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 making an action based on the "category" of the pattern—has been crucial for our survival, and over the past tens of millions of years we have evolved highly sophisticated neural and cognitive systems for such tasks.

1.1 MACHINE PERCEPTION

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

1.2 AN EXAMPLE

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

FEATURE

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

MODEL

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

PREPROCESSING
SEGMENTATION

FEATURE EXTRACTION

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

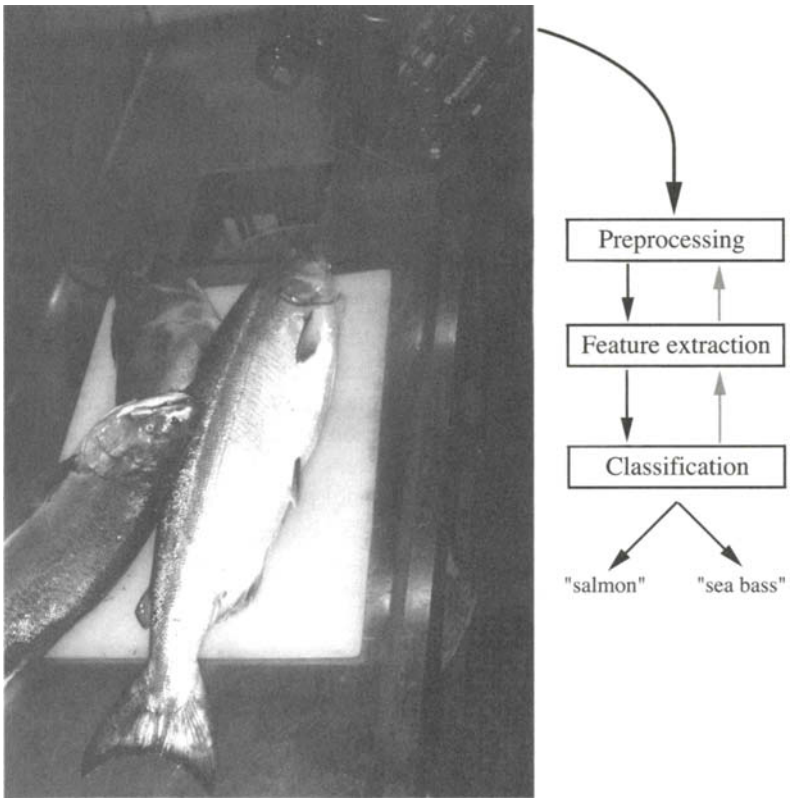

FIGURE 1.1. The objects to be classified are first sensed by a transducer (camera), whose signals are preprocessed. Next the features are extracted and finally the classification is emitted, here either "salmon" or "sea bass." Although the information flow is often chosen to be from the source to the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

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

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

TRAINING SAMPLES

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

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

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

COST

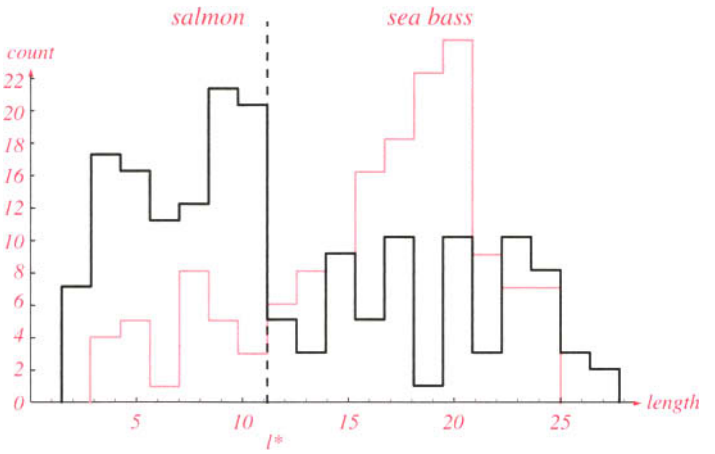

FIGURE 1.2. Histograms for the length feature for the two categories. No single threshold value of the length will serve to unambiguously discriminate between the two categories; using length alone, we will have some errors. The value marked l^* will lead to the smallest number of errors, on average.

