Chapter 7
Computational Learning Theory

As they say, nothing is more practical than a good theory. And indeed, mathematical models of learnability have helped improve our understanding of what it takes to induce a useful classifier from data, and, conversely, why the outcome of a machine-learning undertaking so often disappoints. And so, even though this textbook does not want to be mathematical, it cannot help introducing at least the basic concepts of the *computational learning theory*.

At the core of this theory is the idea of *PAC learning*, a paradigm that makes it possible to quantify learnability. Restricting itself to domains with discrete attributes, the first section of this chapter derives a simple expression that captures the mutual relation between the training-set size and the induced classifier's error rate. Some consequences of this formula are briefly discussed in the two sections that follow. For domains with continuous attributes, the so-called VC-dimension is then introduced.

7.1 PAC Learning

Perhaps the most useful idea contributed by computational learning theory is that of "probably approximate learning," sometimes abbreviated as *PAC learning*. Let us first explain the underlying principles and derive a formula that will then provide some useful guidance.

Assumptions and Definitions The analysis will be easier if we build it around a few simplifying assumptions. First, the training examples—as well as all future examples—are completely noise-free. Second, all attributes are discrete (none of them is continuous-valued). Third, the classifier acquires the formof a logical

M. Kubat, *An Introduction to Machine Learning*,
DOI 10.1007/978-3-319-63913-0_7

expression of attribute values; the expression is *true* for positive examples and *false* for negative examples. And, finally, there exists at least one expression that correctly classifies all training examples.[1]

Each of the logical expressions can then be regarded as one hypothesis about what the classifier should look like. Together, all of the hypotheses form a *hypothesis space* whose size (the number of distinct hypotheses) is $|H|$. Under the assumptions listed above, $|H|$ is a finite number.

Inaccurate Classifier May Still Succeed on the Training Set Available training data rarely exhaust all subtleties of the underlying class; a classifier that labels correctly all training examples may still perform poorly in the future. The frequency of these mistakes is usually lower if we add more training examples because these additions may reflect aspects that the original data failed to represent. The rule of thumb is actually quite simple: the more examples we have, the better the induced classifier.

How many training examples will give us a fair chance of future success? To find the answer, we will first consider a hypothetical classifier whose error rate on the entire instance space is greater than some predefined ϵ. Put another way. the probability that this classifier will label correctly a randomly picked example is less than $1 - \epsilon$. Taking this reasoning one step further, the probability, P, that this imperfect classifier will label correctly m random examples is bounded by the following expression:

$$P \leq (1 - \epsilon)^m \tag{7.1}$$

And here is what this means: with probability $P \leq (1 - \epsilon)^m$, an entire training set consisting of m examples will be correctly classified by a classifier whose error rate actually exceeds ϵ. Of course, this probability is for a realistic m very low. For instance, if $\epsilon = 0.1$ and $m = 20$ (which is a very small set), we will have $P < 0.122$. If we increase the training-set size to $m = 100$ (while keeping the error rate bound by $\epsilon = 0.1$), then P drops to less than 10^{-4}. This is indeed small; but low probability is not the same as impossibility.

Eliminating Poor Classifiers Suppose that an error rate greater than ϵ is deemed unacceptable. What are the chances that a classifier with performance this poor is induced from the given training set, meaning that it classifies correctly all training examples?

The hypothesis space consists of $|H|$ classifiers. Let us consider the theoretical possibility that we evaluate all these classifiers on the m training examples, and then keep only those classifiers that have never made any mistake. Amongthese

[1]The reader will have noticed that all these requirements are satisfied by the "pies" domain from Chap. 1.

"survivors," some will disappoint in the sense that, while being error-free on the training set, their error rates on the entire instance space actually exceed ϵ. Let there be k such classifiers.

The concrete value of k cannot be established without evaluating each single classifier on the entire instance space. This being impossible, all we can say is that $k \leq |H|$, which is a somewhat better situation because $|H|$ is known in many realistic cases, or at least can be calculated.[2]

Let us re-write the upper bound on the probability that at least one of the k offending classifiers will be error-free on the m training examples.

$$P \leq k(1-\epsilon)^m \leq |H|(1-\epsilon)^m \tag{7.2}$$

With this, we have established an upper bound on the probability that m training examples will succeed in eliminating all classifiers whose error rate exceeds ϵ.

