Chapter 2 - Pattern Recognition

The goal of this chapter is to provide the basic understanding of:

- Statistical pattern recognition
- Training of classifiers

Go to next section

2.1 The Pattern Recognition Problem

The human ability to find patterns in the external world is ubiquitous. It is at the core of our ability to respond in a more systematic and reliable manner to external stimuli. Humans do it effortlessly, but the mathematics underlying the analysis and design of pattern-recognition machines are still in their infancy. In the 1930's R.A. Fisher laid out the mathematical principles of statistical pattern recognition, which is one of the most rigorous ways to formulate the problem.

A real-world example will elucidate the principles of statistical pattern recognition at work: Assume that body temperature is used as an indicator of the health of a patient. Experience shows that in the healthy state the body regulates its temperature near 37° Celsius (98.6° F; the low end of normality will not be considered for the sake of simplicity). With viral or bacterial infections the body temperature rises. Any measurement can be thought of as a point in a space called the pattern space, or the *input space* (one dimensional in our example). So if we plot the temperature of individuals on a line (Figure 2-1), we can verify that the region close to 37°C is assigned to healthy individuals and the higher-temperature region is assigned to sick individuals. This natural distribution of points leads to the definition of classes or *category regions* in pattern space. The goal of pattern recognition is to build machines, called classifiers, that will automatically assign measurements to classes.

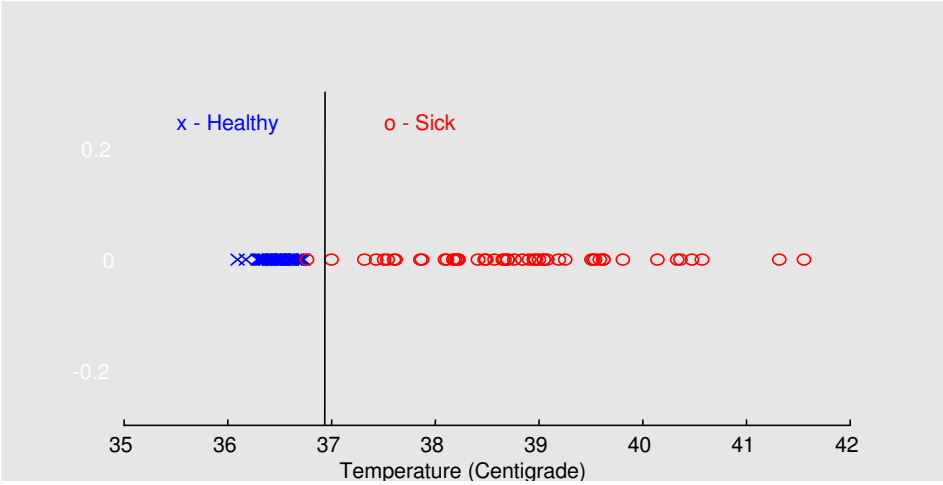


Figure 2-1 The sick/healthy problem in pattern space

A natural way to make the class assignment is to define the *boundary* temperature between sick and healthy individuals. This boundary is called the decision surface. The decision surface is not trivially determined for many real-world problems. If we start measuring the temperature of healthy subjects with a thermometer, we will soon find out that individual temperatures vary from subject to subject, and change for the same subject depending on the hour of the day, the subject's state (e.g. at rest or after exercise), and so on. The same variability occurs in sick individuals (aggravated by the seriousness and type of illness), and there may be overlap

individuals (aggravated by the seriousness and type of illness), and there may be overlap between the temperature of sick and healthy individuals. Thus, we immediately see that the *central problem in pattern recognition is to define the shape and placement of the boundary* so that the class-assignment errors are minimized.

2.1.1 Can Regression be Used for Pattern Recognition?

In Chapter 1 we presented a methodology that builds adaptive machines with the goal of fiting hyperplanes to data points. A legitimate question is to ask whether regression can be used to solve the problem of separating data into classes. The answer is no because the goals are very different.

- In regression, both the input data and desired response are experimental variables (normally real numbers) created by a *single unknown, underlying mechanism*.
- The goal in regression is to find the parameters of the best linear approximation to the input and the desired response pairs.

The regression problem is therefore one of *representing the relationship* between the input and the desired response.

In classification the issue is very different. We accept *a priori* that different input data may be generated by different mechanisms and that *the goal is to separate the data as well as possible into classes*. The desired response is a set of arbitrary labels (a different integer is normally assigned to each one of the classes), so every element of a class will share the same label. Class assignments are mutually exclusive, so a classifier needs a nonlinear mechanism such as an all-or-nothing switch. At a high level of abstraction, both the classification and the regression problems seek systems that transform inputs into desired responses, but the details of the mappings are rather different in the two cases.

We can nevertheless *use the machinery* utilized in linear regression, the Adaline and the LMS rule, as pieces to build pattern classifiers. Let us see how we can do this in NeuroSolutions and what the results are.

NEUROSOLUTIONS EXAMPLE 2.1
Comparing Regression and Classification

Suppose we are given the healthy and sick data, and we arbitrarily assign the value 1 as the desired system response to the healthy class, and the –1 desired response to the sick class. With these assignments we can train the Adaline of Chapter I to fit the input/desired response pairs.

The important question is to find out what the solution means. Notice that for equal numbers of sick and healthy cases, the regression line intersects the temperature line at the mean temperature of the overall data set (healthy and sick cases), which is the centroid of the observations. The regression line is not directly useful for classification. However, one can place a threshold function at the output of the Adaline such that when its output is positive the response will be 1 (healthy), and when it is negative, the response is –1 (sick).

Now we have a classifier, but this does not change the fact that the placement of the regression line was dictated by the linear fit of the data and not by the requirement to separate the two classes as well as possible to minimize the classification errors. With the arrangement of an Adaline followed by a threshold, we have

created our first classifier. But how can we improve its performance, estimate the optimal error rate, and extend it to multiple classes?

NeuroSolutions Example

The Adaline can be applied for classification when the system topology is extended with a threshold as a decision device. However, there is no guarantee of good performance, because the coefficients are being adapted to fit (in the least square sense) the temperature data to the labels 1 and -1, and not to minimize the classification error. This is an especially simple example with only two classes. For the multiple-class case the results become even more fragile. The conclusion is that we need a new methodology to study and design accurate classifiers. The algorithms we developed in Chapter 1, however, will be the basis for much of our future work. All the concepts learning curves, rattling, step sizes, and so on will be applicable.

Go to next section

2.2 Parametric Classifiers

2.2.1 Optimal Decision Boundary Based on Statistical Models of Data

The healthy/sick classification problem can be modeled in the following way: Assume that temperature is a random variable (i.e., a quantity governed by probabilistic laws) generated by two different phenomena, health and sickness, and further assume a probability density function (pdf) for each phenomenon (usually a Gaussian distribution). From the temperature measurements we can obtain the statistical parameters needed to fit the assumed pdf to the data (for Gaussians only the mean and variance need to be estimated; - see Appendix A). Statistical decision theory proposes very general principles to construct the optimal classifier. Fisher showed that the optimal classifier chooses the class c_i that maximizes the *a posteriori* probability $P(c_i|x)$ that the given sample x belongs to the class; that is,

$$x \text{ belongs to } c_i \text{ if } P(c_i|x) > P(c_j|x) \text{ for all } j \neq i \qquad \textbf{(2.1)}$$

The problem is that the *a posteriori* probability cannot be measured directly. However, using Bayes' rule,

$$P(c_i|x) = \frac{p(x|c_i)P(c_i)}{P(x)} \qquad \textbf{(2.2)}$$

one can compute the *a posteriori* probability from $P(c_i)$ the *prior probability* of the classes, multiplied by $p(x|c_i)$, the likelihood that the data x was produced by class c_i and normalized by the probability of $P(x)$. Both $P(c_i)$ and the likelihood can be estimated from the collected data and the assumption of the pdf. $P(x)$ is a normalizing factor that can be left out in most classification cases. understanding Bayes rule

For our example i =1, 2 (healthy, and sick), and $P(c_i)$ can be estimated from the demographics, season, and so on. Figure 2-1 shows data collected from 100 cases. The likelihoods $p(x|c_i)$ can be estimated assuming a Gaussian distribution,

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{(x-\mu)^2}{\sigma^2}\right)} \qquad \textbf{(2.3)}$$

and estimating the means μ_i and standard deviations σ_i of the distributions for sick and healthy individuals from the data. Using the sample mean and variance (N is the number of measurements)

