1

FUNDAMENTALS OF PATTERN RECOGNITION

1.1 BASIC CONCEPTS: CLASS, FEATURE, DATA SET

A wealth of literature in the 1960s and 1970s laid the grounds for modern pattern recognition [90,106,140,141,282,290,305,340,353,386]. Faced with the formidable challenges of real-life problems, elegant theories still coexist with ad hoc ideas, intuition, and guessing.

Pattern recognition is about assigning labels to objects. Objects are described by features, also called attributes. A classic example is recognition of handwritten digits for the purpose of automatic mail sorting. Figure 1.1 shows a small data sample. Each 15×15 image is one object. Its class label is the digit it represents, and the features can be extracted from the binary matrix of pixels.

1.1.1 Classes and Class Labels

Intuitively, a class contains similar objects, whereas objects from different classes are dissimilar. Some classes have a clear-cut meaning, and in the simplest case are mutually exclusive. For example, in signature verification, the signature is either genuine or forged. The true class is one of the two, regardless of what we might deduce from the observation of a particular signature. In other problems, classes might be difficult to define, for example, the classes of left-handed and right-handed people or ordered categories such as "low risk," "medium risk," and "high risk."

Combining Pattern Classifiers: Methods and Algorithms, Second Edition. Ludmila I. Kuncheva.

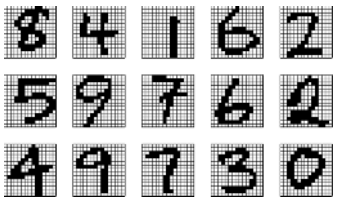


FIGURE 1.1 Example of images of handwritten digits.

We shall assume that there are c possible classes in the problem, labeled from ω_1 to ω_c, organized as a set of labels $\Omega = \{\omega_1, \ldots, \omega_c\}$, and that each object belongs to one and only one class.

1.1.2 Features

Throughout this book we shall consider numerical features. Such are, for example, systolic blood pressure, the speed of the wind, a company's net profit in the past 12 months, the gray-level intensity of a pixel. Real-life problems are invariably more complex than that. Features can come in the forms of categories, structures, names, types of entities, hierarchies, so on. Such nonnumerical features can be transformed into numerical ones. For example, a feature "country of origin" can be encoded as a binary vector with number of elements equal to the number of possible countries where each bit corresponds to a country. The vector will contain 1 for a specified country and zeros elsewhere. In this way one feature gives rise to a collection of related numerical features. Alternatively, we can keep just the one feature where the categories are represented by different values. Depending on the classifier model we choose, the ordering of the categories and the scaling of the values may have a positive, negative, or neutral effect on the relevance of the feature. Sometimes the methodologies for quantifying features are highly subjective and heuristic. For example, sitting an exam is a methodology to quantify a student's learning progress. There are also unmeasurable features that we as humans can assess intuitively but can hardly explain. Examples of such features are sense of humor, intelligence, and beauty.

Once in a numerical format, the feature values for a given object are arranged as an n-dimensional vector $\mathbf{x} = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$. The real space $\mathbb{R}^n$ is called the *feature space*, each axis corresponding to a feature.

Sometimes an object can be represented by multiple, disjoint subsets of features. For example, in identity verification, three different sensing modalities can be used [207]: frontal face, face profile, and voice. Specific feature subsets are measured for each modality and then the feature vector is composed of three sub-vectors, $\mathbf{x} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}]^T$. We call this *distinct pattern representation* after Kittler et al. [207]. As we shall see later, an ensemble of classifiers can be built using distinct pattern representation, with one classifier on each feature subset.

1.1.3 Data Set

The information needed to design a classifier is usually in the form of a labeled *data set* $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$, $\mathbf{z}_j \in \mathbb{R}^n$. The class label of $\mathbf{z}_j$ is denoted by $y_j \in \Omega$, $j = 1, \ldots, N$. A typical data set is organized as a matrix of N rows (objects, also called examples or instances) by n columns (features), with an extra column with the class labels

$$\text{Data set} = \begin{bmatrix} z_{11}, & z_{12}, & \cdots & z_{1n} \\ z_{21}, & z_{22}, & \cdots & z_{2n} \\ \vdots & & & \\ z_{N1}, & z_{N2}, & \cdots & z_{Nn} \end{bmatrix} \quad \text{Labels} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}.$$

Entry $z_{j,i}$ is the value of the i-th feature for the j-th object.

Example 1.1 A shape–color synthetic data set
Consider a data set with two classes, both containing a collection of the following objects: △, □, ○, ▲, ■, and ● . Figure 1.2 shows an example of such a data set. The collections of objects for the two classes are plotted next to one another. Class ω_1 is shaded. The features are only the shape and the color (black or white); the positioning of the objects within the two dimensions is not relevant. The data set contains 256 objects. Each object is labeled in its true class. We can code the color as 0 for white and 1 for black, and the shapes as triangle = 1, square = 2, and circle = 3.

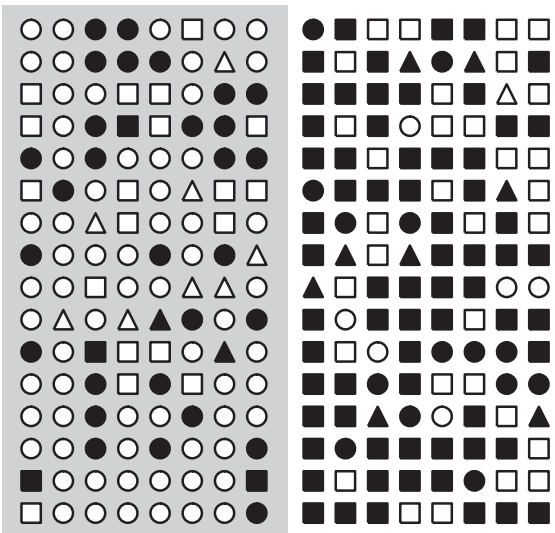


FIGURE 1.2 A shape–color data set example. Class ω_1 is shaded.

Based on the two features, the classes are not completely separable. It can be observed that there are mostly circles in ω_1 and mostly squares in ω_2. Also, the proportion of black objects in class ω_2 is much larger. Thus, if we observe a color and a shape, we can make a decision about the class label. To evaluate the distribution of different objects in the two classes, we can count the number of appearances of each object. The distributions are as follows:

Object	△	□	○	▲	■	●
Class ω_1	9	22	72	1	4	20
Class ω_2	4	25	5	8	79	7
Decision	ω_1	ω_2	ω_1	ω_2	ω_2	ω_1

With the distributions obtained from the given data set, it makes sense to choose class ω_1 if we have a circle (of any color) or a white triangle. For all other possible combinations of values, we should choose label ω_2. Thus using only these two features for labeling, we will make 43 errors (16.8%).

A couple of questions spring to mind. First, if the objects are not discernible, how have they been labeled in the first place? Second, how far can we trust the estimated distributions to generalize over unseen data?

To answer the first question, we should be aware that the features supplied by the user are not expected to be perfect. Typically there is a way to determine the true class label, but the procedure may not be available, affordable, or possible at all. For example, certain medical conditions can be determined only *post mortem*. An early diagnosis inferred through pattern recognition may decide the outcome for the patient. As another example, consider classifying of expensive objects on a production line as good or defective. Suppose that an object has to be destroyed in order to determine the true label. It is desirable that the labeling is done using measurable features that do not require breaking of the object. Labeling may be too expensive, involving time and expertise which are not available. The problem then becomes a pattern recognition one, where we try to find the class label as correctly as possible from the available features.

Returning to the example in Figure 1.2, suppose that there is a third (unavailable) feature which could be, for example, the horizontal axis in the plot. This feature would have been used to label the data, but the quest is to find the best possible labeling method without it.

The second question "How far can we trust the estimated distributions to generalize over unseen data?" has inspired decades of research and will be considered later in this text.

Example 1.2 The Iris data set

The Iris data set was collected by the American botanist Edgar Anderson and subsequently analyzed by the English geneticist and statistician Sir Ronald Aylmer Fisher in 1936 [127]. The Iris data set has become one of the iconic hallmarks of pattern

FIGURE 1.3 Iris flower specimen

recognition and has been used in thousands of publications over the years [39, 348]. This book would be incomplete without a mention of it.

The Iris data still serves as a prime example of a "well-behaved" data set. There are three balanced classes, each represented with a sample of 50 objects. The classes are species of the Iris flower (Figure 1.3): setosa, versicolor, and virginica. The four features describing an Iris flower are sepal length, sepal width, petal length, and petal width. The classes form neat elliptical clusters in the four-dimensional space. Scatter plots of the data in the spaces spanned by the six pairs of features are displayed in Figure 1.4. Class setosa is clearly distinguishable from the other two classes in all projections.

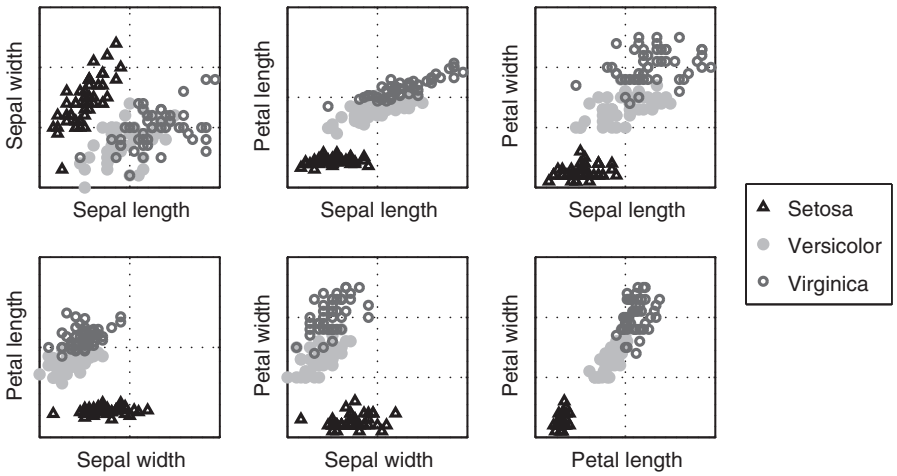


FIGURE 1.4 Scatter plot of the Iris data in the two-dimensional spaces spanned by the six pairs of features.

1.1.4 Generate Your Own Data

Trivial as it might be, sometimes you need a piece of code to generate your own data set with specified characteristics in order to test your own classification method.

1.1.4.1 The Normal Distribution The normal distribution (or also Gaussian distribution) is widespread in nature and is one of the fundamental models in statistics. The one-dimensional normal distribution, denoted $N(\mu, \sigma^2)$, is characterized by mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}$. In n dimensions, the normal distribution is characterized by an n-dimensional vector of the mean, $\boldsymbol{\mu} \in \mathbb{R}^n$, and an $n \times n$ covariance matrix Σ. The notation for an n-dimensional normally distributed random variable is $\mathbf{x} \sim N(\boldsymbol{\mu}, \Sigma)$. The normal distribution is the most natural assumption reflecting the following situation: there is an "ideal prototype" ($\boldsymbol{\mu}$) and all the data are distorted versions of it. Small distortions are more likely to occur than large distortions, causing more objects to be located in the close vicinity of the ideal prototype than far away from it. The scatter of the points around the prototype $\boldsymbol{\mu}$ is associated with the covariance matrix Σ_i.

The probability density function (pdf) of $\mathbf{x} \sim N(\boldsymbol{\mu}, \Sigma)$ is

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}} \sqrt{|\Sigma|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})\right\}, \tag{1.1}$$

where $|\Sigma|$ is the determinant of Σ. For the one-dimensional case, x and μ are scalars, and Σ reduces to the variance σ^2. Equation 1.1 simplifies to

$$p(x) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right\}. \tag{1.2}$$

Example 1.3 Cloud shapes and the corresponding covariance matrices Figure 1.5 shows four two-dimensional data sets generated from the normal distribution with different covariance matrices shown underneath.

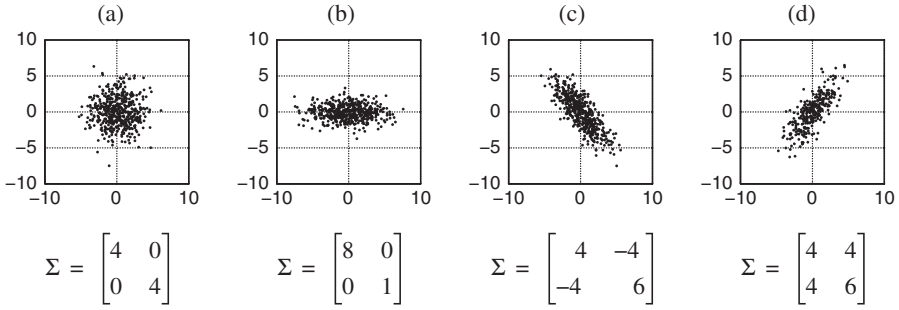


FIGURE 1.5 Normally distributed data sets with mean $[0, 0]^T$ and different covariance matrices shown underneath.

Figures 1.5a and 1.5b are generated with *independent* (noninteracting) features. Therefore, the data cloud is either spherical (Figure 1.5a), or stretched along one or more coordinate axes (Figure 1.5b). Notice that for these cases the off-diagonal entries of the covariance matrix are zeros. Figures 1.5c and 1.5d represent cases where the features are *dependent*. The data for this example was generated using the function `samplegaussian` in Appendix 1.A.1.

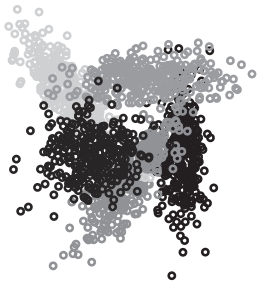
In the case of *independent features* we can decompose the n-dimensional pdf as a product of n one-dimensional pdfs. Let σ_k^2 be the diagonal entry of the covariance matrix Σ for the k-th feature, and μ_k be the k-th component of $\boldsymbol{\mu}$. Then

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}}\sqrt{|\Sigma|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$
$$= \prod_{k=1}^{n}\left(\frac{1}{\sqrt{(2\pi)}\,\sigma_k} \exp\left\{-\frac{1}{2}\left(\frac{x_k-\mu_k}{\sigma_k}\right)^2\right\}\right). \tag{1.3}$$

The *cumulative distribution function* for a random variable $X \in \mathbb{R}$ with a normal distribution, $\Phi(z) = P(X \leq z)$, is available in tabulated form from most statistical textbooks.[1]

1.1.4.2 Noisy Geometric Figures Sometimes it is useful to generate your own data set of a desired shape, prevalence of the classes, overlap, and so on. An example of a challenging classification problem with five Gaussian classes is shown in Figure 1.6 along with the MATLAB code that generates and plots the data.