To become useful, the last expression has to be modified. We know from mathematics that $1-\epsilon < e^{-\epsilon}$, which means that $(1-\epsilon)^m < e^{-m\epsilon}$. With this in mind, we are able to express the upper bound in an exponential form:

$$P \leq |H| \cdot e^{-m\epsilon} \tag{7.3}$$

Suppose we want this probability to be lower than some user-set δ:

$$|H| \cdot e^{-m\epsilon} \leq \delta \tag{7.4}$$

Taking the logarithm of both sides, and rearranging the terms, we obtain the formula that we will work with in the next few pages:

$$m > \frac{1}{\epsilon}(\ln |H| + \ln \frac{1}{\delta}) \tag{7.5}$$

"Probably Approximately Correct" Learning The main reason we have gone through all the details of the derivation is that the reader thus gets a better grasp of the meanings and interpretations of the involved variables which may otherwise be a bit confusing. This is also why these variables are summarized in Table 7.1 for quick reference.

We have now reached a stage where we are able to define some important concepts. A classifier with error rate below ϵ is deemed *approximately correct*; and δ is the *probability* that this approximately correct classifier will be induced from m training examples (m being a finite number). Hence the name of the whole paradigm: *p*robably *a*pproximately *c*orrect learning, or simply *PAC learning*. For the needs of this chapter, we will say that a class is *not PAC-learnable* if the number of examples

[2]Recall that in the "pies" domain from Chap. 1, the size of the hypothesis space was $|H| = 2^{108}$. Of these hypotheses, 2^{96} classified correctly the entire training set.

Table 7.1 The variables involved in our studies of PAC-learnability

m	...	The number of the training examples
$\|H\|$	...	The size of the hypothesis space
ϵ	...	The classifier's maximum permitted error rate
δ	...	The probability that a classifier with error rate greater than ϵ is error-free on the training set

needed to satisfy the given (ϵ, δ)-requirements is so high that we cannot expect a training set of this size ever to be available—or, if it *is* available, then the learning software will need impracticably long time (say, thousands of years) to induce from it the classifier.

Interpretation Inequality (7.5) specifies how many training examples, m, are needed if, with probability at least δ, a classifier with error rate lower than ϵ is to be induced. Note that this result does not depend on the concrete machine-learning technique, only on the size, $|H|$, of the hypothesis space defined by the given type of classifier.

An important thing to remember is that m grows linearly in $1/\epsilon$. For instance, if we strengthen the limit on the error rate from $\epsilon = 0.2$ to $\epsilon = 0.1$, we will need (at least in theory) twice as many training examples to have the same chance, δ, of success. At the same time, the reader should also notice that m is less sensitive to changes in δ, growing only logarithmically in $1/\delta$.

However, we must not forget that our derivation was something like a worst-case analysis. As a result, the bound it has given us is less tight than a perfectionist might desire. For instance, the derivation allowed the possibility that $k = |H|$, which is clearly too pessimistic for the vast majority of practical applications. Inequality (7.5) should therefore never be interpreted as telling the engineer how many training examples to use. Rather, it should be seen as a guideline that makes it possible to compare the learnability of alternative classifier types. We will pursue this idea in the next section.

What Have You Learned?

To make sure you understand this topic, try to answer the following questions. If you have problems, return to the corresponding place in the preceding text.

- What is the meaning of the four variables (m, $|H|$, ϵ, and δ) used in Inequality (7.5)?
- What does the term *PAC learning* refer to? Under what circumstances do we say that a class is *not PAC-learnable*?
- Derive Inequality (7.5). Discuss the meaning and practical benefits of this result, and explain why it should only be regarded as a worst-case analysis.

7.2 Examples of PAC Learnability

Inequality (7.5) tells us how learnability, defined by the (ϵ, δ)-requirements, depends on the size of the hypothesis space. Let us illustrate this result on two concrete types of classifiers.

Conjunctions of Boolean Attributes: Hypothesis Space Suppose that all attributes are boolean, that all data are noise-free, and that an example's class is known to be determined by a logical conjunction of attribute values: if *true*, then the example is positive, otherwise it is deemed negative. For instance, the labels of the training examples may be determined by the following expression:

```
att1 = true AND att3 = false
```

This means that an example is labeled as `pos` if the value of the first attribute is *true* and the value of the third attribute is *false*, regardless of the value of any other attribute. An example that fails to satisfy these two conditions is labeled as `neg`.

