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 applications, 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 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 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.

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 loosing 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." 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.

PRE-PROCESSING

SEGMENTATION

FEATURE EXTRACTION

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, (somehow) 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 — the average lightness of the fish scales. Now we are very careful to eliminate variations in illumination, since they can only obscure the models and corrupt our new classifier. The resulting histograms, shown in Fig. 1.3, are much more satisfactory — the classes are much better separated.

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 decision boundary to avoid antagonizing our customers, even if it means that more salmon makes its way into the cans of sea bass. In this case, then, we should move our decision boundary x^* 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

COST

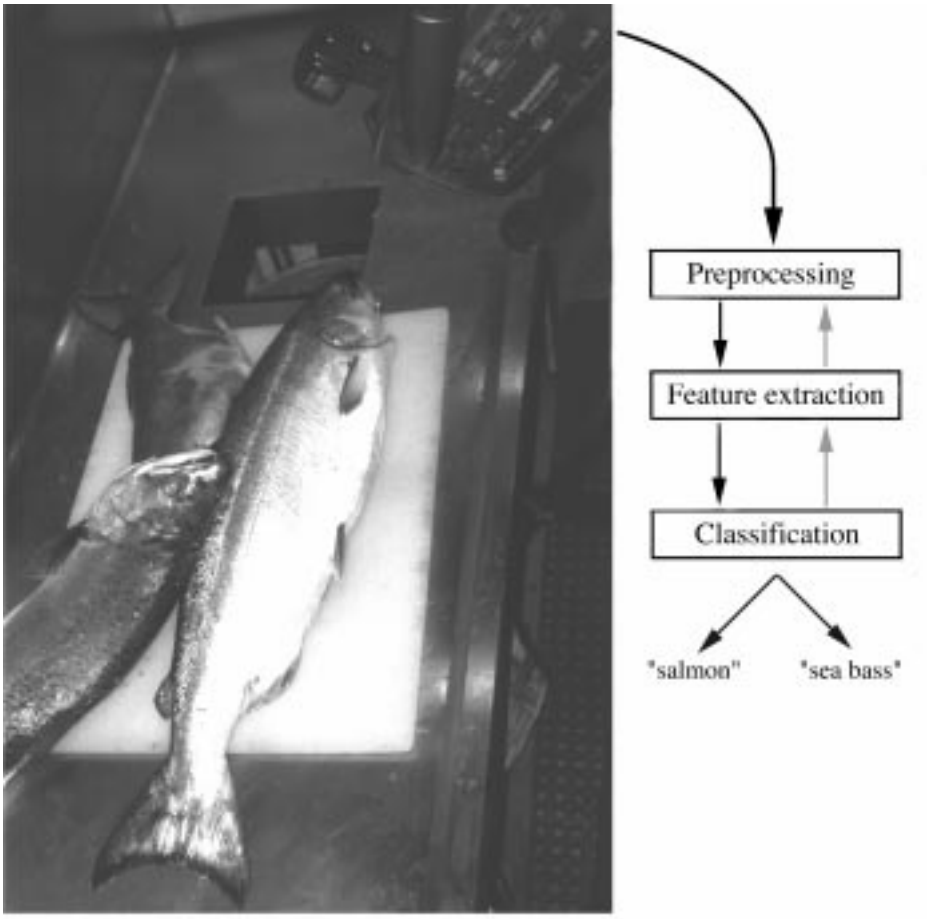


Figure 1.1: The objects to be classified are first sensed by a transducer (camera), whose signals are preprocessed, then the features extracted and finally the classification emitted (here either "salmon" or "sea bass"). Although the information flow is often chosen to be from the source to the classifier ("bottom-up"), some systems employ "top-down" flow as well, 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.

salmon — i.e., the more costly this type of error — the lower we should set the decision threshold x^* in Fig. 1.3.

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. DECISION THEORY

Even if we know the costs associated with our decisions and choose the optimal decision boundary 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, though, that no other single visual feature yields better performance than that based on lightness. To improve recognition, then, we must resort to the use

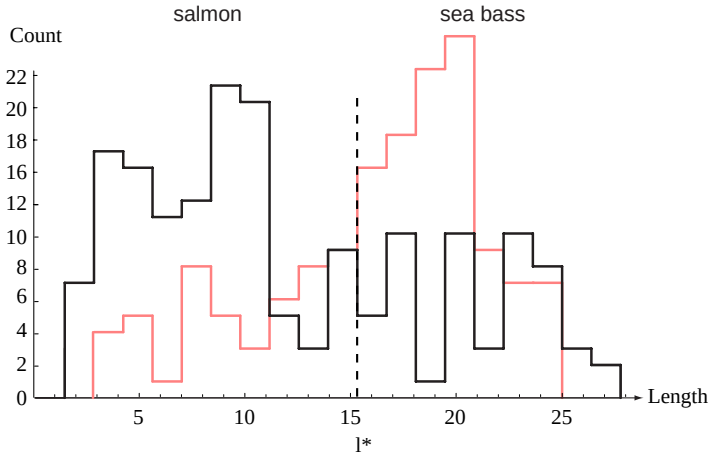


Figure 1.2: Histograms for the length feature for the two categories. No single threshold value l^* (decision boundary) will serve to unambiguously discriminate between the two categories; using length alone, we will have some errors. The value l^* marked will lead to the smallest number of errors, on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

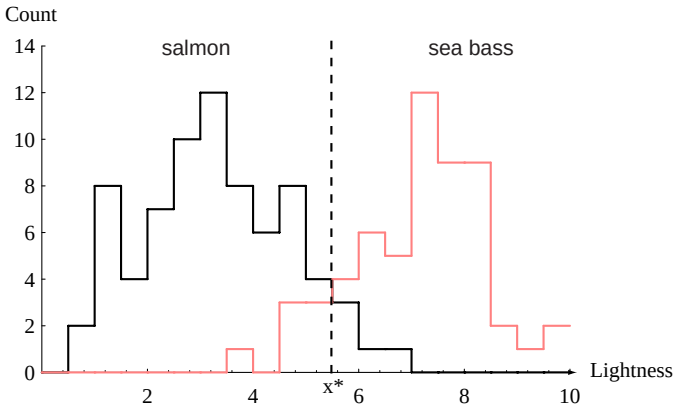


Figure 1.3: Histograms for the lightness feature for the two categories. No single threshold value x^* (decision boundary) will serve to unambiguously discriminate between the two categories; using lightness alone, we will have some errors. The value x^* marked will lead to the smallest number of errors,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

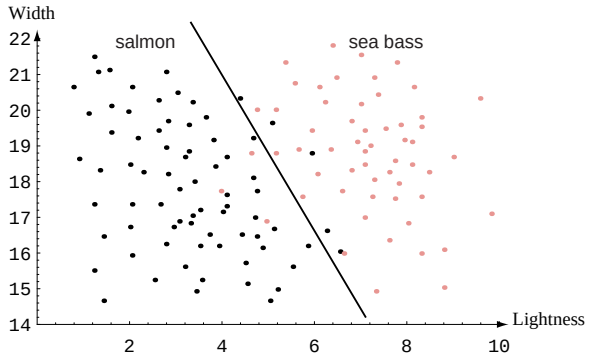


Figure 1.4: The two features of lightness and width for sea bass and salmon. The dark line might serve as a decision boundary of our classifier. Overall classification error on the data shown is lower than if we use only one feature as in Fig. 1.3, but there will still be some errors.

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 patterns in one region we will call the fish a sea bass, and 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 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?

Suppose that other features are too expensive or 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

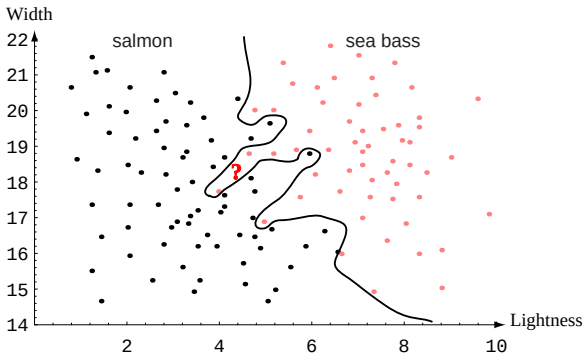


Figure 1.5: Overly complex models for the fish will lead to decision boundaries that are complicated. While such a decision may lead to perfect classification of our training samples, it would lead to poor performance on future patterns. The novel test point marked ? is evidently most likely a salmon,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

GENERALIZATION

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, i.e., 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, since 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.

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 most pattern 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 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 cost

* 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.

