CHAPTER

1

INTRODUCTION

The ease with which we recognize a face, understand spoken words, read handwritten characters, identify our car keys in our pocket by feel, and decide whether an apple is ripe by its smell belies the astoundingly complex processes that underlie these acts of pattern recognition. Pattern recognition—the act of taking in raw data and taking an action based on the "category" of the pattern—has been crucial for our survival, and over the past tens of millions of years we have evolved highly sophisticated neural and cognitive systems for such tasks.

1.1 MACHINE PERCEPTION

It is natural that we should seek to design and build machines that can recognize patterns. From automated speech recognition, fingerprint identification, optical character recognition, DNA sequence identification, and much more, it is clear that reliable, accurate pattern recognition by machine would be immensely useful. Moreover, in solving the myriad problems required to build such systems, we gain deeper understanding and appreciation for pattern recognition systems in the natural world—most particularly in humans. For some problems, such as speech and visual recognition, our design efforts may in fact be influenced by knowledge of how these are solved in nature, both in the algorithms we employ and in the design of special-purpose hardware.

1.2 AN EXAMPLE

To illustrate the complexity of some of the types of problems involved, let us consider the following imaginary and somewhat fanciful example. Suppose that a fish-packing plant wants to automate the process of sorting incoming fish on a conveyor belt according to species. As a pilot project it is decided to try to separate sea bass from salmon using optical sensing. We set up a camera, take some sample images, and begin to note some physical differences between the two types of fish—length, lightness, width, number and shape of fins, position of the mouth, and so on—and these suggest *features* to explore for use in our classifier. We also notice noise or

FEATURE

variations in the images—variations in lighting, position of the fish on the conveyor, even "static" due to the electronics of the camera itself.

MODEL

Given that there truly are differences between the population of sea bass and that of salmon, we view them as having different *models*—different descriptions, which are typically mathematical in form. The overarching goal and approach in pattern classification is to hypothesize the class of these models, process the sensed data to eliminate noise (not due to the models), and for any sensed pattern choose the model that corresponds best. Any techniques that further this aim should be in the conceptual toolbox of the designer of pattern recognition systems.

PREPROCESSING
SEGMENTATION
FEATURE EXTRACTION

Our prototype system to perform this very specific task might well have the form shown in Fig. 1.1. First the camera captures an image of the fish. Next, the camera's signals are *preprocessed* to simplify subsequent operations without losing relevant information. In particular, we might use a *segmentation* operation in which the images of different fish are somehow isolated from one another and from the background. The information from a single fish is then sent to a *feature extractor*, whose purpose is to reduce the data by measuring certain "features" or "properties."

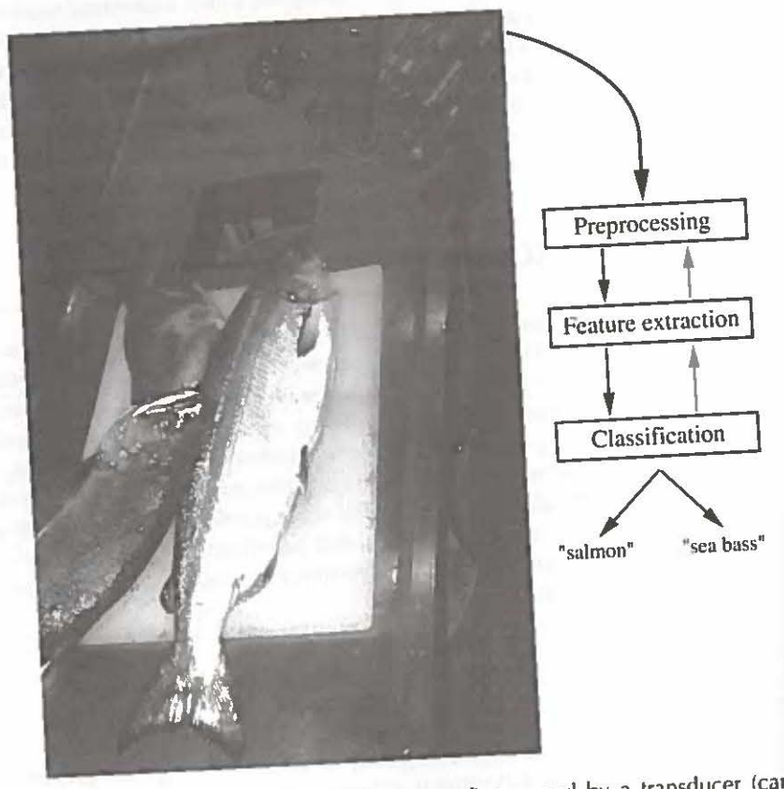


FIGURE 1.1. The objects to be classified are first sensed by a transducer (camera), whose signals are preprocessed. Next the features are extracted and finally the classification is emitted, here either "salmon" or "sea bass." Although the information flow is often chosen to be from the source to the classifier, some systems employ information flow in which earlier levels of processing can be altered based on the tentative or preliminary response in later levels (gray arrows). Yet others combine two or more stages into a unified step, such as simultaneous segmentation and feature extraction.

These features (or, more precisely, the values of these features) are then passed to a *classifier* that evaluates the evidence presented and makes a final decision as to the species.

The preprocessor might automatically adjust for average light level, or threshold the image to remove the background of the conveyor belt, and so forth. For the moment let us pass over how the images of the fish might be segmented and consider how the feature extractor and classifier might be designed. Suppose somebody at the fish plant tells us that a sea bass is generally longer than a salmon. These, then, give us our tentative *models* for the fish: Sea bass have some typical length, and this is greater than that for salmon. Then length becomes an obvious feature, and we might attempt to classify the fish merely by seeing whether or not the length l of a fish exceeds some critical value l^*. To choose l^* we could obtain some *design* or *training samples* of the different types of fish, make length measurements, and inspect the results.

TRAINING SAMPLES

Suppose that we do this and obtain the histograms shown in Fig. 1.2. These disappointing histograms bear out the statement that sea bass are somewhat longer than salmon, on average, but it is clear that this single criterion is quite poor; no matter how we choose l^*, we cannot reliably separate sea bass from salmon by length alone.

Discouraged, but undeterred by these unpromising results, we try another feature, namely the average lightness of the fish scales. Now we are very careful to eliminate variations in illumination, because they can only obscure the models and corrupt our new classifier. The resulting histograms and critical value x^*, shown in Fig. 1.3, are much more satisfactory: The classes are much better separated.

COST

So far we have tacitly assumed that the consequences of our actions are equally costly: Deciding the fish was a sea bass when in fact it was a salmon was just as undesirable as the converse. Such a symmetry in the *cost* is often, but not invariably, the case. For instance, as a fish-packing company we may know that our customers easily accept occasional pieces of tasty salmon in their cans labeled "sea bass," but they object vigorously if a piece of sea bass appears in their cans labeled "salmon." If we want to stay in business, we should adjust our decisions to avoid antagonizing our customers, even if it means that more salmon makes its way into the cans of

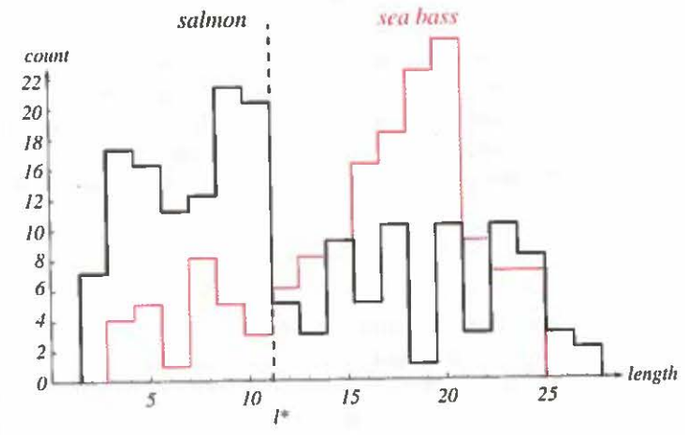


FIGURE 1.2. Histograms for the length feature for the two categories. No single threshold value of the length will serve to unambiguously discriminate between the two categories; using length alone, we will have some errors. The value marked l^* will lead to the smallest number of errors, on average.

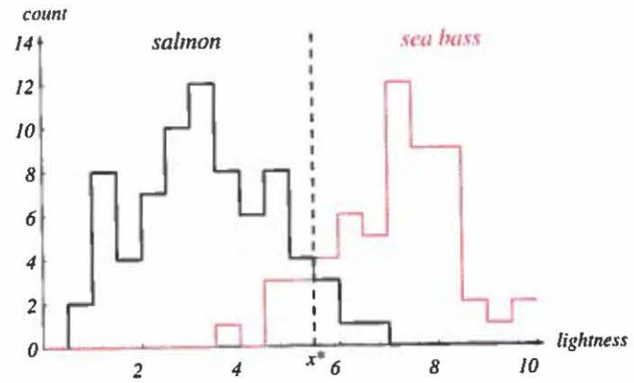


FIGURE 1.3. Histograms for the lightness feature for the two categories. No single threshold value x^* (decision boundary) will serve to unambiguously discriminate between the two categories; using lightness alone, we will have some errors. The value x^* marked will lead to the smallest number of errors, on average.

sea bass. In this case, then, we should move our decision boundary to smaller values of lightness, thereby reducing the number of sea bass that are classified as salmon (Fig. 1.3). The more our customers object to getting sea bass with their salmon (i.e., the more costly this type of error) the lower we should set the decision threshold x^* in Fig. 1.3.

DECISION THEORY

Such considerations suggest that there is an overall single cost associated with our decision, and our true task is to make a decision rule (i.e., set a decision boundary) so as to minimize such a cost. This is the central task of *decision theory* of which pattern classification is perhaps the most important subfield.

Even if we know the costs associated with our decisions and choose the optimal critical value x^*, we may be dissatisfied with the resulting performance. Our first impulse might be to seek yet a different feature on which to separate the fish. Let us assume, however, that no other single visual feature yields better performance than that based on lightness. To improve recognition, then, we must resort to the use of *more* than one feature at a time.

In our search for other features, we might try to capitalize on the observation that sea bass are typically wider than salmon. Now we have two features for classifying fish—the lightness x_1 and the width x_2. If we ignore how these features might be measured in practice, we realize that the feature extractor has thus reduced the image of each fish to a point or *feature vector* $\mathbf{x}$ in a two-dimensional *feature space*, where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

Our problem now is to partition the feature space into two regions, where for all points in one region we will call the fish a sea bass, and for all points in the other we call it a salmon. Suppose that we measure the feature vectors for our samples and obtain the scattering of points shown in Fig. 1.4. This plot suggests the following rule for separating the fish: Classify the fish as sea bass if its feature vector falls above the *decision boundary* shown, and as salmon otherwise.

DECISION BOUNDARY

This rule appears to do a good job of separating our samples and suggests that perhaps incorporating yet more features would be desirable. Besides the lightness

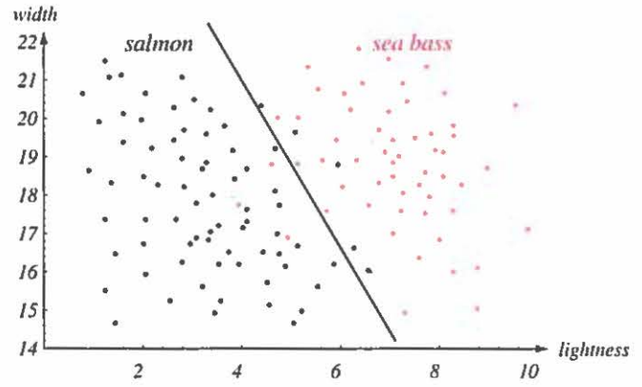


FIGURE 1.4. The two features of lightness and width for sea bass and salmon. The dark line could serve as a decision boundary of our classifier. Overall classification error on the data shown is lower than if we use only one feature as in Fig. 1.3, but there will still be some errors.

and width of the fish, we might include some shape parameter, such as the vertex angle of the dorsal fin, or the placement of the eyes (as expressed as a proportion of the mouth-to-tail distance), and so on. How do we know beforehand which of these features will work best? Some features might be redundant. For instance, if the eye color of all fish correlated perfectly with width, then classification performance need not be improved if we also include eye color as a feature. Even if the difficulty or computational cost in attaining more features is of no concern, might we ever have *too many* features—is there some "curse" for working in very high dimensions?

Suppose that other features are too expensive to measure, or provide little improvement (or possibly even degrade the performance) in the approach described above, and that we are forced to make our decision based on the two features in Fig. 1.4. If our models were extremely complicated, our classifier would have a decision boundary more complex than the simple straight line. In that case all the training patterns would be separated perfectly, as shown in Fig. 1.5. With such a "solution," though, our satisfaction would be premature because the central aim of designing a classifier is to suggest actions when presented with *novel* patterns, that is, fish not yet seen. This is the issue of *generalization*. It is unlikely that the complex decision boundary in Fig. 1.5 would provide good generalization—it seems to be "tuned" to the particular training samples, rather than some underlying characteristics or true model of all the sea bass and salmon that will have to be separated.

GENERALIZATION

Naturally, one approach would be to get more training samples for obtaining a better estimate of the true underlying characteristics, for instance the probability distributions of the categories. In some pattem recognition problems, however, the amount of such data we can obtain easily is often quite limited. Even with a vast amount of training data in a continuous feature space though, if we followed the approach in Fig. 1.5 our classifier would give a horrendously complicated decision boundary—one that would be unlikely to do well on novel patterns.

Rather, then, we might seek to "simplify" the recognizer, motivated by a belief that the underlying models will not require a decision boundary that is as complex as that in Fig. 1.5. Indeed, we might be satisfied with the slightly poorer performance on the training samples if it means that our classifier will have better performance

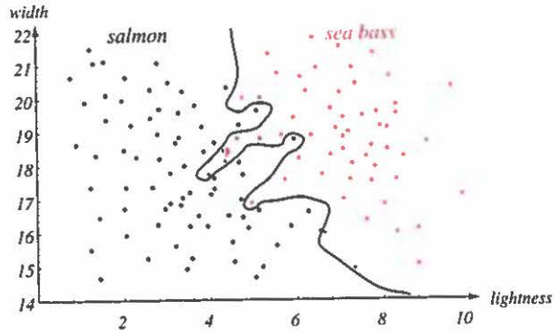


FIGURE 1.5. Overly complex models for the fish will lead to decision boundaries that are complicated. While such a decision may lead to perfect classification of our training samples, it would lead to poor performance on future patterns. The novel test point marked ? is evidently most likely a salmon, whereas the complex decision boundary shown leads it to be classified as a sea bass.

on novel patterns.* But if designing a very complex recognizer is unlikely to give good generalization, precisely how should we quantify and favor simpler classifiers? How would our system automatically determine that the simple curve in Fig. 1.6 is preferable to the manifestly simpler straight line in Fig. 1.4 or the complicated boundary in Fig. 1.5? Assuming that we somehow manage to optimize this tradeoff, can we then *predict* how well our system will generalize to new patterns? These are some of the central problems in *statistical pattern recognition*.

For the same incoming patterns, we might need to use a drastically different task or cost function, and this will lead to different actions altogether. We might, for instance, wish instead to separate the fish based on their sex—all females (of either species) from all males—if we wish to sell roe. Alternatively, we might wish to cull

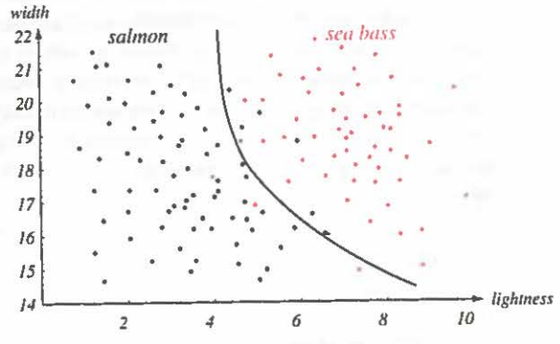


FIGURE 1.6. The decision boundary shown might represent the optimal tradeoff between performance on the training set and simplicity of classifier, thereby giving the highest accuracy on new patterns.

*The philosophical underpinnings of this approach derive from William of Occam (1284–1347?), who advocated favoring *simpler* explanations over those that are needlessly complicated: *Entia non sunt multiplicanda praeter necessitatem* ("Entities are not to be multiplied without necessity"). Decisions based on overly complex models often lead to lower accuracy of the classifier.

the damaged fish (to prepare separately for cat food), and so on. Different decision tasks may require features and yield boundaries quite different from those useful for our original categorization problem.

This makes it quite clear that our decisions are fundamentally task- or cost-specific, and that creating a single *general purpose* artificial pattern recognition device—that is, one capable of acting accurately based on a wide variety of tasks—is a profoundly difficult challenge. This, too, should give us added appreciation of the ability of humans to switch rapidly and fluidly between pattern recognition tasks.

Because classification is, at base, the task of recovering the model that generated the patterns, different classification techniques are useful depending on the type of candidate models themselves. In statistical pattern recognition we focus on the statistical properties of the patterns (generally expressed in probability densities), and this will command most of our attention in this book. Here the model for a pattern may be a single specific set of features, though the actual pattern sensed has been corrupted by some form of random noise. Occasionally it is claimed that *neural* pattern recognition (or neural network pattern classification) should be considered its own discipline, but despite its somewhat different intellectual pedigree, we will consider it a close descendant of statistical pattern recognition, for reasons that will become clear. If instead the model consists of some set of crisp logical rules, then we employ the methods of *syntactic* pattern recognition, where rules or grammars describe our decision. For example, we might wish to classify an English sentence as grammatical or not. Here crisp rules, rather than statistical descriptions of word frequencies or correlations, are appropriate.

It was necessary in our fish example to choose our features carefully, and hence achieve a *representation* (as in Fig. 1.6) that enabled reasonably successful pattern classification. A central aspect in virtually every pattern recognition problem is that of achieving such a "good" representation, one in which the structural relationships among the components are simply and naturally revealed, and one in which the true (unknown) model of the patterns can be expressed. In some cases, patterns should be represented as vectors of real-valued numbers, in others ordered lists of attributes, in yet others descriptions of parts and their relations, and so forth. We seek a representation in which the patterns that lead to the same action are somehow "close" to one another, yet "far" from those that demand a different action. The extent to which we create or learn a proper representation and how we quantify near and far apart will determine the success of our pattern classifier. A number of additional characteristics are desirable for the representation. We might wish to favor a small number of features, which might lead to (a) simpler decision regions, and (b) a classifier easier to train. We might also wish to have features that are robust—that is, relatively insensitive to noise or other errors. In practical applications we may need the classifier to act *quickly*, or use few electronic components, memory or processing steps.

ANALYSIS BY SYNTHESIS

A central technique, when we have insufficient training data, is to incorporate knowledge of the problem domain. Indeed, the less the training data, the more important is such knowledge—for instance, how the patterns themselves were produced. One method that takes this notion to its logical extreme is that of *analysis by synthesis*, where in the ideal case one has a model of how each pattern is generated. Consider speech recognition. Amidst the manifest acoustic variability among the possible "dee"s that might be uttered by different people, one thing they have in common is that they were all produced by lowering the jaw slightly, opening the mouth, placing the tongue tip against the roof of the mouth after a certain delay, and so on. We might assume that "all" the acoustic variation is due to the happenstance of whether the talker is male or female, old or young, with different overall pitches, and so forth.

At some deep level, such a "physiological" model (or so-called "motor" model) for production of the "dee" utterances is appropriate, and different (say) from that for "doo" and indeed all other utterances. *If* this underlying model of production can be determined from the sound (and that is a very big *if*), then we can classify the utterance by how it was produced. That is to say, the production representation may be the "best" representation for classification. Our pattern recognition systems should then analyze (and hence classify) the input pattern based on how one would have to synthesize that pattern. The trick is, of course, to recover the generating parameters from the sensed pattern.

Consider the difficulty in making a recognizer of all types of chairs—standard office chair, contemporary living room chair, beanbag chair, and so forth—based on an image. Given the astounding variety in the number of legs, material, shape, and so on, we might despair of ever finding a representation that reveals the unity within the class of chair. Perhaps the only such unifying aspect of chairs is *functional*: A chair is a stable artifact that supports a human sitter, including back support. Thus we might try to deduce such functional properties from the image; and the property "can support a human sitter" is very indirectly related to the orientation of the larger surfaces, and it would need to be answered in the affirmative even for a beanbag chair. Of course, this requires some reasoning about the properties and naturally touches upon computer vision rather than pattern recognition proper.

Without going to such extremes, many real-world pattern recognition systems seek to incorporate at least *some* knowledge about the method of production of the patterns or their functional use in order to ensure a good representation, though of course the goal of the representation is classification, not reproduction. For instance, in optical character recognition (OCR) one might confidently assume that handwritten characters are written as a sequence of strokes and might first try to recover a stroke representation from the sensed image and then deduce the character from the identified strokes.

1.2.1 Related Fields

Pattern classification differs from classical statistical *hypothesis testing*, wherein the sensed data are used to decide whether or not to reject a *null hypothesis* in favor of some alternative hypothesis. Roughly speaking, if the probability of obtaining the data given some null hypothesis falls below a "significance" threshold, we reject the null hypothesis in favor of the alternative. Hypothesis testing is often used to determine whether a drug is effective, where the null hypothesis is that it has no effect. Hypothesis testing might be used to determine whether the fish on the conveyor belt belong to a single class (all salmon, for instance)—the null hypothesis—or instead from two classes (the alternative).

IMAGE PROCESSING

Pattern classification differs, too, from *image processing*. In image processing, the input is an image and the output is an image. Image processing steps often include rotation, contrast enhancement, and other transformations which preserve all the original information. Feature extraction, such as finding the peaks and valleys of the intensity, loses information (but hopefully preserves everything relevant to the task at hand).

ASSOCIATIVE MEMORY

As just described, *feature extraction* takes in a pattern and produces feature values. The number of features is virtually always chosen to be fewer than the total necessary to describe the complete target of interest, and this leads to a loss in information. In acts of *associative memory*, the system takes in a pattern and emits another pattern which is representative of a general group of patterns. It thus reduces the information

somewhat, but rarely to the extent that pattern classification does. In short, because of the crucial role of a *decision* in pattern recognition information, it is fundamentally an information reduction process. You cannot reconstruct a pattern given only its category membership. The classification step represents an even more radical loss of information, reducing the original several thousand bits representing all the color of each of several thousand pixels down to just a few bits representing the chosen category (a single bit in our fish example.)

REGRESSION

Three closely interrelated fields, which are often employed in pattern recognition research, are regression, interpolation, and density estimation. In *regression*, we seek to find some functional description of data, often with the goal of predicting values for new input. Linear regression—in which the function is linear in the input variables—is by far the most popular and well studied form of regression. We might, for instance, feel that the length of a salmon varies linearly with its age or with weight, and take measurements of the age and length of many typical salmon and then use linear regression to find the coefficients.

INTERPOLATION

In *interpolation* we know or can easily deduce the function for certain ranges of input; the problem is then to infer the function for intermediate ranges of input. Thus we might know how the length of a salmon varies as age in the first two weeks of life, and above two years of age. We might then use any of a variety of interpolatation methods to infer how the length depends upon age between two weeks and two years of age.

DENSITY ESTIMATION

Density estimation is the problem of estimating the density (or probability) that a member of a certain category will be found to have particular features.

These fields are often employed—explicitly or implicitly—as first steps in pattern recognition. For instance, we shall see several methods for estimating the densities of different categories; an unknown pattern is then classified according to which category is the most probable. While these fields are highly developed and useful, we shall only indirectly address them as they relate to pattern classification.

1.3 PATTERN RECOGNITION SYSTEMS

In describing our hypothetical fish classification system, we distinguished between the three different operations of preprocessing, feature extraction and classification (see Fig. 1.1). Figure 1.7 shows a slightly more elaborate diagram of the components of a typical pattern recognition system. To understand the problem of designing such a system, we must understand the problems that each of these components must solve. Let us consider the operations of each component in turn, and reflect on the kinds of problems that can arise.

1.3.1 Sensing

The input to a pattern recognition system is often some kind of a transducer, such as a camera or a microphone array. The difficulty of the problem may well depend on the characteristics and limitations of the transducer—its bandwidth, resolution, sensitivity, distortion, signal-to-noise ratio, latency, etc. As important as it is in practice, the design of sensors for pattern recognition is beyond the scope of this book.

1.3.2 Segmentation and Grouping

In our fish example, we tacitly assumed that each fish was isolated, separate from others on the conveyor belt, and could easily be distinguished from the conveyor belt.

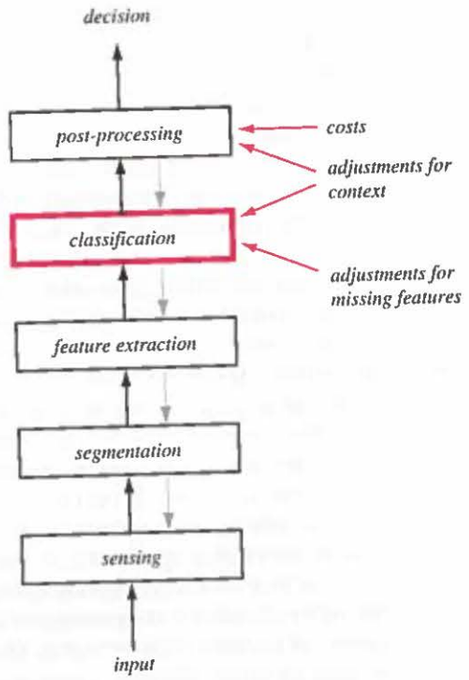


FIGURE 1.7. Many pattern recognition systems can be partitioned into components such as the ones shown here. A sensor converts images or sounds or other physical inputs into signal data. The segmentor isolates sensed objects from the background or from other objects. A feature extractor measures object properties that are useful for classification. The classifier uses these features to assign the sensed object to a category. Finally, a post processor can take account of other considerations, such as the effects of context and the costs of errors, to decide on the appropriate action. Although this description stresses a one-way or "bottom-up" flow of data, some systems employ feedback from higher levels back down to lower levels (gray arrows).

In practice, the fish would often be abutting or overlapping, and our system would have to determine where one fish ends and the next begins—the individual patterns have to be *segmented*. If we have already recognized the fish then it would be easier to segment their images. But how can we segment the images before they have been categorized, or categorize them before they have been segmented? It seems we need a way to know when we have switched from one model to another, or to know when we just have background or "no category." How can this be done?

Segmentation is one of the deepest problems in pattern recognition. In automated speech recognition, we might seek to recognize the individual sounds (e.g., phonemes, such as "ss," "k," ...) and then put them together to determine the word. But consider two nonsense words, "sklee" and "skloo." Speak them aloud and notice that for "skloo" you push your lips forward (so-called "rounding" in anticipation of the upcoming "oo") *before* you utter the "ss." Such rounding influences the sound of the "ss," lowering the frequency spectrum compared to the "ss" sound in "sklee"—a phenomenon known as anticipatory coarticulation. Thus, the "oo" phoneme reveals its presence in the "ss" *earlier* than the "k" and "l" which nominally occur *before* the "oo" itself! How do we segment the "oo" phoneme from the others when they are so manifestly intermingled? Or should we even try? Perhaps we are focusing on groupings of the wrong size, and that the most useful unit for recognition is somewhat larger.

