

Notes for SIMG-784
Digital Image Processing and Pattern
Recognition

Harvey E. Rhody

Winter Quarter 1997

Contents

1	Overview	5
1.1	Introduction	5
1.2	Abstract Model	8
1.2.1	Decision Model	11
1.3	Example: Recognize Sports Figures	13
1.4	Classification Rules	14
1.5	Problems	17
2	Decision Theory	19
2.1	The Decision Function	20
2.1.1	Bayes Rule for Minimum Risk	23
2.1.2	Computational Considerations	24
2.2	Problems	35
3	Multivariate Normal Model	37
3.1	Introduction	37
3.1.1	Modeling Experimental Data	39
3.1.2	Example: Three Classes with Two Features	40
3.1.3	Calculation of Model Parameters	44
3.1.4	Minimum Error Decisions	44
3.1.5	Decision Regions and Decision Boundaries with minumum error decisions	46
3.2	Problems	48
3.3	Computer programs	51

Chapter 1

Overview

1.1 Introduction

Pattern recognition is a natural ability that is so fluent that we don't think about it. Like many other natural abilities, it is something we would like to build into machines so they could do more for us. To give this ability to a machine it is necessary to understand it in detail so we can create the required components and systems. We have made a lot of progress in this task, and we are able to build useful pattern recognition systems. However, their capabilities are far from those of natural beings. We would like to understand the current methods and master the tools for creating better systems.

Natural recognition involves the perception of objects in nature and responding to them appropriately. In animals, perception begins with the senses which produce stimuli to the central nervous system which then causes a response such as attack or flight. Some objects, such as the appearance of food or a threat, cause dramatic action while others that do not concern the creature produce little or no response. In addition to having the creature responses, humans also have the ability to describe the objects and events that are sensed. A linguistic description is a symbolic representation that associates words with things in nature. Included in the linguistic description are the names of classes of objects that are perceived.

If you look around a room and see various items of furniture, you are

easily able to classify them as `chair`¹, `table`², `desk`³ and so on. There are many objects in the world that would be included in any one of the classes, and each object is different from all the others at some level of detail. Can you think of a way to define each of these classes such that it unambiguously and simply includes all objects that belong to it and no others? The definition, to be useful in robotics, would have to be in terms that can be related to the activity of possible robot sensors.

Objects in nature simply exist on their own terms. They are put into categories by observers, but the classification they are given does not affect what they “really” are. Classification is in some sense arbitrary. This gives us the flexibility to construct the classes as we wish, to best suit our own needs. It is not necessary to give things symbolic names to classify them, as when a dog recognizes a rabbit or vice versa. We interpret animals as responding in terms of classes, but I doubt that they think that way. However, we use categories to name things and it is a useful conceptual shorthand. Biological classification is a science⁴ that we impose on nature to support the development of our understanding. The creatures and plants don’t care.

Classifiers for robotic systems often depend on the use of a *pattern* which serves as a representative example or model of the class. The pattern does not necessarily exist as a real physical example, but must exist as a model within the brain of the robot. If the model was obtained from a real object, then the pattern exists in both senses. Pattern recognition is implemented by the comparison of observations to internal models of classes.

Pattern recognition, as it is ordinarily used in the engineering literature, means the classification of observations through the use of internal models. Recognition is limited to classification, and the two words are synonyms in this domain. Key issues are tied to sensing mechanisms, means for internal representation of models, means for comparison of observations to models, and systems for making decisions.

¹A dictionary definition is: “A piece of furniture consisting of a seat, legs, back, and often arms, designed to accommodate one person.” Note the mixture of parts and function. Of course, the parts would have to be defined for a robot, as well as their relationships. How would one define the function so it could be understood?

²An article of furniture supported by one or more vertical legs and having a flat horizontal surface.

³A piece of furniture typically having a flat or sloping top for writing and often drawers or compartments.

⁴The systematic grouping of organisms into categories on the basis of evolutionary or structural relationships between them; taxonomy.

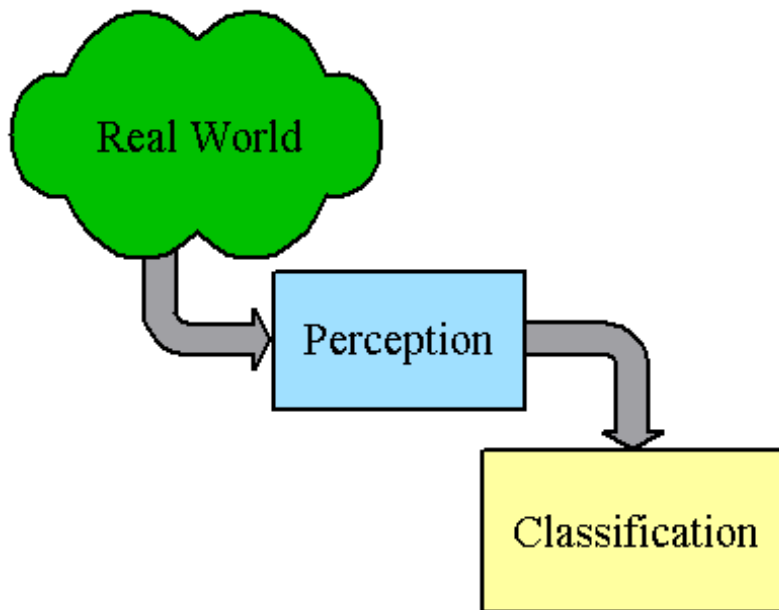


Figure 1.1: The basic recognition processes modeled as two steps: perception and recognition.

It is helpful to visualize the pattern recognition process as consisting of perception and recognition, as illustrated in Figure 1.1. The task of perception is to receive information about the real world and put it in the form of some kind of signals for the recognition system. The recognition system must then classify the signals in terms of one or more objects that the system knows about.

When the recognition system has completed its task it will have constructed an internal list of labels that are associated with particular physical items in the real world. The degree to which the list is accurate and complete depends upon the capability of the recognizer. The performance must be judged in relation to the required use of the recognizer. Simple recognizers can be very effective in very constrained environments with well-structured tasks. More sophisticated recognizers are needed for more open environments and less structured tasks.

Humans and animals do not build internal lists in any literal sense. Rather, they build much more sophisticated internal representations of the perceived environment. The subject of these representations and the means

to achieve them is at the frontier of current research. The machines we build do indeed make use of lists or their equivalent. These crude representations of reality can be very useful in many automation applications and are the domain of pattern recognition.

1.2 Abstract Model

Pattern recognition may be visualized with an abstract model. This model incorporates the basic elements of many pattern recognition problems without dealing with the specifics of their solution. It provides the first, most general, level of system structure and an idea of the workings of pattern recognizers without getting into implementation specifics.

The abstract model makes use of four "spaces" or sets of objects that are related by operational processes. The objects in each space may be used in various arrangements. The spaces are: physical reality (\mathcal{R}), signals (\mathcal{S}), features (\mathcal{F}) and patterns (\mathcal{P}). The goal is to associate elements in \mathcal{P} with elements in \mathcal{R} .

Physical reality, represented by the symbol \mathcal{R} , is the world of physical objects that exist in space and time. A portion of this space will be the focus of attention in a given situation. An observer can sense objects in this space by their emanations of light, sound, heat, etc. Objects in \mathcal{R} that are not of interest to the application may be left undescribed and may form part of the background. The background may produce responses in the sensors, which may be confusing (noise) or helpful (context).

A sensor is the part of the observer that responds to the emanations of objects in physical reality. The emanations that fall on the sensor produce signals in a form that can be processed by the observer. A signal is usually a simple transcription of the part of the emanation that the sensor is responsive to. For example, the voltage or current produced by a microphone in response to a sound pressure wave is a signal. We will represent the signal space by \mathcal{S} , and describe each signal as an element $s \in \mathcal{S}$. It is common to describe signals in a mathematical form, such as functions over time and space, but it is useful to remember that signals have physical existence and conform to physical constraints. Amplifiers and filters are physical devices that manipulate physical signals and algorithms are tools to manipulate mathematical signals. Digital signal processing provides means to implement algorithms in computers, where they can work on numerical representations of real signals.

Whether to do a process as a filter or as an algorithm is an implementation issue, not a fundamental conceptual issue.

It is important that the sensors be tuned to the physical environment so the objects of interest can be sensed. We are blind to ultraviolet and radio frequency radiation and would not know they exist without artificial sensors. The signals are specific to the sensors and are a transcription of a portion of the physical characteristics of the scene. All beings, natural and artificial, are limited by their sensors. What is lost at the sensor cannot be used by other processes.

Signals produced by sensors have a physical existence and therefore are objects in physical reality. A photograph is a signal in the form of colored patterns on a piece of paper produced by photographing a scene. The photograph is a flat representation of the scene, and is clearly not the same thing as the scene. The photograph can be observed at another time so that the state of the scene at an earlier time can be examined. The photograph itself can be examined as a physical object to determine its own physical properties. An image can be extracted from the photograph and converted to a computer compatible form by scanning. It then is represented by arrays of numbers, which can be processed in various ways. A physical image can be produced by printing a representation of the internal computer array onto paper, film or other suitable medium or displaying it on a computer screen. All these forms are simply signals that represent the original reality in greater or lesser degree.

An image may also be created, not by capturing physical reality, but by direct manipulation of the image medium. This artistic endeavor produces physical objects that have no causal scenic origin. As images they can create impressions in the mind of the viewer and as physical objects they can be scenic subjects. We will not consider images or other signals created for artistic reasons in this study, but it may be philosophically interesting to observe the relationship.

Signals may be directly produced by other forms of activity, such as computer aided design. An architectural design may be viewed as a virtual reality walk-through before it ever exists as a physical structure. An interesting reversal of signals into nature.

The design of pattern recognition system relies on the idea that only certain kinds of objects are to be sensed in a scene. The goal is to detect these objects of interest, not to describe the entire scene. The set of object models is the pattern space, \mathcal{P} . A pattern is a model for a particular *kind* of

object, such as **chair**, or even **restaurant chair**. The intent is to detect occurrences of elements of \mathcal{P} in the scene by examination of the signals produced by sensing both the objects and the scene in general. One key problem is the separation of the objects from the background in the signal milieu.

Recognition is accomplished by use of *features* that are extracted from the signals. A system that separates **football linemen** from **jockeys** might involve the measurement of height, weight and shoe size. The values (height, weight, shoe-size) are a set of features that, taken together, form a feature vector. The mathematical space \mathcal{F} of vectors that could be achieved by a particular set of features is called *feature space*. In this case, feature space would be in three dimensions, bounded and constrained to positive feature values.

Because the feature vectors contain numerical entries, it is possible to define a variety of distance measures. If \mathbf{x} and \mathbf{y} are two elements of \mathcal{F} , then the distance $d(\mathbf{x}, \mathbf{y})$ may be used to measure their similarity. The use of distance as a measure of similarity is a common technique in pattern recognition. In fact, it is extremely difficult to build pattern recognition systems that don't use some form of distance measure. All our systems will use some kind of distance measure. Therefore, (\mathcal{F}, d) will be a vector space, which makes all of the powerful mathematical tools associated with that discipline available to us.

The features used in a particular application depend upon the signals that are available and the capability of the processing system. Some sort of analysis must be done to extract the feature values from the signals. The use of a new signal processing algorithm may make available features that had not been used before and may solve a difficult recognition problem. For example, the fast Fourier transform (FFT) is a numerical algorithm that facilitates the use of spectral analysis and makes it possible to recognize objects with particular spectral signatures. The selection of features depends upon the insight of the designer, the characteristics of the sensors, and the availability of necessary algorithms. As we shall see, recognition is relatively simple when a good set of features is available and nearly impossible when they aren't. A large part of the art of pattern recognition system design is related to sensor characteristics and feature selection.

The pattern recognizer operates by deciding which object class to associate with each feature vector. Each $\mathbf{x} \in \mathcal{F}$ is classified as a pattern $\omega \in \mathcal{P}$. The last stage of the recognizer is therefore a decision-making device that

maps the continuous vector space \mathcal{F} into the discrete pattern space \mathcal{P} . The decision function $D : \mathcal{F} \rightarrow \mathcal{P}$ is chosen to maximize some performance measure, such as the average number of correct decisions or the lowest average cost. The modeling of the decision process is facilitated by concepts from probability and statistics. We will find that pattern recognition has many connections to the discipline of decision science.