The task for the machine-learning program is to find a conjunction that correctly labels all training examples. The set of all conjunctions permitted by our informal definition forms the hypothesis space, $|H|$. What is the size of this space?

In a logical conjunction of the kind specified above, each attribute is either *true* or *false* or irrelevant. This gives us three possibilities for the first attribute, times three possibilities for the second, and so on, times three possibilities for the last, n-th, attribute. The size of the hypothesis space is therefore $|H| = 3^n$.

Conjunctions of Boolean Attributes: PAC-Learnability Suppose that a training set consisting of noise-free examples is presented to some machine-learning program that is capable of inducing classifiers of the just-defined form.[3] To satisfy the last of the assumptions listed at the beginning of Sect. 7.1, we will assume that at least one of the logical conjunctions classifies correctly all training examples.

Since $\ln |H| = \ln 3^n = n \ln 3$, we can re-write Inequality (7.5) in the following form:

$$m > \frac{1}{\epsilon}(n \ln 3 + \ln \frac{1}{\delta}) \tag{7.6}$$

With this, we have obtained a conservative lower bound on the number of training examples that are needed if our (ϵ, δ)-requirements are to be satisfied: with probability δ, the induced classifier (error-free on the training set) will exhibit error rate less than ϵ on the entire instance space. Note that the value of this expression grows linearly in the number of attributes, n. Theoretically speaking, then, if n is doubled, then twice as many training examples will be needed if, with probability limited by δ, classifiers with error rates above the predefined ϵ are to be weeded out.

[3] For instance, a variation of the hill-climbing search from Sect. 1.2 might be used to this end.

Any Boolean Function: Hypothesis Space Let us now investigate a broader class of classifiers, namely those defined by *any* boolean function, allowing for all three basic logical operators (AND, OR, and NOT) as well as any combination of parentheses. As before, we will assume that the examples are described by n boolean attributes, and that they are noise-free.

What is the size of this hypothesis space?

From n boolean attributes, 2^n different examples can be created. This defines the size of the instance space. For any subset of these 2^n examples, there exists at least one logical function that is *true* for all examples from this subset (labeling them as `pos`) and *false* for all examples from outside this subset (labeling them as `neg`). Two logical functions will be regarded as identical from the classification point of view if each of them labels any example with the same class; that is, if they never differ in their "opinion" about any example's class. The number of logical functions that are mutually distinct in their classification behavior then has to be the same as the number of the subsets of the instance space.

A set consisting of X elements is known to have 2^X subsets. Since our specific instance space consists of 2^n examples, the number of its subsets is 2^{2^n}—and this is the size of our hypothesis space:

$$|H| = 2^{2^n} \tag{7.7}$$

Any Boolean Function: PAC-Learnability Since $\ln |H| = \ln 2^{2^n} = 2^n \ln 2$, Inequality (7.5) acquires the following form:

$$m > \frac{1}{\epsilon}(2^n \ln 2 + \ln \frac{1}{\delta}) \tag{7.8}$$

We conclude that the lower bound on the number of the training examples that are needed if the (ϵ, δ)-requirements are to be satisfied grows here exponentially in the number of attributes, n. Such growth is prohibitive for any realistic value of n. For instance, even if we add only a single attribute, $n + 1$, the value of $\ln |H|$ will double because $\ln |H| = 2^{n+1} \ln 2$ which is twice a much as $2^n \ln 2$. And if we add ten attributes, $n + 10$, then the value of $\ln |H|$ increases a thousand times because $\ln |H| = 2^{n+10} \ln 2 = 2^{10} \cdot 2^n \ln 2 = 1024 \cdot 2^n \cdot \ln 2$.

This observation is enough to convince us that a classifier in this general form is *not PAC-learnable*.

A Word of Caution We must be careful not to jump to conclusions. As pointed out earlier, the derivation of Inequality (7.5)—a worst-case analysis of sorts—relied on quite a few simplifying assumptions that render the obtained bounds very conservative. In reality, the number of the training examples needed for the induction of a reliable classifier is much lower than that indicated by our "magic formula."