One possible way to generate data with specific geometric shapes is detailed below. Suppose that each of the c classes is described by a shape, governed by parameter t.



```
x1 = samplegaussian(300,[0 0],[4 0;0 4]);
x2 = samplegaussian(300,[4 6],[8 0;0 1]);
x3 = samplegaussian(300,[-2 5],[4 -4;-4 6]);
x4 = samplegaussian(300,[3 -2],[4 4;4 6]);
x5 = samplegaussian(300,[7 0],[1 0;0 10]);
data = [x1;x2;x3;x4;x5];
t = ones(300,1);
labels = [t;2*t;3*t;4*t;5*t];
scatter(data(:,1),data(:,2),[],...
    labels,'linewidth',2.5)
axis equal off
```

FIGURE 1.6 An example of five Gaussian classes generated using the `samplegaussian` function from Appendix 1.A.1.

[1] $\Phi(z)$ can be approximated with error at most 0.005 for $0 \leq z \leq 2.2$ as [150]

$$\Phi(z) = 0.5 + \frac{z(4.4-z)}{10}.$$

The noise-free data is calculated from t, and then noise is added. Let t_i be the parameter for class ω_i, and $[a_i, b_i]$ be the interval for t_i describing the shape of the class. Denote by p_i the desired prevalence of class ω_i. Knowing that $p_1 + \cdots + p_c = 1$, we can calculate the approximate number of samples in a data set of N objects. Let N_i be the desired number of objects from class ω_i. The first step is to sample uniformly N_i values for t_i from the interval $[a_i, b_i]$. Subsequently, we find the coordinates $x_1, \ldots, x_n$ for each element of t_i. Finally, noise is added to all values. (We can use the `randn` MATLAB function for this purpose.) The noise could be scaled by multiplying the values by different constants for the different features. Alternatively, the noise could be scaled with the feature values or the values of t_i.

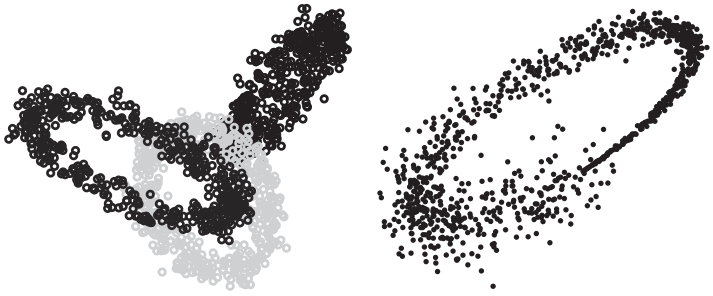
Example 1.4 Ellipses data set

The code for producing this data set is given in Appendix 1.A.1. We used the parametric equations for two-dimensional ellipses:

$$x(t) = x_c + a\cos(t)\cos(\phi) - b\sin(t)\sin(\phi),$$
$$y(t) = y_c + a\cos(t)\sin(\phi) - b\sin(t)\cos(\phi),$$

where (x_c, y_c) is the center of the ellipse, a and b are respectively the major and the minor semi-axes of the ellipse, and ϕ is the angle between the x-axis and the major axis. To traverse the whole ellipse, parameter t varies from 0 to 2π.

Figure 1.7a shows a data set where the random noise is the same across both features and all values of t. The classes have equal proportions, with 300 points from each class. Using a single ellipse with 1000 points, Figure 1.7b demonstrates the effect of scaling the noise with the parameter t. The MATLAB code is given in Appendix 1.A.1.



(a) (b)

FIGURE 1.7 (a) The three-ellipse data set; (b) one ellipse with noise variance proportional to the parameter t.

1.1.4.3 Rotated Checker Board Data. This is a two-dimensional data set which spans the unit square $[0, 1] \times [0, 1]$. The classes are placed as the light and the dark squares of a checker board and then the whole board is rotated at an angle α. A

`samplecb(5000,0.2,pi/6)` $(a = 0.2, \alpha = \frac{\pi}{6})$ `samplecb(5000,0.5,-pi/3)` $(a = 0.5, \alpha = -\frac{\pi}{3})$

FIGURE 1.8 Rotated checker board data (100,000 points in each plot).

parameter a specifies the side of the individual square. For example, if $a = 0.5$, there will be four squares in total before the rotation. Figure 1.8 shows two data sets, each containing 5,000 points, generated with different input parameters. The MATLAB function `samplecb(N,a,alpha)` in Appendix 1.A.1 generates the data.

The properties which make this data set attractive for experimental purposes are:

- The two classes are perfectly separable.
- The classification regions for the same class are disjoint.
- The boundaries are not parallel to the coordinate axes.
- The classification performance will be highly dependent on the sample size.

1.2 CLASSIFIER, DISCRIMINANT FUNCTIONS, CLASSIFICATION REGIONS

A *classifier* is any function that will assign a class label to an object $\mathbf{x}$:

$$D : \mathbb{R}^n \to \Omega. \tag{1.4}$$

In the "canonical model of a classifier" [106], *c discriminant functions* are calculated

$$g_i : \mathbb{R}^n \to \mathbb{R}, \quad i = 1, \ldots, c, \tag{1.5}$$

each one yielding a score for the respective class (Figure 1.9). The object $\mathbf{x} \in \mathbb{R}^n$ is labeled to the class with the highest score. This labeling choice is called the *maximum membership rule*. Ties are broken randomly, meaning that $\mathbf{x}$ is assigned randomly to one of the tied classes.

The discriminant functions partition the feature space $\mathbb{R}^n$ into *c decision regions* or *classification regions* denoted $\mathcal{R}_1, \ldots, \mathcal{R}_c$:

$$\mathcal{R}_i = \left\{ \mathbf{x} \,\middle|\, \mathbf{x} \in \mathbb{R}^n, g_i(\mathbf{x}) = \max_{k=1,\ldots,c} g_k(\mathbf{x}) \right\}, \quad i = 1, \ldots, c. \tag{1.6}$$

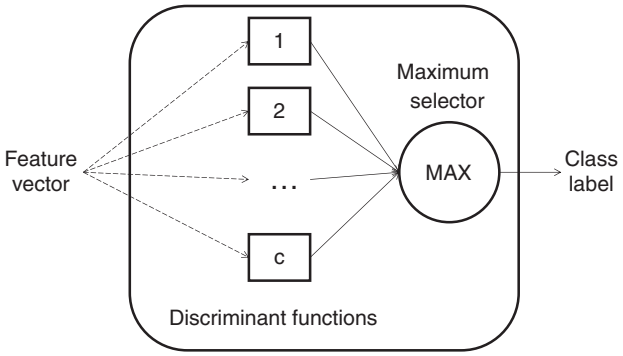


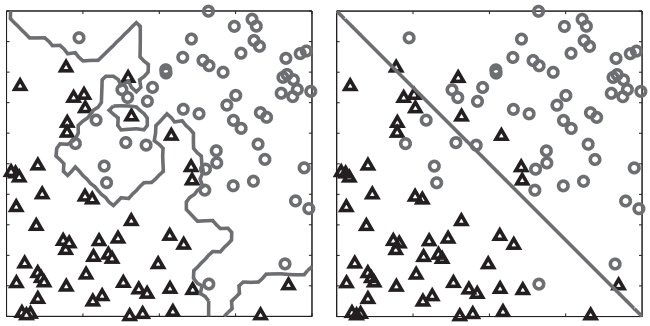
FIGURE 1.9 Canonical model of a classifier. An n-dimensional feature vector is passed through c discriminant functions, and the largest function output determines the class label.

The decision region for class ω_i is the set of points for which the i-th discriminant function has the highest score. According to the maximum membership rule, all points in decision region $\mathcal{R}_i$ are assigned to class ω_i. The decision regions are specified by the classifier D, or equivalently, by the discriminant functions G. The boundaries of the decision regions are called *classification boundaries* and contain the points for which the highest discriminant functions tie. A point on the boundary can be assigned to any of the bordering classes. If a decision region $\mathcal{R}_i$ contains data points from the labeled set $\mathbf{Z}$ with true class label $\omega_j, j \neq i$, classes ω_i and ω_j are called *overlapping*. If the classes in $\mathbf{Z}$ can be separated completely by a hyperplane (a point in $\mathbb{R}$, a line in $\mathbb{R}^2$, a plane in $\mathbb{R}^3$), they are called *linearly separable*.

Note that overlapping classes in a given partition can be nonoverlapping if the space was partitioned in a different way. If there are no identical points with different class labels in the data set $\mathbf{Z}$, we can *always* partition the feature space into pure classification regions. Generally, the smaller the overlapping, the better the classifier. Figure 1.10 shows an example of a two-dimensional data set and two sets of classification regions. Figure 1.10a shows the regions produced by the nearest neighbor classifier, where every point is labeled as its nearest neighbor. According to these boundaries and the plotted data, the classes are nonoverlapping. However, Figure 1.10b shows the optimal classification boundary and the optimal classification regions which guarantee the minimum possible error for unseen data generated from the same distributions. According to the optimal boundary, the classes are overlapping. This example shows that by striving to build boundaries that give a perfect split we may over-fit the training data.

Generally, *any* set of functions $g_1(\mathbf{x}), \ldots, g_c(\mathbf{x})$ is a set of *discriminant functions*. It is another matter how successfully these discriminant functions separate the classes.

Let $G^* = \{g_1^*(\mathbf{x}), \ldots, g_c^*(\mathbf{x})\}$ be a set of *optimal* (in some sense) discriminant functions. We can obtain infinitely many sets of *optimal* discriminant functions from G^* by applying a monotonic transformation $f(g_i^*(\mathbf{x}))$ that preserves the order of the function values for every $\mathbf{x} \in \mathbb{R}^n$. For example, $f(\zeta)$ can be a $\log(\zeta)$ or a^ζ, for $a > 1$.



(a) (b)

FIGURE 1.10 Classification regions obtained from two different classifiers: (a) the 1-nn boundary (nonoverlapping classes); (b) the optimal boundary (overlapping classes).

Applying the same f to all discriminant functions in G^*, we obtain an equivalent set of discriminant functions. Using the maximum membership rule, $\mathbf{x}$ will be labeled to the same class by any of the equivalent sets of discriminant functions.

1.3 CLASSIFICATION ERROR AND CLASSIFICATION ACCURACY

It is important to know how well our classifier performs. The *performance* of a classifier is a compound characteristic, whose most important component is the classification accuracy. If we were able to try the classifier on all possible input objects, we would know exactly how accurate it is. Unfortunately, this is hardly a possible scenario, so an estimate of the accuracy has to be used instead.

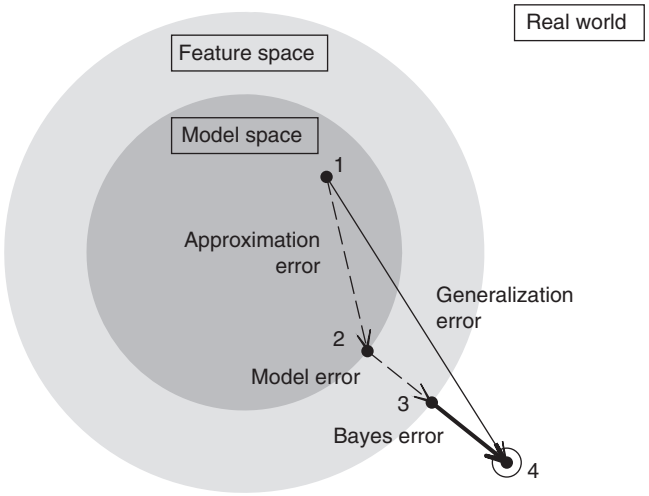
Classification error is a characteristic dual to the classification accuracy in that the two values sum up to 1

$$\text{Classification error} = 1 - \text{Classification accuracy}.$$

The quantity of interest is called the *generalization error*. This is the expected error of the trained classifier on unseen data drawn from the distribution of the problem.

1.3.1 Where Does the Error Come From? Bias and Variance

Why cannot we design the perfect classifier? Figure 1.11 shows a sketch of the possible sources of error. Suppose that we have chosen the classifier model. Even with a perfect training algorithm, our solution (marked as 1 in the figure) may be away from the best solution with this model (marked as 2). This *approximation error* comes from the fact that we have only a finite data set to train the classifier. Sometimes the training algorithm is not guaranteed to arrive at the optimal classifier with the given data. For example, the backpropagation training algorithm converges to a local



1: Our solution

2: Best possible solution with the chosen model

3: Best possible solution with the available features

4: The "real thing"

FIGURE 1.11 Composition of the generalization error.

minimum of the criterion function. If started from a different initialization point, the solution may be different. In addition to the approximation error, there may be a *model error*. Point 3 in the figure is the best possible solution in the given feature space. This point may not be achievable with the current classifier model. Finally, there is an irreducible part of the error, called *the Bayes error*. This error comes from insufficient representation. With the available features, two objects with the same feature values may have different class labels. Such a situation arose in Example 1.1.

Thus the true generalization error P_{G} of a classifier D trained on a given data set $\mathbf{Z}$ can be decomposed as

$$P_{\mathrm{G}}(D, \mathbf{Z}) = P_{\mathrm{A}}(\mathbf{Z}) + P_{\mathrm{M}} + P_{\mathrm{B}}, \tag{1.7}$$

where $P_{\mathrm{A}}(\mathbf{Z})$ is the approximation error, P_{M} is the model error, and P_{B} is the Bayes error. The first term in the equation can be thought of as variance due to using different training data or non-deterministic training algorithms. The second term, P_{M}, can be taken as the bias of the model from the best possible solution.

The difference between bias and variance is explained in Figure 1.12. We can liken building the perfect classifier to shooting at a target. Suppose that our training algorithm generates different solutions owing to different data samples, different initialisations, or random branching of the training algorithm. If the solutions are

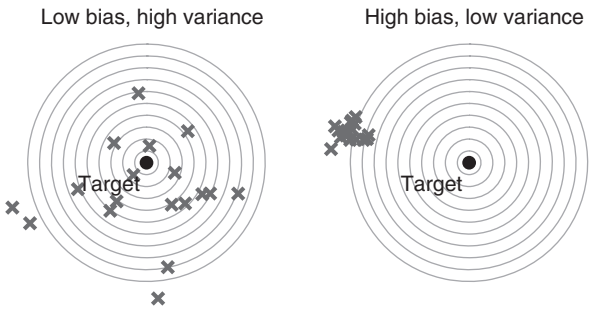


FIGURE 1.12 Bias and variance.

grouped together, variance is low. Then the distance to the target will be more due to the bias. Conversely, widely scattered solutions indicate large variance, and that can account for the distance between the shot and the target.