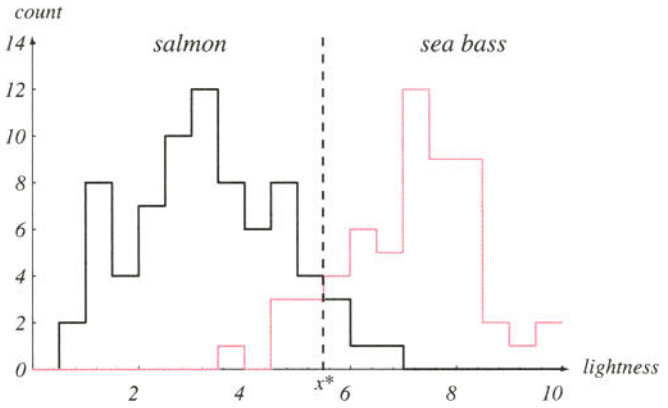

FIGURE 1.3. Histograms for the lightness feature for the two categories. No single threshold value x^* (decision boundary) will serve to unambiguously discriminate between the two categories; using lightness alone, we will have some errors. The value x^* marked will lead to the smallest number of errors,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

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

DECISION THEORY

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

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

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

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

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

DECISION BOUNDARY

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

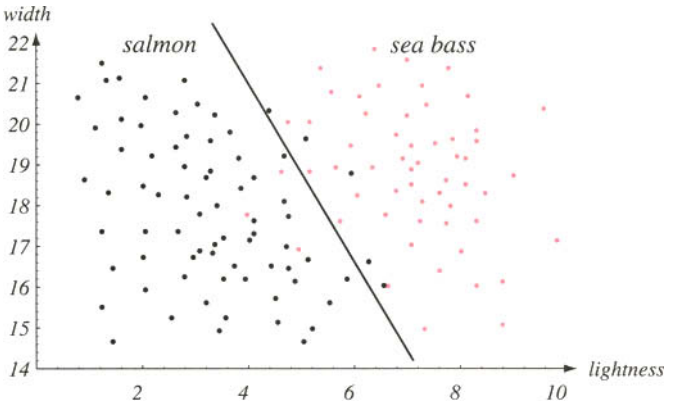

FIGURE 1.4. The two features of lightness and width for sea bass and salmon. The dark line could serve as a decision boundary of our classifier. Overall classification error on the data shown is lower than if we use only one feature as in Fig. 1.3, but there will still be some errors.

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

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

GENERALIZATION

Naturally, one approach would be to get more training samples for obtaining a better estimate of the true underlying characteristics, for instance the probability distributions of the categories. In some 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

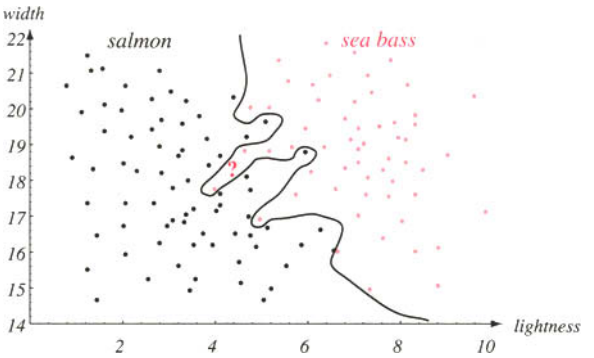

FIGURE 1.5. Overly complex models for the fish will lead to decision boundaries that are complicated. While such a decision may lead to perfect classification of our training samples, it would lead to poor performance on future patterns. The novel test point marked ? is evidently most likely a salmon,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

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

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

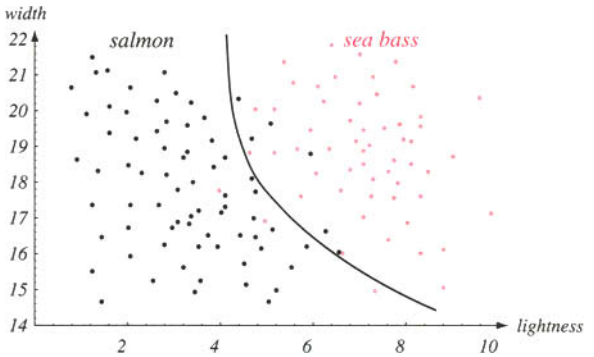

FIGURE 1.6. The decision boundary shown might represent the optimal tradeoff between performance on the training set and simplicity of classifier, thereby giving the highest accuracy on new patterns.