Closely related to the problem of segmentation is the problem of recognizing or grouping together the various parts of a composite object. The letter **i** or the symbol = have two connected components, but we see them as one symbol. We effortlessly read a simple word such as **BEATS**. But consider this: Why didn't we read instead *other* words that are perfectly good subsets of the full pattern, such as **BE**, **BEAT**, **EAT**, **AT**, and **EATS**? Why don't they enter our minds, unless explicitly brought to our attention? Or when we saw the **B** why didn't we read a **P** or an **I**, which are "there" within the **B**? Conversely, how is it that we can read the two unsegmented words in **POLOPONY**—without placing the *entire* input into a single word category?

MEREOLOGY

This is the problem of *subsets and supersets*—formally part of *mereology*, the study of part/whole relationships. It appears as though the best classifiers try to incorporate as much of the input into the categorization as "makes sense," but not too much. How can this be done automatically?

1.3.3 Feature Extraction

The conceptual boundary between feature extraction and classification proper is somewhat arbitrary: An ideal feature extractor would yield a representation that makes the job of the classifier trivial; conversely, an omnipotent classifier would not need the help of a sophisticated feature extractor. The distinction is forced upon us for practical, rather than theoretical reasons.

The traditional goal of the feature extractor is to characterize an object to be recognized by measurements whose values are very similar for objects in the same category, and very different for objects in different categories. This leads to the idea of seeking *distinguishing features* that are *invariant* to irrelevant transformations of the input. In our fish example, the absolute location of a fish on the conveyor belt is irrelevant to the category, and thus our representation should also be insensitive to the absolute location of the fish. Ideally, in this case we want the features to be invariant to translation, whether horizontal or vertical. Because rotation is also irrelevant for classification, we would also like the features to be invariant to rotation. Finally, the size of the fish may not be important—a young, small salmon is still a salmon. Thus, we may also want the features to be invariant to scale. In general, features that describe properties such as shape, color and many kinds of texture are invariant to translation, rotation and scale.

INVARIANT FEATURES

TRANSLATION ROTATION

SCALE

The problem of finding rotation invariant features from an overhead image of a fish on a conveyor belt is simplified by the fact that the fish is likely to be lying flat, and the axis of rotation is always parallel to the camera's line of sight. A more general invariance would be for rotations about an arbitrary line in three dimensions. The image of even such a "simple" object as a coffee cup undergoes radical variation as the cup is rotated to an arbitrary angle: The handle may become *occluded*—that is, hidden by another part. The bottom of the inside volume come into view, the circular lip appear oval or a straight line or even obscured, and so forth. Furthermore, if the distance between the cup and the camera can change, the image is subject to projective distortion. How might we ensure that the features are invariant to such complex transformations? Or should we define different subcategories for the image of a cup and achieve the rotation invariance at a higher level of processing?

OCCLUSION

PROJECTIVE DISTORTION

In speech recognition, we want features that are invariant to translations in time and to changes in the overall amplitude. We may also want features that are in-

RATE

sensitive to the duration of the word, i.e., invariant to the *rate* at which the pattern evolves. Rate variation is a serious problem in speech recognition. Not only do different people talk at different rates, but even a single talker may vary in rate, causing the speech signal to change in complex ways. Likewise, cursive handwriting varies in complex ways as the writer speeds up—the placement of dots on the i's, and cross bars on the t's and f's, are the first casualties of rate increase, while the appearance of l's and e's are relatively inviolate. How can we make a recognizer that changes its representations for some categories *differently* from that for others under such rate variation?

A large number of highly complex transformations arise in pattern recognition, and many are domain specific. We might wish to make our handwritten optical character recognizer insensitive to the overall thickness of the pen line, for instance. Far more severe are transformations such as *nonrigid deformations* that arise in three-dimensional object recognition, such as the radical variation in the image of your hand as you grasp an object or snap your fingers. Similarly, variations in illumination or the complex effects of cast shadows may need to be taken into account.

DEFORMATION

As with segmentation, the task of feature extraction is much more problem- and domain-dependent than is classification proper, and thus requires knowledge of the domain. A good feature extractor for sorting fish would probably be of little use for identifying fingerprints, or classifying photomicrographs of blood cells. However, some of the principles of pattern classification can be used in the design of the feature extractor. Although the pattern classification techniques presented in this book cannot substitute for domain knowledge, they can be helpful in making the feature values less sensitive to noise. In some cases, they can also be used to select the most valuable features from a larger set of candidate features.

FEATURE SELECTION

1.3.4 Classification

The task of the classifier component proper of a full system is to use the feature vector provided by the feature extractor to assign the object to a category. Most of this book is concerned with the design of the classifier. Because perfect classification performance is often impossible, a more general task is to determine the probability for each of the possible categories. The abstraction provided by the feature-vector representation of the input data enables the development of a largely domain-independent theory of classification.

The degree of difficulty of the classification problem depends on the variability in the feature values for objects in the same category relative to the difference between feature values for objects in different categories. The variability of feature values for objects in the same category may be due to complexity, and may be due to *noise*. We define noise in very general terms: any property of the sensed pattern which is not due to the true underlying model but instead to randomness in the world or the sensors. All nontrivial decision and pattern recognition problems involve noise in some form. What is the best way to design a classifier to cope with this variability? What is the best performance that is possible?

NOISE

One problem that arises in practice is that it may not always be possible to determine the values of all of the features for a particular input. In our hypothetical system for fish classification, for example, it may not be possible to determine the width of the fish because of occlusion by another fish. How should the categorizer compensate? Since our two-feature recognizer never had a single-variable criterion value x^* determined in anticipation of the possible absence of a feature (cf. Fig. 1.3), how shall it make the best decision using only the feature present? The naïve method, of merely assuming that the value of the missing feature is zero or the average of the values for the patterns already seen, is provably nonoptimal. Likewise, how should we train a classifier or use one when some features are missing?

1.3.5 Post Processing

A classifier rarely exists in a vacuum. Instead, it is generally to be used to recommend actions (put this fish in this bucket, put that fish in that bucket), each action having an associated cost. The post-processor uses the output of the classifier to decide on the recommended action.

ERROR RATE

Conceptually, the simplest measure of classifier performance is the classification error rate—the percentage of new patterns that are assigned to the wrong category. Thus, it is common to seek minimum-error-rate classification. However, it may be much better to recommend actions that will minimize the total expected cost, which is called the *risk*. How do we incorporate knowledge about costs and how will they affect our classification decision? Can we estimate the total risk and thus tell whether our classifier is acceptable even before we field it? Can we estimate the lowest possible risk of *any* classifier, to see how close ours meets this ideal, or whether the problem is simply too hard overall?

RISK

CONTEXT

The post-processor might also be able to exploit *context*—input-dependent information other than from the target pattern itself—to improve system performance. Suppose in an optical character recognition system we encounter a sequence that looks like `T/-\E C/-\T`. Even though the system may be unable to classify each `/-\` as an isolated character, in the context of English it is clear that the first instance should be an `H` and the second an `A`. Context can be highly complex and abstract. The utterance "jeetyet?" may seem nonsensical, unless you hear it spoken by a friend in the context of the cafeteria at lunchtime—"did you eat yet?" How can such a visual and temporal context influence your recognition of speech?

In our fish example we saw how using multiple features could lead to improved recognition. We might imagine that we could also do better if we used multiple classifiers, each classifier operating on different aspects of the input. For example, we might combine the results of acoustic recognition and lip reading to improve the performance of a speech recognizer.

MULTIPLE CLASSIFIERS

If all of the classifiers agree on a particular pattern, there is no difficulty. But suppose they disagree. How should a "super" classifier *pool the evidence* from the component recognizers to achieve the best decision? Imagine calling in ten experts for determining whether or not a particular fish is diseased. While nine agree that the fish is healthy, one expert does not. Who is right? It may be that the lone dissenter is the only one familiar with the particular very rare symptoms in the fish, and is in fact correct. How would the "super" categorizer know when to base a decision on a minority opinion, even from an expert in one small domain who is not well-qualified to judge throughout a broad range of problems?

We have asked more questions in this section than we have answered. Our purpose was to emphasize the complexity of pattern recognition problems and to dispel naïve hope that any single approach has the power to solve all pattern recognition problems. The methods presented in this book are primarily useful for the classification step. We shall see that they also have relevance to those segmentation, feature extraction and post-processing problems that are not highly domain-dependent. However, performance on difficult pattern recognition problems generally requires exploiting domain-specific knowledge.

1.4 THE DESIGN CYCLE

The design of a pattern recognition system usually entails the repetition of a number of different activities: data collection, feature choice, model choice, training, and evaluation. In this section we present an overview of this design cycle (Fig. 1.8) and consider some of the problems that frequently arise.

1.4.1 Data Collection

Data collection can account for surprisingly large part of the cost of developing a pattern recognition system. It may be possible to perform a preliminary feasibility study with a small set of "typical" examples, but much more data will usually be needed to assure good performance in the fielded system. How do we know when we have collected an adequately large and representative set of examples for training and testing the system?

1.4.2 Feature Choice

The choice of the distinguishing features is a critical design step and depends on the characteristics of the problem domain. Having access to example data, such as

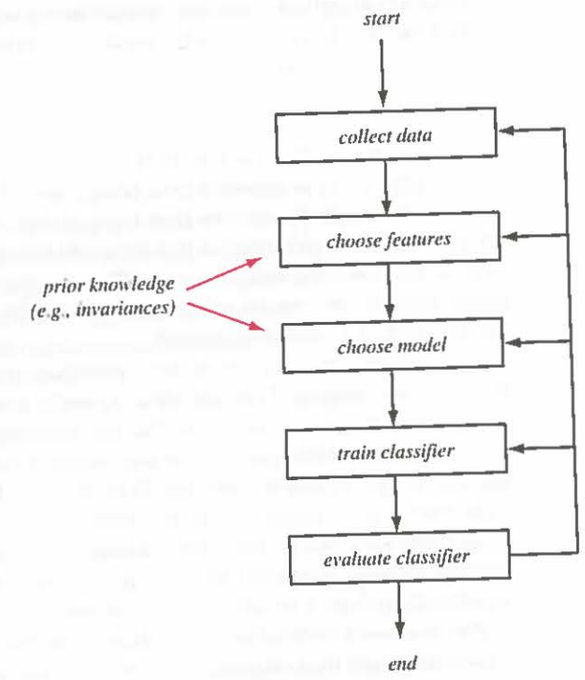


FIGURE 1.8. The design of a pattern recognition system involves a design cycle similar to the one shown here. Data must be collected, both to train and to test the system. The characteristics of the data impact both the choice of appropriate discriminating features and the choice of models for the different categories. The training process uses some or all of the data to determine the system parameters. The results of evaluation may call for repetition of various steps in this process in order to obtain satisfactory results.

PRIOR KNOWLEDGE

pictures of fish on the conveyor belt, will certainly be valuable for choosing a feature set. However, *prior knowledge* also plays a major role.

In our hypothetical fish-classification example, prior knowledge about the lightness of the different fish categories helped in the design of a classifier by suggesting a promising feature. Incorporating prior knowledge can be far more subtle and difficult. In some applications the knowledge ultimately derives from information about the production of the patterns, as we saw in analysis-by-synthesis. In others the knowledge may be about the *form* of the underlying categories, or specific attributes of the patterns, such as the fact that a face has two eyes, one nose, and so on.

In selecting or designing features, we obviously would like to find features that are simple to extract, invariant to irrelevant transformations, insensitive to noise, and useful for discriminating patterns in different categories. How do we combine prior knowledge and empirical data to find relevant and effective features?

1.4.3 Model Choice

We might have been unsatisfied with the performance of our fish classifier in Figs. 1.4 and 1.5, and thus jumped to an entirely different class of model, for instance one based on some function of the number and position of the fins, the color of the eyes, the weight, shape of the mouth, and so on. How do we know when a hypothesized model differs significantly from the true model underlying our patterns, and thus a new model is needed? In short, how are we to know to reject a class of models and try another one? Are we as designers reduced to random and tedious trial and error in model selection, never really knowing whether we can expect improved performance? Or might there be principled methods for knowing when to jettison one class of models and invoke another?

1.4.4 Training

In general, the process of using data to determine the classifier is referred to as *training* the classifier. Much of this book is concerned with the many different procedures for training classifiers and choosing models.

We have already seen many problems that arise in the design of pattern recognition systems. No universal methods have been found for solving all of these problems. However, the repeated experience of the last quarter century has been that the most effective methods for developing classifiers involve learning from example patterns. Throughout this book, we shall see again and again how methods of learning relate to these central problems, and how they are essential in the engineering of pattern recognition systems.

1.4.5 Evaluation

When we went from the use of one feature to two in our fish classification problem, it was essentially the result of an evaluation that the error rate we could obtain with one feature was inadequate, and that it was possible to do better. When we went from the simple straight-line classifier in Fig. 1.4 to the more complicated model illustrated in Fig. 1.5, it was again the result of an evaluation that it was possible to do still better. Evaluation is important both to measure the performance of the system and to identify the need for improvements in its components.

While an overly complex system may allow perfect classification of the training samples, it is unlikely perform well on new patterns. This situation is known as *overfitting*. One of the most important areas of research in statistical pattern classification is determining how to adjust the complexity of the model—not so simple that it cannot explain the differences between the categories, yet not so complex as to give poor classification on novel patterns. Are there principled methods for finding the best (intermediate) complexity for a classifier?

OVERFITTING

1.4.6 Computational Complexity

Some pattern recognition problems can be "solved" using algorithms that are highly impractical. For instance, we might try to hand label all possible 20×20 binary pixel images with a category label for optical character recognition, and use table lookup to classify incoming patterns. Although we might in theory achieve error-free recognition, the labeling time and storage requirements would be quite prohibitive since it would require a labeling each of $2^{20\times 20} \approx 10^{120}$ patterns. Thus the computational resources necessary and the computational complexity of different algorithms are of considerable practical importance.

In more general terms, we may ask how an algorithm scales as a function of the number of feature dimensions, or the number of patterns or the number of categories. What is the tradeoff between computational ease and performance? In some problems we know we can design an excellent recognizer, but not within the engineering constraints. How can we optimize *within* such constraints? We are typically less concerned with the complexity of learning, which is done in the laboratory, than with the complexity of making a decision, which is done with the fielded application. While computational complexity generally correlates with the complexity of the hypothesized model of the patterns, these two notions are conceptually different.

1.5 LEARNING AND ADAPTATION

In the broadest sense, any method that incorporates information from training samples in the design of a classifier employs learning. Because nearly all practical or interesting pattern recognition problems are so hard that we cannot guess the best classification decision ahead of time, we shall spend the great majority of our time here considering learning. Creating classifiers then involves positing some general form of model, or form of the classifier, and using training patterns to learn or estimate the unknown parameters of the model. Learning refers to some form of algorithm for reducing the error on a set of training data. A range of *gradient descent* algorithms that alter a classifier's parameters in order to reduce an error measure now permeate the field of statistical pattern recognition, and these will demand a great deal of our attention. Learning comes in several general forms.

1.5.1 Supervised Learning

In supervised learning, a teacher provides a category label or cost for each pattern in a training set, and seeks to reduce the sum of the costs for these patterns. How can we be sure that a particular learning algorithm is powerful enough to learn the solution to a given problem and that it will be stable to parameter variations? How can we determine if it will converge in finite time or if it will scale reasonably with the number of training patterns, the number of input features or the number of categories? How can we ensure that the learning algorithm appropriately favors "simple" solutions (as in Fig. 1.6) rather than complicated ones (as in Fig. 1.5)?

1.5.2 Unsupervised Learning

In *unsupervised learning* or *clustering* there is no explicit teacher, and the system forms clusters or "natural groupings" of the input patterns. "Natural" is always defined explicitly or implicitly in the clustering system itself; and given a particular set of patterns or cost function, different clustering algorithms lead to different clusters. Often the user will set the hypothesized number of different clusters ahead of time, but how should this be done? How do we avoid inappropriate representations?

1.5.3 Reinforcement Learning

The most typical way to train a classifier is to present an input, compute its tentative category label, and use the known target category label to improve the classifier. For instance, in optical character recognition, the input might be an image of a character, the actual output of the classifier the category label "R," and the desired output a "B." In *reinforcement learning* or *learning with a critic*, no desired category signal is given; instead, the only teaching feedback is that the tentative category is right or wrong. This is analogous to a critic who merely states that something is right or wrong, but does not say specifically *how* it is wrong. In pattern classification, it is most common that such reinforcement is binary—either the tentative decision is correct or it is not. How can the system learn from such nonspecific feedback?

CRITIC

1.6 CONCLUSION

At this point the reader may be overwhelmed by the number, complexity, and magnitude of the subproblems of pattern recognition. Furthermore, these subproblems are rarely addressed in isolation and they are invariably interrelated. Thus for instance in seeking to reduce the complexity of our classifier, we might affect its ability to deal with invariance. We point out, however, that the good news is at least threefold: (1) There is an "existence proof" that many of these problems can indeed be solved—as demonstrated by humans and other biological systems, (2) mathematical theories solving some of these problems have in fact been discovered, and finally (3) there remain many fascinating unsolved problems providing opportunities for progress.

SUMMARY BY CHAPTERS

This book first addresses those cases where a great deal of information about the models is known (such as the probability densities, category labels,...) and moves, chapter by chapter, toward problems where the form of the distributions are unknown and even the category membership of training patterns is unknown. We begin in Chapter 2 (Bayesian Decision Theory) by considering the ideal case in which the probability structure underlying the categories is known perfectly. While this sort of situation rarely occurs in practice, it permits us to determine the optimal (Bayes) classifier against which we can compare all other classifiers. Moreover, in some problems

it enables us to predict the error we will get when we generalize to novel patterns. In Chapter 3 (Maximum-Likelihood and Bayesian Parameter Estimation) we address the case when the full probability structure underlying the categories is not known, but the general *forms* of their distributions *are* known. Thus the uncertainty about a probability distribution is represented by the values of some unknown parameters, and we seek to determine these parameters to attain the best categorization. In Chapter 4 (Nonparametric Techniques) we move yet further from the Bayesian ideal, and assume that we have *no* prior parameterized knowledge about the underlying probability structure; in essence our classification will be based on information provided by training samples alone. Classic techniques such as the nearest-neighbor algorithm and potential functions play an important role here.

Then in Chapter 5 (Linear Discriminant Functions) we return somewhat toward the general approach of parameter estimation. We shall assume that the so-called "discriminant functions" are of a very particular form—namely linear—in order to derive a class of incremental training rules. Next, in Chapter 6 (Multilayer Neural Networks) we see how some of the ideas from such linear discriminants can be extended to a class of very powerful algorithms for training multilayer neural networks; these neural techniques have a range of useful properties that have made them a mainstay in contemporary pattern recognition research. In Chapter 7 (Stochastic Methods) we discuss simulated annealing, the Boltzmann learning algorithm and other stochastic methods which can avoid some of the estimation problems that plague other neural methods. Chapter 8 (Nonmetric Methods) moves beyond models that are statistical in nature to ones that can be best described by logical rules. Here we discuss tree-based algorithms such as CART (which can also be applied to statistical data) and syntactic-based methods based on grammars.

Chapter 9 (Algorithm-Independent Machine Learning) is both the most important chapter and the most difficult one in the book. Some of the results described there—those related to bias and variance, degrees of freedom, the desire for "simple" classifiers, and computational complexity—are subtle but nevertheless crucial both theoretically and practically. In some sense, the other chapters can only be fully understood (or used) in light of the results presented here.

We conclude in Chapter 10 (Unsupervised Learning and Clustering) by addressing the case when input training patterns are not labeled, and where our recognizer must determine the cluster structure. We also treat a related problem, that of learning with a critic, in which the teacher provides only a single bit of information during the presentation of a training pattern—"yes," that the classification provided by the recognizer is correct, or "no," it isn't.

BIBLIOGRAPHICAL AND HISTORICAL REMARKS

Classification is among the first crucial steps in making sense of the blooming buzzing confusion of sensory data that intelligent systems confront. In the Western world, the foundations of pattern recognition can be traced to Plato [2], which were later extended by Aristotle [1], who distinguished between an "essential property" (which would be shared by all members in a class or "natural kind" as he put it) from an "accidental property" (which could differ among members in the class). Pattern recognition can be cast as the problem of finding such essential properties of a category. In the Eastern world, the first Zen patriarch, Bodhidharma, would point at things and demand students to answer "What is that?" as a way of confronting the deepest issues in mind, the identity of objects, and the nature of classification and decision [3]. It has been a central theme in the discipline of philosophical epistemology, the study of the nature of knowledge. A more modern treatment of some philosophical problems of pattern recognition, relating to the technical matter in the current book, can be found in references [22, 4] and [18]. A delightful and particularly insightful book on the foundations of artificial intelligence, including pattern recognition, is reference [10].

There are a number of overviews and reference books that can be recommended, including references [5] and [6]. The modern literature on decision theory and pattern recognition is now overwhelming, and comprises dozens of journals, thousands of books and conference proceedings and innumerable articles; it continues to grow rapidly. While some disciplines such as statistics [8], machine learning [17], and neural networks [9], expand the foundations of pattern recognition, others, such as computer vision [7, 19] and speech recognition [16], rely on it heavily. Perceptual Psychology, Cognitive Science [13], Psychobiology [21], and Neuroscience [11] analyze how pattern recognition is achieved in humans and other animals. The extreme view that everything in human cognition—including rule-following and logic—can be reduced to pattern recognition is presented in reference [14]. Pattern recognition techniques have been applied in virtually every scientific and technical discipline.

BIBLIOGRAPHY

[1] Aristotle, Robin Waterfield, and David Bostock. *Physics*. Oxford University Press, Oxford, UK, 1996.

[2] Allan Bloom. *The Republic of Plato*. Basic Books, New York, second edition, 1991.

[3] Bodhidharma. *The Zen Teachings of Bodhidharma*. North Point Press, San Francisco, CA, 1989.

[4] Mikhail M. Bongard. *Pattern Recognition*. Spartan Books, Washington, D.C., 1970.

[5] Chi-hau Chen, Louis François Pau, and Patrick S. P. Wang, editors. *Handbook of Pattern Recognition & Computer Vision*. World Scientific, Singapore, second edition, 1993.

[6] Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, New York, 1996.

[7] Marty Fischler and Oscar Firschein. *Readings in Computer Vision: Issues, Problems, Principles and Paradigms*. Morgan Kaufmann, San Mateo, CA, 1987.

[8] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, second edition, 1990.

[9] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwood City, CA, 1991.

[10] Douglas Hofstadter. *Gödel, Escher, Bach: An Eternal Golden Braid*. Basic Books, New York, 1979.

[11] Eric R. Kandel and James H. Schwartz. *Principles of Neural Science*. Elsevier, New York, second edition, 1985.

[12] Immanuel Kant. *Critique of Pure Reason*. Prometheus Books, New York, 1990.

[13] George F. Luger. *Cognitive Science: The Science of Intelligent Systems*. Academic Press, New York, 1994.

[14] Howard Margolis. *Patterns, Thinking, and Cognition: A Theory of Judgement*. University of Chicago Press, Chicago, IL, 1987.

[15] Karl Raimund Popper. *Popper Selections*. Princeton University Press, Princeton, NJ, 1985.

[16] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, Englewood Cliffs, NJ, 1993.

[17] Jude W. Shavlik and Thomas G. Dietterich, editors. *Readings in Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1990.

[18] Brian Cantwell Smith. *On the Origin of Objects*. MIT Press, Cambridge, MA, 1996.

[19] Louise Stark and Kevin Bowyer. *Generic Object Recognition Using Form & Function*. World Scientific, River Edge, NJ, 1996.

[20] Donald R. Tveter. *The Pattern Recognition Basis of Artificial Intelligence*. IEEE Press, New York, 1998.

[21] William R. Uttal. *The Psychobiology of Sensory Coding*. HarperCollins, New York, 1973.

[22] Satoshi Watanabe. *Knowing and Guessing: A Quantitative Study of Inference and Information*. Wiley, New York, 1969.

CHAPTER

2

BAYESIAN DECISION THEORY

2.1 INTRODUCTION

Bayesian decision theory is a fundamental statistical approach to the problem of pattern classification. This approach is based on quantifying the tradeoffs between various classification decisions using probability and the costs that accompany such decisions. It makes the assumption that the decision problem is posed in probabilistic terms, and that all of the relevant probability values are known. In this chapter we develop the fundamentals of this theory and we show how it can be viewed as being simply a formalization of common-sense procedures; in subsequent chapters we will consider the problems that arise when the probabilistic structure is not completely known.

While we will give a quite general, abstract development of Bayesian decision theory in Section 2.2, we begin our discussion with a specific example. Let us reconsider the hypothetical problem posed in Chapter 1 of designing a classifier to separate two kinds of fish: sea bass and salmon. Suppose that an observer watching fish arrive along the conveyor belt finds it hard to predict what type will emerge next and that the sequence of types of fish appears to be random. In decision-theoretic terminology we would say that as each fish emerges nature is in one or the other of the two possible states: Either the fish is a sea bass or the fish is a salmon. We let ω denote the *state of nature*, with $\omega = \omega_1$ for sea bass and $\omega = \omega_2$ for salmon. Because the state of nature is so unpredictable, we consider ω to be a variable that must be described probabilistically.

STATE OF NATURE

If the catch produced as much sea bass as salmon, we would say that the next fish is equally likely to be sea bass or salmon. More generally, we assume that there is some *a priori probability* (or simply *prior*) $P(\omega_1)$ that the next fish is sea bass, and some prior probability $P(\omega_2)$ that it is salmon. If we assume there are no other types of fish relevant here, then $P(\omega_1)$ and $P(\omega_2)$ sum to one. These prior probabilities reflect our prior knowledge of how likely we are to get a sea bass or salmon before the fish actually appears. It might, for instance, depend upon the time of year or the choice of fishing area.

PRIOR

Suppose for a moment that we were forced to make a decision about the type of fish that will appear next without being allowed to see it. For the moment, we shall assume that any incorrect classification entails the same cost or consequence, and

that the only information we are allowed to use is the value of the prior probabilities. If a decision must be made with so little information, it seems logical to use the following *decision rule*: Decide ω_1 if $P(\omega_1) > P(\omega_2)$; otherwise decide ω_2.

DECISION RULE

This rule makes sense if we are to judge just one fish, but if we are to judge many fish, using this rule repeatedly may seem a bit strange. After all, we would always make the same decision even though we know that *both* types of fish will appear. How well it works depends upon the values of the prior probabilities. If $P(\omega_1)$ is very much greater than $P(\omega_2)$, our decision in favor of ω_1 will be right most of the time. If $P(\omega_1) = P(\omega_2)$, we have only a fifty-fifty chance of being right. In general, the probability of error is the smaller of $P(\omega_1)$ and $P(\omega_2)$, and we shall see later that under these conditions no other decision rule can yield a larger probability of being right.