1.3.2 Estimation of the Error

Assume that a labeled data set $\mathbf{Z}_{\text{ts}}$ of size $N_{\text{ts}} \times n$ is available for testing the accuracy of our classifier, D. The most natural way to calculate an estimate of the error is to run D on all the objects in $\mathbf{Z}_{\text{ts}}$ and find the proportion of misclassified objects, called sometimes the *apparent error rate*

$$\hat{P}_D = \frac{N_{\text{error}}}{N_{\text{ts}}}. \tag{1.8}$$

Dual to this characteristic is the apparent classification accuracy which is calculated by $1 - \hat{P}_D$.

To look at the error from a probabilistic point of view, we can adopt the following model. The classifier commits an error with probability P_D on any object $\mathbf{x} \in \mathbb{R}^n$ (a wrong but useful assumption). Then the number of errors has a binomial distribution with parameters (P_D, N_{ts}). An estimate of P_D is $\hat{P}_D$. If N_{ts} and P_D satisfy the rule of thumb: $N_{\text{ts}} > 30$, $\hat{P}_D \times N_{\text{ts}} > 5$, and $(1 - \hat{P}_D) \times N_{\text{ts}} > 5$, the binomial distribution can be approximated by a normal distribution. The 95% confidence interval for the error is

$$\left[\hat{P}_D - 1.96\sqrt{\frac{\hat{P}_D(1-\hat{P}_D)}{N_{\text{ts}}}},\ \ \hat{P}_D + 1.96\sqrt{\frac{\hat{P}_D(1-\hat{P}_D)}{N_{\text{ts}}}} \right]. \tag{1.9}$$

By calculating the confidence interval we estimate how well *this* classifier (D) will fare on unseen data from the same problem. Ideally, we will have a large representative testing set, which will make the estimate precise.

1.3.3 Confusion Matrices and Loss Matrices

To find out how the errors are distributed across the classes we construct a *confusion matrix* using the testing data set, $\mathbf{Z}_{\text{ts}}$. The entry a_{ij} of such a matrix denotes the number of elements from $\mathbf{Z}_{\text{ts}}$ whose true class is ω_i, and which are assigned by D to class ω_j. The estimate of the classification accuracy can be calculated as the trace of the matrix divided by the total sum of the entries. The additional information that the confusion matrix provides is *where* the misclassifications have occurred. This is important for problems with a large number of classes where a high off-diagonal entry of the matrix might indicate a difficult two-class problem that needs to be tackled separately.

Example 1.5 Confusion matrix for the Letter data

The Letters data set, available from the UCI Machine Learning Repository Database, contains data extracted from 20,000 black-and-white images of capital English letters. Sixteen numerical features describe each image ($N = 20{,}000, c = 26, n = 16$). For the purpose of this illustration we used the hold-out method. The data set was randomly split into halves. One half was used for training a linear classifier, and the other half was used for testing. The labels of the testing data were matched to the labels obtained from the classifier, and the 26×26 confusion matrix was constructed. If the classifier was ideal, and all assigned and true labels were matched, the confusion matrix would be diagonal.

Table 1.1 shows the row in the confusion matrix corresponding to class "H." The entries show the number of times that true "H" is mistaken for the letter in the respective column. The boldface number is the diagonal entry showing how many times "H" has been correctly recognized. Thus, from the total of 350 examples of "H" in the testing set, only 159 have been labeled correctly by the classifier. Curiously, the largest number of mistakes, 33, are for the letter "O." Figure 1.13 visualizes the confusion matrix for the letter data set. Darker color signifies a higher value. The diagonal shows the darkest color, which indicates the high correct classification rate (over 69%). Three common misclassifications are indicated with arrows in the figure.

TABLE 1.1 The "H"-row in the Confusion Matrix for the Letter Data Set Obtained from a Linear Classifier Trained on 10,000 Points

"H" labeled as:	A	B	C	D	E	F	G	H	I	J	K	L	M
Times:	1	6	1	18	0	1	2	**159**	0	0	30	0	1

"H" labeled as:	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Times:	27	33	2	9	21	0	0	11	4	3	20	1	0

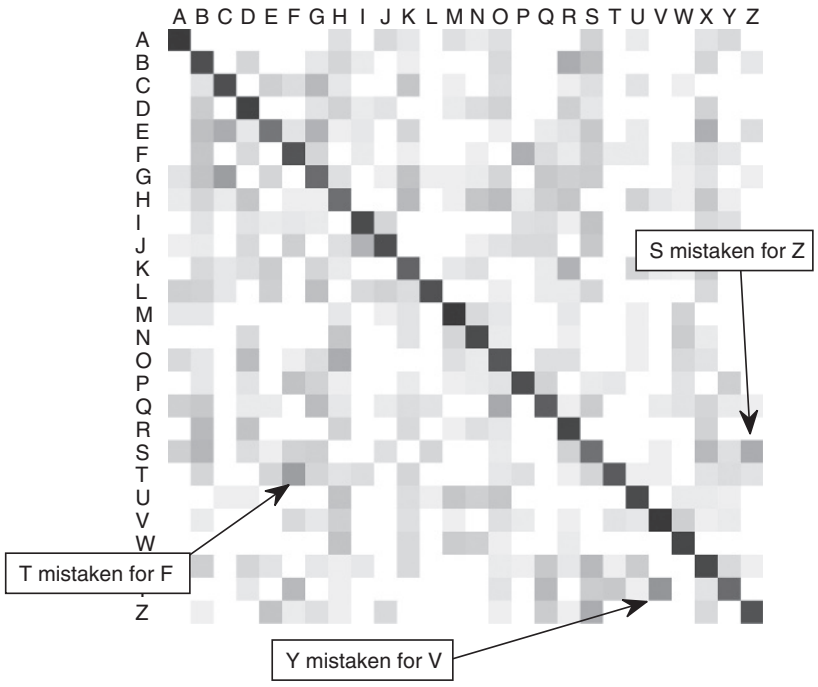


FIGURE 1.13 Graphical representation of the confusion matrix for the letter data set. Darker color signifies a higher value.

The errors in classification are not equally costly. To account for the different costs of mistakes, we introduce the *loss matrix*. We define a loss matrix with entries λ_{ij} denoting the loss incurred by assigning label ω_i, given that the true label of the object is ω_j. If the classifier is "unsure" about the label, it may refuse to make a decision. An extra class called "refuse-to-decide" can be added to the set of classes. Choosing the extra class should be less costly than choosing a wrong class. For a problem with c original classes and a refuse option, the loss matrix is of size $(c + 1) \times c$. Loss matrices are usually specified by the user. A zero–one loss matrix is defined as $\lambda_{ij} = 0$ for $i = j$ and $\lambda_{ij} = 1$ for $i \neq j$; that is, all errors are equally costly.

1.3.4 Training and Testing Protocols

The estimate $\hat{P}_D$ in Equation 1.8 is valid only for the given classifier D and the testing set from which it was calculated. It is possible to train a better classifier from different training data sampled from the distribution of the problem. What if we seek to answer the question "How well can this *classifier model* solve the problem?"

Suppose that we have a data set $\mathbf{Z}$ of size $N \times n$, containing n-dimensional feature vectors describing N objects. We would like to use as much as possible of the data

to build the classifier (*training*), and also as much as possible unseen data to test its performance (*testing*). However, if we use all data for training and the same data for testing, we might *overtrain* the classifier. It could learn perfectly the available data but its performance on unseen data cannot be predicted. That is why it is important to have a separate data set on which to examine the final product. The most widely used training/testing protocols can be summarized as follows [216]:

- *Resubstitution*. Design classifier D on $\mathbf{Z}$ and test it on $\mathbf{Z}$. $\hat{P}_D$ is likely optimistically biased.
- *Hold-out*. Traditionally, split $\mathbf{Z}$ randomly into halves; use one half for training and the other half for calculating $\hat{P}_D$. Splits in other proportions are also used.
- *Repeated hold-out (Data shuffle)*. This is a version of the hold-out method where we do L random splits of $\mathbf{Z}$ into training and testing parts and average all L estimates of P_D calculated on the respective testing parts. The usual proportions are 90% for training and 10% for testing.
- *Cross-validation*. We choose an integer K (preferably a factor of N) and randomly divide $\mathbf{Z}$ into K subsets of size N/K. Then we use one subset to test the performance of D trained on the union of the remaining $K-1$ subsets. This procedure is repeated K times choosing a different part for testing each time.

  To get the final value of $\hat{P}_D$ we average the K estimates.

  To reduce the effect of the single split into K folds, we can carry out repeated cross-validation. In an $M \times K$-fold cross validation, the data is split M times into K folds, and a cross-validation is performed on each such split. This procedure results in $M \times K$ estimates of $\hat{P}_D$, whose average produces the desired estimate. A 10×10-fold cross-validation is a typical choice of such a protocol.
- *Leave-one-out*. This is the cross-validation protocol where $K = N$, that is, one object is left aside, the classifier is trained on the remaining $N-1$ objects, and the left out object is classified. $\hat{P}_D$ is the proportion of the N objects misclassified in their respective cross-validation fold.
- *Bootstrap*. This method is designed to correct for the optimistic bias of resubstitution. This is done by randomly sampling with replacement L sets of cardinality N from the original set $\mathbf{Z}$. Approximately 37% $(1/e)$ of the data will not be chosen in a bootstrap replica. This part of the data is called the "out-of-bag" data. The classifier is built on the bootstrap replica and assessed on the out-of-bag data (testing data). L such classifiers are trained, and the error rates on the respective testing data are averaged. Sometimes the resubstitution and the out-of-bag error rates are taken together with different weights [216].

Hold-out, repeated hold-out and cross-validation can be carried out with *stratified sampling*. This means that the proportions of the classes are preserved as close as possible in all folds.

Pattern recognition has now outgrown the stage where the computation resource (or lack thereof) was the decisive factor as to which method to use. However, even with the modern computing technology, the problem has not disappeared. The ever

growing sizes of the data sets collected in different fields of science and practice pose a new challenge. We are back to using the good old hold-out method, first because the others might be too time-consuming, and second, because the amount of data might be so excessive that small parts of it will suffice for training and testing. For example, consider a data set obtained from retail analysis, which involves hundreds of thousands of transactions. Using an estimate of the error over, say, 10,000 data points, can conveniently shrink the confidence interval and make the estimate sufficiently reliable.

It is now becoming common practice to use three instead of two data sets: one for training, one for *validation*, and one for testing. As before, the testing set remains unseen during the training process. The validation data set acts as pseudo-testing. We continue the training process until the performance improvement on the training set is no longer matched by a performance improvement on the validation set. At this point the training should be stopped so as to avoid overtraining. Not all data sets are large enough to allow for a validation part to be cut out. Many of the data sets from the UCI Machine Learning Repository Database[2] [22], often used as benchmarks in pattern recognition and machine learning, may be unsuitable for a three-way split into training/validation/testing. The reason is that the data subsets will be too small and the estimates of the error on these subsets would be unreliable. Then stopping the training at the point suggested by the validation set might be inadequate, the estimate of the testing accuracy might be inaccurate, and the classifier might be poor because of the insufficient training data.

When multiple training and testing sessions are carried out, there is the question of which of the classifiers built during this process we should use in the end. For example, in a 10-fold cross-validation, we build 10 different classifiers using different data subsets. The above methods are only meant to give us an estimate of the accuracy of a certain model built for the problem at hand. We rely on the assumption that the classification accuracy will change smoothly with the changes in the size of the training data [99]. Therefore, if we are happy with the accuracy and its variability across different training subsets, we should finally train a our chosen classifier on the whole data set.

1.3.5 Overtraining and Peeking

Testing should be done on previously *unseen* data. All parameters should be tuned on the training data. A common mistake in classification experiments is to select a feature set using the given data, and *then* run experiments with one of the above protocols to evaluate the accuracy of that set. This problem is widespread in bioinformatics and neurosciences, aptly termed "peeking" [308, 346, 348, 370]. Using the same data is likely to lead to an optimistic bias of the error.

Example 1.6 Tuning a parameter on the testing set is wrong
Let $D(r)$ be a classifier with a parameter r such that varying r leads to different training accuracies. To demonstrate this effect, here we took a random training sample of

[2] http://www.ics.uci.edu/∼mlearn/MLRepository.html

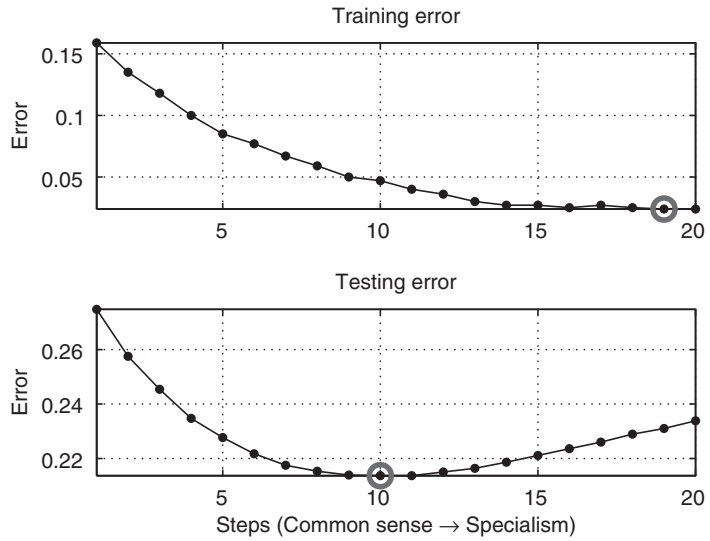


FIGURE 1.14 Example of overtraining: letter data set.

1000 objects from the letters data set. The remaining 19,000 objects were used for testing. A quadratic discriminant classifier (QDC) was used.[3] We vary a parameter r, $r \in [0, 1]$, called the regularization parameter, which determines to what extent we sacrifice adjustment to the given data in favor of robustness. For $r = 0$ there is no regularization; we have more accuracy on the training data and less certainty that the classifier will perform well on unseen data. For $r = 1$, the classifier might be less accurate on the training data but can be expected to perform at the same rate on unseen data. This dilemma can be translated into everyday language as "specific expertise" versus "common sense." If the classifier is trained to expertly recognize a certain data set, it might have this data-specific expertise and little common sense. This will show as high testing error. Conversely, if the classifier is trained to have good common sense, even if not overly successful on the training data, we might expect it to have common sense with any data set drawn from the same distribution.

In the experiment, r was decreased for 20 steps, starting with $r_0 = 0.4$ and taking r_{k+1} to be $0.8 \times r_k$. Figure 1.14 shows the training and the testing errors for the 20 steps.

