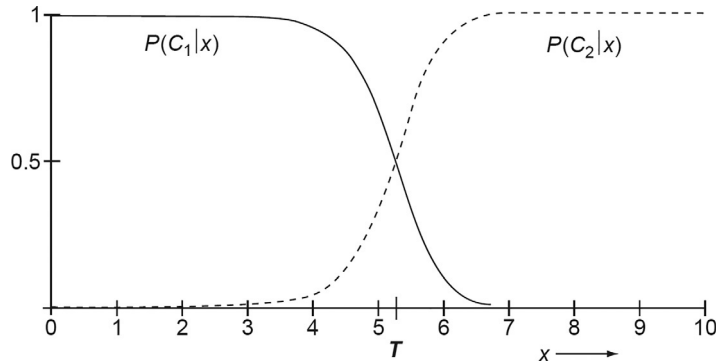
is the required normalization factor.

Bayes' theorem, then, allows us to combine the a priori probabilities of class membership with the measurement and the class-specific pdf, to compute, for each class, the probability that the measured object belongs to that class. Fig. 9.3 shows the *a posteriori* probabilities for this example. For any nuclear diameter,  $x$ , the solid curve gives the probability that a cell having that diameter belongs to class 1. The dotted curve gives the probability that the cell belongs to class 2.

In our cell-sorting example, we would assign the object to class 1 (i.e., we would call it normal) if

$$P(C_1 | x) > P(C_2 | x) \quad (9.7)$$

and assign it to class 2 (abnormal) otherwise. At the decision threshold,  $T$ , where equality holds in Eq. (9.7), we may assign arbitrarily. The classifier defined by this decision rule is



**Fig. 9.3** A posteriori probabilities for the two-class problem.

called a maximum-likelihood classifier because it maximizes the probability of a correct assignment.

Note that, in Fig. 9.3, cells with nuclear diameter less than  $4\mu\text{m}$  can be confidently assigned to class 1, while cells larger than  $6\mu\text{m}$  easily can be called abnormal. It is for cells with nuclear diameter near  $5\mu\text{m}$  that one would expect misclassification errors to occur.



## 9.4 The Three-Feature, Three-Class Case

We next consider the case where there are three types of objects, and three measurements are made on each. The particular example we use here is the classification of pixels in a color image. The three measurements made on each pixel are the red, green, and blue intensity values. We assume that each pixel belongs to one of three classes: the interior of a normal cell, the interior of an abnormal cell, or the background.

Each pixel can be considered to represent a point in a three-dimensional color space. Thus each of the different-colored objects in the image will correspond to a “cloud” of points in color space, and classification becomes the task of isolating these clusters. More specifically, we wish to define a set of decision surfaces that carve up the space into three disjoint regions, one for each class.

### 9.4.1 The Bayes Classifier

One straightforward and quite powerful approach is the use of the Bayes maximum-likelihood classifier. It generates second-order surfaces that partition the color space into disjoint regions, one for each object type. Assuming Gaussian distributions for the clusters of points in color space, the Bayes classifier maximizes the probability that each pixel will be assigned correctly.

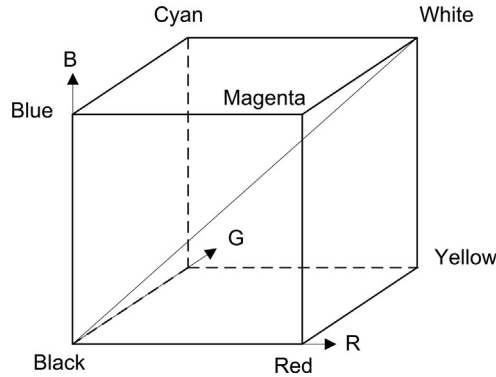


Fig. 9.4 The three-dimensional RGB color space.

We illustrate the use of the Bayes classifier with a simple example. We assume that a three-color RGB system (Red, Green, Blue) is used to digitize a fluorescent microscope image. The vector of gray levels at a single pixel location is

$$\mathbf{x} = [x_j] = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

where

$$j = \begin{cases} 1 \Rightarrow \text{red} \\ 2 \Rightarrow \text{green} \\ 3 \Rightarrow \text{blue} \end{cases} \quad (9.8)$$

We further assume the images contain two types of objects, normal and abnormal cells. Both types of cells bind the blue fluor; the normals also bind the green fluor, but the abnormal cells pick up the red fluor instead. One would expect a three-dimensional histogram (scatterplot) of the color space to show three clusters of points, one each for background, normal cells, and abnormal cells. Since the background is dark, its cluster will fall near the origin of the color space. The normal cells will give rise to a cloud of points near the cyan corner, while the abnormal cells will fall near the magenta corner, as in Fig. 9.4.

#### 9.4.1.1 Prior Probabilities

Let us assume that the area of a typical image is 90% occupied by background and 10% by cells. Further assume that, overall, only 10% of the cells are abnormal. Thus the vector of prior probabilities, where  $P_i = P\{\text{pixel belongs to class } i\}$ , is

$$\mathbf{P} = \begin{bmatrix} 0.90 \\ 0.09 \\ 0.01 \end{bmatrix}$$

where

$$i = \begin{cases} 1 & \Rightarrow \text{background} \\ 2 & \Rightarrow \text{normal} \\ 3 & \Rightarrow \text{abnormal} \end{cases} \quad (9.9)$$

#### 9.4.1.2 Classifier Training

The first step is to train the classifier to recognize the three types of pixels. For this we require a training set containing pixels that are known to fall in the background, inside normal, and inside abnormal cells. It is the statistics of these training set pixels that constitute the knowledge that the classifier has about the problem. Estimating these statistics is the process of classifier training.

#### 9.4.1.3 The Mean Vector

Using the training set, we calculate, for each class,  $i$ , the mean pixel brightness in each color,  $j$ . That is,

$$\mu_{ij} = \frac{1}{N_i} \sum_{k=1}^{N_i} x_{ijk} \quad (9.10)$$

where  $N_i$  is the number of pixels in class  $i$ , and  $x_{ijk}$  is the value, in color  $j$ , of pixel  $k$  in class  $i$  of the training set. The mean vector for class  $i$  is

$$\boldsymbol{\mu}_i = \begin{bmatrix} \mu_{i1} \\ \mu_{i2} \\ \mu_{i3} \end{bmatrix} \quad (9.11)$$

This vector, the mean vector of the training set, is an estimate of the mean of the entire population of class  $i$  pixels. If the training set is both adequately large and representative of the population, it will be a good estimate. Otherwise, it will be a poor estimate, and classifier accuracy will suffer, as will our ability to predict how well it will work.