For the engineer seeking to choose the most appropriate learning technique, the main contribution of Inequality (7.5) is that it helps him compare PAC-learnability of classifiers constrained by different "languages." For instance, we have seen that a

conjunction of attribute values can be learned from a reasonably sized training set, whereas a general concept defined by *any* boolean function usually cannot.

Some other corollaries of this analysis will be the subject of Sect. 7.3.

What Have You Learned?

To make sure you understand this topic, try to answer the following questions. If you have problems, return to the corresponding place in the preceding text.

- What is the size of a hypothesis space consisting of conjunctions of attribute values? Substitute this size to Inequality (7.5) and discuss the result.
- Can you find the value of $|H|$ for some other class of classifiers?
- Explain, in plain English, why a boolean function in its general form is not *PAC-learnable*.

7.3 Some Practical and Theoretical Consequences

The theoretical analysis from the first section of this chapter offers a broader perspective of the learning task as well as clues whose significance cannot be overstated. Their benefits range from purely intellectual satisfaction enjoyed by a theoretician to practical guidelines appreciated by the most down-to-earth engineer. Let us take a quick look at some of them.

Bias and Learnability Section 7.2 investigated *PAC-learnability* of classes whose descriptions are limited to conjunctions of attribute values. A constraint of this kind is called a *bias* for a specific type of class.

Allowing for some attributes to be ignored as irrelevant, we calculated the size of the hypothesis space thus defined as $|H| = 3^n$. If we strengthen the bias by insisting that every single attribute must be involved, we will reduce the size of the hypothesis space: $|H| = 2^n$. This is because every attribute in the class-describing expression is either *true* or *false*, whereas in the previous case, a third possibility ("ignored") was permitted.

Many other biases exist, some limiting the number of terms in the conjunction, others preferring a disjunction or some pre-specified combination of conjunctions and disjunctions, or imposing yet another constraint. What matters, for our discussion, is that each bias is likely to result in a different size of the hypothesis space. And, as we have seen, this size affects learnability.

No Learning Without a Bias In the absence of *any* bias, allowing for any boolean function, we have established that the value of $\ln |H|$ grows exponentially in the number of attributes, $\ln |H| = 2^n \ln 2$, which means that the class is in this most general form *not PAC-learnable*. The explanation is simple. The unconstrained

hypothesis space is so vast that there is a reasonable chance that one may find a classifier that labels correctly the entire training set, and yet exhibits poor performance on future examples. Put bluntly, the induced classifier cannot be trusted.

It is in *this* sense that we say, with a grain of salt, that "there is no learning without a bias."

The thing to remember is that the machine-learning adventure can only succeed if the engineer constrains the hypothesis space by a meaningful bias. It stands to reason, however, that this bias should not be misleading. The reader will recall that our analysis assumed that the hypothesis space *does* contain the solution, and that the examples are noise-free.[4]

Occam's Razor Quite often, the engineer has an opportunity to choose from two or more biases. For instance, the class to be learned is described by a conjunction of attribute values, in which each single attribute plays a part. A weaker bias that permits the absence of some attributes from the conjunctions also includes the case where *zero* attributes are absent, and is therefore correct, too. In the former case, we have $\ln |H| = n \ln 2$; and in the latter, $\ln |H| = n \ln 3$, which is bigger. We thus have two correct biases, each defining a hypothesis space of a different size. Which of them to prefer?

The attentive reader no doubt already knows the answer. In Inequality (7.5), the number of training examples needed for successful learning depends on $\ln |H|$. A lower value of this term indicates that fewer examples are needed if we want to satisfy the given (ϵ, δ)-requirements. The engineer will thus benefit by choosing the bias whose hypothesis space is smaller.

By the way, scientists have been using a similar rule for centuries: in a situation where two different hypotheses can explain a certain phenomenon, it is assumed that the simpler one has a higher chance of success. The fact that this principle, *Occam's Razor*, has been named after a scholastic theologian who died in the fourteenth century indicates that the use of this rule pre-dates modern science. The word *razor* is here to emphasize that, when formulating a hypothesis, we better slice away all unnecessary information.

That mathematicians have now been able to find a formal proof of the validity of this principle in the field of machine learning, and even to quantify the scope of its utility, is a remarkable achievement.

