(0.9, 0.6), (0.9, 0.8). The dotted circle is the one that encircles all five points, and its radius is equal to $\sqrt{0.2^2 + 0.2^2} = 0.2\sqrt{2} = 2\rho$.

There are $N_1 = 59$ points in class ω_1 and $N_2 = 61$ in class ω_2. The areas (volumes) of the two circles are $V_1 = 4\pi\rho^2$ and $V_2 = \pi\rho^2$, respectively, for the two classes. Hence, according to Eq. (2.116) and ignoring the risk related terms, we have

$$\frac{V_2}{V_1} = \frac{\pi\rho^2}{4\pi\rho^2} = 0.25$$

and since 0.25 is less than 59/61 and the classes are equiprobable, the point (0.7, 0.6) is classified to class ω_2.

2.5.7 The Naive-Bayes Classifier

The goal in this section, so far, was to present various techniques for the estimation of the probability density functions $p(\boldsymbol{x}|\omega_i),\ i = 1, 2, \ldots, M$, required by the Bayes classification rule, based on the available training set, X. As we have already stated, in order to safeguard good estimates of the pdfs the number of training samples, N, must be large enough. To this end, the demand for data increases exponentially fast with the dimension, l, of the feature space. Crudely speaking, if N could be regarded as a good number of training data points for obtaining sufficiently accurate estimates of a pdf in an one-dimensional space, then N^l points would be required for an l-dimensional space. Thus, large values of l make the accurate estimation of a multidimensional pdf a bit of an "illusion" since in practice data is hard to obtain. Loosely speaking, data can be considered to be something like money. It is never enough! Accepting this reality, one has to make concessions about the degree of accuracy that is expected from the pdf estimates. One widely used approach is to assume that individual features $x_j,\ j = 1, 2, \ldots, l$, are statistically independent. Under this assumption, we can write

$$p(\boldsymbol{x}|\omega_i) = \prod_{j=1}^{l} p(x_j|\omega_i), \quad i = 1, 2, \ldots, M$$

The scenario is now different. To estimate l one-dimensional pdfs, for each of the classes, lN data points would be enough in order to obtain good estimates, instead of N^l. This leads to the so-called *naive-Bayes* classifier, which assigns an unknown sample $\boldsymbol{x} = [x_1, x_2, \ldots, x_l]^T$ to the class

$$\omega_m = \arg\max_{\omega_i} \prod_{j=1}^{l} p(x_j|\omega_i), \quad i = 1, 2, \ldots, M$$

It turns out that the naive-Bayes classifier can be very robust to violations of its independence assumption, and it has been reported to perform well for many real-world data sets. See, for example, [Domi 97].

Example 2.10

The discrete features case: In Section 2.2, it was stated that in the case of discrete-valued features the only required change in the Bayesian classification rule is to replace probability density functions with probabilities. In this example, we will see how the associated with the naive Bayes classifier assumption of statistical independence among the features simplifies the Bayesian classification rule.

Consider the feature vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_l]^T$ with binary features, that is, $x_i \in \{0, 1\}$, $i = 1, 2, \ldots, l$. Also let the respective class-conditional probabilities be $P(x_i = 1|\omega_1) = p_i$ and $P(x_i = 1|\omega_2) = q_i$. According to the Bayesian rule, given the value of $\boldsymbol{x}$, its class is decided according to the value of the likelihood ratio

$$\frac{P(\omega_1)P(\boldsymbol{x}|\omega_1)}{P(\omega_2)P(\boldsymbol{x}|\omega_2)} > (<) 1 \tag{2.119}$$

for the minimum probability error rule (the minimum risk rule could also be used).

The number of values that $\boldsymbol{x}$ can take, for all possible combinations of x_i, amounts to 2^l. If we do not adopt the independence assumption, then one must have enough training data in order to obtain probability estimates for each one of these values (probabilities add to one, thus $2^l - 1$ estimates are required). However, adopting statistical independence among the features, we can write

$$P(\boldsymbol{x}|\omega_1) = \prod_{i=1}^{l} p_i^{x_i}(1 - p_i)^{1-x_i}$$

and

$$P(\boldsymbol{x}|\omega_2) = \prod_{i=1}^{l} q_i^{x_i}(1 - q_i)^{1-x_i}$$

Hence, the number of required probability estimates is now $2l$, that is, the p_i's and q_i's. It is interesting to note that, taking the logarithm of both sides in (2.119), one ends up with a *linear* discriminant function similar to the hyperplane classifier of Section 2.4, that is,

$$g(\boldsymbol{x}) = \sum_{i=1}^{l} \left(x_i \ln \frac{p_i}{q_i} + (1 - x_i) \ln \frac{1 - p_i}{1 - q_i} \right) + \ln \frac{P(\omega_1)}{P(\omega_2)} \tag{2.120}$$

which can easily be brought into the form of

$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 \tag{2.121}$$

where

$$\boldsymbol{w} = \left[\ln \frac{p_1(1 - q_1)}{q_1(1 - p_1)}, \ldots, \ln \frac{p_l(1 - q_l)}{q_l(1 - p_l)} \right]^T$$

and

$$w_0 = \sum_{i=1}^{l} \ln \frac{1 - p_i}{1 - q_i} + \ln \frac{P(\omega_1)}{P(\omega_2)}$$

Binary features are used in a number of applications where one has to decide based on the presence or not of certain attributes. For example, in medical diagnosis, 1 can represent a normal value in a medical test and a 0 an abnormal one.

2.6 THE NEAREST NEIGHBOR RULE

A variation of the *k*NN density estimation technique results in a *suboptimal*, yet popular in practice, nonlinear classifier. Although this does not fall in the Bayesian framework, it fits nicely at this point. In a way, this section could be considered as a bridge with Chapter 4. The algorithm for the so-called *nearest neighbor rule* is summarized as follows. Given an unknown feature vector $\boldsymbol{x}$ and a distance measure, then:

- Out of the N training vectors, identify the k nearest neighbors, *regardless* of class label. k is chosen to be odd for a two class problem, and in general not to be a multiple of the number of classes M.
- Out of these k samples, identify the number of vectors, k_i, that belong to class $\omega_i, i = 1, 2, \ldots, M$. Obviously, $\sum_i k_i = k$.
- Assign $\boldsymbol{x}$ to the class ω_i with the maximum number k_i of samples.

Figure 2.25 illustrates the k-NN rule for the case of $k = 11$. Various distance measures can be used, including the Euclidean and Mahalanobis distance.

The simplest version of the algorithm is for $k = 1$, known as the *nearest neighbor (NN)* rule. In other words, a feature vector $\boldsymbol{x}$ is assigned to the class of its nearest neighbor! Provided that the number of training samples is large enough, this simple

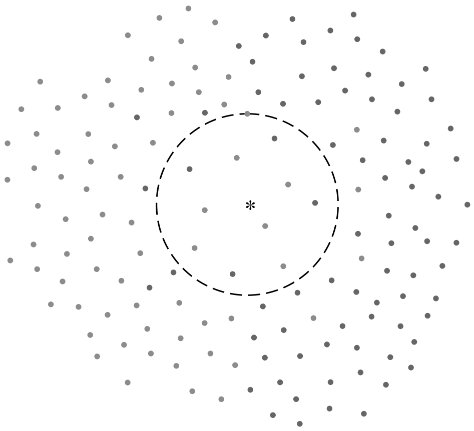

FIGURE 2.25

Using the 11-NN rule, the point denoted by a "star" is classified to the class of the red points. Out of the eleven nearest neighbors seven are red and four are black. The circle indicates the area within which the eleven nearest neighbors lie.