The chain $\mathcal{R} \rightarrow \mathcal{S} \rightarrow \mathcal{F} \rightarrow \mathcal{P}$ represents the pattern recognition system. In many cases the physical setting \mathcal{R} and the patterns to be recognized \mathcal{P} are given and it is the task of the designer to construct the elements between them. The *arrows* \rightarrow represent processes and the sets represent intermediate signals and features. The processes and the elements of \mathcal{S} and \mathcal{F} can be varied, within practical constraints in designing the system. We will find that there are many variations that are possible. Our goal is to build a level of understanding and a kit of useful tools so that a wide range of applications can be addressed. It is worth keeping in mind the fact that in most applications the idea of a “best system” involves many considerations, such as complexity, cost, and error rate. The design usually consists of selecting the right structure and then maximizing the performance of that structure. It is often the case that choosing the right structure is the critical step, for which one must depend upon insight, intuition, experience and art. “Optimum” is a word that is used too often because it focuses on maximizing performance within a given structure and specified domain rather than the harder problem of selecting an appropriate structure from which to begin.

We may look for inspiration to natural biological pattern recognizers. We are beginning to learn how natural systems work and we are amazed at their speed, accuracy and adaptability. However, we are not able to duplicate either the physiology or the function of substantial parts of natural systems. The designer of an artificial system may try to use some knowledge gained by observing natural systems, but, in the end, must develop and implement a particular set of sensors and algorithms.

1.2.1 Decision Model

Pattern recognition may be viewed as the process of associating a pattern that exists in the real world with a model the internal pattern space of the recognition system. As a mathematical structure, we assume that the pattern space is a finite list, $\mathcal{P} = \{\omega_1, \omega_2, \dots, \omega_N\}$, and that some unknown member ω has occurred in \mathcal{R} . The observation produces a feature vector $\mathbf{x} \in \mathcal{F}$.

We assume that $\omega \in \mathcal{P}$ so that the task is to decide which member of \mathcal{P} produced the observation. Our decision, $\hat{\omega} = \omega_j \in \mathcal{P}$, should minimize our risk or maximize our payoff, at least on the average. If we can model the cost function and decision process then we can optimize it.

Let us assign a cost to each decision. Let l_{ij} denote the cost of making decision $\hat{\omega} = \omega_j$ when the correct choice is $\omega = \omega_i$. The cost is viewed as a “loss,” and in most applications the cost of a correct decision, l_{ii} , is set to zero while the other terms are positive. Suppose that an experiment is run n times and that n_{ij} is the number of times event ω_i is interpreted as event ω_j by the decision device. Then the average cost per trial is

$$L = \frac{1}{n} \sum_{i,j} n_{ij} l_{ij} \quad (1.1)$$

Under suitable assumptions concerning the repetition of a large number of experiments under identical conditions, the ratio n_{ij}/n can be interpreted as the probability p_{ij} . The average cost is then closely approximated by the expected value

$$E(L) = \sum_{i,j} p_{ij} l_{ij} \quad (1.2)$$

One reasonable design approach is to minimize $E(L)$ ⁵.

When the cost function is

$$l_{ij} = \begin{cases} 0, & i = j \\ 1, & i \neq j \end{cases} \quad (1.3)$$

the above model simplifies to minimizing the error probability (The proof is left as a homework problem.) In a nutshell, we are saying that all errors have the same cost so that the only way to minimize the expected cost is to minimize the expected number of errors.

If one can determine the probabilities associated with a given classifier, then it is simple to calculate the decision “cost.” In some cases reasonable-sounding assumptions can be made that enable the probabilities to be computed.

⁵Here we glide past some difficult issues, such as how an experiment can really be repeated a large number of times under identical conditions; how one determines whether or not it is possible to tell if the conditions are identical, and so on. This is the route into statistical modeling, and it is good to be wary of the assumptions that are implicitly involved. We will not deal with these issues in any significant way in this course.

Many classifiers have parameters that can be adjusted to modify their performance. Examples are threshold settings or summing weights. It is possible to keep adjusting the parameters until the lowest cost is obtained for that classifier structure for the given application. This is what is usually meant by optimization.

It turns out that it is easier to generalize the description of classifier algorithms than it is to describe a general framework for feature selection. Therefore, sensor design is often taken as a “given” in pattern recognition design, and effort is focused on the classifier algorithm. However, it is likely to be the case that much more can be gained by getting a better feature set than can be gained by working on the classifier algorithm. But one also has to know how to make the best use of the available information under any circumstances, and that is the subject of this course.

1.3 Example: Recognize Sports Figures

This is a completely useless conceptual application dreamt up to illustrate some of the structure of decision modeling. You will be able to construct your own examples once you have digested this one.

We suppose that you have to design a robot to classify sports figures. The first task is to build a machine that can tell the difference between football linemen and jockeys. Suppose the observations are numerical, so they can be plotted in a coordinate system. As an example, consider the features of **height** and **weight**. Then the data may be plotted in a diagram such as Figure 1.2. Example data from these classes is listed in Table 1.1.

It would not be difficult to design a classifier that would label the points as either **football lineman** or **jockey** on the basis of their location in this *feature space*. The data clearly falls into two well-defined clusters, and each cluster can be given an appropriate label.

In this problem there are two classes and two features. In general the number of classes and the number of features need not be the same⁶. The number of axes of the coordinate system is determined by the number of features, and the number of clusters is determined by the number of classes. If we added another class, such as **swimmers** to the data set, then the feature space might look like Figure 1.3. Example data from these classes is listed

⁶You probably don't have to be told that, but sometimes we form accidental associations that are hard to get rid of.

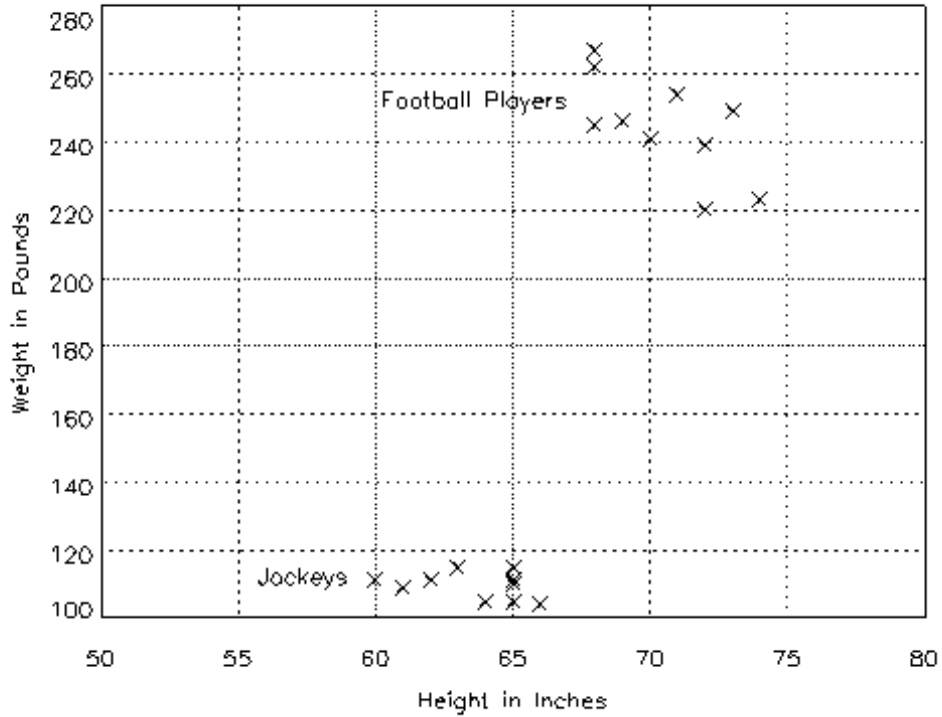


Figure 1.2: Feature space data for football linemen and jockeys.

in Table 1.2. There should now be three clusters, but it is more difficult to identify them.

1.4 Classification Rules

Suppose that we have observed a number of known sports figures and know where their measurements fall in feature space. How should the system process the data from a new figure to determine his sport?

One sensible approach would be to compare the features of the unknown figure to those of known figures. This approach is actually used in some cases

Jockey		Football	
Ht	Wt	Ht	Wt
64	105	69	246
61	109	68	254
65	111	72	220
63	115	72	239
65	105	74	223
66	104	70	241
65	115	71	254
62	111	73	249
60	111	68	267
65	110	68	262

Table 1.1: Example observations of jockeys and football linemen.

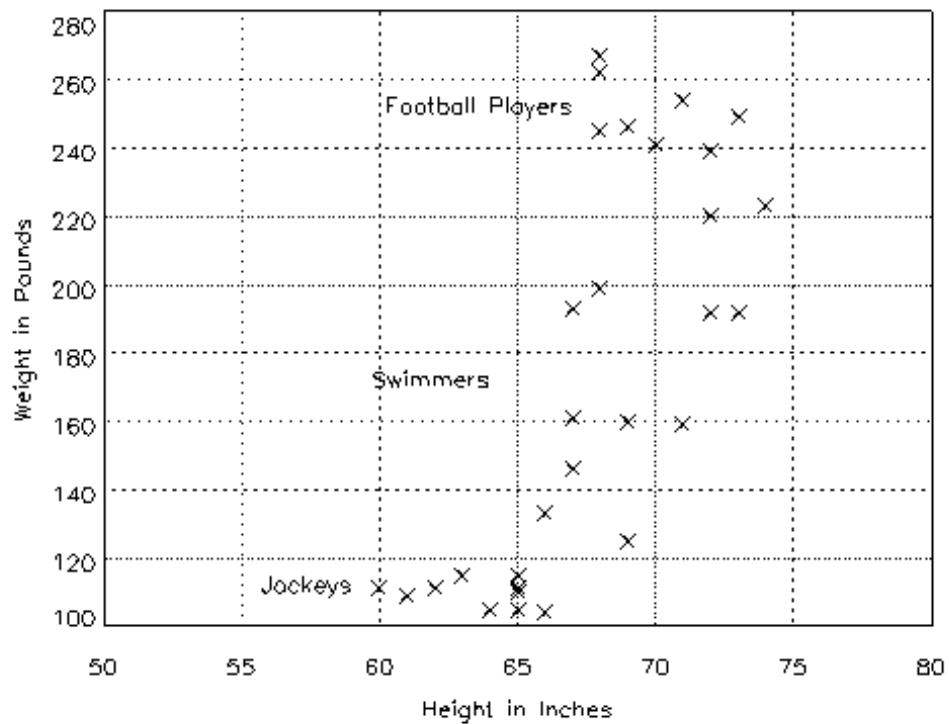


Figure 1.3: Feature space data for football linemen, jockeys and swimmers.

Jockey		Football		Swimmer	
Ht	Wt	Ht	Wt	Ht	Wt
64	105	69	246	66	133
61	109	68	254	71	159
65	111	72	220	73	192
63	115	72	239	67	193
65	105	74	223	69	125
66	104	70	241	72	192
65	115	71	254	69	160
62	111	73	249	67	161
60	111	68	267	67	146
65	110	68	262	68	199

Table 1.2: Feature space data for football linemen, jockeys and swimmers.

(nearest neighbor classifiers). An unknown point \mathbf{u} is classified in the same class as its nearest neighbor (1-nn) or in the class with the closest k points k -nn. This is simple, but may be computationally intensive. Think about how you would do it in a computer to see where the calculations would build up.

Another approach is to compare the unknown to a *model* of each class. When the data for the known figures falls into well-separated clusters, the models are simple to construct. Each model can be a point somewhere within its cluster. It could be an actual figure or a mathematical point, such as the centroid of the cluster. This approach is the foundation of such methods as gaussian maximum likelihood (GML). It is clearly dependent on the assumption of good well-separated clusters.

Yet another approach is to carve the feature space up into regions, with one region for each class. Classifying a point is then equivalent to determining its region in feature space. If it is easy to describe the regions, then this is a good approach. Optimizing the classifier then consists of drawing good region boundaries.

