to this theoretical limit. Alas, in practice we rarely, if ever, know the Bayes error rate. Even if we did know this error rate, it would not help us much in designing a classifier except to tell us that further training and data collection is futile. Thus the Bayes error is generally of theoretical interest. What other fundamental principles and properties might be of greater use in designing classifiers?

Before we address such problems, we should clarify the meaning of the title of this chapter. "Algorithm-independent" here refers, first, to those mathematical foundations that do not depend upon the particular classifier or learning algorithm used. Our upcoming discussion of bias and variance is just as valid for methods based on neural networks or the nearest-neighbor or model-dependent maximum-likelihood. Second, we mean techniques that can be used in conjunction with different learning algorithms, or provide guidance in their use. For example, cross-validation and resampling techniques can be used with any of a large number of training methods. Of course by the very general notion of an algorithm, these too are algorithms, technically speaking, but we discuss them because of their breadth of applicability and their independence from details of the learning techniques used.

In this chapter we shall see, first, that no pattern classification method is inherently superior to any other, or even to random guessing; it is the type of problem, prior distribution, and other information that determine which form of classifier should provide the best performance. We shall then explore several ways to quantify and adjust the "match" between a learning algorithm and the problem it addresses. In any particular problem there are differences between classifiers, of course, and thus we show that with certain assumptions we can estimate their accuracies (even, for instance, before the candidate classifier is fully trained) and compare different classifiers. Finally, we shall see methods for integrating component or "expert" classifiers, which themselves might implement quite different algorithms.

We shall present the results that are most important for pattern recognition practitioners, generally skipping over mathematical proofs that can be found in the original research referenced in the Bibliographical and Historical Remarks section.

9.2 LACK OF INHERENT SUPERIORITY OF ANY CLASSIFIER

We now turn to the central question posed above: If we are interested solely in the generalization performance, are there any reasons to prefer one classifier or learning algorithm over another? If we make no prior assumptions about the nature of the classification task, can we expect any classification method to be superior or inferior overall? Can we even find an algorithm that is overall superior to (or inferior to) random guessing?

9.2.1 No Free Lunch Theorem

As summarized in the *No Free Lunch Theorem*, the answer to these and several related questions is "no." If the goal is to obtain good generalization performance, there are no context-independent or usage-independent reasons to favor one learning or classification method over another. If one algorithm seems to outperform another in a particular situation, it is a consequence of its fit to the particular pattern recognition problem, not the general superiority of the algorithm. When confronting a new pattern recognition problem, appreciation of this theorem reminds us to focus on the aspects that matter most—prior information, data distribution, amount of training data, and cost or reward functions. The theorem also justifies a healthy skepticism regarding studies that purport to demonstrate the overall superiority of a particular learning or recognition algorithm.

First we should consider more closely the method by which we judge the generalization performance of a classifer. Up to here, we have estimated such performance by means of a test data set, sampled independently, as is the training set. In some cases, this approach has some unexpected drawbacks when applied to comparing classifiers. In a discrete problem, for example, when the training set and test set are very large, they necessarily overlap, and we are testing on training patterns. Further, virtually any powerful algorithm such as the nearest-neighbor algorithm, unpruned decision trees, or neural networks with sufficient number of hidden nodes can learn the training set perfectly. Second, for low-noise or low-Bayes error cases, if we use an algorithm powerful enough to learn the training set, then the upper limit of the i.i.d. error decreases as the training set size increases.

OFF-TRAINING SET ERROR

Thus, in order to compare learning algorithms, we will use the *off-training set error*—the error on points *not* in the training set, as will become clear. If the training set is very large, then the maximum size of the off-training set data is necessarily small.

For simplicity consider a two-category problem, where the training set $\mathcal{D}$ consists of patterns $\mathbf{x}^i$ and associated category labels $y_i = \pm 1$ for $i = 1, \ldots, n$ generated by the unknown target function to be learned, $F(\mathbf{x})$, where $y_i = F(\mathbf{x}^i)$. In most cases of interest there is a random component in $F(\mathbf{x})$ and thus the same input could lead to different categories, giving nonzero Bayes error. At first we shall assume that the feature set is discrete; this simplifies notation and allows the use of summation and probabilities rather than integration and probability densities. The general conclusions hold in the continuous case as well, but the required technical details would cloud our discussion.