This example is intended to demonstrate the overtraining phenomenon in the process of varying a parameter, therefore we will look at the tendencies in the error curves. While the training error decreases steadily with r, the testing error decreases to a certain point, and then increases again. This increase indicates overtraining, where the classifier becomes too much of a data-specific expert and loses common sense. A common mistake in this case is to declare that the QDC has a testing error of 21.37% (the minimum in the bottom plot). The mistake is in that the *testing* set was used to find the best value of r.

[3]Discussed in Chapter 2.

The problem of peeking, largely due to unawareness of its caveats, is alarmingly common in application studies on feature selection. In view of this, we discuss this issue further in Chapter 9.

1.4 EXPERIMENTAL COMPARISON OF CLASSIFIERS

There is no single "best" classifier. Classifiers applied to different problems and trained by different algorithms perform differently [107, 110, 173, 196]. Comparative studies are usually based on extensive experiments using a number of simulated and real data sets. When talking about experiment design, I cannot refrain from quoting again and again a masterpiece of advice by George Nagy titled *Candide's practical principles of experimental pattern recognition* [287] (Just a note—this is a joke! DO NOT DO THIS!)

- Comparison of classification accuracies. *Comparisons against algorithms proposed by others are distasteful and should be avoided. When this is not possible, the following Theorem of Ethical Data Selection may prove useful.*
- Theorem. *There exists a set of data for which a candidate algorithm is superior to any given rival algorithm. This set may be constructed by omitting from the test set any pattern which is misclassified by the candidate algorithm.*
- Replication of experiments. *Since pattern recognition is a mature discipline, the replication of experiments on new data by independent research groups, a fetish in the physical and biological sciences, is unnecessary. Concentrate instead on the accumulation of novel, universally applicable algorithms.*
- Casey's caution. *Do not ever make your experimental data available to others; someone may find an obvious solution that you missed.*

Albeit meant to be satirical, the above principles are surprisingly widespread and closely followed! Speaking seriously now, the rest of this section gives some practical tips and recommendations.

A point raised by Duin [110] is that the performance of a classifier depends upon the expertise and the *willingness* of the designer. There is not much to be done for classifiers with fixed structures and training procedures (called "automatic" classifiers in [110]). For classifiers with many training parameters however, we can make them work or fail. Keeping in mind that there are no rules defining a fair comparison of classifiers, here are a few (non-Candide's) guidelines:

1. Pick the training procedures in advance and keep them fixed during training. When publishing, give enough detail so that the experiment is reproducible by other researchers.
2. Compare modified versions of classifiers with the *original* (nonmodified) classifier. For example, a distance-based modification of k-nearest neighbors (k-nn) should be compared with the standard k-nn first, and then with other classifier models. If a slight modification of a certain model is being compared with

TABLE 1.2 The 2 × 2 Relationship Table with Counts

	D_2 correct (1)	D_2 wrong (0)
D_1 correct (1)	N_{11}	N_{10}
D_1 wrong (0)	N_{01}	N_{00}

Total, $N_{11} + N_{10} + N_{01} + N_{00} = N_{ts}$

a totally different classifier, then it is not clear who deserves the credit—the modification or the original model itself.

3. Make sure that all the information about the data is utilized by all classifiers to the largest extent possible. For example, a clever initialization of a method can make it favorite among a group of equivalent but randomly initialized methods.
4. Make sure that the testing set has not been seen at any stage of the training.
5. If possible, give also the complexity of the classifier: training and running times, memory requirements, and so on.

1.4.1 Two Trained Classifiers and a Fixed Testing Set

Suppose that we have two trained classifiers which have been run on the same testing data giving testing accuracies of 98% and 96%, respectively. Can we claim that the first classifier is significantly better than the second one?

McNemar test. The testing results for two classifiers D_1 and D_2 on a testing set with N_{ts} objects can be organized as shown in Table 1.2. We consider two output values: 0 for incorrect classification and 1 for correct classification. Thus N_{pq} is the number of objects in the testing set with output p from the first classifier and output q from the second classifier, $p, q \in \{0, 1\}$.

The null hypothesis H_0 is that there is no difference between the accuracies of the two classifiers. If the null hypothesis is correct, then the expected counts for both off-diagonal entries in Table 1.2 are $\frac{1}{2}(N_{01} + N_{10})$. The discrepancy between the expected and the observed counts is measured by the following statistic:

$$s = \frac{\left(|N_{01} - N_{10}| - 1\right)^2}{N_{01} + N_{10}}, \tag{1.10}$$

which is approximately distributed as χ^2 with 1 degree of freedom. The "−1" in the numerator is a continuity correction [99]. The simplest way to carry out the test is to calculate s and compare it with the tabulated χ^2 value for, say, level of significance[4] $\alpha = 0.05$. If $s > 3.841$, we reject the null hypothesis and accept that the

[4]The *level of significance* of a statistical test is the probability of rejecting H_0 when it is true, in other words, the probability to "convict the innocent." This error is called *Type I error*. The alternative error, when we do not reject H_0 when it is in fact incorrect, is called *Type II error*. The corresponding name for it would be "free the guilty." Both errors are needed in order to characterize a statistical test. For example,

two classifiers have significantly different accuracies. A MATLAB function for this test, called `mcnemar`, is given in Appendix 1.A.2.

Example 1.7 A comparison on the Iris data

We took the first two features of the Iris data (Example 1.2) and classes "versicolor" and "virginica." The data was split into 50% training and 50% testing parts. The testing data is plotted in Figure 1.15. The linear and the quadratic discriminant classifiers

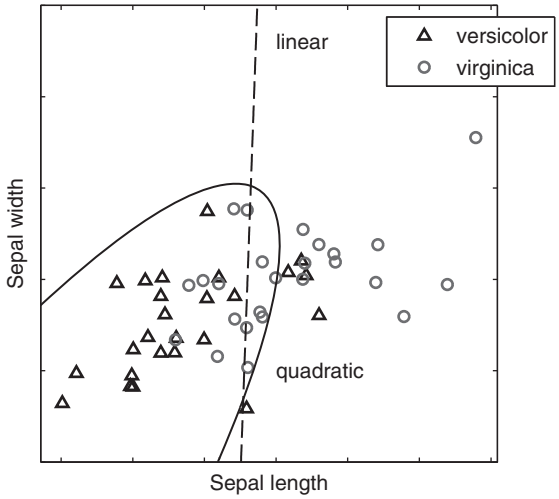


FIGURE 1.15 Testing data from the Iris data set and the decision boundaries of the linear and the quadratic discriminant classifiers.

(LDC and QDC, both detailed later) were trained on the training data. Their decision boundaries are plotted in Figure 1.15.

The confusion matrices of the two classifiers are as follows:

	LDC			QDC	
	Versicolor	Virginica		Versicolor	Virginica
Versicolor	20	5	Versicolor	20	5
Virginica	8	17	Virginica	14	11

Taking LDC to be classifier 1 and QDC, classifier 2, the values in Table 1.2 are as follows: $N^{11} = 31$, $N^{10} = 0$, $N^{01} = 6$, and $N^{00} = 13$. The difference is due to the six virginica objects in the "loop." These are correctly labeled by QDC and mislabeled by LDC. From Equation 1.10,

$$s = \frac{(|0-6|-1)^2}{0+6} = \frac{25}{6} \approx 4.1667. \tag{1.11}$$

if we always accept H_0, there will be no Type I error at all. However, in this case the Type II error might be large. Ideally, both errors should be small.

Since the calculated s is greater than the tabulated value of 3.841, we reject the null hypothesis and accept that LDC and QDC are significantly different. Note that the Iris data is hardly large enough to be suitable for the hold-out protocol. It was used here for the purpose of the illustration only.

1.4.2 Two Classifier Models and a Single Data Set

Dietterich [99] details four important sources of variation that have to be taken into account when comparing classifier models.

1. *The choice of the testing set*. Different testing sets may rank differently classifiers which otherwise have the same accuracy across the whole population. Therefore, it is dangerous to draw conclusions from a single testing experiment, especially when the data size is small.
2. *The choice of the training set*. Some classifier models are called *unstable* [47] because small changes in the training set can cause substantial changes of the classifier trained on this set. Examples of unstable classifiers are decision tree classifiers and some neural networks.[5] Unstable classifiers are versatile models which are capable of adapting, so that most or all training examples are correctly classified. The instability of such classifiers is, in a way, the pay-off for their versatility. As we shall see later, unstable classifiers play a major role in classifier ensembles. Here we note that the variability with respect to the training data has to be accounted for.
3. *The internal randomness of the training algorithm*. Some training algorithms have a random component. This might be the initialization of the parameters of the classifier which are then fine-tuned (e.g., the backpropagation algorithm for training neural networks) or a stochastic procedure for tuning the classifier. Thus the trained classifier might be different for the same training set and even for the same initialization of the parameters.
4. *The random classification error.* Dietterich [99] considers the possibility of having mislabeled objects in the testing data as the fourth source of variability.

The above list suggests that multiple training and testing sets should be used, and multiple training runs should be carried out. Consider the task of comparing two classifier models (methods) over the *same data set* using one of the multi-test protocols. Let $E_{i,j}$ be the error of classifier i, $i \in \{1, 2\}$, for the j-th testing set, $j = 1, \ldots, K$. We can apply the traditional paired t-test for comparing the errors where $E_{1,j}$ are paired with $E_{2,j}$. However, this test may be inadequate because it does not take into account the fact that the training data used to build the classifier models are dependent [12, 89, 99, 286]. For example, in a K-fold cross-validation experiment, fold 1 will be used once for testing and $K - 1$ times for training the classifier. This may lead to overly liberal outcome of the statistical test, allowing the discovery of nonexisting significant differences between the two models. To avoid that, Nadeau and Bengio propose an amendment to the calculation of the *variance* of the error obtained as the average of T testing errors [286].

[5] All classifier models mentioned will be discussed later.

In the paired t-test, we calculate the differences $d_1, \ldots, d_T$, where $d_j = E_{1,j} - E_{2,j}$ for all j, and check the hypothesis that the mean of these differences is 0. Let σ_{d} be the empirical standard deviation of the differences. If the training data sets were independent, the standard deviation of the *mean* difference would be $\sigma_{\mathrm{d}'} = \frac{\sigma_{\mathrm{d}}}{\sqrt{T}}$. To account for the fact that the training data sets are not independent, we use instead

$$\sigma_{\mathrm{d}'} = \sigma_{\mathrm{d}} \sqrt{\frac{1}{T} + \frac{N_{\mathrm{testing}}}{N_{\mathrm{training}}}}, \tag{1.12}$$

where N_{training} and N_{testing} are the sizes of the training and the testing sets, respectively. For a K-fold cross-validation,

$$\sigma_{\mathrm{d}'} = \sigma_{\mathrm{d}} \sqrt{\frac{1}{K} + \frac{1}{K-1}} = \sigma_{\mathrm{d}} \sqrt{\frac{2K-1}{K(K-1)}} \quad . \tag{1.13}$$

This amendment holds for cross-validation, repeated cross-validation, data shuffle, and the bootstrap methods.

Example 1.8 Correction of the variance for multiple testing sets

This example presents a Monte Carlo simulation to illustrate the need for the variance correction. Consider two Gaussian classes as shown in Figure 1.16. The classes have means $(-1, 0)$ and $(1, 0)$, and identity covariance matrices. We generated 200 data sets from this distribution; 20 points from class 1 and 20 points from class 2 in each data set. An example of such a set is circled in Figure 1.16. With each data set, we carried out 30 data shuffle runs by splitting the data into 90% training (36 data points) and 10% testing (4 data points). The LDC (detailed later) was trained on the training part and tested on the testing part.

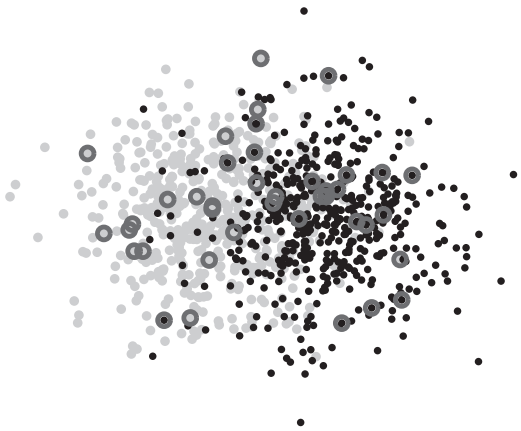


FIGURE 1.16 Scatter plot of the two Gaussian classes. One of the 40-point data sets sampled from these classes is marked with circles.

Let $e_1, \ldots, e_{200}$ be the estimates of the classification error in the data shuffle experiment. Value e_i is the average of 30 testing errors with data set i (the data shuffle

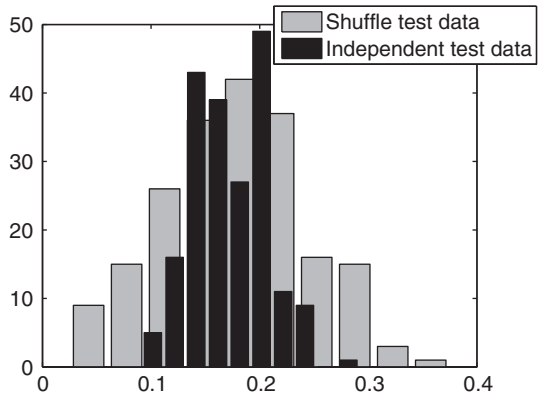


FIGURE 1.17 Histograms of the classification error for the two experiments.

protocol). Denote the mean and the standard deviation of these errors by μ_e and σ_e, respectively.

Consider now a matching experiment where we sample independently $200 \times 30 = 6000$ training sets of 36 objects and testing sets of 4 objects, storing the results as 200 batches of 30 runs. Denote the errors from the 200 batches by $q_1, \ldots, q_{200}$. Denote the mean and the standard deviation of these errors by μ_q and σ_q, respectively. Figure 1.17 shows the histograms for e and q. While the means of both errors are about 15.9% (the theoretical error), the spreads of the two histograms are different. In this example we obtained $\sigma_e = 0.0711$ and $\sigma_q = 0.0347$.

For each data shuffle experiment, we calculated not only the error e_i but also the standard deviation s_i. If the training and testing data were independently drawn, the standard error of the mean would be $\bar{s}_i = \frac{s_i}{\sqrt{30}}$. The average of $\bar{s}_i$ across i would be close to σ_e. However, this calculation gives a value of 0.0320, which is closer to σ_q than to σ_e, and does not properly reflect the larger spread seen in the histogram. Now we apply the correction and use $s_i^* = s_i\sqrt{\frac{1}{30} + \frac{1}{9}}$. The average of s_i^* across i is 0.0667, which is much closer to the observed σ_e.