One characteristic we will want in a classification system is the ability to add new classes. We will find that some systems make this relatively easy (such as adding a new model in GML or some examples of the new class in k -nn) while others require a restructuring and retraining (such as most neural nets).

Adding or deleting features is another desirable capability. GML may be

the simplest for this because it involves updating only the models. However, because it involves changing the dimensionality of the feature space it is a process that always involves some work.

Our goal in this course is to gain a level of understanding of all of the common pattern recognition systems and the ability to make intelligent choices for particular applications. To do this we must work at multiple levels of thought: the problem, the system structure and the implementation of the chosen structure. We need computational theories, algorithms and ways to implement them. Each topic is relatively easy to understand, and is more so if it is kept in its proper position in our conceptual structure.

1.5 Problems

1. Show that the expected decision loss for a pattern classifier in which the losses for wrong decisions are equal is minimized by making all of the error probabilities equal. If the overall error probability is p_e then the minimum expected cost is p_e if the cost of an error is normalized to unity.
2. Create a model for each of the classes jockey, football player and swimmer by finding the vector that is the mean value of each class. Given a new observation $\mathbf{x} = (70, 180)$, decide which model is closest. This will require that you consider various distance measures.
3. How would the approach of using distance measures to decide which class an unknown belongs to be affected if the data was on a different scale. For example, what if the heights were logged in meters rather than inches? What techniques should be considered to account for the fact that different features may have completely different scale magnitudes?

Chapter 2

Decision Theory

Decisions and Feature Space

We will now examine the process of classifying observations on the basis of feature vectors. Let $\mathbf{x} \in \mathcal{F}$ represent a feature vector that is produced by an observation. The decision system must then select a particular pattern from the set \mathcal{P} of possible patterns. We seek a system by which the decisions can be made that will minimize the expected loss.

Assume that the feature vector contains the numerical measurements of M features. The features might have particular names such as `height` or `weight`, or just listed symbolically as f_1, f_2, \dots, f_M . In the case of sports figures we used `height` and `weight` as features, and that space had $M = 2$ dimensions. If we had added another feature, such as `shoe size`, we would have had $M = 3$. The number of features, and thus the dimensionality of \mathcal{F} , depends on problem model. A particular feature vector¹ $\mathbf{x} = [x_1, x_2, \dots, x_M]$ will contain numerical values. We permit all real numbers for the feature values, although a particular model may restrict the range. For example, a feature such as height or weight must have a non-negative value.

In all of our models the number of patterns will be finite. We will represent pattern space by $\mathcal{P} = \{\omega_1, \omega_2, \dots, \omega_c\}$ where ω_i is the name of pattern i and c is the number of different classes. The pattern names are simply symbolic and serve to identify the different patterns: `football lineman` and `jockey`

¹The question arises: should vectors be rows or columns of values? It doesn't matter until we do matrix calculations or build computer data structures. When we do that we will make the choice that is most convenient. For now, a vector is just a list of values.

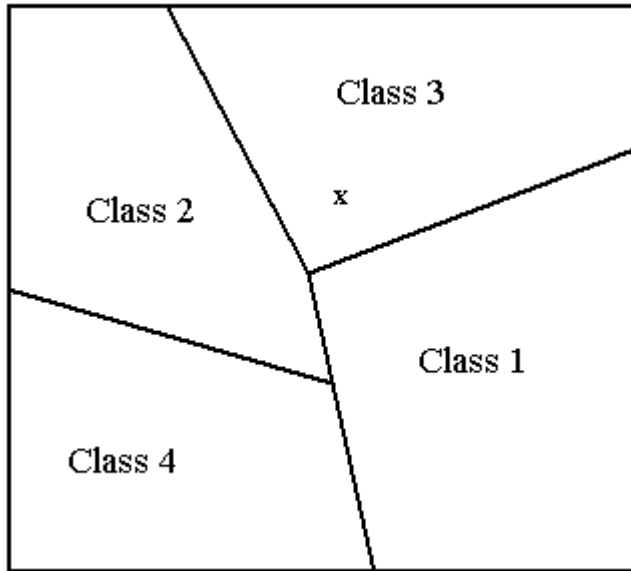


Figure 2.1: Example of a feature space that has been divided into four class regions. The point \mathbf{x} would be classified in Class 3.

are equally good pattern symbols. We will use the mathematical symbols in the abstract model and names of things in particular situations when it suits our fancy.

The function $D : \mathcal{F} \rightarrow \mathcal{P}$ matches each feature vector to a pattern. Let \mathcal{F}_k be the subset of \mathcal{F} which is associated with pattern ω_k . That is, $\mathbf{x} \in \mathcal{F}_k \Rightarrow D(\mathbf{x}) = \omega_k$. Then the subset \mathcal{F}_k is equivalent to pattern ω_k , and it is not unusual to see the subsets of feature space labeled with the associated pattern names. An example of a feature space that has been partitioned into four class regions is shown in Figure 2.1. The unknown feature pattern \mathbf{x} falls into region \mathcal{F}_3 of feature space and would be classified as class $\omega_3 = \text{Class 3}$.

2.1 The Decision Function

Suppose that we have chosen a set of features. Each observation produces a particular set of measurements of these features, and can be expressed as a vector, \mathbf{x} . we decide which class caused the observation by the partition

of feature space that contains \mathbf{x} . The *designer* must determine the best partitioning of feature space; that is, the one that yields the best performance under some criterion.

Choosing a decision function D is equivalent to partitioning feature space into N disjoint sets. In mathematical terms, we require

$$\bigcup_{k=1}^N \mathcal{F}_k = \mathcal{F} \quad (2.1)$$

$$\mathcal{F}_i \cap \mathcal{F}_j = \phi \quad (2.2)$$

If the components of the feature vectors have continuous values then there are an infinite number of possible partitions of a feature space. Even when the feature vectors components have discrete values the number of partitions can be extremely large (see problem 1). The task of the designer of a pattern recognition algorithm is to find the best partition within the large number of possible partitions.

At this point it will be convenient to have notation that will distinguish between the pattern that actually occurred in the real world to cause an observation \mathbf{x} and the pattern that our system decided was the cause of \mathbf{x} . We will use α to represent the pattern that is actually present and ω to represent the one that is detected. (From α to ω ; α is real and ω is the “guess.”)

The process of searching for the best decision function, which is equivalent to the best partition of feature space, can be put into a fairly general analytical form by making use of the ideas of probability. When we come to a particular design problem we will have to translate the general ideas into terms that fit the specifics of that case. To be clear, we will state the general ideas in the form of assumptions.

Assumption 1: A function $P(\mathbf{x}|\alpha_i)$ can be defined that measures the probability of observing feature vector \mathbf{x} given that pattern α_i is the real-world cause.

This assumption says that it is possible to define the probability of a particular (**height, weight**) combination given that the athlete is a jockey.

Assumption 2: The a priori probabilities for the pattern classes in the real world are known.

That is, there is a measure such that

$$P(\alpha_i) \geq 0 \quad (2.3)$$

$$\sum_{i=1}^c P(\alpha_i) = 1 \quad (2.4)$$

Given these two assumptions, we can compute all of the other probabilities of interest. For example, the probability that pattern α_i occurs in the real world and causes the feature vector \mathbf{x} to be observed is simply the joint probability

$$P(\alpha_i, \mathbf{x}) = P(\alpha_i)P(\mathbf{x}|\alpha_i) \quad (2.5)$$

The probability that α_i occurs in the real world and produces a feature vector in the region \mathcal{F}_j is found by summing $P(\alpha_i, \mathbf{x})$ over \mathcal{F}_j . This is the same as the probability of deciding on pattern ω_j when the real event was α_i .

$$P(\alpha_i, \omega_j) = \int_{\mathcal{F}_j} P(\alpha_i, \mathbf{x}) d\mathbf{x} = \int_{\mathcal{F}_j} P(\alpha_i)P(\mathbf{x}|\alpha_i) d\mathbf{x} \quad (2.6)$$

If \mathbf{x} is discrete then the integrals are replaced by appropriate sums.

The set of probabilities $P(\alpha_i, \omega_j)$ can be displayed as an $N \times N$ array of numbers called the *confusion matrix*. The probability of a correct decision is the sum of the diagonal terms and the probability of error is the sum of the off-diagonal terms.

Assumption 3: A cost function $\lambda(\omega_j|\alpha_i)$ can be defined for every pair of patterns. This is the cost of making decision ω_j when the real event was α_i .

If these three assumptions are true, then we have the analytical tools that are needed to design and evaluate decision rules. Let D be a particular partition of feature space. That enables us to compute the probabilities $P(\alpha_i, \omega_j)$ of the confusion matrix. The expected cost associated with D is then

$$L(D) = \sum_{i=1}^c \sum_{j=1}^c P(\alpha_i, \omega_j) \lambda(\omega_j|\alpha_i) \quad (2.7)$$

This cost is changed by changing the partition of feature space that is expressed by the decision rule.

Let D^* be the decision rule that minimizes (2.7). We call D^* the *Bayes decision rule* because it involves Bayesian probabilities. It is the best decision rule under the above assumptions. Let us look now at methods to find D^* .

2.1.1 Bayes Rule for Minimum Risk

Bayes rule enables us to compute the probability of observing any particular feature vector. It can be written as

$$P(\mathbf{x}) = \sum_{i=1}^c P(\alpha_i)P(\mathbf{x}|\alpha_i) \quad (2.8)$$

By Assumptions 1 and 2, we know all the ingredients for this calculation. We can now compute the reverse conditional probabilities

$$P(\alpha_i|\mathbf{x}) = \frac{P(\alpha_i)P(\mathbf{x}|\alpha_i)}{P(\mathbf{x})} \quad (2.9)$$

The joint probability $P(\alpha_i, \omega_j)$ can be expressed as

$$P(\alpha_i, \omega_j) = \int_{\mathcal{F}_j} P(\mathbf{x})P(\alpha_i|\mathbf{x})d\mathbf{x} \quad (2.10)$$

We can bring the feature space into the cost expression by substituting (2.10) into (2.7). This yields

$$L(D) = \sum_{i=1}^c \sum_{j=1}^c \lambda(\omega_j|\alpha_i) \int_{\mathcal{F}_j} P(\mathbf{x})P(\alpha_i|\mathbf{x})d\mathbf{x} \quad (2.11)$$

This can be rearranged as

$$L(D) = \sum_{j=1}^c \int_{\mathcal{F}_j} P(\mathbf{x}) \sum_{i=1}^c \lambda(\omega_j|\alpha_i)P(\alpha_i|\mathbf{x})d\mathbf{x} \quad (2.12)$$

The integral can be brought outside if we make use of a partition function. Let

$$Z_j(\mathbf{x}) = \begin{cases} .1 & \text{if } \mathbf{x} \in \mathcal{F}_j \\ 0 & \text{elsewhere} \end{cases} \quad (2.13)$$

Then the loss can be expressed as

$$L(D) = \int_{\mathcal{F}} P(\mathbf{x}) \sum_{j=1}^c \left(\sum_{i=1}^c \lambda(\omega_j|\alpha_i)P(\alpha_i|\mathbf{x}) \right) Z_j(\mathbf{x})d\mathbf{x} \quad (2.14)$$

The inner term is called the *conditional risk* of making decision ω_j given the observation \mathbf{x} .

$$\lambda_j(\mathbf{x}) = \sum_{i=1}^c \lambda(\omega_j|\alpha_i)P(\alpha_i|\mathbf{x}) \quad (2.15)$$

Now, for any particular \mathbf{x} only one partition function is nonzero. If we then let

$$\lambda(\mathbf{x}) = \sum_{j=1}^c \lambda_j(\mathbf{x}) Z_j(\mathbf{x}) \quad (2.16)$$

we have $\lambda(\mathbf{x}) = \lambda_j(\mathbf{x})$ in the region defined by $Z_j(\mathbf{x})$. The overall loss can be minimized by choosing $Z_j(\mathbf{x})$ to contain those values of \mathbf{x} where $\lambda_j(\mathbf{x}) \leq \lambda_k(\mathbf{x})$ for $k = 1, 2, \dots, c$. Let us define the decision function that is defined by this rule as the *Bayes decision function*, D^* . Let $\lambda^*(\mathbf{x})$ be the loss that is associated with the decision for each \mathbf{x} . Then the minimum expected loss, which is called the *Bayes risk*, is

$$L(D^*) = \int_{\mathcal{F}} P(\mathbf{x}) \lambda^*(\mathbf{x}) d\mathbf{x} \quad (2.17)$$

