Table of Contents

CHAPTER II - PATTERN RECOGNITION 2
1. THE PATTERN RECOGNITION PROBLEM 2
2. STATISTICAL FORMULATION OF CLASSIFIERS 6
3. CONCLUSIONS 30
UNDERSTANDING BAYES RULE 32
BAYESIAN THRESHOLD 33
MINIMUM ERROR RATE 34
PARAMETRIC AND NONPARAMETRIC CLASSIFIERS 35
MAHALANOBIS DISTANCE 36
COVARIANCE 37
DERIVATION OF QUADRATIC DISCRIMINANT 38
BAYES CLASSIFIER 39
SHAPES OF 2D DISCRIMINANTS 40
PARAMETRIC AND NONPARAMETRIC TRAINING 42
TRADE-OFFS OF PARAMETRIC TRAINING 42
R. A. FISHER 43
PATTERN RECOGNITION 43
PATTERN SPACE 44
CLASSES 44
CLASSIFIER 44
DECISION SURFACE 44
DISCRIMINANT FUNCTIONS 44
TRAINING THE CLASSIFIER 44
OPTIMAL CLASSIFIER 45
OPTIMAL DISCRIMINANT FUNCTION 45
LINEAR MACHINE 45
A POSTERIORI PROBABILITY 45
LIKELIHOOD 45
PROBABILITY DENSITY FUNCTION 45
EQ2 45
ADALINE 46
EQ.1 46
EQ.6 46
EQ.8 46
EQ.10 46
CONVEX 46
EQ.9 47
LMS 47
EQ.7 47
WIDROW 47
EQ.4 47
EQ.3 47
DUDA 48
FUKUNAGA 48
ILL-POSED 48
SIZE OF FEATURE SPACE 48
COVER'S THEOREM 49
VAPNIK 50
NILSSON 50
AFFINE 50

Chapter II - Pattern Recognition

Version 2.0

This Chapter is Part of:

Neural and Adaptive Systems: Fundamentals Through Simulation© *by*

Jose C. Principe
Neil R. Euliano
W. Curt Lefebvre

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

- Statistical pattern recognition
- Training of classifiers
  - 1.The pattern recognition problem
  - 2. Optimal Parametric classifiers
  - 3. Conclusions

Go to next section
Go to the Appendix

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 mathematical principles underlying the analysis and design of pattern recognition machines is still in its infancy. In the 30's R.A. Fisher laid out the mathematical principles of statistical Pattern Recognition which is one of the most principled ways to cope with the problem.

A real world example will elucidate the principles of statistical pattern recognition at work: Assume that the body temperature is utilized as an indicator of the health of a patient. Experience shows that in the healthy state the body regulates the body temperature near

37° degrees 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 one plots temperature of individuals on a line (Figure 1), we will see 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 *category regions* (classes) in pattern space. The goal of pattern recognition is to build machines, called classifiers , that will automatically assign measurements to classes.

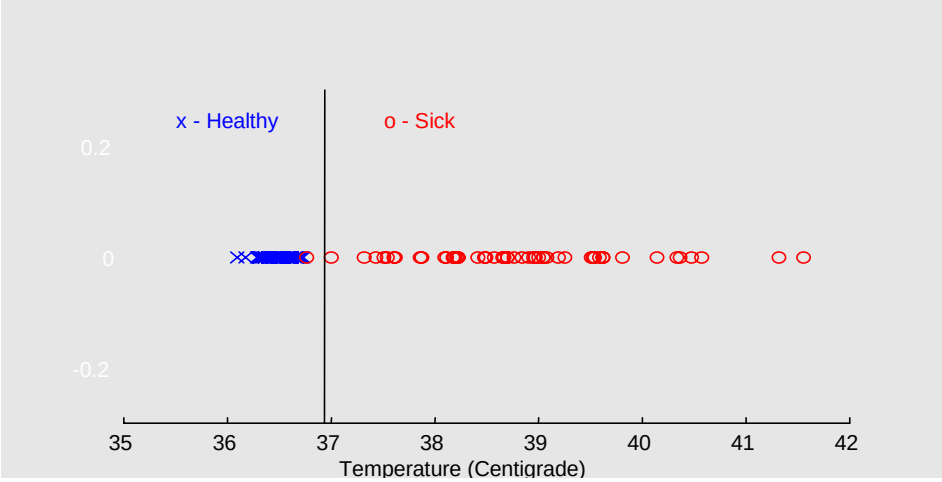


Figure 1**.** The sick/healthy problem in pattern space.

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 one gets a thermometer and starts measuring the temperature of healthy subjects, we will soon find out that individual temperatures vary from subject to subject, and change for the same subject depending upon the hour of the day, the subject state (i.e. rest or after exercise), etc. The same variability occurs in sick individuals (aggravated by the seriousness and type of illness), and there may be overlap between the temperature of sick and healthy

individuals. So, 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.

1.1. Can regression be used for pattern recognition?

We just presented in Chapter I a methodology that builds adaptive machines with the goal of fiting hyperplanes to data points. A legitimate question is to ask if regression can be used to solve the problem of separating data into classes. The answer is negative because the goals are very different.

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

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

In classification the issue is very different. We accept *a priori* that the input data was generated by different mechanisms and *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 very high level of abstraction, both the classification and the regression problems seek systems that transform inputs into desired responses. But the details of this mapping are rather different in the two cases.

We can nevertheless *use the machinery* utilized in linear regression, i.e. the adaptive system called 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 1

2.1 Comparing regression and classification

Suppose we are given the healthy and sick data, and we arbitrarily assign the value one as the desired system response to the healthy class, and the desired response of -1 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 number 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 one (healthy), and when it is negative the response is -1.

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. So with the arrangement of an adaline followed by a threshold we created our first classifier. But how can we improve upon its performance, estimate the optimal error rate, and extend it to multiple classes?

NeuroSolutions Example

The machinery used to adapt 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,-1, and not to minimize the classification error. This is a specially simple example with only two classes. For the multiple class case the results become even more fragile. So the conclusion is that we need a new methodology to study and design accurate classifiers. The machinery and algorithms we developed in chapter one, however, will be the basis for much of our future work. All of the concepts of learning curves, rattling, step sizes, etc. will all be

applicable.

Go to next section

2. Statistical Formulation of Classifiers

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 one 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 the Appendix). 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, i.e.

x belongs to c_i if $P(c_i|x) > P(c_j|x)$ for all $j \neq i$ **Equation 1**

The problem is that the *a posteriori* probability can not be measured directly. But using Bayes' rule

$$P(c_i|x) = \frac{p(x|c_i)P(c_i)}{P(x)}$$ **Equation 2**

one can compute the *a posteriori* probability from P(ci) the *prior probability* of the classes, multiplied by p(x|ci), the likelihood that the data *x* was produced by class c_i and normalized by the probability of P(x). Both P(ci) 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), P(ci) can be estimated from the demographics, season, etc. Figure 1 shows data collected from 100 cases. The likelihoods p(x|ci) can be estimated assuming a Gaussian distribution

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

Equation 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

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

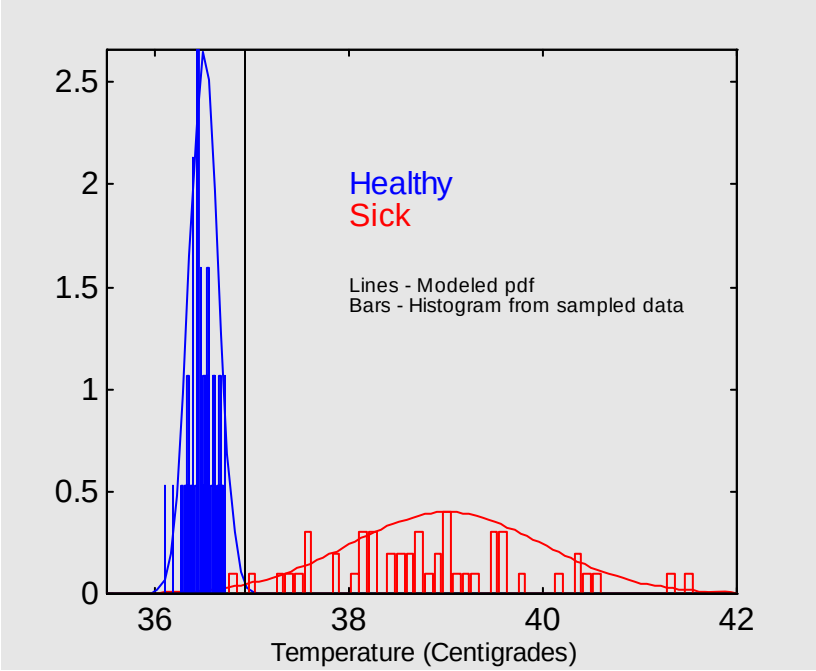
Equation 4

for this data set gives (N is the number of measurements)

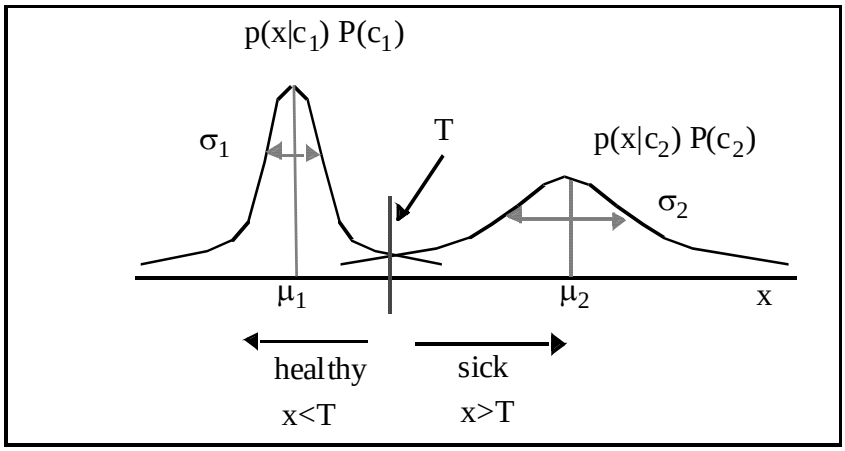
Temperature	1,000 Measurements	100 Measurements
Health	Mean = 36.50 Standard Deviation = 0.15	Mean = 36.49 Standard Deviation = 0.14
Sick	Mean = 39.00 Standard Deviation = 1	Mean = 38.83 Standard Deviation = 1.05

Table 1. Statistical measures for Figure 1 data.

