Chapter Four

Mechanistic Explanations of Nature

What might we look for?

> *I hope it will not shock experimental physicists too much if I say that we do not accept their observations unless they are confirmed by theory.*
> Arthur Eddington[1]

On February 28, 1953, Francis Crick announced to the patrons of the Eagle pub in Cambridge, England, that he and James Watson had discovered the "secret of life." What they had discovered was the double-stranded helical structure of DNA, the molecule that by then was suspected to be the carrier of heredity. This structure, with the two strands containing identical information, was suggestive of the process by which cells might copy their DNA during replication. The two strands simply separate, each strand carrying all the information it needs to give rise to a new double-stranded version of itself in its own new cell.

Something that became alarmingly clear after the content of the human genome had become largely known is that knowledge of the DNA sequence does not by itself unlock all the secrets of life. The sequence specifies the circuits of human biochemistry, but in a code we understand only partially. More than half a century after Crick and Watson's discovery, we still know little about how knowledge of the sequence can be exploited to understand the physical processes inside the living cell or to help cure disease. Despite everything that we know we do not know, we do have some insight into the computational nature of DNA. A strand of DNA consists of a sequence of nucleobases, each of which is one of four different chemicals, adenine, guanine, thymine,

and cytosine. The sequence of bases contains the information that is carried by the living cell and inherited by its offspring. It may seem elementary, but it is still noteworthy that the way the information is represented in DNA is the same as it is represented in a Turing machine, as a sequence of symbols from a fixed alphabet. In the case of DNA the alphabet has the four symbols A, G, T, C, standing for the four nucleobases. And as Turing showed, a one-dimensional sequence of symbols from a fixed finite alphabet can describe and support all computations.

That the information in DNA is stored in a sequence is simply the first and most immediate of the many ways in which biology may be viewed as computational. When the cell divides, the base sequences are scanned for copying much like Turing machine tapes are during computations. Random mutations are realized by bases changing one to another just like randomized Turing machines would change a symbol. Since errors may be made in copying, methods are also needed for correcting errors.

The operations carried out in living cells and larger structures, such as our neural networks, can be usefully viewed as computations at many deeper levels as well. One is at the level of the protein expression circuits that the DNA sequences define. At any one time some of the proteins are expressed (produced) in the cell, and these in turn cause other proteins to be expressed according to the interdependencies specified in the protein expression circuit. On a different scale, the nervous system can be viewed equally as a very large circuit that performs elaborate computations that we as yet also understand only a little.

We can also ask the higher level question of how these protein or neural circuits are themselves created and maintained. Evolution is realized by modifications in the DNA sequences and hence in the protein circuits. These modifications can be regarded as computations also. With regard to neural networks, organisms learn during life by adapting their neurons in response to events. These adaptations are again computations.

An early example of a computational view of biology was given by Turing himself, in his theory of morphogenesis, or the development of shape. This has had considerable influence on thinking about how the many cells of the embryo can differentiate themselves and take up their various roles in a complex organism, despite having arisen from one unspecialized cell. Among other things, Turing suggested that the wide variation of the dappled patterns on animal fur, whether Dalmatians or leopards, might be accounted