2.1.2 Computational Considerations

The conditional risk function can be rearranged to reduce the amount of computation and avoid potential numerical problems. The term $P(\alpha_i|\mathbf{x})$ in (2.15) can be computed by using (2.8) and (2.9). However, this can lead to computational problems when $P(\mathbf{x})$ is very small or zero. However, we can substitute (2.9) into (2.15) and rearrange the equation.

$$\lambda_j(\mathbf{x}) = \sum_{i=1}^c \lambda(\omega_j|\alpha_i) \frac{P(\alpha_i)P(\mathbf{x}|\alpha_i)}{P(\mathbf{x})} \quad (2.18)$$

Since $P(\mathbf{x})$ is common to all of the terms it can be factored out. After rearrangement, we can define a new risk function

$$L_j(\mathbf{x}) = \lambda_j(\mathbf{x})P(\mathbf{x}) = \sum_{i=1}^c \lambda(\omega_j|\alpha_i)P(\alpha_i)P(\mathbf{x}|\alpha_i) \quad (2.19)$$

Choosing j so that $L_j(\mathbf{x})$ is minimized for any \mathbf{x} is the same as choosing j to minimize $\lambda_j(\mathbf{x})$. However, the sum on the right hand side is much easier to calculate since it is based strictly on the three quantities that are given in Assumptions 1-3. We now let

$$L(\mathbf{x}) = \sum_{j=1}^c L_j(\mathbf{x}) Z_j(\mathbf{x}) \quad (2.20)$$

x	0	1	2	3	4	5	6
$P(x H)$	0.300	0.250	0.150	0.100	0.080	0.070	0.050
$P(x T)$	0.050	0.070	0.080	0.100	0.150	0.250	0.300
$P(x)$	0.175	0.160	0.115	0.100	0.115	0.160	0.175
$P(H x)$	0.857	0.781	0.652	0.500	0.348	0.219	0.143
$P(T x)$	0.143	0.219	0.348	0.500	0.652	0.781	0.857
$\lambda_1(x)$	0.143	0.219	0.348	0.500	0.652	0.781	0.857
$\lambda_2(x)$	0.857	0.781	0.652	0.500	0.348	0.219	0.143
$\lambda^*(x)$	0.143	0.219	0.348	0.500	0.348	0.219	0.143
$L_1(x)$	0.025	0.035	0.04	0.05	0.075	0.125	0.15
$L_2(x)$	0.15	0.125	0.075	0.05	0.04	0.035	0.025
$L^*(x)$	0.025	0.035	0.04	0.05	0.04	0.035	0.025
$Z_1(x)$	1	1	1	1	0	0	0
$Z_2(x)$	0	0	0	0	1	1	1
Decision	H	H	H	H	T	T	T

Table 2.1: Analysis table for binary decision example.

where $Z_j(\mathbf{x})$ is chosen in the same way. When that is done, we have the optimum partition, and call the risk function $L^*(\mathbf{x})$. The average Bayes risk can then be written

$$L(D^*) = \int_{\mathcal{F}} L^*(\mathbf{x}) d\mathbf{x} \quad (2.21)$$

Example: Binary decision problem

An experiment has two outcomes, say H and T . A trial of the experiment causes a value x to be printed. The values of x and the conditional probabilities $P(x|H)$ and $P(x|T)$ are shown in the table below. On the basis of an observation of x you have to decide whether H or T has occurred.

It is known that $P(H) = P(T) = 0.5$, and the cost of making an incorrect decision is the same for each kind of mistake. We will model the costs as $\lambda(H|T) = \lambda(T|H) = 1$, $\lambda(H|H) = \lambda(T|T) = 0$.

The probability of observing each value of x can be computed by using (2.8), and then the conditional probabilities $P(H|x)$ and $P(T|x)$ can be computed using (2.9). These are the next three rows of the table.

The losses can be calculated by using (2.15), which in this case is expressed as

$$\begin{aligned}\lambda_1(x) &= \lambda(H|H)P(H|x) + \lambda(H|T)P(T|x) = P(T|x) \\ \lambda_2(x) &= \lambda(T|H)P(H|x) + \lambda(T|T)P(T|x) = P(H|x)\end{aligned}$$

These values are shown for each x in the next two rows of the array. The minimum loss for each x is the smaller of the two loss values. This is the value $\lambda^*(x)$ shown for each x . Rather than computing $\lambda_1(x)$ and $\lambda_2(x)$ we can compute $L_1(x)$ and $L_2(x)$ by

$$\begin{aligned}L_1(x) &= P(T)P(x|T) \\ L_2(x) &= P(H)P(x|H)\end{aligned}$$

These are shown in the next two rows of the table. Note that $L_1(x)$ and $L_2(x)$ have the same order relationship as $\lambda_1(x)$ and $\lambda_2(x)$.

Finally, the partitions can be chosen. These are indicated by the functions $D_1(x)$ and $D_2(x)$, which take on values 1 or 0 to indicate whether or not that value of x is included in that partition. Finally, the decision associated with each x is shown in the bottom row.

The probability of error, which is equal to the expected loss, can be calculated by using (2.17). For this problem in which there are a finite number of x values the computation is expressed as a sum

$$L(D^*) = \sum_x P(x)\lambda^*(x) = \sum_x L^*(x) = 0.25$$

It can be verified by direct computation that no other partition will yield a lower cost for the set of assumptions stated at the beginning of this example.

Patterns in Additive Noise

The model we will examine here is probably the most common in pattern recognition, not to mention signal detection in radar and communications. It is a basic model with that provides an elementary analysis for many different applications.

We assume that the observed feature vector is the sum of a model feature vector and a noise perturbation. Use of this model makes it possible to develop a tractable mathematical analysis that is a reasonable approximation

to reality in many different situations. The model becomes particularly simple to use when it is assumed that the added noise has a normal distribution. This model is the foundation beneath matched filters and other “optimum” decision structures.

The general approach is to assume that there is a finite set of events that can be observed through a sensing system that produces some kind of signal. The observed signal is the sum of a predictable signal and noise. The signals can be converted to vectors by a variety of means, one of which is sampling.

Let $\mathcal{P} = \{\alpha_1, \dots, \alpha_c\}$ be the set of real object classes. Let us assume that occurrence of α_j causes a signal $s_j(t)$ to be observed, and let \mathbf{s}_j be the corresponding signal vector. For example, \mathbf{s}_j may then be the set of samples $\mathbf{s}_j = \{s_j(t_1), s_j(t_2), \dots, s_j(t_M)\}$. The value of M is determined by the time-bandwidth product of the signal.

Corresponding to $\mathcal{P} = \{\alpha_1, \dots, \alpha_c\}$ there is then the signal vector space $\mathcal{S} = \{\mathbf{s}_1, \dots, \mathbf{s}_c\}$. However they may be obtained, the vectors \mathbf{s}_j are produced by the objects α_j , $j = 1 : c$. The fact that we don't have to be specific about the physical origins of \mathbf{s} means that we can apply the technique to many different problems.

Noise in the environment and in the observation system will make it impossible to clearly observe the signal. The effect of the noise may be to add a random value to each component of the signal. Rather than observing \mathbf{s}_j , we end up observing $\mathbf{x} = \mathbf{s}_j + \mathbf{z}$ where \mathbf{z} is a random vector perturbation due to the noise. Even though there are a finite number of signal vectors, there are a continuum of values for \mathbf{x} . The feature space \mathcal{F} is the space of possible observations represented by \mathbf{x} .

The noise is assumed to have some probability density function $q(\mathbf{z})$. It is also assumed that the noise is independent of the signal. The probability $P(\mathbf{x}|\alpha_j)$ can then be written in the simple form

$$P(\mathbf{x}|\alpha_j) = q(\mathbf{x} - \mathbf{s}_j) \quad \text{for } 1 \leq j \leq c \quad (2.22)$$

Given an observed feature vector \mathbf{x} , the probability $P(\mathbf{x}|\alpha_j)$ depends only on the vector difference $\mathbf{x} - \mathbf{s}_j$. Since \mathbf{x} is known by observation and all of the signal vectors are known, each of the probabilities can be calculated.

The assumption that the noise is additive and that the probability distribution of the noise is known is sufficient to fulfill the requirements of Assumption 1. If we also assume that the requirements of Assumption 2 are satisfied because the *a priori* probabilities $P(\alpha_i)$ are known and that the loss

$\lambda(\omega_j|\alpha_j)$ is known then we have enough information to compute the expected loss.

$$L_j(\mathbf{x}) = \sum_j \lambda(\omega_j|\alpha_i)P(\alpha_j)P(\mathbf{x}|\alpha_j) = \sum_j \lambda(\omega_j|\alpha_i)P(\alpha_j)q(\mathbf{x} - \mathbf{s}_j) \quad (2.23)$$

The point \mathbf{x} should be associated with the partition \mathcal{F}_m for which $L_m(\mathbf{x})$ is minimum.

Binary Decision Case

With the binary decision case it is possible to develop a likelihood ratio test.

$$L_1(\mathbf{x}) = \lambda(\omega_1|\alpha_1)P(\alpha_1)q(\mathbf{x} - \mathbf{s}_1) + \lambda(\omega_1|\alpha_2)P(\alpha_2)q(\mathbf{x} - \mathbf{s}_2) \quad (2.24)$$

$$L_2(\mathbf{x}) = \lambda(\omega_2|\alpha_1)P(\alpha_1)q(\mathbf{x} - \mathbf{s}_1) + \lambda(\omega_2|\alpha_2)P(\alpha_2)q(\mathbf{x} - \mathbf{s}_2) \quad (2.25)$$

The point \mathbf{x} is assigned to region \mathcal{F}_1 if $L_1(\mathbf{x}) \leq L_2(\mathbf{x})$, otherwise it is assigned to region \mathcal{F}_2 . The point belongs to \mathcal{F}_1 if

$$L_1(\mathbf{x}) \leq L_2(\mathbf{x}) \quad (2.26)$$

This equation can be arranged so that all of the terms that involve \mathbf{x} are on one side of the equation. We can say that \mathbf{x} belongs to \mathcal{F}_1 if

$$\frac{q(\mathbf{x} - \mathbf{s}_2)}{q(\mathbf{x} - \mathbf{s}_1)} \leq \frac{(\lambda(\omega_2|\alpha_1) - \lambda(\omega_1|\alpha_1))P(\alpha_1)}{(\lambda(\omega_1|\alpha_2) - \lambda(\omega_2|\alpha_2))P(\alpha_2)} \quad (2.27)$$

The term on the left is called the *likelihood ratio* and the term on the right is called the *threshold*. We define the terms

$$\Lambda(\mathbf{x}) = \frac{q(\mathbf{x} - \mathbf{s}_2)}{q(\mathbf{x} - \mathbf{s}_1)} \quad (2.28)$$

$$\eta = \frac{(\lambda(\omega_2|\alpha_1) - \lambda(\omega_1|\alpha_1))P(\alpha_1)}{(\lambda(\omega_1|\alpha_2) - \lambda(\omega_2|\alpha_2))P(\alpha_2)} \quad (2.29)$$

$$\begin{aligned} \text{If } \Lambda(\mathbf{x}) \leq \eta & \text{ then } \mathbf{x} \in \mathcal{F}_1 \\ \text{If } \Lambda(\mathbf{x}) > \eta & \text{ then } \mathbf{x} \in \mathcal{F}_2 \end{aligned} \quad (2.30)$$

Monotonic Noise Probability

The loss functions and the likelihood ratio depend only on the vector differences, $\mathbf{x} - \mathbf{s}_j$. This leads to a great simplification of the analysis particularly when $q(\mathbf{z})$ is a function only of a measure of a length of \mathbf{z} .