Consider two models A and B, and T estimates of the classification error obtained through cross-validation or data shuffle. Denote these estimates by $a_1, a_2, \ldots, a_T$ and $b_1, b_2, \ldots, b_T$, respectively. The null hypothesis of the test, H_0, is that there is no difference between the mean errors of A and B for the given data set. The alternative hypothesis, H_1, is that there is difference. The step-by-step procedure for carrying out the amended paired t-test (two-tailed) is as follows:

1. Calculate the differences $d_i = a_i - b_i$, $i = 1, \ldots, T$. Calculate the mean and the standard deviation of d_i

$$m_{\mathrm{d}} = \frac{1}{T}\sum_{i=1}^{T} d_i, \qquad s_{\mathrm{d}} = \sqrt{\frac{1}{T-1}\sum_{i=1}^{T}(d_i - m_{\mathrm{d}})^2}\,.$$

2. Calculate the amended standard error of the mean

$$s'_{\rm d} = s_{\rm d} \sqrt{\frac{1}{T} + \frac{N_{\rm testing}}{N_{\rm training}}} \, .$$

3. Calculate the test statistic $t_{\rm d} = \frac{m_{\rm d}}{s'_{\rm d}}$ and the degrees of freedom $\text{df} = T - 1$.
4. For a two-tailed t-test, find the p-value as

$$p = 2 \, F_t(-|t_{\rm d}|, \text{df}),$$

where F_t is the Student's t cumulative distribution function.

If we set the alternative hypothesis H_1 to be "A has lower error than B" (one-tailed test), the p-value should be calculated as

$$p = F_t(t_{\rm d}, \text{df}).$$

Comparing the obtained p-value with the chosen level of significance α, we reject H_0 if $p < \alpha$ and accept it otherwise. Function `tvariance` in Appendix 1.A.2 can be used for this calculation.

Example 1.9 Paired t-test with corrected variance

Suppose that the values of the error (in %) in a 10-fold cross-validation experiment were as follows:

Model A:	7.4	18.1	13.7	17.5	13.0	12.5	8.9	12.1	12.4	7.4
Model B:	9.9	11.0	5.7	12.5	2.7	6.6	10.6	6.4	12.5	7.8

The mean of the difference between the errors of models A and B is $m = 3.73$ and the standard deviation is $s = 4.5090$. The standard error of the mean is therefore $\bar{s} = s/\sqrt{10} \approx 1.4259$. The p-value for the (traditional) two-tailed paired t-test ($10 - 1 = 9$ degrees of freedom) is

$$p = 2 \times F_t \left(-\frac{|m|}{\bar{s}}, \text{df} \right) = 2 \times F_t \left(-\frac{3.73}{1.4259}, 9 \right) \approx 0.0280 \, .$$

According to this test, at $\alpha = 0.05$, we can accept the alternative hypothesis that there is significant difference between the two classifier models. Knowing that the training and testing data were not independent, the amended standard deviation is $\bar{s}' = s \left(\sqrt{\frac{1}{10} + \frac{1}{9}} \right) \approx 2.0717$. The corrected p-value is

$$p' = 2 \times F_t \left(-\frac{|m|}{\bar{s}'}, \text{df} \right) = 2 \times F_t \left(-\frac{3.73}{2.0717}, 9 \right) \approx 0.1053.$$

This result does not give us ground to reject the null hypothesis and declare that there is difference between the two means.

The two examples highlight the importance of the variance correction when the training and testing data are dependent.

1.4.3 Two Classifier Models and Multiple Data Sets

Over the years, researchers have developed affinity for using extensively the UCI Machine Learning Repository [22] for drawing a sample of data sets and running comparative experiments on these [347]. We tend to over-tune our classification algorithms to these data sets and may ignore in the process data sets that present a real-life challenge [347]. If the claim is that algorithm A is better than algorithm B in general, then a large and diverse collection of data sets should be used.

The Wilcoxon signed rank test. Demšar proposes that the data sets chosen for the comparison of models A and B may be thought of as independent trials, but dissuades the reader from using a paired t-test [89]. The classification errors of different data sets are hardly commensurable. To bypass this problem, the Wilcoxon signed rank test was deemed more suitable. Let $a_1, a_2, \ldots, a_N$ and $b_1, b_2, \ldots, b_N$ in this context denote the error estimates of models A and B for the N *data sets* chosen for the experiment. These estimates can be obtained through any of the protocols, for example a 10-fold cross-validation. Again let $d_i = a_i - b_i$, $i = 1, \ldots, N$ be the differences of the errors. The Wilcoxon signed rank test does not take into account the exact value of d_i, only its relative magnitude. The null hypothesis of the test is that the data in vector d_i come from a continuous, symmetric distribution with zero median, against the alternative that the distribution does not have zero median. The test is applied in the following steps:[6]

1. Rank the absolute values of the distances $|d_i|$ so that the smallest distance receives rank 1 and the largest distance receives rank N. If there is a tie, all the ranks are shared so that the total sum stays $1 + 2 + \cdots + N$. For example, if there are four equal smallest distances, each will be assigned rank $(1 + 2 + 3 + 4)/4 = 2.5$. Thus each data set receives a rank r_i.
2. Split the ranks into positive and negative depending on the sign of d_i and calculate the sums:

$$R^+ = \sum_{d_i>0} r_i + \frac{1}{2}\sum_{d_i=0} r_i, \qquad R^- = \sum_{d_i<0} r_i + \frac{1}{2}\sum_{d_i=0} r_i.$$

3. Take as the test statistic $T = \min(R^+, R^-)$ and compare it with the critical value for the respective number of data sets N and the chosen level of significance. Table 1.A.1(a) in Appendix 1.A.2 gives the critical values for this test for $6 \le N \le 25$. For values $N > 25$, the following statistic is approximately normally distributed [89]:

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}}.$$

[6]Function `signrank` from the Statistics Toolbox of MATLAB can be used to calculate the p-value for this test.

The sign test. A simpler but less powerful alternative to this test is the sign test. This time we do not take into account the magnitude of the differences, only their sign. By doing so, we further avoid the problem of noncommensurable errors or differences thereof. For example, an error difference of 2% for a given data set may be more relevant than a difference of 4% on another data set. It is common practice to count WINS, DRAWS, and LOSSES, with or without statistical significance attached to these. The sign test is based on the intuition that if models A and B are equivalent, each one will score better than the other on approximately $N/2$ of the N data sets. Demšar [89] gives a useful table for checking the significance of the difference between models A and B tested on N data sets based on the sign test. We reproduce the table (with a small correction) as Table 1.A.1(b) and explain the calculation of the critical values in Appendix 1.A.2.

The table contains the required number of wins of A over B in order to reject H_0 and claim that model A is better than model B. The ties are split equally between A and B. For $N > 25$ data sets, we can use the normal approximation of the binomial distribution (mean $N/2$ and standard deviation $\sqrt{N}/2$). If the number of wins for A is greater than $N/2 + 1.96\sqrt{N}/2$, A is significantly better than B at $\alpha = 0.05$ (see [89] for more details).

1.4.4 Multiple Classifier Models and Multiple Data Sets

Demšar [89] recommends the Friedman test followed by the pairwise Nemenyi test for this task.

Friedman test with Iman and Davenport amendment. This is a nonparametric alternative of the analysis-of-variance (ANOVA) test. The classifier models are ranked on each of the N data sets. The best classifier receives rank 1 and the worst receives rank N. Tied ranks are shared equally as explained above. Let r_i^j be the rank of classifier model j on data set i, where $i = 1, \ldots, N$ and $j = 1, \ldots, M$. Let $R_j = \frac{1}{N}\sum_{i=1}^{N} r_i^j$ be the average rank of model j. The test statistic is

$$x_F^2 = \frac{12N}{M(M+1)}\left(\sum_{j=1}^{M} R_j^2 - \frac{M(M+1)^2}{4}\right). \tag{1.14}$$

The null hypothesis of the test H_0 is that all classifier models are equivalent. Under the null hypothesis, x_F^2 follows the χ^2 distribution with $M-1$ degrees of freedom (for $N > 10$ and $M > 5$ [89]).[7] Demšar advocates an amendment of the test statistic proposed by Iman and Davenport:

$$F_F = \frac{(N-1)x_F^2}{N(M-1) - x_F^2}, \tag{1.15}$$

[7]Function `friedman` from the Statistics Toolbox of MATLAB can be used to calculate the p-value for this test. Note that the MATLAB implementation contains an additional correction for tied ranks. This gives a slightly different test statistic compared to Equation 1.14 if there are tied ranks. See http://www.unistat.com/.

which follows the F-distribution with $(M-1)$ and $(M-1)(N-1)$ degrees of freedom. The statistic's value is compared with the tabled critical values for the F-distribution (available in standard statistics textbooks), and if F_F is larger, we reject H_0 and accept that there is difference between the classifier models. Instead of using pre-tabulated critical values, a MATLAB function `imandavenport` for calculating the p-value of this test is given in Appendix 1.A.2.

Example 1.10 Comparison of 11 classifier models on 20 data sets

This is a fictional example which demonstrates the calculation of the test statistic for the Friedman test and its modification. Table 1.3 displays the classification errors of the 11 classifier models for the 20 data sets. The data sets were arbitrarily named, just for fun, as the first 20 chemical elements of the periodic table. The "classification errors" were generated independently, as the absolute values of a normally distributed random variable with mean 0 and standard deviation 10, subsequently rounded to one decimal place. Thus, we do not expect to find significant differences between the models.

The corresponding ranks are shown in Table 1.4. Notice the shared ranks: both models C6 and C11 have the minimum error rate of 0.1 for data set "Nitrogen" and therefore equally share ranks 1 and 2, both models receiving rank 1.5. The average ranks are given in the bottom row. The Friedman statistic calculated as in

TABLE 1.3 Classification Errors of 11 Classifier Models on 20 Data Sets (A Fictional Example)

Data set	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
Hydrogen	4.1	14.1	5.8	2.3	22.5	1.4	21.8	8.0	9.1	15.5	5.4
Helium	5.0	1.4	6.2	2.6	11.7	6.7	12.0	4.7	2.4	8.7	5.6
Lithium	0.8	12.9	5.4	9.0	12.0	8.5	3.7	0.5	6.5	1.5	19.8
Beryllium	1.6	4.9	4.8	21.6	6.6	0.6	24.1	3.5	12.7	3.9	5.4
Boron	5.3	6.2	1.1	0.9	3.3	2.9	7.4	12.6	5.5	13.2	1.4
Carbon	7.2	9.3	3.4	9.5	15.1	6.1	3.1	10.4	0.9	8.0	6.2
Nitrogen	8.5	2.8	9.3	11.7	8.9	0.1	5.4	4.3	3.5	1.4	0.1
Oxygen	8.0	2.0	0.1	17.4	9.6	4.1	1.2	0.8	9.9	4.2	11.1
Fluorine	7.3	1.4	11.4	3.7	4.9	9.4	9.6	17.9	4.3	8.2	1.9
Neon	16.9	8.8	4.3	11.9	4.4	8.4	2.5	8.0	10.0	8.5	11.2
Sodium	3.9	0.8	1.8	9.5	2.0	10.9	18.9	4.4	6.3	3.6	2.5
Magnesium	5.1	6.0	6.6	14.2	9.8	3.2	12.2	6.3	10.5	27.5	15.6
Aluminum	4.1	3.7	5.8	0.3	15.7	0.2	3.8	15.3	5.1	15.1	12.0
Silicon	10.7	8.4	16.2	2.6	3.7	11.6	0.5	27.3	3.3	4.3	2.4
Phosphorus	9.7	2.8	0.2	5.2	2.2	4.9	19.5	1.7	16.4	2.3	10.0
Sulfur	2.7	35.7	4.3	6.8	5.4	12.2	5.7	4.8	19.1	8.3	19.2
Chlorine	6.3	34.1	20.3	6.7	2.6	15.9	0.8	14.1	0.4	8.3	6.3
Argon	6.3	11.5	13.3	6.8	11.0	5.3	0.8	9.7	7.0	5.0	7.5
Potassium	0.8	7.9	3.2	3.1	5.5	2.6	7.9	2.9	1.7	23.2	2.1
Calcium	13.8	12.8	8.3	3.8	21.7	3.9	10.4	11.7	15.4	7.9	7.7

TABLE 1.4 Ranks of the 11 Classifier Models on the 20 Data Sets (Fictional Example)

Data set	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
Hydrogen	3.0	8.0	5.0	2.0	11.0	1.0	10.0	6.0	7.0	9.0	4.0
Helium	5.0	1.0	7.0	3.0	10.0	8.0	11.0	4.0	2.0	9.0	6.0
Lithium	2.0	10.0	5.0	8.0	9.0	7.0	4.0	1.0	6.0	3.0	11.0
Beryllium	2.0	6.0	5.0	10.0	8.0	1.0	11.0	3.0	9.0	4.0	7.0
Boron	6.0	8.0	2.0	1.0	5.0	4.0	9.0	10.0	7.0	11.0	3.0
Carbon	6.0	8.0	3.0	9.0	11.0	4.0	2.0	10.0	1.0	7.0	5.0
Nitrogen	8.0	4.0	10.0	11.0	9.0	1.5	7.0	6.0	5.0	3.0	1.5
Oxygen	7.0	4.0	1.0	11.0	8.0	5.0	3.0	2.0	9.0	6.0	10.0
Fluorine	6.0	1.0	10.0	3.0	5.0	8.0	9.0	11.0	4.0	7.0	2.0
Neon	11.0	7.0	2.0	10.0	3.0	5.0	1.0	4.0	8.0	6.0	9.0
Sodium	6.0	1.0	2.0	9.0	3.0	10.0	11.0	7.0	8.0	5.0	4.0
Magnesium	2.0	3.0	5.0	9.0	6.0	1.0	8.0	4.0	7.0	11.0	10.0
Aluminum	5.0	3.0	7.0	2.0	11.0	1.0	4.0	10.0	6.0	9.0	8.0
Silicon	8.0	7.0	10.0	3.0	5.0	9.0	1.0	11.0	4.0	6.0	2.0
Phosphorus	8.0	5.0	1.0	7.0	3.0	6.0	11.0	2.0	10.0	4.0	9.0
Sulfur	1.0	11.0	2.0	6.0	4.0	8.0	5.0	3.0	9.0	7.0	10.0
Chlorine	4.5	11.0	10.0	6.0	3.0	9.0	2.0	8.0	1.0	7.0	4.5
Argon	4.0	10.0	11.0	5.0	9.0	3.0	1.0	8.0	6.0	2.0	7.0
Potassium	1.0	9.5	7.0	6.0	8.0	4.0	9.5	5.0	2.0	11.0	3.0
Calcium	9.0	8.0	5.0	1.0	11.0	2.0	6.0	7.0	10.0	4.0	3.0
R_j	5.22	6.28	5.50	6.10	7.10	4.88	6.28	6.10	6.05	6.55	5.95

Equation 1.14 is 6.9182. The p-value for the χ^2 distribution with $M - 1 = 10$ degrees of freedom is 0.7331. This value supports H_0: equal classifier models.