Let $\mathcal{H}$ denote the (discrete) set of hypotheses, or possible sets of parameters to be learned. A particular hypothesis $h(\mathbf{x}) \in \mathcal{H}$ could be described by quantized weights in a neural network, or parameters $\boldsymbol{\theta}$ in a functional model, or sets of decisions in a tree, and so on. Furthermore, $P(h)$ is the prior probability that the algorithm will produce hypothesis h after training; note that this is *not* the probability that h is correct. Next, $P(h|\mathcal{D})$ denotes the probability that the algorithm will yield hypothesis h when trained on the data $\mathcal{D}$. In deterministic learning algorithms such as the nearest-neighbor and decision trees, $P(h|\mathcal{D})$ will be everywhere zero except for a single hypothesis h. For stochastic methods (such as neural networks trained from random initial weights), or stochastic Boltzmann learning, $P(h|\mathcal{D})$ can be a broad distribution. Let E be the error for a zero-one or other loss function.

How shall we judge the generalization quality of a learning algorithm? Because we are not given the target function, the natural measure is the expected value of the error given $\mathcal{D}$, summed over all possible targets. This scalar value can be expressed as a weighted "inner product" between the distributions $P(h|\mathcal{D})$ and $P(F|\mathcal{D})$, as follows:

$$\mathcal{E}[E|\mathcal{D}] = \sum_{h,F} \sum_{\mathbf{x} \notin \mathcal{D}} P(\mathbf{x})[1 - \delta(F(\mathbf{x}), h(\mathbf{x}))] P(h|\mathcal{D}) P(F|\mathcal{D}), \qquad (1)$$

where for the moment we assume there is no noise. The familiar Kronecker delta function, $\delta(\cdot, \cdot)$, has value 1 if its two arguments match, and value 0 otherwise. Equation 1 states that the expected error rate, given a fixed training set $\mathcal{D}$, is related to the sum over all possible inputs weighted by their probabilities, $P(\mathbf{x})$, as well as the "alignment" or "match" of the learning algorithm, $P(h|\mathcal{D})$, to the actual posterior $P(F|\mathcal{D})$. The important insight provided by this equation is that without prior knowledge concerning $P(F|\mathcal{D})$, we can prove little about any particular learning algorithm $P(h|\mathcal{D})$, including its generalization performance.

The expected off-training-set classification error when the true function is $F(\mathbf{x})$ and the probability for the kth candidate learning algorithm is $P_k(h(\mathbf{x})|\mathcal{D})$ is given by

$$\mathcal{E}_k(E|F, n) = \sum_{\mathbf{x}\notin\mathcal{D}} P(\mathbf{x})[1 - \delta(F(\mathbf{x}), h(\mathbf{x}))]P_k(h(\mathbf{x})|\mathcal{D}). \tag{2}$$

Although we shall not provide a formal proof, we are now in a position to give a precise statement of the No Free Lunch Theorem.*

■ **Theorem 9.1. (No Free Lunch)** For any two learning algorithms $P_1(h|\mathcal{D})$ and $P_2(h|\mathcal{D})$, the following are true, independent of the sampling distribution $P(\mathbf{x})$ and the number n of training points:

1. Uniformly averaged over all target functions F, $\mathcal{E}_1(E|F, n) - \mathcal{E}_2(E|F, n) = 0$
2. For any fixed training set $\mathcal{D}$, uniformly averaged over F, $\mathcal{E}_1(E|F, \mathcal{D}) - \mathcal{E}_2(E|F, \mathcal{D}) = 0$
3. Uniformly averaged over all priors $P(F)$, $\mathcal{E}_1(E|n) - \mathcal{E}_2(E|n) = 0$
4. For any fixed training set $\mathcal{D}$, uniformly averaged over $P(F)$, $\mathcal{E}_1(E|\mathcal{D}) - \mathcal{E}_2(E|\mathcal{D}) = 0$

Part 1 says that uniformly averaged over all target functions the expected off-training set error for all learning algorithms is the same, that is,

$$\sum_F \sum_{\mathcal{D}} P(\mathcal{D}|F)\,[\mathcal{E}_1(E|F, n) - \mathcal{E}_2(E|F, n)] = 0, \tag{3}$$

for any two learning algorithms. In short, no matter how clever we are at choosing a "good" learning algorithm $P_1(h|\mathcal{D})$ and a "bad" algorithm $P_2(h|\mathcal{D})$ (perhaps even random guessing, or a constant output), if all target functions are equally likely, then the "good" algorithm will not outperform the "bad" one. Stated more generally, there are no i and j such that for all $F(\mathbf{x})$, $\mathcal{E}_i(E|F, n) > \mathcal{E}_j(E|F, n)$. Furthermore, no matter what algorithm we use, there is at least one target function for which random guessing is a better algorithm.