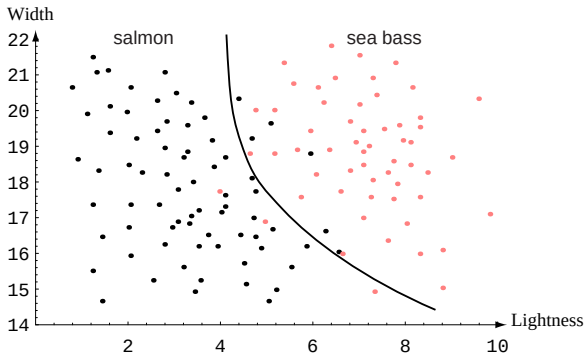


Figure 1.6: The decision boundary shown might represent the optimal tradeoff between performance on the training set and simplicity of classifier.

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 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 — i.e., 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.

Since 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, and here statistical descriptions (word frequencies, word correlations, etc.) are inapapropriate.

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 is 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 represen-

tation 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 simpler decision regions, and a classifier easier to train. We might also wish to have features that are robust, i.e., 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 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 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 insure 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 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. For typical values of this criterion, there is a strong bias or predilection in favor of the null hypothesis; even though the alternate hypothesis may be more probable, we might not be able to reject the null hypothesis. 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 (the null hypothesis) or from two classes (the alternative). In contrast, given some data, pattern classification seeks to find the most probable hypothesis from a set of hypotheses — "this fish is probably a salmon."

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, lose information (but hopefully preserve 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. 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.)

1.3 The Sub-problems of Pattern Classification

We have alluded to some of the issues in pattern classification and we now turn to a more explicit list of them. In practice, these typically require the bulk of the research and development effort. Many are domain or problem specific, and their solution will depend upon the knowledge and insights of the designer. Nevertheless, a few are of sufficient generality, difficulty, and interest that they warrant explicit consideration.

1.3.1 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. Generally speaking, 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

surely be of little use for identifying fingerprints, or classifying photomicrographs of blood cells. How do we know which features are most promising? Are there ways to automatically learn which features are best for the classifier? How many shall we use?

1.3.2 Noise

The lighting of the fish may vary, there could be shadows cast by neighboring equipment, the conveyor belt might shake — all reducing the reliability of the feature values actually measured. We define *noise* very general terms: any property of the sensed pattern due not to the true underlying model but instead to randomness in the world or the sensors. All non-trivial decision and pattern recognition problems involve noise in some form. In some cases it is due to the transduction in the signal and we may consign to our preprocessor the role of cleaning up the signal, as for instance visual noise in our video camera viewing the fish. An important problem is knowing somehow whether the variation in some signal is noise or instead to complex underlying models of the fish. How then can we use this information to improve our classifier?

1.3.3 Overfitting

In going from Fig 1.4 to Fig. 1.5 in our fish classification problem, we were, implicitly, using a more complex model of sea bass and of salmon. That is, we were adjusting the complexity of our classifier. While an overly complex model may allow perfect classification of the training samples, it is unlikely to give good classification of novel patterns — a situation 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?

1.3.4 Model Selection

We might have been unsatisfied with the performance of our fish classifier in Figs. 1.4 & 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? Can we automate the process?

1.3.5 Prior Knowledge

In one limited sense, we have already seen how 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.

1.3.6 Missing Features

Suppose that during classification, the value of one of the features cannot be determined, for example the width of the fish because of *occlusion* by another fish (i.e., the other fish is in the way). How should the categorizer compensate? Since our two-feature recognizer never had a single-variable threshold 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 naive method, of merely assuming that the value of the missing feature is zero or the average of the values for the training patterns, is provably non-optimal. Likewise we occasionally have missing features during the creation or learning in our recognizer. How should we train a classifier or use one when some features are missing?

OCCLUSION

1.3.7 Mereology

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?

This is the problem of *subsets and supersets* — formally part of mereology, the study of part/whole relationships. It is closely related to that of prior knowledge and segmentation. In short, how do we recognize or group together the "proper" number of elements — neither too few nor too many? 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?

1.3.8 Segmentation

In our fish example, we have tacitly assumed that the fish were isolated, separate on the conveyor belt. In practice, they 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 them. 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 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, as we saw in subsets and

supersets (Sect. 1.3.7). A related problem occurs in connected cursive handwritten character recognition: How do we know where one character "ends" and the next one "begins"?

1.3.9 Context

We might be able to use *context* — input-dependent information other than from the target pattern itself — to improve our recognizer. For instance, it might be known for our fish packing plant that if we are getting a sequence of salmon, that it is highly likely that the next fish will be a salmon (since it probably comes from a boat that just returned from a fishing area rich in salmon). Thus, if after a long series of salmon our recognizer detects an ambiguous pattern (i.e., one very close to the nominal decision boundary), it may nevertheless be best to categorize it too as a salmon. We shall see how such a simple correlation among patterns — the most elementary form of context — might be used to improve recognition. But how, precisely, should we incorporate such information?

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 speech recognition?

1.3.10 Invariances

In seeking to achieve an optimal representation for a particular pattern classification task, we confront the problem of *invariances*. In our fish example, the absolute position on the conveyor belt is irrelevant to the category and thus our representation should also be insensitive to absolute position of the fish. Here we seek a representation that is invariant to the transformation of *translation* (in either horizontal or vertical directions). Likewise, in a speech recognition problem, it might be required only that we be able to distinguish between utterances regardless of the particular moment they were uttered; here the "translation" invariance we must ensure is in *time*.

The "model parameters" describing the orientation of our fish on the conveyor belt are horrendously complicated — due as they are to the sloshing of water, the bumping of neighboring fish, the shape of the fish net, etc. — and thus we give up hope of ever trying to use them. These parameters are irrelevant to the model parameters that interest us anyway, i.e., the ones associated with the differences between the fish categories. Thus here we try to build a classifier that is invariant to transformations such as rotation.

ORIENTATION The orientation of the fish on the conveyor belt is irrelevant to its category. Here the transformation of concern is a two-dimensional rotation about 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 hidden, the bottom of the inside volume come into view, the circular lip appear oval or a straight line or even obscured, and so forth. How might we insure that our pattern recognizer is invariant to such complex changes?

SIZE The overall size of an image may be irrelevant for categorization. Such differences might be due to variation in the range to the object; alternatively we may be genuinely unconcerned with differences between sizes — a young, small salmon is still a salmon.

For patterns that have inherent temporal variation, we may want our recognizer to be insensitive to the *rate* at which the pattern evolves. Thus a slow hand wave and a fast hand wave may be considered as equivalent. Rate variation is a deep problem in speech recognition, of course; not only do different individuals 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?

RATE

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 *non-rigid 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

The symmetries just described are continuous — the pattern can be translated, rotated, sped up, or deformed by an arbitrary amount. In some pattern recognition applications other — *discrete* — symmetries are relevant, such as flips left-to-right, or top-to-bottom.

DISCRETE SYMMETRY

In all of these invariances the problem arises: How do we determine whether an invariance is present? How do we efficiently incorporate such knowledge into our recognizer?

1.3.11 Evidence Pooling

In our fish example we saw how using *multiple* features could lead to improved recognition. We might imagine that we could do better if we had several component *classifiers*. If these categorizers 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 if a particular fish is diseased or not. 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?

1.3.12 Costs and Risks

We should realize that 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 or risk. Conceptually, the simplest such risk is the classification error: what percentage of new patterns are called the wrong category. However the notion of risk is far more general, as we shall see. We often design our classifier to recommend actions that minimize some total expected cost or risk. Thus, in some sense, the notion of category itself derives from the cost or task. How do we incorporate knowledge about such risks and how will they affect our classification decision?

Finally, 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?

1.3.13 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 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 complexity of different algorithms is of importance, especially for practical applications.

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 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.

This section has catalogued some of the central problems in classification. It has been found that the most effective methods for developing classifiers involve learning from examples, i.e., from a set of patterns whose category is known. Throughout this book, we shall see again and again how methods of learning relate to these central problems, and are essential in the building of classifiers.

1.4 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 classification decision ahead of time, we shall spend the great majority of our time here considering learning. Creating classifiers then involves posit 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.

1.4.1 Supervised Learning

In supervised learning, a teacher provides a category label or cost for each pattern in a training set, and we seek 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 scale reasonably with the number of training patterns, the number of input features or with the perplexity of the problem? How can we insure that the learning algorithm appropriately favors "simple" solutions (as in Fig. 1.6) rather than complicated ones (as in Fig. 1.5)?

1.4.2 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?

1.4.3 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. (Thus only binary feedback is given to the classifier; reinforcement learning also describes the case where a single scalar signal, say some number between 0 and 1, is given by the teacher.) In pattern classification, it is most common that such reinforcement is binary — either the tentative decision is correct or it is not. (Of course, if our problem involves just two categories and equal costs for errors, then learning with a critic is equivalent to standard supervised learning.) How can the system learn which are important from such non-specific feedback?