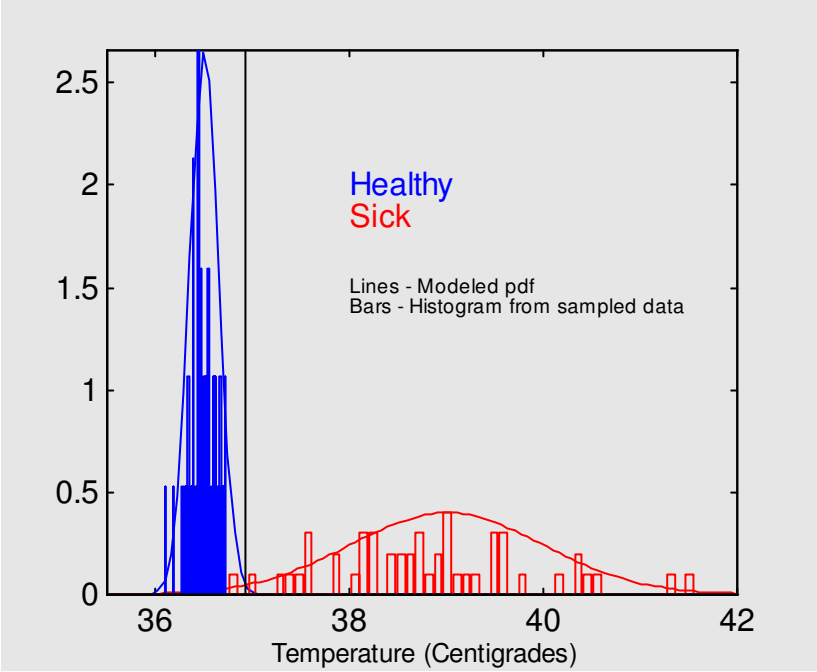
$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 \qquad \textbf{(2.4)}$$

for this data set gives the results of Table 2.1

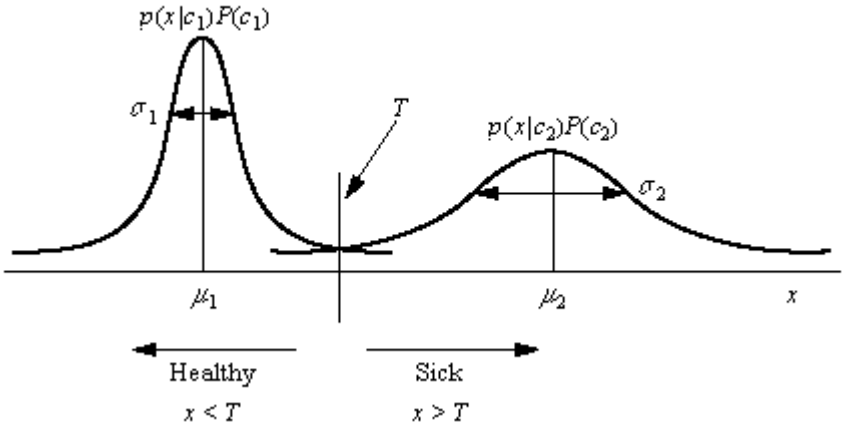
Table 2-1 Statistical Measures for Figure 2-1 Data

Temperature	1,000 Measurements	100 Measurements
Healthy	Mean = 36.50 Standard Deviation = 0.15	Mean = 36.49 Standard Deviation = 0.14
Sick	Mean = 39.00 Standard Deviation = 1	Mean = 38.83 Standard Deviation = 1.05

The separation boundary, that is, the temperature x = T for which the two *a posteriori* probabilities are identical, can be computed for the one-dimensional case with simple algebra. In this case the optimal threshold is T = 37 C (Figure 2-2). Bayesian threshold



(a)



(b)

Figure 2-2(a) Sampled data distributions, (b) Bayes threshold T

It is rather easy to optimally classify healthy/sick cases using this methodology. Given a temperature *x* from an individual, we compute Eq. 2.2 for both classes and assign the label healthy or sick according to the one that produces the largest value (see Eq. 2.1). Alternatively, we can compare the measurement to T and decide immediately to use the label healthy if $x < T$, or sick if $x > T$. Notice that to the left of T the scaled likelihood of class healthy is larger than for the class sick, so measurements that fall in this area are more likely produced by healthy subjects and should be assigned to the healthy class. Similarly, the measurements that fall toward the right have a higher likelihood of being produced by sick cases.

The class assignment is not error free. In fact, the tail of the healthy likelihood extends to the right of the intercept point, and the tail of the sick likelihood extends to the left of T. The error in the classification is given by the sum of the areas under these tails, so the smaller the overlap the better the classification accuracy. The maximum *a posteriori* probability assignment (Eq. 2.1) minimizes this probability of error (minimum error rate) and is therefore optimal.

Metric for Classification

There are important conclusions to be taken from the preceeding example. For a problem with given class variances, if we increase the distance between the class means, the overlap will decrease and the classification becomes more accurate. This is reminiscent of the distance in Euclidean space when we think of the class centers as two points in space. However, *we cannot just look at the class means to estimate the classification error*, since the error depends on the overlap between the class likelihoods. The tails of the Gaussians are controlled by the class variance, so we can have cases where the means are very far apart, but the variances are so large that the overlap between likelihoods is still high. Inversely, the class means can be close to each other, but if the class variances are very small the classification can still be done with small error.

Hence, separability between Gaussian-distributed classes is a function of both the mean and the variance of each class. As we saw in the Bayesian threshold the placement of the decision surface is determined by the class distance normalized by the class variances. We can encapsulate this idea by saying that *the metric for classification is not Euclidean*, but also involves the dispersion (variance) of each class. If we closely analyze the exponent for the Gaussian distribution (Eq. 2.3), we can immediately see that the value of the function depends not only on μ but also on σ. The value of *p(x)* depends on the distance of *x* from the mean normalized by the variance. This distance is called the Mahalanobis distance.

Following this simple principle of estimating *a posteriori* probabilities, an optimal classifier can be built that is able to use temperature to discriminate between healthy and sick subjects. Once again, optimal does not mean that the process will be error free, only that the system will minimize the number of mistakes when the variable temperature is utilized.

2.2.2 Discriminant Functions

Assume we have N measurements $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_N$, where each measurement $\mathbf{x}_k$ is a vector with D components

$$\mathbf{x}_k = [x_{k1}, x_{k2}, ..., x_{kD}] \qquad \textbf{(2.5)}$$

and can be visualized as a point in the D-dimensional pattern space. Following Eq. 2.1, the class assignment by Bayes' rule is based on a comparison of likelihoods scaled by the corresponding *a priori* probability. Alternatively, the measurement $\mathbf{x}_k$ will be assigned to class *i* if

$$g_i(\mathbf{x}_k) > g_j(\mathbf{x}_k) \text{ for all } j \neq i \qquad \textbf{(2.6)}$$

Each scaled likelihood can be thought of as a discriminant function $g(\mathbf{x})$, that is, a function that assigns a "score" to every point in the input space. Each class has its individual scoring function, yielding higher values for the points that belong to the class. Discriminant functions intersect in the input space defining a decision surface, where the scores are equal (Figure 2-3). *Thus, decision surfaces partition the input space into regions where one of the discriminants is larger than the others.* Each region is then assigned to the class associated with the largest discriminant.

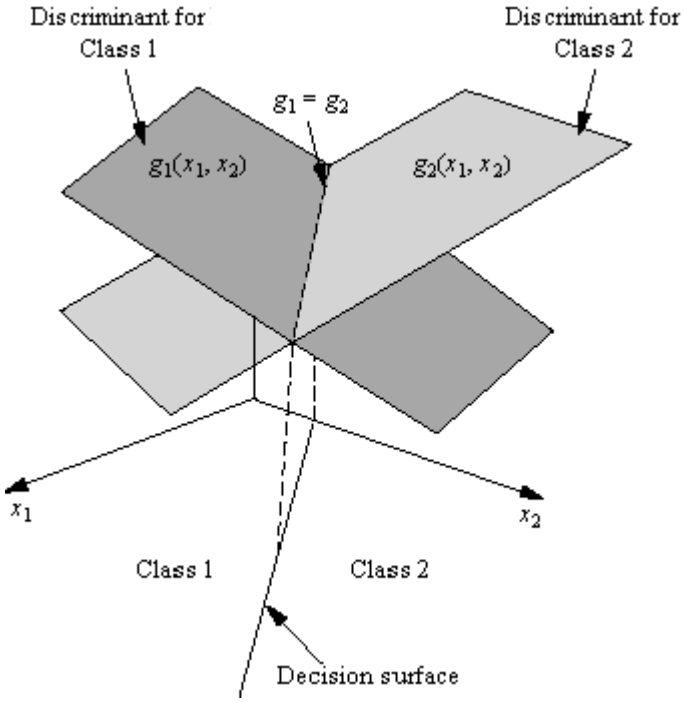


Figure 2-3 Discriminant functions and the decision surface.