The separation boundary, i.e. 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). Bayesian threshold .



a)



b)

Figure 2 . a) Sampled data distributions, b) Bayes threshold

It is rather easy to classify optimally healthy/sick cases using this methodology. Given a temperature *x* from an individual, one computes Eq 2 for both classes and assigns the label healthy or sick according to the one that produces the largest value (see Eq.1). Alternatively, one can compare the measurement to T and decide immediately 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, so should be assigned to the healthy class. Similarly, the measurements that fall towards the right, have a higher likelihood of being produced by sick cases.

Notice also that the class assignment is not error free. In fact, the tail of the healthy likelihood extends to the right of the intersect point, and the tail of the sick likelihood extends to the left of T. The error in the classification is exactly given by the sum of the areas under these tails. So the smaller the overlap the better the classification accuracy. The maximum *posteriori* probability assignment (Eq.1) minimizes this probability of error (minimum error rate), and is therefore optimum.

2.1.1 Metric for Classification

There are important conclusions to be taken from this example. For a problem with given class variances, if we increase the distance between the class means the overlap will decrease, i.e. the classes are more separable 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 can not just look at the class mean distance to estimate the classification error*, since the error depends upon 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 threhold what counts for placement of the decision surface is the class distance normalized by the class variances. We can encapsulate this idea by saying that *the metric for classification is not Euclidean*, but involves also the dispersion (variance) of each class. If we analyze closely the

exponent for the Gaussian distribution (Eq. 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 ditance of *x* from the mean normalized by the variance. This distance is called 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, optimum 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. 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 (vectors will be denoted in bold font) with D components

$$\mathbf{x}_k = [x_{k1}, x_{k2}, ..., x_{kD}] \qquad \textbf{Equation 5}$$

and can be imagined as a point in the D-dimensional Pattern Space . Following Eq.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

$$\mathbf{x}_k \text{ belongs to } c_i \text{ if } \quad g_i(\mathbf{x}_k) > g_j(\mathbf{x}_k) \quad \text{for all } j \neq i \qquad \textbf{Equation 6}$$

Each scaled likelihood can be thought of as a discriminant function g(**x**), i.e. 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 will intersect in the input space defining a decision surface , where the scores are equal (Figure 3). *So 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.

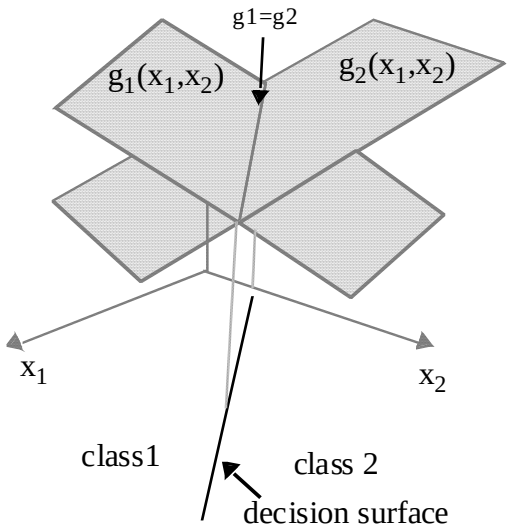


Figure 3. Discriminant functions and the decision surface.

In this view, the optimum classifier just compares discriminant functions (one per class) and chooses the class according to the discriminant $g_i(\mathbf{x})$ which provides the largest value for the measurement $\mathbf{x}_k$ (Eq.6). A block diagram for the general case of C classes is presented in Figure 4.

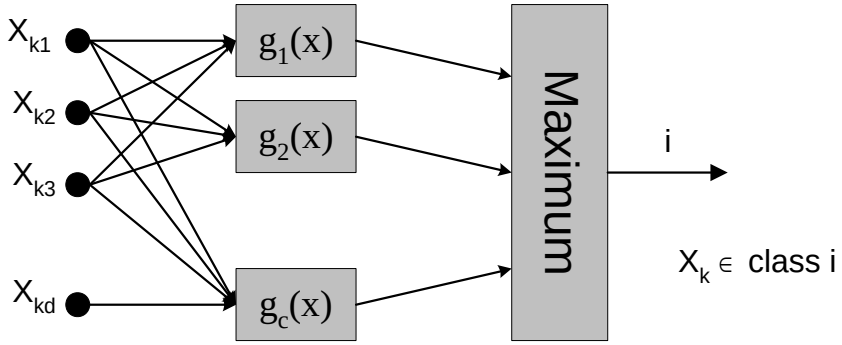


Figure 4 General parametric classifier for c classes

The blocks labelled gi(**x**) compute the discriminants from the input data, and the block labelled maximum selects the largest value according to Eq.1 or Eq.6 . So, studying how the optimal classifier works, one arrives 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

2.3. A two dimensional pattern recognition example

The temperature example is too simple (the input is 1-D) to illustrate the full method of deriving decision boundaries based on statistical models of the data, the variety of separation surfaces, and the details/difficulty of the design. We will treat here the two dimensional case, because we can still use “pictures” to help guide our reasoning, and the method can be generalized to any number of dimensions.

Let us consider the following problem: Suppose that one wishes to classify males and females in a population by using the measurements of 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 gender is ½. Figure 5 shows the scatter plot of measurements for the two classes.

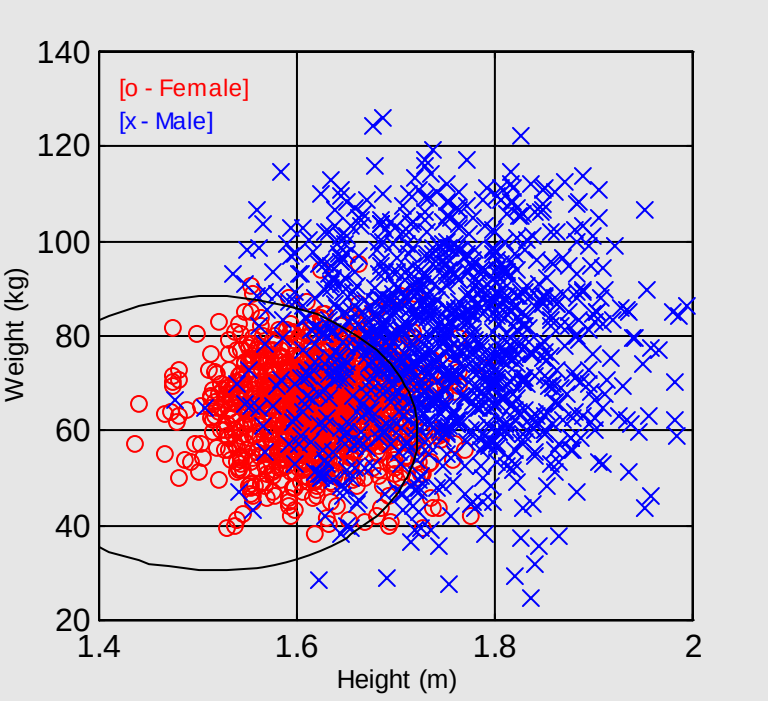


Figure 5. Scatter plot of the weight and height data with the optimal decision surface.

The goal is to determine the placement of the decision surface for optimal classification. According to our previous discussion of the one dimensional case (see the Bayesian threshold), this is achieved by estimating the means an 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,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 2-D, two class case (normal distributed) with equal *a priori* probabilities, the classification will be a function of a normalized distance between the class centers, called mahalanobis distance

$$r^2 = (\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)$$ **Equation 7**

where μ is the class center vector and $\mathbf{\Sigma}$ is the covariance matrix of the input data in 2-D space. Notice that instead of the class variances 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)

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

Equation 8

Here the classes are equiprobable, so the last term will be the same for each discriminant and can be dropped. Since this is a 2-D problem, d=2, and i=1, 2 because we have two classes. When the discriminants are equated together to find the decision surface, we see that in fact its placement is going to be a function of the Mahalonobis distance. *So 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 shows the estimates for the class covariances and means considering 1,000 and 100 samples per class. We will solve the problem with 1,000 samples first, by computing the discriminants for each class.