Applying the amendment from Equation 1.15, we arrive at $F_F = 0.6808$. The p-value of the F-test with $(M - 1)$ and $(M - 1)(N - 1)$ degrees of freedom is 0.7415, again supporting H_0.

The post-hoc test. If H_0 is rejected, Demšar [89] proposes the use of Nemenyi post-hoc test to find exactly where the differences are. All pairs of classifiers are examined. Two classifiers are declared different if their average ranks differ by more than a given critical value. For instance, for a pair of classifiers i and j, a test statistic is calculated using the average ranks R_i and R_j:

$$z = \frac{R_i - R_j}{\sqrt{\frac{M(M+1)}{6N}}}. \tag{1.16}$$

The number of pairwise comparisons $M(M - 1)/2$ determines the level of significance for this z-value. If the desired level of significance is α, the difference will be flagged as significant if the obtained p-value is smaller than $\frac{2\alpha}{M(M-1)}$. When a classifier model is singled out and compared with the remaining $M - 1$ models, the scaling constant

is just $(M-1)$ (Bonferroni-Dunn correction of the family-wise error). García and Herrera [147] explain in detail further step-wise procedures for post-hoc comparing of pairs of classifiers. The MATLAB code for both Nemenyi and Bonferroni-Dunn post-hoc tests is given in Appendix 1.A.2.

Example 1.11 Post-hoc tests

The fictional comparison example was slightly modified. A constant of 0.8 was subtracted from the first column of Table 1.3, and all values in this column were multiplied by 0.5. This made classifier C1 better than all other classifier models. The ranks changed correspondingly, leading to the following average ranks:

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
R_j	3.10	6.47	5.65	6.35	7.25	5.28	6.42	6.30	6.20	6.80	6.17
Nemenyi					■					■	
Bonferroni		■		■	■		■	■	■	■	■

The Friedman test statistic is 21.7273, giving a p-value of 0.0166. The Iman and Davenport amendment gives $F_F = 2.3157$ and a p-value of 0.0136. According to both tests, there is a difference among the 11 classifier models. The Nemenyi post-hoc test found significant differences at $\alpha < 0.05$ between C1 and C5 and also between C1 and C10 (two-tailed test). Nominating C1 as the classifier of interest, the Bonferroni–Dunn post-hoc test found C1 to be better (smaller error) than all classifiers except C3 and C6 (one-tailed test). The results from the post-hoc tests are shown underneath the average ranks above. A black square indicates that significant difference was found at $\alpha < 0.05$.

1.5 BAYES DECISION THEORY

1.5.1 Probabilistic Framework

Although many types of uncertainty exist, the probabilistic model fits surprisingly well in most pattern recognition problems. We assume that the class label ω is a random variable taking values in the set $\Omega = \{\omega_1, \ldots, \omega_c\}$. The prior probabilities, $P(\omega_i)$, $i = 1, \ldots, c$, constitute the probability mass function (pmf) of the variable ω:

$$0 \leq P(\omega_i) \leq 1, \qquad \text{and} \qquad \sum_{i=1}^{c} P(\omega_i) = 1. \tag{1.17}$$

We can construct a classifier based on this information only. To make the smallest possible number of mislabelings, we should always label an object with the class of the highest prior probability.

However, by measuring the relevant characteristics of the objects organized as the vector $\mathbf{x} \in \mathbb{R}^n$, we should be able to make a more accurate decision about this

particular object. Assume that the objects from class ω_i are distributed in $\mathbb{R}^n$ according to the *class-conditional pdf* $p(\mathbf{x}|\omega_i)$, where $p(\mathbf{x}|\omega_i) \geq 0$, $\forall \mathbf{x} \in \mathbb{R}^n$, and

$$\int_{\mathbb{R}^n} p(\mathbf{x}|\omega_i) \; d\mathbf{x} = 1, \qquad i = 1, \ldots, c. \tag{1.18}$$

The likelihood of $\mathbf{x} \in \mathbb{R}^n$ is given by the *unconditional pdf*:

$$p(\mathbf{x}) = \sum_{i=1}^{c} P(\omega_i) \; p(\mathbf{x}|\omega_i). \tag{1.19}$$

Given the prior probabilities and the class-conditional pdfs, we can calculate the *posterior probability* that the true class label of the measured $\mathbf{x}$ is ω_i using the Bayes formula

$$P(\omega_i|\mathbf{x}) = \frac{P(\omega_i) \; p(\mathbf{x}|\omega_i)}{p(\mathbf{x})} = \frac{P(\omega_i) \; p(\mathbf{x}|\omega_i)}{\sum_{j=1}^{c} P(\omega_j) \; p(\mathbf{x}|\omega_j)}. \tag{1.20}$$

Equation 1.20 gives the probability mass function of the class label variable ω *for the observed* $\mathbf{x}$. The classification decision for that particular $\mathbf{x}$ should be made with respect to the posterior probability. Choosing the class with the highest posterior probability will lead to the smallest possible error when classifying any object with feature vector $\mathbf{x}$.

The probability model described above is valid for the discrete case as well. Let $\mathbf{x}$ be a discrete variable with possible values in $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_s\}$. The only difference from the continuous-valued case is that instead of class-conditional pdf, we use *class-conditional pmf*, $P(\mathbf{x}|\omega_i)$, giving the probability that a particular value from $\mathbf{V}$ occurs if we draw at random an object from class ω_i. For all pmfs,

$$0 \leq P(\mathbf{x}|\omega_i) \leq 1, \quad \forall \mathbf{x} \in \mathbf{V}, \quad \text{and} \quad \sum_{j=1}^{s} P(\mathbf{v}_j|\omega_i) = 1. \tag{1.21}$$

1.5.2 Discriminant Functions and Decision Boundaries

The posterior probabilities can be used directly as the discriminant functions, that is,

$$g_i(\mathbf{x}) = P(\omega_i|\mathbf{x}), \qquad i = 1, \ldots, c. \tag{1.22}$$

Hence we can rewrite the maximum membership rule as

$$D(\mathbf{x}) = \omega_{i^*} \in \Omega \quad \Longleftrightarrow \quad P(\omega_{i^*}|\mathbf{x}) = \max_{i=1,\ldots,c} \{P(\omega_i|\mathbf{x})\}. \tag{1.23}$$

In fact, a set of discriminant functions leading to the same classification regions would be

$$g_i(\mathbf{x}) = P(\omega_i) \; p(\mathbf{x}|\omega_i), \qquad i = 1, \ldots, c, \tag{1.24}$$

because the denominator of Equation 1.20 is the same for all i, and so will not change the ranking order of g_is. Another useful set of discriminant functions derived from the posterior probabilities is

$$g_i(\mathbf{x}) = \log(P(\omega_i)\, p(\mathbf{x}|\omega_i)), \qquad i = 1, \ldots, c. \tag{1.25}$$

Example 1.12 Decision/classification boundaries

Let $x \in \mathbb{R}$. Figure 1.18 shows two sets of discriminant functions for three normally distributed classes with

$$\begin{aligned} P(\omega_1) &= 0.45, \qquad & p(x|\omega_1) &\sim N\left(4, (2.0)^2\right) \\ P(\omega_2) &= 0.35, \qquad & p(x|\omega_2) &\sim N\left(5, (1.2)^2\right) \\ P(\omega_3) &= 0.20, \qquad & p(x|\omega_3) &\sim N\left(7, (1.0)^2\right). \end{aligned}$$

Figure 1.18a depicts the first set of discriminant functions (Equation 1.24), obtained as $P(\omega_i)\, p(x|\omega_i)$, $i = 1, 2, 3$. The classification boundaries are marked with bullets on the x-axis. The posterior probabilities (Equation 1.22) are depicted as the second set of discriminant functions in Figure 1.18b. The classification regions specified by the boundaries are displayed with different shades of gray. Note that the same regions are found for both sets of discriminant functions.

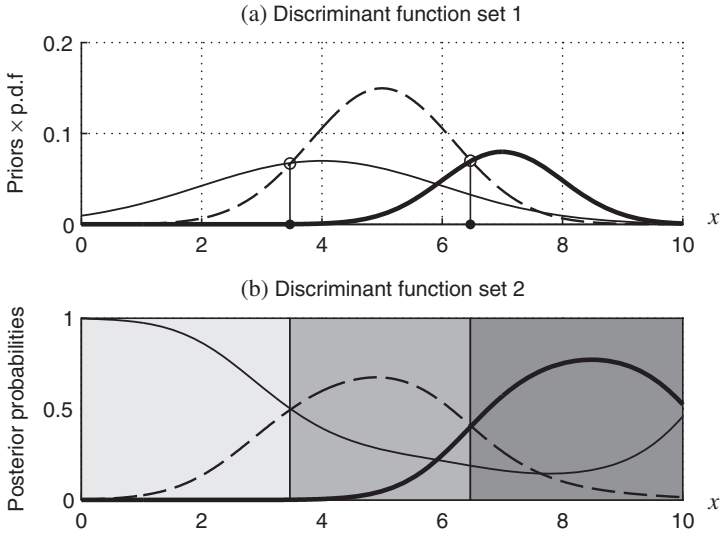


FIGURE 1.18 Plot of two equivalent sets of discriminant functions: (a) $P(\omega_1)p(x|\omega_1)$ (the thin line), $P(\omega_2)p(x|\omega_2)$ (the dashed line), and $P(\omega_3)p(x|\omega_3)$ (the thick line); (b) $P(\omega_1|x)$ (the thin line), $P(\omega_2|x)$ (the dashed line), and $P(\omega_3|x)$ (the thick line).

Sometimes more than two discriminant functions might tie at the boundaries. Ties are resolved randomly.

1.5.3 Bayes Error

Let D^* be a classifier which always assigns the class label with the largest posterior probability. Since for every $\mathbf{x}$ we can only be correct with probability

$$P(\omega_{i^*}|\mathbf{x}) = \max_{i=1,\ldots,c} \{P(\omega_i|\mathbf{x})\}, \tag{1.26}$$

there is some inevitable error. The overall probability of error of D^* is the sum of the errors of each individual $\mathbf{x}$ weighted by its likelihood value, $p(\mathbf{x})$, that is,

$$P_{\mathrm{e}}(D^*) = \int_{\mathbb{R}^n} (1 - P(\omega_{i^*}|\mathbf{x}))p(\mathbf{x})\, d\mathbf{x}. \tag{1.27}$$

It is convenient to split the integral into c integrals, one on each classification region. In this case, $\mathbf{x}$ will be given label ω_{i^*} corresponding to the region's tag where $\mathbf{x}$ belongs. Then

$$P_{\mathrm{e}}(D^*) = \sum_{i=1}^{c} \int_{\mathcal{R}_i^*} (1 - P(\omega_i|\mathbf{x}))p(\mathbf{x})\, d\mathbf{x}, \tag{1.28}$$

where $\mathcal{R}_i^*$ is the classification region for class ω_i, $\mathcal{R}_i^* \cap \mathcal{R}_j^* = \emptyset$ for any $j \neq i$ and $\cup_{i=1}^{c} \mathcal{R}_i^* = \mathbb{R}^n$. Substituting Equation 1.20 into Equation 1.28 and taking into account that $\sum_{i=1}^{c} \int_{\mathcal{R}_i^*} = \int_{\mathbb{R}^n}$,

$$P_{\mathrm{e}}(D^*) = \sum_{i=1}^{c} \int_{\mathcal{R}_i^*} \left(1 - \frac{P(\omega_i)p(\mathbf{x}|\omega_i)}{p(\mathbf{x})}\right) p(\mathbf{x})\, d\mathbf{x} \tag{1.29}$$

$$= \int_{\mathbb{R}^n} p(\mathbf{x})\, d\mathbf{x} - \sum_{i=1}^{c} \int_{\mathcal{R}_i^*} P(\omega_i)p(\mathbf{x}|\omega_i)\, d\mathbf{x} \tag{1.30}$$

$$= 1 - \sum_{i=1}^{c} \int_{\mathcal{R}_i^*} P(\omega_i)p(\mathbf{x}|\omega_i)\, d\mathbf{x}. \tag{1.31}$$

Note that $P_{\mathrm{e}}(D^*) = 1 - P_{\mathrm{c}}(D^*)$, where $P_{\mathrm{c}}(D^*)$ is the overall probability of correct classification of D^*, or the classification accuracy.

Consider a different classifier, D, which produces classification regions $\mathcal{R}_1, \ldots, \mathcal{R}_c$, $\mathcal{R}_i \cap \mathcal{R}_j = \emptyset$ for any $j \neq i$ and $\cup_{i=1}^{c} \mathcal{R}_i = \mathbb{R}^n$. Regardless of the way the regions are formed, the error of D is

$$P_{\mathrm{e}}(D) = \sum_{i=1}^{c} \int_{\mathcal{R}_i} (1 - P(\omega_i|\mathbf{x}))p(\mathbf{x})\, d\mathbf{x}\,. \tag{1.32}$$

The error of D^* is the smallest possible error, called the *Bayes error*. The example below illustrates this concept.

Example 1.13 Bayes error

Consider the simple case of $x \in \mathbb{R}$ and $\Omega = \{\omega_1, \omega_2\}$. Figure 1.19 displays the discriminant functions in the form $g_i(x) = P(\omega_i)p(x|\omega_i)$, $i = 1, 2$, $x \in [0, 10]$.

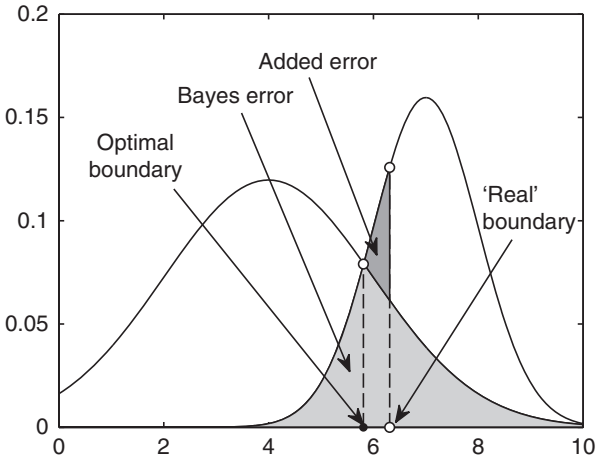


FIGURE 1.19 Plot of two discriminant functions $P(\omega_1)p(x|\omega_1)$ (left curve) and $P(\omega_2)p(x|\omega_2)$ (right curve) for $x \in [0, 10]$. The light-gray area corresponds to the Bayes error, incurred if the optimal decision boundary (denoted by •) is used. The dark-gray area corresponds to the additional error when another boundary (denoted by ∘) is used.