In this view the optimal classifier just compares discriminant functions (one per class) and chooses the class according to the discriminant $g_i(\mathbf{x})$ that provides the largest value for the measurement $\mathbf{x}_k$ (Eq. 2.6). A block diagram for the general case of C classes is presented in Figure 2-4.

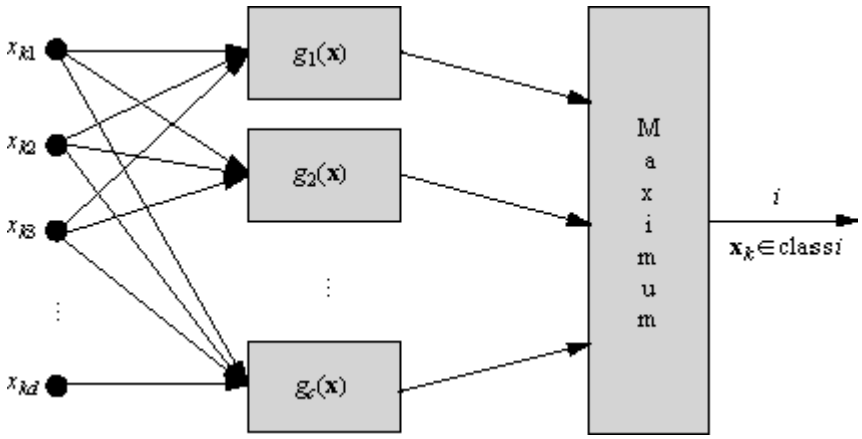


Figure2-4 General parametric classifier for c classes

The blocks labeled $g_i(\mathbf{x})$ compute the discriminants from the input data, and the block labeled maximum selects the largest value according to Eq. 2.1 or Eq. 2.6. Studying how the optimal classifier works, we arrive at the conclusion that the *classifier system creates decision regions bounded by the intersection of discriminant functions*. After this brief introduction, we realize that *machines that implement discriminant functions can be used as pattern classifiers.*
parametric and nonparametric classifiers

2.2.3 A Two-Dimensional Pattern-Recognition Example

The temperature example is too simple (the input is one-dimensional) to illustrate the full method of deriving decision boundaries based on statistical models of the data, the variety of separation surfaces, and the details and difficulty of the design. We will treat here the two-dimensional case, because we can still use pictures to help guide our reasoning. The method, of course, can be generalized to any number of dimensions.

Let us consider the following problem: Suppose that we wishe to classify males and females in a population by using measurements of their height and weight. Since we selected two variables, this is a two-dimensional problem. We are going to assume for simplicity that the distributions of height and weight are multivariate Gaussians and that the probability of occurrence of each sex is ½. Figure 2-5 shows the scatter plot of measurements for the two classes.

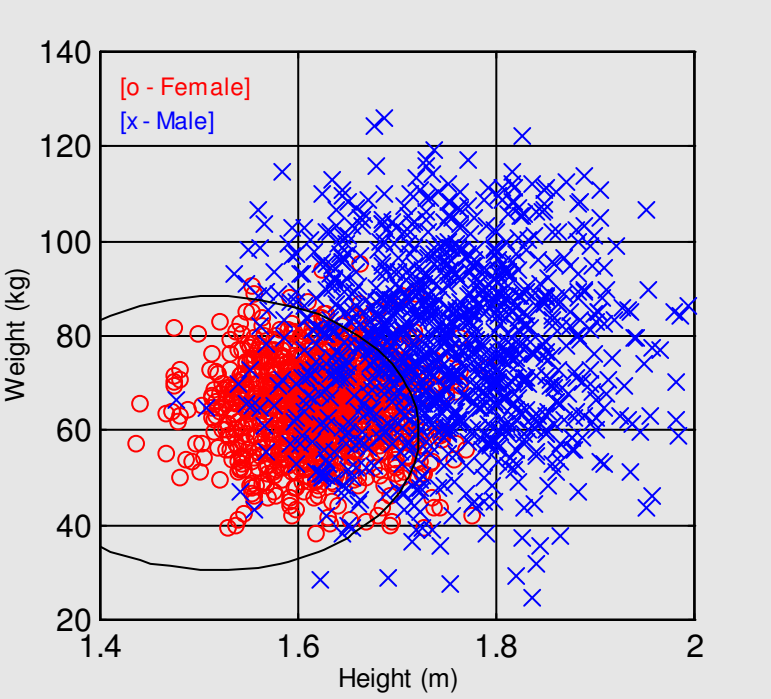


Figure 2-5 Scatter plot of the weight and height data with the optimal decision surface

The goal is to determine the placement of the decision surface for optimal classification. According to our previous discussion of the one-dimensional, this is achieved by estimating the means and standard deviations of the likelihoods from measurements performed in the population. Then the decision boundary is found by solving for $g_i(\mathbf{x}) = g_j(\mathbf{x})$, where i and j are the classes male and female. The difference from our previous example is that here we have a two-dimensional input space.

One can show [Duda and Hart] that for our two-dimensional, two-class case (normally distributed) with equal *a priori* probabilities, the classification will be a function of a normalized distance between the class centers, called the Mahalanobis distance

$$d^2 = (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) \qquad \textbf{(2.7)}$$

where μ is the class center vector and $\mathbf{\Sigma}$ is the covariance matrix of the input data in two-dimensional space. Notice that instead of the class variances, in the multidimensional case we have to compute the covariance matrix that is built from the class variances along each input space dimension. For this case the discriminant function is given by (derivation of quadratic discriminant)

$$g_i(\mathbf{x}) = -1/2(\mathbf{x} - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i) - D/2 \log(2\pi) - 1/2 \log|\Sigma_i| + \log P(c_i) \qquad \textbf{(2.8)}$$

Our example is a two-dimensional problem ($D = 2$) with two classes (i = 1, 2). Since the classes are equiprobable, the last term in Eq. 2.8 will be the same for each discriminant and can be dropped (likewise for the term $D / 2 \log(2\pi)$). When the discriminants for each class are equated

to find the decision surface, we see that in fact its placement is going to be a function of the Mahalanobis distance and class covariances. *As in the one-dimensional case, for classification, what matters is the distance among the cluster means normalized by the respective covariance. This is the metric for classification, and it is beautifully encapsulated in the Mahalanobis distance*.

Table 2.2 shows the estimates for the class covariances and means considering 100 and 1,000 samples per class. We will solve the problem with 1,000 samples first by computing the discriminants for each class.

Table 2-2 Data Statistics

	1,000 Measurements	100 Measurements
Women	Weight Mean = 64.86 Height Mean = 1.62 $Cov = \begin{bmatrix} 90.4401 & 0 \\ 0 & 0.0036 \end{bmatrix}$	Weight Mean = 63.7385 Height Mean = 1.6084 $Cov = \begin{bmatrix} 77.1877 & 0.0139 \\ 0.0139 & 0.0047 \end{bmatrix}$
Men	Weight Mean = 78.02 Height Mean = 1.75 $Cov = \begin{bmatrix} 310.1121 & 0 \\ 0 & 0.0081 \end{bmatrix}$	Weight Mean = 82.5278 Height Mean = 1.7647 $Cov = \begin{bmatrix} 366.3206 & 0.4877 \\ 0.4877 & 0.0084 \end{bmatrix}$

In this case the decision surface is given by the Bayes classifier

$$77.16x_2^2 - 233.95x_2 + 0.0039x_1^2 - 0.4656x_1 + 129.40 = 0$$

which is an equation for a quadratic curve in the input space (Figure 2-5).

2.2.4 Decision Surfaces of Optimal Classifiers

It can be shown (Fukunaga) that *the optimal classifier for Gaussian-distributed classes is always a quadratic*. There are three cases of interest:

- Covariance matrices are diagonal and equal.
- Covariance matrices for each class are equal.
- The general case.