	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 $Cov = \begin{bmatrix} 366.3206 & 0.4877 \\ 0.4877 & 0.0084 \end{bmatrix}$

Table 2. Data Measures

In this case the decision surface is given by 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 5).

2.4. Decision Surfaces of Optimal Classifiers

One can show Fukunaga that the optimal classifier for Gaussian distributed classes is quadratic. There are three cases of interest:

- covariance matrices are diagonal and equal
- covariance matrices for each class are equal,
- and the general case.

For the two first cases both the discriminants and the decision surface default to linear (Figure 6). In the figure we show not only the pdf but also its contour plots. 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 to be explored in the radial direction for optimal classification, so the discriminant defaults to a linear function (an 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.

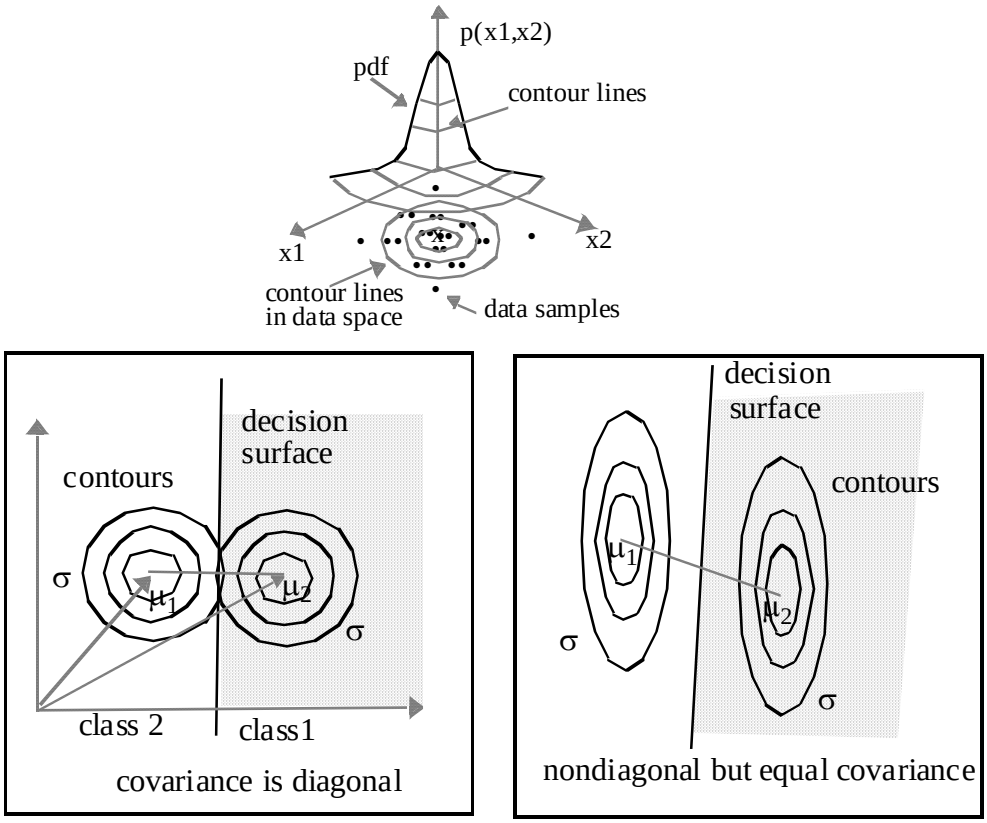


Figure 6. Contour plots for the data clusters that lead to linear discriminant functions.

Figure 7 shows the contours of each data cluster and the discriminant for the arbitrary covariance matrix case. 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 upon the covariance matrix of each class. Knowing what is the shape of one data cluster is not enough to predict the shape of the optimal discriminant. One needs knowledge of BOTH data clusters to find the optimal discriminant. shapes of 2D discriminants

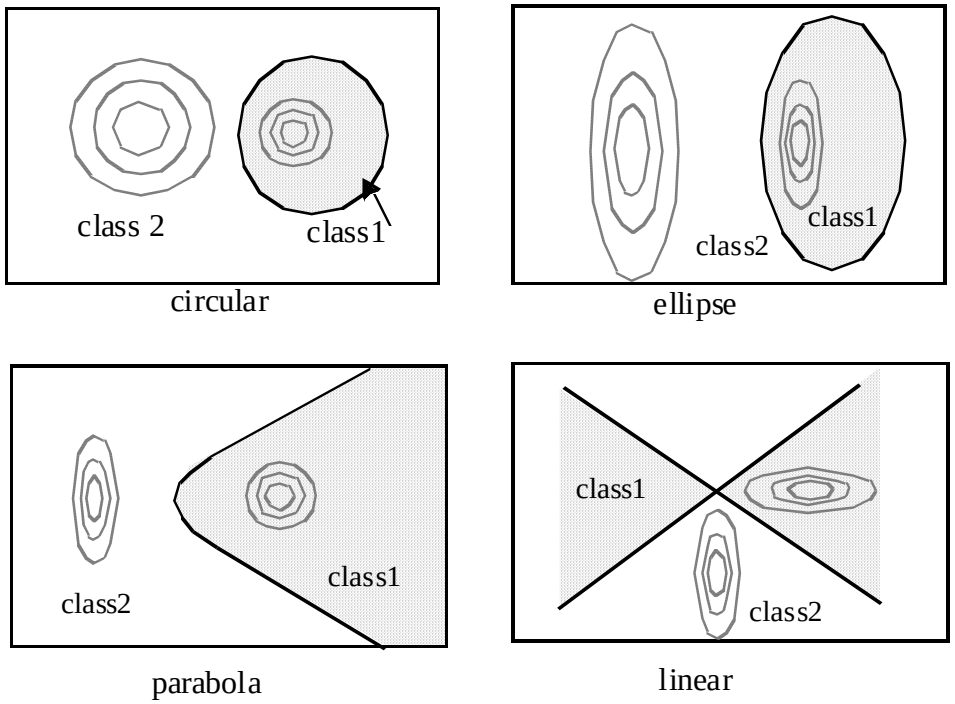


Figure 7. The general case of arbitrary covariance matrices.

Eq.8 shows that the optimal discriminant for Gaussian distributed classes is a quadratic function. This points out that the *relation between cluster distributions and discriminants for optimal classification is not unique*, i.e. there are many possible functional forms for the optimal discriminant (Gaussians and quadratics 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.8 are estimated, we can immediately determine the optimal classifier. For the present example, Figure 8 shows how the two classes modeled distribution looks like.

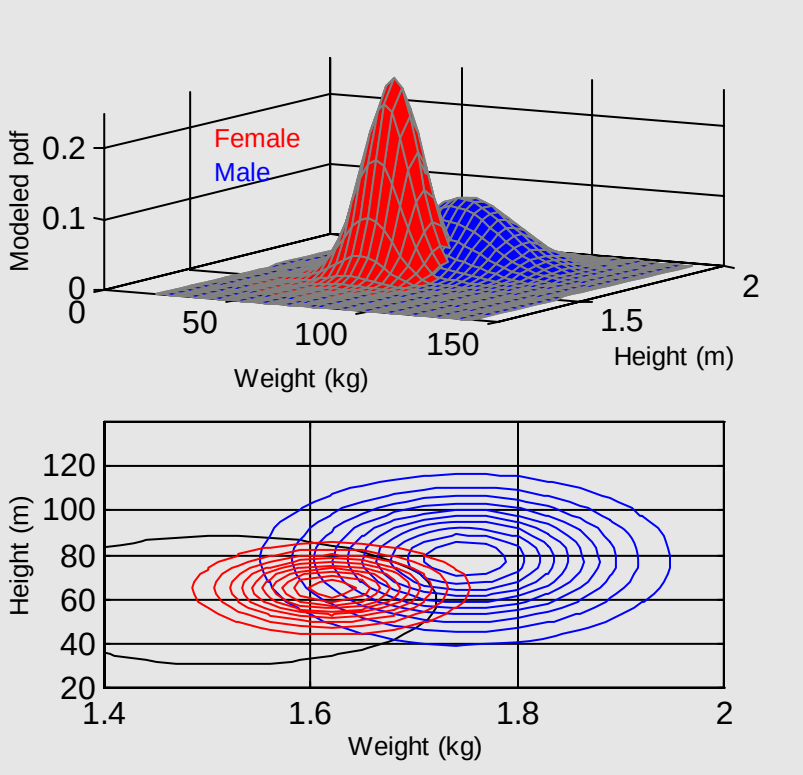


Figure 8. Modeled distribution of figure 5 data.

2.4.1 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 are these assumptions for our problem*. We will illustrate here the effect of the training data size on the estimation of the discriminant function parameters.

To demonstrate this point let us assume that we only had 100 samples for the height/weight example (50 males and 50 females). We extracted randomly these samples from the larger data file, and computed the means and covariances for each class as shown in Table II. 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 quality of the mean estimates is

higher than the covariance. When we build the optimal discriminant function from these parameters (Eq.8) the shape and position in the input space is going to be different when compared to the “ideal” case of 1,000 samples.

Figure 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 2-D (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 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. One should use discriminant functions that have fewer parameters, and that can be robustely estimated from the amount of data that we have for training. These simpler discriminants may be sub-optimal for the problem, but experience shows that many times they perform better than the optimal discriminant. This seems a paradox, but it is not. 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 give many errors if its discriminant functions are not shaped and positioned accurately in the input space.