For two classes,

$$P(\omega_1|x) = 1 - P(\omega_2|x), \tag{1.33}$$

and $P_e(D^*)$ in Equation 1.28 becomes

$$P_e(D^*) = \int_{\mathcal{R}_1^*} (1 - P(\omega_1|\mathbf{x}))p(\mathbf{x})\, d\mathbf{x} + \int_{\mathcal{R}_2^*} (1 - P(\omega_2|\mathbf{x}))p(\mathbf{x})\, d\mathbf{x} \tag{1.34}$$

$$= \int_{\mathcal{R}_1^*} P(\omega_2|\mathbf{x})p(\mathbf{x})\, d\mathbf{x} + \int_{\mathcal{R}_2^*} P(\omega_1|\mathbf{x})p(\mathbf{x})\, d\mathbf{x} \tag{1.35}$$

$$= \int_{\mathcal{R}_1^*} P(\omega_2)p(\mathbf{x}|\omega_2)\, d\mathbf{x} + \int_{\mathcal{R}_2^*} P(\omega_1)p(\mathbf{x}|\omega_1)\, d\mathbf{x}. \tag{1.36}$$

By design, the classification regions of D^* correspond to the true highest posterior probabilities. The bullet on the x-axis in Figure 1.19 splits $\mathbb{R}$ into $\mathcal{R}_1^*$ (to the left) and $\mathcal{R}_2^*$ (to the right). According to Equation 1.36, the Bayes error will be the area under $P(\omega_2)p(\mathbf{x}|\omega_2)$ in $\mathcal{R}_1^*$ plus the area under $P(\omega_1)p(\mathbf{x}|\omega_1)$ in $\mathcal{R}_2^*$. The total area corresponding to the Bayes error is marked in light gray. If the boundary is

shifted to the left or right, additional error will be incurred. We can think of this boundary as coming from classifier D which is an imperfect approximation of D^*. The shifted boundary, depicted by an open circle, is called in this example the "real" boundary. Region $\mathcal{R}_1$ is therefore $\mathcal{R}_1^*$ extended to the right. The error calculated through Equation 1.36 is the area under $P(\omega_2)p(\mathbf{x}|\omega_2)$ in the whole of $\mathcal{R}_1$, and extra error will be incurred, measured by the area shaded in dark gray. Therefore, using the *true* posterior probabilities or an equivalent set of discriminant functions guarantees the smallest possible error rate, called the *Bayes error*.

Since the true probabilities are never available in practice, it is impossible to calculate the exact Bayes error or design the perfect Bayes classifier. Even if the probabilities were given, it will be difficult to find the classification regions in $\mathbb{R}^n$ and calculate the integrals. Therefore, we rely on estimates of the error as discussed in Section 1.3.

1.6 CLUSTERING AND FEATURE SELECTION

Pattern recognition developed historically as a union of three distinct but intrinsically related components: classification, clustering, and feature selection.

1.6.1 Clustering

Clustering aims to find groups in data. "Cluster" is an intuitive concept and does not have a mathematically rigorous definition. The members of one cluster should be similar to one another and dissimilar to the members of other clusters. A clustering algorithm operates on an unlabeled data set $\mathbf{Z}$ and produces a *partition* on it, denoted $P = (\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(c)})$, where $\mathbf{Z}^{(i)} \subseteq \mathbf{Z}$ and

$$\mathbf{Z}^{(i)} \cap \mathbf{Z}^{(j)} = \emptyset, \quad i,j = 1, \ldots, c, \quad i \neq j, \tag{1.37}$$

$$\bigcup_{i=1}^{c} \mathbf{Z}^{(i)} = \mathbf{Z}. \tag{1.38}$$

There is a vast amount of literature on clustering [18, 38, 126, 158, 195] looking for answers to the main questions, among which are:

- Is there really a structure in the data or are we imposing one by our clustering algorithms?
- How many clusters should we be looking for?
- How do we define similarity between objects in the feature space?
- How do we know whether our clustering results are good?

Two main groups of clustering algorithms are *hierarchical clustering* (agglomerative and divisive) and *nonhierarchical clustering*. The nearest neighbor (single

SINGLE LINKAGE CLUSTERING

1. Pick the number of clusters c and a similarity measure $\mathcal{S}(a, b)$ between two objects a and b. Initialize the procedure by defining an individual cluster for each point in $\mathbf{Z}$.
2. Identify the two most similar clusters and join them as a new cluster, replacing the initial two clusters. The similarity between clusters A and B is measured as
$$\min_{a \in A, b \in B} \mathcal{S}(a, b).$$
3. Repeat step 2 until c clusters are found.

FIGURE 1.20 The single linkage clustering algorithm.

c-MEANS CLUSTERING

1. Pick the number of clusters c and a similarity measure $\mathcal{S}(a, b)$ between two objects a and b. Initialize the c cluster centers (i.e., by randomly selecting c points from $\mathbf{Z}$ to be these centers).
2. Label all points in $\mathbf{Z}$ with respect to their similarity to the cluster centers: each point is assigned to the cluster with the most similar center.
3. Calculate the new cluster centers using the points in the respective cluster.
4. Repeat steps 2 and 3 until no change in the centers occurs.

FIGURE 1.21 The c-means (k-means) clustering algorithm.

linkage) clustering algorithm shown in Figure 1.20 is an example of the hierarchical group whereas the c-means clustering algorithm (also called k-means) shown in Figure 1.21 is an example of the nonhierarchical group. Both algorithms are famous for their simplicity and elegance.[8]

Example 1.14 Clustering: there is no "best" algorithm

Consider a two-dimensional data set where 50 points are sampled from each of two normal distributions with means at (0,0) and (3,3), and identity covariance matrices (Figure 1.22a). The single linkage clustering algorithm is known for the "chain effect." An outlier would often present itself as a separate cluster, thereby preventing the algorithm from discovering meaningful balanced clusters. This is illustrated in Figure 1.22b where the two clusters found by the algorithm are plotted with different markers. The c-means algorithm, on the other hand, identifies the two clusters successfully (Figure 1.22c).

[8] Both single-linkage and c-means algorithms are available in many statistical software packages, including the Statistics Toolbox of MATLAB.

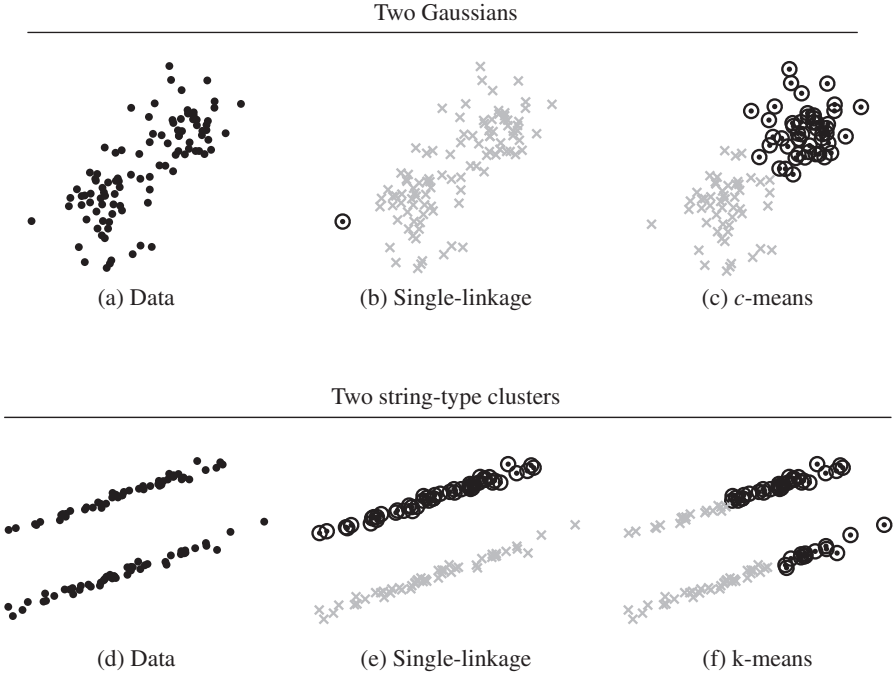


FIGURE 1.22 Examples of single linkage and k-means clustering on two synthetic data sets. The two clusters found by the algorithms are plotted with different markers: circles and gray crosses.

The second data set (Figure 1.22d) consists of two string-shaped clusters. This configuration is correctly identified by the single linkage but fools k-means into cutting both strings and finding nonexistent clusters.

Neither of the two algorithms is perfect, nor are the multitude of existing clustering algorithms. It may prove difficult to pick a suitable clustering algorithm for multi-dimensional data. Ensembles of "clusterers" are deemed to be more robust in that respect.

1.6.2 Feature Selection

Feature selection is the process of reducing the dimensionality of the feature space. Its aim is not only computational convenience but elimination of noise in the data so that it is easier to train an accurate and robust classifier. A myriad of insightful and comprehensive surveys, practitioners' guides, journal special issues, and conference tracks have been devoted to feature selection over the years [3, 42, 83, 164, 196, 214, 263, 284, 346]. Different methods and approaches have been recommended depending on the data types and sizes.

The two major questions that a feature selection method must address, separately or simultaneously, are:

1. Are the features evaluated individually? If not, how do we traverse the class of all subset-candidates?
2. What criterion do we apply to evaluate the merit of a given subset of features?

Consider for now question 1. The simplest way of selecting features is to rank them according to a certain test criterion and cut the list. Starting with key publications in the 1970s [77, 387], it is now well understood that features should be evaluated as a group rather than individually. By selecting the features individually, important dependencies may be overlooked. But evaluating *subsets* of features raises the question of computational complexity. If unlimited resources were available, exhaustive search could be carried out checking each and every possible subset. Sequential methods such as forward and backward selection, as well as floating search [315] have been found to be the best compromise between computation speed and accuracy. Figure 1.23 shows the sequential forward selection algorithm (SFS).

The output of SFS can be taken as the feature ranking determined by the order in which the features enter the set S in the algorithm.

The two basic approaches to question 2 are termed "wrapper" and "filter" [214]. The wrapper approach requires that a classifier model is chosen and trained on a given feature set. Its classification accuracy, evaluated on a validation set, is the measure of quality of that feature set. In the filter approach, some measure of separation between the classes in the space spanned by the feature set is used as a proxy for the classification accuracy. While the wrapper approach has been found to be generally more accurate, the filter approach is faster and easier to apply, which makes it a convenient compromise if a large number of feature subsets must be probed.

The MATLAB function `sfs_filter(a,laba,d)` in Appendix 1.A.3 carries out sequential forward selection of the features of data set a (columns of a) and returns the indices of the d features in order of selection. The criterion for evaluating the feature subset f is the Euclidean distance between the centroids of the classes in the space spanned by f but there are many alternatives offered by the `pdist` MATLAB function used within the code.

SEQUENTIAL FORWARD SELECTION (SFS)

1. Given is a feature set F. Choose a test criterion for a feature subset $f \subseteq F$ and a stopping criterion (e.g., a number of features $d \leq |F|$). Initialize the set of selected features $S = \emptyset$.
2. Taking all features in F that are not yet in S, add temporarily one feature at a time and measure the quality of the new set $S' = S \cup \{x\}, x \in F, x \notin S$.
3. Choose the feature with the highest criterion value and add it permanently to S.
4. Repeat steps 2 and 3 until the stopping criterion is met.

FIGURE 1.23 The sequential forward selection algorithm.

Example 1.15 Feature selection: the peak effect

The data set "sonar" from the UCI ML Repository has 60 features and 2 classes. SFS was applied for feature selection with a filter approach. The quality of a feature subset was measured by the Euclidean distance between the two class centroids in the respective feature space (as in the MATLAB function `sfs_filter(a,laba,d)` in Appendix 1.A.3). One hundred runs were carried out where a randomly sampled half of the data set was used for training and the other half, for testing. The nearest neighbor classifier was applied for evaluating the selected feature subsets. Each split of the data produced a ranking of the 60 features. As an example, suppose that SFS on split j arranged the features as $\{32, 11, 6, 28, \ldots\}$. For this split, feature 32 was the single best, $\{32, 11\}$ was the best pair containing feature 32, $\{32, 11, 6\}$ was the best set of three features containing features 32 and 11, and so on.

Consider plotting the classification accuracy, evaluated on the testing half of the data, when using feature sets $\{32\}$, $\{32, 11\}$, $\{32, 11, 6\}$, $\{32, 11, 6, 28\}$, and so on. It can be expected that the more features we include, the higher the accuracy will be, leading to the best accuracy with all features. The curves for the 100 data splits are shown in Figure 1.24 in gray. The average curve is depicted with a solid black line. The peak and the end points are marked and annotated. It can be seen that SFS reaches better accuracy with fewer features, called the "peak effect." For comparison, 100 random permutations of the features were generated, and an average curve was calculated in the same way as with the SFS rankings. As expected, the average random curve progresses gradually toward the same end point but without the peak effect.

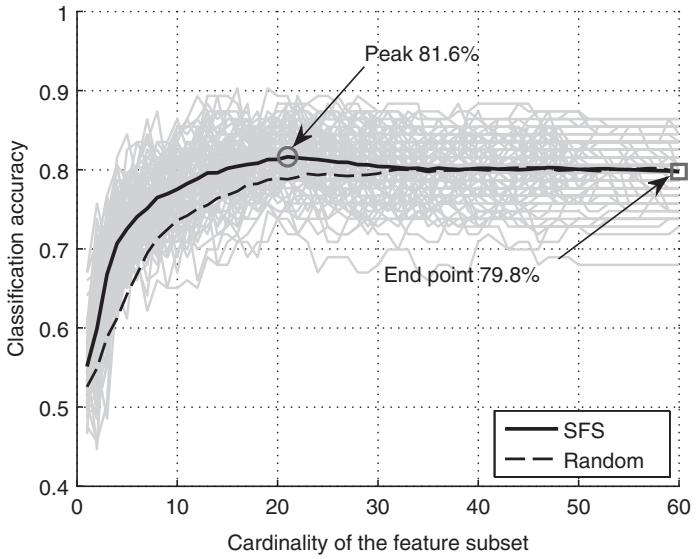


FIGURE 1.24 Illustration of the peak effect in feature selection on the "sonar" data, SFS filter and the nearest neighbor classifier.

This example demonstrates that feature selection could be beneficial not just for reducing computational complexity but also for increasing the classification accuracy. This problem is especially acute for very high-dimensional data where the number of features exceeds by orders of magnitude the number of samples, the so-called "wide" data sets.