In most circumstances we are not asked to make decisions with so little information. In our example, we might for instance use a lightness measurement x to improve our classifier. Different fish will yield different lightness readings, and we express this variability in probabilistic terms; we consider x to be a continuous random variable whose distribution depends on the state of nature and is expressed as $p(x|\omega)$.* This is the *class-conditional probability density* function, the probability density function for x given that the state of nature is ω. (It is also sometimes called state-conditional probability density.) Then the difference between $p(x|\omega_1)$ and $p(x|\omega_2)$ describes the difference in lightness between populations of sea bass and salmon (Fig. 2.1).†

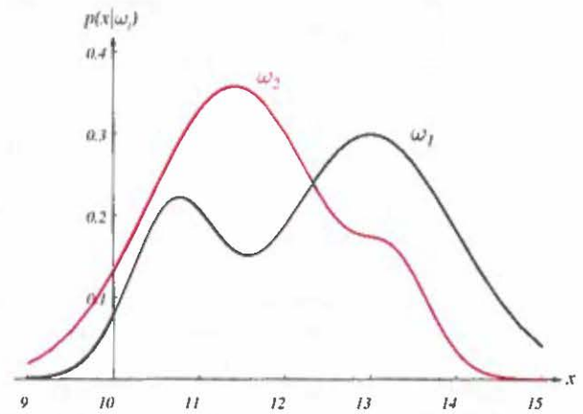


FIGURE 2.1. Hypothetical class-conditional probability density functions show the probability density of measuring a particular feature value x given the pattern is in category ω_i. If x represents the lightness of a fish, the two curves might describe the difference in lightness of populations of two types of fish. Density functions are normalized, and thus the area under each curve is 1.0.

*We generally use an uppercase $P(\cdot)$ to denote a probability mass function and use a lowercase $p(\cdot)$ to denote a probability density function.

†Strictly speaking, the probability density function $p(x|\omega)$ should be written as $p_X(x|\omega)$ to indicate that we are speaking about a particular density function for the random variable X. This more elaborate subscripted notation makes it clear that $p_X(\cdot)$ and $p_Y(\cdot)$ denote two different functions, a fact that is obscured when writing $p(x)$ and $p(y)$. Because this potential confusion rarely arises in practice, we have elected to adopt the simpler notation. Readers who are unsure of our notation or who would like to review probability theory should see Section A.4 of the Appendix.

Suppose that we know both the prior probabilities $P(\omega_j)$ and the conditional densities $p(x|\omega_j)$ for $j = 1, 2$. Suppose further that we measure the lightness of a fish and discover that its value is x. How does this measurement influence our attitude concerning the true state of nature—that is, the category of the fish? We note first that the (joint) probability density of finding a pattern that is in category ω_j *and* has feature value x can be written in two ways: $p(\omega_j, x) = P(\omega_j|x)p(x) = p(x|\omega_j)P(\omega_j)$. Rearranging these leads us to the answer to our question, which is called *Bayes formula*:

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)}, \tag{1}$$

where in this case of two categories

$$p(x) = \sum_{j=1}^{2} p(x|\omega_j)P(\omega_j). \tag{2}$$

Bayes formula can be expressed informally in English by saying that

$$posterior = \frac{likelihood \times prior}{evidence}. \tag{3}$$

POSTERIOR

LIKELIHOOD

EVIDENCE

Bayes formula shows that by observing the value of x we can convert the prior probability $P(\omega_j)$ to the *a posteriori* probability (or *posterior*) probability $P(\omega_j|x)$—the probability of the state of nature being ω_j given that feature value x has been measured. We call $p(x|\omega_j)$ the *likelihood* of ω_j with respect to x, a term chosen to indicate that, other things being equal, the category ω_j for which $p(x|\omega_j)$ is large is more "likely" to be the true category. Notice that it is the product of the likelihood and the prior probability that is most important in determining the posterior probability; the *evidence* factor, $p(x)$, can be viewed as merely a scale factor that guarantees that the posterior probabilities sum to one, as all good probabilities must. The variation of $P(\omega_j|x)$ with x is illustrated in Fig. 2.2 for the case $P(\omega_1) = 2/3$ and $P(\omega_2) = 1/3$.

If we have an observation x for which $P(\omega_1|x)$ is greater than $P(\omega_2|x)$, we would naturally be inclined to decide that the true state of nature is ω_1. Conversely, if $P(\omega_2|x)$ is greater than $P(\omega_1|x)$, we would be inclined to choose ω_2. To justify this decision procedure, let us calculate the probability of error whenever we make a decision. Whenever we observe a particular x, the probability of error is

$$P(error|x) = \begin{cases} P(\omega_1|x) & \text{if we decide } \omega_2 \\ P(\omega_2|x) & \text{if we decide } \omega_1. \end{cases} \tag{4}$$

Clearly, for a given x we can minimize the probability of error by deciding ω_1 if $P(\omega_1|x) > P(\omega_2|x)$ and ω_2 otherwise. Of course, we may never observe exactly the same value of x twice. Will this rule minimize the average probability of error? Yes, because the average probability of error is given by

$$P(error) = \int_{-\infty}^{\infty} P(error, x)\, dx = \int_{-\infty}^{\infty} P(error|x)p(x)\, dx \tag{5}$$

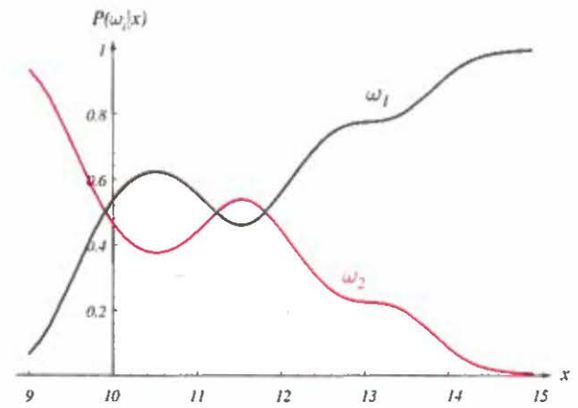


FIGURE 2.2. Posterior probabilities for the particular priors $P(\omega_1) = 2/3$ and $P(\omega_2) = 1/3$ for the class-conditional probability densities shown in Fig. 2.1. Thus in this case, given that a pattern is measured to have feature value $x = 14$, the probability it is in category ω_2 is roughly 0.08, and that it is in ω_1 is 0.92. At every x, the posteriors sum to 1.0.

BAYES DECISION RULE

and if for every x we ensure that $P(error|x)$ is as small as possible, then the integral must be as small as possible. Thus we have justified the following *Bayes decision rule* for minimizing the probability of error:

$$\text{Decide } \omega_1 \text{ if } P(\omega_1|x) > P(\omega_2|x); \quad \text{otherwise decide } \omega_2. \tag{6}$$

Under this rule Eq. 4 becomes

$$P(error|x) = \min\left[P(\omega_1|x), P(\omega_2|x)\right]. \tag{7}$$

EVIDENCE

This form of the decision rule emphasizes the role of the posterior probabilities. By using Eq. 1, we can instead express the rule in terms of the conditional and prior probabilities. First note that the *evidence*, $p(x)$, in Eq. 1 is unimportant as far as making a decision is concerned. It is basically just a scale factor that states how frequently we will actually measure a pattern with feature value x; as mentioned above, its presence in Eq. 1 assures us that $P(\omega_1|x) + P(\omega_2|x) = 1$. By eliminating this scale factor, we obtain the following completely equivalent decision rule:

$$\text{Decide } \omega_1 \text{ if } p(x|\omega_1)P(\omega_1) > p(x|\omega_2)P(\omega_2); \quad \text{otherwise decide } \omega_2. \tag{8}$$

Some additional insight can be obtained by considering a few special cases. If for some x we have $p(x|\omega_1) = p(x|\omega_2)$, then that particular observation gives us no information about the state of nature; in this case, the decision hinges entirely on the prior probabilities. On the other hand, if $P(\omega_1) = P(\omega_2)$, then the states of nature are equally probable; in this case the decision is based entirely on the likelihoods $p(x|\omega_j)$. In general, both of these factors are important in making a decision, and the Bayes decision rule combines them to achieve the minimum probability of error.

2.2 BAYESIAN DECISION THEORY—CONTINUOUS FEATURES

We shall now formalize the ideas just considered, and generalize them in four ways:

- By allowing the use of more than one feature
- By allowing more than two states of nature
- By allowing actions other than merely deciding the state of nature
- By introducing a loss function more general than the probability of error

These generalizations and their attendant notational complexities should not obscure the central points illustrated in our simple example. Allowing the use of more than one feature merely requires replacing the scalar x by the *feature vector* $\mathbf{x}$, where

FEATURE SPACE

$\mathbf{x}$ is in a d-dimensional Euclidean space $\mathbf{R}^d$, called the *feature space*. Allowing more than two states of nature provides us with a useful generalization for a small notational expense. Allowing actions other than classification primarily allows the possibility of rejection—that is, of refusing to make a decision in close cases; this is a

LOSS FUNCTION

useful option if being indecisive is not too costly. Formally, the *loss function* states exactly how costly each action is, and is used to convert a probability determination into a decision. Cost functions let us treat situations in which some kinds of classification mistakes are more costly than others, although we often discuss the simplest case, where all errors are equally costly. With this as a preamble, let us begin the more formal treatment.

Let $\{\omega_1, \ldots, \omega_c\}$ be the finite set of c states of nature ("categories") and let $\{\alpha_1, \ldots, \alpha_a\}$ be the finite set of a possible actions. The loss function $\lambda(\alpha_i|\omega_j)$ describes the loss incurred for taking action α_i when the state of nature is ω_j. Let the feature vector $\mathbf{x}$ be a d-component vector-valued random variable and let $p(\mathbf{x}|\omega_j)$ be the state-conditional probability density function for $\mathbf{x}$, with the probability density function for $\mathbf{x}$ conditioned on ω_j being the true state of nature. As before, $P(\omega_j)$ describes the prior probability that nature is in state ω_j. Then the posterior probability $P(\omega_j|\mathbf{x})$ can be computed from $p(\mathbf{x}|\omega_j)$ by Bayes formula:

$$P(\omega_j|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_j)P(\omega_j)}{p(\mathbf{x})}, \tag{9}$$

where the evidence is now

$$p(\mathbf{x}) = \sum_{j=1}^{c} p(\mathbf{x}|\omega_j)P(\omega_j). \tag{10}$$

Suppose that we observe a particular $\mathbf{x}$ and that we contemplate taking action α_i. If the true state of nature is ω_j, by definition we will incur the loss $\lambda(\alpha_i|\omega_j)$. Because $P(\omega_j|\mathbf{x})$ is the probability that the true state of nature is ω_j, the expected loss associated with taking action α_i is merely

$$R(\alpha_i|\mathbf{x}) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j)P(\omega_j|\mathbf{x}). \tag{11}$$

RISK

In decision-theoretic terminology, an expected loss is called a *risk*, and $R(\alpha_i|\mathbf{x})$ is called the *conditional risk*. Whenever we encounter a particular observation $\mathbf{x}$, we

can minimize our expected loss by selecting the action that minimizes the conditional risk. We shall now show that this *Bayes decision procedure* actually provides the optimal performance.

DECISION RULE

Stated formally, our problem is to find a decision rule against $P(\omega_j)$ that minimizes the overall risk. A general *decision rule* is a function $\alpha(\mathbf{x})$ that tells us which action to take for every possible observation. To be more specific, for every $\mathbf{x}$ the *decision function* $\alpha(\mathbf{x})$ assumes one of the a values $\alpha_1, \ldots, \alpha_a$. The overall risk R is the expected loss associated with a given decision rule. Because $R(\alpha_i|\mathbf{x})$ is the conditional risk associated with action α_i and because the decision rule specifies the action, the overall risk is given by

$$R = \int R(\alpha(\mathbf{x})|\mathbf{x})p(\mathbf{x})\,d\mathbf{x}, \tag{12}$$

where $d\mathbf{x}$ is our notation for a d-space volume element and where the integral extends over the entire feature space. Clearly, if $\alpha(\mathbf{x})$ is chosen so that $R(\alpha_i(\mathbf{x}))$ is as small as possible for every $\mathbf{x}$, then the overall risk will be minimized. This justifies the following statement of the *Bayes decision rule*: To minimize the overall risk, compute the conditional risk

$$R(\alpha_i|\mathbf{x}) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j)P(\omega_j|\mathbf{x}) \tag{13}$$

BAYES RISK

for $i = 1, \ldots, a$ and then select the action α_i for which $R(\alpha_i|\mathbf{x})$ is minimum.* The resulting minimum overall risk is called the *Bayes risk*, denoted R^*, and is the best performance that can be achieved.

2.2.1 Two-Category Classification

Let us consider these results when applied to the special case of two-category classification problems. Here action α_1 corresponds to deciding that the true state of nature is ω_1, and action α_2 corresponds to deciding that it is ω_2. For notational simplicity, let $\lambda_{ij} = \lambda(\alpha_i|\omega_j)$ be the loss incurred for deciding ω_i when the true state of nature is ω_j. If we write out the conditional risk given by Eq. 13, we obtain

$$R(\alpha_1|\mathbf{x}) = \lambda_{11}P(\omega_1|\mathbf{x}) + \lambda_{12}P(\omega_2|\mathbf{x}) \tag{14}$$

$$R(\alpha_2|\mathbf{x}) = \lambda_{21}P(\omega_1|\mathbf{x}) + \lambda_{22}P(\omega_2|\mathbf{x}). \tag{15}$$

There are a variety of ways of expressing the minimum-risk decision rule, each having its own minor advantages. The fundamental rule is to decide ω_1 if $R(\alpha_1|\mathbf{x}) < R(\alpha_2|\mathbf{x})$. In terms of the posterior probabilities, we decide ω_1 if

$$(\lambda_{21} - \lambda_{11})P(\omega_1|\mathbf{x}) > (\lambda_{12} - \lambda_{22})P(\omega_2|\mathbf{x}). \tag{16}$$

Ordinarily, the loss incurred for making an error is greater than the loss incurred for being correct, and both of the factors $\lambda_{21}-\lambda_{11}$ and $\lambda_{12}-\lambda_{22}$ are positive. Thus in practice, our decision is generally determined by the more likely state of nature, although

*Note that if more than one action minimizes $R(\alpha|\mathbf{x})$, it does not matter which of these actions is taken, and any convenient tie-breaking rule can be used.

we must scale the posterior probabilities by the loss differences. By employing Bayes formula, we can replace the posterior probabilities by the prior probabilities and the conditional densities. This results in the equivalent rule, to decide ω_1 if

$$(\lambda_{21} - \lambda_{11})p(\mathbf{x}|\omega_1)P(\omega_1) > (\lambda_{12} - \lambda_{22})p(\mathbf{x}|\omega_2)P(\omega_2), \tag{17}$$

and ω_2 otherwise.

Another alternative, which follows at once under the reasonable assumption that $\lambda_{21} > \lambda_{11}$, is to decide ω_1 if

$$\frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} > \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \frac{P(\omega_2)}{P(\omega_1)}. \tag{18}$$

LIKELIHOOD RATIO

This form of the decision rule focuses on the x-dependence of the probability densities. We can consider $p(\mathbf{x}|\omega_j)$ a function of ω_j (i.e., the likelihood function) and then form the *likelihood ratio* $p(\mathbf{x}|\omega_1)/p(\mathbf{x}|\omega_2)$. Thus the Bayes decision rule can be interpreted as calling for deciding ω_1 if the likelihood ratio exceeds a threshold value that is independent of the observation x.

2.3 MINIMUM-ERROR-RATE CLASSIFICATION

In classification problems, each state of nature is usually associated with a different one of the c classes, and the action α_i is usually interpreted as the decision that the true state of nature is ω_i. If action α_i is taken and the true state of nature is ω_j, then the decision is correct if $i = j$ and in error if $i \neq j$. If errors are to be avoided, it is natural to seek a decision rule that minimizes the probability of error, that is, the *error rate*.

ZERO-ONE LOSS

The loss function of interest for this case is hence the so-called *symmetrical* or *zero-one* loss function,

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases} \qquad i, j = 1, \ldots, c. \tag{19}$$

This loss function assigns no loss to a correct decision, and assigns a unit loss to any error; thus, all errors are equally costly.* The risk corresponding to this loss function is precisely the average probability of error because the conditional risk is

$$\begin{aligned} R(\alpha_i|\mathbf{x}) &= \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j) P(\omega_j|\mathbf{x}) \\ &= \sum_{j \neq i} P(\omega_j|\mathbf{x}) \\ &= 1 - P(\omega_i|\mathbf{x}) \end{aligned} \tag{20}$$

*We note that other loss functions, such as quadratic and linear difference, find greater use in regression tasks where there is a natural ordering on the predictions and we can meaningfully penalize predictions that are "more wrong" than others.

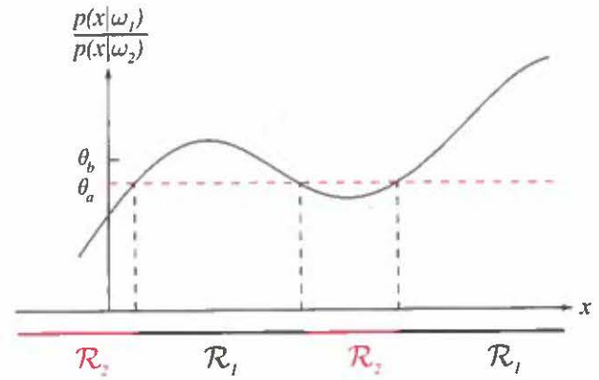


FIGURE 2.3. The likelihood ratio $p(x|\omega_1)/p(x|\omega_2)$ for the distributions shown in Fig. 2.1. If we employ a zero-one or classification loss, our decision boundaries are determined by the threshold θ_a. If our loss function penalizes miscategorizing ω_2 as ω_1 patterns more than the converse, we get the larger threshold θ_b, and hence $\mathcal{R}_1$ becomes smaller.

and $P(\omega_i|\mathbf{x})$ is the conditional probability that action α_i is correct. The Bayes decision rule to minimize risk calls for selecting the action that minimizes the conditional risk. Thus, to minimize the average probability of error, we should select the i that *maximizes* the posterior probability $P(\omega_i|\mathbf{x})$. In other words, for *minimum error rate*:

$$\text{Decide } \omega_i \text{ if } P(\omega_i|\mathbf{x}) > P(\omega_j|\mathbf{x}) \qquad \text{for all } j \neq i. \tag{21}$$

This is the same rule as in Eq. 6. The region in the input space where we decide ω_i is denoted $\mathcal{R}_i$; such a region need not be simply connected.

We saw in Fig. 2.2 some class-conditional probability densities and the posterior probabilities; Fig. 2.3 shows the likelihood ratio $p(x|\omega_1)/p(x|\omega_2)$ for the same case. In general, this ratio can range between zero and infinity. The threshold value θ_a marked is from the same prior probabilities but with a zero-one loss function. Notice that this leads to the same decision boundaries as in Fig. 2.2, as it must. If we penalize mistakes in classifying ω_1 patterns as ω_2 more than the converse (i.e., $\lambda_{21} > \lambda_{12}$), then Eq. 18 leads to the threshold θ_b marked. Note that the range of x values for which we classify a pattern as ω_1 gets smaller, as it should.

*2.3.1 Minimax Criterion

Sometimes we must design our classifier to perform well over a *range* of prior probabilities. For instance, in our fish categorization problem we can imagine that whereas the physical properties of lightness and width of each type of fish remain constant, the prior probabilities might vary widely and in an unpredictable way, or alternatively we want to use the classifier in a different plant where we do not know the prior probabilities. A reasonable approach is then to design our classifier so that the *worst* overall risk for any value of the priors is as small as possible—that is, minimize the maximum possible overall risk.

In order to understand this, we let $\mathcal{R}_1$ denote that (as yet unknown) region in feature space where the classifier decides ω_1 and likewise for $\mathcal{R}_2$ and ω_2, and then we write our overall risk Eq. 12 in terms of conditional risks:

$$R = \int_{\mathcal{R}_1} [\lambda_{11} P(\omega_1)\ p(\mathbf{x}|\omega_1) + \lambda_{12} P(\omega_2)\ p(\mathbf{x}|\omega_2)]\ d\mathbf{x} + \int_{\mathcal{R}_2} [\lambda_{21} P(\omega_1)\ p(\mathbf{x}|\omega_1) + \lambda_{22} P(\omega_2)\ p(\mathbf{x}|\omega_2)]\ d\mathbf{x}. \tag{22}$$

We use the fact that $P(\omega_2) = 1 - P(\omega_1)$ and that $\int_{\mathcal{R}_1} p(\mathbf{x}|\omega_1)\ d\mathbf{x} = 1 - \int_{\mathcal{R}_2} p(\mathbf{x}|\omega_1)\ d\mathbf{x}$ to rewrite the risk as:

$$R(P(\omega_1)) = \overbrace{\lambda_{22} + (\lambda_{12} - \lambda_{22}) \int_{\mathcal{R}_1} p(\mathbf{x}|\omega_2)\ d\mathbf{x}}^{= R_{mm},\ \text{minimax risk}} + P(\omega_1) \underbrace{\left[(\lambda_{11} - \lambda_{22}) - (\lambda_{21} - \lambda_{11}) \int_{\mathcal{R}_2} p(\mathbf{x}|\omega_1)\ d\mathbf{x} - (\lambda_{12} - \lambda_{22}) \int_{\mathcal{R}_1} p(\mathbf{x}|\omega_2)\ d\mathbf{x} \right]}_{= 0 \text{ for minimax solution}}. \tag{23}$$

This equation shows that once the decision boundary is set (i.e., $\mathcal{R}_1$ and $\mathcal{R}_2$ determined), the overall risk is linear in $P(\omega_1)$. If we can find a boundary such that the constant of proportionality is 0, then the risk is independent of priors. This is the *minimax solution*, and the *minimax risk*, R_{mm}, can be read from Eq. 23:

MINIMAX RISK

$$R_{mm} = \lambda_{22} + (\lambda_{12} - \lambda_{22}) \int_{\mathcal{R}_1} p(\mathbf{x}|\omega_2)\ d\mathbf{x} = \lambda_{11} + (\lambda_{21} - \lambda_{11}) \int_{\mathcal{R}_2} p(\mathbf{x}|\omega_1)\ d\mathbf{x}. \tag{24}$$

Figure 2.4 illustrates the approach. Briefly stated, we search for the prior for which the Bayes risk is *maximum*, and the corresponding decision boundary gives the minimax solution. The value of the minimax risk, R_{mm}, is hence equal to the worst Bayes risk. In practice, finding the decision boundary for minimax risk may be difficult, particularly when distributions are complicated. Nevertheless, in some cases the boundary can be determined analytically (Problem 4).

The minimax criterion finds greater use in game theory than it does in traditional pattern recognition. In game theory, you have a hostile opponent who can be expected to take an action maximally detrimental to you. Thus it makes great sense for you to take an action (e.g., make a classification) where your costs—due to your opponent's subsequent actions—are minimized.

*2.3.2 Neyman-Pearson Criterion

In some problems, we may wish to minimize the overall risk subject to a constraint; for instance, we might wish to minimize the total risk subject to the constraint $\int R(\alpha_i|\mathbf{x})\ d\mathbf{x} < constant$ for some particular i. Such a constraint might arise when there is a fixed resource that accompanies one particular action α_i, or when we must not misclassify a pattern from a particular state of nature ω_i at more than some limited frequency. For instance, in our fish example, there might be some government regulation that we must not misclassify more than 1% of salmon as sea bass. We might then seek a decision that minimizes the chance of classifying a sea bass as a salmon subject to this condition.

We generally satisfy such a *Neyman-Pearson criterion* by adjusting decision boundaries numerically. However, for Gaussian and some other distributions, Neyman-Pearson solutions can be found analytically (Problems 6 and 7). We shall have cause to mention Neyman-Pearson criteria again in Section 2.8.3 on operating characteristics.

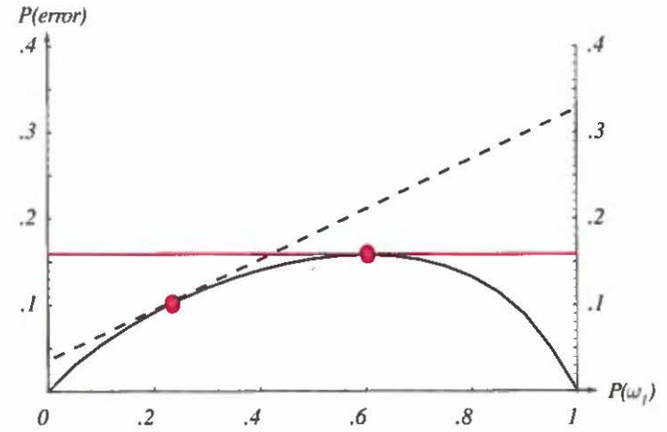


FIGURE 2.4. The curve at the bottom shows the minimum (Bayes) error as a function of prior probability $P(\omega_1)$ in a two-category classification problem of fixed distributions. For each value of the priors (e.g., $P(\omega_1) = 0.25$) there is a corresponding optimal decision boundary and associated Bayes error rate. For any (fixed) such boundary, if the priors are then changed, the probability of error will change as a linear function of $P(\omega_1)$ (shown by the dashed line). The maximum such error will occur at an extreme value of the prior, here at $P(\omega_1) = 1$. To minimize the maximum of such error, we should design our decision boundary for the maximum Bayes error (here $P(\omega_1) = 0.6$), and thus the error will not change as a function of prior, as shown by the solid red horizontal line.

2.4 CLASSIFIERS, DISCRIMINANT FUNCTIONS, AND DECISION SURFACES

2.4.1 The Multicategory Case

There are many different ways to represent pattern classifiers. One of the most useful is in terms of a set of *discriminant functions* $g_i(\mathbf{x})$, $i = 1, \ldots, c$. The classifier is said to assign a feature vector $\mathbf{x}$ to class ω_i if

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \qquad \text{for all } j \neq i. \tag{25}$$

Thus, the classifier is viewed as a network or machine that computes c discriminant functions and selects the category corresponding to the largest discriminant. A network representation of a classifier is illustrated in Fig. 2.5.