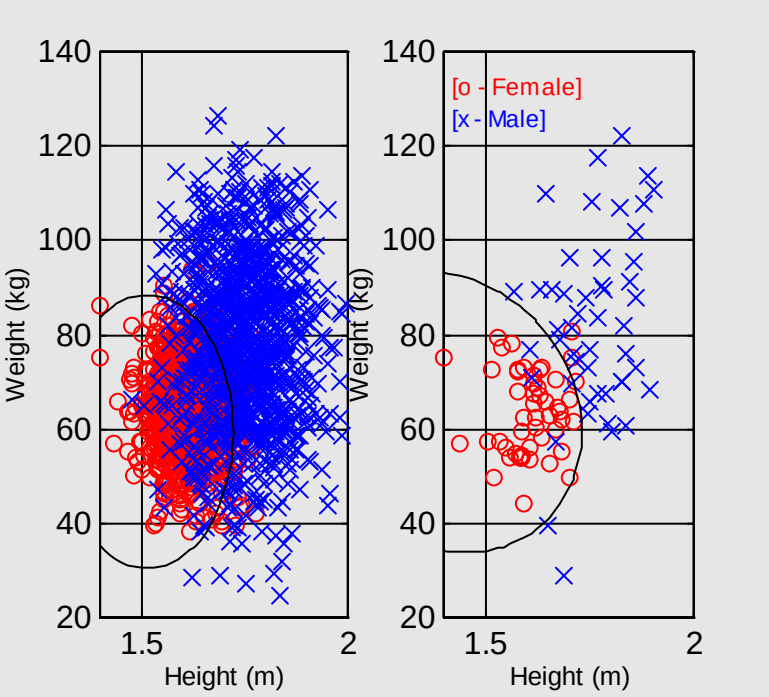


Figure 9- Comparisons of Decision Surfaces

2.5. The Linear Machines

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

$$g_x = w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b = \sum_{i=1} w_i x_i + b$$

Equation 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 D^2 parameters) as we can see in Eq.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 be unfeasible to properly estimate all of these parameters in large dimensional spaces unless we have a tremendous amount of data.

Notice that Eq.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 *wi* and *b* respectively. The system that implements the discriminant of Eq.9 is depicted in Figure 10 and is called a linear machine . Notice that the pattern recognizer block is simpler (versus Figure 4) with the assumption that the unspecified functions g(**x**) are simply 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 (data transmission in stationary noisy channels).

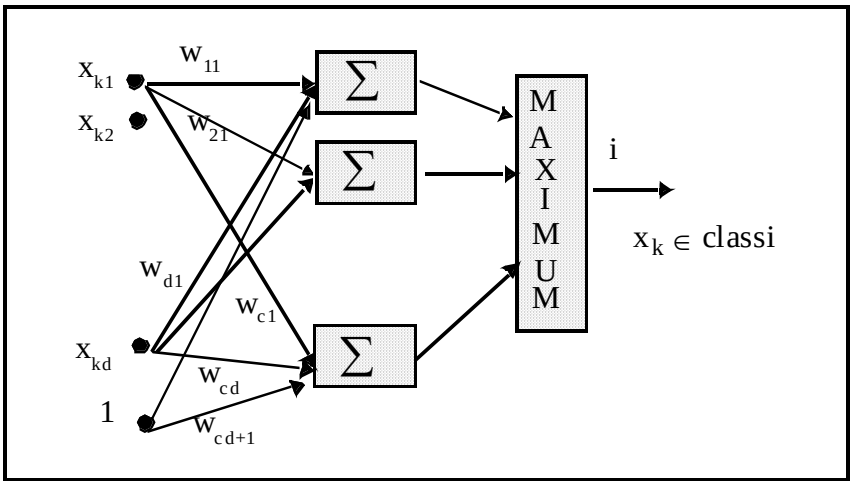


Figure 10**.** Linear classifier for c classes

It is ironic that in our classifier design methodology we are considering again linear discriminant functions. It seems that we are back at the point where we started this chapter, the construction of a classifier based on the linear regression 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 limitations of insufficient data to properly estimate the

parameters of optimal discriminant functions. So the pattern recognition theory 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 utilizes primarily differences in means for classification. If two classes have the same mean the linear classifier will always produce bad results. Examine Figure 11. A linear separation surface will always misclassify approximately ½ of the other class. However, the quadratic discriminant does a much better job because it can utilize the differences in covariance.

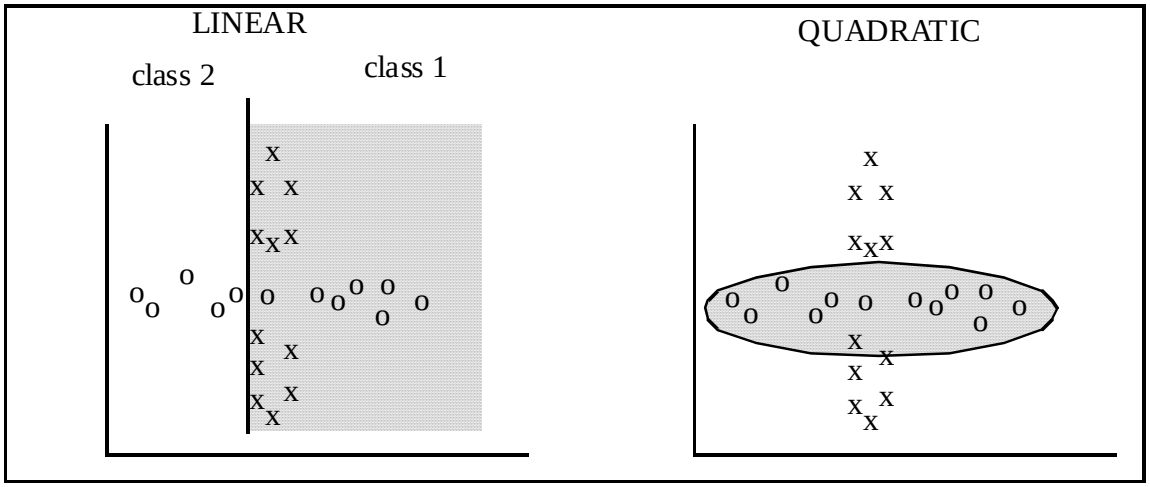


Figure 11. Comparison of the discriminant power of the linear and quadratic classifiers.

2.6 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 12). See Nilsson . The rational 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 by 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 xi . The feature space Φ in this case is of size M=[D(D+3)]/2.

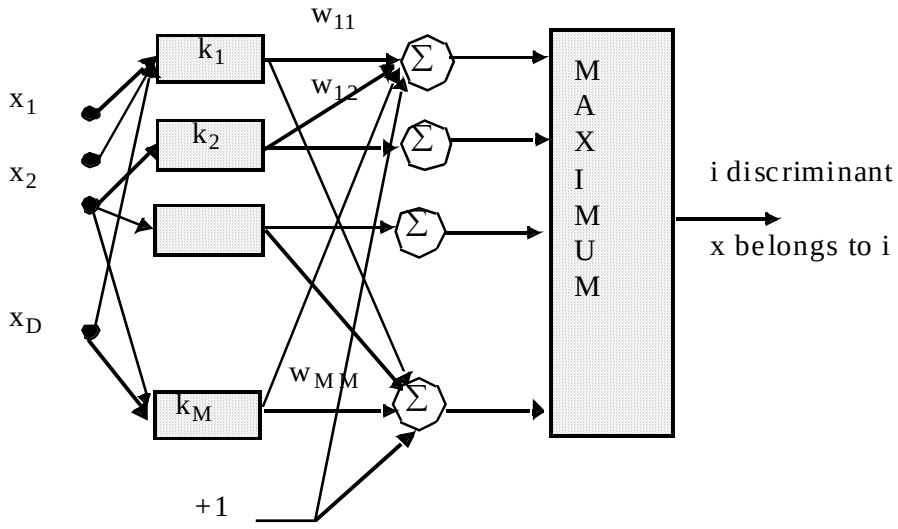


Figure 12 . A kernel Based classifier

There is a large flexibility in choosing the family of functions K(**x**). They need to be nonlinear such as Gaussians, polynomials, trigonometric polynomials, etc. 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. So the general architecture for the kernel classifier is to build a kernel processor (which computes K(**x**)) followed by a linear machine. In the example given above, 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. that 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. We will study it later.

2.7. Classifiers for the two class case

The general classifier can be simplified for the two-class case since only two discriminants 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

$$g_{new}(x) = g_{healthy}(x) - g_{sick}(x) \qquad \textbf{Equation 10}$$

which leads to the following block diagram for implementation

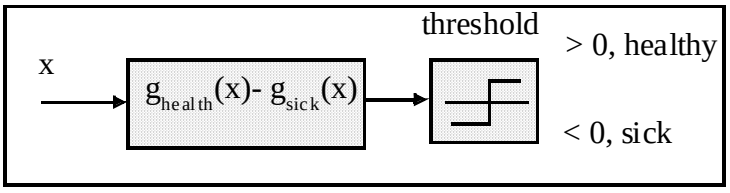


Figure 13**.** Classifier for a two-class problem

Note that g_{new}(**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, etc.