Irrelevant and Redundant Attributes In the types of classes investigated in Sect. 7.2, the lower bound on the number of examples, m, depended on the number of attributes, n. For instance, in the case of conjunctions of attribute values, we now know that $\ln |H| = n \ln 3$; and the number of examples needed to satisfy given (ϵ, δ)-requirements thus grows linearly in n.

[4] Analysis of situations where these requirements are not satisfied would go beyond the scope of an introductory textbook.

The same result enables us to form an opinion about how a class learnability might be affected by the presence of irrelevant or redundant attributes. The higher the number of these attributes, the higher the value of n, which means that more training examples have to be provided if we want to satisfy the (ϵ, δ)-requirements. The lesson is clear: whenever we have a chance to identify (and remove) the less-then-useful attributes, we better do so.

These considerations also explain why it is so difficult to induce a good classifier in domains where only a tiny percentage of attributes carry the relevant information—as is the case in text classification.

What Have You Learned?

To make sure you understand this topic, try to answer the following questions. If you have problems, return to the corresponding place in the preceding text.

- What is meant by the statement, "there is no learning without a bias"? Conversely, under what circumstances will the engineer's bias tend to hurt the learning algorithm's chances of success?
- Explain the meaning of the term, *Occam's Razor*. In what way did mathematicians provide a solid ground for this—formerly philosophical—principle?
- What does Inequality (7.5) tell us about the role of irrelevant and redundant attributes?

7.4 VC-Dimension and Learnability

So far, we have only dealt with domains where the number of hypotheses is finite. Let us now turn our attention to domains where the hypothesis space is infinite, which is often inevitable in applications where at least some of the attributes are continuous-valued. In this situation, it would appear that comparing the learnability of two classes of classifiers is all but impossible because both of them have infinite $|H|$ anyway.

To begin with, the reader will agree that some classes of classifiers (say, polynomial) are more *flexible* than others (say, linear). Surely this flexibility is likely to have some impact on learnability? Indeed it does. Let us take a quick look at how theoreticians quantify learnability under this situation.

Shattered Training Set Consider the three examples in the two-dimensional space depicted on the left of Fig. 7.1. No matter how we distribute the positive and negative labels among them, we will always be able to find a linear classifier that separates the two classes (the classifiers are shown in the same picture). We say that this particular set of examples is "shattered" by the class of linear classifiers.

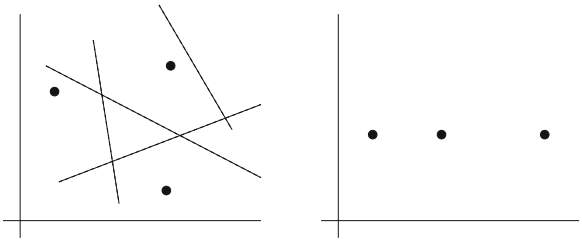

Fig. 7.1 The set of the three points on the *left* is "shattered" by a linear classifier. The set of the three points on the *right* is *not* shattered by a linear classifier because no straight line can separate the point in the middle from those on the sides

This, however, is not the case of the three points on the right. As we can see, these points find themselves all on the same line. If we label the one in the middle as positive and the other two as negative, no linear classifier will ever succeed in separating the two classes from each other. In other words, this particular set of examples is *not* shattered by the linear classifier.

A different class of classifiers will have a different power to shatter a given set of examples. For example, a parabola (a special kind of a quadratic, and thus polynomial function) will shatter the three aligned points on the right; it will even shatter four points that do not lie on the same line. And other classes of classifiers, such as high-order polynomials, will shatter any realistically sized set of examples.

Vapnik-Chervonenkis Dimension The Vapnik-Chervonenkis dimension (usually abbreviated as *VC-dimension*) of a given class of classifiers is defined as the size of the largest set of examples shattered by this class.

We have seen that, in a two-dimensional space, a linear classifier fails to shatter three points that are all on the same line, but that the linear classifier *does* shatter them if they do *not* lie on the same line. At the same time, four points, no matter how we arrange them in a plane, can always be labeled in a way that makes it impossible to separate positive examples from the negative linearly. Since the definition says, "the largest set of examples shattered by this class," we are bound to conclude that the VC-dimension of a linear classifier in a two-dimensional space is $VC_L = 3$.

The point to remember is that the value of the VC-dimension reflects—and quantifies—the geometrical properties of the given class of classifiers.