With the assumption that the training set can be learned by all algorithms we consider, Part 2 states that even if we know $\mathcal{D}$, then averaged over all target functions no learning algorithm yields an off-training set error that is superior to any other, that is,

$$\sum_F [\mathcal{E}_1(E|F, \mathcal{D}) - \mathcal{E}_2(E|F, \mathcal{D})] = 0. \tag{4}$$

Parts 3 and 4 concern nonuniform target function distributions and have related interpretations (Problems 2–5). Example 1 provides an elementary illustration.

*The clever name for the theorem was suggested by David Haussler.

EXAMPLE 1 No Free Lunch for Binary Data

Consider input vectors consisting of three binary features, and a particular target function $F(\mathbf{x})$, as given in the table. Suppose (deterministic) learning algorithm 1 assumes every pattern is in category ω_1 unless trained otherwise, and algorithm 2 assumes every pattern is in ω_2 unless trained otherwise. Thus when trained with $n = 3$ points in $\mathcal{D}$, each algorithm returns a single hypothesis, h_1 and h_2, respectively. In this case the expected errors on the off-training set data are $\mathcal{E}_1(E|F, \mathcal{D}) = 0.4$ and $\mathcal{E}_2(E|F, \mathcal{D}) = 0.6$.

	$\mathbf{x}$	F	h_1	h_2
	000	1	1	1
$\mathcal{D}$	001	−1	−1	−1
	010	1	1	1
	011	−1	1	−1
	100	1	1	−1
	101	−1	1	−1
	110	1	1	−1
	111	1	1	−1

For this target function $F(\mathbf{x})$, clearly algorithm 1 is superior to algorithm 2. But note that the designer does not *know* $F(\mathbf{x})$—indeed, we assume we have no prior information about $F(\mathbf{x})$. The fact that all targets are equally likely means that $\mathcal{D}$ provides no information about $F(\mathbf{x})$. If we wish to compare the algorithms overall, we therefore must average over all such possible target functions consistent with the training data. Part 2 of Theorem 9.1 states that averaged over all possible target functions, there is no difference in off-training set errors between the two algorithms. For each of the 2^5 distinct target functions consistent with the $n = 3$ patterns in $\mathcal{D}$, there is exactly one other target function whose output is inverted for each of the patterns outside the training set, and this ensures that the performances of algorithms 1 and 2 will also be inverted, so that the contributions to the formula in Part 2 cancel. Thus indeed Part 2 of the Theorem as well as Eq. 4 are obeyed.

Figure 9.1 illustrates a result derivable from Part 1 of Theorem 9.1. Each of the six squares represents the set of all possible classification problems; note that this is *not* the standard feature space. If a learning system performs well—higher than average generalization accuracy—over some set of problems, then it must perform worse than average elsewhere, as shown in Fig. 9.1 a. No system can perform well throughout the full set of functions (Fig. 9.1 d); to do so would violate the No Free Lunch Theorem.

In sum, all statements of the form "learning/recognition algorithm 1 is better than algorithm 2" are ultimately statements about the relevant target functions. Hence there is a "conservation theorem" in generalization: For every possible learning algorithm for binary classification the sum of performance over all possible target functions is exactly zero. Thus we cannot achieve positive performance on some problems without getting an equal and opposite amount of negative performance on other problems. While we may hope that we never have to apply any particular algorithm to certain problems, all we can do is trade performance on problems we do not expect to encounter with those that we do expect to encounter. This, along with the other results from the No Free Lunch Theorem, stresses that it is the *assumptions*

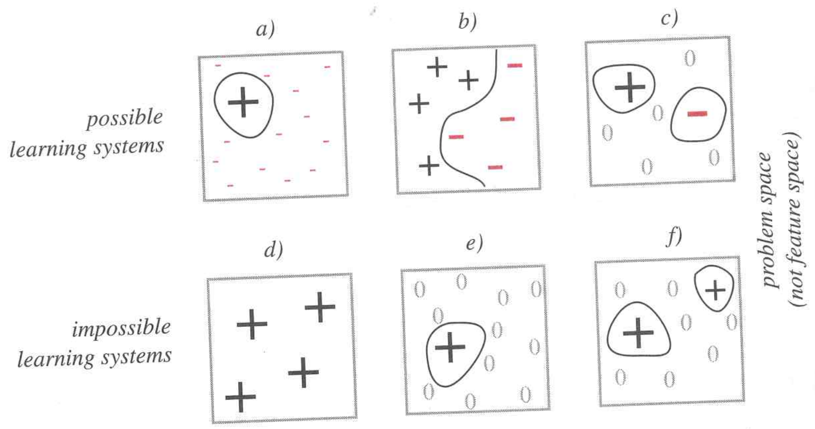
a)
b)
c)
d)
e)
f)
possible learning systems
impossible learning systems
problem space (not feature space)