CRITIC

1.5 Conclusion

At this point the reader may be overwhelmed by the number, complexity and magnitude of these sub-problems. Further, these sub-problems 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, though, that the good news is at least three-fold: 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

The overall organization of this book is to address first those cases where a great deal of information about the models is known (such as the probability densities, category labels, ...) and to move, chapter by chapter, toward problems where the form of the

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 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 Sect. **??**, we begin our discussion with a specific example. Let us reconsider the hypothetical problem posed in Chap. **??** 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 $p(x|\omega_1)$.* This is the *class-conditional probability density* function. Strictly speaking, the probability density function $p(x|\omega_1)$ should be written as $p_X(x|\omega_1)$ 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 $p(y)$. Since 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 Appendix **??**). This is the probability density function for x given that the state of nature is ω_1. (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).

Suppose that we know both the prior probabilities $P(\omega_j)$ and the conditional densities $p(x|\omega_j)$. 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 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*:

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

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

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

* We generally use an upper-case $P(\cdot)$ to denote a probability mass function and a lower-case $p(\cdot)$ to denote a probability density function.

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 psterior 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 $P(\omega_2) = 1/3$.

POSTERIOR

LIKELIHOOD

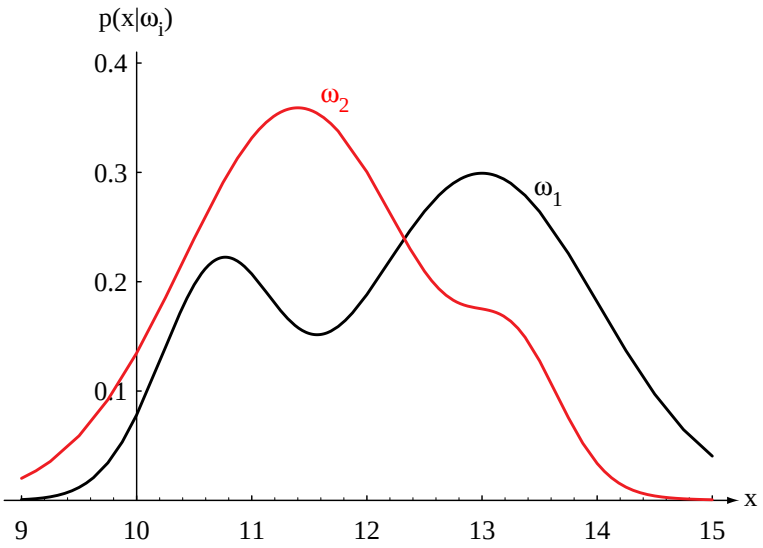


Figure 2.1: Hypothetical class-conditional probability density functions show the probability density of measuring a particular feature value x given the pattern is in category ω_i. If x represents the length of a fish, the two curves might describe the difference in length of populations of two types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

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. Similarly, 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,

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

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

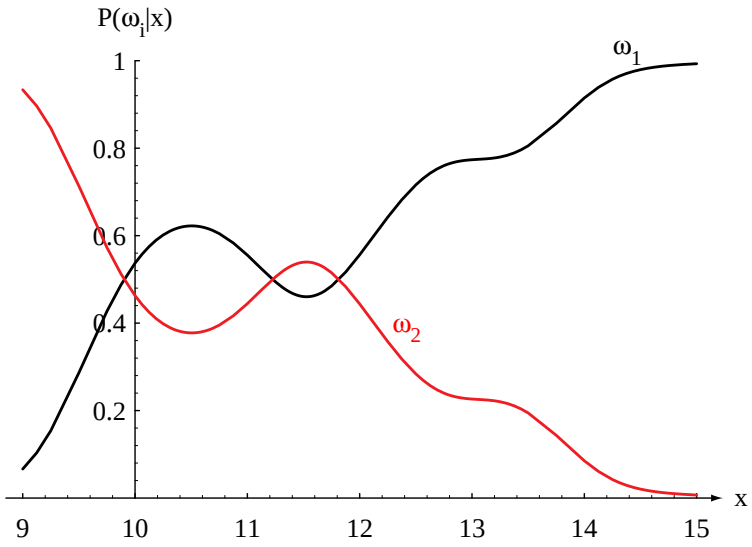


Figure 2.2: Posterior probabilities for the particular priors $P(\omega_1) = 2/3$ and $P(\omega_2) = 1/3$ for the class-conditional probability densities shown in Fig. 2.1. Thus in this case, given that a pattern is measured to have feature value $x = 14$, the probability it is in category ω_2 is roughly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

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

and if for every x we insure 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:

BAYES' DECISION RULE

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

and under this rule Eq. 4 becomes

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

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; 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:

EVIDENCE

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

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 $\mathbf{x}$ 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, i.e., of refusing to make a decision in close cases; this is a 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.

FEATURE SPACE

LOSS FUNCTION

Let $\omega_1, ..., \omega_c$ be the finite set of c states of nature ("categories") and $\alpha_1, ..., \alpha_a$ 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 $\mathbf{x}$ — 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 $\lambda(\alpha_i|\omega_j)$. Since $P(\omega_j|\mathbf{x})$ 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 on an overall risk.

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 $\alpha_1, ..., \alpha_a$. The overall risk R is the expected loss associated with a given decision rule. Since $R(\alpha_i|\mathbf{x})$ is the conditional risk associated with action α_i, and since 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

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

for $i = 1,...,a$ and 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.

BAYES RISK

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

$$\begin{aligned} R(\alpha_1|\mathbf{x}) &= \lambda_{11} P(\omega_1|\mathbf{x}) + \lambda_{12} P(\omega_2|\mathbf{x}) \qquad \text{and} \\ R(\alpha_2|\mathbf{x}) &= \lambda_{21} P(\omega_1|\mathbf{x}) + \lambda_{22} P(\omega_2|\mathbf{x}). \end{aligned} \tag{14}$$

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

$$(\lambda_{21} - \lambda_{11}) P(\omega_1|\mathbf{x}) > (\lambda_{12} - \lambda_{22}) P(\omega_2|\mathbf{x}). \tag{15}$$

* 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.

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 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

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

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

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

This form of the decision rule focuses on the $\mathbf{x}$-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 $\mathbf{x}$.

LIKELIHOOD RATIO

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, i.e., the *error rate*.

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

ZERO-ONE LOSS

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases} \qquad i, j = 1, ..., c. \tag{18}$$

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, since the conditional risk is

$$\begin{aligned} R(\alpha_i|\mathbf{x}) &= \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j)P(\omega_j|\mathbf{x}) \\ &= \sum_{j \neq i} P(\omega_j|\mathbf{x}) \\ &= 1 - P(\omega_i|\mathbf{x}) \end{aligned} \tag{19}$$

* 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.

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*:

$$\text{Decide } \omega_i \text{ if } P(\omega_i|\mathbf{x}) > P(\omega_j|\mathbf{x}) \qquad \text{for all } j \neq i. \tag{20}$$

This is the same rule as in Eq. 6.

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 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. 17 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.

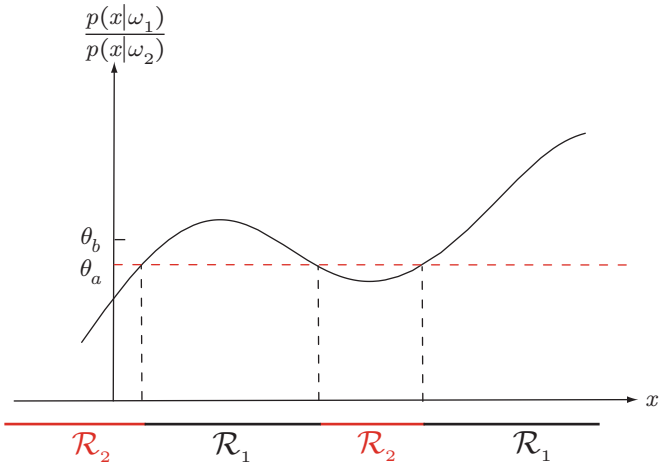


Figure 2.3: The likelihood ratio $p(x|\omega_1)/p(x|\omega_2)$ for the distributions shown in Fig. 2.1. If we employ a zero-one or classification loss, our decision boundaries are determined by the threshold θ_a. If our loss function penalizes miscategorizing ω_2 as ω_1 patterns more than the converse, (i.e., $\lambda_{12} > \lambda_{21}$), we get the larger threshold θ_b, and hence $\mathcal{R}_1$ becomes smaller.