In some cases the length is the ordinary Euclidean length and in others it is a more exotic formula. However, a simplification is obtained in all cases, even the exotic cases, when $q(\mathbf{z})$ depends only on some measure of the length of \mathbf{z} . The reason is that the probability $q(\mathbf{x} - \mathbf{s}_j)$ will be the same at all values of \mathbf{x} that are the same distance from \mathbf{s}_j .

A further simplification occurs when the probability drops off monotonically with length of \mathbf{z} . This is the case for the normal distribution, which is the most commonly used distribution in system modeling, as well as other important probability models. When this is true, $q(\mathbf{x} - \mathbf{s}_j)$ is greatest at $\mathbf{x} = \mathbf{s}_j$ and decreases monotonically as \mathbf{x} moves away from \mathbf{s}_j in any direction.

The distance measure used in these calculations need not be the standard Euclidean distance. It can be any measure that satisfies the postulates for a distance metric. A function d is a distance measure if

$$d(\mathbf{x}_1, \mathbf{x}_2) = 0 \quad \text{if } \mathbf{x}_1 = \mathbf{x}_2 \quad (2.31)$$

$$d(\mathbf{x}_1, \mathbf{x}_2) > 0 \quad \text{if } \mathbf{x}_1 \neq \mathbf{x}_2 \quad (2.32)$$

$$d(\mathbf{x}_1, \mathbf{x}_3) \leq d(\mathbf{x}_1, \mathbf{x}_2) + d(\mathbf{x}_2, \mathbf{x}_3) \quad \text{for all } \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \quad (2.33)$$

The length of a vector is calculated by measuring its distance from the origin, so it is common to write $d(\mathbf{x})$ for $d(\mathbf{x}, \mathbf{0})$.

Some common distance measures are

$$\text{Euclidean:} \quad d(\mathbf{z}) = \sqrt{z_1^2 + z_2^2 + \cdots + z_M^2} \quad (2.34)$$

$$\text{Mahalanobis:} \quad d(\mathbf{z}) = \sqrt{\mathbf{z}\Sigma^{-1}\mathbf{z}'} \quad (2.35)$$

$$\text{Manhattan:} \quad d(\mathbf{z}) = |z_1| + |z_2| + \cdots + |z_M| \quad (2.36)$$

The Euclidean distance measures the length of the straight line from \mathbf{z} to the origin. Contours of equal distance from the origin are spheres in M dimensions. The Mahalanobis distance does essentially the same thing, but weights each of the components differently. This makes contours of equal distance from the origin into ellipsoids. The matrix Σ is the $M \times M$ covariance matrix

$$\Sigma = E[\mathbf{z}'\mathbf{z}] \quad (2.37)$$

where \mathbf{z} is a row vector. We will use the Mahalanobis distance extensively in the next several lectures. For now it is sufficient to realize that it exists. The Manhattan distance measures the length of a walk along city streets from the origin to the point \mathbf{z} . Contours of equal distance from the origin look like diagonals of city blocks (or hyperplanes in higher dimensions).

A function of the form

$$q(\mathbf{z}) = ae^{-bd^m(\mathbf{z})} \quad (2.38)$$

with properly chosen values for the constants a , b and m is a probability function that decreases monotonically with distance of \mathbf{z} from the origin.

The log likelihood ratio is easily calculated in this case. It is

$$\ln \Lambda(\mathbf{x}) = b(d^m(\mathbf{x} - \mathbf{s}_1) - d^m(\mathbf{x} - \mathbf{s}_2)) \quad (2.39)$$

The decision rule is that $\mathbf{x} \in \mathcal{F}_1$ if

$$d^m(\mathbf{x} - \mathbf{s}_1) - d^m(\mathbf{x} - \mathbf{s}_2) \leq \frac{\ln \eta}{b} \quad (2.40)$$

otherwise it belongs to \mathcal{F}_2 . In the case of equal losses and equal *a priori* probabilities, $\ln \eta = 0$ and the decision associates \mathbf{x} with the closest signal point in signal space. This is seen by moving $d^m(\mathbf{x} - \mathbf{s}_2)$ to the opposite side and taking the principal root.

Equation (2.40) is a powerful but simple rule that applies whenever (1) the noise is additive; (2) the probability is an exponentially decreasing function of distance; and (3) the choice is between two patterns. If these conditions are satisfied, then we don't have to do much mathematical analysis to construct the optimum decision rule. We will find below that the rule can be extended to situations with more than two patterns when the losses and *a priori* probabilities are equal.

Partitioning \mathcal{F} with multiple patterns

A general rule can be developed for the case of multiple patterns when the loss function is² $\lambda(\omega_j|\alpha_i) = \delta(i, j)$ for all $1 \leq i, j \leq c$ and the patterns have

² $\delta(i, j) = 1$ if $i = j$ and $\delta(i, j) = 0$ if $i \neq j$.

equal *a priori* probabilities, $P(\alpha_j) = 1/c$, $1 \leq j \leq c$. With these values, (2.23) becomes

$$L_m(\mathbf{x}) = \frac{1}{c} \sum_{j \neq m} q(\mathbf{x} - \mathbf{s}_j) \quad (2.41)$$

This sum is minimized by setting choosing m such that $q(\mathbf{x} - \mathbf{s}_m)$ is maximized. This causes the largest possible term to be left out of the sum, and minimizes the result. Let $m = i$ be this index for the given \mathbf{x} . Then $L_i(\mathbf{x}) \leq L_j(\mathbf{x})$ for $1 \leq j \leq c$. Then it must also be true that $q(\mathbf{x} - \mathbf{s}_i) \geq q(\mathbf{x} - \mathbf{s}_j)$ for $1 \leq j \leq c$.

If $q(\mathbf{z})$ is a monotonically decreasing function of distance then \mathcal{F}_i must contain all the points in \mathcal{F} that are closest to \mathbf{s}_i under the distance measure that is associated with $q(\mathbf{z})$. This is summarized in the following rule:

If patterns are observed as signals in additive noise whose probability decreases monotonically with the length of the noise vector, then the minimum error probability partition of decision space is achieved by associating each \mathbf{x} with the closest signal vector.

Example: Additive White Gaussian Noise

The above situation is particularly easy and useful to illustrate for the case of c signals in white gaussian noise. The observed signal in noise is of the form $x(t) = s_i(t) + z(t)$. The feature vector is constructed by sampling $x(t)$. For simplicity we will assume that all signals have the same finite duration T and that the samples are taken at the points t_1, t_2, \dots, t_M , all of which fall within the signal time interval. White noise has an infinite bandwidth, so that each of the samples has a normal distribution and is statistically independent of all the other samples. We can take the probability density function of each noise sample to be $N(0, \sigma)$.

The feature vector is³

$$\mathbf{x} = \mathbf{s}_i + \mathbf{z}$$

where each vector is the set of M samples of the corresponding waveform.

³We will use column vectors in the equations below.

Because the noise samples are statistically independent, the pdf of $q(\mathbf{z})$ is

$$\begin{aligned} q(\mathbf{z}) &= \prod_{m=1}^M \frac{1}{\sigma\sqrt{2\pi}} e^{-(z_m/\sigma\sqrt{2})^2} \\ &= \frac{1}{(2\pi\sigma^2)^{M/2}} \exp\left(\frac{1}{2\sigma^2} \sum_{m=1}^M z_m^2\right) \\ &= \frac{1}{(2\pi\sigma^2)^{M/2}} e^{-\frac{\mathbf{z}\mathbf{z}'}{2\sigma^2}} \end{aligned} \quad (2.42)$$

The pdf of \mathbf{z} is of the form $q(\mathbf{z}) = ae^{-bd^2(\mathbf{z})}$ where a and b are constants and the squared distance is the vector dot product

$$d^2(\mathbf{z}) = \mathbf{z}\mathbf{z}' = \sum_{m=1}^M z_m^2 \quad (2.43)$$

Therefore, the minimum error probability decision is made by calculating $d^2(\mathbf{x} - \mathbf{s}_i)$, $1 \leq i \leq c$ and selecting the i for which the result is the smallest.

We can write the computation out in terms of \mathbf{x} and \mathbf{s}_i and thereby obtain an even simpler form.

$$\begin{aligned} d^2(\mathbf{x} - \mathbf{s}_i) &= (\mathbf{x} - \mathbf{s}_i)(\mathbf{x} - \mathbf{s}_i)' \\ &= \mathbf{x}'\mathbf{x} - 2\mathbf{s}_i'\mathbf{x} + \mathbf{s}_i'\mathbf{s}_i \end{aligned} \quad (2.44)$$

The term $\mathbf{x}'\mathbf{x}$ is common to all of the distance computations and therefore can be dropped without affecting the decision. The remaining part can be collected into a function

$$g(\mathbf{s}_i, \mathbf{x}) = \mathbf{s}_i\mathbf{x}' - \frac{\mathbf{s}_i\mathbf{s}_i'}{2} \quad (2.45)$$

The decision is made by picking the index i for which $g(\mathbf{s}_i, \mathbf{x})$ is the *largest*. This kind of function is called a *discriminant function*. The last term is proportional to the sum of the squares of the signal samples. It is often called the “energy” $E_i = \mathbf{s}_i'\mathbf{s}_i$. With this definition,

$$g(\mathbf{s}_i, \mathbf{x}) = \mathbf{s}_i\mathbf{x}' - \frac{E_i}{2} \quad (2.46)$$

If all the signals have the same energy then the last term can also be dropped.

The essential calculation is the computation of $\mathbf{s}'_i \mathbf{x}$ for each $1 \leq i \leq c$. The computations are often implemented as matched filters. This is particularly convenient if one uses physical analog filters on the time waveform $x(t)$ rather than a digital filter on \mathbf{x} .

Matched Filter Let us suppose that $s_i(t)$ is zero outside the time interval $[0, T]$. Let $h_i(t) = s_i(t_0 - t)$ where $t_0 \geq T$. It is sometimes the case that one can design an electronic filter in such a way that it has an impulse response equal to $h_i(t)$. The response of such a system to an input $x(t)$ is

$$\begin{aligned} r(t) &= \int_0^t x(\tau) h_i(t - \tau) d\tau \\ &= \int_0^t x(\tau) s_i(\tau + t_0 - t) d\tau \end{aligned} \quad (2.47)$$

At the instant $t = t_0$ one can take a sample of the filter output.

$$\begin{aligned} r(t_0) &= \int_0^{t_0} x(\tau) s_i(\tau) d\tau \\ &= \int_0^T x(\tau) s_i(\tau) d\tau \end{aligned} \quad (2.48)$$

The number $r(t_0)$ is an indication of how well $x(t)$ matches the signal template, $s_i(t)$.

The same computation can be carried out with digital filters. We simply let $h_i(m) = s_i(M + k - m)$ where $k > 0$. The response of a digital filter whose impulse response is $h_i(m)$ to the input sequence $x(m)$ is

$$\begin{aligned} r(n) &= \sum_{m=1}^n x(m) h_i(n + 1 - m) \\ &= \sum_{m=1}^n x(m) s_i(M + k + m - 1 - n) \end{aligned} \quad (2.49)$$

At the instant $n = M + k - 1$,

$$r(M + k - 1) = \sum_{m=1}^M x(m) s_i(m) \quad (2.50)$$

(The upper limit can be set to M because $s_i(m) = 0$ for $M > m$.) The digital filter output at the appropriate sample time is equal to the dot product of the vectors.

The computations in signal space can usually be expressed in either an analog or a digital form. In the days before digital computation it was necessary to rely totally on the analog filter implementations. This made it necessary to constrain the signals to forms that could be readily matched by filters. The availability of digital processing has made this unnecessary, but many of the older forms still persist. One of the reasons is compatibility with older signal space standards.

One of the particularly useful tricks to reduce system complexity was to use a set of orthogonal basis functions to describe the signals. In modern terms, this translates into the use of orthogonal basis vectors.