Learnability in Continuous Domains The concept of VC-dimension makes it possible for us to deal with learnability in continuous domains. Let us give here a slightly modified version of a famous theorem, omitting certain technicalities that are irrelevant for our specific needs:

If the VC-dimension, d, of a classifier class, H, is finite, then an error rate below ϵ can be achieved with confidence $1 - \delta$ if the target class is identical with some hypothesis $h \in H$, and if the number of the training examples, m, satisfies the following inequality:

Table 7.2 VC-dimensions for some hypothesis classes in R^n

Hypothesis class	VC-dimension
Hyperplane	$n + 1$
Hypersphere	$n + 2$
Quadratic	$\frac{(n+1)(n+2)}{2}$
r-Order polynomial	$\binom{n+r}{r}$

$$m \geq \max\left(\frac{4}{\epsilon}\log\frac{2}{\delta}, \frac{8d}{\epsilon}\log\frac{13}{\epsilon}\right) \tag{7.9}$$

Note that this means that the lower bound on m is either $(\frac{4}{\epsilon}\log\frac{2}{\delta})$ or $(\frac{8d}{\epsilon}\log\frac{13}{\epsilon})$, whichever is greater.

The engineer interprets this result as telling him that he can then trust any classifier which correctly classifies the entire training set of size m, regardless of the machine-learning algorithm that has induced the classifier.

Note that the number of examples necessary for PAC learning grows linearly in the VC-dimension. This, however, is no reason to rejoice. Thing is, VC-dimensions of many realistic classes of classifiers have a way of growing very fast with the growing number of attributes—see below.

Some Example VC-Dimensions Table 7.2 lists some VC-dimensions for linear and polynomial classifiers. The reader may want to contemplate the inevitable trade-off: more complex classes of classifiers are more likely to contain the correct solution, but the number of training examples needed for success increases so dramatically that a classifier from this class cannot be deemed learnable.

Indeed, in the case of higher-order polynomials, the demands on the training-set size become all but prohibitive. For instance, the VC-dimension of a second-order polynomial in 100 dimensions (a fairly normal number of attributes) is as follows:

$$d = \frac{102 \cdot 101}{2 \cdot 1} = 5050$$

This is much larger than that of a linear classifier, but perhaps still acceptable. At any rate, the value is sufficiently high to make the first term in Inequality (7.9) small enough (by comparison) to be ignored.

If, however, we increase the polynomial's order to $r = 4$, the VC-dimension will become all but prohibitive:

$$d = \frac{104 \cdot 103 \cdot 102 \cdot 101}{4 \cdot 3 \cdot 2 \cdot 1} = 4{,}598{,}126$$

Polynomials of higher order therefore better be avoided. On the other hand, the VC-dimensions of neural networks and decision trees (see Chaps. 5 and 6) are known to be more affordable—which is why these classifiers are preferred.

A Word of Caution Just as in the case of Inequality (7.5), this one (Formula (7.9)) is the result of a worst-case analysis during which certain simplifying assumptions were made. The results therefore should not be interpreted as telling us how many examples are needed in any concrete application.

Importantly, in realistic applications, examples are not labeled arbitrarily. Since the examples of the same class are somehow similar to each other, they tend to be clustered together in the instance space.

What Have You Learned?

To make sure you understand this topic, try to answer the following questions. If you have problems, return to the corresponding place in the preceding text.

- How does the number of examples that are necessary for the (ϵ, δ)-requirements grow with the increasing VC-dimension, d?
- What is the VC-dimension of a linear classifier in the two-dimensional continuous space?
- What is the VC-dimension of an n-dimensional polynomial of the r-th order? Why does the high value of this VC-dimension mean that polynomial classifiers may not be a good choice?