#### 9.4.1.4 Covariance

We calculate the covariance matrix for each class [2,5,8] as

$$\mathbf{S}_{ij_1, j_2} = \frac{1}{N_i - 1} \left[ \sum_{k=1}^{N_i} x_{ij_1 k} x_{ij_2 k} - N_i \mu_{ij_1} \mu_{ij_2} \right] \quad (9.12)$$

#### 9.4.1.5 Variance and Standard Deviation

The diagonal elements of a covariance matrix are the variances of the features, for that class. The variance is the square of the standard deviation. That is,

$$\sigma_{ij}^2 = S_{ijj} \quad \text{and} \quad \sigma_{ij} = \sqrt{S_{ijj}} \quad (9.13)$$

#### 9.4.1.6 Correlation

From the covariance matrix we can compute the correlation matrix for each class [2,5,8]. For class  $i$ , this is

$$\mathbf{C}_{ij_1j_2} = \frac{\mathbf{S}_{ij_1j_2}}{\sigma_{ij_1}\sigma_{ij_2}} \quad (9.14)$$

The elements of the correlation matrix are bounded by  $\pm 1$ . A correlation of 1.0 means the corresponding two features are proportional to each other. A correlation of  $-1$  means each is proportional to the negative of the other. A correlation of 0 means the two features are uncorrelated. Using highly correlated features is not only redundant, but it can actually degrade classifier accuracy. One can either combine highly correlated features (by averaging, for example), or simply discard all but one of them. In this example we assume the three features are not highly correlated.

#### 9.4.1.7 The pdf

The probability density function for class  $i$  is

$$p_i(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{S}_i|}} \exp \left[ -\frac{1}{2} |(\mathbf{x} - \boldsymbol{\mu}_i)^T \mathbf{S}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)| \right] \quad (9.15)$$

where  $\mathbf{x}$  is the vector of RGB values for a pixel, as in Eq. (9.8), and  $n=3$  is the number of features in use. The superscript  $T$  indicates the matrix transpose, and  $\mathbf{S}_i^{-1}$  is the inverse of the covariance matrix for class  $i$ . Eq. (9.15) is the multidimensional generalization of Eq. (9.3).

#### 9.4.1.8 Classification

Each class of pixels is now characterized by its prior probability, mean vector, and covariance matrix. The classifier has been trained. We now have enough statistical information about the problem to begin classifying pixels. By Bayes' rule, the likelihood that an unknown pixel having color vector  $\mathbf{x}$  belongs to class  $i$  is

$$L_i = P_i \frac{pdf_i(\mathbf{x})}{p(\mathbf{x})} \quad \text{where} \quad p(\mathbf{x}) = pdf_1(\mathbf{x}) + pdf_2(\mathbf{x}) + pdf_3(\mathbf{x}) \quad (9.16)$$

or

$$L_i = \frac{P_i}{p(\mathbf{x}) \sqrt{(2\pi)^3 |\mathbf{S}_i|}} \exp \left[ -\frac{1}{2} |(\mathbf{x} - \boldsymbol{\mu}_i)^T \mathbf{S}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)| \right] \quad (9.17)$$



Thus we can compute the three likelihoods (one for each class) and assign the pixel to the most likely (largest likelihood) class. If the pdfs are, as we have assumed, Gaussian (normal) density functions, then no other partitioning of color space will result in lower overall error rates [2].

#### 9.4.1.9 Log Likelihoods

Since the logarithm is a monotonic function, we can take the log of both sides of Eq. (9.17) and use the resulting value for classification purposes. Eq. (9.17) then becomes

$$\ln(L_i) = \ln(P_i) - \frac{1}{2} |(\mathbf{x} - \boldsymbol{\mu}_i)^T \mathbf{S}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)| - \frac{3}{2} \ln(2\pi) - \frac{1}{2} \ln(|\mathbf{S}_i|) - \ln(p(\mathbf{x})) \quad (9.18)$$

The third and fifth terms are constants and, for classification purposes, can be omitted, leaving

$$LL_i = \ln(P_i) - \frac{1}{2} |(\mathbf{x} - \boldsymbol{\mu}_i)^T \mathbf{S}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)| - \frac{1}{2} \ln(|\mathbf{S}_i|) \quad (9.19)$$

The first term accounts for the prior probabilities, while the third term accounts for the within-class scattering of the features. The larger this variation, the less confidently one can assign the pixel to that class. The second term is the square of the *Mahalanobis distance*. It represents the variance-normalized distance, in feature space, from the unknown color to the class mean.

#### 9.4.1.10 The Mahalanobis Distance Classifier

We can simplify the Bayes' classifier further by computing only the Mahalanobis distance from an unknown pixel to each class mean

$$D_i = \sqrt{\frac{1}{2} |(\mathbf{x} - \boldsymbol{\mu}_i)^T \mathbf{S}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)|} \quad (9.20)$$

and assigning each pixel to the class having the nearest mean. This results in what is called a *Mahalanobis distance classifier*. It corresponds to the special case where the prior probabilities are equal among the classes, and likewise for the within-class variations. This distance classifier is sometimes used when the prior probabilities are unknown and the covariance matrix cannot be estimated accurately, due to limited training set size. Distance in feature space can be computed in other ways as well (e.g., Euclidean distance), and this gives rise to other types of distance classifiers.

#### 9.4.1.11 Uncorrelated features

While 30 or so pixels per class in the training set might be sufficient to estimate the feature means and variances, considerably more might be required to estimate the off-diagonal elements of the covariance matrix. If the training set is necessarily small, one solution is to set the off-diagonal elements to zero. This is equivalent to assuming that the features are

uncorrelated. Under pressure of limited training set size, this can yield a more stable and better performing classifier than one designed around inadequately estimated covariances. Using a distance classifier, which automatically assumes uncorrelated features, results in an even simpler classifier. Since there are so many pixels in an image, however, it is possible to accumulate quite large training sets, and covariances could be estimated quite accurately in this three-feature, three-class example.

### 9.4.2 A Numerical Example

To illustrate the operation of the three-class Bayes classifier we include a numerical example using six pixels from each class in the training set. While this is hopelessly inadequate for any real case, it serves to illustrate the calculations. This example will permit readers who choose to implement a Bayes classifier to check their implementation for numerical accuracy.