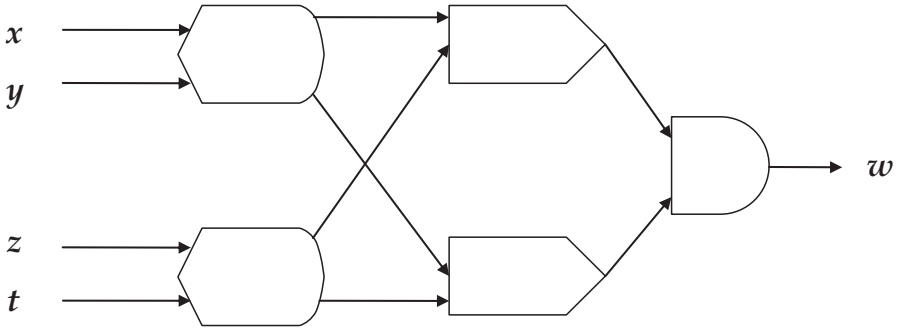

Figure 4.1 An illustration of a circuit. A situation is described by values *x*, *y*, *z*, and *t* that are input to the circuit. The value of the response of the circuit is *w*. Each circuit component performs some operation on the input values or on the results of previous operations. A circuit can be regarded as a general computation where the dependence among the various inputs, outputs, and intermediate computed values can be made explicit, as in this diagram. A neural or protein circuit will be effective if its response is beneficial to the owner of the circuit in typically encountered situations. For theoryless decisions it is sufficient that the circuit be effective in situations that are most frequently encountered by the owner—no theory or understanding of why it is effective is needed. Ecorithms are the mechanisms by which such circuits are acquired and kept in tune.

for by random variation during development, even if the animals are genetically all identical. Turing demonstrated his suggestions by simulations (as shown in Figure 4.2). He was giving one of the earliest examples of computational science, the idea that facts about the world can be discovered not just by physical experimentation or by positing theories, but also by computational simulations. Such simulations can sometimes pursue the consequences of hypothesized theories beyond where mathematical analysis is able to go.

Biology therefore is based on complex mechanisms at many different levels that are as yet little understood. What Crick and Watson had done was to discover the physical substrate on which heritable information is represented, much like silicon is the physical substrate of present-day computers. For both substrates it is impressive how the exacting requirements imposed on them can be achieved with as much miniaturization and economy as they are. However, no one would say that the secret of computers is in the silicon, since computers can be equally well realized in many other physical substrates, though perhaps not quite so economically at present. Indeed, one reason that computer development has been as rapid as it has is

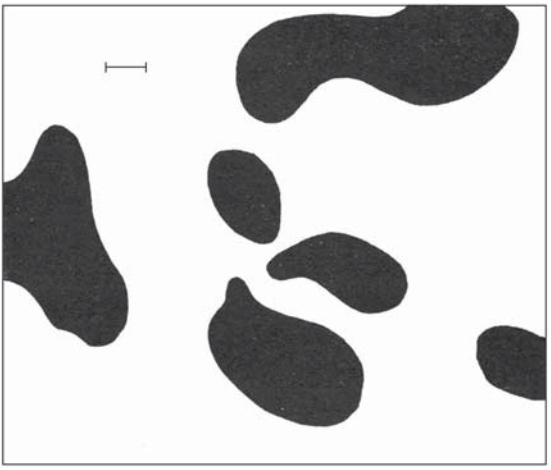

Figure 4.2 A dappled pattern reminiscent of animal fur derived by Turing by means of a computational process.[2] The particular pattern obtained is determined by minor random variations made in early stages of the process, rather than by any preprogrammed or genetic element. The short horizontal line is a scale indicator of the generating process. One gets very different but equally natural-looking patterns every time one runs the randomized process. Turing comments that he obtained this diagram in a "few hours by a manual computation"—evidently he did not have a machine available. (Copyright © 1952, The Royal Society)

that computer scientists made a conceptual separation at the very beginning between the physical technology in which the computer was implemented and the algorithmic content of what was being executed on the machines. This enabled hardware, software, and algorithms to evolve independently, and at their own spectacular rates.

Making similar headway in our study of biology, whether evolutionary or cognitive, demands the same separation of algorithm and substrate.[3] The distinction made here between a physical object and the information processing it performs is self-evident for anyone dealing with computers. The distinction is in no way subtle. Even for a traffic light one can easily distinguish between its symbolic function and its physical construction. But perhaps these distinctions were not quite so obvious in former times. The mind-body problem of Descartes and his followers may have been an earlier reference to such a distinction. But now when computers are ubiquitous, there is no reason for confusing "what it does" and "what does it."

Before moving on to the question of which algorithms might be realized by biology, I note that learning theory may inform the investigation of biology in a different sense also. A biologist performing experiments can be regarded as a learner who wishes to uncover the complex mechanisms of a particular system. As we shall see, there are inherent limits to the complexity of mechanisms that can be learned. The limits on what is learnable, which we will see in the next chapter, may be viewed as warning signals that the accumulation of experimental behavioral data by itself may not necessarily lead to progress in understanding how a system works. Individual human behaviors have been closely observed and widely recorded for thousands of years, yet we understand little about the mechanisms of the brain that gave rise to these behaviors.