For the two first cases, both the optimal discriminants and the decision surface are linear (Figure 2-6). These plots tell us how the density of samples decreases away from the center of the cluster (the mean of the Gaussian). The optimal discriminant function depends on each cluster shape, and it is in principle a quadratic. When the cluster shapes *are circularly symmetric with the same variance*, there is no difference in the radial direction for optimal classification, so the discriminant defaults to a linear function (a hyperplane). The decision surface is also linear and

is perpendicular to the line that joins the two cluster centers. For the same *a priori* probability the decision surface is the perpendicular bisector of the line joining the class means.

Even when the shapes of the clusters are skewed equally (the contour plots of each cluster are ellipses with equal axes), there is no information to be explored in the radial direction, so the optimal discriminants are still linear. But now the decision surface is a line that is slanted with respect to the line joining the two cluster means.

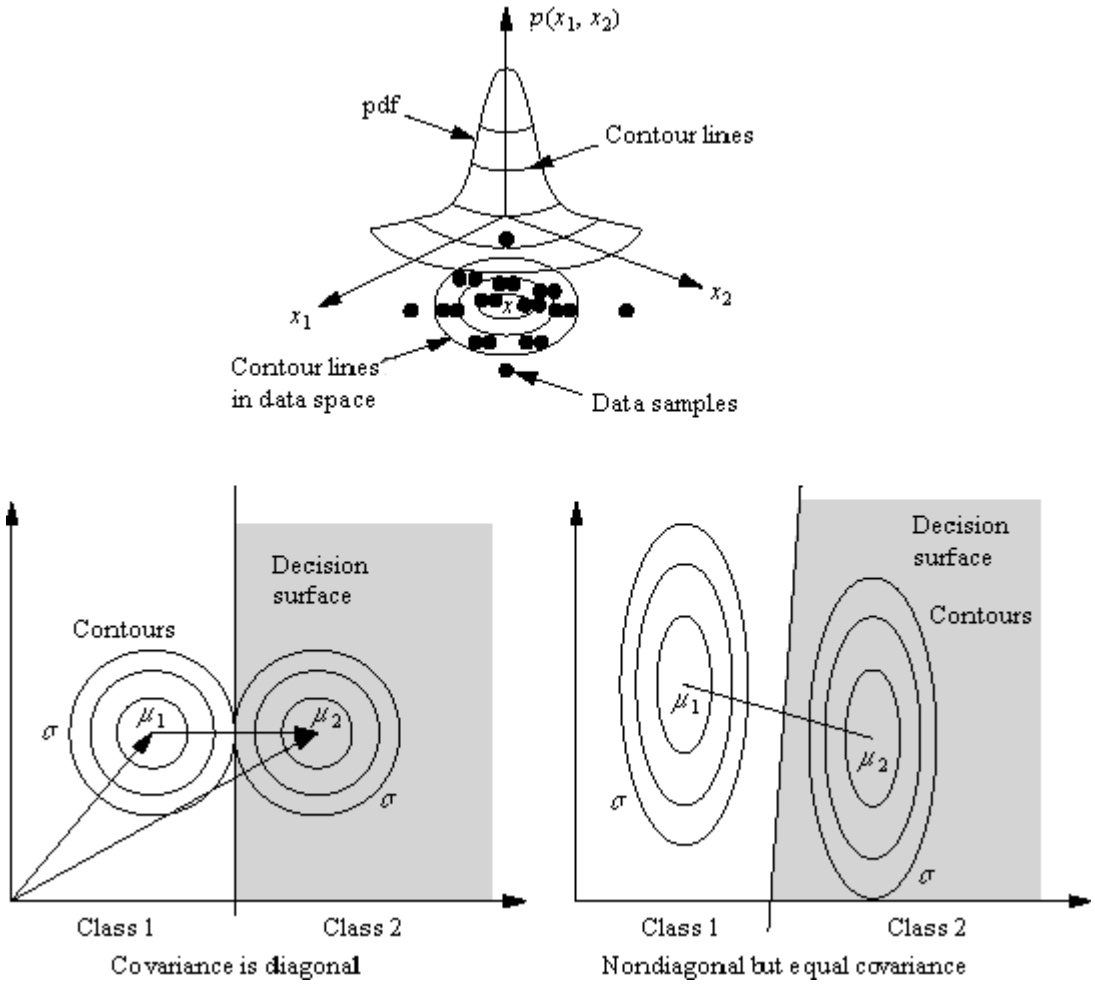


Figure 2-6 Contour plots for the data clusters that lead to linear discriminant functions

Figure 2-7 shows the contours of each data cluster and the discriminant for the case of arbitrary covariance matrices. Notice the large repertoire of decision surfaces for the two-class case when we assume Gaussian distributions. The important point is that the shape of the discriminants is highly dependent on the covariance matrix of each class. Knowing the shape of one data cluster is not enough to predict the shape of the optimal discriminant. We need knowledge about BOTH data clusters to find the optimal discriminant. shapes of 2D discriminants

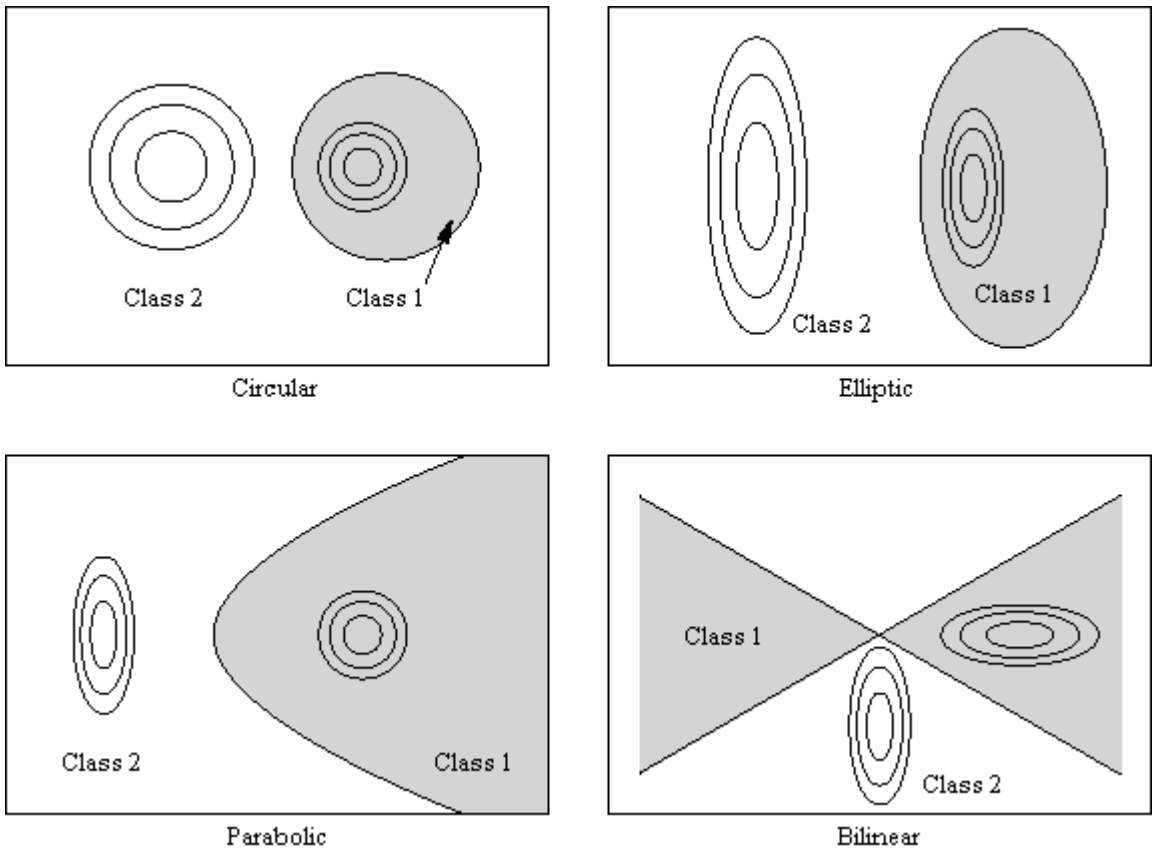


Figure 2-7 The general case of arbitrary covariance matrices

Eq. 2.8 shows that the optimal discriminant for Gaussian-distributed classes is a quadratic function. This points out that the *relationship between cluster distributions and discriminants for optimal classification is not unique*, i.e. there are many possible functional forms for the optimal discriminant (Gaussians, quadratics, and hyperplanes for this case). Observe that the parameters of the discriminant functions are a direct function of the parameters of the class likelihoods, so once the parameters of Eq. 2.8 are estimated, we can immediately determine the optimal classifier. For the female-male example, Figure 2-8 shows the modeled distribution of the two classes.