Assume the training set is

$$\begin{bmatrix} 33 & 31 & 46 & 57 & 18 \\ 42 & 10 & 24 & 38 & 56 \\ 62 & 50 & 34 & 21 & 33 \end{bmatrix} \begin{bmatrix} 6 & 28 & 47 & 21 & 58 \\ 96 & 116 & 126 & 84 & 73 \\ 70 & 82 & 96 & 117 & 90 \end{bmatrix} \begin{bmatrix} 78 & 115 & 122 & 76 & 134 \\ 12 & 52 & 34 & 70 & 22 \\ 81 & 100 & 146 & 78 & 70 \end{bmatrix} \quad (9.21)$$

for the RGB color values of five pixels each of background, normal cells, and abnormal cells, respectively. The mean vectors for the three classes are then

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 37 \\ 34 \\ 40 \end{bmatrix} \quad \boldsymbol{\mu}_2 = \begin{bmatrix} 32 \\ 99 \\ 91 \end{bmatrix} \quad \boldsymbol{\mu}_3 = \begin{bmatrix} 105 \\ 38 \\ 95 \end{bmatrix} \quad (9.22)$$

The covariance matrices are.

$$\begin{aligned} \mathbf{S}_1 &= \begin{bmatrix} 223.5 & -79 & -112.25 \\ -79 & 310 & -58.5 \\ -112.25 & -58.5 & 257.5 \end{bmatrix} & \mathbf{S}_2 &= \begin{bmatrix} 428.5 & -24 & 86.25 \\ -24 & 482 & -79.75 \\ 86.25 & -79.75 & 306 \end{bmatrix} \\ \mathbf{S}_3 &= \begin{bmatrix} 700 & -154.5 & 265.75 \\ -154.5 & 542 & 21.5 \\ 265.75 & 21.5 & 934 \end{bmatrix} \end{aligned} \quad (9.23)$$

from which the standard deviations (square roots of diagonal elements) are

$$\boldsymbol{\sigma}_1 = \begin{bmatrix} 15.0 \\ 17.6 \\ 16.0 \end{bmatrix} \quad \boldsymbol{\sigma}_2 = \begin{bmatrix} 20.7 \\ 22.0 \\ 17.5 \end{bmatrix} \quad \boldsymbol{\sigma}_3 = \begin{bmatrix} 26.5 \\ 23.3 \\ 30.6 \end{bmatrix} \quad (9.24)$$

the correlation matrices (Eq. 9.14) are

$$\begin{aligned} \mathbf{C}_1 &= \begin{bmatrix} 1 & -.300 & -.468 \\ -.300 & 1 & -.207 \\ -.468 & -.207 & 1 \end{bmatrix} & \mathbf{C}_2 &= \begin{bmatrix} 1 & -.053 & .238 \\ -.053 & 1 & -.208 \\ .238 & -.208 & 1 \end{bmatrix} \\ \mathbf{C}_3 &= \begin{bmatrix} 1 & -.251 & .329 \\ -.251 & 1 & .030 \\ .329 & .030 & 1 \end{bmatrix} \end{aligned} \quad (9.25)$$

and the determinants of the covariance matrices are

$$|\mathbf{S}_1| = 1.053 \times 10^7 \quad |\mathbf{S}_2| = 5.704 \times 10^7 \quad |\mathbf{S}_3| = 2.917 \times 10^8 \quad (9.26)$$

Suppose we have an unknown pixel having color vector

$$\mathbf{x} = \begin{bmatrix} 38 \\ 80 \\ 78 \end{bmatrix} \quad (9.27)$$

The three likelihoods (from Eq. 9.17) are

$$\mathbf{L} = \begin{bmatrix} 0.000015 \\ 0.088083 \\ 0.000213 \end{bmatrix} \quad (9.28)$$

and the pixel would be assigned to class 2. The log likelihoods, with constant terms dropped (Eq. 9.19), are

$$\mathbf{LL} = \begin{bmatrix} -20.9 \\ -12.3 \\ -18.3 \end{bmatrix} \quad (9.29)$$

Again, the pixel would be assigned to class 2. The Mahalanobis distances to the class means (Eq. 9.20) are

$$\mathbf{D} = \begin{bmatrix} 3.57 \\ 0.97 \\ 1.99 \end{bmatrix} \quad (9.30)$$

This pixel would be assigned to class 2 by a distance classifier as well, since it is the closest. Thus all three classifiers would assign this pixel to class 2.



## 9.5 Classifier Performance

Once a classifier has been designed and trained, it is necessary to test it to establish its accuracy. This is usually done by classifying a test set of known objects and tabulating the number of errors. If the test set is the same as the training set, the performance estimates

will be optimistically biased. If it includes none of the training data, they will be pessimistically biased. If the test set is large, the effects of this bias will be slight. If the number of available preclassified objects is small, one can use the “round robin” or “leave one out” method. Here the classifier is trained on all but one of the objects and tested on the remaining object. This process is repeated until every object has been used for testing. The results of the various experiments are then averaged together to estimate the error rates.

### 9.5.1 The Confusion Matrix

A very handy tool for specifying the accuracy of a multiclass classifier is the confusion matrix. This is an  $N \times N$  matrix,  $\mathbf{C}$ , where  $N$  is the number of classes. The columns of  $\mathbf{C}$  correspond to the classes that objects actually belong to, while the rows of  $\mathbf{C}$  correspond to the classes to which objects can be assigned. Thus the element  $c_{ij}$  corresponds to the situation of an object that belongs to class  $j$  being assigned to class  $i$ . That is, true class =  $j$ , assigned class =  $i$ .

There are several ways one can set up the confusion matrix to summarize the results of a classifier test. A *raw confusion matrix* results when the value of each element is simply set to the number of times the corresponding situation occurred in a particular test of the classifier. Other values, however, may be more useful. For example, *sensitivity* is defined, for each class, as the probability that an object belonging to that class will be correctly assigned. We obtain an estimate of the sensitivity matrix by dividing the raw confusion matrix elements by the total number of objects in the true class. Each element, then, shows what fraction of the objects that actually belong to that class are assigned to that class. The columns of the sensitivity matrix sum to unity.

*Specificity*, for a particular class, is defined as 1 minus the ratio of the number of objects incorrectly assigned to that class divided by the total number of objects that are not in that class. Specificity is seldom a very useful parameter because it almost always takes on values quite close to unity. A more useful specification is *positive predictive value* (PPV). This is the probability that an object assigned to a class actually belongs to that class. The PPV matrix is estimated by dividing the elements in each row of the raw confusion matrix by the total number of objects assigned to that class. In this case the rows sum to unity.