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 write our overall risk Eq. 12 in terms of conditional risks:

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

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:

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

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. 22:

MINIMAX RISK

$$\begin{aligned} R_{mm} &= \lambda_{22} + (\lambda_{12} - \lambda_{22}) \int_{\mathcal{R}_1} p(\mathbf{x}|\omega_2)\ d\mathbf{x} \\ &= \lambda_{11} + (\lambda_{21} - \lambda_{11}) \int_{\mathcal{R}_2} p(\mathbf{x}|\omega_1)\ d\mathbf{x}. \end{aligned} \tag{23}$$

Figure 2.4 illustrates the approach. Briefly stated, we search for the prior for which the Bayes risk is *maximum*, 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 3).

The minimax criterion finds greater use in game theory then 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.

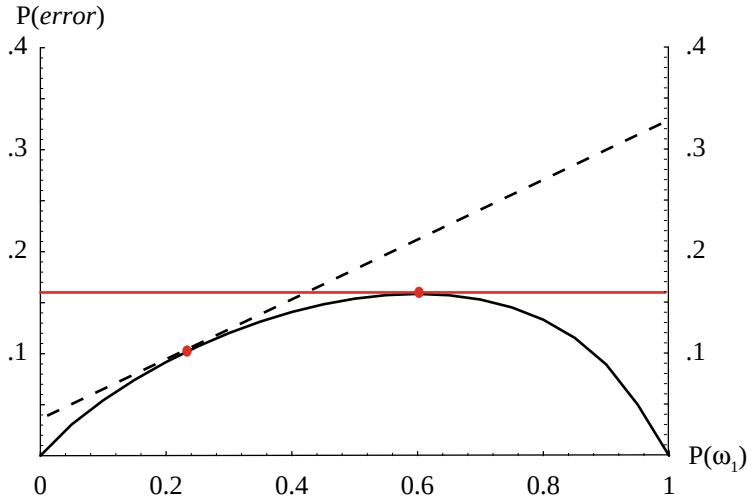


Figure 2.4: The curve at the bottom shows the minimum (Bayes) error as a function of prior probability $P(\omega_1)$ in a two-category classification problem of fixed distributions. For each value of the priors (e.g., $P(\omega_1) = 0.25$) there is a corresponding optimal decision boundary and associated 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.

2.3.2 *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 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 5 & 6). We shall have cause to mention Neyman-Pearson criteria again in Sect. 2.8.3 on operating characteristics.

2.4 Classifiers, Discriminant Functions and Decision Surfaces

2.4.1 The Multi-Category Case

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

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \qquad \text{for all } j \neq i. \tag{24}$$

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.

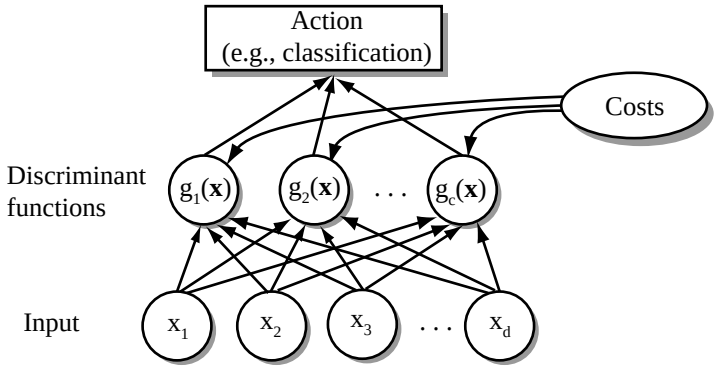


Figure 2.5: The functional structure of a general statistical pattern classifier which includes d inputs and c discriminant functions $g_i(\mathbf{x})$. A subsequent step determines which of the discriminant values is the maximum, and categorizes the input pattern accordingly. The arrows show the direction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

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

$$g_i(\mathbf{x}) = P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}|\omega_j)P(\omega_j)} \tag{25}$$

$$g_i(\mathbf{x}) = p(\mathbf{x}|\omega_i)P(\omega_i) \tag{26}$$

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

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*, $\mathcal{R}_1$,...,$\mathcal{R}_c$. 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

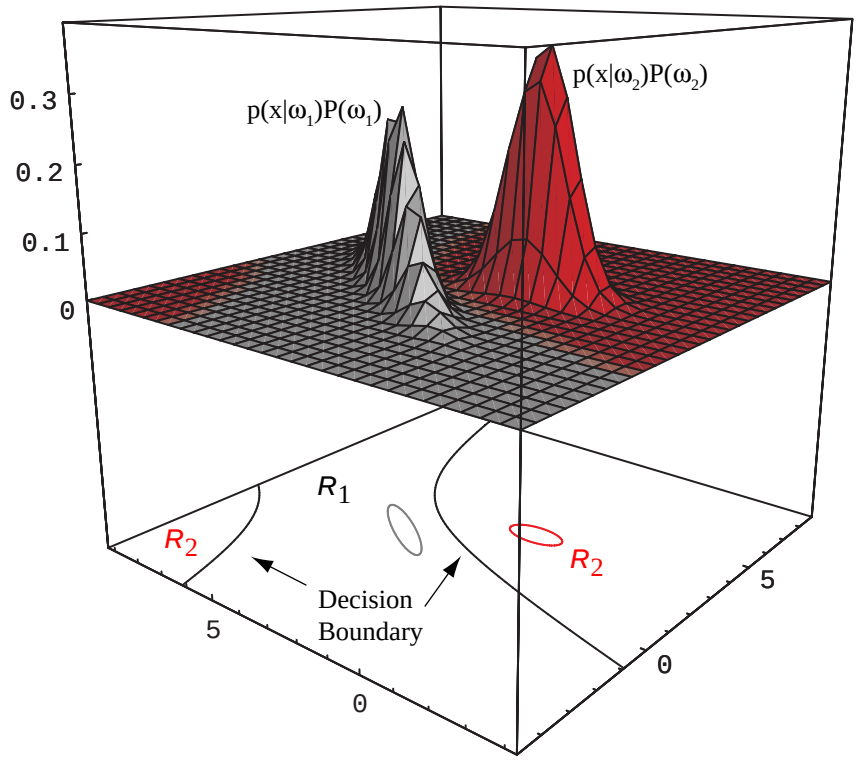


Figure 2.6: In this two-dimensional two-category classifier, the probability densities are Gaussian (with $1/e$ ellipses shown), the decision boundary consists of two hyperbolas, and thus the decision region $\mathcal{R}_2$ is not simply connected.

2.4.2 The Two-Category Case

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 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 DICHOTOMIZER

$$g(\mathbf{x}) \equiv g_1(\mathbf{x}) - g_2(\mathbf{x}), \tag{28}$$

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. 25 & 27) are particularly convenient:

$$g(\mathbf{x}) = P(\omega_1|\mathbf{x}) - P(\omega_2|\mathbf{x}) \tag{29}$$

$$g(\mathbf{x}) = \ln\frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} + \ln\frac{P(\omega_1)}{P(\omega_2)}. \tag{30}$$

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. 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, viz., 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.

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

$$\mathcal{E}[f(x)] \equiv \int_{-\infty}^{\infty} f(x)p(x)dx. \tag{31}$$

If we have samples in a set $\mathcal{D}$ from a discrete distribution, we must sum over all samples as

$$\mathcal{E}[f(x)] = \sum_{x\in\mathcal{D}} f(x)P(x), \tag{32}$$

where $P(x)$ is the probability mass at x. We shall often have call to calculate expected values — by these and analogous equations defined in higher dimensions (see Appendix Secs. **??**, **??** & **??**).*

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

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

2.5.1 Univariate Density

We begin with the continuous univariate normal or Gaussian density,

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right], \tag{33}$$

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

$$\mu \equiv \mathcal{E}[x] = \int_{-\infty}^{\infty} x p(x)\, dx, \tag{34}$$

VARIANCE

and where the expected squared deviation or *variance* is

$$\sigma^2 \equiv \mathcal{E}[(x-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 p(x)\, dx. \tag{35}$$

MEAN

The univariate normal density is completely specified by two parameters: its mean μ and variance σ^2. For simplicity, we often abbreviate Eq. 33 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).