After Alan Turing died, Max Newman, his mentor and friend, described in an obituary the central theme that had inspired Turing's many contributions to science: "The varied titles of Turing's published work disguise its unity of purpose. The central problem with which he started, and to which he constantly returned, is the extent and the limitations of mechanistic explanations of nature."[4] This is an insightful characterization, which we owe to someone who had known Turing well. It emphasizes the need for studying "extent and limitations," both of which were to become fundamental characteristics of computer science. The characterization further asserts the novelty of Turing's quest in suggesting that, while the established sciences—physics, chemistry, and biology—also aim for mechanistic explanations, nature also requires explanations of a kind that these older sciences do not address. In Turing's mid-twentieth-century writings one can already detect the pulse of the twenty-first century. Turing's place in history is assured by his discovery and successful pursuit of this previously unsuspected dimension to science.

Chapter Five

The Learnable

How can one draw general lessons from particular experiences?

> *All generalizations are false, including this one.*
> Mark Twain

5.1 Cognition

The idea that biological and cognitive processes should be viewed as computations appeared almost immediately upon the discovery of universal computation, and it was discussed by the early pioneers, including Turing and von Neumann. Because of subsequent slow progress in making this connection concrete or useful, some have despaired that it can never be made into more than metaphor, and that for fundamental reasons it cannot be made into a science. I disagree. I believe that developing any new science is fraught with challenges, and that we are making progress in this area at about the pace that might be reasonable to expect.

The universality of computation is what justifies this approach to cognition. Some have complained that the favored metaphor for the brain in every age has been the most complicated mechanism known at the time. Since the computer is currently that most complex mechanism, is it not a fallacy to adopt that metaphor? I would argue that the computer analogy goes beyond the fact that the computer is another complicated mechanism. What makes it different this time is the widely agreed universality of computation over all processes that we regard as mechanistic.

Turing and von Neumann both shared a second crucial insight: that mathematical logic, from which computation theory had emerged, was not the right grounding for a computational description of either thinking or life. In particular, Turing has the following memorable conclusion to his paper describing noncomputability results in logic: "The results which have been described in this article are mainly of a negative character, setting certain bounds to what we can hope to achieve purely by reasoning. These, and some other results of mathematical logic, may be regarded as going some way toward a demonstration, within mathematics itself, of the inadequacy of 'reason' unsupported by common sense."[1] This passage may be the first occurrence in science of the idea that common sense is somehow superior to reason. It foreshadows ample computational experience in the years that followed. While computers are extremely good at reasoning using mathematical logic, they find common sense much more challenging.

We are faced with two issues as a result: identifying what it is about common sense that logic fails to capture, and whether there is a scientific road to the problem of common sense. The first issue, I argue, is a result of mathematical logic requiring a theoryful world in which to function well. Common sense corresponds to a capability of making good predictive decisions in the realm of the theoryless. To address the second issue we need therefore a theory of the general nature of the theoryless. As I shall argue, the road we must take in that direction is paved with ecorithms.

The algorithms studied most widely in computer science aim to solve instances of some specific problem, such as integer multiplication or the Traveling Salesman Problem. These algorithms, by design, already incorporate the expertise needed for solving them. Ecorithms are also algorithms, but they have an important additional nature. They use generic learning techniques to acquire knowledge from their environment so that they can perform effectively in the environment from which they have learned. They achieve this effectiveness not by intensive design, but by making use of knowledge they have learned. The designed-in expertise is limited to generic learning capabilities, and their use. Understanding ecorithms requires developments beyond basic algorithmic theory. One now needs to analyze not only the algorithm itself but also the algorithm's relationship with its environment.

The theory of probably approximately correct, or PAC, learning deals with this relationship between the algorithm and its environment. It addresses the

fundamental question of how a limited entity can cope in a world that in comparison is limitless, and does so while keeping to an absolute minimum any assumptions about that limitless world.