A Bayes classifier is easily and naturally represented in this way. For the general case with risks, we can let $g_i(\mathbf{x}) = -R(\alpha_i|\mathbf{x})$, because the maximum discriminant

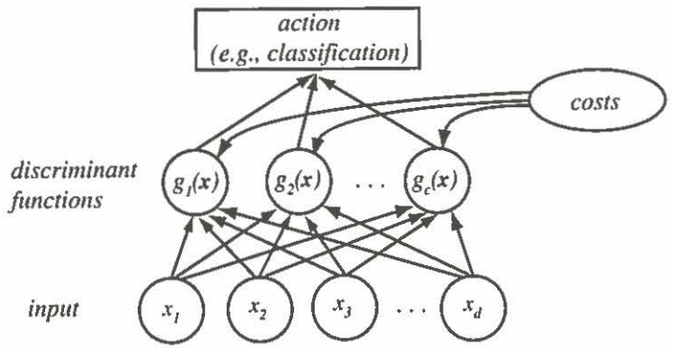


FIGURE 2.5. The functional structure of a general statistical pattern classifier which includes d inputs and c discriminant functions $g_i(\mathbf{x})$. A subsequent step determines which of the discriminant values is the maximum, and categorizes the input pattern accordingly. The arrows show the direction of the flow of information, though frequently the arrows are omitted when the direction of flow is self-evident.

function will then correspond to the minimum conditional risk. For the minimum-error-rate case, we can simplify things further by taking $g_i(\mathbf{x}) = P(\omega_i|\mathbf{x})$, so that the maximum discriminant function corresponds to the maximum posterior probability.

Clearly, the choice of discriminant functions is not unique. We can always multiply all the discriminant functions by the same positive constant or shift them by the same additive constant without influencing the decision. More generally, if we replace every $g_i(\mathbf{x})$ by $f(g_i(\mathbf{x}))$, where $f(\cdot)$ is a monotonically increasing function, the resulting classification is unchanged. This observation can lead to significant analytical and computational simplifications. In particular, for minimum-error-rate classification, any of the following choices gives identical classification results, but some can be much simpler to understand or to compute than others:

$$g_i(\mathbf{x}) = P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}|\omega_j)P(\omega_j)} \tag{26}$$

$$g_i(\mathbf{x}) = p(\mathbf{x}|\omega_i)P(\omega_i) \tag{27}$$

$$g_i(\mathbf{x}) = \ln p(\mathbf{x}|\omega_i) + \ln P(\omega_i), \tag{28}$$

where ln denotes natural logarithm.

Even though the discriminant functions can be written in a variety of forms, the decision rules are equivalent. The effect of any decision rule is to divide the feature space into c *decision regions*, $\mathcal{R}_1, \ldots, \mathcal{R}_c$. If $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$, then $\mathbf{x}$ is in $\mathcal{R}_i$, and the decision rule calls for us to assign $\mathbf{x}$ to ω_i. The regions are separated by *decision boundaries*, surfaces in feature space where ties occur among the largest discriminant functions (Fig. 2.6).

DECISION REGION

2.4.2 The Two-Category Case

DICHOTOMIZER

While the two-category case is just a special instance of the multicategory case, it has traditionally received separate treatment. Indeed, a classifier that places a pattern in one of only two categories has a special name—a *dichotomizer*.* Instead of using

*A classifier for more than two categories is called a polychotomizer.

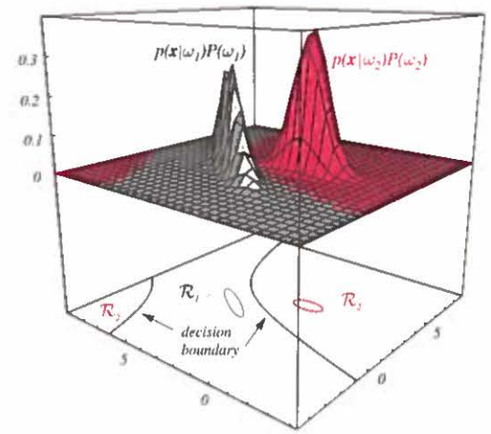


FIGURE 2.6. In this two-dimensional two-category classifier, the probability densities are Gaussian, the decision boundary consists of two hyperbolas, and thus the decision region $\mathcal{R}_2$ is not simply connected. The ellipses mark where the density is $1/e$ times that at the peak of the distribution.

two discriminant functions g_1 and g_2 and assigning $\mathbf{x}$ to ω_1 if $g_1 > g_2$, it is more common to define a single discriminant function

$$g(\mathbf{x}) \equiv g_1(\mathbf{x}) - g_2(\mathbf{x}), \tag{29}$$

and to use the following decision rule: Decide ω_1 if $g(\mathbf{x}) > 0$; otherwise decide ω_2. Thus, a dichotomizer can be viewed as a machine that computes a single discriminant function $g(\mathbf{x})$, and classifies $\mathbf{x}$ according to the algebraic sign of the result. Of the various forms in which the minimum-error-rate discriminant function can be written, the following two (derived from Eqs. 26 and 28) are particularly convenient:

$$g(\mathbf{x}) = P(\omega_1|\mathbf{x}) - P(\omega_2|\mathbf{x}) \tag{30}$$

$$g(\mathbf{x}) = \ln \frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} + \ln \frac{P(\omega_1)}{P(\omega_2)}. \tag{31}$$

2.5 THE NORMAL DENSITY

The structure of a Bayes classifier is determined by the conditional densities $p(\mathbf{x}|\omega_i)$ as well as by the prior probabilities $P(\omega_i)$. Of the various density functions that have been investigated, none has received more attention than the multivariate normal or Gaussian density. To a large extent this attention is due to its analytical tractability. However, the multivariate normal density is also an appropriate model for an important situation, namely, the case where the feature vectors $\mathbf{x}$ for a given class ω_i are continuous-valued, randomly corrupted versions of a single typical or prototype vector $\boldsymbol{\mu}_i$. In this section we provide a brief exposition of the multivariate normal density, focusing on the properties of greatest interest for classification problems.

EXPECTATION

First, recall the definition of the *expected value* of a scalar function $f(x)$, defined for some density $p(x)$:

$$\mathcal{E}[f(x)] \equiv \int_{-\infty}^{\infty} f(x)p(x)\,dx. \tag{32}$$

If the values of the feature x are restricted to points in a discrete set $\mathcal{D}$, we must sum over all samples as

$$\mathcal{E}[f(x)] = \sum_{x \in \mathcal{D}} f(x)P(x), \tag{33}$$

where $P(x)$ is the probability mass at x. We shall occassionally need to calculate expected values—by these and analogous equations defined in higher dimensions (see Appendix Sections A.4.2, A.4.5 and A.4.9).*

2.5.1 Univariate Density

We begin with the continuous univariate normal or Gaussian density,

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right], \tag{34}$$

for which the *expected value* of x (an average, here taken over the feature space) is

$$\mu \equiv \mathcal{E}[x] = \int_{-\infty}^{\infty} xp(x)\,dx, \tag{35}$$

VARIANCE

and where the expected squared deviation or *variance* is

$$\sigma^2 \equiv \mathcal{E}[(x-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 p(x)\,dx. \tag{36}$$

MEAN

The univariate normal density is completely specified by two parameters: its mean μ and variance σ^2. For simplicity, we often abbreviate Eq. 34 by writing $p(x) \sim N(\mu, \sigma^2)$ to say that x is distributed normally with mean μ and variance σ^2. Samples from normal distributions tend to cluster about the mean, with a spread related to the standard deviation σ (Fig. 2.7).

ENTROPY

There is a deep relationship between the normal distribution and *entropy*. We discuss entropy in greater detail in Appendix Section A.7, but for now we merely state that the entropy of a distribution is given by

$$H(p(x)) = -\int p(x)\,\ln p(x)\,dx, \tag{37}$$

NAT
BIT

and measured in *nats*; if a $\log_2$ is used instead, the unit is the *bit*. The entropy measures the fundamental uncertainty in the values of points selected randomly from a

*We will often use somewhat loose engineering terminology and refer to a single point as a "sample." Statisticians, however, always refer to a sample as a *collection* of points, and they discuss "a sample of size n." When taken in context, there are rarely ambiguities in such usage.

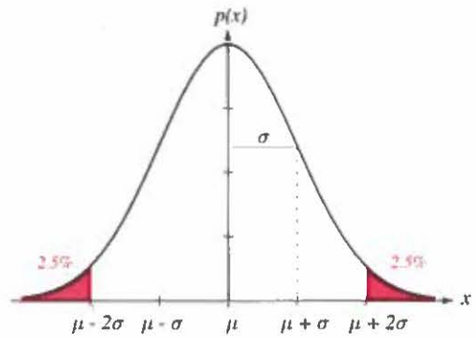


FIGURE 2.7. A univariate normal distribution has roughly 95% of its area in the range $|x-\mu| \leq 2\sigma$, as shown. The peak of the distribution has value $p(\mu) = 1/\sqrt{2\pi}\sigma$.

CENTRAL LIMIT THEOREM

distribution. It can be shown that the normal distribution has the maximum entropy of all distributions having a given mean and variance (Problem 20). Moreover, as stated by the *Central Limit Theorem*, the aggregate effect of the sum of a large number of small, independent random disturbances will lead to a Gaussian distribution (Computer exercise 5). Because many patterns—from fish to handwritten characters to some speech sounds—can be viewed as some ideal or prototype pattern corrupted by a large number of random processes, the Gaussian is often a good model for the actual probability distribution.

2.5.2 Multivariate Density

The general multivariate normal density in d dimensions is written as

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right], \tag{38}$$

COVARIANCE MATRIX
INNER PRODUCT

where $\mathbf{x}$ is a d-component column vector, $\boldsymbol{\mu}$ is the d-component *mean vector*, $\boldsymbol{\Sigma}$ is the d-by-d *covariance matrix*, and $|\boldsymbol{\Sigma}|$ and $\boldsymbol{\Sigma}^{-1}$ are its determinant and inverse, respectively. Further, we let $(\mathbf{x}-\boldsymbol{\mu})^t$ denote the transpose of $\mathbf{x}-\boldsymbol{\mu}$.* Our notation for the *inner product* is

$$\mathbf{a}^t\mathbf{b} = \sum_{i=1}^{d} a_i b_i, \tag{39}$$

which is often called a *dot product*. For simplicity, we often abbreviate Eq. 38 as $p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

Formally, we have

$$\boldsymbol{\mu} \equiv \mathcal{E}[\mathbf{x}] = \int \mathbf{x}p(\mathbf{x})\,d\mathbf{x} \tag{40}$$

*The mathematical expressions for the multivariate normal density are greatly simplified by employing the concepts and notation of linear algebra. Readers who are unsure of our notation or who would like to review linear algebra should see Appendix Section A.2.

and

$$\boldsymbol{\Sigma} \equiv \mathcal{E}[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^t] = \int (\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^t p(\mathbf{x})\, d\mathbf{x}, \tag{41}$$

where the expected value of a vector or a matrix is found by taking the expected values of its components. In other words, if x_i is the ith component of $\mathbf{x}$, μ_i the ith component of $\boldsymbol{\mu}$, and σ_{ij} the ijth component of $\boldsymbol{\Sigma}$, then

$$\mu_i = \mathcal{E}[x_i] \tag{42}$$

and

$$\sigma_{ij} = \mathcal{E}[(x_i-\mu_i)(x_j-\mu_j)]. \tag{43}$$

COVARIANCE

STATISTICAL INDEPENDENCE

The covariance matrix $\boldsymbol{\Sigma}$ is always symmetric and positive semidefinite. We shall restrict our attention to the case in which $\boldsymbol{\Sigma}$ is positive definite, so that the determinant of $\boldsymbol{\Sigma}$ is strictly positive.* The diagonal elements σ_{ii} are the variances of the respective x_i (i.e., σ_i^2), and the off-diagonal elements σ_{ij} are the *covariances* of x_i and x_j. We would expect a positive covariance for the length and weight features of a population of fish, for instance. If x_i and x_j are *statistically independent*, then $\sigma_{ij} = 0$. If all the off-diagonal elements are zero, $p(\mathbf{x})$ reduces to the product of the univariate normal densities for the components of $\mathbf{x}$.

Linear combinations of jointly normally distributed random variables, independent or not, are normally distributed. In particular, if $p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $\mathbf{A}$ is a d-by-k matrix and $\mathbf{y} = \mathbf{A}^t\mathbf{x}$ is a k-component vector, then $p(\mathbf{y}) \sim N(\mathbf{A}^t\boldsymbol{\mu}, \mathbf{A}^t\boldsymbol{\Sigma}\mathbf{A})$, as illustrated in Fig. 2.8. In the special case where $k = 1$ and $\mathbf{A}$ is a unit-length vector $\mathbf{a}$, $y = \mathbf{a}^t\mathbf{x}$ is a scalar that represents the projection of $\mathbf{x}$ onto a line in the direction of $\mathbf{a}$; in that case $\mathbf{a}^t\boldsymbol{\Sigma}\mathbf{a}$ is the variance of the projection of $\mathbf{x}$ onto $\mathbf{a}$. In general then, knowledge of the covariance matrix allows us to calculate the dispersion of the data in any direction, or in any subspace.

It is sometimes convenient to perform a coordinate transformation that converts an arbitrary multivariate normal distribution into a spherical one—that is, one having a covariance matrix proportional to the identity matrix $\mathbf{I}$. If we define $\boldsymbol{\Phi}$ to be the matrix whose columns are the orthonormal eigenvectors of $\boldsymbol{\Sigma}$, and $\boldsymbol{\Lambda}$ the diagonal matrix of the corresponding eigenvalues, then the transformation

$$\mathbf{A}_w = \boldsymbol{\Phi}\boldsymbol{\Lambda}^{-1/2} \tag{44}$$

WHITENING TRANSFORM

applied to the coordinates ensures that the transformed distribution has covariance matrix equal to the identity matrix. In signal processing, $\mathbf{A}_w$ yields a so-called *whitening* transform, because it makes the spectrum of eigenvalues of the transformed distribution uniform.

The multivariate normal density is completely specified by $d + d(d+1)/2$ parameters, namely the elements of the mean vector $\boldsymbol{\mu}$ and the independent elements of the covariance matrix $\boldsymbol{\Sigma}$. Samples drawn from a normal population tend to fall in a single cloud or cluster (Fig. 2.9); the center of the cluster is determined by the mean vector, and the shape of the cluster is determined by the covariance matrix. It

*If sample vectors are drawn from a linear subspace, $|\boldsymbol{\Sigma}| = 0$ and $p(\mathbf{x})$ is degenerate. This occurs, for example, when one component of $\mathbf{x}$ has zero variance, or when two components are identical or multiples of one another.

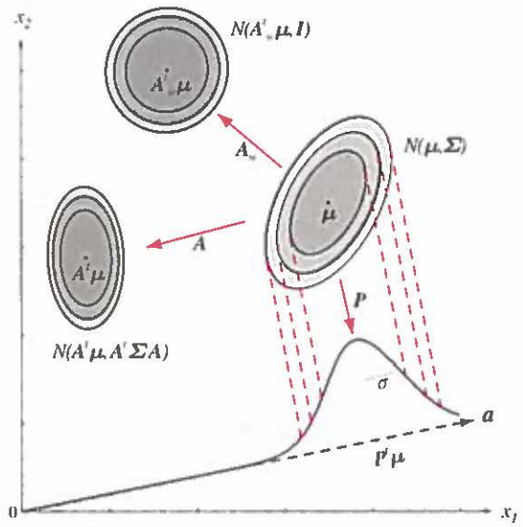


FIGURE 2.8. The action of a linear transformation on the feature space will convert an arbitrary normal distribution into another normal distribution. One transformation, $\mathbf{A}$, takes the source distribution into distribution $N(\mathbf{A}^t\boldsymbol{\mu}, \mathbf{A}^t\boldsymbol{\Sigma}\mathbf{A})$. Another linear transformation—a projection $\mathbf{P}$ onto a line defined by vector $\mathbf{a}$—leads to $N(\mu, \sigma^2)$ measured along that line. While the transforms yield distributions in a different space, we show them superimposed on the original x_1x_2-space. A whitening transform, $\mathbf{A}_w$, leads to a circularly symmetric Gaussian, here shown displaced.

follows from Eq. 38 that the loci of points of constant density are hyperellipsoids for which the quadratic form $(\mathbf{x}-\boldsymbol{\mu})^t\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})$ is constant. The principal axes of these hyperellipsoids are given by the eigenvectors of $\boldsymbol{\Sigma}$ (described by $\boldsymbol{\Phi}$); the eigenvalues (described by $\boldsymbol{\Lambda}$) determine the lengths of these axes. The quantity

$$r^2 = (\mathbf{x}-\boldsymbol{\mu})^t\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) \tag{45}$$

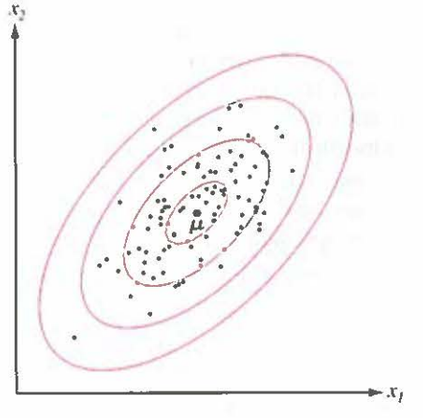


FIGURE 2.9. Samples drawn from a two-dimensional Gaussian lie in a cloud centered on the mean $\boldsymbol{\mu}$. The ellipses show lines of equal probability density of the Gaussian.

MAHALANOBIS DISTANCE

is sometimes called the squared *Mahalanobis distance* from $\mathbf{x}$ to $\boldsymbol{\mu}$. Thus, the contours of constant density are hyperellipsoids of constant Mahalanobis distance to $\boldsymbol{\mu}$ and the volume of these hyperellipsoids measures the scatter of the samples about the mean. It can be shown (Problems 15 and 16) that the volume of the hyperellipsoid corresponding to a Mahalanobis distance r is given by

$$V = V_d |\boldsymbol{\Sigma}|^{1/2} r^d, \tag{46}$$

where V_d is the volume of a d-dimensional unit hypersphere:

$$V_d = \begin{cases} \pi^{d/2}/(d/2)! & d \text{ even} \\ 2^d \pi^{(d-1)/2} (\frac{d-1}{2})!/d! & d \text{ odd.} \end{cases} \tag{47}$$

Thus, for a given dimensionality, the scatter of the samples varies directly with $|\boldsymbol{\Sigma}|^{1/2}$ (Problem 17).

2.6 DISCRIMINANT FUNCTIONS FOR THE NORMAL DENSITY

In Section 2.4.1 we saw that the minimum-error-rate classification can be achieved by use of the discriminant functions

$$g_i(\mathbf{x}) = \ln p(\mathbf{x}|\omega_i) + \ln P(\omega_i). \tag{48}$$

This expression can be readily evaluated if the densities $p(\mathbf{x}|\omega_i)$ are multivariate normal—that is, if $p(\mathbf{x}|\omega_i) \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. In this case, then, from Eq. 38 we have

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i) - \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln |\boldsymbol{\Sigma}_i| + \ln P(\omega_i). \tag{49}$$

Let us examine the discriminant function and resulting classification for a number of special cases.

2.6.1 Case 1: $\boldsymbol{\Sigma}_i = \sigma^2 \mathbf{I}$

The simplest case occurs when the features are statistically independent and when each feature has the same variance, σ^2. In this case the covariance matrix is diagonal, being merely σ^2 times the identity matrix $\mathbf{I}$. Geometrically, this corresponds to the situation in which the samples fall in equal-size hyperspherical clusters, the cluster for the ith class being centered about the mean vector $\boldsymbol{\mu}_i$. The computation of the determinant and the inverse of $\boldsymbol{\Sigma}_i$ is particularly easy: $|\boldsymbol{\Sigma}_i| = \sigma^{2d}$ and $\boldsymbol{\Sigma}_i^{-1} = (1/\sigma^2)\mathbf{I}$. Because both $|\boldsymbol{\Sigma}_i|$ and the $(d/2) \ln 2\pi$ term in Eq. 49 are independent of i, they are unimportant additive constants that can be ignored. Thus we obtain the simple discriminant functions

$$g_i(\mathbf{x}) = -\frac{\|\mathbf{x} - \boldsymbol{\mu}_i\|^2}{2\sigma^2} + \ln P(\omega_i), \tag{50}$$

EUCLIDEAN NORM

where $\|\cdot\|$ denotes the *Euclidean norm*, that is,

$$\|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = (\mathbf{x} - \boldsymbol{\mu}_i)^t (\mathbf{x} - \boldsymbol{\mu}_i). \tag{51}$$

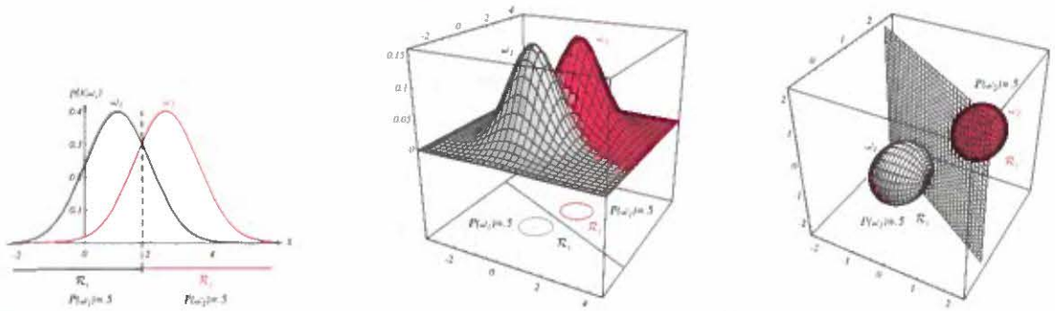


FIGURE 2.10. If the covariance matrices for two distributions are equal and proportional to the identity matrix, then the distributions are spherical in d dimensions, and the boundary is a generalized hyperplane of $d-1$ dimensions, perpendicular to the line separating the means. In these one-, two-, and three-dimensional examples, we indicate $p(\mathbf{x}|\omega_i)$ and the boundaries for the case $P(\omega_1) = P(\omega_2)$. In the three-dimensional case, the grid plane separates $\mathcal{R}_1$ from $\mathcal{R}_2$.

If the prior probabilities are not equal, then Eq. 50 shows that the squared distance $\|\mathbf{x} - \boldsymbol{\mu}\|^2$ must be normalized by the variance σ^2 and offset by adding $\ln P(\omega_i)$; thus, if $\mathbf{x}$ is equally near two different mean vectors, the optimal decision will favor the a priori more likely category.

Regardless of whether the prior probabilities are equal or not, it is not actually necessary to compute distances. Expansion of the quadratic form $(\mathbf{x} - \boldsymbol{\mu}_i)^t (\mathbf{x} - \boldsymbol{\mu}_i)$ yields

$$g_i(\mathbf{x}) = -\frac{1}{2\sigma^2}[\mathbf{x}^t\mathbf{x} - 2\boldsymbol{\mu}_i^t\mathbf{x} + \boldsymbol{\mu}_i^t\boldsymbol{\mu}_i] + \ln P(\omega_i), \tag{52}$$

which appears to be a quadratic function of $\mathbf{x}$. However, the quadratic term $\mathbf{x}^t\mathbf{x}$ is the same for all i, making it an ignorable additive constant. Thus, we obtain the equivalent *linear discriminant functions*

LINEAR DISCRIMINANT

$$g_i(\mathbf{x}) = \mathbf{w}_i^t \mathbf{x} + w_{i0}, \tag{53}$$

where

$$\mathbf{w}_i = \frac{1}{\sigma^2}\boldsymbol{\mu}_i \tag{54}$$

and

$$w_{i0} = \frac{-1}{2\sigma^2}\boldsymbol{\mu}_i^t\boldsymbol{\mu}_i + \ln P(\omega_i). \tag{55}$$

THRESHOLD BIAS

We call w_{i0} the *threshold* or *bias* for the ith category.

LINEAR MACHINE

A classifier that uses linear discriminant functions is called a *linear machine*. This kind of classifier has many interesting theoretical properties, some of which will be discussed in Chapter 5. At this point we merely note that the decision surfaces for a linear machine are pieces of hyperplanes defined by the linear equations $g_i(\mathbf{x}) = g_j(\mathbf{x})$ for the two categories with the highest posterior probabilities. For our particular case, this equation can be written as

$$\mathbf{w}^t(\mathbf{x} - \mathbf{x}_0) = 0, \tag{56}$$

where

$$\mathbf{w} = \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \tag{57}$$

and

$$\mathbf{x}_0 = \frac{1}{2}(\boldsymbol{\mu}_i + \boldsymbol{\mu}_j) - \frac{\sigma^2}{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2} \ln \frac{P(\omega_i)}{P(\omega_j)}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j). \tag{58}$$

These equations define a hyperplane through the point $\mathbf{x}_0$ and orthogonal to the vector $\mathbf{w}$. Because $\mathbf{w} = \boldsymbol{\mu}_i - \boldsymbol{\mu}_j$, the hyperplane separating $\mathcal{R}_i$ and $\mathcal{R}_j$ is orthogonal to the line linking the means. If $P(\omega_i) = P(\omega_j)$, the second term on the right of Eq. 58 vanishes, and thus the point $\mathbf{x}_0$ is halfway between the means, and the hyperplane is the perpendicular bisector of the line between the means (Fig. 2.11). If $P(\omega_i) \neq P(\omega_j)$, the point $\mathbf{x}_0$ shifts away from the more likely mean. Note, however, that if the variance σ^2 is small relative to the squared distance $\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2$, then the position of the decision boundary is relatively insensitive to the exact values of the prior probabilities.

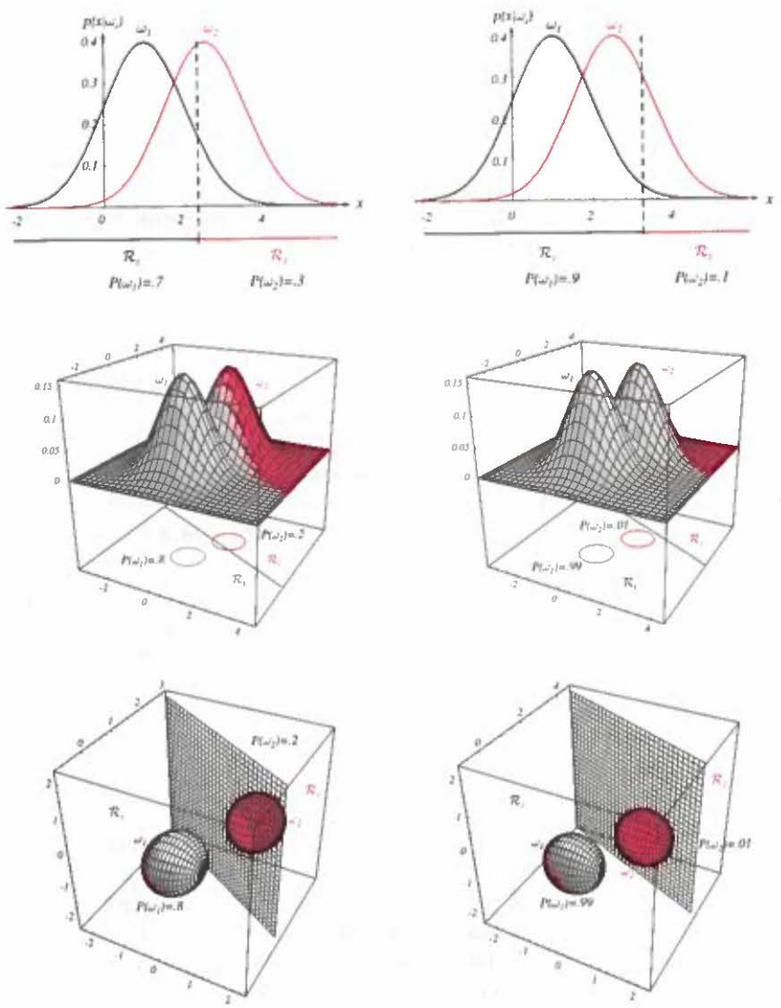


FIGURE 2.11. As the priors are changed, the decision boundary shifts; for sufficiently disparate priors the boundary will not lie between the means of these one-, two- and three-dimensional spherical Gaussian distributions.