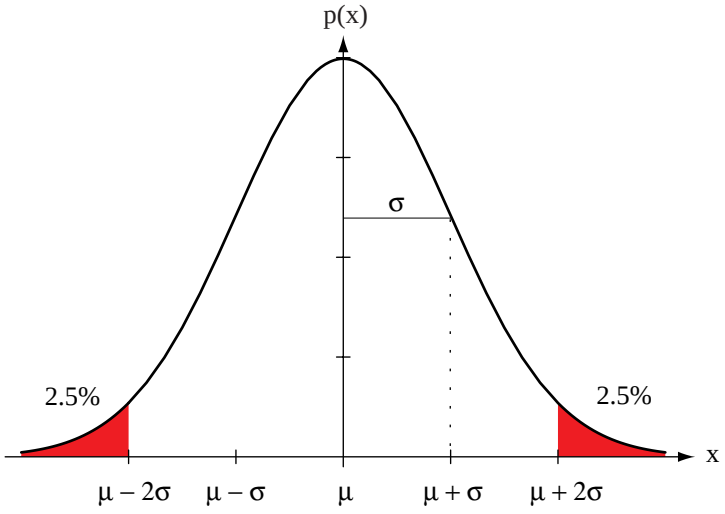


Figure 2.7: A univariate normal distribution has roughly 95% of its area in the range $|x-\mu| \leq 2\sigma$, as shown. The peak of the distribution has value $p(\mu) = 1/\sqrt{2\pi}\sigma$.

ENTROPY

There is a deep relationship between the normal distribution and *entropy*. We shall consider entropy in greater detail in Chap. **??**, but for now we merely state that the entropy of a distribution is given by

$$H(p(x)) = -\int p(x) \ln p(x)\, dx, \tag{36}$$

NAT

and measured in *nats*. If a $\log_2$ is used instead, the unit is the *bit*. The entropy is a non-negative quantity that describes the fundamental uncertainty in the values of points

BIT

selected randomly from a 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 a large number of small, independent random disturbances will lead to a Gaussian distribution (Computer exercise **??**). 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.

CENTRAL LIMIT THEOREM

2.5.2 Multivariate Density

The general multivariate normal density in d dimensions is written as

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right], \tag{37}$$

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

COVARIANCE MATRIX

INNER PRODUCT

$$\mathbf{a}^t\mathbf{b} = \sum_{i=1}^{d} a_i b_i, \tag{38}$$

and often called a *dot product*.

For simplicity, we often abbreviate Eq. 37 as $p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \mathbf{\Sigma})$. Formally, we have

$$\boldsymbol{\mu} \equiv \mathcal{E}[\mathbf{x}] = \int \mathbf{x} p(\mathbf{x})\ d\mathbf{x} \tag{39}$$

and

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

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 $\mathbf{\Sigma}$, then

$$\mu_i = \mathcal{E}[x_i] \tag{41}$$

and

$$\sigma_{ij} = \mathcal{E}[(x_i-\mu_i)(x_j-\mu_j)]. \tag{42}$$

The covariance matrix $\mathbf{\Sigma}$ is always symmetric and positive semidefinite. We shall restrict our attention to the case in which $\mathbf{\Sigma}$ is positive definite, so that the determinant of $\mathbf{\Sigma}$ 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*, $\sigma_{ij} = 0$. If

COVARIANCE

STATISTICAL INDEPENDENCE

* 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 **??**.

† If sample vectors are drawn from a linear subspace, $|\mathbf{\Sigma}| = 0$ 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.

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 $\mathbf{A}$ 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, i.e., 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 $\mathbf{A}_w = \boldsymbol{\Phi}\boldsymbol{\Lambda}^{-1/2}$ applied to the coordinates insures that the transformed distribution has covariance matrix equal to the identity matrix. In signal processing, the transform $\mathbf{A}_w$ is called a *whitening* transformation, since it makes the spectrum of eigenvectors of the transformed distribution uniform.

WHITENING TRANSFORM

The multivariate normal density is completely specified by $d + d(d+1)/2$ parameters — 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. If follows from Eq. 37 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

$$r^2 = (\mathbf{x} - \boldsymbol{\mu})^t\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}) \tag{43}$$

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 & 16) that the volume of the hyperellipsoid corresponding to a Mahalanobis distance r is given by

MAHALANOBIS DISTANCE

$$V = V_d|\boldsymbol{\Sigma}|^{1/2}r^d, \tag{44}$$

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

$$V_d = \begin{cases} \pi^{d/2}/(d/2)! & d \text{ even} \\ 2^d\pi^{(d-1)/2}(\frac{d-1}{2})!/(d)! & d \text{ odd.} \end{cases} \tag{45}$$

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

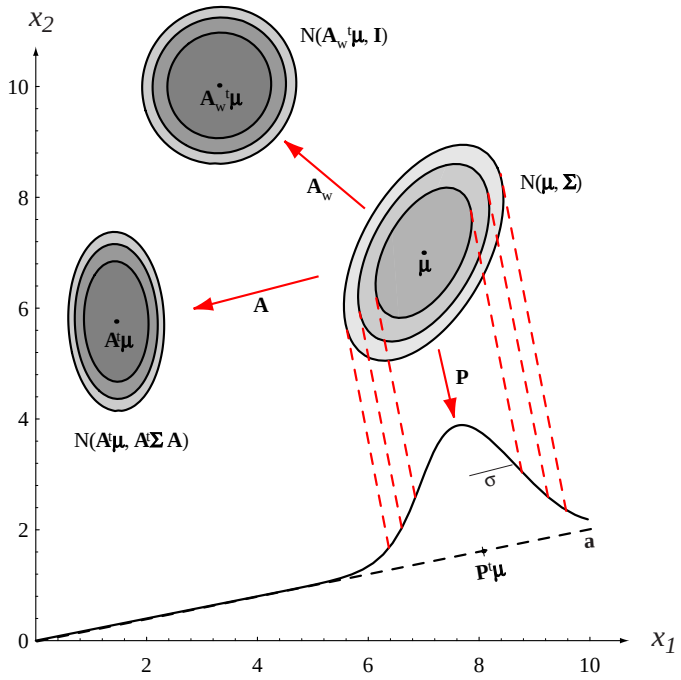


Figure 2.8: The action of a linear transformation on the feature space will convert an arbitrary normal distribution into another normal distribution. One transformation, $\mathbf{A}$, takes the source distribution into distribution $N(\mathbf{A}^t\boldsymbol{\mu}, \mathbf{A}^t\boldsymbol{\Sigma}\mathbf{A})$. Another linear transformation — a projection $\mathbf{P}$ onto line $\mathbf{a}$ — leads to $N(\mu, \sigma^2)$ measured along $\mathbf{a}$. While the transforms yield distributions in a different space, we show them superimposed on the original $x_1 - x_2$ space. A whitening transform leads to a circularly symmetric Gaussian, here shown displaced.

2.6 Discriminant Functions for the Normal Density

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

$$g_i(\mathbf{x}) = \ln\, p(\mathbf{x}|\omega_i) + \ln\, P(\omega_i). \tag{46}$$

This expression can be readily evaluated if the densities $p(\mathbf{x}|\omega_i)$ are multivariate normal, i.e., if $p(\mathbf{x}|\omega_i) \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. In this case, then, from Eq. 37 we have

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln |\boldsymbol{\Sigma}_i| + \ln\, P(\omega_i). \tag{47}$$

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,

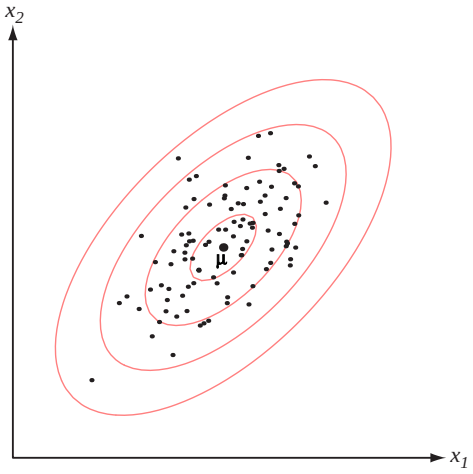


Figure 2.9: Samples drawn from a two-dimensional Gaussian lie in a cloud centered on the mean $\boldsymbol{\mu}$. The red ellipses show lines of equal probability density of the Gaussian.