5.2 The Problem of Induction

Living organisms from the lowliest have some capability to adapt. They learn to avoid doing actions that are detrimental to themselves in favor of those that are beneficial.

In real-world environments an almost limitless number of distinct possible situations may occur. A useful learning capability therefore always needs to provide a significant component of generalization; a learned behavior has to be effective not only in situations that are identical to ones previously experienced but also in any number of novel ones. For this reason I identify generalization as the core of the learning phenomenon. Remembering a list of a hundred words shown once may be a challenge for us humans, but this is best regarded as a bug of our neurobiology, the legacy hardware architecture our species inherited. Because our brains lack the means of manipulating memory addresses in the way computers are able to do, and because each neuron is connected to only a small fraction of the others, memorization is unnecessarily difficult.[2] However, we humans are excellent at generalizing, a skill that is both philosophically fraught and difficult to endow in our computers.

There is a difficulty in placing generalization at the core of learning, at least for philosophers, who have argued for millennia that it is difficult to make a logical argument for rationally inferring anything from one situation to another that one has never before experienced. This is known as the problem of induction. Aristotle said that there are two forms of argument, syllogistic and inductive.[3] Here I interpret these words to mean that if one has a certain belief, then the belief was arrived at either by logical deduction (syllogism) from things already believed, or by induction (generalization) from particular experiences. In this formulation it is induction that is the more basic since it enables primary beliefs, whereas logical deduction requires some previous beliefs.

The main paradox of induction is the apparent contradiction between the following two of its facets. On the one hand, if no assumptions are made about the world, then clearly induction cannot be justified, because the world could conceivably be adversarial enough to ensure that the future is

exactly the opposite of whatever prediction has just been made. This skeptical position is ancient. For example, the philosopher Sextus Empiricus wrote some 1,800 years ago:

> [The dogmatists] claim that the universal is established from the particulars by means of induction. If this is so, they will effect it by reviewing either all the particulars or only some of them. But if they review only some, their induction will be unreliable, since it is possible that some of the particulars omitted in the induction may contradict the universal. If, on the other hand, their review is to include all the particulars, theirs will be an impossible task, because particulars are infinite and indefinite. Thus it turns out, I think, that induction, viewed from both ways, rests on a shaky foundation.[4]

On the other hand, and in apparent contradiction to this argument, successful induction abounds all around us. Generation after generation, millions of children learn everyday concepts, such as dogs and cats, chairs and tables, after seeing examples of them, rather than precise definitions. Each child will typically see few examples of each concept, and the examples different children see will in general be different. Nevertheless, when asked to categorize a new example as to whether it is a cat or a dog, children will agree with each other on a high percentage of occasions, perhaps surprisingly high given the paucity and variability of the information they have been provided. From this we have to conclude that generalization or induction is a pervasive phenomenon exhibited by children. It is as routine and reproducible a phenomenon as objects falling under gravity. It is reasonable to expect a quantitative scientific explanation of this highly reproducible phenomenon.

While these two facets, the difficulty of justifying induction without assumptions, on the one hand, and the pervasiveness of induction, on the other, are on the surface contradictory, they are not implacably inconsistent. There may exist some acceptable assumptions that hold for the reproducible, naturally occurring form of induction, and under which induction is rigorously justifiable. I argue that this is exactly the case, and that just two assumptions are sufficient to give a quantitatively compelling account of induction. Further, these two particular assumptions are also necessary and unavoidable.

The first assumption is the Invariance Assumption: The context in which the generalization is to be applied cannot be fundamentally different from that in which it was made. If I move from one city to another, then I can benefit from my previous experience only on the assumption that things are not too different in the two cities. To put it a bit more mathematically, this assumption requires that the functional relationships and the probability distribution D that characterizes how frequently different situations arise remain somehow constant over time. It is important to note that the Invariance Assumption does not require that the world not change at all. It requires only that there are some regularities that remain true. These regularities may even specify how the world tends to change with time: If we observe the Sun going down toward the horizon in an interval of an hour, we expect that in the next hour the sun will go down even closer to the horizon, rather than that it will repeat the previous positions in a zig-zag fashion. Or, as the Wall Street financier J. P. Morgan, on being asked by a questioner for a prediction about the future course of the stock market, said: "It will fluctuate."