It is important at this point to go back to our NeuroSolutions example 1 where we built a

classifier from an adaline followed by a threshold function and compare that solution with Figure 13. One 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 towards the temperatures of sick individuals. So the sign of the regression effectively implements $g_{new}(\mathbf{x})$. There are several problems with this solution:

- First, there is no principled way to choose the values of the desired response, and they affect tremendously the placement of the regression line (try 0.9 and -0.1 and see how the threshold changes).
- Second, it is not easy to generalize the scheme for multiple classes (the sign information can only be used for the two class case). As we saw, classification requires a discriminant function per class.
- Thirdly, 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 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 where the classification error (the error after the threshold) is used to train the network.

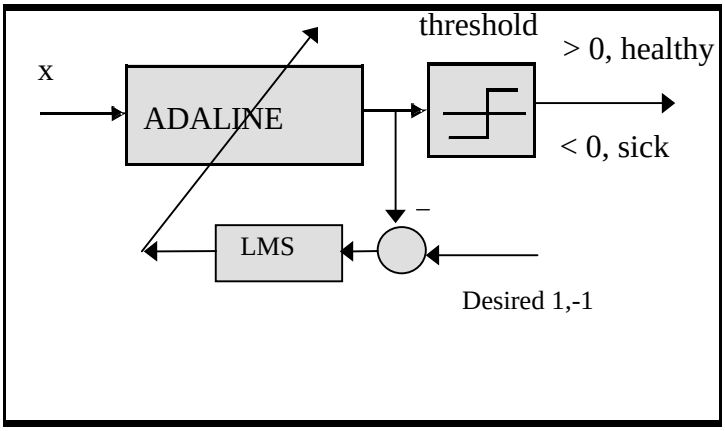


Figure 14. Schematic training of the adaline with threshold.

Nevertheless, the adaline followed by a threshold as shown in example 1 can implement a classifier. It was applied in the 1960's by Widrow and Hoff for classification purposes.

2.8. 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 this data is utilized 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 I.

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, and
- the *functional form* of the discriminant function.

There are two different ways to utilize the data for training parametric classifiers (Figure 15): they are called parametric and nonparametric training (do not confuse parametric classifiers with parametric training).

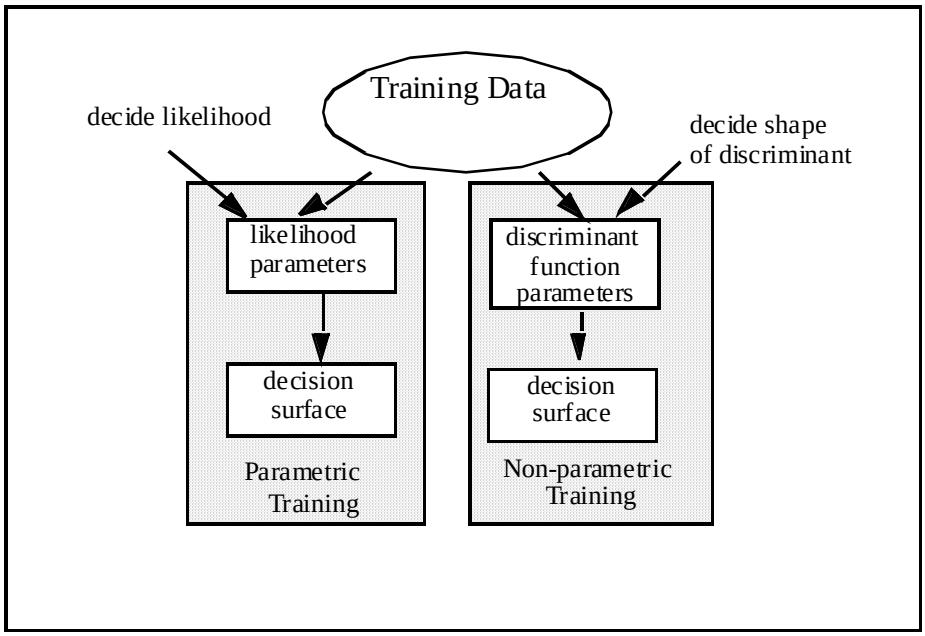


Figure 15. Parametric and nonparametric training of a classifier

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 parameters are estimated from the data. For instance, for Gaussian distributed pattern categories, one needs to estimate the mean vector, the covariance (normally using the sample mean and the sample covariance) and the class probabilities to apply Eq.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 non-parametric training. Very frequently nonparametric training utilizes iterative algorithms to find the best position of the discriminant functions. However, the designer

has to address directly the two fundamental issues of parametric classifier design, i.e. the functional form of discriminant functions and their placement in pattern space.

2.8.1. Parametric versus nonparametric training

Let us raise an important issue. Is there any advantage in using nonparametric training? The optimal discriminant function depends upon 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. So, 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 optimum 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 above that the performance of the quadratic classifier depends upon 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*, i.e. for the number of free parameters in the optimal classifier there is not enough data to properly estimate its discriminant function parameters. trade-offs of parametric training

  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 (like the linear) as we showed in shapes of 2D discriminants . These discriminants have fewer parameters and are less sensitive to estimation errors (take a look at Table II and compare the estimations for the means and variances) so they should be used instead of the quadratic when the data is not enough to estimate the parameters accurately.

  This raises the question of utilizing the data to directly estimate the parameters of the

discriminant functions, i.e. use 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. The difficulties that are brought by nonparametric training are twofold:

- deciding the shape of the discriminant function for each class, and
- ways to adapt the classifier parameters.

We can use the ideas of iterated training algorithms to adapt the classifier parameters. In Chapter I we trained the adaline parameters directly from the data, so effectively the linear regressor was nonparametrically trained. We can also *foresee the use of the gradient descent method explained in Chapter I to adjust the parameters of the discriminant functions* such that a measure of the misclassifications (output error) is minimized. Hence we have a methodology to place the discriminant function and are left with the choice of the functional form of the discriminant function, which unfortunately does not have a clear cut methodology.

2.8.2. Issues in nonparametric training

The central problem in nonparametric training of parametric classifiers can be re-enunciated as the selection of an appropriate functional form for the discriminant function which:

- produces small classification error, and
- have 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. So solving realistic problems may require more versatile machines. These more versatile machines are called *semi-parametric* classifiers because they still work with parametric discriminant functions, but they are able to implement a larger class of discriminant shapes (eventually any shape which makes them universal approximators). Semi-parametric classifiers are very promising because they are an excellent compromise between versatility and number of

trainable parameters. Artificial neural networks are one of the most exciting type of semi-parametric classifiers and will be the main subject of our study.

The other issue is to find fast and efficient algorithms that are able to adapt the parameters of the discriminant functions.We now know of training methods based on gradient descent learning that are pretty robust as we have demontrated in Chapter I. They will be extended in the next chapter for classifiers.

Go to next section

3. Conclusions

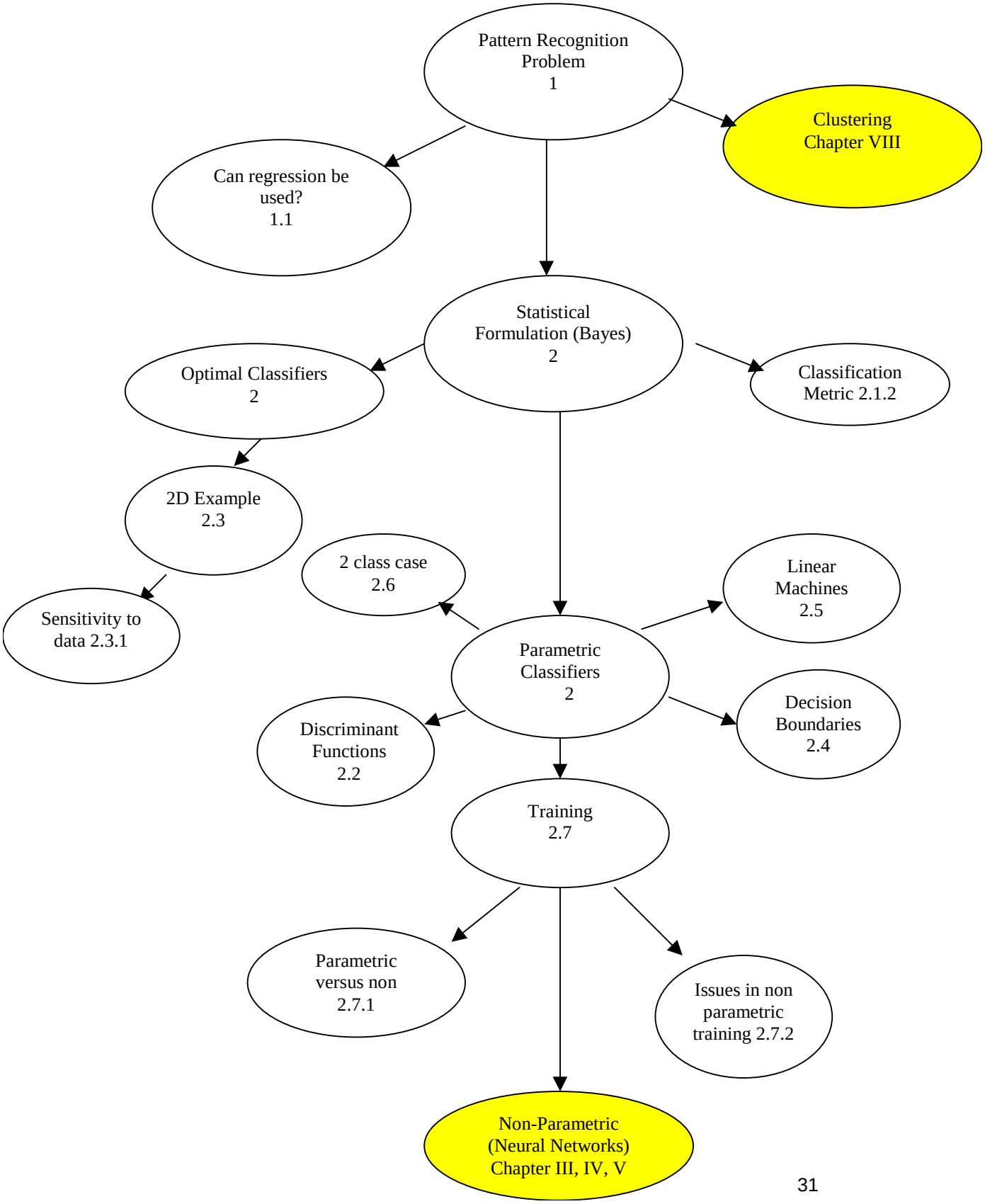
In this short chapter we covered the fundamentals of pattern recognition. We started by reviewing briefly the problem of pattern recognition from a statistical perspective. We provided the concepts and definitions to understand the role of a pattern recognizer. We covered the Bayes classifier and showed that the optimal classifier is quadratic. Sometimes sub-optimal 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 is provided. These concepts are going to be very important when we discuss artificial neural networks in the next chapter.