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 $\boldsymbol{\Sigma}_i^{-1} = (1/\sigma^2)\mathbf{I}$. Since both $|\boldsymbol{\Sigma}_i|$ and the $(d/2)\ \ln\ 2\pi$ term in Eq. 47 are independent of i, they are unimportant additive constants that can be ignored. Thus we obtain the simple discriminant functions

$$g_i(\mathbf{x}) = -\frac{\|\mathbf{x} - \boldsymbol{\mu}_i\|^2}{2\sigma^2} + \ln\ P(\omega_i), \tag{48}$$

EUCLIDEAN NORM

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

$$\|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = (\mathbf{x} - \boldsymbol{\mu}_i)^t(\mathbf{x} - \boldsymbol{\mu}_i). \tag{49}$$

If the prior probabilities are not equal, then Eq. 48 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

$$g_i(\mathbf{x}) = -\frac{1}{2\sigma^2}[\mathbf{x}^t\mathbf{x} - 2\boldsymbol{\mu}_i^t\mathbf{x} + \boldsymbol{\mu}_i^t\boldsymbol{\mu}_i] + \ln\ P(\omega_i), \tag{50}$$

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

$$g_i(\mathbf{x}) = \mathbf{w}_i^t\mathbf{x} + w_{i0}, \tag{51}$$

where

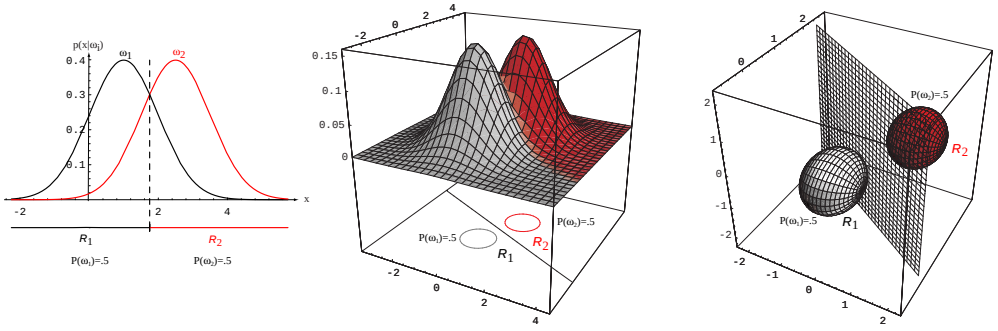


Figure 2.10: If the covariances of two distributions are equal and proportional to the identity matrix, then the distributions are spherical in d dimensions, and the boundary is a generalized hyperplane of $d-1$ dimensions, perpendicular to the line separating the means. In these 1-, 2-, and 3-dimensional examples, we indicate $p(\mathbf{x}|\omega_i)$ and the boundaries for the case $P(\omega_1) = P(\omega_2)$. In the 3-dimensional case, the grid plane separates $\mathcal{R}_1$ from $\mathcal{R}_2$.

$$\mathbf{w}_i = \frac{1}{\sigma^2}\boldsymbol{\mu}_i \tag{52}$$

and

$$w_{i0} = \frac{-1}{2\sigma^2}\boldsymbol{\mu}_i^t\boldsymbol{\mu}_i + \ln\, P(\omega_i). \tag{53}$$

We call w_{i0} the *threshold* or *bias* in the ith direction.

THRESHOLD

BIAS

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 detail in Chap. **??**. 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

LINEAR MACHINE

$$\mathbf{w}^t(\mathbf{x} - \mathbf{x}_0) = 0, \tag{54}$$

where

$$\mathbf{w} = \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \tag{55}$$

and

$$\mathbf{x}_0 = \frac{1}{2}(\boldsymbol{\mu}_i + \boldsymbol{\mu}_j) - \frac{\sigma^2}{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2}\ln\frac{P(\omega_i)}{P(\omega_j)}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j). \tag{56}$$

This equation defines a hyperplane through the point $\mathbf{x}_0$ and orthogonal to the vector $\mathbf{w}$. Since $\mathbf{w} = \boldsymbol{\mu}_i - \boldsymbol{\mu}_j$, 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. 56 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

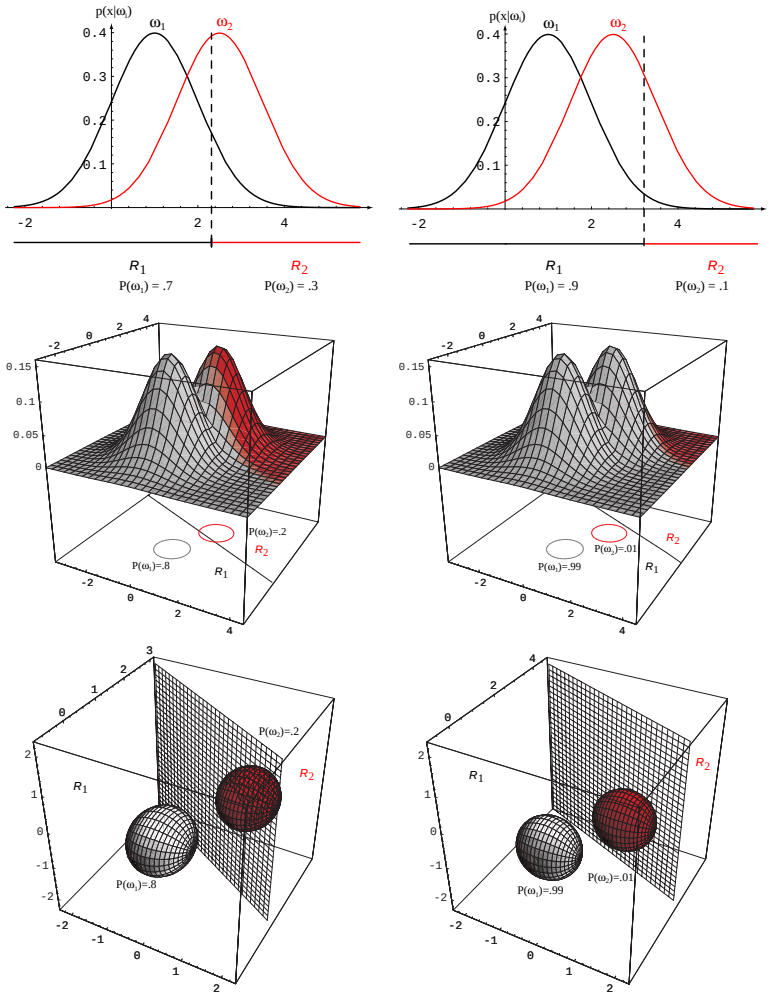


Figure 2.11: As the priors are changed, the decision boundary shifts; for sufficiently disparate priors the boundary will not lie between the means of these 1-, 2- and 3-dimensional spherical Gaussian distributions.

σ^2 is small relative to the squared distance $\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|$, then the position of the decision boundary is relatively insensitive to the exact values of the prior probabilities.

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 Chap. **??** Sect. **??** 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 $\boldsymbol{\mu}_i$. Since both $|\boldsymbol{\Sigma}_i|$ and the $(d/2)\ \ln 2\pi$ term in Eq. 47 are independent of i, they can be ignored as superfluous additive constants. This simplification leads to the discriminant functions

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) + \ln\ P(\omega_i). \tag{57}$$

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. 57, the resulting discriminant functions are again linear:

$$g_i(\mathbf{x}) = \mathbf{w}_i^t \mathbf{x} + w_{i0}, \tag{58}$$

where

$$\mathbf{w}_i = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_i \tag{59}$$

and

$$w_{i0} = -\frac{1}{2} \boldsymbol{\mu}_i^t \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_i + \ln\ P(\omega_i). \tag{60}$$

Since 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

$$\mathbf{w}^t(\mathbf{x} - \mathbf{x}_0) = 0, \tag{61}$$

where

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) \tag{62}$$

and

$$\mathbf{x}_0 = \frac{1}{2}(\boldsymbol{\mu}_i + \boldsymbol{\mu}_j) - \frac{\ln\ [P(\omega_i)/P(\omega_j)]}{(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^t \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j). \tag{63}$$

Since $\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$ 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 $\mathbf{x}_0$ which is halfway between the means if the prior probabilities are equal. 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.

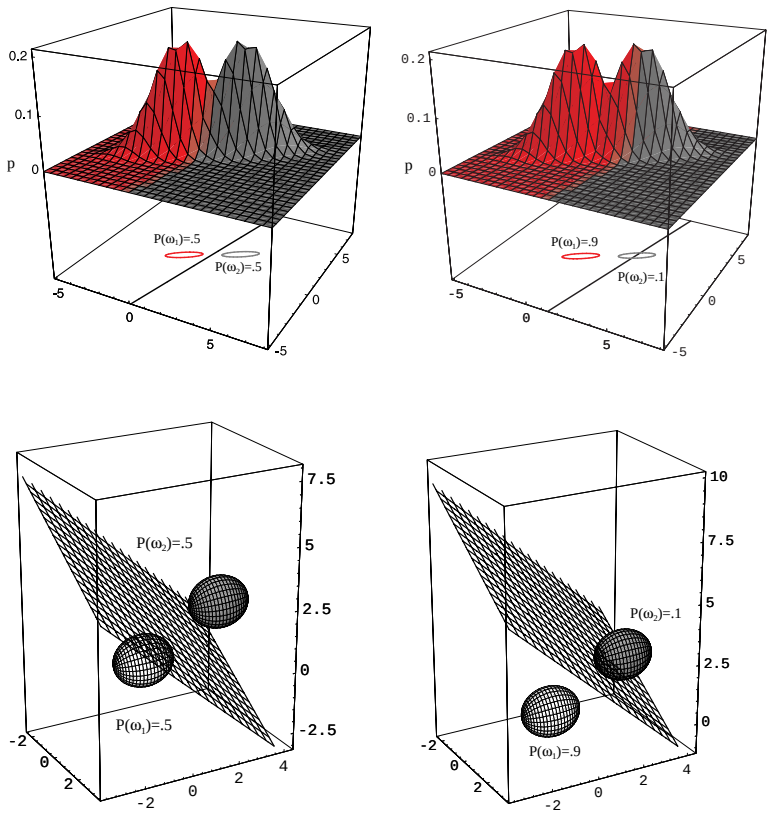


Figure 2.12: Probability densities (indicated by the surfaces in two dimensions and ellipsoidal surfaces in three dimensions) and decision regions for equal but asymmetric Gaussian distributions. The decision hyperplanes need not be perpendicular to the line connecting the means.