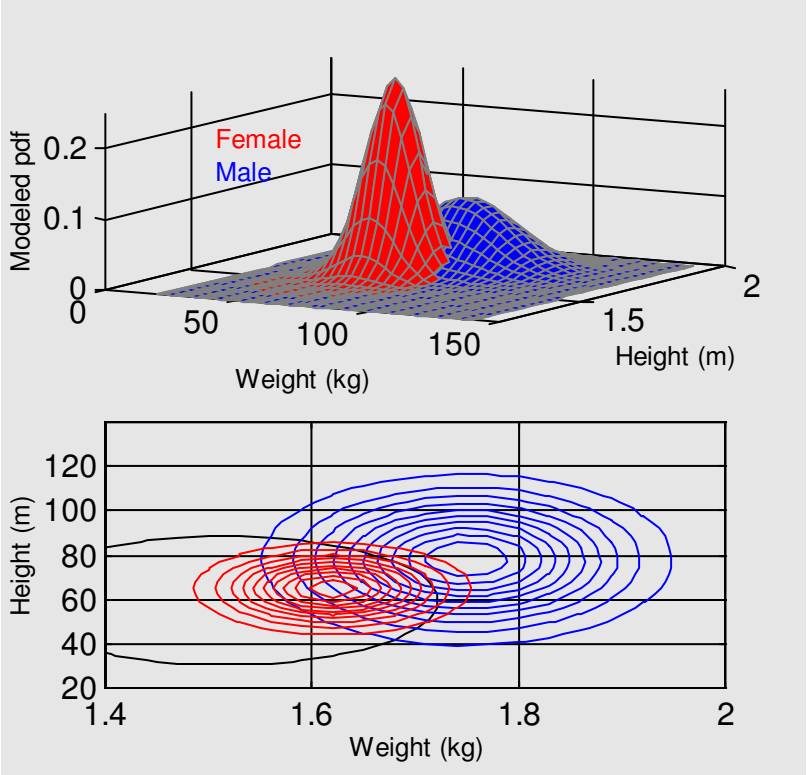


Figure 2-8 Modeled distribution of Figure 2-5 data

Discriminant Sensitivity to the Size of the Data

We have developed a strategy that is able to construct optimal decision surfaces from the data under the assumptions that the pdf of each class is Gaussian. This is a powerful procedure, but it is based on assumptions about the pdf of the input and also requires enough data to estimate the parameters of the discriminant functions with little error. The *ultimate quality of the results will depend upon how valid these assumptions are for our problem*. We illustrate here the effect of the training data size on the estimation of the discriminant function parameters.

Let us assume that we had only 100 samples for the height/weight example (50 males and 50 females). We extracted these samples randomly from the larger data file and computed the means and covariances for each class, as shown in Table 2.2. Just by inspection you can see that the parameters changed quite a bit. For instance, the covariances are no longer diagonal matrices, and the elements also have different values. Note also that the mean estimates are more accurate than the covariance estimates. When we build the optimal discriminant function from these parameters (Eq. 2.8) the shape and position in the input space is going to be different from the "ideal" case of 1,000 samples.

Figure 2-9 shows the differences in shape and placement of the optimal decision surface for the two data sets. In this case the differences are significant, producing different classification accuracy, but the decision surfaces still have the same overall shape. But remember that this is a simple problem in two-dimensions (7 parameters to be estimated with 50 samples per class). In higher-dimensional spaces the number of parameters to be estimated may be of the same order

higher-dimensional spaces the number of parameters to be estimated may be of the same order of magnitude of the data samples, and in this case catastrophic differences may occur. So what can we do to design classifiers that are less sensitive to the *a priori* assumptions and the estimation of parameters?

The answer is not clear-cut, but it is related to the simplicity of the functional form of the discriminant. We should use discriminant functions that have fewer parameters and that can be robustly estimated from the amount of data that we have for training. These simpler discriminants may be suboptimal for the problem, but experience shows that many times they perform better than the optimal discriminant. This seems a paradox, but the reason can be found in the brittleness of the parameter estimation. Even if we use the quadratic discriminant (which is optimal for Gaussian-distributed classes), the classifier may perform poorly if its discriminant functions are not shaped and positioned accurately in the input space.

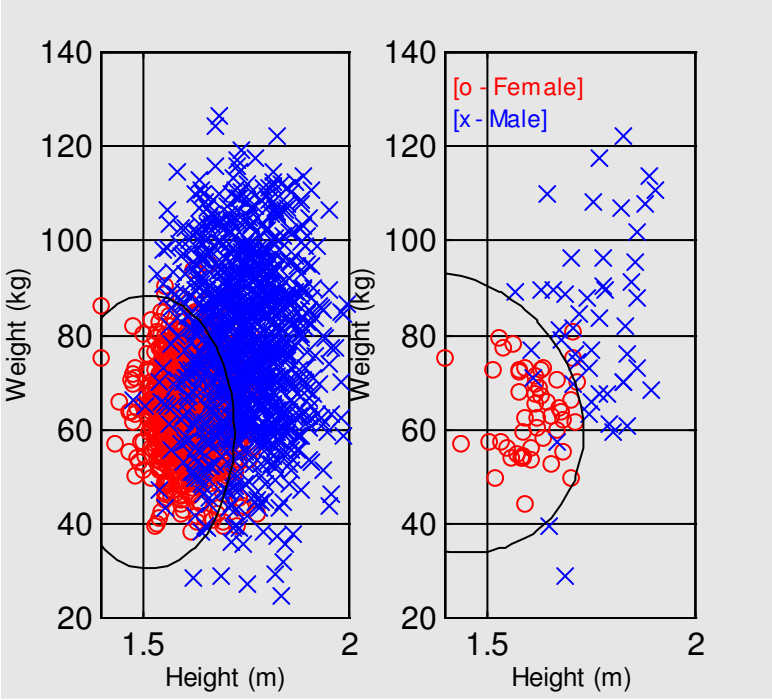


Figure 2-9 Comparisons of decision surfaces

Go to next section

2.3 Linear and Nonlinear Classifier Machines

2.3.1 The Linear Machines

Thus far we have encountered three types of discriminant functions: the linear, the quadratic and the Gaussian. Let us compare them in terms of the number of free parameters for D-dimensional data. The *linear discriminant function* given by

$$g_x = w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b = \sum_{i=1}^{D} w_i x_i + b \qquad \textbf{(2.9)}$$

has a number of parameters that increases linearly with the dimension D of the space. The discriminant function with the next higher degree polynomial, the quadratic, has a square dependence on the dimensionality of the space (i.e., it has approximately D^2 parameters) as we can see in Eq. 2.8. The Gaussian gave rise to a quadratic discriminant by taking the logarithm. Although quadratics are the optimal discriminants for Gaussian-distributed clusters, it may not be feasible to properly estimate all these parameters in large-dimensional spaces unless we have a tremendous amount of data.

Notice that Eq. 2.9 is a *parametric equation for a hyperplane* in D-dimensions, which we already encountered in linear regression (although there it had the function of modeling the input-output relationship). The hyperplane can be rotated and translated (an affine transformation) by changing the values of the free parameters w_i and b, respectively. The system that implements the discriminant of Eq. 2.9 is depicted in Figure 2-10 and is called a linear machine. Notice that the pattern-recognizer block is simpler (compared with Figure 2-4) because the unspecified functions g(**x**) become simply a sum of products. We have seen that the linear machine is even optimal for Gaussian distributed classes with equal variances, which is a case of practical relevance (for example data transmission in stationary, noisy channels).