FIGURE 9.1. The No Free Lunch Theorem shows the generalization performance on the off-training set data that *can* be achieved (top row) and also shows the performance that *cannot* be achieved (bottom row). Each square represents all possible classification problems consistent with the training data—this is *not* the familiar feature space. A + indicates that the classification algorithm has generalization higher than average, a – indicates lower than average, and a 0 indicates average performance. The size of a symbol indicates the amount by which the performance differs from the average. For instance, part a shows that it is possible for an algorithm to have high accuracy on a small set of problems so long as it has mildly poor performance on all other problems. Likewise, part b shows that it is possible to have excellent performance throughout a large range of problem, but this will be balanced by very poor performance on a large range of other problems. It is impossible, however, to have good performance throughout the full range of problems, shown in part d. It is also impossible to have higher-than-average performance on some problems while having average performance everywhere else, shown in part e.

about the learning domains that are relevant. Another practical import of the theorem is that even popular and theoretically grounded algorithms will perform poorly on some problems, ones in which the learning algorithm and the posterior happen not to be "matched," as governed by Eq. 1. Practitioners must be aware of this possibility, which arises in real-world applications. Expertise limited to a small range of methods, even powerful ones such as neural networks, will not suffice for all classification problems. Experience with a broad range of techniques is the best insurance for solving arbitrary new classification problems.

9.2.2 Ugly Duckling Theorem

While the No Free Lunch Theorem shows that in the absence of assumptions we should not prefer any learning or classification algorithm over another, an analogous theorem addresses features and patterns. Roughly speaking, the Ugly Duckling Theorem states that in the absence of assumptions there is no privileged or "best" feature representation, and that even the notion of similarity between patterns depends implicitly on assumptions that may or may not be correct.

Because we are using discrete representations, we can use logical expressions or "predicates" to describe a pattern, much as in Chapter 8. If we denote a binary feature attribute by f_i, then a particular pattern might be described by the predicate "f_1 *AND* f_2," another pattern might be described as "*NOT* f_2," and so on. Likewise

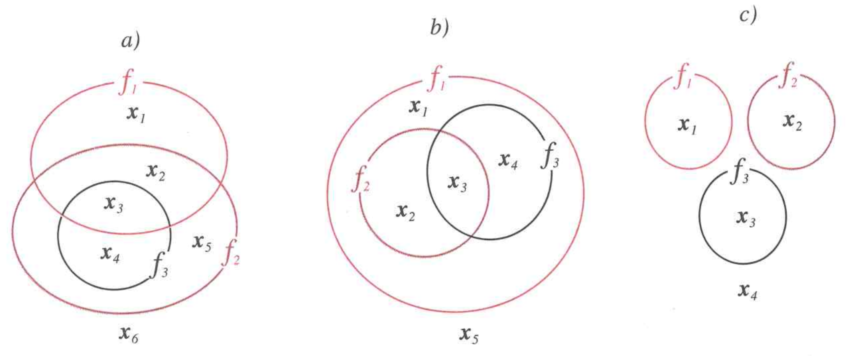

FIGURE 9.2. Patterns $\mathbf{x}_i$, represented as d-tuples of binary features f_i, can be placed in Venn diagram (here $d = 3$); the diagram itself depends upon the classification problem and its constraints. For instance, suppose f_1 is the binary feature attribute `has_legs`, f_2 is `has_right_arm` and f_3 the attribute `has_right_hand`. Thus in part a pattern $\mathbf{x}_1$ denotes a person who has legs but neither arm nor hand; $\mathbf{x}_2$ a person who has legs and an arm, but no hand; and so on. Notice that the Venn diagram expresses the biological constraints associated with real people: it is impossible for someone to have a right hand but no right arm. Part c expresses different constraints, such as the biological constraint of mutually exclusive eye colors. Thus attributes f_1, f_2 and f_3 might denote `brown`, `green`, and `blue`, respectively, and a pattern $\mathbf{x}_i$ describes a real person, whom we can assume cannot have eyes that differ in color.

we could have a predicate involving the patterns themselves, such as $\mathbf{x}_1$ *OR* $\mathbf{x}_2$. Figure 9.2 shows how patterns can be represented in a Venn diagram.