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. 47 is the $(d/2)\ln 2\pi$ term, and the resulting discriminant functions are inherently quadratic:

$$g_i(\mathbf{x}) = \mathbf{x}^t\mathbf{W}_i\mathbf{x} + \mathbf{w}_i^t\mathbf{x} + w_{i0}, \tag{64}$$

where

$$\mathbf{W}_i = -\frac{1}{2}\boldsymbol{\Sigma}_i^{-1}, \tag{65}$$

$$\mathbf{w}_i = \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i \tag{66}$$

and

$$w_{i0} = -\frac{1}{2}\boldsymbol{\mu}_i^t\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| + \ln P(\omega_i). \tag{67}$$

HYPER-QUADRIC

The decision surfaces are *hyperquadrics*, and can assume any of the general forms — hyperplanes, pairs of hyperplanes, hyperspheres, hyperellipsoids, hyperparaboloids, and hyperhyperboloids of various types (Problem 29). Even in one dimension, for arbitrary covariance the decision regions need not be simply connected (Fig. 2.13). The two- and three-dimensional examples in Fig. 2.14 & 2.15 indicate how these different forms can arise. These variances are indicated by the contours of constant probability density.

The extension of these results to more than two categories is straightforward though 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.

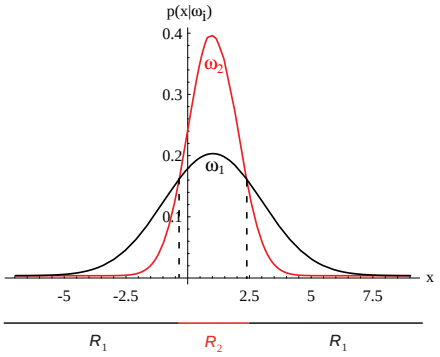


Figure 2.13: Non-simply connected decision regions can arise in one dimensions for Gaussians having unequal variance.

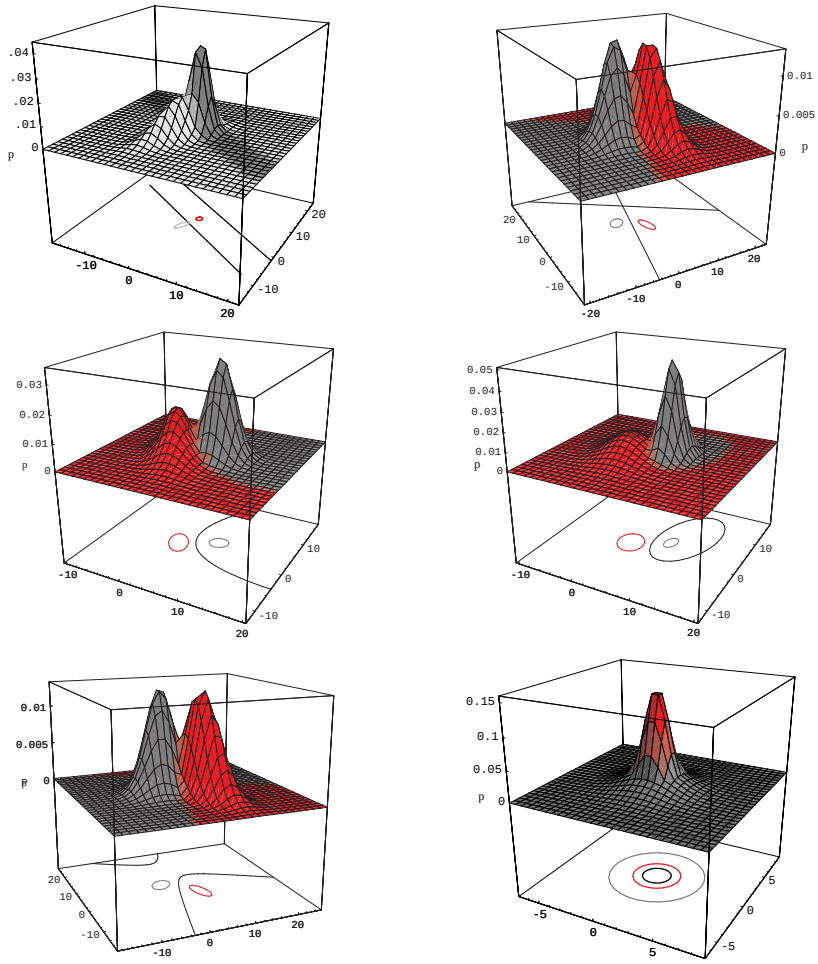


Figure 2.14: Arbitrary Gaussian distributions lead to Bayes decision boundaries that are general hyperquadrics. Conversely, given any hyperquadratic, one can find two Gaussian distributions whose Bayes decision boundary is that hyperquadric.

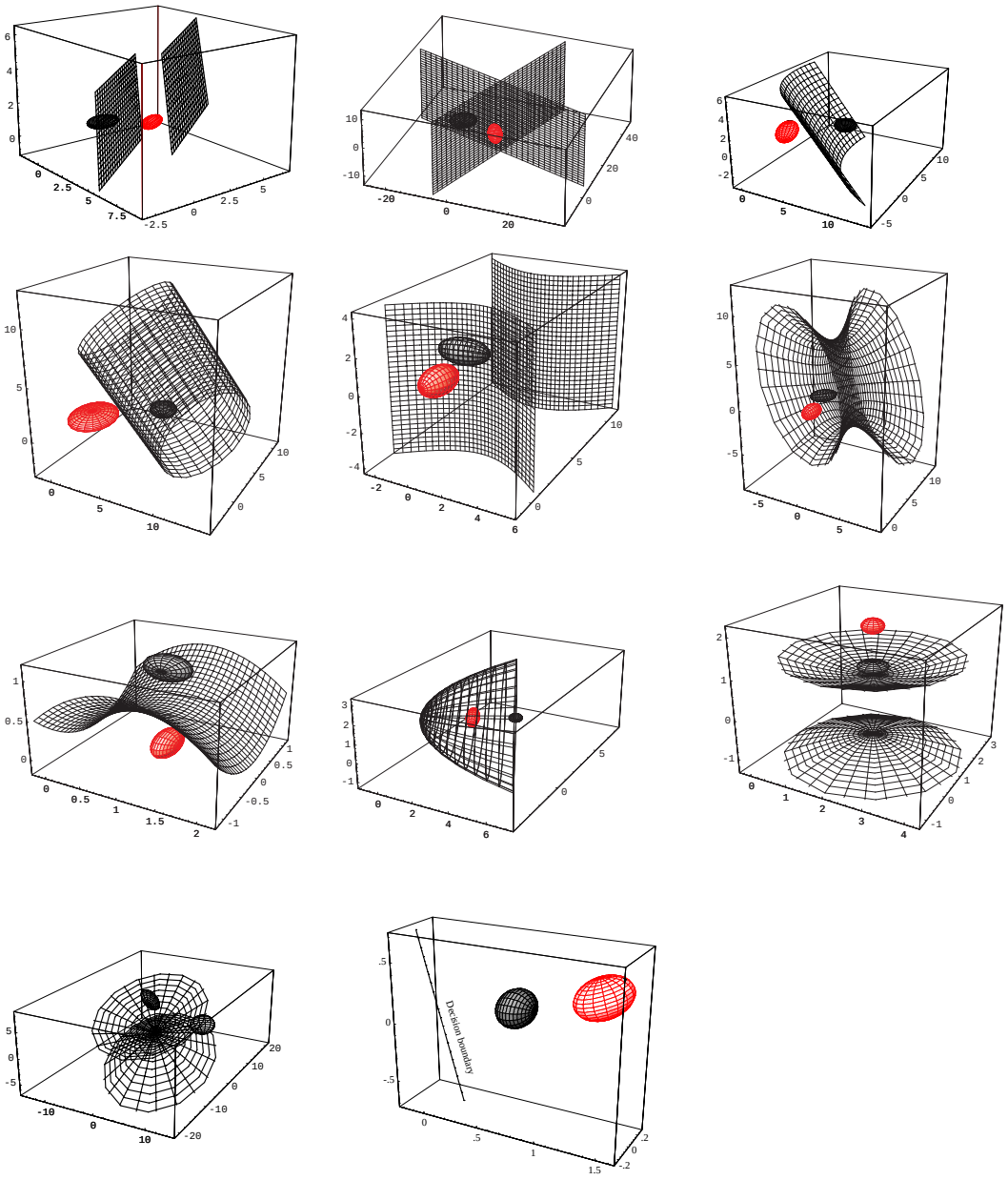


Figure 2.15: Arbitrary three-dimensional Gaussian distributions yield Bayes decision boundaries that are two-dimensional hyperquadrics. There are even degenerate cases in which the decision boundary is a line.