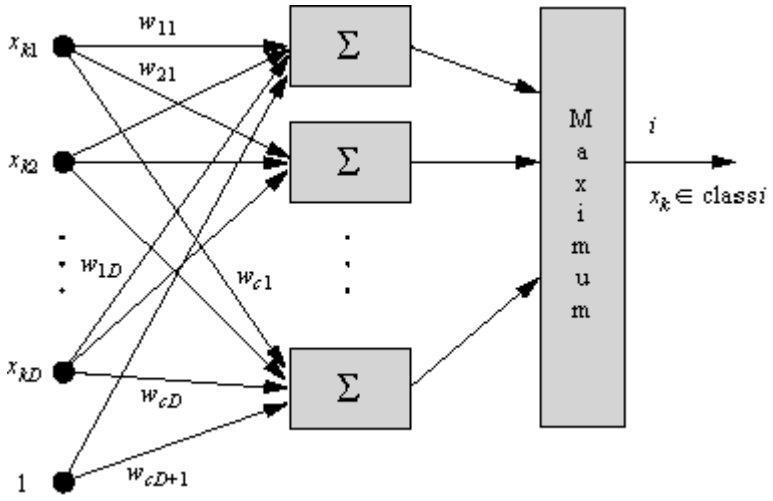


Figure 2-10 Linear classifier for c classes

It is ironic that in our classifier design methodology we are again considering linear functions for discrimination. It seems that we are back at the point where we started this chapter, the construction of a classifier based on the linear regressor followed by a threshold. But notice that now we have a much better idea of what we are seeking. The pattern-recognition theory tells us that we may use linear discriminants, but *we use one per class*, not a regression line linking all the input data with the class labels. We also now know that the linear discriminant may not be optimal but may be the best we can do because of the practical limitation of insufficient data to properly estimate the parameters of the optimal discriminant functions. The pattern-recognition theory thus gave us the insight to seek better solutions.

It is important to stress that the linear discriminant is less powerful than the quadratic discriminant. A linear discriminant primarily utilizes differences in means for classification. If two classes have the same mean, the linear classifier will always produce poor results. Examine Figure 2-11. A linear separation surface will always misclassify approximately half of the other class. However, the quadratic discriminant does a much better job because it can utilize the differences in the covariances.

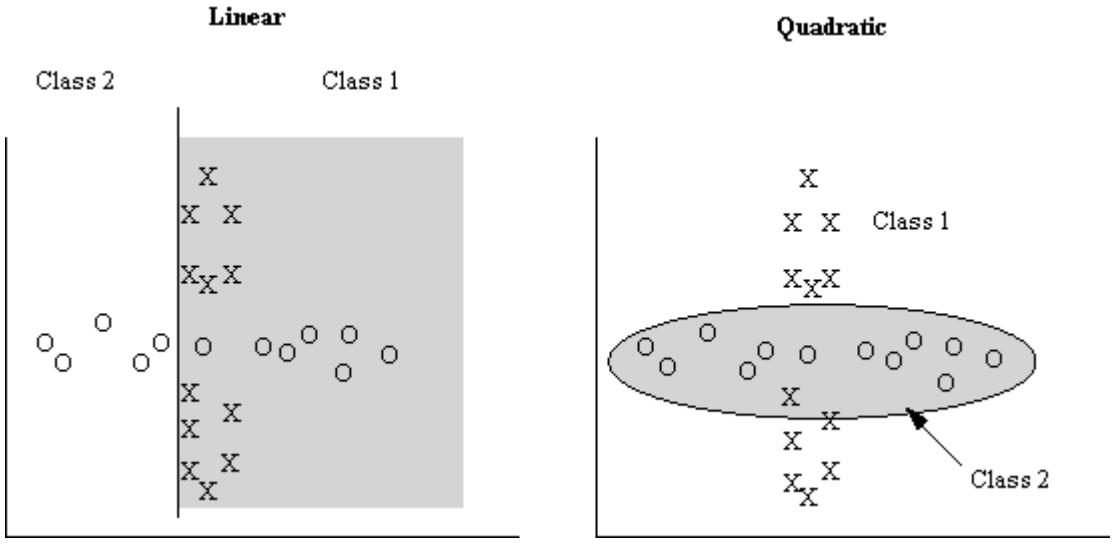


Figure 2-11 Comparison of the discriminant power of the linear and quadratic classifiers

2.3.2 Kernel-Based Machines

A more sophisticated learning machine architecture is obtained by implementing a nonlinear mapping from the input to another space, followed by a linear discriminant function (Figure 2-12). See Nilsson. The rationale of this architecture is motivated by Cover's theorem. Cover basically states that any pattern-recognition problem is linearly separable in a sufficiently high dimensionality space, so the goal is to map the input space to another space, called the feature space Φ, using nonlinear transformations. Let us assume that the mapping from the input space $\mathbf{x} = [x_1,\ldots x_D]$ to the higher dimensional Φ space is a one-to-one mapping operated by a kernel function family,

$$K(\mathbf{x}) = \{k_1(\mathbf{x}), \ldots, k_M(\mathbf{x})\}$$

applied to the input. For instance, we can construct a quadratic mapping in this way by equating the first D components of K to x_i^2, the next D(D-1)/2 components to all pairs, $x_i x_j \quad i \neq j$, and the last D components to x_i The feature space Φ in this case is of size M=[D(D+3)]/2.

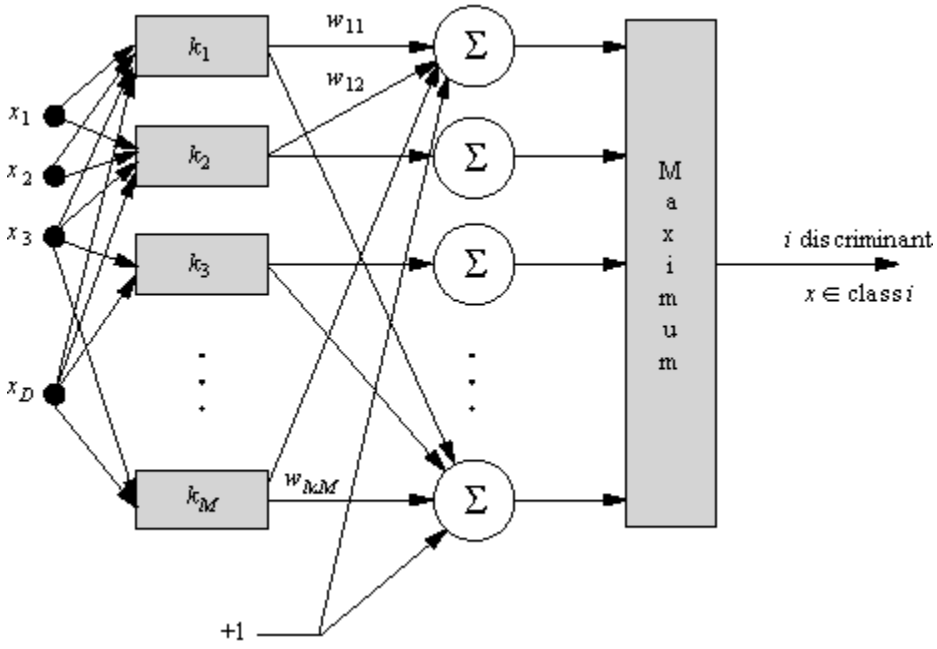


Figure 2-12 A kernel-based classifier

There is great flexibility in choosing the family of functions K(**x**). They need to be nonlinear, such as Gaussians, polynomials, trigonometric polynomials, and so on. Then in Φ space we can construct a linear discriminant function as

$$g(\mathbf{x}) = w_1 k_1(\mathbf{x}) + \ldots + w_M k_M(\mathbf{x}) + b$$

As before the problem is to select the set of weight vector **w** in Φ space that classifies the problem with minimum error. The general architecture for the kernel classifier is to build a kernel processor (which computes K(**x**)) followed by a linear machine. In the previous example, we actually constructed a quadratic discriminator. The major advantage of the kernel-based machine is that it decouples the capacity of the machine (the number of free parameters) from the size of the input space. size of feature space

Recently Vapnik has shown that if K(**x**) are symmetric functions that obey the Mercer condition (i.e. K(**x**) represents an inner product in the feature space), the solution for the discriminant function problem is greatly simplified. The Mercer condition basically states that the weights can be computed without ever solving the problem in the higher dimensional space Φ, which gives rise to a new classifier called the Support Vector Machine (SVM). We will study the SVM later.

2.3.3 Classifiers for the Two Class Case

The general classifier can be simplified for the two-class case, since only two discriminant functions are necessary. It is sufficient to subtract the two discriminant functions and assign the classes based on the sign of a single, new discriminant function. For instance, for the sick-healthy classification

$$g_{new}(x) = g_{healthy}(x) - g_{sick}(x) \qquad \textbf{(2.10)}$$

which leads to the block diagram in Figure 2-13.