Below we shall need to count predicates, and for clarity it helps to consider a particular Venn diagram, such as that in Fig. 9.3. This is the most general Venn diagram based on two features, because for every configuration of f_1 and f_2 there is indeed a pattern. Here predicates can be as simple as "$\mathbf{x}_1$," or more complicated, such as "$\mathbf{x}_1$ *OR* $\mathbf{x}_2$ *OR* $\mathbf{x}_4$," and so on.

RANK

The *rank* r of a predicate is the number of the simplest or indivisible elements it contains. The tables below show the predicates of rank 1, 2, and 3 associated with the Venn diagram of Fig. 9.3.* Not shown is the fact that there is but one predicate of rank $r = 4$, the disjunction of the $\mathbf{x}_1, \ldots, \mathbf{x}_4$, which has the logical value `True`. If we

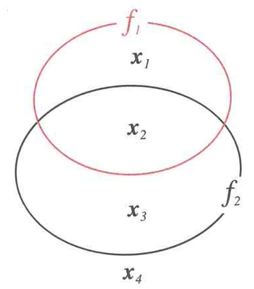

FIGURE 9.3. The Venn for a problem with no constraints on two features. Thus all four binary attribute vectors can occur.

*Technically speaking, we should use set operations rather than logical operations when discussing the Venn diagram, writing $\mathbf{x}_1 \cup \mathbf{x}_2$ instead of $\mathbf{x}_1$ OR $\mathbf{x}_2$. Nevertheless, we use logical operations here for consistency with the rest of the text.

let n be the total number of regions in the Venn diagram (i.e., the number of distinct possible patterns), then there are $\binom{n}{r}$ predicates of rank r, as shown at the bottom of the table.

rank $r = 1$

$\mathbf{x}_1$	f_1 *AND NOT* f_2
$\mathbf{x}_2$	f_1 *AND* f_2
$\mathbf{x}_3$	f_2 *AND NOT* f_1
$\mathbf{x}_4$	*NOT* (f_1 *OR* f_2)

$\binom{4}{1} = 4$

rank $r = 2$

$\mathbf{x}_1$ OR $\mathbf{x}_2$	f_1
$\mathbf{x}_1$ OR $\mathbf{x}_3$	f_1 XOR f_2
$\mathbf{x}_1$ OR $\mathbf{x}_4$	NOT f_2
$\mathbf{x}_2$ OR $\mathbf{x}_3$	f_2
$\mathbf{x}_2$ OR $\mathbf{x}_4$	NOT(f_1 AND f_2)
$\mathbf{x}_3$ OR $\mathbf{x}_4$	NOT f_1

$\binom{4}{2} = 6$

rank $r = 3$

$\mathbf{x}_1$ OR $\mathbf{x}_2$ OR $\mathbf{x}_3$	f_1 OR f_2
$\mathbf{x}_1$ OR $\mathbf{x}_2$ OR $\mathbf{x}_4$	f_1 OR NOT f_2
$\mathbf{x}_1$ OR $\mathbf{x}_3$ OR $\mathbf{x}_3$	NOT(f_1 AND f_2)
$\mathbf{x}_2$ OR $\mathbf{x}_3$ OR $\mathbf{x}_4$	f_2 OR NOT f_1

$\binom{4}{3} = 4$

The total number of predicates in the absence of constraints is

$$\sum_{r=0}^{n} \binom{n}{r} = (1+1)^n = 2^n, \qquad (5)$$

and thus for the $d = 4$ case of Fig. 9.3, there are $2^4 = 16$ possible predicates (Problem 9). Note that Eq. 5 applies only to the case where there are no constraints; for Venn diagrams that do incorporate constraints, such as those in Fig. 9.2, the formula does not hold (Problem 10).

Now we turn to our central question: In the absence of prior information, is there a principled reason to judge any two distinct patterns as more or less similar than two other distinct patterns? A natural and familiar measure of similarity is the number of features or attributes shared by two patterns, but even such an obvious measure presents conceptual difficulties.