If the prior probabilities $P(\omega_i)$ are the same for all c classes, then the ln $P(\omega_i)$ term becomes another unimportant additive constant that can be ignored. When this happens, the optimum decision rule can be stated very simply: To classify a feature vector $\mathbf{x}$, measure the Euclidean distance $\|\mathbf{x} - \boldsymbol{\mu}_i\|$ from each $\mathbf{x}$ to each of the c mean vectors, and assign $\mathbf{x}$ to the category of the nearest mean. Such a classifier is called a *minimum-distance classifier*. If each mean vector is thought of as being an ideal prototype or template for patterns in its class, then this is essentially a *template-matching* procedure (Fig. 2.10), a technique we will consider again in Chapter 4 on the nearest-neighbor algorithm.

MINIMUM-DISTANCE CLASSIFIER

TEMPLATE-MATCHING

2.6.2 Case 2: $\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}$

Another simple case arises when the covariance matrices for all of the classes are identical but otherwise arbitrary. Geometrically, this corresponds to the situation in which the samples fall in hyperellipsoidal clusters of equal size and shape, the cluster for the ith class being centered about the mean vector $\boldsymbol{\mu}_i$. Because both $|\boldsymbol{\Sigma}_i|$ and the $(d/2) \ln 2\pi$ term in Eq. 49 are independent of i, they can be ignored as superfluous additive constants. This simplification leads to the discriminant functions

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) + \ln P(\omega_i). \tag{59}$$

If the prior probabilities $P(\omega_i)$ are the same for all c classes, then the ln $P(\omega_i)$ term can be ignored. In this case, the optimal decision rule can once again be stated very simply: To classify a feature vector $\mathbf{x}$, measure the squared Mahalanobis distance $(\mathbf{x} - \boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_i)$ from $\mathbf{x}$ to each of the c mean vectors, and assign $\mathbf{x}$ to the category of the nearest mean. As before, unequal prior probabilities bias the decision in favor of the a priori more likely category.

Expansion of the quadratic form $(\mathbf{x} - \boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_i)$ results in a sum involving a quadratic term $\mathbf{x}^t \boldsymbol{\Sigma}^{-1} \mathbf{x}$ which here is independent of i. After this term is dropped from Eq. 59, the resulting discriminant functions are again linear:

$$g_i(\mathbf{x}) = \mathbf{w}_i^t \mathbf{x} + w_{i0}, \tag{60}$$

where

$$\mathbf{w}_i = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_i \tag{61}$$

and

$$w_{i0} = -\frac{1}{2} \boldsymbol{\mu}_i^t \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_i + \ln P(\omega_i). \tag{62}$$

Because the discriminants are linear, the resulting decision boundaries are again hyperplanes (Fig. 2.10). If $\mathcal{R}_i$ and $\mathcal{R}_j$ are contiguous, the boundary between them

has the equation

$$\mathbf{w}^t(\mathbf{x} - \mathbf{x}_0) = 0, \tag{63}$$

where

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) \tag{64}$$

and

$$\mathbf{x}_0 = \frac{1}{2}(\boldsymbol{\mu}_i + \boldsymbol{\mu}_j) - \frac{\ln[P(\omega_i)/P(\omega_j)]}{(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^t\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j). \tag{65}$$

Because $\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$ is generally not in the direction of $\boldsymbol{\mu}_i - \boldsymbol{\mu}_j$, the hyperplane separating $\mathcal{R}_i$ and $\mathcal{R}_j$ is generally not orthogonal to the line between the means. However, it does intersect that line at the point $\mathbf{x}_0$; if the prior probabilities are equal then $\mathbf{x}_0$ is halfway between the means. If the prior probabilities are not equal, the optimal boundary hyperplane is shifted away from the more likely mean (Fig. 2.12). As before, with sufficient bias the decision plane need not lie between the two mean vectors.

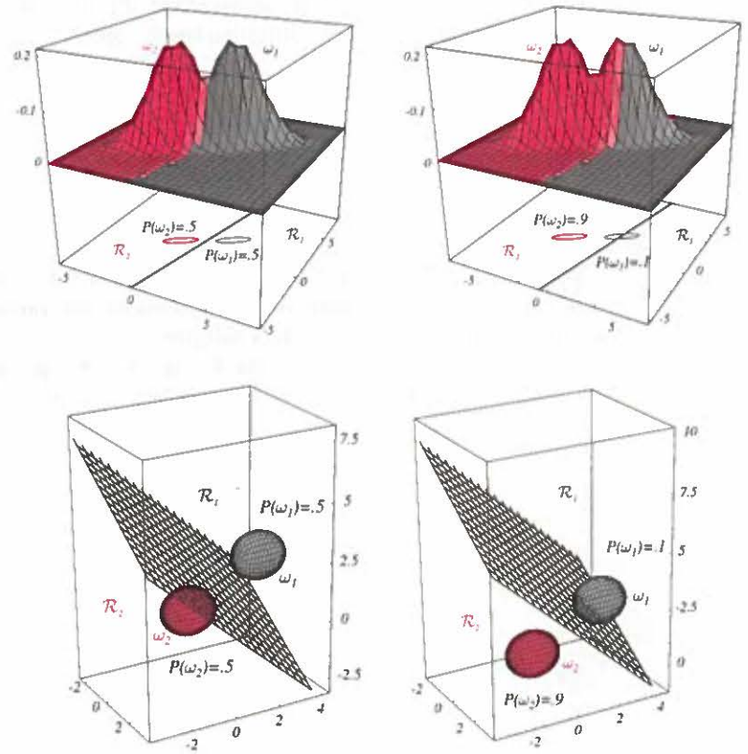


FIGURE 2.12. Probability densities (indicated by the surfaces in two dimensions and ellipsoidal surfaces in three dimensions) and decision regions for equal but asymmetric Gaussian distributions. The decision hyperplanes need not be perpendicular to the line connecting the means.

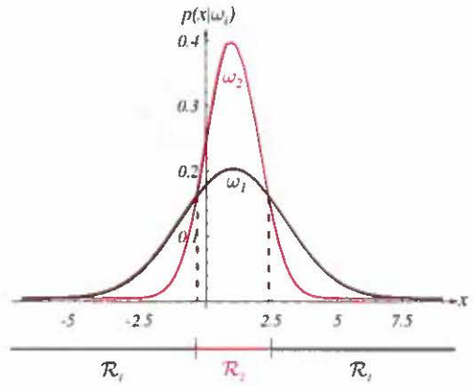


FIGURE 2.13. Non-simply connected decision regions can arise in one dimensions for Gaussians having unequal variance.

2.6.3 Case 3: $\boldsymbol{\Sigma}_i$ = arbitrary

In the general multivariate normal case, the covariance matrices are different for each category. The only term that can be dropped from Eq. 49 is the $(d/2)\ \ln 2\pi$ term, and the resulting discriminant functions are inherently quadratic:

$$g_i(\mathbf{x}) = \mathbf{x}^t\mathbf{W}_i\mathbf{x} + \mathbf{w}_i^t\mathbf{x} + w_{i0}, \tag{66}$$

where

$$\mathbf{W}_i = -\frac{1}{2}\boldsymbol{\Sigma}_i^{-1}, \tag{67}$$

$$\mathbf{w}_i = \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i \tag{68}$$

and

$$w_{i0} = -\frac{1}{2}\boldsymbol{\mu}_i^t\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| + \ln P(\omega_i). \tag{69}$$

HYPERQUADRIC

In the two-category case, the decision surfaces are *hyperquadrics*, and they can assume any of the general forms: hyperplanes, pairs of hyperplanes, hyperspheres, hyperellipsoids, hyperparaboloids, and hyperhyperboloids of various types (Problem 30). Even in one dimension, for arbitrary variance the decision regions need not be simply connected (Fig. 2.13). The two- and three-dimensional examples in Figs. 2.14 and 2.15 indicate how these different forms can arise.

The extension of these results to more than two categories is straightforward though here we need to keep clear which two of the total c categories are responsible for any boundary segment. Figure 2.16 shows the decision surfaces for a four-category case made up of Gaussian distributions. Of course, if the distributions are more complicated, the decision regions can be even more complex, though the same underlying theory holds there too.

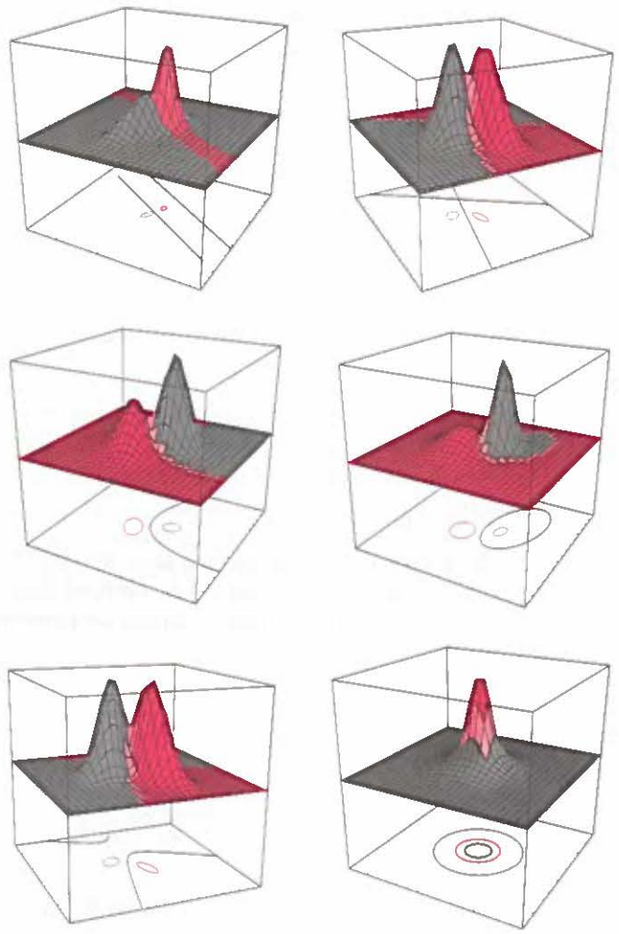


FIGURE 2.14. Arbitrary Gaussian distributions lead to Bayes decision boundaries that are general hyperquadrics. Conversely, given any hyperquadric, one can find two Gaussian distributions whose Bayes decision boundary is that hyperquadric. These variances are indicated by the contours of constant probability density.

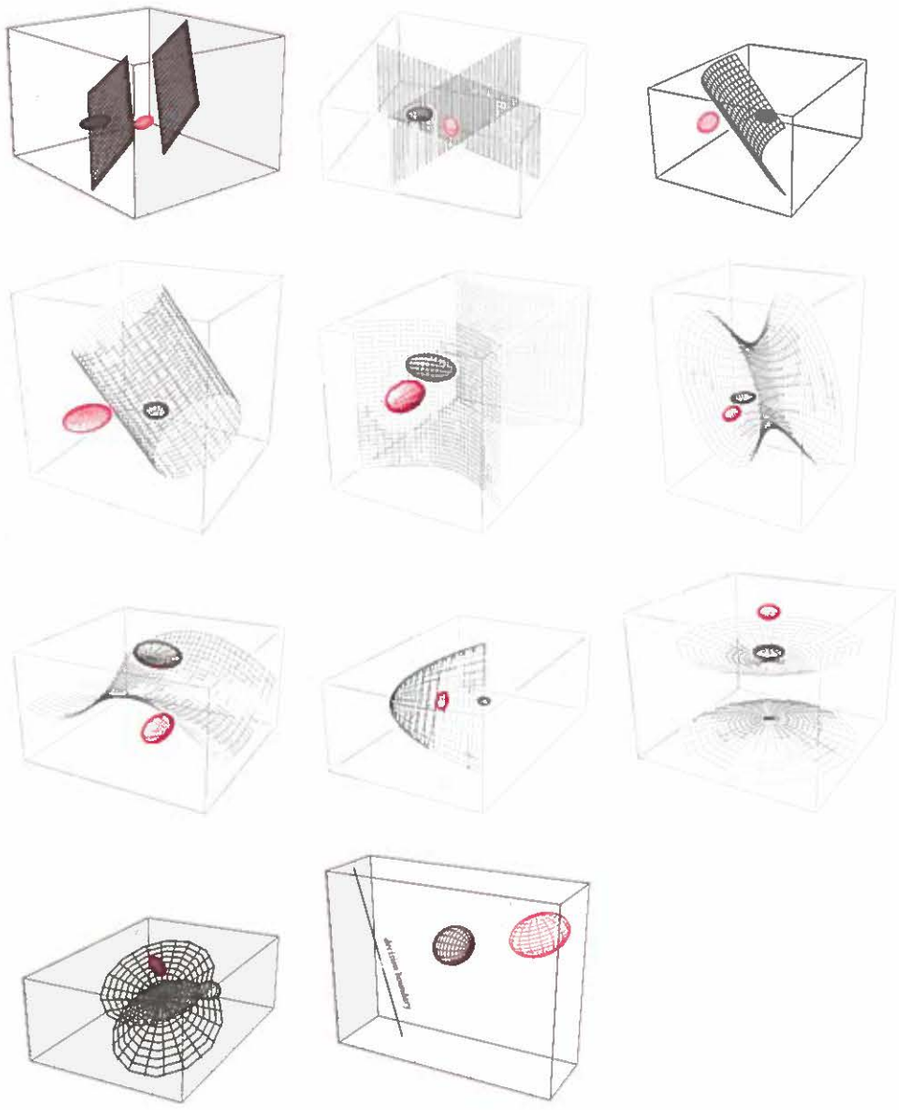
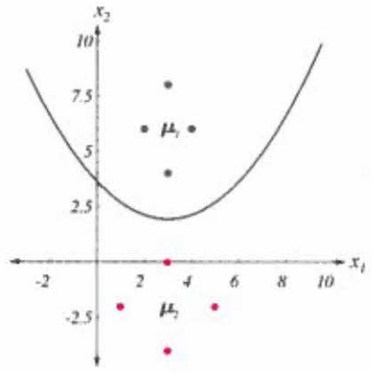


FIGURE 2.15. Arbitrary three-dimensional Gaussian distributions yield Bayes decision boundaries that are two-dimensional hyperquadrics. There are even degenerate cases in which the decision boundary is a line.

EXAMPLE 1 Decision Regions for Two-Dimensional Gaussian Data

To clarify these ideas, we explicitly calculate the decision boundary for the two-category two-dimensional data in the Example figure.



The computed Bayes decision boundary for two Gaussian distributions, each based on four data points.

Let ω_1 be the set of the four black points, and ω_2 the red points. Although we will spend much of the next chapter understanding how to estimate the parameters of our distributions, for now we simply assume that we need merely calculate the means and covariances by the discrete versions of Eqs. 40 and 41; they are found to be:

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 3 \\ 6 \end{bmatrix}; \quad \boldsymbol{\Sigma}_1 = \begin{pmatrix} 1/2 & 0 \\ 0 & 2 \end{pmatrix} \text{ and } \boldsymbol{\mu}_2 = \begin{bmatrix} 3 \\ -2 \end{bmatrix}; \quad \boldsymbol{\Sigma}_2 = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}.$$

The inverse matrices are then,

$$\boldsymbol{\Sigma}_1^{-1} = \begin{pmatrix} 2 & 0 \\ 0 & 1/2 \end{pmatrix} \quad \text{and} \quad \boldsymbol{\Sigma}_2^{-1} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}.$$

We assume equal prior probabilities, $P(\omega_1) = P(\omega_2) = 0.5$, and substitute these into the general form for a discriminant, Eqs. 66–69, setting $g_1(\mathbf{x}) = g_2(\mathbf{x})$ to obtain the decision boundary:

$$\mathbf{x}_2 = 3.514 - 1.125\mathbf{x}_1 + 0.1875\mathbf{x}_1^2.$$

This equation describes a parabola with vertex at $\binom{3}{1.83}$. Note that despite the fact that the variance in the data along the $\mathbf{x}_2$ direction for both distributions is the same, the decision boundary does not pass through the point $\binom{3}{2}$, midway between the means, as we might have naively guessed. This is because for the ω_1 distribution, the probability distribution is "squeezed" in the $\mathbf{x}_1$-direction more so than for the ω_2 distribution. The ω_1 distribution is increased along the $\mathbf{x}_2$ direction (relative to that for the ω_2 distribution). Thus the decision boundary lies slightly lower than the point midway between the two means, as can be seen in the decision boundary.

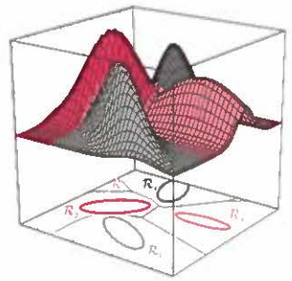


FIGURE 2.16. The decision regions for four normal distributions. Even with such a low number of categories, the shapes of the boundary regions can be rather complex.

*2.7 ERROR PROBABILITIES AND INTEGRALS

We can obtain additional insight into the operation of a general classifier—Bayes or otherwise—if we consider the sources of its error. Consider first the two-category case, and suppose the dichotomizer has divided the space into two regions $\mathcal{R}_1$ and $\mathcal{R}_2$ in a possibly nonoptimal way. There are two ways in which a classification error can occur; either an observation $\mathbf{x}$ falls in $\mathcal{R}_2$ and the true state of nature is ω_1, or $\mathbf{x}$ falls in $\mathcal{R}_1$ and the true state of nature is ω_2. Because these events are mutually exclusive and exhaustive, the probability of error is

$$\begin{aligned} P(error) &= P(\mathbf{x} \in \mathcal{R}_2, \omega_1) + P(\mathbf{x} \in \mathcal{R}_1, \omega_2) \\ &= P(\mathbf{x} \in \mathcal{R}_2|\omega_1)P(\omega_1) + P(\mathbf{x} \in \mathcal{R}_1|\omega_2)P(\omega_2) \\ &= \int_{\mathcal{R}_2} p(\mathbf{x}|\omega_1)P(\omega_1)\,d\mathbf{x} + \int_{\mathcal{R}_1} p(\mathbf{x}|\omega_2)P(\omega_2)\,d\mathbf{x}. \end{aligned} \tag{70}$$

This result is illustrated in the one-dimensional case in Fig. 2.17. The two integrals in Eq. 70 represent the pink and the gray areas in the tails of the functions $p(\mathbf{x}|\omega_i)P(\omega_i)$. Because the decision point x^* (and hence the regions $\mathcal{R}_1$ and $\mathcal{R}_2$) were chosen arbitrarily for that figure, the probability of error is not as small as it might be. In particular, the triangular area marked "reducible error" can be eliminated if the decision boundary is moved to x_B. This is the Bayes optimal decision boundary and gives the lowest probability of error. In general, if $p(\mathbf{x}|\omega_1)P(\omega_1) > p(\mathbf{x}|\omega_2)P(\omega_2)$, it is advantageous to classify $\mathbf{x}$ as in $\mathcal{R}_1$ so that the smaller quantity will contribute to the error integral; this is exactly what the Bayes decision rule achieves.

In the multicategory case, there are more ways to be wrong than to be right, and it is simpler to compute the probability of being correct. Clearly,

$$\begin{aligned} P(correct) &= \sum_{i=1}^{c} P(\mathbf{x} \in \mathcal{R}_i, \omega_i) \\ &= \sum_{i=1}^{c} P(\mathbf{x} \in \mathcal{R}_i|\omega_i)P(\omega_i) \\ &= \sum_{i=1}^{c} \int_{\mathcal{R}_i} p(\mathbf{x}|\omega_i)P(\omega_i)\,d\mathbf{x}. \end{aligned} \tag{71}$$

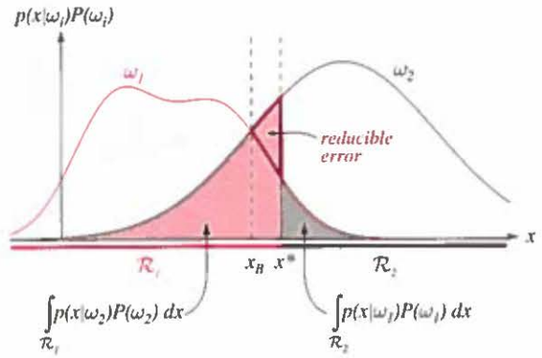


FIGURE 2.17. Components of the probability of error for equal priors and (nonoptimal) decision point x^*. The pink area corresponds to the probability of errors for deciding ω_1 when the state of nature is in fact ω_2; the gray area represents the converse, as given in Eq. 70. If the decision boundary is instead at the point of equal posterior probabilities, x_B, then this reducible error is eliminated and the total shaded area is the minimum possible; this is the Bayes decision and gives the Bayes error rate.

The general result of Eq. 71 depends neither on how the feature space is partitioned into decision regions nor on the form of the underlying distributions. The Bayes classifier maximizes this probability by choosing the regions so that the integrand is maximal for all $\mathbf{x}$; no other partitioning can yield a smaller probability of error.

*2.8 ERROR BOUNDS FOR NORMAL DENSITIES

The Bayes decision rule guarantees the lowest average error rate, and we have seen how to calculate the decision boundaries for normal densities. However, these results do not tell us what the probability of error actually *is*. The full calculation of the error for the Gaussian case would be quite difficult, especially in high dimensions, because of the discontinuous nature of the decision regions in the integral in Eq. 71. However, in the two-category case the general error integral of Eq. 5 can be approximated analytically to give us an upper bound on the error.

2.8.1 Chernoff Bound

To derive a bound for the error, we need the following inequality:

$$\min[a, b] \leq a^{\beta} b^{1-\beta} \quad \text{for } a, b \geq 0 \text{ and } 0 \leq \beta \leq 1. \tag{72}$$

To understand this inequality we can, without loss of generality, assume $a \geq b$. Thus we need only show that $b \leq a^{\beta} b^{1-\beta} = (a/b)^{\beta} b$. But this inequality is manifestly valid, because $(a/b)^{\beta} \geq 1$. Using Eqs. 7 and 1, we apply this inequality to the vector form of Eq. 5 and get the bound:

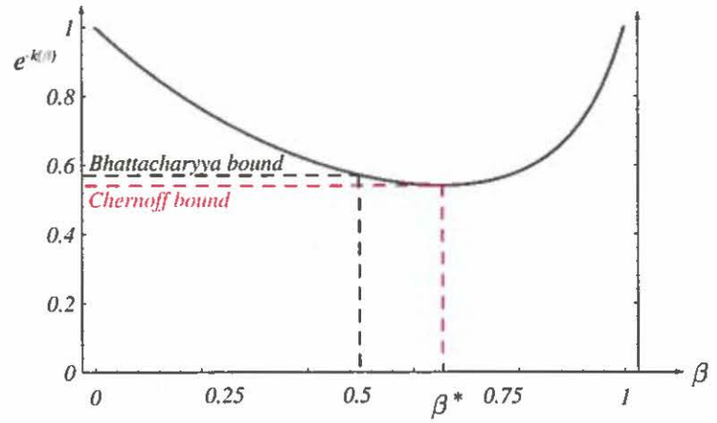


FIGURE 2.18. The Chernoff error bound is never looser than the Bhattacharyya bound. For this example, the Chernoff bound happens to be at $\beta^* = 0.66$, and is slightly tighter than the Bhattacharyya bound ($\beta = 0.5$).

$$P(error) \leq P^{\beta}(\omega_1) P^{1-\beta}(\omega_2) \int p^{\beta}(\mathbf{x}|\omega_1) p^{1-\beta}(\mathbf{x}|\omega_2)\, d\mathbf{x} \qquad \text{for} 0 \leq \beta \leq 1. \tag{73}$$

Note especially that this integral is over *all* feature space—we do not need to impose integration limits corresponding to decision boundaries.

If the conditional probabilities are normal, the integral in Eq. 73 can be evaluated analytically (Problem 36), yielding

$$\int p^{\beta}(\mathbf{x}|\omega_1) p^{1-\beta}(\mathbf{x}|\omega_2)\, d\mathbf{x} = e^{-k(\beta)} \tag{74}$$

where

$$k(\beta) = \frac{\beta(1-\beta)}{2} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^t [\beta \boldsymbol{\Sigma}_1 + (1-\beta)\boldsymbol{\Sigma}_2]^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) + \frac{1}{2} \ln \frac{|\beta \boldsymbol{\Sigma}_1 + (1-\beta)\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|^{\beta} |\boldsymbol{\Sigma}_2|^{1-\beta}}. \tag{75}$$

The graph in Fig. 2.18 shows a typical example of how $e^{-k(\beta)}$ varies with β. The *Chernoff bound* on $P(error)$ is found by analytically or numerically finding the value of β that minimizes $e^{-k(\beta)}$ and then substituting the results in Eq. 73. The key benefit here is that this optimization is in the one-dimensional β space, despite the fact that the distributions themselves might be in a space of arbitrarily high dimension.

2.8.2 Bhattacharyya Bound

The general dependence of the Chernoff bound upon β shown in Fig. 2.18 is typical of a wide range of problems; The bound is loose for extreme values (i.e., $\beta \to 1$ and $\beta \to 0$), and it is tighter for intermediate ones. While the precise value of the optimal β depends upon the parameters of the distributions and the prior probabilities, a computationally simpler but slightly less tight bound can be derived by simply setting $\beta = 1/2$. This result is the so-called *Bhattacharyya bound* on the error, where Eq. 73 then has the form

$$P(error) \leq \sqrt{P(\omega_1)P(\omega_2)} \int \sqrt{p(\mathbf{x}|\omega_1)p(\mathbf{x}|\omega_2)}\, d\mathbf{x}$$
$$= \sqrt{P(\omega_1)P(\omega_2)}e^{-k(1/2)}, \tag{76}$$

where by Eq. 75 we have for the Gaussian case:

$$k(1/2) = 1/8(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^t \left[\frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2}\right]^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) + \frac{1}{2} \ln \frac{\left|\frac{\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2}{2}\right|}{\sqrt{|\boldsymbol{\Sigma}_1||\boldsymbol{\Sigma}_2|}}. \tag{77}$$

The Chernoff and Bhatacharyya bounds may still be used even if the underlying distributions are not Gaussian. However, for distributions that deviate markedly from a Gaussian, the bounds will not be informative (Problem 34).

EXAMPLE 2 Error Bounds for Gaussian Distributions

It is a straightforward matter to calculate the Bhattacharyya bound for the two-dimensional data sets of Example 1. Substituting the means and covariances of Example 1 into Eq. 77, we find $k(1/2) = 4.11$, and thus by Eqs. 76 and 77 the Bhattacharyya bound on the error is $P(error) \leq 0.016382$.