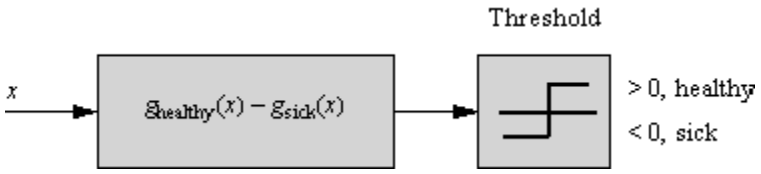


Figure 2-13 Classifier for a two-class problem

Note that $g_{new}(\mathbf{x})$ divides the space into two regions that are assigned to each class. For this reason, this surface obeys the definition of a decision surface, and its dimension is one less than the original data space dimension. For the 1-D case it is a threshold (a point) at 37°C. But will be a line (1-D surface) in 2-D space, and so on.

It is important at this point to go back to our NeuroSolutions Example 2.1 where we built a classifier from an Adaline followed by a threshold function, and compare that solution with Figure 2-13. We can conclude that the Adaline is effectively implementing the discriminant function $g_{new}(\mathbf{x})$. Due to the particular way that we defined the labels (1,-1), the regression line will be positive in the region of the temperatures for healthy individuals and negative toward the temperatures of sick individuals. The sign of the regression thus effectively implements $g_{new}(\mathbf{x})$. There are several problems with this solution:

- First, there is no rigorous way to choose the values of the desired response, and they tremendously affect the placement of the regression line (try 0.9, and -0.1 instead of ± and see how the threshold changes).

- Second, it is not easy to generalize the scheme for multiple classes (the sign information can be used only for the two-class case). As we saw, classification requires a discriminant function per class.

- Third, the way that the Adaline was adapted has little to do with minimizing the classification error. The error for training comes from the difference between the Adaline output (before the threshold) and the class labels (Figure 2-14*). We are using a nonlinear system, but the information to adapt it is still derived from the linear part.*

Only under very restricted conditions will this scheme yield the optimum classifier. In the following chapter we will learn to implement a classifier in which the classification error (the error after the threshold) is used to train the network.

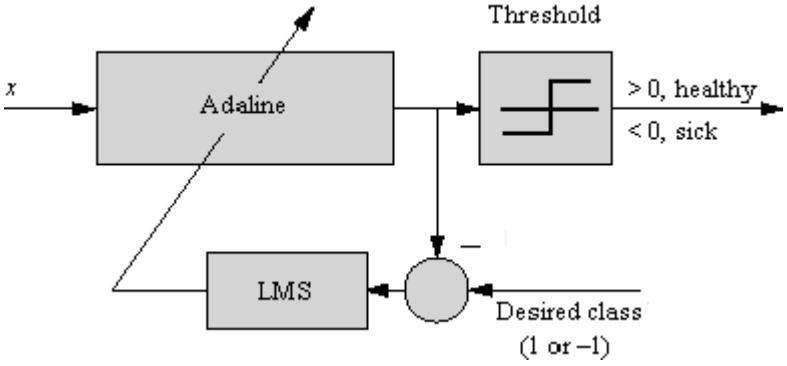


Figure 2-14 Schematic training of the Adaline with threshold

The Adaline followed by a threshold, as shown in Example 2.1, was applied in the 1960s by Widrow and Hoff for classification purposes.

Go to next section

2.4 Methods of Training Parametric Classifiers

The methods that we present in this book assume that there is little information available to help us make principled decisions regarding the parameter values of the discriminant functions. Therefore, *the parameters must be estimated from the available data*. One must first collect sufficient data that covers all the possible cases of interest, then use this data to select the parameters that produce the smallest possible error. This is called training the classifier, and we found a very similar methodology in Chapter 1.

The accuracy of a classifier is dictated by the location and shape of the decision boundary in pattern space. Since the decision boundary is obtained by the intersection of discriminant functions, there are two fundamental issues in designing accurate parametric classifiers (i.e., classifiers that accept a functional form for their discriminant functions):

- The *placement* of the discriminant function in pattern space
- The *functional form* of the discriminant function

There are two different ways to use the data for training parametric classifiers (Figure 2-15 – do not confuse parametric classifiers with parametric training).

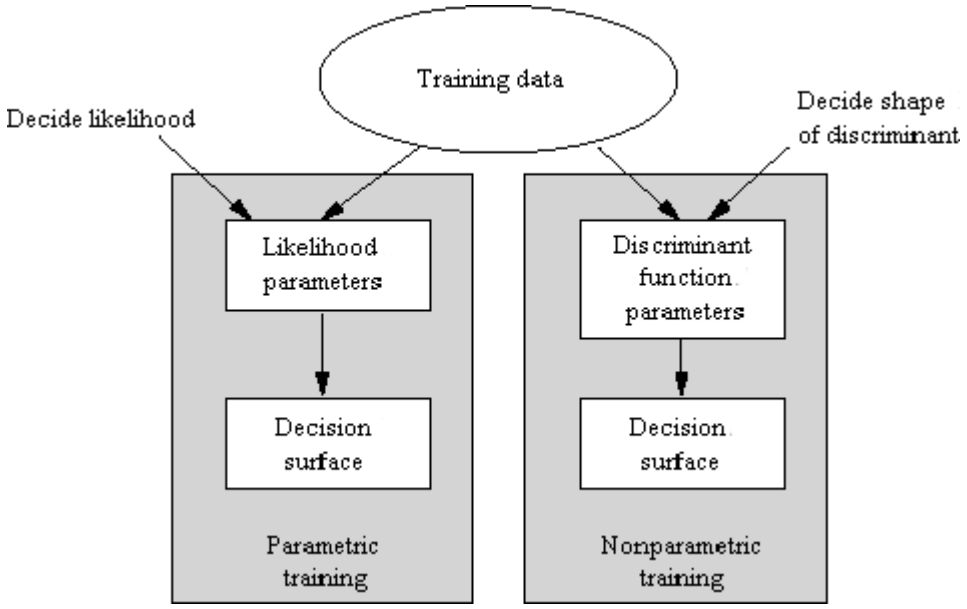


Figure 2-15 Parametric and nonparametric training of a classifier

In *parametric training* each pattern category is described by some known functional form for which its parameters are unknown. The decision surface can then be analytically defined as a function of these unknown parameters. The method of designing classifiers based on statistical models of the data belongs to parametric training. We describe the data clusters by class likelihoods, and the discriminants and decision surfaces can be determined when these

likelihoods, and the discriminants and decision surfaces can be determined when these parameters are estimated from the data. For instance, for Gaussian-distributed pattern categories, we need to estimate the mean vector, the covariance (normally using the sample mean and the sample covariance) and the class probabilities to apply Eq. 2.1. Unfortunately, due to the analytic way in which discriminants are related to the likelihoods, only a handful of distributions have been studied, and the Gaussian is almost always utilized.

In *nonparametric training* the free parameters of the classifier's discriminant functions are *directly estimated from the input data*. Assumptions about the data distribution are never needed in nonparametric training. Very frequently, nonparametric training utilizes iterative algorithms to find the best position of the discriminant functions. However, the designer still has to directly address the two fundamental issues of parametric classifier design, that is, the functional form of discriminant functions and their placement in pattern space.

2.4.1 Parametric versus Nonparametric Training

Let us raise an important issue. Is there any advantage in using nonparametric training? The optimal discriminant function depends on the distribution of the data in pattern space. When the boundary is defined by statistical data modeling (parametric training), optimal classification is achieved by the choice of good data models and appropriate estimation of their parameters. This looks like a perfectly fine methodology to design classifiers. Therefore, in principle, there seems to be no advantage in nonparametric training, which starts the classifier design process by selecting the discriminant functional form "out of the blue." In reality, there are some problems with parametric training for the following reasons:

- Poor choice of likelihood models. When we select a data model (e.g. the Gaussian distribution), we may be mistaken, so the classifier may not be the optimal classifier after all. Conversely, estimating the form of the pdf from finite training sets is an ill-posed problem, so this selection is always problematic.

- Too many parameters for the optimal discriminant. We saw earlier that the performance of the quadratic classifier depends on the quality of the parameter estimation. When we have few data points, we may not get a sufficiently good estimation of the parameters, and classification accuracy suffers. As a rule of thumb we should have 10 data samples for each free parameter in the classifier. If this is not the case we should avoid using the quadratic discriminator. One can say that *most real-world problems are data bound*; that is, for the number of free parameters in the optimal classifier, there is not enough data to properly estimate its discriminant function parameters.