Basis Vectors The evaluation of the discriminant functions by (2.46)

requires c filtering operations, one for each signal waveform. It is often possible to reduce the number of vector multiplications (or the number of matched filters) by use of basis vectors to represent the signals. Suppose that the dimensionality of feature space is K . Then this space can be spanned by a set of orthonormal vectors $\mathcal{U} = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_K\}$. These vectors have the property

$$\mathbf{u}_k \mathbf{u}'_m = \delta(k, m) \quad (2.51)$$

Any signal vector can be represented by a vector sum

$$\mathbf{s}_i = \sum_{k=1}^K s_{ik} \mathbf{u}_k \quad (2.52)$$

where the coefficients are

$$s_{ik} = \mathbf{u}'_k \mathbf{s}_i \quad (2.53)$$

The dot product $\mathbf{x}' \mathbf{s}'_i$ can then be represented as

$$\mathbf{x}' \mathbf{s}_i = \sum_{k=1}^K s_{ik} \mathbf{x}' \mathbf{u}_k \quad (2.54)$$

The terms $\mathbf{x}' \mathbf{u}'_k$ appear in all the c equations, $1 \leq i \leq c$. Let $x_k = \mathbf{x}' \mathbf{u}'_k$, $1 \leq k \leq K$. If $K < M$ then the number of required vector products is

reduced. The discriminant functions can then be calculated by

$$g_i(\mathbf{s}_i, \mathbf{x}) = \sum_{k=1}^K s_{ik}x_k + \frac{E_i}{2} \quad (2.55)$$

2.2 Problems

1. A particular observation system is capable of producing a finite set of feature vectors. If there are N possible vectors \mathbf{x}_i , $i = 1 : N$ and there are c classes, how many partitions are possible? This is the number of possible decision functions.
2. Compute the decision partition and the expected loss for the data for the decision example in the text, except with the costs $\lambda(H|T) = 2$, $\lambda(T|H) = 1$, $\lambda(H|H) = \lambda(T|T) = 0$.
3. This problem addresses binary events in additive noise. Suppose that there are two events α_1 and α_2 which produce “signals” $s_1 = -1$ and $s_2 = 1$. These signals are observed in additive noise with the probability density function $z = N(0, \sigma)$. Describe the optimum detector for the case $P(\alpha_1) = P(\alpha_2)$ and find an expression for the error probability for the system as a function of σ . You can use a table of the normal distribution function or an error function, such as that provided by the IDL ERRORF function, to plot the error probability as a function of σ .
4. Repeat the above problem for the case $P(\alpha_1) = 0.7$.
5. Modify the analysis in Problem 3 and Problem 4 to enable the errors to have different costs. What would be the expected loss for the optimum decision function if $\lambda(\omega_1|\alpha_2) = 2$ and $\lambda(\omega_2|\alpha_1) = 1$ and a correct decision cost nothing?

Chapter 3

Multivariate Normal Model

3.1 Introduction

In this lecture we will examine the use of the multivariate normal model and its use in pattern classification. A point in feature space will be represented by a row vector \mathbf{x} of M dimensions. $\mathbf{x} = x_1, \dots, x_M$, where x_i is the value of feature number i . Particular points in feature space may be referred to as $\mathbf{x}_1, \mathbf{x}_2$ and so on. We will illustrate some of the concepts by using figures based on $M = 2$, but real problems often use much larger values of M . The mathematics is the same in higher dimensions, but we can't draw the figures.

The multivariate normal distribution has a relatively simple expression that is almost independent of the number of dimensions. This and other properties make it very flexible as a modeling tool. The expression is

$$p_N(\mathbf{x}) = \frac{1}{(2\pi)^{M/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}[(\mathbf{x}-\mu)\Sigma^{-1}(\mathbf{x}-\mu)']} \quad (3.1)$$

We will use the notation $p_N(\mathbf{x}, \mu, \Sigma)$ to refer to the above expression when we want to make the parameters evident but don't want to write out the whole equation. The subscript N means that the distribution is normal.

The parameter μ is a vector that represents the mean value of \mathbf{x} . The value μ_r is the mean value of x_r . The parameter Σ is the covariance matrix. The ij element of Σ is the covariance

$$s_{ij} = E[(x_i - \mu_i)(x_j - \mu_j)] \quad (3.2)$$

Since $s_{ij} = s_{ji}$ and $s_{ii} > 0$, the matrix Σ is positive definite. All of the eigenvalues of Σ are real and positive and all of the eigenvectors are linearly

independent. These properties are especially useful in certain numerical calculations.

The contours on which $p_N(\mathbf{x}) = c$, where c is some constant, are of particular interest. These are the contours of constant probability. If we set the expression in (3.1) equal to a constant, we see that the required condition for a contour is that the exponent equal a constant. This leads to

$$(\mathbf{x} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})' = \ln \left(c(2\pi)^{M/2} \sqrt{|\boldsymbol{\Sigma}|} \right) = c_1 \quad (3.3)$$

This defines an ellipsoid in M -dimensional space.

It can be shown that the term on the left satisfies the requirements to be a distance measure. This distance measure, which is commonly used in statistics, is called the Mahalanobis distance.

Mahalanobis Distance The Mahalanobis distance between two points x and y is

$$d_{\boldsymbol{\Sigma}}(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \mathbf{y})'} \quad (3.4)$$

where $\boldsymbol{\Sigma}$ is a positive definite symmetric matrix.

Bivariate Normal Distribution

The bivariate normal distribution has $M = 2$. The mean is a vector of length $M = 2$ and the covariance is a 2×2 matrix. This is the simplest case and one that has figures that we can draw. We can write out the various vectors and matrices in symbolic form to see their behaviour. We will calculate estimated values for these symbols in examples where we have numerical feature data. Let the mean vector and covariance matrix be represented by

$$\boldsymbol{\mu} = [m_1, m_2] \quad \text{and} \quad \boldsymbol{\Sigma} = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} \quad (3.5)$$

The inverse covariance matrix is

$$\boldsymbol{\Sigma}^{-1} = \frac{\begin{bmatrix} s_{22} & -s_{12} \\ -s_{21} & s_{11} \end{bmatrix}}{s_{11}s_{22} - s_{12}s_{21}} \quad (3.6)$$

When we multiply out the vector products, we find that the probability density function is

$$p_X(\mathbf{x}) = \frac{1}{2\pi\sqrt{s_{11}s_{22} - s_{12}s_{21}}} \cdot \exp\left(-\frac{(x_1 - m_1)^2 s_{22} - (s_{12} + s_{21})(x_1 - m_1)(x_2 - m_2) + s_{11}(x_2 - m_2)^2}{2(s_{11}s_{22} - s_{12}s_{21})}\right)$$

The exponent is a quadratic form in the components of $\mathbf{x} = [x_1, x_2]$. Setting the exponent to a constant will provide contours of constant probability. The shape of these contours will be ellipses. They will be concentric about the mean value of the distribution. This will be illustrated with an example.

Numerical Example Consider the $M = 2$ case with mean and covariance given by

$$\mu = [1, 1] \quad \text{and} \quad \Sigma = \begin{bmatrix} 1 & -0.7 \\ -0.7 & 1 \end{bmatrix} \quad (3.7)$$

We find that $|\Sigma| = 0.51$ and

$$\Sigma^{-1} = \frac{1}{0.51} \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix} = \begin{bmatrix} 1.9608 & 1.3725 \\ 1.3725 & 1.9608 \end{bmatrix} \quad (3.8)$$

A plot of the probability function and contours of constant probability is shown in Figure 3.1

Here the contours of constant probability are ellipses that are centered on the mean feature values. Each of the contours has an equation of the form

$$1.9608(x_1 - 1)^2 + 2.755(x_1 - 1)(x_2 - 1) + 1.9608(x_2 - 1)^2 = c_1 \quad (3.9)$$

where c_1 is an appropriate constant. This is the form of an equation of an ellipse. Note that the ellipses are concentric about the mean vector $\mu = [1, 1]$ and are tilted because of the nonzero terms off of the diagonal in the covariance matrix. The direction of the tilt of the ellipse axis is determined by the sign of the off-diagonal terms.

3.1.1 Modeling Experimental Data

The normal distribution may be used to model experimental data by computing the mean and covariance functions for that data. Of course, the model

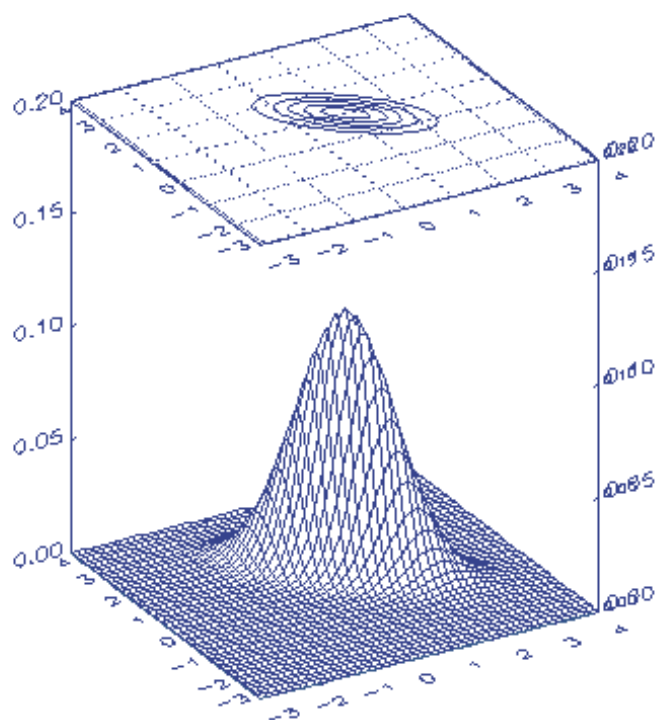


Figure 3.1: A bivariate normal distribution centered on $(1,1)$ with negatively correlated features.

will only give reasonable results when the data is clustered in a normal fashion.

When the data is distributed in three or fewer dimensions it is possible to display the distribution of feature points and get an idea about the quality of the cluster for each class.

3.1.2 Example: Three Classes with Two Features

Some experimental data involving $M = 2$ features are displayed in Figure 3.2. Note that this data falls into three relatively distinct clusters. We would therefore expect some success in modeling each cluster with a normal distribution. The contour lines of a normal distribution with the same mean and covariance are overlaid on the data clusters.

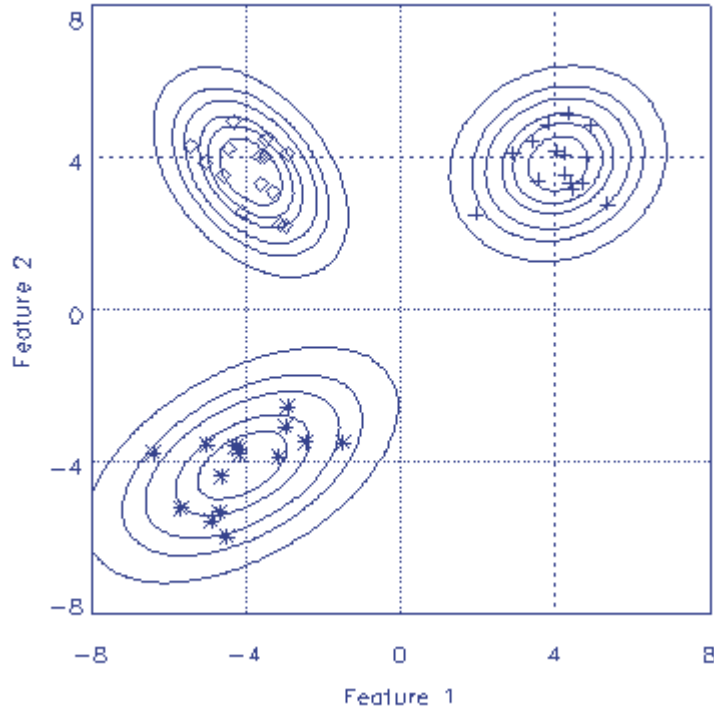


Figure 3.2: A scatter plot of feature data with contours corresponding to normal distributions fit to each cluster.

The data used in this example is tabulated in Table 3.1. There are fifteen examples of each type of object, and each object has two features.