A tighter bound on the error can be approximated by searching numerically for the Chernoff bound of Eq. 75, which for this problem gives 0.016380. Numerical integration of Eq. 5 gives an error rate of 0.0021; thus the bounds here are not particularly tight. Such numerical integration is often impractical for Gaussians in higher than two or three dimensions.

2.8.3 Signal Detection Theory and Operating Characteristics

Another measure of distance between two Gaussian distributions has found great use in experimental psychology, radar detection and other fields. Suppose we are interested in detecting a single weak pulse, such as a dim flash of light or a weak radar reflection. Our model is, then, that at some point in the detector there is an internal signal (such as a voltage) x, whose value has mean μ_2 when the external signal (pulse) is present, and mean μ_1 when it is not present. Because of random noise—within and outside the detector itself—the actual value is a random variable. We assume the distributions are normal with different means but the same variance—that is, $p(x|\omega_i) \sim N(\mu_i, \sigma^2)$—as shown in Fig. 2.19.

DISCRIMINABILITY

The detector (classifier) employs a threshold value x^* for determining whether the external pulse is present, but suppose we, as experimenters, do not have access to this value (nor to the means and standard deviations of the distributions). We seek to find some measure of the ease of discriminating whether the pulse is present or not, in a form independent of the choice of x^*. Such a measure is the *discriminability*, which describes the inherent and unchangeable properties due to noise and the strength of the external signal, but not on the decision strategy (i.e., the actual choice of x^*). This discriminability is defined as

$$d' = \frac{|\mu_2 - \mu_1|}{\sigma}. \tag{78}$$

A high d' is of course desirable.

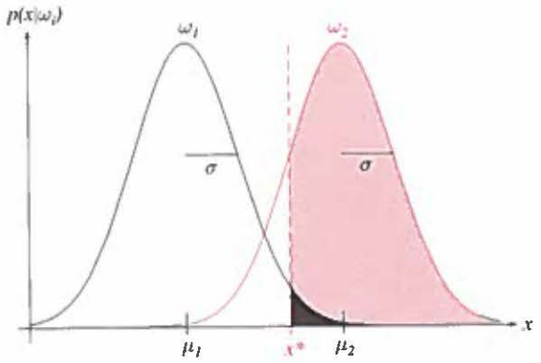


FIGURE 2.19. During any instant when no external pulse is present, the probability density for an internal signal is normal, that is, $p(x|\omega_1) \sim N(\mu_1, \sigma^2)$; when the external signal is present, the density is $p(x|\omega_2) \sim N(\mu_2, \sigma^2)$. Any decision threshold x^* will determine the probability of a hit (the pink area under the ω_2 curve, above x^*) and of a false alarm (the black area under the ω_1 curve, above x^*).

While we do not know μ_1, μ_2, σ or x^*, we assume here that we know the state of nature and the decision of the system. Such information allows us to find d'. To this end, we consider the following four probabilities:

- $P(x > x^*|x \in \omega_2)$: a *hit*—the probability that the internal signal is above x^* given that the external signal is present
- $P(x > x^*|x \in \omega_1)$: a *false alarm*—the probability that the internal signal is above x^* despite there being no external signal present
- $P(x < x^*|x \in \omega_2)$: a *miss*—the probability that the internal signal is below x^* given that the external signal is present
- $P(x < x^*|x \in \omega_1)$: a *correct rejection*—the probability that the internal signal is below x^* given that the external signal is not present.

If we have a large number of trials (and we can assume x^* is fixed, albeit at an unknown value), we can determine these probabilities experimentally, in particular the hit and false alarm rates. We plot a point representing these rates on a two-dimensional graph. If the densities are fixed but the threshold x^* is changed, then our hit and false alarm rates will also change. Thus we see that for a given discriminability d', our point will move along a smooth curve—a *receiver operating characteristic* or ROC curve (Fig. 2.20).

RECEIVER OPERATING CHARACTERISTIC

The great benefit of this signal detection framework is that we can distinguish operationally between *discriminability* and *decision bias*: While the former is an inherent property of the detector system, the latter is due to the receiver's implied but changeable loss matrix. Through any pair of hit and false alarm rates passes one and only one ROC curve; thus, so long as neither rate is exactly 0 or 1, we can determine the discriminability from these rates (Problem 39). Moreover, if the Gaussian assumption holds, a determination of the discriminability (from an arbitrary x^*) allows us to calculate the Bayes error rate—the most important property of any classifier. If the actual error rate differs from the Bayes rate inferred in this way, we should alter the threshold x^* accordingly.

It is a simple matter to generalize the above discussion and apply it to two categories having arbitrary multidimensional distributions, Gaussian or not. Suppose we

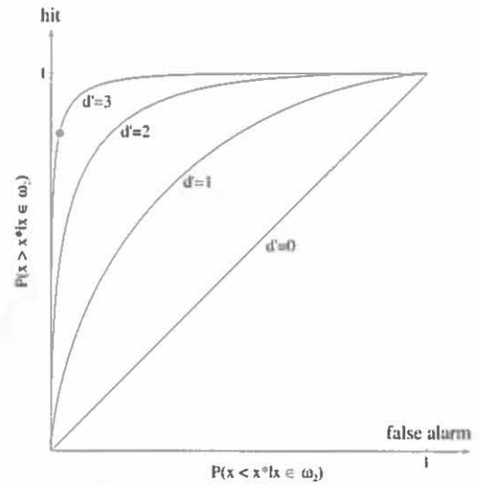


FIGURE 2.20. In a receiver operating characteristic (ROC) curve, the abscissa is the probability of false alarm, $P(x > x^*|x \in \omega_1)$, and the ordinate is the probability of hit, $P(x > x^*|x \in \omega_2)$. From the measured hit and false alarm rates (here corresponding to x^* in Fig. 2.19 and shown as the red dot), we can deduce that $d' = 3$.

have two distributions $p(\mathbf{x}|\omega_1)$ and $p(\mathbf{x}|\omega_2)$ which overlap, and thus have nonzero Bayes classification error. Just as we saw above, any pattern actually from ω_2 could be properly classified as ω_2 (a "hit") or misclassified as ω_1 (a "false alarm"). Unlike the one-dimensional case above, however, there may be *many* decision boundaries that correspond to a particular hit rate, each with a different false alarm rate. Clearly here we cannot determine a fundamental measure of discriminability without knowing more about the underlying decision rule than just the hit and false alarm rates.

In a rarely attainable ideal, we can imagine that our measured hit and false alarm rates are *optimal*—for example, that of all the decision rules giving the measured hit rate, the rule that is actually used is the one having the minimum false alarm rate. If we constructed a multidimensional classifier—regardless of the distributions used—we might try to characterize the problem in this way, though it would probably require great computational resources to search for such optimal hit and false alarm rates.

OPERATING CHARACTERISTIC

In practice, instead we forgo optimality, and simply vary a single control parameter for the decision rule and plot the resulting hit and false alarm rates—a curve called merely an *operating characteristic*. It is traditional to use a control parameter that can yield, at extreme values, either a vanishing false alarm or a vanishing hit rate, just as can be achieved with a very large or a very small x^* in an ROC curve. We should note that since the distributions can be arbitrary, the operating characteristic need not be symmetric (Fig. 2.21); in rare cases it need not even be concave down at all points.

Classifier operating curves are of value for problems where the loss matrix λ_{ij} might be changed. If the operating characteristic has been determined as a function of the control parameter ahead of time, it is a simple matter, when faced with a new loss function, to deduce the control parameter setting that will minimize the expected risk (Problem 39).

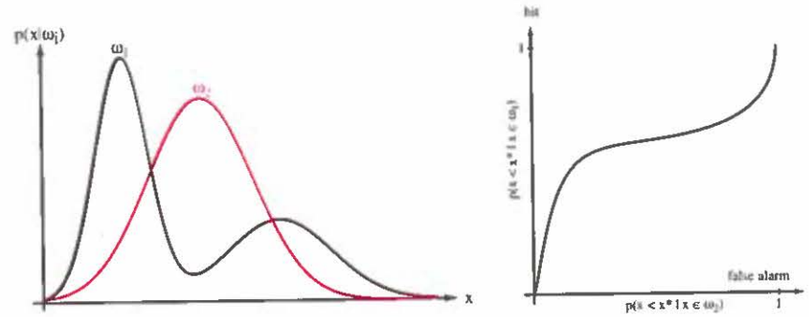


FIGURE 2.21. In a general operating characteristic curve, the abscissa is the probability of false alarm, $P(x \in \mathcal{R}_2|x \in \omega_1)$, and the ordinate is the probability of hit, $P(x \in \mathcal{R}_2|x \in \omega_2)$. As illustrated here, operating characteristic curves are generally not symmetric, as shown at the right.

2.9 BAYES DECISION THEORY—DISCRETE FEATURES

Until now we have assumed that the feature vector $\mathbf{x}$ could be any point in a d-dimensional Euclidean space, $\mathbf{R}^d$. However, in many practical applications the components of $\mathbf{x}$ are binary-, ternary-, or higher-integer-valued, so that $\mathbf{x}$ can assume only one of m discrete values $\mathbf{v}_1, \ldots, \mathbf{v}_m$. In such cases, the probability density function $p(\mathbf{x}|\omega_j)$ becomes singular; integrals of the form

$$\int p(\mathbf{x}|\omega_j)\, d\mathbf{x} \tag{79}$$

must then be replaced by corresponding sums, such as

$$\sum_{\mathbf{x}} P(\mathbf{x}|\omega_j), \tag{80}$$

where we understand that the summation is over all values of $\mathbf{x}$ in the discrete distribution.* Bayes formula then involves probabilities, rather than probability densities:

$$P(\omega_j|\mathbf{x}) = \frac{P(\mathbf{x}|\omega_j)P(\omega_j)}{P(\mathbf{x})}, \tag{81}$$

where

$$P(\mathbf{x}) = \sum_{j=1}^{c} P(\mathbf{x}|\omega_j)P(\omega_j). \tag{82}$$

The definition of the conditional risk $R(\alpha|\mathbf{x})$ is unchanged, and the fundamental Bayes decision rule remains the same: To minimize the overall risk, select the action α_i for which $R(\alpha_i|\mathbf{x})$ is minimum, or stated formally,

$$\alpha^* = \arg\min_i R(\alpha_i|\mathbf{x}). \tag{83}$$

*Technically speaking, Eq. 80 should be written as $\sum_k P(\mathbf{v}_k|\omega_j)$ where $P(\mathbf{v}_k|\omega_j)$ is the conditional probability that $\mathbf{x} = \mathbf{v}_k$, given that the state of nature is ω_j.

The basic rule to minimize the error rate by maximizing the posterior probability is also unchanged as are the discriminant functions of Eqs. 26–28, given the obvious replacement of densities $p(\cdot)$ by probabilities $P(\cdot)$.

2.9.1 Independent Binary Features

As an example of a classification involving discrete features, consider the two-category problem in which the components of the feature vector are binary-valued and conditionally independent. To be more specific we let $\mathbf{x} = (x_1, \ldots, x_d)^t$, where the components x_i are either 0 or 1, with probabilities

$$p_i = \Pr[x_i = 1|\omega_1] \tag{84}$$

and

$$q_i = \Pr[x_i = 1|\omega_2]. \tag{85}$$

This is a model of a classification problem in which each feature gives us a yes/no answer about the pattern. If $p_i > q_i$, we expect the ith feature to give a "yes" answer more frequently when the state of nature is ω_1 than when when it is ω_2. (As an example, consider two factories each making the same automobile, each of whose d components could be functional or defective. If it were known how the factories differed in their reliabilities for making each component, then this model could be used to judge which factory manufactured a given automobile based on the knowledge of which features are functional and which defective.) By assuming conditional independence we can write $P(\mathbf{x}|\omega_i)$ as the product of the probabilities for the components of $\mathbf{x}$. Given this assumption, a particularly convenient way of writing the class-conditional probabilities is as follows:

$$P(\mathbf{x}|\omega_1) = \prod_{i=1}^{d} p_i^{x_i}(1 - p_i)^{1-x_i} \tag{86}$$

and

$$P(\mathbf{x}|\omega_2) = \prod_{i=1}^{d} q_i^{x_i}(1 - q_i)^{1-x_i}. \tag{87}$$

Then the likelihood ratio is given by

$$\frac{P(\mathbf{x}|\omega_1)}{P(\mathbf{x}|\omega_2)} = \prod_{i=1}^{d} \left(\frac{p_i}{q_i}\right)^{x_i} \left(\frac{1 - p_i}{1 - q_i}\right)^{1-x_i} \tag{88}$$

and consequently Eq. 31 yields the discriminant function

$$g(\mathbf{x}) = \sum_{i=1}^{d} \left[x_i \ln \frac{p_i}{q_i} + (1 - x_i) \ln \frac{1 - p_i}{1 - q_i}\right] + \ln \frac{P(\omega_1)}{P(\omega_2)}. \tag{89}$$

We note especially that this discriminant function is linear in the x_i and thus we can write

$$g(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i + w_0, \tag{90}$$

where

$$w_i = \ln \frac{p_i(1 - q_i)}{q_i(1 - p_i)} \qquad i = 1, \ldots, d \tag{91}$$

and

$$w_0 = \sum_{i=1}^{d} \ln \frac{1 - p_i}{1 - q_i} + \ln \frac{P(\omega_1)}{P(\omega_2)}. \tag{92}$$

Let us examine these results to see what insight they can give. Recall first that we decide ω_1 if $g(\mathbf{x}) > 0$ and ω_2 if $g(\mathbf{x}) \leq 0$. We have seen that $g(\mathbf{x})$ is a weighted combination of the components of $\mathbf{x}$. The magnitude of the weight w_i indicates the relevance of a "yes" answer for x_i in determining the classification. If $p_i = q_i$, x_i gives us no information about the state of nature, and $w_i = 0$, just as we might expect. If $p_i > q_i$, then $1 - p_i < 1 - q_i$ and w_i is positive. Thus in this case a "yes" answer for x_i contributes w_i votes for ω_1. Furthermore, for any fixed $q_i < 1$, w_i gets larger as p_i gets larger. On the other hand, if $p_i < q_i$, w_i is negative and a "yes" answer contributes $|w_i|$ votes for ω_2.

The condition of feature independence leads to a very simple (linear) classifier; of course if the features were not independent, a more complicated classifier would be needed. We shall come across this again for systems with continuous features but note here that the more independent we can make the features, the simpler the classifier can be.

The prior probabilities $P(\omega_i)$ appear in the discriminant only through the threshold weight w_0. Increasing $P(\omega_1)$ increases w_0 and biases the decision in favor of ω_1, whereas decreasing $P(\omega_1)$ has the opposite effect. Geometrically, the possible values for $\mathbf{x}$ appear as the vertices of a d-dimensional hypercube; the decision surface defined by $g(\mathbf{x}) = 0$ is a hyperplane that separates ω_1 vertices from ω_2 vertices.

EXAMPLE 3 Bayesian Decisions for Three-Dimensional Binary Data

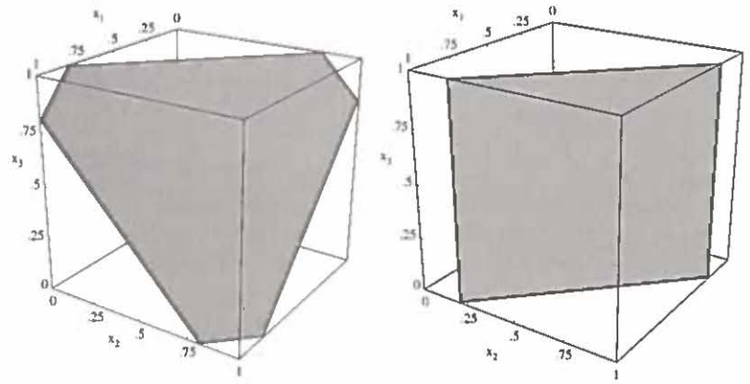
Consider a two-class problem having three independent binary features with known feature probabilities. Let us construct the Bayesian decision boundary if $P(\omega_1) = P(\omega_2) = 0.5$ and the individual components obey $p_i = 0.8$ and $q_i = 0.5$ for $i = 1, 2, 3$. By Eqs. 91 and 92 we have that the weights are

$$w_i = \ln \frac{.8(1 - .5)}{.5(1 - .8)} = 1.3863$$

and the bias value is

$$w_0 = \sum_{i=1}^{3} \ln \frac{1 - .8}{1 - .5} + \ln \frac{.5}{.5} = 1.2.$$

The surface $g(\mathbf{x}) = 0$ from Eq. 90 is shown on the left of the figure. Indeed, as we might have expected, the boundary places points with two or more "yes" answers into category ω_1, because that category has a higher probability of having any feature take value 1.



The decision boundary for the example involving three-dimensional binary features. On the left we show the case $p_i = .8$ and $q_i = .5$. On the right we use the same values except $p_3 = q_3$, which leads to $w_3 = 0$ and a decision surface parallel to the x_3 axis.

Suppose instead that while the prior probabilities remained the same, our individual components obeyed $p_1 = p_2 = 0.8$, $p_3 = 0.5$ and $q_1 = q_2 = q_3 = 0.5$. In this case feature x_3 gives us no predictive information about the categories, and hence the decision boundary is parallel to the x_3 axis. Note that in this discrete case there is a large range in positions of the decision boundary that leaves the categorization unchanged, as is particularly clear in the figure on the right.

*2.10 MISSING AND NOISY FEATURES

If we know the full probability structure of a problem, we can construct the (optimal) Bayes decision rule. Suppose we develop a Bayes classifier using uncorrupted data, but our input (test) data are then corrupted in particular known ways. How can we classify such corrupted inputs to obtain a minimum error now?

There are two analytically solvable cases of particular interest: when some of the features are *missing*, and when they are corrupted by a *noise source* with known properties. In each case our basic approach is to recover as much information about the underlying distribution as possible and use the Bayes decision rule.

2.10.1 Missing Features

Suppose we have a Bayesian (or other) recognizer for a problem using two features, but that for a particular pattern to be classified, one of the features is missing. For example, we can easily imagine that the lightness can be measured from a portion of a fish, but the width cannot because of occlusion by another fish.

We can illustrate with four categories a somewhat more general case (Fig. 2.22). Suppose that for a particular test pattern the feature x_1 is missing, and the measured value of x_2 is $\hat{x}_2$. Clearly if we assume that the missing value is the *mean* of all the x_1 values (i.e., $\bar{x}_1$), we will classify the pattern as ω_3. However, if the priors are equal, ω_2 would be a better decision, because the figure implies that $p(\hat{x}_2|\omega_2)$ is the largest of the four likelihoods.

To clarify our derivation we let $\mathbf{x} = [\mathbf{x}_g, \mathbf{x}_b]$, where $\mathbf{x}_g$ represents the known or "good" features and $\mathbf{x}_b$ represents the "bad" ones—that is, either unknown or missing. We seek the Bayes rule given the good features, and for that the posterior

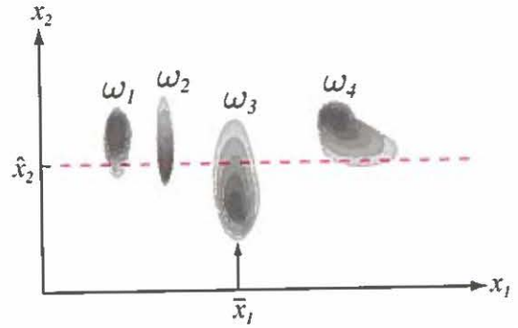


FIGURE 2.22. Four categories have equal priors and the class-conditional distributions shown. If a test point is presented in which one feature is missing (here, x_1) and the other is measured to have value $\hat{x}_2$ (red dashed line), we want our classifier to classify the pattern as category ω_2, because $p(\hat{x}_2|\omega_2)$ is the largest of the four likelihoods.

probabilities are needed. In terms of the good features the posteriors are

$$P(\omega_i|\mathbf{x}_g) = \frac{p(\omega_i, \mathbf{x}_g)}{p(\mathbf{x}_g)} = \frac{\int p(\omega_i, \mathbf{x}_g, \mathbf{x}_b)\, d\mathbf{x}_b}{p(\mathbf{x}_g)}$$

$$= \frac{\int P(\omega_i|\mathbf{x}_g, \mathbf{x}_b)\, p(\mathbf{x}_g, \mathbf{x}_b)\, d\mathbf{x}_b}{p(\mathbf{x}_g)}$$

$$= \frac{\int g_i(\mathbf{x})\, p(\mathbf{x})\, d\mathbf{x}_b}{\int p(\mathbf{x})\, d\mathbf{x}_b}, \tag{93}$$

where $g_i(\mathbf{x}) = g_i(\mathbf{x}_g, \mathbf{x}_b) = P(\omega_i|\mathbf{x}_g, \mathbf{x}_b)$ is one form of our discriminant function.

MARGINAL

We refer to $\int p(\omega_i, \mathbf{x}_g, \mathbf{x}_b)\, d\mathbf{x}_b$, as a *marginal distribution*; we say the full joint distribution is marginalized over the variable $\mathbf{x}_b$. In short, Eq. 93 shows that we must integrate (marginalize) the posterior probability over the bad features. Finally we use the Bayes decision rule on the resulting posterior probabilities, that is, choose ω_i if $P(\omega_i|\mathbf{x}_g) > P(\omega_j|\mathbf{x}_g)$ for all i and j. In Chapter 3 we shall consider the Expectation-Maximization (EM) algorithm, which addresses a related problem involving missing features.

2.10.2 Noisy Features

It is a simple matter to generalize the results of Eq. 93 to the case where a particular feature has been corrupted by statistically independent noise. For instance, in our fish classification example, we might have a reliable measurement of the length, while variability of the light source might degrade the measurement of the lightness. We assume we have uncorrupted (good) features $\mathbf{x}_g$, as before, and a *noise model*, expressed as $p(\mathbf{x}_b|\mathbf{x}_t)$. Here we let $\mathbf{x}_t$ denote the true value of the observed $\mathbf{x}_b$ features, that is, without the noise present; in short, the $\mathbf{x}_b$ are observed instead of the true $\mathbf{x}_t$. We assume that if $\mathbf{x}_t$ were known, $\mathbf{x}_b$ would be independent of ω_i and $\mathbf{x}_g$. From such an assumption we get:

$$P(\omega_i|\mathbf{x}_g, \mathbf{x}_b) = \frac{\int p(\omega_i, \mathbf{x}_g, \mathbf{x}_b, \mathbf{x}_t)\, d\mathbf{x}_t}{p(\mathbf{x}_g, \mathbf{x}_b)}. \tag{94}$$

CHAPTER

3

MAXIMUM-LIKELIHOOD AND BAYESIAN PARAMETER ESTIMATION

3.1 INTRODUCTION

In Chapter 2 we saw how we could design an optimal classifier if we knew the prior probabilities $P(\omega_i)$ and the class-conditional densities $p(\mathbf{x}|\omega_i)$. Unfortunately, in pattern recognition applications we rarely, if ever, have this kind of complete knowledge about the probabilistic structure of the problem. In a typical case we merely have some vague, general knowledge about the situation, together with a number of

TRAINING DATA

design samples or *training data*—particular representatives of the patterns we want to classify. The problem, then, is to find some way to use this information to design or train the classifier.

One approach to this problem is to use the samples to estimate the unknown probabilities and probability densities, and then use the resulting estimates as if they were the true values. In typical supervised pattern classification problems, the estimation of the prior probabilities presents no serious difficulties (Problem 3). However, estimation of the class-conditional densities is quite another matter. The number of available samples always seems too small, and serious problems arise when the dimensionality of the feature vector $\mathbf{x}$ is large. If we know the number of parameters in advance and our general knowledge about the problem permits us to parameterize the conditional densities, then the severity of these problems can be reduced significantly. Suppose, for example, that we can reasonably assume that $p(\mathbf{x}|\omega_i)$ is a normal density with mean $\boldsymbol{\mu}_i$ and covariance matrix $\boldsymbol{\Sigma}_i$, although we do not know the exact values of these quantities. This knowledge simplifies the problem from one of estimating an unknown *function* $p(\mathbf{x}|\omega_i)$ to one of estimating the *parameters* $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$.

MAXIMUM-LIKELIHOOD
BAYESIAN ESTIMATION

The problem of parameter estimation is a classical one in statistics, and it can be approached in several ways. We shall consider two common and reasonable procedures, namely, *maximum-likelihood* estimation and *Bayesian* estimation. Although the results obtained with these two procedures are frequently nearly identical,

BAYESIAN LEARNING

the approaches are conceptually quite different. Maximum-likelihood and several other methods view the parameters as quantities whose values are fixed but unknown. The best estimate of their value is defined to be the one that maximizes the probability of obtaining the samples actually observed. In contrast, Bayesian methods view the parameters as random variables having some known prior distribution. Observation of the samples converts this to a posterior density, thereby revising our opinion about the true values of the parameters. In the Bayesian case, we shall see that a typical effect of observing additional samples is to sharpen the *a posteriori* density function, causing it to peak near the true values of the parameters. This phenomenon is known as *Bayesian learning*. In either case, we use the posterior densities for our classification rule, as we have seen before.

It is important to distinguish between supervised learning and unsupervised learning. In both cases, samples $\mathbf{x}$ are assumed to be obtained by selecting a state of nature ω_i with probability $P(\omega_i)$, and then independently selecting $\mathbf{x}$ according to the probability law $p(\mathbf{x}|\omega_i)$. The distinction is that with supervised learning we know the state of nature (class label) for each sample, whereas with unsupervised learning we do not. As one would expect, the problem of unsupervised learning is the more difficult one. In this chapter we shall consider only the supervised case, deferring consideration of unsupervised learning to Chapter 10.

Finally, there are nonparametric procedures for transforming the feature space in the hope that it may be possible to employ parametric methods in the transformed space. These discriminant analysis methods include the Fisher linear discriminant, which provides an important link between the parametric techniques of Chapter 3 and the adaptive techniques of Chapters 5 and 6 and some methods in feature selection described in Chapter 10.

3.2 MAXIMUM-LIKELIHOOD ESTIMATION

Maximum-likelihood estimation methods have a number of attractive attributes. First, they nearly always have good convergence properties as the number of training samples increases. Furthermore, maximum-likelihood estimation often can be simpler than alternative methods, such as Bayesian techniques or other methods presented in subsequent chapters.

3.2.1 The General Principle

Suppose that we separate a collection of samples according to class, so that we have c data sets, $\mathcal{D}_1, \ldots, \mathcal{D}_c$, with the samples in $\mathcal{D}_j$ having been drawn independently according to the probability law $p(\mathbf{x}|\omega_j)$. We say such samples are *i.i.d.*—independent and identically distributed random variables. We assume that $p(\mathbf{x}|\omega_j)$ has a known parametric form, and is therefore determined uniquely by the value of a parameter vector $\boldsymbol{\theta}_j$. For example, we might have $p(\mathbf{x}|\omega_j) \sim N(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, where $\boldsymbol{\theta}_j$ consists of the components of $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$. To show the dependence of $p(\mathbf{x}|\omega_j)$ on $\boldsymbol{\theta}_j$ explicitly, we write $p(\mathbf{x}|\omega_j)$ as $p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$. Our problem is to use the information provided by the training samples to obtain good estimates for the unknown parameter vectors $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_c$ associated with each category.