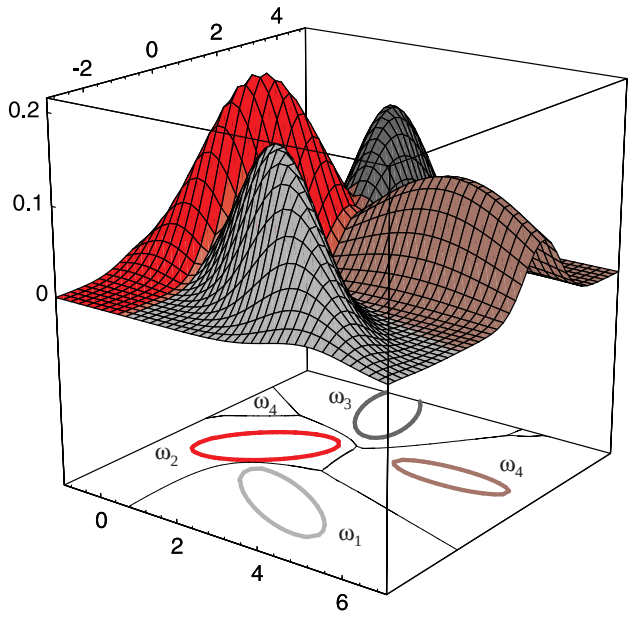


Figure 2.16: The decision regions for four normal distributions. Even with such a low number of categories, the shapes of the boundary regions can be rather complex.

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. 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. 39 & 40; they are found to be:

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

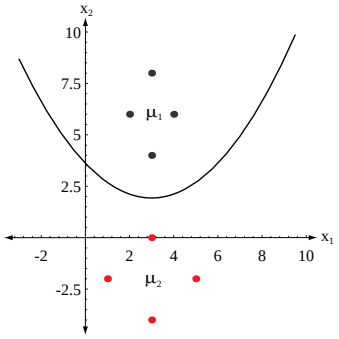
The inverse matrices are then,

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

We assume equal prior probabilities, $P(\omega_1) = P(\omega_2) = 0.5$, and substitute these into the general form for a discriminant, Eqs. 64 – 67, setting $g_1(\mathbf{x}) = g_2(\mathbf{x})$ to obtain the decision boundary:

$$x_2 = 3.514 - 1.125x_1 + 0.1875x_1^2.$$

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 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 x_1-direction more so than for the ω_2 distribution. Because the overall prior probabilities are the same (i.e., the integral over space of the probability density), the distribution is increased along the 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.



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

Chapter 3

Maximum likelihood and Bayesian parameter estimation

3.1 Introduction

In Chap. ?? 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 *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.

TRAINING DATA

One approach to this problem is to use the samples to estimate the unknown probabilities and probability densities, and to 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 $\boldsymbol{\Sigma}_i$.

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, *maximum likelihood* estimation and *Bayesian* estimation. Although the results obtained with these two procedures are frequently nearly identical, the approaches

MAXIMUM LIKELIHOOD

BAYESIAN ESTIMATION

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 a priori 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.

BAYESIAN LEARNING

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 Chap. **??**.

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. Further, maximum likelihood estimation often can be simpler than alternate 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 sets, $\mathcal{D}_1, ..., \mathcal{D}_c$, 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 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 $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_c$ 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, $\mathbf{x}_1, ..., \mathbf{x}_n$. Then, since 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 Chap. **??** 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).

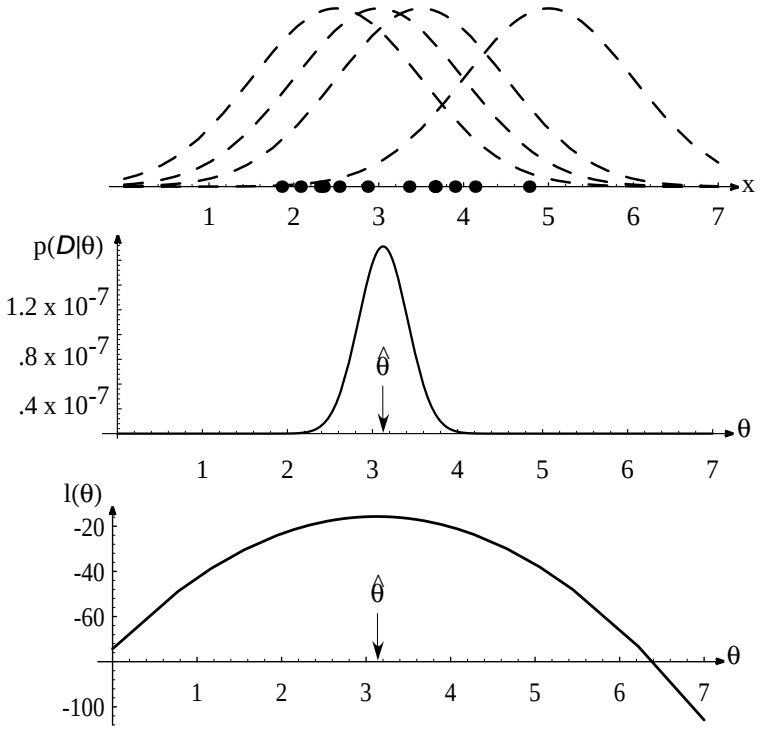


Figure 3.1: The top graph shows several training points in one dimension, known or assumed to be drawn from a Gaussian of a particular variance, but unknown mean. Four of the infinite number of candidate source distributions are shown in dashed lines. The middle figures shows the likelihood $p(\mathcal{D}|\theta)$ 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 — i.e., the log-likelihood $l(\theta)$, shown at the bottom. Note especially that the likelihood lies in a different space from $p(x|\hat{\theta})$, and the two can have different functional forms.

For analytical purposes, it is usually easier to work with the logarithm of the likelihood than with the likelihood itself. Since 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 set is p, then we let $\boldsymbol{\theta}$ denote

the p-component vector $\boldsymbol{\theta} = (\theta_1, ..., \theta_p)^t$, and $\nabla_{\boldsymbol{\theta}}$ 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, i.e.,

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

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

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

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 (Sec. 3.5.1).

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 $l(\boldsymbol{\theta})p(\boldsymbol{\theta})$. 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 parameter space (e.g., an overall rotation), the density will change, and our MAP solution need no longer be appropriate (Sec. 3.5.2).

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

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

To see how maximum likelihood methods results 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

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

Identifying $\boldsymbol{\theta}$ with $\boldsymbol{\mu}$, we see from Eq. 9 that the maximum likelihood estimate for $\boldsymbol{\mu}$ must satisfy

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

that is, each of the d components of $\hat{\boldsymbol{\mu}}$ must vanish. 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 $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$

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 $\hat{\mu}=\hat{\theta}_1$, $\hat{\sigma}^2=\hat{\theta}_2$ 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, though, 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

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

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 $(\mathbf{x}_k-\hat{\boldsymbol{\mu}})(\mathbf{x}_k-\hat{\boldsymbol{\mu}})^t$. Since 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:*

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

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

Elementary *unbiased* estimators for σ^2 and $\boldsymbol{\Sigma}$ are given by

* There should be no confusion over this use of the *statistical* term bias, and that for an offset in neural networks and many other places.

$$\mathcal{E}\left[\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2\right]=\sigma^2 \qquad \text{and} \tag{21}$$

$$\mathbf{C}=\frac{1}{n-1}\sum_{k=1}^{n}(\mathbf{x}_k-\hat{\boldsymbol{\mu}})(\mathbf{x}_k-\hat{\boldsymbol{\mu}})^t, \tag{22}$$

where $\mathbf{C}$ is the so-called *sample covariance matrix*, as explored in Problem 33. 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.

SAMPLE COVARIANCE

ABSOLUTELY UNBIASED

ASYMPTOTICALLY UNBIASED

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 will 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 Chap. **??**.

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 allows us to convert a distribution on this variable into a posterior probability density.