The mean and covariance for each class can be readily calculated. The method is shown in the next section.

$$\mu_1 = [4.074, 3.855] \quad \Sigma_1 = \begin{bmatrix} 0.727 & 0.087 \\ 0.087 & 0.609 \end{bmatrix} \quad \Sigma_1^{-1} = \begin{bmatrix} 1.4 & -0.2 \\ -0.2 & 1.67 \end{bmatrix} \quad (3.10)$$

Example:

$$\mu_2 = [-4.087, -4.07] \quad \Sigma_2 = \begin{bmatrix} 1.65 & 0.6347 \\ 0.634 & 0.974 \end{bmatrix} \quad \Sigma_2^{-1} = \begin{bmatrix} 0.806 & -0.525 \\ -0.525 & 1.368 \end{bmatrix} \quad (3.11)$$

Class α_1		Class α_2		Class α_3	
Feature 1	Feature 2	Feature 1	Feature 2	Feature 1	Feature 2
3.41	4.44	-4.26	-3.56	-4.31	4.97
3.81	4.89	-2.92	-2.54	-4.58	3.53
4.40	3.35	-2.96	-3.07	-3.61	3.33
5.34	2.77	-4.89	-5.54	-3.64	4.04
4.24	4.09	-2.45	-3.47	-2.90	4.13
4.25	3.56	-4.15	-3.77	-5.04	3.89
2.92	4.12	-5.02	-3.54	-3.50	4.05
4.92	4.85	-1.49	-3.49	-2.97	2.22
3.56	3.41	-4.68	-5.31	-3.47	4.48
4.44	3.18	-4.49	-5.97	-3.13	2.24
4.03	4.17	-4.16	-3.64	-4.11	2.56
4.33	5.14	-6.39	-3.76	-5.42	4.34
4.80	4.00	-3.14	-3.85	-4.45	4.24
1.96	2.52	-4.63	-4.35	-3.31	3.13
4.70	3.33	-5.68	-5.19	-3.61	3.32

Table 3.1: Table data for three classes of objects described by two features.

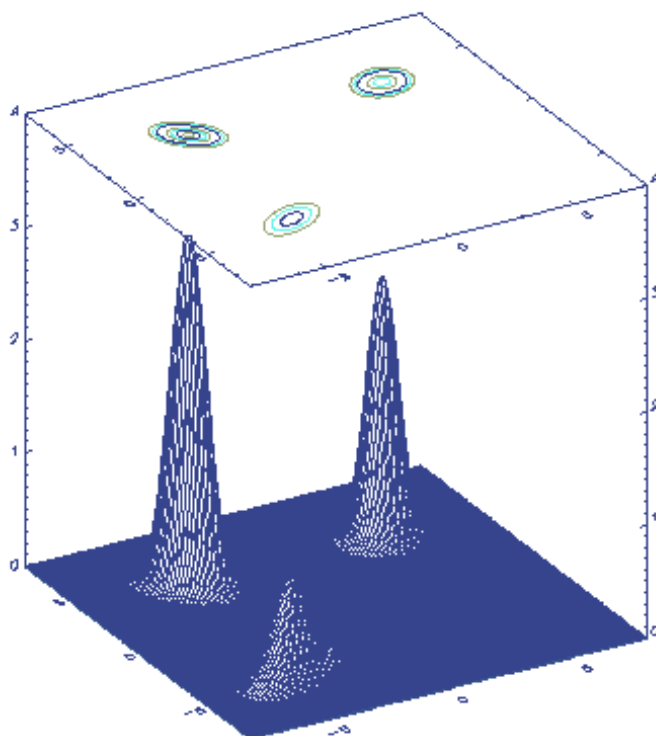


Figure 3.3: An illustration of the normal distribution models for three data clusters showing the surfaces and contours of the models.

$$\mu_3 = [-3.87, 3.63] \quad \Sigma_3 = \begin{bmatrix} 0.569 & -0.263 \\ -0.263 & 0.683 \end{bmatrix} \quad \Sigma_3^{-1} = \begin{bmatrix} 2.14 & 0.823 \\ 0.823 & 1.78 \end{bmatrix} \quad (3.12)$$

Note that in Σ_1 and Σ_2 the off-diagonal terms are positive, showing a positive correlation between the feature values, while Σ_3 has negative off-diagonal terms, showing negative correlation between its features. The contours in Figure 3.2 for class α_1 are nearly circular, showing that the cross correlation is very small (0.087). The contours for α_2 are quite elongated and have a positive slope in the coordinate plane, showing a fairly strong positive correlation (0.635). The contours for α_3 are less elongated and show a neg-

ative slope, corresponding to the negative correlation (-0.26). (You can find the cluster for each class by looking at the coordinates of the mean feature vector.)

3.1.3 Calculation of Model Parameters

The advantage of using the normal distribution is that each class can be modeled by a simple parametric function. The mean vector and covariance matrix for each class are easily calculated from labeled data.

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N_k} \mathbf{x}_n \quad (3.13)$$

where N_k is the number of examples of class k feature vectors. This calculation has to be done separately for each class since each has a different mean vector. We can also calculate the terms of the covariance matrix using the data for each class.

$$s_{ij} = \frac{1}{N_k - 1} \sum_{n=1}^{N_k} (x_{ni} - \mu_i)(x_{nj} - \mu_j)$$

where x_{ni} is component i of the feature vector \mathbf{x}_n in class k .

In IDL it is convenient to arrange the data for a particular class as a matrix X in which each row represents a feature vector for a different data point. There are as many rows as feature vectors for a given class and as many columns as there are features. Then the mean is given by the function `mean(X)` and the covariance is given by `cov(X)`. The matrix X would correspond to the pair of columns in Table 3.1 associated with the class of interest.

3.1.4 Minimum Error Decisions

The expected loss in associating a given observation \mathbf{x} with class ω_j is given by

$$L_j(\mathbf{x}) = \sum_{i=1}^c \lambda(\omega_j|\alpha_i) P(\alpha_i) p(\mathbf{x}|\alpha_i) \quad (3.14)$$

If $\lambda(\omega_i|\alpha_i) = 0$ and $\lambda(\omega_j|\alpha_i) = 1$ for all $j \neq i$, the above reduces to

$$L_j(\mathbf{x}) = 1 - P(\mathbf{x}|\alpha_j)$$

Hence, the probability of error is minimized by associating \mathbf{x} with the class that maximizes $P(\mathbf{x}|\alpha_j)$. The probability $P(\mathbf{x}|\alpha_j)$ is modeled as a normal distribution with mean μ_j and covariance Σ_j . That is, we assume that we know the distribution once we know which class was chosen. The expression for the probability is given by specializing Equation 3.1 for each class.

$$P(\mathbf{x}|\alpha_j) = \frac{1}{(2\pi)^{M/2}|\Sigma_j|^{1/2}} e^{-\frac{1}{2}[(\mathbf{x}-\mu_j)\Sigma_j^{-1}(\mathbf{x}-\mu_j)']} \quad (3.15)$$

When we are using the normal distribution as a model, it is more convenient to maximize $\ln(P(\mathbf{x}|\alpha_j))$. If we eliminate terms that are the same for all j , we find that we should maximize the function

$$g_j(\mathbf{x}) = -\ln|\Sigma_j| - (\mathbf{x} - \mu_j)\Sigma_j^{-1}(\mathbf{x} - \mu_j)' \quad (3.16)$$

This function is a *discriminant function*. Choosing the value of j that maximizes $g_j(\mathbf{x})$ is equivalent to choosing the j that minimizes the loss. In the above equation μ_j and Σ_j are the mean and covariance for class j . The discriminant functions for the previous example are shown in Figure 3.4. The decision boundaries correspond to the contours where the decision functions meet. These are shown on a contour plot in Figure 3.5. It is easy to see why the boundaries are curved contours in this example.

The term on the right is the square of the Mahalanobis distance measure from the point \mathbf{x} to the point μ_j . If we write

$$d_j^2(\mathbf{x}, \mu_j) = (\mathbf{x} - \mu_j)\Sigma_j^{-1}(\mathbf{x} - \mu_j)'$$

we have a distance measure in Σ_j units. The discriminant function can be written as

$$g_j(\mathbf{x}) = c_j - d_{\Sigma_j}^2(\mathbf{x}, \mu_j) \quad (3.17)$$

where

$$c_j = -\ln|\Sigma_j|$$

is a constant, with a different value for each j . For a given point \mathbf{x} , we need to simply calculate $g_j(\mathbf{x})$ for $j = 1, \dots, k$ and pick the one that is greatest. This will give us the decision that minimizes the probability of error. The computation of $g_j(\mathbf{x})$, in turn, involves calculation of $d_{\Sigma_j}^2(\mathbf{x}, \mu_j)$. Because of the minus sign, moving the point \mathbf{x} closer to μ_j increases $g_j(\mathbf{x})$. Note, however, that because the constants c_j are different for each class we don't always associate \mathbf{x} with the closest μ_j . The constants will not vary with class when all the Σ_j are equal. In that case, some even nicer simplifications occur, as discussed below.

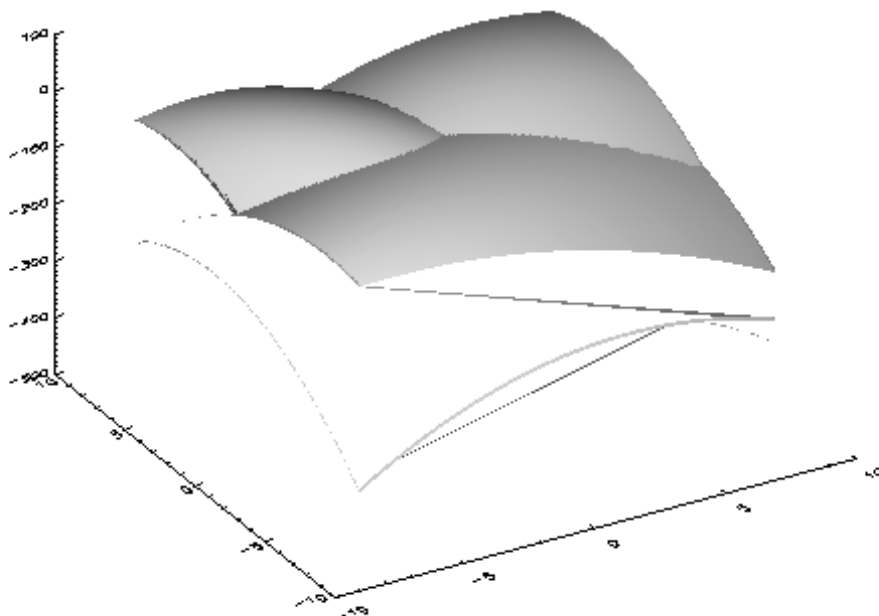


Figure 3.4: Discriminant function for the three data classes shown in the previous example.

3.1.5 Decision Regions and Decision Boundaries with minimum error decisions

A decision region \mathcal{F}_i consists of all the points such that $L_i(\mathbf{x}) < L_j(\mathbf{x})$ for all $j \neq i$. One can find the decision boundary between regions \mathcal{F}_i and \mathcal{F}_j by finding the points that fall on the contour

$$L_i(\mathbf{x}) = L_j(\mathbf{x}) \quad (3.18)$$

Tracing the boundaries between regions is helpful in visualizing the layout of feature space. However, one has to be careful, because a point on the boundary between \mathcal{F}_i and \mathcal{F}_j may actually belong to \mathcal{F}_m . This will happen if $L_m(\mathbf{x})$ is the lowest loss. Equation (3.18) only looks for the points that

are on the boundary between two particular regions. Further examination is usually needed.

We can find the decision boundary for the minimum error criterion by solving

$$g_i(\mathbf{x}) = g_j(\mathbf{x}) \quad (3.19)$$

In general, these boundaries are quadratic surfaces of dimension $M - 1$ in the M -dimensional feature space. If we substitute (3.18) into (3.19) and rearrange we find

$$(\mathbf{x} - \mu_i)\Sigma_i^{-1}(\mathbf{x} - \mu_i)' - (\mathbf{x} - \mu_j)\Sigma_j^{-1}(\mathbf{x} - \mu_j)' = \ln \frac{|\Sigma_j|}{|\Sigma_i|} \quad (3.20)$$

An example of decision boundaries in a 2D feature space is shown in Figure 3.5. The boundaries are nonlinear, and serve to separate the three classes on a pair-by-pair basis. Note that the three boundaries cross at a single point.