I.I.D.

To simplify treatment of this problem, we shall assume that samples in $\mathcal{D}_i$ give no information about $\boldsymbol{\theta}_j$ if $i \neq j$; that is, we shall assume that the parameters for the different classes are functionally independent. This permits us to work with each

class separately and to simplify our notation by deleting indications of class distinctions. With this assumption we thus have c separate problems of the following form: Use a set $\mathcal{D}$ of training samples drawn independently from the probability density $p(\mathbf{x}|\boldsymbol{\theta})$ to estimate the unknown parameter vector $\boldsymbol{\theta}$.

Suppose that $\mathcal{D}$ contains n samples, $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Then, because the samples were drawn independently, we have

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{k=1}^{n} p(\mathbf{x}_k|\boldsymbol{\theta}). \tag{1}$$

Recall from Chapter 2 that, viewed as a function of $\boldsymbol{\theta}$, $p(\mathcal{D}|\boldsymbol{\theta})$ is called the *likelihood* of $\boldsymbol{\theta}$ with respect to the set of samples. The *maximum-likelihood estimate* of $\boldsymbol{\theta}$ is, by definition, the value $\hat{\boldsymbol{\theta}}$ that maximizes $p(\mathcal{D}|\boldsymbol{\theta})$. Intuitively, this estimate corresponds to the value of $\boldsymbol{\theta}$ that in some sense best agrees with or supports the actually observed training samples (Fig. 3.1).

For analytical purposes, it is usually easier to work with the logarithm of the likelihood than with the likelihood itself. Because the logarithm is monotonically increasing, the $\hat{\boldsymbol{\theta}}$ that maximizes the log-likelihood also maximizes the likelihood. If $p(\mathcal{D}|\boldsymbol{\theta})$ is a well-behaved, differentiable function of $\boldsymbol{\theta}$, $\hat{\boldsymbol{\theta}}$ can be found by the standard methods of differential calculus. If the number of parameters to be estimated is

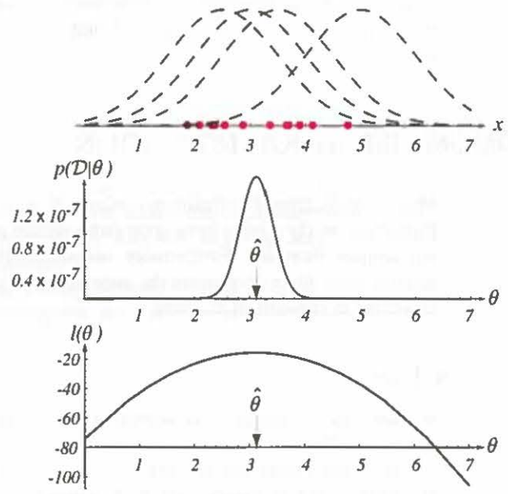


FIGURE 3.1. The top graph shows several training points in one dimension, known or assumed to be drawn from a Gaussian of a particular variance, but unknown mean. Four of the infinite number of candidate source distributions are shown in dashed lines. The middle figure shows the likelihood $p(\mathcal{D}|\theta)$ as a function of the mean. If we had a very large number of training points, this likelihood would be very narrow. The value that maximizes the likelihood is marked $\hat{\theta}$; it also maximizes the logarithm of the likelihood—that is, the log-likelihood $l(\theta)$, shown at the bottom. Note that even though they look similar, the likelihood $p(\mathcal{D}|\theta)$ is shown as a function of θ whereas the conditional density $p(x|\theta)$ is shown as a function of x. Furthermore, as a function of θ, the likelihood $p(\mathcal{D}|\theta)$ is not a probability density function and its area has no significance.

p, then we let $\boldsymbol{\theta}$ denote the p-component vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)^t$, and we let $\nabla_{\boldsymbol{\theta}}$ be the gradient operator

$$\nabla_{\boldsymbol{\theta}} \equiv \begin{bmatrix} \frac{\partial}{\partial \theta_1} \\ \vdots \\ \frac{\partial}{\partial \theta_p} \end{bmatrix}. \tag{2}$$

LOG-LIKELIHOOD

We define $l(\boldsymbol{\theta})$ as the *log-likelihood* function*

$$l(\boldsymbol{\theta}) \equiv \ln p(\mathcal{D}|\boldsymbol{\theta}). \tag{3}$$

We can then write our solution formally as the argument $\boldsymbol{\theta}$ that maximizes the log-likelihood, that is,

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} l(\boldsymbol{\theta}), \tag{4}$$

where the dependence on the data set $\mathcal{D}$ is implicit. Thus we have from Eq. 1

$$l(\boldsymbol{\theta}) = \sum_{k=1}^{n} \ln p(\mathbf{x}_k|\boldsymbol{\theta}) \tag{5}$$

and

$$\nabla_{\boldsymbol{\theta}} l = \sum_{k=1}^{n} \nabla_{\boldsymbol{\theta}} \ln p(\mathbf{x}_k|\boldsymbol{\theta}). \tag{6}$$

Thus, a set of necessary conditions for the maximum-likelihood estimate for $\boldsymbol{\theta}$ can be obtained from the set of p equations

$$\nabla_{\boldsymbol{\theta}} l = \mathbf{0}. \tag{7}$$

A solution $\hat{\boldsymbol{\theta}}$ to Eq. 7 could represent a true global maximum, a *local* maximum or minimum, or (rarely) an inflection point of $l(\boldsymbol{\theta})$. One must be careful, too, to check if the extremum occurs at a boundary of the parameter space, which might not be apparent from the solution to Eq. 7. If all solutions are found, we are guaranteed that one represents the true maximum, though we might have to check each solution individually (or calculate second derivatives) to identify which is the global optimum. Of course, we must bear in mind that $\hat{\boldsymbol{\theta}}$ is an estimate; it is only in the limit of an infinitely large number of training points that we can expect that our estimate will equal to the true value of the generating function.

MAXIMUM A POSTERIORI

MODE

We note in passing that a related class of estimators—*maximum a posteriori* or MAP estimators—find the value of $\boldsymbol{\theta}$ that maximizes $l(\boldsymbol{\theta})p(\boldsymbol{\theta})$, where $p(\boldsymbol{\theta})$ describes the prior probability of different parameter values. Thus a maximum-likelihood estimator is a MAP estimator for the uniform or "flat" prior. As such, a MAP estimator finds the peak, or *mode* of a posterior density. The drawback of MAP estimators is that if we choose some arbitrary nonlinear transformation of the

*Of course, the base of the logarithm can be chosen for convenience, and in most analytic problems, base e is most appropriate. For that reason we will generally use ln rather than log or $\log_2$.

parameter space (e.g., an overall rotation), the density will change, and our MAP solution need no longer be appropriate (Section 3.5.2).

3.2.2 The Gaussian Case: Unknown $\boldsymbol{\mu}$

To see how maximum-likelihood methods apply to a specific case, suppose that the samples are drawn from a multivariate normal population with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. For simplicity, consider first the case where only the mean is unknown. Under this condition, we consider a sample point $\mathbf{x}_k$ and find

$$\ln p(\mathbf{x}_k|\boldsymbol{\mu}) = -\frac{1}{2}\ln\left[(2\pi)^d|\boldsymbol{\Sigma}|\right] - \frac{1}{2}(\mathbf{x}_k - \boldsymbol{\mu})^t\boldsymbol{\Sigma}^{-1}(\mathbf{x}_k - \boldsymbol{\mu}) \tag{8}$$

and

$$\nabla_{\boldsymbol{\mu}} \ln p(\mathbf{x}_k|\boldsymbol{\mu}) = \boldsymbol{\Sigma}^{-1}(\mathbf{x}_k - \boldsymbol{\mu}). \tag{9}$$

Identifying $\boldsymbol{\theta}$ with $\boldsymbol{\mu}$, we see from Eqs. 6, 7 and 9 that the maximum-likelihood estimate for $\boldsymbol{\mu}$ must satisfy

$$\sum_{k=1}^{n}\boldsymbol{\Sigma}^{-1}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}) = \mathbf{0}. \tag{10}$$

Multiplying by $\boldsymbol{\Sigma}$ and rearranging, we obtain

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_k. \tag{11}$$

This is a very satisfying result. It says that the maximum-likelihood estimate for the unknown population mean is just the arithmetic average of the training samples—the *sample mean*, sometimes written $\hat{\boldsymbol{\mu}}_n$ to clarify its dependence on the number of samples. Geometrically, if we think of the n samples as a cloud of points, the sample mean is the centroid of the cloud. The sample mean has a number of desirable statistical properties as well, and one would be inclined to use this rather obvious estimate even without knowing that it is the maximum-likelihood solution.

SAMPLE MEAN

3.2.3 The Gaussian Case: Unknown μ and Σ

In the more general (and more typical) multivariate normal case, neither the mean $\boldsymbol{\mu}$ nor the covariance matrix $\boldsymbol{\Sigma}$ is known. Thus, these unknown parameters constitute the components of the parameter vector $\boldsymbol{\theta}$. Consider first the univariate case with $\theta_1 = \mu$ and $\theta_2 = \sigma^2$. Here the log-likelihood of a single point is

$$\ln p(x_k|\boldsymbol{\theta}) = -\frac{1}{2}\ln 2\pi\theta_2 - \frac{1}{2\theta_2}(x_k - \theta_1)^2 \tag{12}$$

and its derivative is

$$\nabla_{\boldsymbol{\theta}} l = \nabla_{\boldsymbol{\theta}} \ln p(x_k|\boldsymbol{\theta}) = \begin{bmatrix} \frac{1}{\theta_2}(x_k - \theta_1) \\ -\frac{1}{2\theta_2} + \frac{(x_k - \theta_1)^2}{2\theta_2^2} \end{bmatrix}. \tag{13}$$

Applying Eq. 7 to the full log-likelihood leads to the conditions

$$\sum_{k=1}^{n}\frac{1}{\hat{\theta}_2}(x_k - \hat{\theta}_1) = 0 \tag{14}$$

and

$$-\sum_{k=1}^{n}\frac{1}{\hat{\theta}_2} + \sum_{k=1}^{n}\frac{(x_k - \hat{\theta}_1)^2}{\hat{\theta}_2^2} = 0, \tag{15}$$

where $\hat{\theta}_1$ and $\hat{\theta}_2$ are the maximum-likelihood estimates for θ_1 and θ_2, respectively. By substituting $\hat{\mu} = \hat{\theta}_1$ and $\hat{\sigma}^2 = \hat{\theta}_2$ and doing a little rearranging, we obtain the following maximum-likelihood estimates for μ and σ^2:

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n}x_k \tag{16}$$

and

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{k=1}^{n}(x_k - \hat{\mu})^2. \tag{17}$$

While the analysis of the multivariate case is basically very similar, considerably more manipulations are involved (Problem 6). Just as we would predict, however, the result is that the maximum-likelihood estimates for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are given by

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_k \tag{18}$$

and

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{k=1}^{n}(\mathbf{x}_k - \hat{\boldsymbol{\mu}})(\mathbf{x}_k - \hat{\boldsymbol{\mu}})^t. \tag{19}$$

Thus, once again we find that the maximum-likelihood estimate for the mean vector is the sample mean. The maximum-likelihood estimate for the covariance matrix is the arithmetic average of the n matrices $(\mathbf{x}_k - \hat{\boldsymbol{\mu}})(\mathbf{x}_k - \hat{\boldsymbol{\mu}})^t$. Because the true covariance matrix is the expected value of the matrix $(\mathbf{x} - \hat{\boldsymbol{\mu}})(\mathbf{x} - \hat{\boldsymbol{\mu}})^t$, this is also a very satisfying result.

3.2.4 Bias

BIAS

The maximum-likelihood estimate for the variance σ^2 is *biased*; that is, the expected value over all data sets of size n of the sample variance is not equal to the true variance:*

$$\mathcal{E}\left[\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right] = \frac{n-1}{n}\sigma^2 \neq \sigma^2. \tag{20}$$

*The word "bias" is often used to refer generally to an offset. The bias in a statistical estimate is unrelated to the bias weights in a discriminant function or a multilayer neural network.

We shall return to a more general consideration of bias in Chapter 9, but for the moment we can verify Eq. 20 for an underlying distribution with nonzero variance, σ^2, in the extreme case of $n = 1$, in which the expectation value is given by $\mathcal{E}[\cdot] = 0 \neq \sigma^2$. The maximum-likelihood estimate of the covariance matrix is similarly biased.

An elementary *unbiased* estimator for $\boldsymbol{\Sigma}$ is given by

$$\mathbf{C} = \frac{1}{n-1}\sum_{k=1}^{n}(\mathbf{x}_k - \hat{\boldsymbol{\mu}})(\mathbf{x}_k - \hat{\boldsymbol{\mu}})^t, \tag{21}$$

SAMPLE COVARIANCE

ABSOLUTELY UNBIASED

ASYMPTOTICALLY UNBIASED

where $\mathbf{C}$ is the so-called *sample covariance matrix*, as explored in Problem 30. If an estimator is unbiased for *all* distributions, as for example the variance estimator in Eq. 21, then it is called *absolutely unbiased*. If the estimator tends to become unbiased as the number of samples becomes very large, as for instance Eq. 20, then the estimator is *asymptotically unbiased*. In many pattern recognition problems with large training data sets, asymptotically unbiased estimators are acceptable.

Clearly, $\widehat{\boldsymbol{\Sigma}} = [(n-1)/n]\mathbf{C}$, and $\widehat{\boldsymbol{\Sigma}}$ is asymptotically unbiased; these two estimates are essentially identical when n is large. However, the existence of two similar but nevertheless distinct estimates for the covariance matrix may be disconcerting, and it is natural to ask which one is "correct." Of course, for $n > 1$ the answer is that these estimates are neither right nor wrong—they are just different. What the existence of two actually shows is that no single estimate possesses all of the properties we might desire. For our purposes, the most desirable property is rather complex—we want the estimate that leads to the best classification performance. While it is usually both reasonable and sound to design a classifier by substituting the maximum-likelihood estimates for the unknown parameters, we might well wonder if other estimates might not lead to better performance. Below we address this question from a Bayesian viewpoint.

If we have a reliable model for the underlying distributions and their dependence upon the parameter vector $\boldsymbol{\theta}$, the maximum-likelihood classifier can give excellent results. But what if our model is wrong? Do we nevertheless get the best classifier in our assumed set of models? For instance, what if we assume that a distribution comes from $N(\mu, 1)$ but instead it actually comes from $N(\mu, 10)$? Will the value we find for $\theta = \mu$ by maximum-likelihood yield the best of all classifiers of the form derived from $N(\mu, 1)$? Unfortunately, the answer is "no," and an illustrative counterexample is given in Problem 7 where the so-called *model error* is large indeed. This points out the need for reliable information concerning the models: If the assumed model is very poor, we cannot be assured that the classifier we derive is the best, even among our model set. We shall return to the problem of choosing among candidate models in Chapter 9.

3.3 BAYESIAN ESTIMATION

We now consider the Bayesian estimation or Bayesian learning approach to pattern classification problems. Although the answers we get by this method will generally be nearly identical to those obtained by maximum-likelihood, there is a conceptual difference: Whereas in maximum-likelihood methods we view the true parameter vector we seek, $\boldsymbol{\theta}$, to be fixed, in Bayesian learning we consider $\boldsymbol{\theta}$ to be a random variable, and training data allow us to convert a distribution on this variable into a posterior probability density.

3.3.1 The Class-Conditional Densities

The computation of posterior probabilities $P(\omega_i|\mathbf{x})$ lies at the heart of Bayesian classification. Bayes formula allows us to compute these probabilities from the prior probabilities $P(\omega_i)$ and the class-conditional densities $p(\mathbf{x}|\omega_i)$, but how can we proceed when these quantities are unknown? The general answer to this question is that the best we can do is to compute $P(\omega_i|\mathbf{x})$ using all of the information at our disposal. Part of this information might be prior knowledge, such as knowledge of the functional forms for unknown densities and ranges for the values of unknown parameters. Part of this information might reside in a set of training samples. If we again let $\mathcal{D}$ denote the set of samples, then we can emphasize the role of the samples by saying that our goal is to compute the posterior probabilities $P(\omega_i|\mathbf{x}, \mathcal{D})$. From these probabilities we can obtain the Bayes classifier.

Given the sample $\mathcal{D}$, Bayes formula then becomes

$$P(\omega_i|\mathbf{x}, \mathcal{D}) = \frac{p(\mathbf{x}|\omega_i, \mathcal{D})P(\omega_i|\mathcal{D})}{\sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \mathcal{D})P(\omega_j|\mathcal{D})}. \tag{22}$$

As this equation suggests, we can use the information provided by the training samples to help determine both the class-conditional densities and the prior probabilities.

Although we could maintain this generality, we shall henceforth assume that the true values of the prior probabilities are known or obtainable from a trivial calculation; thus we substitute $P(\omega_i) = P(\omega_i|\mathcal{D})$. Furthermore, because we are treating the supervised case, we can separate the training samples by class into c subsets $\mathcal{D}_1, \ldots, \mathcal{D}_c$, with the samples in $\mathcal{D}_i$ belonging to ω_i. As we mentioned when addressing maximum-likelihood methods, in most cases of interest (and in all of the cases we shall consider), the samples in $\mathcal{D}_i$ have no influence on $p(\mathbf{x}|\omega_j, \mathcal{D})$ if $i \neq j$. This has two simplifying consequences. First, it allows us to work with each class separately, using only the samples in $\mathcal{D}_i$ to determine $p(\mathbf{x}|\omega_i, \mathcal{D})$. Used in conjunction with our assumption that the prior probabilities are known, this allows us to write Eq. 22 as

$$P(\omega_i|\mathbf{x}, \mathcal{D}) = \frac{p(\mathbf{x}|\omega_i, \mathcal{D}_i)P(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \mathcal{D}_j)P(\omega_j)}. \tag{23}$$

Second, because each class can be treated independently, we can dispense with needless class distinctions and simplify our notation. In essence, we have c separate problems of the following form: Use a set $\mathcal{D}$ of samples drawn independently according to the fixed but unknown probability distribution $p(\mathbf{x})$ to determine $p(\mathbf{x}|\mathcal{D})$. This is the central problem of Bayesian learning.

3.3.2 The Parameter Distribution

Although the desired probability density $p(\mathbf{x})$ is unknown, we assume that it has a known parametric form. The only thing assumed unknown is the value of a parameter vector $\boldsymbol{\theta}$. We shall express the fact that $p(\mathbf{x})$ is unknown but has known parametric form by saying that the function $p(\mathbf{x}|\boldsymbol{\theta})$ is completely known. Any information we might have about $\boldsymbol{\theta}$ prior to observing the samples is assumed to be contained in a *known* prior density $p(\boldsymbol{\theta})$. Observation of the samples converts this to a posterior density $p(\boldsymbol{\theta}|\mathcal{D})$, which, we hope, is sharply peaked about the true value of $\boldsymbol{\theta}$.

Note that we have converted our problem of learning a probability density function to one of estimating a parameter vector. To this end, our basic goal is to compute $p(\mathbf{x}|\mathcal{D})$, which is as close as we can come to obtaining the unknown $p(\mathbf{x})$. We do this by integrating the joint density $p(\mathbf{x}, \boldsymbol{\theta}|\mathcal{D})$ over $\boldsymbol{\theta}$. That is,

$$p(\mathbf{x}|\mathcal{D}) = \int p(\mathbf{x}, \boldsymbol{\theta}|\mathcal{D})\, d\boldsymbol{\theta}, \tag{24}$$

where the integration extends over the entire parameter space. Now we can always write $p(\mathbf{x}, \boldsymbol{\theta}|\mathcal{D})$ as the product $p(\mathbf{x}|\boldsymbol{\theta}, \mathcal{D})p(\boldsymbol{\theta}|\mathcal{D})$. Because the selection of $\mathbf{x}$ and that of the training samples in $\mathcal{D}$ is done independently, the first factor is merely $p(\mathbf{x}|\boldsymbol{\theta})$. That is, the distribution of $\mathbf{x}$ is known completely once we know the value of the parameter vector. Thus, Eq. 24 can be rewritten as

$$p(\mathbf{x}|\mathcal{D}) = \int p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})\, d\boldsymbol{\theta}. \tag{25}$$

This key equation links the desired class-conditional density $p(\mathbf{x}|\mathcal{D})$ to the posterior density $p(\boldsymbol{\theta}|\mathcal{D})$ for the unknown parameter vector. If $p(\boldsymbol{\theta}|\mathcal{D})$ peaks very sharply about some value $\hat{\boldsymbol{\theta}}$, we obtain $p(\mathbf{x}|\mathcal{D}) \simeq p(\mathbf{x}|\hat{\boldsymbol{\theta}})$, i.e., the result we would obtain by substituting the estimate $\hat{\boldsymbol{\theta}}$ for the true parameter vector. This result rests on the assumption that $p(\mathbf{x}|\boldsymbol{\theta})$ is smooth, and that the tails of the integral are not important. These conditions are typically but not invariably the case. In general, if we are less certain about the exact value of $\boldsymbol{\theta}$, this equation directs us to average $p(\mathbf{x}|\boldsymbol{\theta})$ over the possible values of $\boldsymbol{\theta}$. Thus, when the unknown densities have a known parametric form, the samples exert their influence on $p(\mathbf{x}|\mathcal{D})$ through the posterior density $p(\boldsymbol{\theta}|\mathcal{D})$. We should also point out that in practice, the integration in Eq. 25 can be performed numerically, for instance by Monte-Carlo simulation.

3.4 BAYESIAN PARAMETER ESTIMATION: GAUSSIAN CASE

In this section we use Bayesian estimation techniques to calculate the *a posteriori* density $p(\boldsymbol{\theta}|\mathcal{D})$ and the desired probability density $p(\mathbf{x}|\mathcal{D})$ for the case where $p(\mathbf{x}|\boldsymbol{\mu}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

3.4.1 The Univariate Case: $p(\mu|\mathcal{D})$

Consider the case where $\boldsymbol{\mu}$ is the only unknown parameter. For simplicity we treat first the univariate case, that is,

$$p(x|\mu) \sim N(\mu, \sigma^2), \tag{26}$$

where the only unknown quantity is the mean μ. We assume that whatever prior knowledge we might have about μ can be expressed by a *known* prior density $p(\mu)$. Later we shall make the further assumption that

$$p(\mu) \sim N(\mu_0, \sigma_0^2), \tag{27}$$

where both μ_0 and σ_0^2 are known. Roughly speaking, μ_0 represents our best prior guess for μ, and σ_0^2 measures our uncertainty about this guess. The assumption that the prior distribution for μ is normal will simplify the subsequent mathematics. However, the crucial assumption is not so much that the prior distribution for μ is normal, but that it is known.

Having selected the prior density for μ, we can view the situation as follows. Imagine that a value is drawn for μ from a population governed by the probability law $p(\mu)$. Once this value is drawn, it becomes the true value of μ and completely determines the density for x. Suppose now that n samples $x_1, \ldots, x_n$ are independently drawn from the resulting population. Letting $\mathcal{D} = \{x_1, \ldots, x_n\}$, we use Bayes formula to obtain

$$\begin{aligned} p(\mu|\mathcal{D}) &= \frac{p(\mathcal{D}|\mu)p(\mu)}{\int p(\mathcal{D}|\mu)p(\mu)\, d\mu} \\ &= \alpha \prod_{k=1}^{n} p(x_k|\mu)p(\mu), \end{aligned} \tag{28}$$

where α is a normalization factor that depends on $\mathcal{D}$ but is independent of μ. This equation shows how the observation of a set of training samples affects our ideas about the true value of μ; it relates the prior density $p(\mu)$ to an *a posteriori* density $p(\mu|\mathcal{D})$. Because $p(x_k|\mu) \sim N(\mu, \sigma^2)$ and $p(\mu) \sim N(\mu_0, \sigma_0^2)$, we have

$$\begin{aligned} p(\mu|\mathcal{D}) &= \alpha \prod_{k=1}^{n} \overbrace{\frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x_k - \mu}{\sigma}\right)^2\right]}^{p(x_k|\mu)} \overbrace{\frac{1}{\sqrt{2\pi}\sigma_0} \exp\left[-\frac{1}{2}\left(\frac{\mu - \mu_0}{\sigma_0}\right)^2\right]}^{p(\mu)} \\ &= \alpha' \exp\left[-\frac{1}{2}\left(\sum_{k=1}^{n}\left(\frac{\mu - x_k}{\sigma}\right)^2 + \left(\frac{\mu - \mu_0}{\sigma_0}\right)^2\right)\right] \\ &= \alpha'' \exp\left[-\frac{1}{2}\left[\left(\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}\right)\mu^2 - 2\left(\frac{1}{\sigma^2}\sum_{k=1}^{n} x_k + \frac{\mu_0}{\sigma_0^2}\right)\mu\right]\right], \end{aligned} \tag{29}$$

where factors that do not depend on μ have been absorbed into the constants α, α', and α''. Thus, $p(\mu|\mathcal{D})$ is an exponential function of a quadratic function of μ, i.e., is again a normal density. Because this is true for any number of training samples, $p(\mu|\mathcal{D})$ remains normal as the number n of samples is increased, and $p(\mu|\mathcal{D})$ is said to be a *reproducing density* and $p(\mu)$ is said to be a *conjugate prior*. If we write $p(\mu|\mathcal{D}) \sim N(\mu_n, \sigma_n^2)$, then μ_n and σ_n^2 can be found by equating coefficients in Eq. 29 with corresponding coefficients in the generic Gaussian of the form

REPRODUCING DENSITY

$$p(\mu|\mathcal{D}) = \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left[-\frac{1}{2}\left(\frac{\mu - \mu_n}{\sigma_n}\right)^2\right]. \tag{30}$$

Identifying coefficients in this way yields

$$\frac{1}{\sigma_n^2} = \frac{n}{\sigma^2} + \frac{1}{\sigma_0^2} \tag{31}$$

and

$$\frac{\mu_n}{\sigma_n^2} = \frac{n}{\sigma^2}\hat{\mu}_n + \frac{\mu_0}{\sigma_0^2}, \tag{32}$$

where $\hat{\mu}_n$ is the sample mean

$$\hat{\mu}_n = \frac{1}{n}\sum_{k=1}^{n} x_k. \tag{33}$$

We solve explicitly for μ_n and σ_n^2 and obtain

$$\mu_n = \left(\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\right)\hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2+\sigma^2}\mu_0 \tag{34}$$

and

$$\sigma_n^2 = \frac{\sigma_0^2\sigma^2}{n\sigma_0^2+\sigma^2}. \tag{35}$$

These equations show how the prior information is combined with the empirical information in the samples to obtain the *a posteriori* density $p(\mu|\mathcal{D})$. Roughly speaking, μ_n represents our best guess for μ after observing n samples, and σ_n^2 measures our uncertainty about this guess. Because σ_n^2 decreases monotonically with n—approaching σ^2/n as n approaches infinity—each additional observation decreases our uncertainty about the true value of μ. As n increases, $p(\mu|\mathcal{D})$ becomes more and more sharply peaked, approaching a Dirac delta function as n approaches infinity. This behavior is commonly known as *Bayesian learning* (Fig. 3.2).