To appreciate such difficulties, consider first a simple example. Suppose attributes f_1 and f_2 represent `blind_in_right_eye` and `blind_in_left_eye`, respectively. If we base similarity on shared features, person $\mathbf{x}_1 = \{1, 0\}$ (blind only in the right eye) is maximally different from person $\mathbf{x}_2 = \{0, 1\}$ (blind only in the left eye). In particular, in this scheme $\mathbf{x}_1$ is more similar to a totally blind person and to a normally sighted person than he is to $\mathbf{x}_2$. But this result may prove unsatisfactory; we can easily envision many circumstances where we would consider a person blind in just the right eye to be "similar" to one blind in just the left eye. Such people might be permitted to drive automobiles, for instance. Furthermore, a person blind in just one eye would differ significantly from totally blind person who would not be able to drive.

A second, related point is that there are always multiple ways to represent vectors (or tuples) of attributes. For instance, in the above example, we might use alternative features f_1' and f_2' to represent `blind_in_right_eye` and `same_in_both_eyes`, respectively, and then the four types of people would be represented as shown in the tables.

	f_1	f_2
$\mathbf{x}_1$	0	0
$\mathbf{x}_2$	0	1
$\mathbf{x}_3$	1	0
$\mathbf{x}_4$	1	1

f_1'	f_2'
0	1
0	0
1	0
1	1

Of course there are other representations, each more or less appropriate to the particular problem at hand. In the absence of prior information, though, there is no principled reason to prefer one of these representations over another.

We must then still confront the problem of finding a principled measure of the similarity between two patterns, given some representation. The only plausible candidate measure in this circumstance would be the number of *predicates* (rather than the number of features) the patterns share. Consider two distinct patterns (in some representation) $\mathbf{x}_i$ and $\mathbf{x}_j$, where $i \neq j$. Regardless of the constraints in the problem (i.e., the Venn diagram), there are, of course, no predicates of rank $r = 1$ that are shared by the two patterns. There is but one predicate of rank $r = 2$; it is $\mathbf{x}_i \; OR \; \mathbf{x}_j$. A predicate of rank $r = 3$ must contain three patterns, two of which are $\mathbf{x}_i$ and $\mathbf{x}_j$. Because there are d patterns total, there are then $\binom{d-2}{1} = d - 2$ predicates of rank 3 that are shared by $\mathbf{x}_i$ and $\mathbf{x}_j$. Likewise, for an arbitrary rank r, there are $\binom{d-2}{r-2}$ predicates shared by the two patterns, where $2 \leq r \leq d$. The total number of predicates shared by the two patterns is thus the sum

$$\sum_{r-2}^{d} \binom{d-2}{r-2} = (1+1)^{d-2} = 2^{d-2}. \qquad (6)$$

Note the key result: Equation 6 is *independent* of the choice of $\mathbf{x}_i$ and $\mathbf{x}_j$ (so long as they are distinct). Thus we conclude that the number of predicates shared by two distinct patterns is *constant*, and *independent* of the patterns themselves (Problem 11). We conclude that if we judge similarity based on the number of predicates that patterns share, then any two distinct patterns are "equally similar." This is stated formally as the following theorem:

■ **Theorem 9.2. (Ugly Duckling)** Given that we use a finite set of predicates that enables us to distinguish any two patterns under consideration, the number of predicates shared by any two such patterns is constant and independent of the choice of those patterns. Furthermore, if pattern similarity is based on the total number of predicates shared by two patterns, then any two patterns are "equally similar." *

In summary, then, the Ugly Duckling Theorem states something quite basic yet often overlooked: There is no problem-independent or privileged or "best" set of features or feature attributes. Moreover, while the above was derived using d-tuples of binary values, it also applies to a continuous feature space too, if such a space is discretized (at any resolution). The theorem forces us to acknowledge that even the apparently simple notion of similarity between patterns is fundamentally based on implicit assumptions about the problem domain (Problem 12).

9.2.3 Minimum Description Length (MDL)

It is sometimes claimed that the minimum description length principle provides justification for preferring one type of classifier over another—specifically, "simpler" classifiers over "complex" ones. Briefly stated, the approach purports to find some irreducible, smallest representation of all members of a category (much like a "sig nal"); all variation among the individual patterns is then "noise." The argument i

*The theorem gets its fanciful name from the following counterintuitive statement: Assuming that sim larity is based on the number of shared predicates, an ugly duckling A is as similar to beautiful swan B a beautiful swan C is to B, given that these items differ at all from one another.