Case of equal covariance matrices

When the classes all have the same covariance, we can drop the subscripts on the Σ matrices and write (3.20) as

$$(\mathbf{x} - \mu_i)\Sigma^{-1}(\mathbf{x} - \mu_i)' - (\mathbf{x} - \mu_j)\Sigma^{-1}(\mathbf{x} - \mu_j)' = 0 \quad (3.21)$$

This is the equation of a $M - 1$ dimensional plane that divides the M dimensional feature space into regions.

If there is reason to believe that all of the classes have equal covariance, then one can compute a single covariance matrix by subtracting the mean value from the data for each class and pooling the results. When that is done, we find that the covariance function for the sample data of Table 3.1 is

$$\Sigma = \begin{bmatrix} 0.939 & 0.146 \\ 0.146 & 0.721 \end{bmatrix}$$

We can now see in Figure 3.6 that all of the boundaries are straight lines. This will always be the case with a multivariate normal distribution when the classes have equal covariance matrices. It is worth emphasizing that in this problem the covariance matrices of the individual classes are really not equal. However, by using a covariance matrix that is computed with the pooled data we are forced to use the same covariance matrix with each class. The effect can be seen by comparing Figures 3.3 and 3.6.

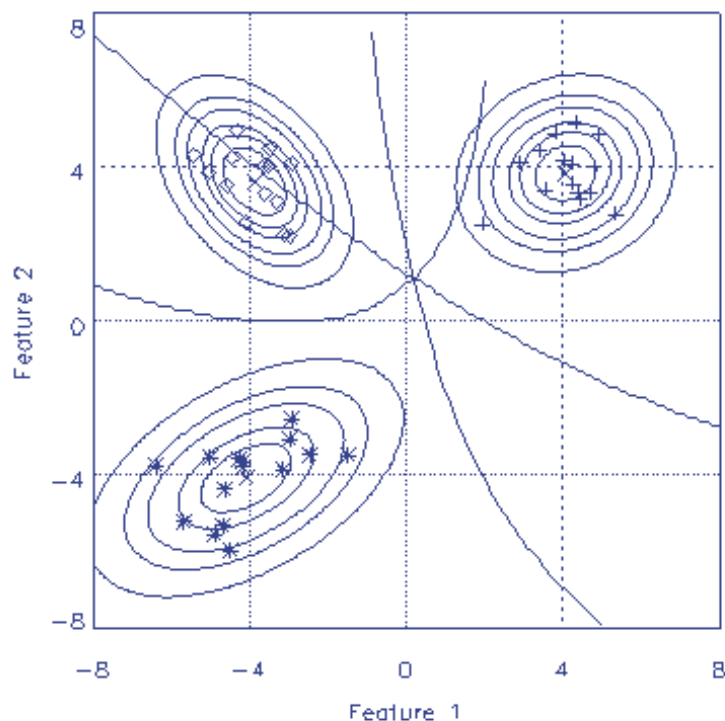


Figure 3.5: An illustration of the boundaries between the decision regions for a system with three classes. The individual covariance functions were used to compute the discriminant functions, leading to the curved boundaries.

3.2 Problems

1. The feature space for this pattern recognition problem is two dimensional and there are two pattern classes denoted by α_1 and α_2 . The probability density function for the feature vector \mathbf{x} , for each of the classes is normal. The parameters for each class are

$$\mu_1 = [4, 0] \quad \Sigma_1 = \begin{bmatrix} 1 & .2 \\ .2 & 1 \end{bmatrix}$$

$$\mu_2 = [0, -4] \quad \Sigma_2 = \begin{bmatrix} 1 & .2 \\ .2 & 1 \end{bmatrix}$$

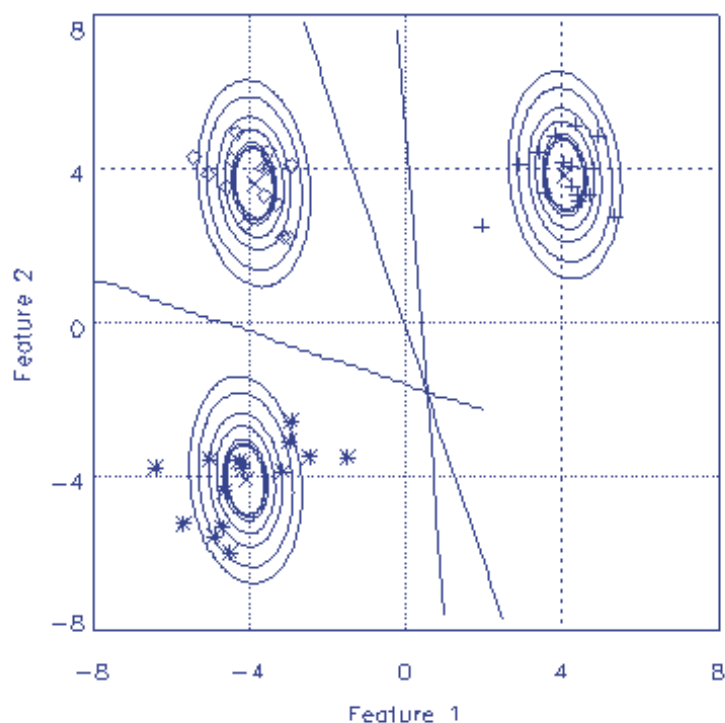


Figure 3.6: An illustration of the boundaries between the decision regions when a pooled covariance matrix is used. This produces linear boundaries that are not necessarily in the optimum positions.

- Sketch the equal probability contours for $P(\mathbf{x}|\alpha_1)$ and $P(\mathbf{x}|\alpha_2)$.
- Assume that $P(\alpha_1) = P(\alpha_2)$ and that a minimum-error decision rule is to be used. Find the equation of the boundary between the regions \mathcal{F}_1 and \mathcal{F}_2 .
- Assume that $P(\alpha_1) = 2P(\alpha_2)$ and that a minimum-error decision rule is to be used. Find the equation of the boundary between the regions \mathcal{F}_1 and \mathcal{F}_2 .
- Assume that $P(\alpha_1) = P(\alpha_2)$ and that the decision risks are given by $\lambda(\omega_1|\alpha_2) = 1$, $\lambda(\omega_2|\alpha_2) = 2$, and $\lambda(\omega_1|\alpha_1) = \lambda(\omega_2|\alpha_1) = 0$. Find the equation of the boundary between the regions \mathcal{F}_1 and \mathcal{F}_2 .

(e) Assume that $P(\alpha_1) = 2P(\alpha_2)$ and that the decision risks are given by $\lambda(\omega_1|\alpha_2) = 1$, $\lambda(\omega_2|\alpha_2) = 2$, and $\lambda(\omega_1|\alpha_1) = \lambda(\omega_2|\alpha_1) = 0$. Find the equation of the boundary between the regions \mathcal{F}_1 and \mathcal{F}_2 .

2. The feature space for this pattern recognition problem is two dimensional and there are two pattern classes denoted by α_1 and α_2 . The probability density function for the feature vector \mathbf{x} , for each of the classes is normal. The parameters for each class are

$$\begin{aligned} \mu_1 &= [4, 0] & \Sigma_1 &= \begin{bmatrix} 1 & .2 \\ .2 & 1 \end{bmatrix} \\ \mu_2 &= [0, -4] & \Sigma_2 &= \begin{bmatrix} 1 & -.2 \\ -.2 & 1 \end{bmatrix} \end{aligned}$$

(a) Sketch the equal probability contours for $P(\mathbf{x}|\alpha_1)$ and $P(\mathbf{x}|\alpha_2)$.

(b) Assume that $P(\alpha_1) = P(\alpha_2)$ and that a minimum-error decision rule is to be used. Find the equation of the boundary between the regions \mathcal{F}_1 and \mathcal{F}_2 .

(c) Assume that $P(\alpha_1) = 2P(\alpha_2)$ and that a minimum-error decision rule is to be used. Find the equation of the boundary between the regions \mathcal{F}_1 and \mathcal{F}_2 .

(d) Assume that $P(\alpha_1) = P(\alpha_2)$ and that the decision risks are given by $\lambda(\omega_1|\alpha_2) = 1$, $\lambda(\omega_2|\alpha_2) = 2$, and $\lambda(\omega_1|\alpha_1) = \lambda(\omega_2|\alpha_1) = 0$. Find the equation of the boundary between the regions \mathcal{F}_1 and \mathcal{F}_2 .

(e) Assume that $P(\alpha_1) = 2P(\alpha_2)$ and that the decision risks are given by $\lambda(\omega_1|\alpha_2) = 1$, $\lambda(\omega_2|\alpha_2) = 2$, and $\lambda(\omega_1|\alpha_1) = \lambda(\omega_2|\alpha_1) = 0$. Find the equation of the boundary between the regions \mathcal{F}_1 and \mathcal{F}_2 .

3. The feature space for this pattern recognition problem is three dimensional and there are two pattern classes denoted by α_1 and α_2 . The probability density function for the feature vector \mathbf{x} , for each of the

classes is normal. The parameters for each class are

$$\mu_1 = [-1, -1, -1] \quad \Sigma_1 = \begin{bmatrix} 1 & 0.5 & 0 \\ 0.5 & 1 & 0.5 \\ 0 & 0.5 & 1 \end{bmatrix}$$

$$\mu_2 = [1, 1, 1] \quad \Sigma_2 = \begin{bmatrix} 1 & 0.5 & 0 \\ 0.5 & 1 & 0.5 \\ 0 & 0.5 & 1 \end{bmatrix}$$

Assume that $P(\alpha_1) = P(\alpha_2)$ and that a minimum-error decision rule is to be used. Find the equation of the boundary between the regions \mathcal{F}_1 and \mathcal{F}_2 .

4. Reproduce equations (3.10), (3.11) and (3.12) by using the data in Table 3.1. You may want to write a computer program using functions such as those given at the end of this chapter.
5. Find the mean and covariance matrix for each class for the jockey, football player and swimmer data and plot the equal probability contours for the feature vectors for each class assuming that the data can be fit with a bivariate normal distribution. Plot the decision boundaries on the feature space for the minimum error probability decision rule.
6. Find the pooled covariance matrix for the jockey, football player and swimmer data. Plot the equal probability contours and the decision region boundaries for the minimum error probability decision rule under the assumption that each class has a covariance equal to the one you calculated from the pooled data. What is the effect on the location of the decision boundaries compared with your answer to the previous problem?

3.3 Computer programs

Computer programs to calculate the mean and covariance can easily be written in IDL. Listed below are programs for functions `mean(H)` and `cov(H)` where `H` is an array of feature data. All of the data in `H` is assumed to be from the same class. To find the mean and covariance for all of the classes it is necessary to run the computation with a data array for each class. `H`

is assumed to be arranged with each feature vector as a row. There are as many rows as there are examples and as many columns as there are features.

```

FUNCTION MEAN,H
;+
; MU=MEAN(H) returns an array that has one less array dimension
; than does H. That is, if H is a 2D array then MU is a 1D
; vector. If H is a 3D array then MU is a 2D array formed by
; averaging over the third dimension of H. Each value of MU is
; the average value over the highest dimension of H.
; If H is a scalar then H is returned. MU is always of type FLOAT
; unless H is of type DOUBLE, COMPLEX or DOUBLE COMPLEX, in
; which case MU is the same type as H.
;-
S=SIZE(H)
; S[0] is the number of dimensions of H
; S[S[0]] is the number of elements in the highest dimension
; of H.
IF S[0] EQ 0 THEN RETURN,H $
ELSE RETURN,TOTAL(H,S[0])/FLOAT(S[S[0]])
END
=====
FUNCTION COV,H
;+
; C=COV(H) returns the covariance matrix for a table of data
; in an array H. H is assumed to be arranged with each example
; as a row and each column as a feature. C is an array of size
; NxN where N is the number of columns of H. C is always of type
; FLOAT unless H is of type DOUBLE, COMPLEX or DOUBLE COMPLEX,
; in which case C is the same type as H.
;-
S=SIZE(H)
MU=MEAN(H)
MUV=(FLTARR(1,S[2])+1)##MU
X=H-MUV
RETURN,X#TRANSPOSE(X)/FLOAT(S[2]-1)
END

```