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

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

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

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

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

ANALYSIS BY SYNTHESIS

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

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

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

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

1.2.1 Related Fields

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

IMAGE PROCESSING

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

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

ASSOCIATIVE MEMORY

acts of *associative memory*, the system takes in a pattern and emits another pattern which is representative of a general group of patterns. It thus reduces the information

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

REGRESSION

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

INTERPOLATION

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

DENSITY ESTIMATION

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

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

1.3 PATTERN RECOGNITION SYSTEMS

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

1.3.1 Sensing

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

1.3.2 Segmentation and Grouping

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

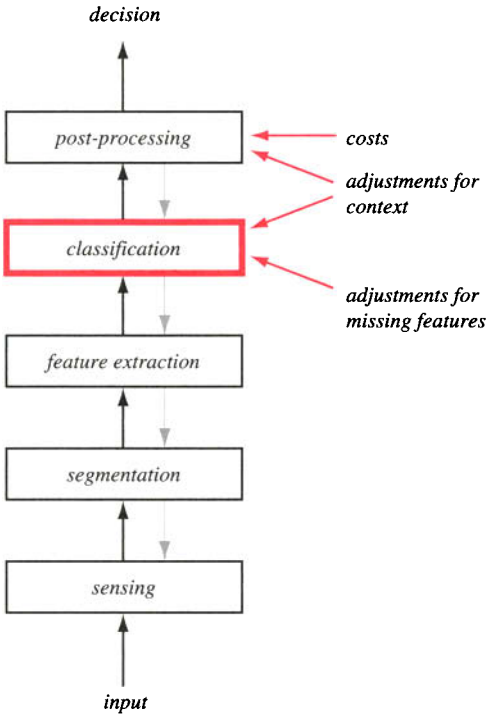

FIGURE 1.7. Many pattern recognition systems can be partitioned into components such as the ones shown here. A sensor converts images or sounds or other physical inputs into signal data. The segmentor isolates sensed objects from the background or from other objects. A feature extractor measures object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

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

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

ings of the wrong size, and that the most useful unit for recognition is somewhat larger.

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

MEREOLOGY

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

1.3.3 Feature Extraction

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

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

INVARIANT FEATURES

TRANSLATION

ROTATION

SCALE

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

OCCLUSION

PROJECTIVE DISTORTION

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

RATE

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

DEFORMATION

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

FEATURE SELECTION

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

1.3.4 Classification

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

NOISE

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

One problem that arises in practice is that it may not always be possible to determine the values of all of the features for a particular input. In our hypothetical system for fish classification, for example, it may not be possible to determine the width of the fish because of occlusion by another fish. How should the categorizer compensate? Since our two-feature recognizer never had a single-variable criterion value x^* determined in anticipation of the possible absence of a feature (cf. Fig. 1.3), how shall it make the best decision using only the feature present? The naïve method,

of merely assuming that the value of the missing feature is zero or the average of the values for the patterns already seen, is provably nonoptimal. Likewise, how should we train a classifier or use one when some features are missing?