NeuroSolutions Examples

2.1 Comparing regression and classification

Chapter II


Pattern Recognition Problem 1
Clustering Chapter VIII
Can regression be used? 1.1
Statistical Formulation (Bayes) 2
Optimal Classifiers 2
Classification Metric 2.1.2
2D Example 2.3
Sensitivity to data 2.3.1
2 class case 2.6
Linear Machines 2.5
Parametric Classifiers 2
Discriminant Functions 2.2
Decision Boundaries 2.4
Training 2.7
Parametric versus non 2.7.1
Issues in non parametric training 2.7.2
Non-Parametric (Neural Networks) Chapter III, IV, V

Go to next Chapter

Go to the Table of Contents

Go to the Appendix

Understanding Bayes rule

There are two types of probabilities associated with an event: the *a priori* and the *a posteriori* probabilities. Suppose that the event is " x belongs to class c_1". The *a priori* probability is evaluated prior to any measurements, so it is called *a priori*. If there is no measurement, then the *a priori* probability has to be defined by the relative frequency of the classes, which is $P(c_1)$. However, we can also estimate the probability of the event after making some measurements. For a given *x* we can ask what is the probability that *x* belongs to class c_1, and we denote it by $P(c_1|x)$. This is the *a posteriori* probability.

According to statistical pattern recognition, for classification what matters are the *a posteriori* probabilities $P(c_i|x)$. But they are generally unknown. Bayes rule provides a way to estimate the *a posteriori* probabilities. In fact, eq2 tells that we can compute the posterior probability by multiplying the prior for the class ($P(c_i)$) with the likelihood that the data was produced by class i. The likelihood $p(x|c_i)$ is the conditional of the data given the class, i.e. if the class is c_i what is the likelihood that the sample *x* is produced by the class? The likelihood can be estimated from the data by assuming a probability density function (pdf). Normally the pdf is the Gaussian distribution. So, using Bayes rule one has a way to estimate *a posteriori* probabilities from data.

Return to text

Bayesian threshold

The general methodology is to equate the two *a posteriori* probabilities, substitute the likelihoods and obtain the value of *x* (temperature). For Gaussian distributions (eq. 3), notice that *x* appears in the exponent, which complicates the mathematics a little bit. However, if we take the natural logarithm of each side of the equation we do not change the solution since the logarithm is a monotonically increasing function. This simplifies the solution a lot. Let us do this for the two-class case. We get

$$P(c_1)\sigma_2 e^{-\frac{1}{2}\left(\frac{x-\mu_1}{\sigma_1}\right)^2} = P(c_2)\sigma_1 e^{-\frac{1}{2}\left(\frac{x-\mu_2}{\sigma_2}\right)^2}$$

Now taking the logarithm

$$\ln(P(c_1)) + \ln(\sigma_2) - \frac{1}{2}\left(\frac{x-\mu_1}{\sigma_1}\right)^2 = \ln(P(c_2)) + \ln(\sigma_1) - \frac{1}{2}\left(\frac{x-\mu_2}{\sigma_2}\right)^2$$

It is clear that the solution involves the solution of a quadratic equation in *x*. With simple algebra the solution can be found, which corresponds to the threshold T,

$$\frac{x^2}{2}\left(\frac{1}{\sigma_2^2} - \frac{1}{\sigma_1^2}\right) + x\left(\frac{\mu_1}{\sigma_1^2} - \frac{\mu_2}{\sigma_2^2}\right) - \left[\ln\left(\frac{P(c_1)}{P(c_2)}\right) + \ln\left(\frac{\sigma_1}{\sigma_2}\right) + \frac{1}{2}\left(\frac{\mu_1^2}{\sigma_1^2} - \frac{\mu_2^2}{\sigma_2^2}\right)\right] = 0$$

The solution is rather easy to find when $\sigma_1=\sigma_2$, since in this case the second order term vanishes and the solution is

$$x = \frac{\mu_1 + \mu_2}{2} + k$$

where k is dependent upon the ratio of *a priori* probabilities. This solution has a clear interpretation. When the variances are the same and the classes are equally probable, the threshold is placed halfway between the cluster means. If in our problem the two variances were the same the value for x=T= 37.75 C .

The *a priori* probabilities shift the threshold left or right. If the *a priori* probability of class 2 is smaller than the *a priori* probability of class 1 the threshold should be shifted towards the class with smaller probability. This also makes sense because if the *a priori* probability of one class is larger, we should increase the region that corresponds to this class to make fewer mistakes.

For the general case of σ_1 different from σ_2 one has to solve the quadratic equation. The distributions intersect in two points (two roots), but only one is the threshold (it has to be within the means). In our case for 1,000 measurements, the solutions are x_1=34.35 and x_2=37.07, so the threshold should be set at 37.07 C to optimally classify sick from healthy. Notice that this result was obtained with the assumptions of Gaussianity, the *a priori* probabilities chosen, and the given population (our measurements).

Note that the different variance of the classes effectively moved the threshold to the left, i.e. in the direction of the smallest variance. This makes sense because a smaller variance means that the data is more concentrated around the mean, so the threshold should also be moved closer to the class mean. Therefore we conclude that the threshold selection is dependent upon the variances of each cluster. What matters for classification is a new distance which is not only a function of the distance between cluster means but also the variances of the clusters.

Return to Text

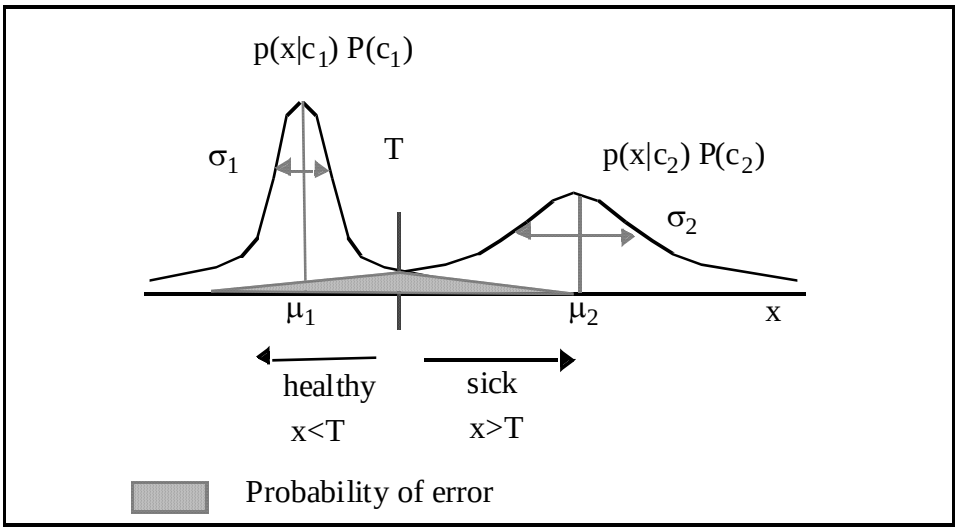
minimum error rate

The probability of error is computed by adding the area under the likelihood of class 1 in the decision region of class 2 with the area under the likelihood of class 2 in the decision region of class 1. Since the decision region is a function of the threshold chosen, the errors depend upon the threshold. As we can see from the figure*, the error is associated with the tails of the distributions*. In order to estimate this error one needs to integrate the likelihoods in certain areas of the input space, which becomes very difficult in high

dimensional space. The probability of error is

$$P(error) = \int_{R_2} p(x|c_1)P(c_1)dx + \int_{R_1} p(x|c_2)P(c_2)dx$$

where R_1 and R_2 are the regions assigned to class 1 and class 2 respectively, so it is a function of the threshold. One can show Fukunaga that the minimum error rate is achieved with the Bayes rule, i.e. by selecting the threshold such that the *a posteriori* probability is maximized. This result also makes sense intuitively (see the figure below).