When analyzing the performance of a classifier, one finds the sensitivity matrix and the PPV matrix to be very useful. In short, the sensitivity matrix tells you where each type of object is going, while the PPV matrix tells you what is going into each of the classes. Studying these two matrices can yield considerable insight into the strengths and weaknesses of a particular classifier.

As an example, consider the sensitivity matrix and PPV matrix shown in Fig. 9.5. They correspond to the three-class pixel classifier example discussed previously. We see from Fig. 9.5 that this classifier has two problems. First, 9% of the pixels in abnormal

Sens	True Class			
Assigned Class		1	2	3
	1	98%	2%	1%
	2	1%	96%	9%
	3	1%	2%	90%

PPV	True Class			
Assigned Class		1	2	3
	1	98%	1%	1%
	2	2%	96%	2%
	3	2%	7%	91%

Fig. 9.5 Confusion matrices.

cells are being called normal. This could lead to abnormal cells being missed. Second, 7% of the pixels that are called abnormal are actually normal. This could lead to false positive errors. We would conclude that this classifier needs to be improved in its ability to discriminate between pixels in the normal and abnormal classes.



## 9.6 Bayes Risk

A useful generalization of the maximum likelihood Bayes classifier allows one to bias the classifier so as to reduce the occurrence of certain costly types of misclassification errors, in exchange for making more of other, less serious, errors [2,5]. In our three-class example, there are nine elements in the confusion matrix. Three correspond to correct decisions, while the remaining six represent different types of errors. Suppose that it is considered to be more serious to confuse a pixel from an abnormal cell with one from a normal cell, or to call a normal pixel abnormal, than it is to make any of the four other possible errors. The minimum Bayes risk classifier allows us to account for this difference in the severity of errors.

### 9.6.1 The Minimum-Risk Classifier

We begin by setting up a *cost matrix*. It has the same format as the confusion matrix, except that its elements represent the “cost” of having that situation occur. Specifically,  $C_{ij}$  represents the cost of assigning to class  $j$  a pixel that actually belongs to class  $i$ . If  $i=j$ , this corresponds to a correct classification, and a cost of zero might be assigned to those elements. If all misclassification errors are equally unfortunate, then ones could be placed in all of the off-diagonal elements. In this case a maximum likelihood classifier results. However, larger values can be assigned to cost matrix elements that correspond to the more serious errors. One possible cost matrix for our three-class pixel classifier example is

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 4 \\ 1 & 2 & 0 \end{bmatrix} \quad (9.31)$$

Here we have said (1) that correct classifications cost nothing, (2) that calling an abnormal pixel normal has a cost of 4, and (3) that calling a normal pixel abnormal has a cost of 2, while (4) the four remaining errors have unit cost. Note that the actual cost values are unitless, and their values are relevant only in relation to one another.

Given a cost matrix, we can set up the Bayes classifier to minimize its long-term cost of operation. The Bayes risk for assignment to a particular class is the cost of each outcome times the likelihood of that outcome, summed over all possible assignments to that class. It can be computed, for the unknown pixel of Eq. (9.27), as

$$R_i = \sum_{j=1}^3 C_{i,j} L_j \quad \mathbf{R} = \begin{bmatrix} 0.0883 \\ 0.0009 \\ 0.1762 \end{bmatrix} \quad (9.32)$$

In this example the unknown pixel would be assigned to Class 2 because the Bayes risk is lowest there. We would expect this classifier to perform better in practice than the maximum likelihood classifier since the two most serious error rates have been reduced (but at the expense of more of the less serious errors).



## 9.7 Relationships Among Bayes Classifiers

Note that the minimum-risk classifier is the most general Bayes classifier. The maximum-likelihood classifier is a special case of that, namely when all costs are set to be equal. Further reductions in generality result when the *a priori* probabilities are assumed to be equal, or the off-diagonal covariances are assumed to be 0. The minimum-distance classifier is a further restricted special case that results when both the prior probabilities and the within-class variation are ignored.

In this numerical example all three forms of the Bayes classifier, the minimum-risk, maximum likelihood, and minimum distance classifiers, assigned the unknown pixel to the same class. This will not be the case in general, as objects that fall near the decision boundaries will be assigned differently by the different classifiers. Objects that fall near their class mean will be classified correctly by any of the classifiers. It is the outliers, those that defy the assumption of Gaussian statistics, that make classification difficult.



## 9.8 The Choice of a Classifier

If a considerable amount of training data is available, one can simply estimate the required pdfs and use those estimates in the classification process. Such classifiers are called “nonparametric.” Often, however, it is difficult to obtain large numbers of preclassified objects. In this case one can assume a particular functional form for the pdf (the Gaussian, for example) and use the training data only to estimate the parameters of the pdf. This

gives rise to a “parametric classifier.” Considerably less training data is then required for training, and one benefits from the powerful mathematics that have been developed for those cases.

It is often useful to begin a classifier design effort with a classical Bayes classifier, as described previously. At the very least, this establishes a baseline of performance against which other types of classifiers can be tested and evaluated. Further, if the underlying assumptions are met, the Bayes classifier, assuming Gaussian statistics, may well perform as well as or better than any other classifier.

Problems arise when the underlying pdfs do not fit the assumed functional form. The classifier’s performance, and one’s predictions of its accuracy, are only as good as the underlying assumptions. It is rather difficult to prove that a population of objects actually fits, for example, a Gaussian (normal) distribution. As a rule of thumb, if the marginal distributions (one-dimensional histograms) are unimodal and symmetrical, one can often assume multivariate Gaussian statistics (although there are no guarantees, and notable exceptions exist). Even so, the Gaussian pdf decays quite rapidly for values distant from the mean. Any occurrence of data beyond  $4\sigma$ , for example, is essentially impossible. For actual data, however, this may not match reality. In such cases one can use a “heavy-tailed” pdf that decays to zero more slowly than the Gaussian. Such distributions include the log-normal, Cauchy, Levy, power-law, and student’s  $t$ -distributions [9].