The second assumption is the Learnable Regularity Assumption. We are quite good, but possibly not perfect, at categorizing. If we look into an aquarium, we can fairly reliably distinguish between plants and animals, even species we have not seen before. We must be doing this by applying some criterion that distinguishes animals from plants. These criteria can be viewed as regularities in the world. Such regularities have been discussed as such by philosophers, notably by David Hume in the eighteenth century. Computer science adds at least two further levels to this discussion. First, it is essential to require that any useful criterion or regularity be detectable: Whether the criterion applies to an instance should be resolvable by a feasible computation. For example, the number of measurements we need to make on the object in question, and the number of operations we need to perform on the measurements to test whether the criterion of being an animal holds or not, should be polynomially bounded. A criterion that cannot be applied in practice is not useful.

However, the induction phenomenon has a second, even more severe, further constraint on it. It is not sufficient that the regularity or criterion just exist or even that it is detectable. To explain induction it is also necessary to explain how an individual can acquire the detection algorithm for the regularity in the first place. In particular, this acquisition must be feasible,

requiring only realistic resources and only a modest number of interactions with the world. Of course, different kinds of regularity may require different levels of learning effort. Think about the night sky. Data about the positions of the visible objects has been available to our ancestors from the beginning, there for anyone to see. A little systematic observation revealed the easy-to-learn regularity that all the objects are in fixed positions relative to each other, except the few we call planets. It took thousands of years before someone, namely Kepler, discovered the much-more-difficult-to-discover regularity that the planets move in ellipses.

The Invariance and Learnable Regularity assumptions may seem restricting, but in fact they are liberating: They free the learner from certain responsibilities that are impossible to realize. The Invariance Assumption requires only that predictions hold for examples drawn from the same source as the examples were drawn during learning. If we learn from naturally occurring examples, then we only need to make good predictions about other natural examples. In the case of learning to distinguish animals from plants, this would imply, for example, that accurate predictions on artificial or mythical cases are not required. Computer-generated images of fictitious hybrids, designed to split human opinion exactly fifty-fifty as to whether they are plant or animal, will not be relevant to our interpretation of the induction phenomenon.

The Learnable Regularity Assumption also imposes some liberating limitations. It requires that some regularity exists, and that this regularity be effectively detectable for any example. It goes further in insisting that this regularity be learnable with moderate effort. A case that therefore need not be encompassed is where the examples are natural but then encrypted by some method that cannot be efficiently reversed. Thus the pictures of the animals or plants can be encoded so that they cannot be deciphered by any efficient computational process. This does not remove the regularity from the data if the original data is still recoverable in principle, even if only by an infeasibly laborious computational process. But if it is not practical at all to discover the regularity, then the regularity is no longer a *learnable* regularity, and it need not be addressed.

5.3 Induction in an Urn

I will show that these two minimal assumptions—the Invariance and Learnable Regularity assumptions—enable us to explain the possibility of

induction rigorously. PAC learning is based on these two assumptions of invariance and learnable regularity. The next several sections will develop this idea in more depth.

Suppose you are presented with an urn containing millions of marbles, each one with a number written on it. You can reach in and draw a marble at random from the urn. You are allowed to draw 100 marbles. Your task is to determine which numbers occur at least once among all the millions of marbles in the urn. Is this possible?

The answer is clearly "no" if no assumptions are made at all, since it is possible that all the marbles have different numbers written on them. Any 100 draws will then fail to identify the numbers on the remaining millions of marbles. On the other hand, the answer is clearly "yes" under certain extreme assumptions. For example, if it is known that all the marbles are identical, then a single draw would give complete knowledge about all the marbles.