The long-lasting and proliferate research on feature selection has delivered a refined collection of excellent feature selection algorithms such as the floating search [315], fast correlation-based filters (FCBF) [428], RELIEF [205], and SVM-RFE [165]. Yet, with the new challenges posed by the very high dimensionality of modern data, there is room for development. Saeys et al. [346] highlight the potential of ensemble feature selection methods to improve the stability and accuracy of the individual methods. We will touch upon feature selection *for ensembles* and *by ensembles* further in the book.

1.7 CHALLENGES OF REAL-LIFE DATA

Finally, pattern recognition branched out tremendously in the past couple of decades, taking what were curious little niches in the past into powerful independent research streams in their own right. Real-life data pose challenges such as

- *Unbalanced classes.* Many times the class of interest is like a "needle in a haystack." An example is detecting a face in an image. Suppose that the gray image has 500 rows and 600 columns of pixels, and a face is expected to be within a 50-by-50 square of pixels. Then there are $(500 - 49) \times (600 - 49) = 248,501$ candidate squares. If the image is a photograph of a person, the class "face" will contain a handful of objects (squares containing predominantly the face), and class "nonface" will contain all remaining squares. Thus class "face" will be a minute fraction of the data.
- *Uncertain labels.* Sometimes the labels of the objects cannot be assigned precisely. Take, for example, emotion recognition. Affective computing is gaining importance in psychological research, entertainment, and gaming industries. However, it is hardly possible to pinpoint and label the experienced emotion.
- *Massive volumes.* Computational costs, algorithmic tractability, and statistical validity of the results are only a few of the problems with very high-dimensional data and massive sample sizes.
- *Nonstationary distributions.* The data set collected at a certain time may become obsolete if the circumstances or the problem characteristic change. Adaptive classification is needed for such cases.

Standard and custom-tailored classifier ensemble methods are quickly turning into one of the most favorite tools in all these areas.

APPENDIX

1.A.1 DATA GENERATION

```
%-----------------------------------------------------------------------%
function x = samplegaussian(N,mu,Sigma)
mu = mu(:); R = chol(Sigma);
for i = 1:N
    x(i,:) = mu' + (R'*randn(size(mu)))';
end
%-----------------------------------------------------------------------%
```

```
%-----------------------------------------------------------------------%
% Subplot (a) ---
% Introduce the ellipse function
elx = @(t,xc,a,b,phi) xc+a*cos(t)*cos(phi)-b*sin(t)*sin(phi) ;
ely = @(t,yc,a,b,phi) yc+a*cos(t)*sin(phi)+b*sin(t)*cos(phi) ;

% Calculate the  ellipse equations
N = 500;
t = rand(1,N)*2*pi; % sample random points from the figure
el1x = elx(t,-6,2,6,-2); el1y = ely(t,0,2,6,-2);  % ellipse 1
el2x = elx(t,-2,4,3,-1); el2y = ely(t,-2,4,3,-1); % ellipse 2
el3x = elx(t,2,4,1,0.9); el3y = ely(t,4,4,1,0.9); % ellipse 3

% Add noise
edata = [el1x(:), el1y(:); el2x(:), el2y(:); el3x(:), el3y(:)];
edata = edata + randn(size(edata))*0.5;
w = ones(numel(el1x),1);
elabels = [w;w*2;w*3];

% Plot the data
figure, hold on
scatter(edata(:,1),edata(:,2),[],elabels,'linewidth',2.5)
axis equal off

% Subplot (b) ---
t = rand(1,1000)*2*pi; % sample random points from the figure
el1x = elx(t,0,3,9,-1); el1y = ely(t,0,3,9,-1);
el1x = el1x + randn(size(el1x)).*t*.2;
el1y = el1y + randn(size(el1y)).*t*.2;
figure
plot(el1x,el1y,'k.','markersize',15);
axis equal off
%-----------------------------------------------------------------------%
```

```
%-------------------------------------------------------------------%
function [d, labd] = samplecb(N,a,alpha)
d = rand(N,2);
d_transformed = [d(:,1)*cos(alpha)-d(:,2)*sin(alpha),...
    d(:,1)*sin(alpha)+d(:,2)*cos(alpha)];
s = ceil(d_transformed(:,1)/a)+floor(d_transformed(:,2)/a);
labd = 2 - mod(s,2);
%-------------------------------------------------------------------%
```

1.A.2 COMPARISON OF CLASSIFIERS

1.A.2.1 MATLAB Functions for Comparing Classifiers

The output of all hypothesis-testing functions is in the form [*H*,*p*], where *H* is 0 if the null hypothesis is accepted, and 1 if the null hypothesis is rejected at significance level 0.05. The output *p* is the test *p*-value.

```
%-------------------------------------------------------------------%
function [H,p] = mcnemar(labels1, labels2, true_labels)
% --- McNemar test for two classifiers
% Needs Statistics Toolbox
% (all labels are integers 1,2,...)
v1 = labels1(:) == true_labels(:);
v2 = labels2(:) == true_labels(:);
t2(1,1) = sum(~v1&~v2);t2(1,2) = sum(~v1&v2);
t2(2,1) = sum(v1&~v2);t2(2,2) = sum(v1&v2);
% the two-way table [N00,N01;N10,N11]
% calculate the test statistic
if any([t2(1,2),t2(2,1)])
    if t2(1,2) + t2(2,1) > 25
        x2 = (abs(t2(1,2)-t2(2,1))-1)^2/(t2(1,2)+t2(2,1));
        % find the p-value
        p = 1 - chi2cdf(x2,1);
    else % exact test using binomial distribution
        % t2(1,2) is compared to a binomial distribution
        % with size parameter equal to t2(1,2) + t2(2,1)
        % and "probability of success" = 0.5,
        p=binocdf(min(t2(1,2),t2(2,1)),t2(1,2)+t2(2,1),0.5)...
           + 1 - binocdf(max(t2(1,2),t2(2,1))-1,t2(1,2)+...
                t2(2,1),0.5);
    end
else % identical classifiers
    p = 1;
end
```

```
% calculate the hypothesis outcome at significance level 0.05
% H = 0 if the null hypothesis holds; H = 1 otherwise.
H = p < 0.05;
%-------------------------------------------------------------------%
```

```
%-------------------------------------------------------------------%
function [H,p] = tvariance(x,y,ts_tr_ratio)
% --- paired t-test with corrected variance
% Needs Statistics Toolbox
d = x - y;
md = mean(d); stdd = std(d);
se_corrected = stdd * sqrt(1/numel(x) + ts_tr_ratio);
t = md / se_corrected; % the test statistic
% two-tailed test
p = 2 * tcdf(-abs(t),numel(x)-1);
% calculate the hypothesis outcome at significance level 0.05
% H = 0 if the null hypothesis holds; H = 1 otherwise.
H = p < 0.05;
%-------------------------------------------------------------------%
```

```
%-------------------------------------------------------------------------%
function [H,p] = imandavenport(a)
% --- Iman and Davenport test for N classifiers on M data sets
% Needs Statistics Toolbox
% a_ji is the error of model j on data set i
% N rows, M columns
[N,M] = size(a);

r = ranks(a')'; R = mean(r);
x2F =12*N/(M*(M+1))*(sum(R.^2) - M*(M+1)^2/4);

% ===
% MATLAB Stats Toolbox variant with additional correction
% for tied ranks:
% [~,t] = friedman(a,1,'off');
% x2F = t{2,5} % Friedman chi^2 statistic
% ===

FF = (N-1) * x2F / (N*(M-1) - x2F); % amended
p = 1 - fcdf(FF,(M-1),(M-1)*(N-1)); % p-value from the F-distribution
% calculate the hypothesis outcome at significance level 0.05
% H = 0 if the null hypothesis holds; H = 1 otherwise.
H = p < 0.05;
end

function ran = ranks(a)
```

```
[maxr,maxcol] = size(a);
ran = zeros(size(a));
for i = 1:maxcol
    [~, rr] = sort(a(:,i)); [~, b2] = sort(rr);
    for j = 1 : maxr % check for ties
        inr = a(:,i) == a(j,i);
        b2(inr) = mean(b2(inr));
    end
    ran(:,i) = b2;
end
end
%-------------------------------------------------------------------------%
```

```
%-------------------------------------------------------------------%
function [H,p] = nemenyiposthoc(a)
% --- Nemenyi post-hoc test
% Needs Statistics Toolbox & function RANKS
% a_ji is the error of model j on data set i
% N rows, M columns
[N,M] = size(a);
r = ranks(a')'; R = mean(r);
const = M * (M-1) / 2;
for i = 1:M-1
    for j = i+1:M
        z = (R(i)-R(j))/sqrt(M*(M+1)/(6*N));
        p(i,j) = 2*normcdf(-abs(z)); % two-tailed test
        p(j,i) = p(i,j);
    end
    p(i,i) = 1;
end
p(M,M) = 1;
p = min(1,p*const);

% calculate the hypothesis outcome at significance level 0.05
% H = 0 if the null hypothesis holds; H = 1 otherwise.
H = p < 0.05;
end
%-------------------------------------------------------------------%
```

```
%-------------------------------------------------------------------%
function [H,p] = bonferroniposthoc(a)
% --- Bonferroni-Dunn post-hoc test
% Needs Statistics Toolbox & function RANKS
% a_ji is the error of model j on data set i
```

```
% N rows, M columns
% The classifier of interest is in column 1 of a.
% The output contains M-1 results from the
% comparisons of columns 2:M with column 1

[N,M] = size(a);

r = ranks(a')'; R = mean(r);
const = M - 1;

for i = 2:M
    z = (R(1)-R(i))/sqrt(M*(M+1)/(6*N));
    % p(i-1) = 2*normcdf(-abs(z)); % two-tailed test
    p(i-1) = normcdf(z); % one-tailed test
end
p = min(1,p*const);

% calculate the hypothesis outcome at significance level 0.05
% H = 0 if the null hypothesis holds; H = 1 otherwise.
H = p < 0.05;
end
%-------------------------------------------------------------------%
```

1.A.2.2 Critical Values for Wilcoxon and Sign Test

Table 1.A.1 shows the critical values for comparing two classifier models on N data sets. Sub-table (a) gives the values for the Wilcoxon signed rank test (two-tailed), and sub-table (b), the values for the sign test (one-tailed). For sub-table (b), classifier model A is better than B if it wins on w_α or more data sets. Here we explain the calculation of the critical values for the sign test.

Suppose that in comparing classifier models A and B, both were tested on N data sets. Model A was found to be better on K out of the N sets. Can we claim that A is better than B? The null hypothesis of our test, H_0, is that there is no difference between A and B. Then the probability that A wins over B on a randomly chosen data set is $1/2$. The number of data sets where A wins over B in N attempts follows a binomial distribution with parameters N and $1/2$. Under the null hypothesis, the probability that A wins on K or fewer data sets is

$$F_{\mathrm{b}}(K, N, 0.5) = 0.5^N \sum_{i=0}^{K} \frac{N!}{i!(N-i)!}, \tag{1.A.1}$$

where F_{b} is the cumulative distribution function of the binomial distribution. The alternative hypothesis, H_1, is that A is better than B (one-tailed test). To reject the

TABLE 1.A.1 Table of the Critical Values for Comparing Two Classifier Models on N Data Sets. (a) Wilcoxon Signed Rank Test (Two Tailed), (b) Sign Test (One Tailed). For Sub-table (b), Classifier Model A Is Better Than B if It Wins on w_α or More Data Sets

	(a) α				(b) α	
N	0.1	0.05	0.01	N	$w_{0.10}$	$w_{0.05}$
				5	5	5
6	0	–	–	6	6	6
7	2	0	–	7	6	7
8	4	2	0	8	7	7
9	6	3	2	9	7	8
10	8	5	3	10	8	9
11	11	7	5	11	9	9
12	14	10	7	12	9	10
13	17	13	10	13	10	10
14	21	16	13	14	10	11
15	25	20	16	15	11	12
16	30	24	20	16	12	12
17	35	28	23	17	12	13
18	40	33	28	18	13	13
19	46	38	32	19	13	14
20	52	43	38	20	14	15
21	59	49	43	21	14	15
22	66	56	49	22	15	16
23	73	62	55	23	16	16
24	81	69	61	24	16	17
25	89	77	68	25	17	18

null hypothesis and accept H_1 at level of significance α, K must be large enough. For example, let $N = 5$. Then $F_{\mathrm{b}}(K, N, 0.5)$ is

K	0	1	2	3	4	5
$F_{\mathrm{b}}(K, 5, 0.5)$	0.0313	0.1875	0.5000	0.8125	0.9688	1.0000

Because of the discrete nature of the problem, we cannot achieve the desired level of significance exactly. If the null hypothesis is correct, the probability of observing K or more wins (A better than B) is

$$1 - F_{\mathrm{b}}(K - 1, 5, 0.5). \tag{1.A.2}$$

We must set the critical value of K so that this probability is smaller or equal to α. Any number of wins greater than or equal to this critical value will allow us to reject

the null hypothesis at the desired level of significance α or better. Let $\alpha = 0.05$. We have

$$1 - F_{\mathrm{b}}(4, 5, 0.5) = 0.0313 \tag{1.A.3}$$

and

$$1 - F_{\mathrm{b}}(3, 5, 0.5) = 0.1875. \tag{1.A.4}$$

Then the critical value K^* is obtained from $K^* - 1 = 4$, hence $K^* = 5$.

Therefore, to construct the table with the critical values, we find

$$K' = \arg \min_{0 \le K \le N-1} \left\{ F_b(K, N, 0.5) \ge 1 - \alpha \right\}, \tag{1.A.5}$$

and set $K^* = K' + 1$ as the critical value.

1.A.3 FEATURE SELECTION

```
%-------------------------------------------------------------------------%
function S = sfs_filter(a,laba,d)
% --- Sequential Forward Selection - filter approach
% a - data set
% laba - labels 1,2,3,...,
% d - desired number of features
% S - indices of the selected features
%     (in order of selection)

c = max(laba); % number of classes
n = size(a,2); % number of features
F = ones(1,n); % features to choose from
S = []; % chosen subset (empty)

% calculate class means
x = zeros(c,n);
for k = 1:c
    x(k,:) = mean(a(laba == k,:),1);
end

for i = 1:d
    Remaining = find(F); % features not selected yet
    for j = 1:numel(Remaining)
        Sdash = S;
        % temporarily add one feature
```

```
        Sdash = [Sdash Remaining(j)];
        % calculate the criterion
        crit(j) = mean(pdist(x(:,Sdash)));
    end
    % choose the best feature to add
    [~,best] = max(crit);
    S = [S Remaining(best)]; % add the best feature
    F(Remaining(best)) = 0; % remove from F
end
%-------------------------------------------------------------------%
```