### 9.8.1 Subclassing

Even if the marginal distributions are not unimodal and symmetrical, one can do things to make them unimodal and symmetrical. If the feature histograms of a class are multimodal (i.e., they have two or more peaks) one would suspect that two or more distinct subclasses exist within the class. By subdividing the class, one can often achieve unimodal pdfs, but with a larger number of classes. This is a fair trade if it justifies the assumption of Gaussian statistics. An example is shown in Fig. 9.6. This is the single-feature, two-class example used earlier in this chapter. Here the nuclear diameter histogram of normal cells is unimodal, but that of the abnormal cells is bimodal.

If we reexamine the training set we may find that two distinct populations of cells exist within the abnormal class. In this case we can establish a new class called “atypical” and assign some of the previously “abnormal” cells to it, based on their morphology. The result is a three-class problem where the feature histograms are unimodal (Fig. 9.7). This is a very profitable trade if it permits the assumption of Gaussian statistics.

### 9.8.2 Feature Normalization

An asymmetric or non-Gaussian pdf often can be corrected by a suitably designed nonlinear transformation of the feature values [5]. A feature histogram, normalized to

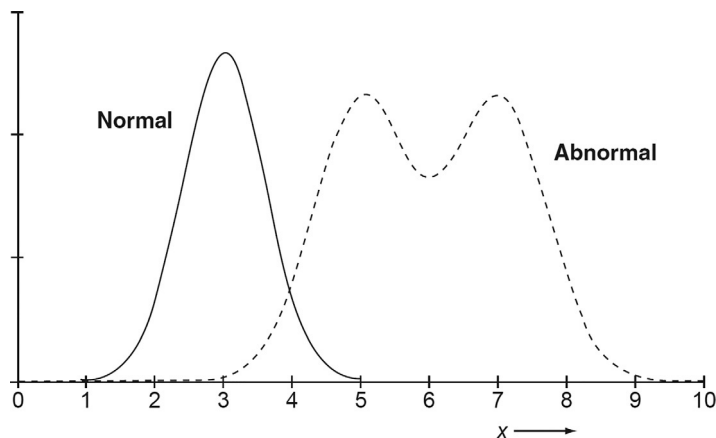


Fig. 9.6 A two-class problem with a bimodal pdf.

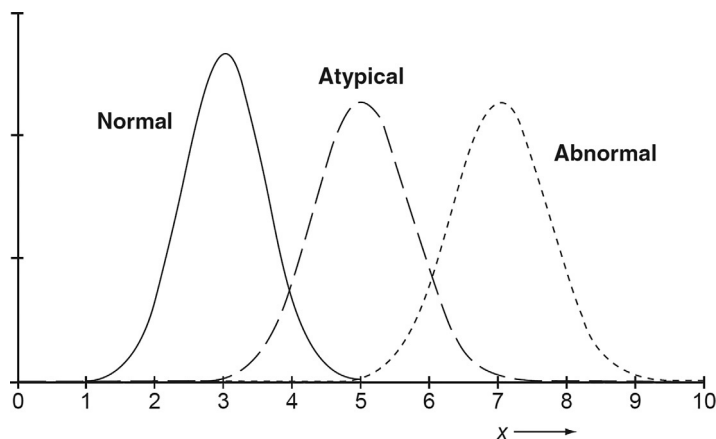


Fig. 9.7 The use of subclassing to eliminate a bimodal pdf. The result is a three-class problem with unimodal pdfs.

unit area, is an estimate of the pdf of that feature. The *cumulative distribution function* (CDF) is the integral of the pdf, that is,

$$P(x) = \int_0^x p(u) du = \frac{1}{A_0} \int_0^x H(u) du \quad (9.33)$$

where  $H(x)$  is the histogram,  $p(x)$  is the pdf,  $P(x)$  is the CDF, and  $A_0$  is the area under the histogram. The CDF is quite a well-behaved function, increasing monotonically from 0



to 1. If it is used to transform feature  $x$  into a new feature  $y$ , that is,  $y = P(x)$ , then the feature  $y$  will have a flat histogram (uniform distribution). As a special case, the CDF corresponding to a Gaussian pdf will transform a feature with a Gaussian pdf into one with a flat histogram. It follows that the inverse function of the Gaussian CDF will transform a feature with a flat histogram into one with a Gaussian histogram. Thus we can transform a feature so that it has a Gaussian histogram by concatenating two nonlinear transformations:

$$y = P_2(P_1(x)) \quad (9.34)$$

where  $P_1(x)$  is the CDF of the feature, and  $P_2(x)$  is the inverse of the CDF of a Gaussian.  $P_1(x)$  makes the pdf uniform, and  $P_2(x)$  makes it Gaussian.

Note that, in a multifeature classification problem, transforming the individual features to have Gaussian pdfs does not guarantee that the overall multivariate pdf will be Gaussian. As a practical matter, however, such a transformation can make the assumption of Gaussian statistics much less of an approximation. Feature normalization works best in the commonly occurring case where the raw feature histograms are not radically different from a Gaussian to begin with.



## 9.9 Nonparametric Classifiers

If the functional form of the pdfs of the classes is unknown, then the parametric approach cannot be used. In this case one must estimate the pdfs directly from the training data [2]. This generally requires a much larger training set. However, the maximum likelihood and minimum risk formulations still apply.

The basic problem of nonparametric pdf estimation is straightforward: given a set of training samples, model the pdf of the data without making any assumptions about the functional form of the distribution. Suppose we have  $N_j$  training samples from class  $j$ . To estimate the pdf, the  $L$ -dimensional feature space can be partitioned into small regions that are  $L$ -dimensional hypercubes, with volume  $V = h^L$ , where  $h$  is the bin size. Let  $\mathbf{R}$  be such a region and  $k_j$  be the number of samples from class  $j$  falling into  $\mathbf{R}$ , with  $k_j \leq N_j$ . A straightforward estimate of the pdf can be expressed as:

$$\hat{p}(\mathbf{x}|C_j) = \frac{k_j/N_j}{V} \quad (9.35)$$

This basic estimator corresponds to an  $L$ -dimensional histogram. Essentially, the feature space is divided into a finite number of hypercube bins and the probability density at the center of each hypercube is estimated by the fraction of samples in the training set that fall into that hypercube bin. The bin size,  $h$ , and the starting position of the first bin are two “parameters” that determine the shape of the histogram.

The histogram is a simple and effective form of pdf estimation, but it has several drawbacks. The shape of the estimate is affected by both the bin size and the starting point of the bins. The discontinuities of the estimate are not due to the underlying probability density, but are caused by the particular choice of bin locations. A more serious problem is the curse of dimensionality, since the number of bins grows exponentially with the number of dimensions. In high dimensions we would require a very large number of training samples, or else most of the bins would be empty.