The classification error is dependent upon the overlap of the classes. Intuitively, the larger the difference between the cluster centers (for a given variance), the smaller will be the overlap, so the smaller is the overall classification error. Likewise, for the same difference between the cluster means, the error is smaller if the variance of each cluster distribution is smaller. *So we can conclude that what affects the error is a combination of cluster mean difference and their variance*.

Return to text

parametric and nonparametric classifiers

The classifier we just discussed is called a *parametric classifier* because the discriminant

functions have a well defined mathematical functional form (Gaussian) that depends on a set of parameters (mean and variance). For completeness, one should mention *nonparametric classifiers*, where there is no assumed functional form for the discriminants. Classification is solely driven by the data (as in K nearest neighbors – see Fukunaga). These methods require lots of data for acceptable performance, but they are free from assumptions about shape of discriminant functions (or data distributions) that may be erroneous.

Return to Text

mahalanobis distance

The Mahalanobis distance is the exponent of the multivariate Gaussian distribution which is given by

$$p(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{(x-\mu)^T \Sigma^{-1}(x-\mu)}{2}\right)$$

where T means the transpose, $|\Sigma|$ means the determinant of Σ, and Σ^{-1} means the inverse of Σ. Note that in the equation μ is a vector containing the data means in each dimension, i.e. the vector has dimension equal to *d*.

$$\mu = \begin{bmatrix} \mu_1 \\ \dots \\ \mu_d \end{bmatrix}$$

Normally we estimate μ by the sample mean Eq.4 . The covariance is a matrix of dimension *d* x *d* where *d* is the dimension of the input space. The matrix Σ is

$$\Sigma = \begin{bmatrix} \sigma_{11} & .. & \sigma_{1d} \\ \dots & \dots & \dots \\ \sigma_{d1} & \dots & \sigma_{dd} \end{bmatrix}$$

and its elements are the product of dispersions among pairs of dimensions

$$\sigma_{ij} = \frac{1}{N-1}\sum_i \sum_j \left(x_i - \mu_i\right)\left(x_j - \mu_j\right)$$

The covariance measures the variance among pairs of dimensions. Notice the difference in number of elements between the column vector μ (d components) and the matrix Σ (d^2 components). See the Appendix

The Mahalanobis distance formalizes what we have said for the 1-D classification example. Notice that this distance is a normalized distance from the cluster center. In fact, if we assume that Σ= I (identity matrix), we have exactly the Euclidean distance betweeen the cluster centers. But for classification the dispersion of the samples around the cluster mean also affects the placement of thresholds for optimal classification. So it is reasonable to normalize the Euclidean distance by the sample dispersion around the mean what is measured by the covariance matrix.

Return to text

covariance

The covariance matrix was defined in the Mahalanobis distance explanation. The covariance matrix for each class is formed by the sample variance along pairs of directions in the input space. The covariance matrix measures the density of samples of the data cluster in the radial direction from the cluster center in each dimension of the input space. *So it quantifies the shape of the data cluster*.

The covariance matrix is always symmetric and positive semi-definite. We will assume that it is positive definite, i.e. the determinant is always greater than zero. The diagonal elements are the variance of the input data along each dimension. The off-diagonal terms are the covariance along pairs of dimensions. If the data in each dimension are statistically independent, then the off-diagonal terms of Σ are all zero and the matrix

becomes a diagonal matrix.

The structure of the covariance matrix is critical for the placement and shape of the discriminant functions in pattern space. In fact, the distance metric important for classification is normalized by the covariance, so if the class means stay the same but the covariance changes, the placement and shape of the discriminant function will change.

We will show this below with figures.

Return to text

derivation of quadratic discriminant

We saw that Bayes rule chooses classes based on *a posteriori* probabilities. We can think that $P(c_i|x)$ is a discriminant

$$g_i(x) = P(c_i|x) = p(x|c_i)P(c_i)$$

where $p(x|c_i)$ is the likelihood associated with the class c_i. In this expression we omitted P(w) (see eq2) because it is a common factor on all discriminants so will not affect the overall shape nor placement of the boundary. It can therefore be dropped for the definition of the discriminant function.

Now let us take the natural logarithm of this equation and obtain

$$g_i(x) = \ln p(x|c_i) + \ln P(c_i)$$

This is the general form for the discriminant, which depends on the functional form of the likelihood. If the density $p(x|c_i)$ is a multivariate normal distribution we get the equation in the text.

$$g_i(x) = -1/2(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i) - d/2\log(2\pi) - 1/2\log|\Sigma_i| + \log P(c_i)$$

Estimating all the elements of the Σ matrix in high dimensional spaces with adequate

precision becomes a nontrivial problem. Very often the matrix becomes ill-conditioned due to lack of data, resulting in discriminants that have the wrong shape, and so will perform sub-optimally.

Return to Text

Bayes classifier

The optimal classifier (also called the Bayes classifier) is obtained in the same form as for the 1D case. We substitute the means and covariances estimated from the data for each class in Eq.8 . The inverse of the covariance for the women is

$$\Sigma^{-1} = \begin{bmatrix} 0.0111 & 0 \\ 0 & 277.78 \end{bmatrix}$$

and the determinant of Σ is 0.3256 yielding

$$g_w(x) = -0.5\begin{bmatrix} x_1 - 64.86 & x_2 - 1.62 \end{bmatrix}\begin{bmatrix} .00111 & 0 \\ 0 & 277.78 \end{bmatrix}\begin{bmatrix} x_1 - 64.86 \\ x_2 - 1.62 \end{bmatrix} - \log(2\pi) - 0.5\log(0.3256)$$

For the man class the discriminant is

$$g_m(x) = -0.5\begin{bmatrix} x_1 - 78.02 & x_2 - 1.75 \end{bmatrix}\begin{bmatrix} .00032 & 0 \\ 0 & 123.46 \end{bmatrix}\begin{bmatrix} x_1 - 78.02 \\ x_2 - 1.75 \end{bmatrix} - \log(2\pi) - 0.5\log(2.512)$$

The separation surface is obtained by equating $g_x(x) = g_f(x)$ which yields

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

This is a quadratic in 2D space as shown in Figure 5. This surface yields the smallest classification error for this problem. But just by inspection of the figure one can see that many errors are going to be made. So one has to get used to the idea that optimal does

not necessarily means good performance. It simply means the best possible performance with the data we have.

Return to text

shapes of 2D discriminants

Diagonal Covariance matrix

If the two variables are uncorrelated and of the same variance, then the covariance matrix is diagonal

$$\Sigma = \sigma^2 I$$

In this case the Mahalanobis distance defaults to the Euclidean distance

$$\left\| x - \mu_i \right\|^2 = (x - \mu_i)^T (x - \mu_i)$$

and the classifier is called a *minimum distance classifier*. The interesting thing is that the discriminant function for this case defaults to a linear function

$$g_i(x) = w_i^T x + b$$

where $w_i = \frac{1}{\sigma^2}\mu_i$ and $b = -\frac{1}{2\sigma^2}\mu_i^T \mu_i$ since the quadratic term is common to both classes and does not affect the shape of the discriminant. For this case the samples define circular clusters (hyperspherical in multidimensions).

Equal Covariance Matrix

The case of equal covariance matrices for each class ($\Sigma_i = \Sigma$) is still pretty simple. In fact the discriminant is still linear but now the weights and bias of gi(x) in the previous equation are given by $w_i = \Sigma^{-1}\mu_i$ and $b = -\frac{1}{2}\mu_i^T \Sigma^{-1} \mu_i$,which means that each

class is a ellipsoidal cluster of equal size an shape. Figure 6 shows both cases where we would like to point out that the decision regions are both linear functions.

Arbitrary Covariances

This is the most general case, and in this case the general form of the discriminant function of Eq.8 must be used. We can see that this discriminant function is quadratic in *x*

$$g_i(x) = x^T W_i x + w_i^T x + b$$

where $W = -1/2\Sigma_i^{-1}$, $w_i = \Sigma^{-1}\mu_i$ and $b = -\frac{1}{2}\mu_i^T\Sigma^{-1}\mu_i - \frac{1}{2}\log|\Sigma_i|$. The decision region is either a line, circle, ellipse and parabola, depending upon the shape of the individual clusters and their relative position (Figure 8).

These three cases illustrate our previous statement that the covariance matrix is exceptionally important in the definition of the shape (and placement) of the discriminant function. Note that the discriminant function changed from an hyperplane to a quadratic surface depending upon the shape of the covariance matrix of each class.

A classifier built from linear discriminant functions (called a linear classifier) only exploits differences in means among different classes, while the quadratic classifier not only exploit the mean difference but also the difference in “shape” of the data clusters. Hence, if we have two classes with the same mean, the linear classifier will always give very poor results. However, a quadratic classifier may perform better as long as the shape of the two data clusters are different. In fact notice that for Gaussian distributed classes the optimal classifier is a quadratic classifier given by Eq.8 . So there is no need to find more sophisticated (higher order) discriminant functions.