BAYESIAN LEARNING

In general, μ_n is a linear combination of $\hat{\mu}_n$ and μ_0, with coefficients that are nonnegative and sum to one. Thus μ_n always lies somewhere between $\hat{\mu}_n$ and μ_0. If $\sigma_0 \neq 0$, μ_n approaches the sample mean as n approaches infinity. If $\sigma_0 = 0$, we have a degenerate case in which our prior certainty that $\mu = \mu_0$ is so strong that no number of observations can change our opinion. At the other extreme, if $\sigma_0 \gg \sigma$, we are so uncertain about our prior guess that we take $\mu_n = \hat{\mu}_n$, using only the samples

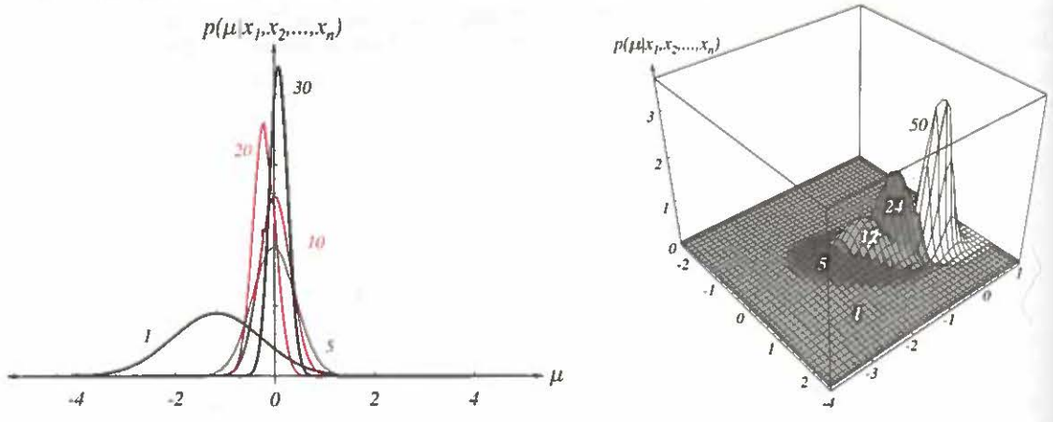


FIGURE 3.2. Bayesian learning of the mean of normal distributions in one and two dimensions. The posterior distribution estimates are labeled by the number of training samples used in the estimation.

DOGMATISM

to estimate μ. In general, the relative balance between prior knowledge and empirical data is set by the ratio of σ^2 to σ_0^2, which is sometimes called the *dogmatism*. If the dogmatism is not infinite, after enough samples are taken the exact values assumed for μ_0 and σ_0^2 will be unimportant, and μ_n will converge to the sample mean.

3.4.2 The Univariate Case: $p(x|\mathcal{D})$

Having obtained the *a posteriori* density for the mean, $p(\mu|\mathcal{D})$, all that remains is to obtain the "class-conditional" density for $p(x|\mathcal{D})$.* From Eqs. 25, 26 and 30 we have

$$\begin{aligned} p(x|\mathcal{D}) &= \int p(x|\mu)p(\mu|\mathcal{D})\,d\mu \\ &= \int \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]\frac{1}{\sqrt{2\pi}\sigma_n}\exp\left[-\frac{1}{2}\left(\frac{\mu-\mu_n}{\sigma_n}\right)^2\right]d\mu \\ &= \frac{1}{2\pi\sigma\sigma_n}\exp\left[-\frac{1}{2}\frac{(x-\mu_n)^2}{\sigma^2+\sigma_n^2}\right] f(\sigma,\sigma_n), \end{aligned} \tag{36}$$

where

$$f(\sigma,\sigma_n) = \int \exp\left[-\frac{1}{2}\frac{\sigma^2+\sigma_n^2}{\sigma^2\sigma_n^2}\left(\mu - \frac{\sigma_n^2 x+\sigma^2\mu_n}{\sigma^2+\sigma_n^2}\right)^2\right]d\mu.$$

That is, as a function of x, $p(x|\mathcal{D})$ is proportional to $\exp[-(1/2)(x-\mu_n)^2/(\sigma^2+\sigma_n^2)]$, and hence $p(x|\mathcal{D})$ is normally distributed with mean μ_n and variance $\sigma^2+\sigma_n^2$:

$$p(x|\mathcal{D}) \sim N(\mu_n, \sigma^2+\sigma_n^2). \tag{37}$$

In other words, to obtain the class-conditional density $p(x|\mathcal{D})$, whose parametric form is known to be $p(x|\mu) \sim N(\mu,\sigma^2)$, we merely replace μ by μ_n and σ^2 by $\sigma^2+\sigma_n^2$. In effect, the conditional mean μ_n is treated as if it were the true mean, and the known variance is increased to account for the additional uncertainty in x resulting from our lack of exact knowledge of the mean μ. This, then, is our final result: the density $p(x|\mathcal{D})$ is the desired class-conditional density $p(x|\omega_j,\mathcal{D}_j)$, and together with the prior probabilities $P(\omega_j)$ it gives us the probabilistic information needed to design the classifier. This is in contrast to maximum-likelihood methods that only make point estimates for $\hat{\mu}$ and $\hat{\sigma}^2$, rather that estimate a *distribution* for $p(x|\mathcal{D})$.

A non-parametric

3.4.3 The Multivariate Case

The treatment of the multivariate case in which Σ is known but $\boldsymbol{\mu}$ is not is a direct generalization of the univariate case. For this reason we shall only sketch the derivation. As before, we assume that

$$p(\mathbf{x}|\boldsymbol{\mu}) \sim N(\boldsymbol{\mu},\Sigma) \quad \text{and} \quad p(\boldsymbol{\mu}) \sim N(\boldsymbol{\mu}_0,\Sigma_0), \tag{38}$$

where Σ, Σ_0, and $\boldsymbol{\mu}_0$ are assumed to be known. After observing a set $\mathcal{D}$ of n independent samples $\mathbf{x}_1,\ldots,\mathbf{x}_n$, we use Bayes formula to obtain

*Recall that for simplicity we dropped class distinctions, but that all samples here come from the same class, say ω_i, and hence $p(\mathbf{x}|\mathcal{D})$ is really $p(\mathbf{x}|\omega_i,\mathcal{D}_i)$.

$$p(\boldsymbol{\mu}|\mathcal{D}) = \alpha \prod_{k=1}^{n} p(\mathbf{x}_k|\boldsymbol{\mu}) p(\boldsymbol{\mu}) \tag{39}$$

$$= \alpha' \exp\left[-\frac{1}{2} \left(\boldsymbol{\mu}^t \left(n\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}_0^{-1} \right) \boldsymbol{\mu} - 2\boldsymbol{\mu}^t \left(\boldsymbol{\Sigma}^{-1} \sum_{k=1}^{n} \mathbf{x}_k + \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0 \right) \right) \right],$$

which has the form

$$p(\boldsymbol{\mu}|\mathcal{D}) = \alpha'' \exp\left[-\frac{1}{2} (\boldsymbol{\mu} - \boldsymbol{\mu}_n)^t \boldsymbol{\Sigma}_n^{-1} (\boldsymbol{\mu} - \boldsymbol{\mu}_n) \right]. \tag{40}$$

Thus, $p(\boldsymbol{\mu}|\mathcal{D}) \sim N(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$, and once again we have a reproducing density. Equating coefficients, we obtain the analogs of Eqs. 34 and 35,

$$\boldsymbol{\Sigma}_n^{-1} = n\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}_0^{-1} \tag{41}$$

and

$$\boldsymbol{\Sigma}_n^{-1} \boldsymbol{\mu}_n = n\boldsymbol{\Sigma}^{-1} \hat{\boldsymbol{\mu}}_n + \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0, \tag{42}$$

where $\hat{\boldsymbol{\mu}}_n$ is the sample mean

$$\hat{\boldsymbol{\mu}}_n = \frac{1}{n} \sum_{k=1}^{n} \mathbf{x}_k. \tag{43}$$

The solution of these equations for $\boldsymbol{\mu}_n$ and $\boldsymbol{\Sigma}_n$ is simplified by knowledge of the matrix identity

$$(\mathbf{A}^{-1} + \mathbf{B}^{-1})^{-1} = \mathbf{A}(\mathbf{A} + \mathbf{B})^{-1}\mathbf{B} = \mathbf{B}(\mathbf{A} + \mathbf{B})^{-1}\mathbf{A}, \tag{44}$$

which is valid for any pair of nonsingular, d-by-d matrices $\mathbf{A}$ and $\mathbf{B}$. After a little manipulation (Problem 16), we obtain the final results:

$$\boldsymbol{\mu}_n = \boldsymbol{\Sigma}_0 \left(\boldsymbol{\Sigma}_0 + \frac{1}{n}\boldsymbol{\Sigma} \right)^{-1} \hat{\boldsymbol{\mu}}_n + \frac{1}{n}\boldsymbol{\Sigma} \left(\boldsymbol{\Sigma}_0 + \frac{1}{n}\boldsymbol{\Sigma} \right)^{-1} \boldsymbol{\mu}_0 \tag{45}$$

(which, as in the univariate case, is a linear combination of $\hat{\boldsymbol{\mu}}_n$ and $\boldsymbol{\mu}_0$) and

$$\boldsymbol{\Sigma}_n = \boldsymbol{\Sigma}_0 \left(\boldsymbol{\Sigma}_0 + \frac{1}{n}\boldsymbol{\Sigma} \right)^{-1} \frac{1}{n}\boldsymbol{\Sigma}. \tag{46}$$

The proof that $p(\mathbf{x}|\mathcal{D}) \sim N(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}+\boldsymbol{\Sigma}_n)$ can be obtained as before by performing the integration

$$p(\mathbf{x}|\mathcal{D}) = \int p(\mathbf{x}|\boldsymbol{\mu}) p(\boldsymbol{\mu}|\mathcal{D})\, d\boldsymbol{\mu}. \tag{47}$$

However, this result can be obtained with less effort by observing that $\mathbf{x}$ can be viewed as the sum of two mutually independent random variables, a random vector $\boldsymbol{\mu}$ with $p(\boldsymbol{\mu}|\mathcal{D}) \sim N(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$ and an independent random vector $\mathbf{y}$ with $p(\mathbf{y}) \sim N(\mathbf{0}, \boldsymbol{\Sigma})$. Because the sum of two independent, normally distributed vectors is again a normally distributed vector whose mean is the sum of the means and whose covariance matrix is the sum of the covariance matrices (Chapter 2, Problem 17), we have

$$p(\mathbf{x}|\mathcal{D}) \sim N(\boldsymbol{\mu}_n, \boldsymbol{\Sigma} + \boldsymbol{\Sigma}_n), \tag{48}$$

and the generalization is complete.

3.5 BAYESIAN PARAMETER ESTIMATION: GENERAL THEORY

We have just seen how the Bayesian approach can be used to obtain the desired density $p(\mathbf{x}|\mathcal{D})$ in a special case—the multivariate Gaussian. This approach can be generalized to apply to any situation in which the unknown density can be parameterized. The basic assumptions are summarized as follows:

- The form of the density $p(\mathbf{x}|\boldsymbol{\theta})$ is assumed to be known, but the value of the parameter vector $\boldsymbol{\theta}$ is not known exactly.
- Our initial knowledge about $\boldsymbol{\theta}$ is assumed to be contained in a known prior density $p(\boldsymbol{\theta})$.
- The rest of our knowledge about $\boldsymbol{\theta}$ is contained in a set $\mathcal{D}$ of n samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ drawn independently according to the unknown probability density $p(\mathbf{x})$.

The basic problem is to compute the posterior density $p(\boldsymbol{\theta}|\mathcal{D})$, because from this we can use Eq. 25 to compute $p(\mathbf{x}|\mathcal{D})$:

$$p(\mathbf{x}|\mathcal{D}) = \int p(\mathbf{x}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D})\, d\boldsymbol{\theta}. \tag{49}$$

By Bayes formula we have

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta})}{\int p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta})\, d\boldsymbol{\theta}}, \tag{50}$$

and by the independence assumption

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{k=1}^{n} p(\mathbf{x}_k|\boldsymbol{\theta}). \tag{51}$$

This constitutes the formal solution to the problem, and Eqs. 50 and 51 illuminate its relation to the maximum-likelihood solution. Suppose that $p(\mathcal{D}|\boldsymbol{\theta})$ reaches a sharp peak at $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$. If the prior density $p(\boldsymbol{\theta})$ is not zero at $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$ and does not change much in the surrounding neighborhood, then $p(\boldsymbol{\theta}|\mathcal{D})$ also peaks at that point. Thus, Eq. 49 shows that $p(\mathbf{x}|\mathcal{D})$ will be approximately $p(\mathbf{x}|\hat{\boldsymbol{\theta}})$, the result one would obtain by using the maximum-likelihood estimate as if it were the true value. If the peak of $p(\mathcal{D}|\boldsymbol{\theta})$ is very sharp, then the influence of prior information on the uncertainty in the true value of $\boldsymbol{\theta}$ can be ignored. In this and even the more general case, however, the Bayesian solution tells us how to use *all* the available information to compute the desired density $p(\mathbf{x}|\mathcal{D})$.

While we have obtained the formal Bayesian solution to the problem, a number of interesting questions remain. One concerns the difficulty of carrying out these computations. Another concerns the convergence of $p(\mathbf{x}|\mathcal{D})$ to $p(\mathbf{x})$. We shall discuss the matter of convergence briefly, and we will later turn to the computational question.

To indicate explicitly the number of samples in a set for a single category, we shall write $\mathcal{D}^n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Then from Eq. 51, if $n > 1$ we obtain

Recursive Bayes

$$p(\mathcal{D}^n|\boldsymbol{\theta}) = p(\mathbf{x}_n|\boldsymbol{\theta})p(\mathcal{D}^{n-1}|\boldsymbol{\theta}). \tag{52}$$

Substituting this in Eq. 50 and using Bayes formula, we see that the posterior density satisfies the recursion relation

$$p(\boldsymbol{\theta}|\mathcal{D}^n) = \frac{p(\mathbf{x}_n|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}^{n-1})}{\int p(\mathbf{x}_n|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}^{n-1})\,d\boldsymbol{\theta}}. \tag{53}$$

RECURSIVE BAYES INCREMENTAL LEARNING

With the understanding that $p(\boldsymbol{\theta}|\mathcal{D}^0) = p(\boldsymbol{\theta})$, repeated use of this equation produces the sequence of densities $p(\boldsymbol{\theta})$, $p(\boldsymbol{\theta}|\mathbf{x}_1)$, $p(\boldsymbol{\theta}|\mathbf{x}_1, \mathbf{x}_2)$, and so forth. (It should be obvious from Eq. 53 that $p(\boldsymbol{\theta}|\mathcal{D}^n)$ depends only on the points in $\mathcal{D}^n$, not the sequence in which they were selected.) This is called the *recursive Bayes* approach to parameter estimation. This is our first example of an *incremental* or *on-line* learning method, where learning goes on as the data are collected. When this sequence of densities converges to a Dirac delta function centered about the true parameter value, we have *Bayesian learning* (Example 1). We shall come across many other, nonincremental learning schemes, where all the training data must be present before learning can take place.

In principle, Eq. 53 requires that we preserve all the training points in $\mathcal{D}^{n-1}$ to calculate $p(\boldsymbol{\theta}|\mathcal{D}^n)$. However, for some distributions, just a few estimates of parameters associated with $p(\boldsymbol{\theta}|\mathcal{D}^{n-1})$ contain all the information needed. Such parameters are the *sufficient statistics* of those distributions, as we shall see in Section 3.6. Some authors reserve the term recursive learning to apply to only those cases where the sufficient statistics are retained—not the training data—when incorporating the information from a new training point. We could call this more restrictive usage *true recursive Bayes learning*.

EXAMPLE 1 Recursive Bayes Learning

Suppose we believe our one-dimensional samples come from a uniform distribution

$$p(x|\theta) \sim U(0, \theta) = \begin{cases} 1/\theta & 0 \le x \le \theta \\ 0 & \text{otherwise,} \end{cases}$$

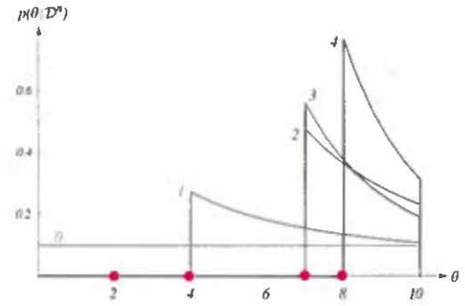
but initially we know only that our parameter is bounded. In particular we assume $0 < \theta \le 10$ (a noninformative or "flat prior" we shall discuss in Section 3.5.2). We will use recursive Bayes methods to estimate θ and the underlying densities from the data $\mathcal{D} = \{4, 7, 2, 8\}$, which were selected randomly from the underlying distribution. Before any data arrive, then, we have $p(\theta|\mathcal{D}^0) = p(\theta) = U(0, 10)$. When our first data point $x_1 = 4$ arrives, we use Eq. 53 to get an improved estimate:

$$p(\theta|\mathcal{D}^1) \propto p(x|\theta)p(\theta|\mathcal{D}^0) = \begin{cases} 1/\theta & \text{for } 4 \le \theta \le 10 \\ 0 & \text{otherwise,} \end{cases}$$

where throughout we will ignore the normalization. When the next data point $x_2 = 7$ arrives, we have

$$p(\theta|\mathcal{D}^2) \propto p(x|\theta)p(\theta|\mathcal{D}^1) = \begin{cases} 1/\theta^2 & \text{for } 7 \le \theta \le 10 \\ 0 & \text{otherwise,} \end{cases}$$

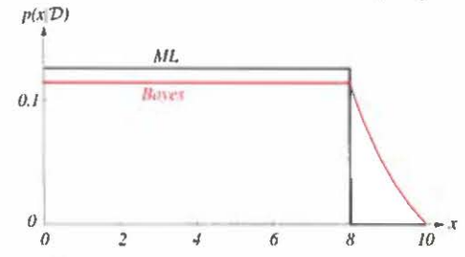
and similarly for the remaining sample points. It should be clear that because each successive step introduces a factor of $1/\theta$ into $p(x|\theta)$, and the distribution is nonzero only for x values above the largest data point sampled, the general form of our solution is $p(\theta|\mathcal{D}^n) \propto 1/\theta^n$ for $\max_x[\mathcal{D}^n] \le \theta \le 10$, as shown in the figure. Given our full data set, the maximum-likelihood solution here is clearly $\hat{\theta} = 8$, and this implies a uniform $p(x|\mathcal{D}) \sim U(0, 8)$.



The posterior $p(\theta|\mathcal{D}^n)$ for the model and n points in the data set in this Example. For $n = 0$, the posterior starts our as a flat, uniform density from 0 to 10, denoted $p(\theta) \sim U(0, 10)$. As more points are incorporated, it becomes increasingly peaked at the value of the highest data point.

According to our Bayesian methodology, which requires the integration in Eq. 49, the density is uniform up to $x = 8$, but has a tail at higher values—an indication that the influence of our prior $p(\theta)$ has not yet been swamped by the information in the training data.

U(0,10) → was assumption good?



Given the full set of four points, the distribution based on the maximum-likelihood solution is $p(x|\hat{\theta}) \sim U(0, 8)$, whereas the distribution derived from Bayesian methods has a small tail above $x = 8$, reflecting the prior information that values of x near 10 are possible.

Whereas the maximum-likelihood approach estimates a *point* in $\boldsymbol{\theta}$ space, the Bayesian approach instead estimates a *distribution*. Technically speaking, then, we cannot directly compare these estimates. It is only when the second stage of inference is done—that is, when we compute the distributions $p(x|\mathcal{D})$, as shown in the above figure—that the comparison is fair.

For most of the typically encountered probability densities $p(\mathbf{x}|\boldsymbol{\theta})$, the sequence of posterior densities does indeed converge to a delta function. Roughly speaking, this implies that with a large number of samples there is only one value for $\boldsymbol{\theta}$ that causes $p(\mathbf{x}|\boldsymbol{\theta})$ to fit the data, that is, that $\boldsymbol{\theta}$ can be determined uniquely from $p(\mathbf{x}|\boldsymbol{\theta})$. When this is the case, $p(\mathbf{x}|\boldsymbol{\theta})$ is said to be *identifiable*. A rigorous proof of convergence under these conditions requires a precise statement of the properties required of $p(\mathbf{x}|\boldsymbol{\theta})$ and $p(\boldsymbol{\theta})$ and considerable care, but presents no serious difficulties (Problem 21).

IDENTIFIABILITY

There are occasions, however, when more than one value of $\boldsymbol{\theta}$ may yield the same value for $p(\mathbf{x}|\boldsymbol{\theta})$. In such cases, $\boldsymbol{\theta}$ cannot be determined uniquely from $p(\mathbf{x}|\boldsymbol{\theta})$, and $p(\mathbf{x}|\mathcal{D}^n)$ will peak near all of the values of $\boldsymbol{\theta}$ that explain the data. Fortunately, this ambiguity is erased by the integration in Eq. 25, because $p(\mathbf{x}|\boldsymbol{\theta})$ is the same for all of these values of $\boldsymbol{\theta}$. Thus, $p(\mathbf{x}|\mathcal{D}^n)$ will typically converge to $p(\mathbf{x})$ whether or not $p(\mathbf{x}|\boldsymbol{\theta})$ is identifiable. While this might make the problem of identifiabilty appear to be moot, we shall see in Chapter 10 that identifiability presents a genuine problem in the case of unsupervised learning.

3.5.1 When Do Maximum-Likelihood and Bayes Methods Differ?

For reasonable prior distributions that do not preclude the true solution, maximum-likelihood and Bayes solutions are equivalent in the asymptotic limit of infinite training data. However since practical pattern recognition problems invariably have a limited set of training data, it is natural to ask when maximum-likelihood and Bayes solutions may be expected to differ, and then which we should prefer.

There are several criteria that will influence our choice. One is computational complexity (Section 3.7.2), and here maximum-likelihood methods are often to be preferred because they require merely differential calculus techniques or gradient search for $\hat{\boldsymbol{\theta}}$, rather than a possibly complex multidimensional integration needed in Bayesian estimation. This leads to another consideration: interpretability. In many cases the maximum-likelihood solution will be easier to interpret and understand because it returns the single best model from the set the designer provided (and presumably understands). In contrast, Bayesian methods give a weighted average of models (parameters), often leading to solutions more complicated and harder to understand than those provided by the designer. The Bayesian approach reflects the remaining uncertainty in the possible models.

Another consideration is our confidence in the prior information, such as in the *form* of the underlying distribution $p(\mathbf{x}|\boldsymbol{\theta})$. A maximum-likelihood solution $p(\mathbf{x}|\hat{\boldsymbol{\theta}})$ must of course be of the assumed parametric form; not so for the Bayesian solution. We saw this difference in Example 1, where the Bayes solution was not of the parametric form originally assumed, that is, a uniform $p(x|\mathcal{D})$. In general, through their use of the full $p(\boldsymbol{\theta}|\mathcal{D})$ distribution Bayesian methods use more of the information brought to the problem than do maximum-likelihood methods. (For instance, in Example 1 the addition of the third training point did not change the maximum-likelihood solution but did refine the Bayesian estimate.) If such information is reliable, Bayes methods can be expected to give better results. Furthermore, general Bayesian methods with a "flat" or uniform prior (i.e., where no prior information is explicitly imposed) are equivalent to maximum-likelihood methods. If there are much data, leading to a strongly peaked $p(\boldsymbol{\theta}|\mathcal{D})$, and the prior $p(\boldsymbol{\theta})$ is uniform or flat, then the MAP estimate is essentially the same as the maximum-likelihood estimate.

When $p(\boldsymbol{\theta}|\mathcal{D})$ is broad, or asymmetric around $\hat{\boldsymbol{\theta}}$, the methods are quite likely to yield $p(\mathbf{x}|\mathcal{D})$ distributions that differ from one another. Such a strong asymmetry (when not due to rare statistical irregularities in the selection of the training data) generally conveys some information about the distribution, just as did the asymmetric role of the threshold θ in Example 1. Bayes methods would exploit such information, whereas maximum-likelihood methods would not (at least not directly). Furthermore, Bayesian methods make more explicit the crucial problem of bias and variance tradeoffs—roughly speaking, the balance between the accuracy of the estimation and its variance, which depend upon the amount of training data. This important matter was irrelevant in Chapter 2, where there was no notion of a finite training set, but it will be crucial in our considerations of the theory of machine learning in Chapter 9.

When designing a classifier by either of these methods, we determine the posterior densities for each category and also classify a test point by the maximum posterior. (If there are costs, summarized in a cost matrix, these can be incorporated as well.) There are three sources of classification error in our final system:

Bayes or Indistinguishability Error: The error due to overlapping densities $p(\mathbf{x}|\omega_i)$ for different values of i. This error is an inherent property of the problem and can never be eliminated.

Model Error: The error due to having an incorrect model. This error can only be eliminated if the designer specifies a model that includes the true model which generated the data. Designers generally choose the model based on knowledge of the problem domain rather than on the subsequent estimation method, and thus the model error in maximum-likelihood and Bayes methods rarely differ.

Estimation Error: The error arising from the fact that the parameters are estimated from a finite sample. This error can best be reduced by increasing the training data.

The relative contributions of these sources depend upon problem, of course. In the limit of infinite training data, the estimation error vanishes, and the total classification error will be the same for both maximum-likelihood and Bayes methods.

In summary, there are strong theoretical and methodological arguments supporting Bayesian estimation, though in practice maximum-likelihood estimation is simpler and, when used for designing classifiers, can lead to classifiers nearly as accurate.

3.5.2 Noninformative Priors and Invariance

Generally speaking, the information about the prior $p(\boldsymbol{\theta})$ derives from the designer's knowledge of the problem domain and as such is beyond our study of the design of classifiers. Nevertheless, in some cases we have guidance in how to create priors that do not impose structure when we believe none exists, and this leads us to the notion of noninformative priors.

Recall our discussion of the role of prior category probabilities in Chapter 2, where, in the absence of other information, we assumed each of c categories equally likely. Analogously, in a Bayesian framework we can have a "noninformative" prior over a parameter for a single category's distribution. Suppose for instance that we are using Bayesian methods to infer from data some position and scale parameters, which we denote μ and σ, which might be the mean and standard deviation of a Gaussian, or the position and width of a triangle distribution, and so on. What prior might we put on these parameters? Consider first the location parameter. Surely the prior distribution should not depend upon our arbitrary choice of origin, that is, we demand *translation invariance*. The only prior to have this property is the uniform