A more advanced nonparametric pdf estimation method makes use of the so-called “Parzen window” for a unit hypercube centered at the origin:

$$\psi(\mathbf{v}) = \begin{cases} 1, & |v_q| \leq \frac{1}{2}, \quad q = 1, 2, \dots, L \\ 0, & \text{otherwise} \end{cases} \quad (9.36)$$

Hence  $\psi((\mathbf{x} - \mathbf{x}_i)/h)$  equals to unity if  $\mathbf{x}_i$  is within the hypercube at  $\mathbf{x}$  and is zero elsewhere. The number of training samples that fall into this hypercube can be written as

$$k = \sum_{i=1}^{N_j} \psi((\mathbf{x} - \mathbf{x}_i)/h)$$

Substituting it in Eq. (9.35), we have

$$\hat{p}(\mathbf{x}|C_j) = \frac{1}{N_j} \sum_{i=1}^{N_j} \frac{1}{V} \psi((\mathbf{x} - \mathbf{x}_i)/h) \quad (9.37)$$

Notice that the Parzen window density estimate resembles the histogram, except that the hypercube locations are determined by the training sample points rather than by the histogram bins. The expression in Eq. (9.37) shows that the estimate  $\hat{p}(\mathbf{x}|C_j)$  is made of an average of functions of  $\mathbf{x}$  and the samples  $\mathbf{x}_i$ . Based on this formulation, we can adopt two basic approaches. We can choose a fixed value of  $k$  and determine the corresponding volume  $V$  from the training samples. This gives rise to the so-called “ $k$  nearest-neighbor” (kNN) approach. Alternatively, we can also choose a fixed value of the volume  $V$  and determine  $k$  from the samples. This leads to the methods commonly referred to as “kernel density estimation” (KDE).

### 9.9.1 Nearest-Neighbor Classifiers

The kNN is a very intuitive nonparametric approach that classifies unknown objects based on their similarity to the samples in the training set. For an unknown object  $\mathbf{x}$ , it finds the  $k$  “nearest” samples  $\mathbf{x}_i$  in the training set and assigns  $\mathbf{x}$  to the class that appears most frequently among them. A great advantage of the kNN approach is that no estimation of the pdf is required since the function is only approximated locally, and all computation is deferred until the classification stage. However, the disadvantages are the memory

requirement to store training samples and the computational complexity required to search for the  $k$  nearest samples during the classification of each unknown object.

On the other hand, with the KDE methods one can generalize the hypercube Parzen window with a smooth nonnegative kernel function  $\psi(\mathbf{x})$  that satisfies the condition  $\int \psi(\mathbf{x}) d\mathbf{x} = 1$ . Just as the Parzen window estimate can be considered a sum of boxes centered at the samples, the smooth kernel estimate is a sum of “bumps” placed at the samples, and the kernel function determines the shape of the bumps. Usually,  $\psi(\mathbf{x})$  is chosen to be a radially symmetric, unimodal pdf, such as the multivariate Gaussian. The kernel function is used essentially for interpolation, and each sample contributes to the estimate according to its distance from  $\mathbf{x}$ . It can be shown [2] that both of these approaches converge to the true pdf as  $N_j \rightarrow \infty$ , that is,  $\lim_{N_j \rightarrow \infty} \hat{p}(\mathbf{x}|C_j) = p(\mathbf{x}|C_j)$ , provided that  $V$  shrinks with  $N_j$ , and  $k$  grows with  $N_j$  properly.

For applications with high dimensional feature space, the curse of dimensionality affects all classifiers without exception. The available training samples are usually inadequate to obtain an accurate estimation in these cases. One solution to the problem is to choose independent features so that  $p(\mathbf{x}|C_j) = \prod_{i=1}^L p(x_i|C_j)$ , by mapping the original features using a proper subspace transformation such as the independent component analysis (ICA) [10]. Thus the problem of estimating an  $L$ -dimensional multivariate pdf  $p(\mathbf{x}|C_j)$  is collapsed to that of estimating multiple one-dimensional univariate pdfs  $p(x_i|C_j)$ ,  $i = 1, 2, \dots, L$ . This way, the training set size requirement becomes much easier to meet.



## 9.10 Feature Selection

Ideally one would prefer to use a rather small number of highly discriminating, uncorrelated features. Increasing the number of features increases the dimensionality, and hence the volume of the feature space [11–13]. This, in turn, increases the requirements for training set and test set size [2–6]. Adding features that have poor discrimination or are highly correlated with the other features can actually degrade classifier performance [2].

### 9.10.1 Feature Reduction

There are well-developed mathematical procedures for reducing a large number of features down to a smaller number without severely limiting the discriminating power of the set. *Principal component analysis* (PCA) [8] and *linear discriminant analysis* (LDA), also known as *Fisher discriminant analysis* (FDA) [14], discussed shortly, are among the best-known subspace methods that can be used to reduce the dimensionality of the feature space. Both generate a new set of features, each of which is a linear combination of the original features. In both cases the new features are ranked so that one can select only a few of the most useful ones, thereby reducing the number of features.

### 9.10.1.1 Principal Component Analysis

In general, suppose  $\mathbf{x}$  is an  $L$ -dimensional feature vector and  $\mathbf{W}$  is a  $Q \times L$  matrix; then

$$y_i = \sum_{j=1}^L w_{i,j} x_j, \quad i = 1, 2, \dots, Q \quad \text{or} \quad \mathbf{y} = \mathbf{W}\mathbf{x} \quad (9.38)$$

defines a linear transformation of the vector  $\mathbf{x}$ . The result is a  $Q \times 1$  vector  $\mathbf{y}$ , which is a projection of  $\mathbf{x}$  onto a linear subspace defined by the transform matrix  $\mathbf{W}$ . Each element  $y_i$  is the inner product of a basis vector, which is the  $i$ th row of  $\mathbf{W}$ , with the input vector  $\mathbf{x}$ .