7.5 Summary and Historical Remarks

- The *Computational Learning Theory* has been able to establish certain limits on the number of the training examples needed for successful classifier induction. These limits depend on two fundamental parameters: the threshold on error rate, ϵ, and the upper bound, δ, on the probability that m training examples will succeed in eliminating all classifiers whose error rate is above ϵ.
- If all attributes are discrete, the minimum number of examples needed to satisfy the (ϵ, δ)-requirements is determined by the size, $|H|$, of the given hypothesis space. Here is the classical formula:

$$m > \frac{1}{\epsilon}(\ln |H| + \ln \frac{1}{\delta}) \tag{7.10}$$

- In a domain where some attributes are continuous, the minimum number of examples needed to satisfy the (ϵ, δ)-requirements is determined by the so-called VC-dimension of the given class of classifiers. Specifically, if the value of the

VC-dimension is denoted by d, then the number of the needed examples is determined by the following inequality:

$$m \geq \max\left(\frac{4}{\epsilon}\log\frac{2}{\delta}, \frac{8d}{\epsilon}\log\frac{13}{\epsilon}\right) \tag{7.11}$$

- Inequalities (7.5) and (7.9) have been found by a worst-case analysis. In reality, therefore, much smaller training sets are usually sufficient for the induction of high-quality classifiers. The main benefit of these formulas is that they help us compare learnability in different machine-learning paradigms.

Historical Remarks The principles underlying the idea of PAC learning were proposed by Valiant [91]. The classical paper on what later came to be known as VC-dimension is Vapnik and Chervonenkis [94]; somewhat more elaborate version was later developed by Vapnik [92]. The idea to apply VC-dimension to learnability, and to the investigation of Occam's Razor is due to Blumer et al. [6]. Tighter bounds were later found, for instance, by Shawe-Taylor et al. [86]. VC-dimensions of linear and polynomial classifiers can be derived from the results published by Cover [18]. Readers interested in learning more about the *Computational Learning Theory* will greatly benefit from the excellent (even if somewhat older) book by Kearns and Vazirani [41].

7.6 Exercises and Thought Experiments

The exercises are to solidify the acquired knowledge. The ambition of the suggested thought experiments is to let the reader see this chapter's ideas in a different light and, somewhat immodestly, to provoke his or her independent thinking.

Exercises

1. Suppose that the instance space is defined by the attributes used in the "pies" domain from Chap. 1. Determine the size of the hypothesis space if the classifier is to be a conjunction of attribute values. Consider both cases: the one that assumes that some attributes might be ignored as irrelevant (or redundant), and the one that insists that all attributes must take part in the conjunction.
2. Return to the case of conjunctions of boolean attributes from Sect. 7.2. How many more examples will have to be used (in the worst-case analysis) if we change the required error rate from $\epsilon = 0.2$ to $\epsilon = 0.05$? Conversely, how will the size of the necessary training set be affected by changes in δ?
3. Again, consider the case where all attributes are boolean, and the classifier has the form of a conjunction of attribute values. What is the size of the hypothesis space

if the conjunction is permitted to involve exactly three attributes? For instance, here is one conjunction from this class:

```
att1 = true AND attr2 = false AND att3 = false
```

4. Consider a domain with $n = 20$ continuous-valued attributes. Calculate the VC-dimension for a classifier that has the form of a quadratic function, and compare it with that of a third-order polynomial.

   Second, suppose that the engineer has realized that half of the attributes are irrelevant. Having removed them, he now has $n = 10$. How will this reduction affect the VC-dimensions of the two classifiers?
5. Compare the *PAC-learnability* of the boolean function involving 8 attributes with the *PAC-learnability* of a quadratic classifier in a domain with 4 numeric attributes

Give It Some Thought

1. A certain role in Inequality (7.5) is played by δ, a term that quantifies the probability that a successful classifier will be induced from the given training set. Under what conditions can the impact of δ be neglected?
2. From the perspective of *PAC-learnability*, is there a difference between irrelevant and redundant attributes?
3. We have seen that a classifier is not *PAC-learnable* in the absence of a bias. The bias, however, many not be known. Suggest a learning procedure that would induce a classifier in the form of a boolean expression in this case. (Hint: consider two or more alternative biases.)
4. In the past, some machine-learning scientists considered the idea of converting continuous attributes into discrete ones by a process called *discretization*. By this they meant dividing the range of attribute values into intervals, each interval treated as a boolean attribute (the given numeric value either is or is not in the given interval).

   Suppose you are considering two ways of dividing the range $[0, 100]$. The first consists of two subintervals, $[0, 50]$, $[51, 100]$, and the second consists of ten equally sized subintervals: $[0, 10], \ldots [91, 100]$. Discuss the advantages and disadvantages of the two options from the perspective of *PAC-learnability*.