1.3.5 Post Processing

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

ERROR RATE

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

RISK

CONTEXT

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

MULTIPLE CLASSIFIERS

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

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

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

1.4 THE DESIGN CYCLE

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

1.4.1 Data Collection

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

1.4.2 Feature Choice

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

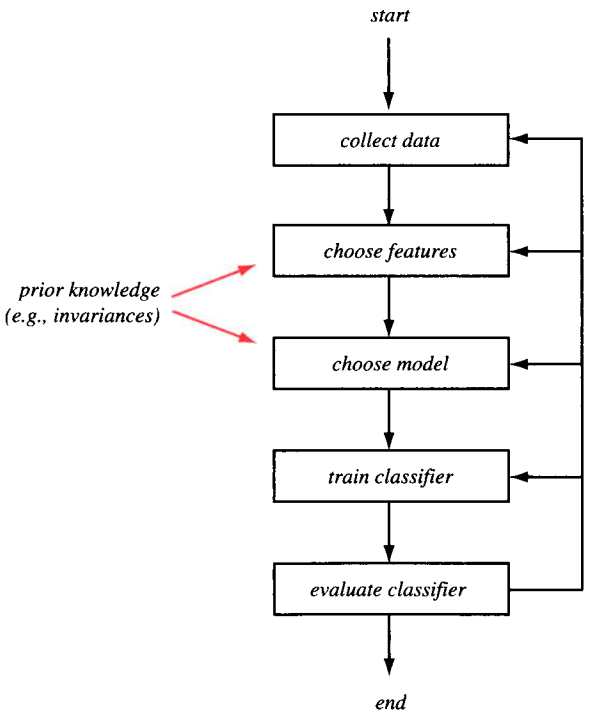

FIGURE 1.8. The design of a pattern recognition system involves a design cycle similar to the one shown here. Data must be collected, both to train and to test the system. The characteristics of the data affect both the choice of appropriate discriminating features and the choice of models for the different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

PRIOR KNOWLEDGE

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

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

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

1.4.3 Model Choice

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

1.4.4 Training

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

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

1.4.5 Evaluation

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

OVERFITTING

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

1.4.6 Computational Complexity

Some pattern recognition problems can be "solved" using algorithms that are highly impractical. For instance, we might try to hand label all possible 20×20 binary pixel images with a category label for optical character recognition, and use table lookup to classify incoming patterns. Although we might in theory achieve error-free recognition, the labeling time and storage requirements would be quite prohibitive since it would require a labeling each of $2^{20\times20} \approx 10^{120}$ patterns. Thus the computational resources necessary and the computational complexity of different algorithms are of considerable practical importance.

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

1.5 LEARNING AND ADAPTATION

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

1.5.1 Supervised Learning

In supervised learning, a teacher provides a category label or cost for each pattern in a training set, and seeks to reduce the sum of the costs for these patterns. How can we be sure that a particular learning algorithm is powerful enough to learn the solution to a given problem and that it will be stable to parameter variations? How can we determine if it will converge in finite time or if it will scale reasonably with

the number of training patterns, the number of input features or the number of categories? How can we ensure that the learning algorithm appropriately favors "simple" solutions (as in Fig. 1.6) rather than complicated ones (as in Fig. 1.5)?

1.5.2 Unsupervised Learning

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

1.5.3 Reinforcement Learning

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

CRITIC

1.6 CONCLUSION

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

SUMMARY BY CHAPTERS

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

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

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

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

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

BIBLIOGRAPHICAL AND HISTORICAL REMARKS

Classification is among the first crucial steps in making sense of the blooming buzzing confusion of sensory data that intelligent systems confront. In the Western world, the foundations of pattern recognition can be traced to Plato [2], which were later extended by Aristotle [1], who distinguished between an "essential property" (which would be shared by all members in a class or "natural kind" as he put it) from an "accidental property" (which could differ among members in the class). Pattern recognition can be cast as the problem of finding such essential properties of a category. In the Eastern world, the first Zen patriarch, Bodhidharma, would point at things and demand students to answer "What is that?" as a way of confronting the deepest issues in mind, the identity of objects, and the nature of classification

and decision [3]. It has been a central theme in the discipline of philosophical epistemology, the study of the nature of knowledge. A more modern treatment of some philosophical problems of pattern recognition, relating to the technical matter in the current book, can be found in references [22, 4] and [18]. A delightful and particularly insightful book on the foundations of artificial intelligence, including pattern recognition, is reference [10].

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

BIBLIOGRAPHY

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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