Consider a set of  $L$ -dimensional sample feature vectors  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M$ . Without loss of generality, we can assume these are zero-mean vectors since we can always redefine  $\mathbf{x} = \mathbf{x}' - \boldsymbol{\mu}$ , where  $\boldsymbol{\mu}$  is the mean vector of all these samples. Then  $\mathbf{X}$  is an  $L \times M$  data matrix whose columns comprise the  $M$  sample vectors  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M$ , and  $\mathbf{S}_t = \mathbf{X}\mathbf{X}^T$  is defined as the total scatter matrix of the sample vectors. The aim of PCA is to find the transform matrix of a subspace whose basis vectors correspond to the maximum-scatter directions in the original  $L$ -dimensional feature space. Therefore the PCA transform matrix,  $\mathbf{W}_{PCA}$ , is chosen to maximize the determinant of the total scatter matrix of the projected samples

$$\mathbf{W}_{PCA} = \arg \max_{\mathbf{W}} |\tilde{\mathbf{S}}_t| \quad (9.39)$$

where  $\tilde{\mathbf{S}}_t = \mathbf{W}\mathbf{S}_t\mathbf{W}^T$ . The solution to this equation is the transformation matrix,  $\mathbf{W}$ , constructed so that its row vectors are the eigenvectors,  $\mathbf{w}_j$ , of the scatter matrix,  $\mathbf{S}_t$ , arranged in the order of decreasing magnitude of the corresponding eigenvalues  $\lambda_j$ , that is,

$$\mathbf{S}_t \mathbf{w}_j = \lambda_j \mathbf{w}_j, \quad j = 1, 2, \dots, Q \quad (9.40)$$

where the  $\lambda_j$ s are nonzero eigenvalues associated with the eigenvectors  $\mathbf{w}_j$ ;  $Q$  denotes the rank of  $\mathbf{S}_t$  and it cannot exceed the lesser of  $L$  and  $M$ .

Because of the maximum-scatter projection, PCA provides an optimal transformation for representing the original data vector,  $\mathbf{x}$ , from a lower-dimensional subspace in terms of minimum mean square error (MSE) [2]. Let  $\hat{\mathbf{W}}_{PCA}$  be the  $R \times L$  matrix ( $R < L$ ) formed by discarding the lower  $L-R$  rows of  $\mathbf{W}_{PCA}$ . Then the transformed  $R \times 1$  vector  $\hat{\mathbf{y}}$  is given by  $\hat{\mathbf{y}} = \hat{\mathbf{W}}_{PCA} \mathbf{x}$ . The  $\mathbf{x}$  vector can still be reconstructed as  $\hat{\mathbf{x}} = \hat{\mathbf{W}}_{PCA}^T \hat{\mathbf{y}}$  with approximation error given by  $MSE = \sum_{k=R+1}^L \lambda_k$ . Overall, PCA decorrelates the new features and maximizes their variance. The number of new PCA features is equal to the number of original features, but one can decide how many of them to use.

### 9.10.1.2 Linear Discriminant Analysis

Unlike PCA, LDA seeks a linear subspace that best discriminates among object classes, rather than the one that represents samples with least MSE. Specifically, LDA selects the transform matrix  $\mathbf{W}_{\text{LDA}}$  in such a way that the ratio of the between-class scatter to the within-class scatter is maximized [14]. If we define the between-class scatter matrix as

$$\mathbf{S}_b = \sum_{i=1}^c M_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \quad (9.41)$$

and the within-class scatter matrix as

$$\mathbf{S}_w = \sum_{i=1}^c \sum_{j=1}^{M_i} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T \quad (9.42)$$

where  $M_i$  is the number of samples in class  $i$ ,  $c$  is the number of object classes,  $\boldsymbol{\mu}_i$  is the mean of type  $i$  sample vectors, and  $\boldsymbol{\mu}$  is the total mean of sample vectors of all classes. The optimization criterion here is to maximize the determinant ratio of between-class and within-class scatters of the projected samples

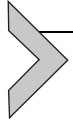
$$\mathbf{W}_{\text{LDA}} = \arg \max_{\mathbf{W}} \left\{ \frac{|\tilde{\mathbf{S}}_b|}{|\tilde{\mathbf{S}}_w|} \right\} \quad (9.43)$$

where  $\tilde{\mathbf{S}}_b = \mathbf{W} \mathbf{S}_b \mathbf{W}^T$  and  $\tilde{\mathbf{S}}_w = \mathbf{W} \mathbf{S}_w \mathbf{W}^T$ . It has been proven [14] that if  $\mathbf{S}_w$  is non-singular, the determinant ratio in Eq. (9.43) is maximized when the row vectors of the transform matrix,  $\mathbf{W}$ , are the generalized eigenvectors of  $\mathbf{S}_w^{-1} \mathbf{S}_b$  corresponding to

$$\mathbf{S}_w^{-1} \mathbf{S}_b \mathbf{w}_i = \lambda_i \mathbf{w}_i, \quad i = 1, 2, \dots, m \quad (9.44)$$

where  $\lambda_i$ ,  $i = 1, 2, \dots, m$  are the generalized eigenvalues, and  $m$  is the number of nonzero generalized eigenvectors,  $m \leq c - 1$ . Notice that the dimensionality of the LDA subspace is upper-bounded by  $c - 1$ , meaning that the total number of LDA features is 1 less than the number of classes. This is because  $\mathbf{S}_b$  is of rank  $c - 1$  or less. Also, since the rank of  $\mathbf{S}_w$  is, at most,  $M - c$ ,  $M$  must be greater than or equal to  $L + c$  in order to ensure that  $\mathbf{S}_w$  does not become singular.

In summary, since LDA maximizes the ability of the new features to discriminate among the classes, it is generally considered to be more effective than PCA for feature reduction prior to classification.



## 9.11 Neural Networks

A completely different approach to classification is the use of artificial neural networks (ANNs) [7]. Here a network is composed of one or more layers of interconnected processing elements (PEs). Each PE creates its output as a weighted sum of its inputs (see Fig. 9.8). The feature values are the inputs to the first layer, and the output values of the final layer are used to assign the object to a class.

The ANN is trained by adjusting the weighting factors in each of its PEs. A large training set of preclassified objects is presented to the network repeatedly, and in random order. Each time, the weights are adjusted to bring the output value toward its correct value. The training process is continued until the error rate stops declining.