Very often we are forced to trade optimality for robustness in the estimation of parameters. In spite of the fact that the quadratic classifier is optimal, sometimes the data can be classified with enough accuracy by simpler discriminants (such as the linear), as we showed in the shapes of 2-D discriminants. These discriminants have fewer parameters and are less sensitive to estimation errors (take a look at Table 2.2 and compare the estimations for the means and variances), so they should be used instead of the quadratic discriminants when the data is not enough to estimate the parameters accurately.

This raises the question of using the data to directly estimate the parameters of the discriminant

This raises the question of using the data to directly estimate the parameters of the discriminant functions, that is choosing a nonparametric training approach. What we gain is classifiers that are insensitive to the assumption on the pdf of the data clusters. We can also control in a more direct way the number of free parameters of the classifier.

2.4.2 Issues in Nonparametric Training

The difficulties that are brought about by nonparametric training are twofold:

- Deciding the shape of the discriminant function for each class
- Finding ways to adapt the parameters of the classifier

Shape of Discriminant Functions

The central problem in nonparametric training of parametric classifiers can be re-stated as the selection of an appropriate functional form for the discriminant function that meets these two criteria:

- It produces small classification error
- It has as few parameters as possible to enable robust estimation from the available data

The linear-machine decision boundaries are always convex because they are built from the superposition of linear discriminant functions. This is very restrictive, and solving realistic problems may thus require more versatile machines. The kernel-based classifiers are one option. They form convex boundaries in the high dimensional Φ space, but not necessarily in the input space. Another class of more versatile machines in called *semi-parametric* classifiers because they still work with parametric discriminant functions but are able to implement a larger class of discriminant shapes (eventually any shape, which makes them universal approximators). Semiparametric classifiers are very promising because they are an excellent compromise between versatility and the number of trainable parameters. Artificial neural networks are one of the most exciting types of semiparametric classifiers and will be the main subject of our study.

Adaptation of Classifier Parameters

We can use the ideas of iterated training algorithms to adapt the classifier parameters. In Chapter 1 we trained the Adaline parameters directly from the data, so the linear regressor was effectively nonparametrically trained. We can also *foresee the use of the gradient descent method explained in Chapter 1 to adjust the parameters of the discriminant functions* so that a measure of the misclassifications (output error) is minimized. Gradient descent will be extended in the next chapter for classifiers.

Go to next section

2.5 Conclusions

In this short chapter we covered the fundamental principles of pattern recognition. We started by briefly reviewing the problem of pattern recognition from a statistical perspective. We provided the concepts and definitions to understand the role of a pattern recognizer. We covered Bayes' classifier and showed that the optimal classifier is quadratic. Sometimes suboptimal classifiers perform better when the data is scarce and the input space is large, in particular, when the input is projected into a large feature space, as done in kernel classifiers.

The linear classifier was also reviewed, and a possible implementation was provided. These concepts are going to be very important when we discuss artificial neural networks in the next chapter.

Go to next section

2.6 Exercises

2.1 Modify the desired response to (1,0) in NeuroSolutions Example 2.1 (use the "normalization" function of the "stream" page of the "file" inspector). Notice how the threshold changes. Elaborate on the dependence of linear machines in the actual values of the desired response. Is this good or bad?

2.2 In the chapter 2 folder of the data directory (CDROM) we provide data that can be used for classification (prob2.2 train and test). The last column is the class label. This data exists in a 2-D space such that we can still visualize the discriminant functions. It is also a 2-class problem for simplicity. In this problem you will design a linear classifier, assuming first that each class can be modeled by a Gaussian distribution. You should first find the parameters for each Gaussian, and then design the classifier. Plot the separation surface over the data.

2.3 For the data in Problem 2, design the optimal Bayes' classifier and compare performance with the linear classifier by determining the percent of correct classifications. Plot also the separation surface and compare it with Problem 2. Do you think that for this data set the Bayes' classifier is optimal?

2.4 If you want to use a linear classifier but train it nonparametrically can you do it? What is missing? Why is it that LMS cannot be used?

2.5 Compare the performance of the Bayes' classifier developed in Problem 3 when you use only the first 10 percent of the data for training.

2.6 Estimate the probability of error of the Bayes' classifier developed in Problem 3.

2.7 State in your own words the reason why the metric of classification is not Euclidean.

2.8 In the ch2 folder of the data directory (CD-ROM) we provide data that can be used for classification (prob2.8 train and test). The last column is the class label. In this case the problem is a bit more difficult, with three classes in a higher (five) dimensional space. Develop a linear classifier and test the results. You may want to use Matlab or a similar program. Is it appropriate to develop a Bayes classifier to improve performance?

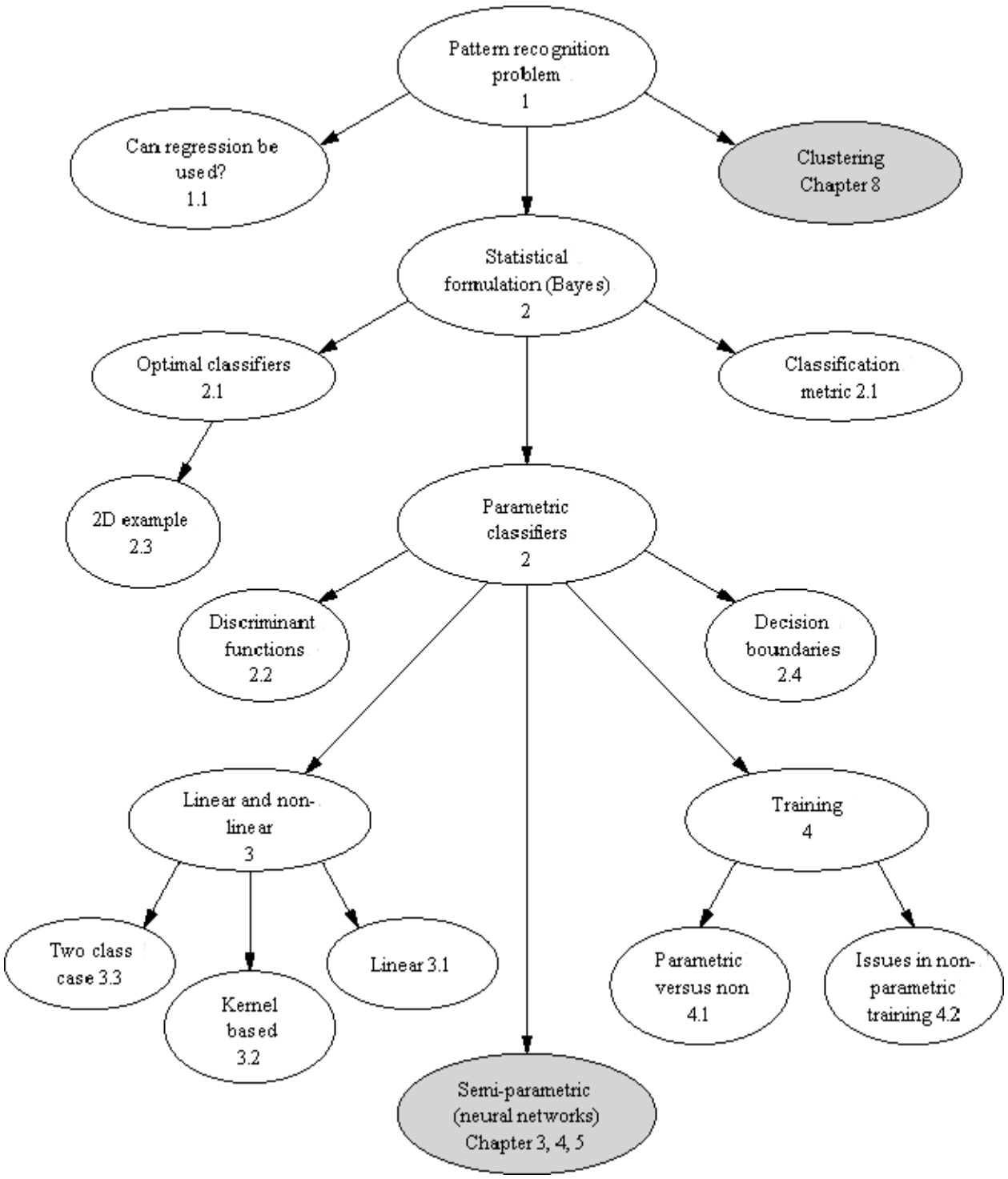
Go to next section

2.7 NeuroSolutions Examples

2.1 Comparing Regression and Classification

Go to next section

2.8 Concept Map For Chapter 2



Go to Next Chapter