The improvement in performance between the linear and the quadratic discriminant comes at a computational cost. In fact, the number of parameters estimated for the arbitrary covariance case is 7 per class for the quadratic (increases as the square of the dimension due to W), while it is simply 3 for each of the linear cases. So quadratic

discriminants require more data samples to estimate reliably their parameters.

Return to text

parametric and nonparametric training

Parametric training uses the data to estimate the parameters of the data models, which are then utilized to specify the discriminant functions. This was the method utilized in the statistical modeling of the data.

Alternatively, the parameters of the discriminant function can be estimated in a way which relaxes the assumptions necessary for parametric training. Instead of using the data to estimate the parameters of the assumed data distributions to compute the likelihoods, we can *pre-select* a functional form for the discriminant function (e.g. a linear discriminant or a quadratic discriminant) and adjust its parameters *directly* from the data. Most oftern we use iterated learning rules to accomplish this adaptation. This alternate procedure is commonly called *non-parametric training.*

Return to text

trade-offs of parametric training

In parametric training one needs to estimate the probability density function of the input data, which is an ill-posed problem from finite number of observations. So to solve the classification problem we are forced to solve a much harder problem of estimating the pdf. This is not done in practice. What we do is to hypothesize a pdf and then find its parameters from the available data.

We saw before (Eq.6) that the essence of the classification test can be maintained, even when the discriminant functions are not derived from the assumed statistical properties of the data. There is quite a bit of freedom in the selection of the shape of the discriminant

function. Just remember that the Gaussian discriminant is effectively equivalent to a quadratic discriminant. Different discriminant functions can provide the same classification accuracy (i.e. the same decision surface), which leads naturally to the search for simpler functional forms for the discriminant functions.

Moreover, one may have to trade optimality for robustness. For instance, the optimal discriminant function of Eq.1 for the Gaussian model requires the estimation of class means and covariances. In high dimensional spaces the estimation of covariances requires large data sets, otherwise the estimated values may be far from the true ones. Poor parameter estimation will lead to misplaced/mishaped discriminant functions, which will then produce higher classification errors.

For instance, we saw that the quadratic discriminant is optimal for Gaussian distributed clusters. But if we want to classify hand written digits from a 20x20 image (a 400 dimensional input), the quadratic discriminant requires more than 160,000 parameters (400x400 covariance matrix). In order to reliably estimate all the entries of this matrix one would need 1,600,000 data samples (using the rule of thumb that one needs 10 times more samples than parameters). So, it is impractical sometimes to utilize the optimal discriminant function, which points to alternate ways of designing and training classifiers.

Return to text

R. A. Fisher

Was a British statistician who proposed in the late 20's the use of the Maximum Likelihood principle to solve the discriminant analysis problem (i.e. a rule to distinguish between two sets of data).

Pattern Recognition

is the creation of categories from input data using implicit or explicit data relationships. What is similar among some data exemplars is further contrasted with dissimilarities

across the data ensemble, and the concept of data class emerges. Due to the imprecise nature of the process, it is no surprise that statistics has played a major role in the basic principles of pattern recognition.

Pattern Space

is the space of the input data. Each multivariate (N variables) data sample can be thought as a point in a multidimensional (N dimensional) space.

classes

are the natural divisions of the input data produced by the phenomenon under study (i.e. sick and healthy in this case).

classifier

is a machine that automatically divides input data into classes.

decision surface

is the boundary (eventually multidimensional) between the input data classes.

discriminant functions

is a function g(x) that evaluates every position in pattern space and produces a large value for one class and low values for all the others.

training the classifier

involves defining the parameters of the discriminant function parameters from the input data (the training set)

optimal classifier

the classifier that minimizes the classification error given the observations.

optimal discriminant function

the discriminant that produces the best possible classification.

linear machine

is a parametric classifier where the discriminant functions are hyperplanes.

a posteriori probability

is the probability of an event after some measurements are made.

likelihood

is the probability density function of each event.

probability density function

intuitively, is the function that specifies the probability of a given event in an experiment. It is the limit of the histogram for arbitrary large number of trials. See the Appendix for a definition.

eq2

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

adaline

stands for ADAptive LInear Element, and is the processing element proposed by Widrow that implements a weighted sum of inputs.

Eq.1

x belongs to c_i if $P(c_i|x) > P(c_j|x)$ for all j $\neq$ i

Eq.6

$\boldsymbol{x}_k$ *belongs to* c_i *if* $g_i(\boldsymbol{x}_k) > g_j(\boldsymbol{x}_k)$ *for all* $j \neq i$

Eq.8

$$g_i(x) = -1/2(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i) - d/2\log(2\pi) - 1/2\log|\Sigma_i| + \log P(c_i)$$

Eq.10

$$g_x = w_1x_1 + w_2x_2 + \ldots + w_Dx_D + b = \sum_{i=1} w_ix_i + b$$

convex

a surface is convex when any point in a line joining two points in the surface belongs to the surface.

Eq.9

$$g_{new}(x) = g_{healthy}(x) - g_{sick}(x)$$

LMS

$w(n+1) = w(n) + \eta x(n)\varepsilon(n)$

Eq.7

$$r^2 = (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)$$

Widrow

Widrow also utilized the adaline for classification by including a nonlinearity after the linear PE.

Widrow and Hoff, “Adaptive switching circuits”, IRE WESCON Convention Record, 96-104, 1960.

Eq.4

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

Eq.3

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

Duda

Duda R. and Hart P., Pattern Classification and Scene Analysis, Wiley, 1973.

Fukunaga

Fukunaga K., Statistical Pattern Recognition, Academic Press, 1990.

ill-posed

a problem is ill-posed when a small modifications on the input produces large change in the outcome.

size of feature space

The conventional way of thinking about features in pattern recognition has been as a dimensionality reduction operation. A feature is a characteristic of the input data that preserves discriminability. For instance, color, or edges in an image. Features are traditionally obtained with projections to a sub-space, that is, features have smaller dimensionality than the input. For 50 years, the quest in pattern recognition has been to find small dimensional projections that preserve the most information about the input. One realizes that a projection to a sub-space will reduce the discriminability somewhat, but experience has shown that for many problems a handful of small dimensional projections produce workable classifiers.

The motivation to perform subspace projections is related to the “curse of dimensionality” we already mentioned. If we want to build the optimal Bayes classifier in a high dimensional space we have to estimate too many parameters with the available data. One way to conquer this difficulty is to project the data to a smaller dimensionality space (feature extraction) and there develop the optimal classifier. Since the feature space is smaller there will be less parameters to estimate. The difficulty has been in choosing the

features and their number. Moreover, if the data is hard to classify in the original space, very likely it will also be difficult to classify in the smaller dimensional feature space. Nevertheless this has been the traditional way to think about feature spaces.

The view we just expressed of using Cover's theorem to go to a higher dimensional feature space has been a curiosity in pattern recognition until the recent work of Vapnik with the Support Vector Machines. High dimensional spaces produce sparse data clusters, i.e. no matter how many data samples we might have, if the size of the space is sufficient large, there will always lots of room among the data clusters. This means that the classification problem is potentially linearly separable. So a very simple classifier can do the job. The interesting thing is that the generalization can still be controlled by enforcing a large margin. We will discuss this later.

Return to Text

cover's theorem

Cover's theorem draws from a very interesting (and non-intuitive) result. Let us assume that we a have a kernel based machine with M+1 weights. We want to classify N patterns. There are 2N possible divisions of these patterns into two categories (called dichotomies). The question is what is the probability that a given dichotomy chosen at random is implementable by our kernel based machine?

Cover showed that the probability is

$$P_{N,M} = \begin{cases} 2^{1-N} \sum_{i=0}^{M} \binom{N-1}{i} & N > M \\ 1 & N \leq M \end{cases}$$

This means that for spaces of size M larger than the number of samples N the probability is actually one, that is, we can always divide any data into two classes with probability 1. This probability has a sharp knee (the probability approaches one rapidly) at N=2(M+1),

and this value is normally defined as the capacity C of the learning machine. So if we set the number of free parameters (weights) of our learning machine above its capacity we are almost certain to classify the input patterns correctly. For a linear classifier in D space the capacity is C=2D+1. However, kernel based classifiers allow us to go to a higher dimensional space and set their capacity independent of the input space dimension. This decoupling between the input space dimension and the machine capacity is the great advantage of the kernel based machine.

Return to Text

Vapnik

The nature of statistical Learning Theory, Springer Verlag, 1995, pp 136.

Nilsson

Mathematical Foundation of Learning Machines, 1990 Morgan Kaufmann.

Affine

An affine transformation specifies any linear transformation between the space coordinates. We can think of them as a rotation and a translation of the space.

Index

1

1. The Pattern Recognition Problem 2

2

2. Parametric classifiers 4

9

9. Conclusions 21

B

Bayes classifier 28
Bayesian threhold 23

C

Chapter II-Multilayer Perceptrons 2
covariance 26, 27

D

derivation of quadratic discriminant 27

M

mahalanobis distance 26
minimum error rate 25

P

parametric and nonparametric classifiers 25
parametric and nonparametric training 30

S

shapes of 2D discriminants 28

T

trade-offs of parametric classifiers 30

U

Understanding Bayes rule 23