The computation performed by such a PE is a function of a dot product, namely

$$O = g(\mathbf{X} \cdot \mathbf{W}) = g \left[ \sum_{i=1}^N x_i w_i \right] = g(S) \quad (9.45)$$

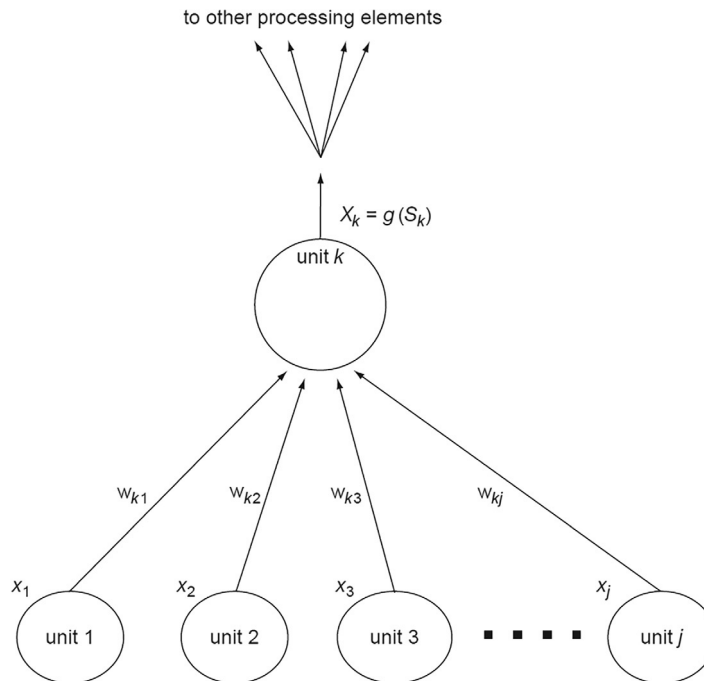


Fig. 9.8 A neural network processing layer.

where  $O$  is the (scalar) output,  $\mathbf{X}$  is the input vector, and  $\mathbf{W}$  is the weight vector associated with that processing element. The weights are adjusted during the training process, and they remain fixed during routine usage.

The weighted sum is subjected to a nonlinear transformation by the *activation function*,  $g(S)$ , which has a sigmoid (S-curve) shape. It is monotonically increasing, differentiable, and it asymptotically approaches 0 and 1 at large negative and positive values of its argument, respectively. An example is

$$g(S) = \frac{1}{1 + e^{-S}} \quad (9.46)$$

The primary purpose of the activation function is to restrict the output of the PE to the range  $[0,1]$ . By convention, outputs are all positive, but interconnection weights can be either positive or negative.

One advantage of the ANN is that it is not necessary to know the statistics (i.e., pdfs) of the features in order to develop a functioning classifier. Further, the decision surfaces that the ANN can implement in feature space are more complex than the second-order surfaces that the parametric Bayes classifier, for example, generates. This can be helpful when the pdfs are multimodal.

A disadvantage of the ANN, as compared to the statistical classifiers previously discussed, is that it is a “black box,” and one is hard pressed to understand or explain its behavior. It also lacks the rich analytical underpinning of the classical approach that provides guidance in the design and development process. This makes it difficult to prove optimality or to predict error rates. Further, if the training is not done properly, on representative training sets of sufficient size, then the net can “overfit” or “memorize” the training set, that is, perform well on the training set but not generalize to objects previously unseen.

Artificial neural networks are discussed in much greater detail in [Chapter 15](#).



## 9.12 Summary of Important Points

1. A well-trained Bayes classifier can be quite effective at multiclass, multifeature classification, even in the presence of considerable noise.
2. One should pay particular attention to numerical precision issues since some of the parameters in the probability calculations can become quite large or quite small.
3. If the marginal distributions are unimodal and symmetrical, it may be useful to assume Gaussian statistics (multivariate normal pdfs).
4. A nonlinear transformation can make a feature's pdf symmetrical.
5. A multimodal pdf suggests the presence of subclasses. Judicious use of subclassing and feature transformations can often make the Gaussian assumption work.

6. When a particular functional form for the pdf (the Gaussian, for example) is known, less training data is required since it is used only to estimate the parameters. This gives rise to a parametric classifier.
7. If the functional form of the pdf is not given or it is known to be non-Gaussian, one must estimate the pdfs directly from the training data. Such classifiers are nonparametric, and they usually require considerably more training data.
8. When available training samples are inadequate to estimate the pdfs accurately, one can choose independent features by mapping the original features using a proper transformation such as independent component analysis. With this method the problem of estimating a multivariate pdf is simplified to that of estimating multiple univariate pdfs, thereby considerably reducing the size requirements of the training set.
9. PCA and LDA are two well-known techniques for reducing a large number of features down to a smaller number without losing their discriminating power. PCA decorrelates the new features and maximizes their variance, whereas LDA maximizes the ability of the features to discriminate among the classes.
10. An ANN classifier has the advantages that it is not necessary to know the statistics of the features in order to function, and the decision surfaces it can implement in feature space are more complex than the second-order surfaces that the parametric Bayes classifier generates. However, its disadvantages are that it is a “black box” and it is difficult to prove optimality or to predict error rates. It also lacks the rich analytical underpinning that supports the design of statistical classifiers.

## References

- [1] B.S. White, K.R. Castleman, Estimating cell populations, *Pattern Recogn.* 13 (5) (1981) 365–370.
- [2] R.O. Duda, *Pattern Classification*, second ed., Wiley, 2007.
- [3] S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*, second ed., Elsevier Academic Press, 2020.
- [4] C.M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2011.
- [5] K.R. Castleman, *Digital Image Processing*, Prentice-Hall, 1996.
- [6] W. Meisel, *Computer-Oriented Approaches to Pattern Recognition*, Academic Press, New York, 1972.
- [7] R. Schalkoff, *Pattern Recognition—Statistical, Structural and Neural Approaches*, John Wiley & Sons, New York, 1992.
- [8] L. Ott, W. Mendenhall, *Understanding Statistics*, fifth ed., PWS-KENT, Boston, 1990.
- [9] C. Forbes, M. Evans, M. Hastings, B. Peacock, *Statistical Distributions*, fourth ed., John Wiley and Sons, 2011.
- [10] A. Hyvärinen, J. Karhunen, E. Oja, *Independent Component Analysis*, John Wiley & Sons, 2001.
- [11] R. Fisher, The statistical utilization of multiple measurements, *Ann. Eugen.* 8 (1938) 376–386.
- [12] I.T. Young, Further considerations of sample size and feature size, *IEEE Trans. IT-24* (6) (1978) 773–775.
- [13] A.K. Jain, B. Chandrasekaran, Dimensionality and sample size considerations in pattern recognition practice, in: *Handbook of Statistics*, vol. 2, North Holland Publishing Company, 1982, pp. 835–855.
- [14] L. Kanal, B. Chandrasekaran, On dimensionality and sample size in statistical pattern recognition, *Pattern Recogn.* 3 (1971) 225–234.