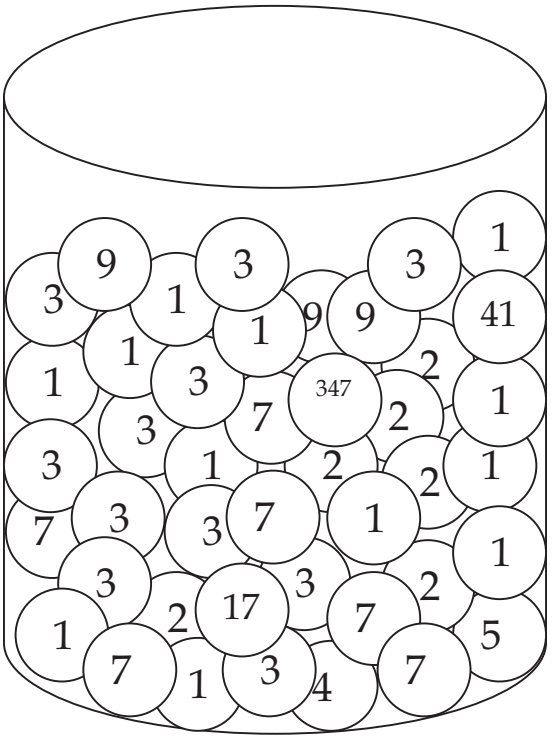

Figure 5.1 Given a large urn containing millions of marbles, can one induce which marble types are in the urn from a small sample drawn at random?

What is a little more striking is that there is a qualified positive answer under some more interesting intermediate conditions. In particular, if the number of different kinds of marble in the urn is not one but a small number, such as five, then one can still achieve a useful level of generalization.

The reason why having a fixed number of marble types makes some level of induction possible is simply the following. Any marble that occurs frequently enough will be among the 100 drawn with high probability, and hence the 100 marbles drawn will be representative of the most frequently occurring marbles, unless, of course, you had particularly bad luck with the draw.

To argue this more precisely, first consider the specific case that as many as 50 percent of the marbles are of one type. Suppose this 50 percent all have a 3 written on them. Then the probability that all 100 random draws miss a 3 is $(1/2)^{100} = 0.000000000000000000000000000000078886\ldots$. This is the same as the probability that 100 tosses of a fair coin all come up tails. The likelihood of this occurring is, of course, extremely small—so small, in fact, that if an experiment of tossing 100 coins had been repeated every nanosecond since the currently estimated date of the Big Bang, then the probability that this eventuality would have ever happened is still less than 1 in 2,000. Therefore, for the urn in which 50 percent of the marbles are 3s, we are safe to conclude that after 100 draws, with overwhelming probability, we will have seen at least one 3, and hence a representative of at least 50 percent of the contents of the urn.

Such a very specific assumption, that 50 percent of the marbles are the same, is not essential either. The knowledge that the number of different marble types is small also gives a sufficient principled basis for induction. Suppose that we know that there are at most five different marble types, but have no idea of their relative frequency. It turns out that after 100 picks we should have high confidence that we have seen representatives of over 80 percent of the contents of the urn. The argument goes like this. If any of the five marble types occurs with frequency more than 5 percent, then the probability that that type was missed all 100 times is less than $(1 - 0.05)^{100} = (0.95)^{100} < 0.6$ percent. Since there are at most five such types, the probability that any one of these frequent ones has been missed is less than five times this quantity, namely 3 percent. There can be at most four types that occur with frequency less than 5 percent each, and hence the rare types account for less than 20 percent of all the marbles between them. Combining the two sources of

error, we conclude that with probability at least 97 percent no marble type that occurs with probability greater than 5 percent has been missed, in which case the missing marble types can account for at most 20 percent of the contents of the urn. Hence you will have drawn representatives of at least 80 percent of the marbles, unless you are unlucky and miss some common types. But you are unlucky in this way less than 3 percent of the time.

In other words, you can predict with 97 percent confidence that after 100 picks you will have seen representatives of at least 80 percent of the contents of the urn. This is in spite of the fact that the distribution of the various types of marbles is arbitrary and unknown to you. It may be that each of the five occurs with equal 20 percent probability. Or it may be that 92 percent are of one kind and the other four kinds occur with frequency 2 percent each. Or it may be that three of the marble types are extremely rare, each occurring 0.1 percent of the time, and the remaining two each occurs 49.85 percent of the time. The claim is equally valid for these three cases, as for any other.

All we need to make this claim are our two assumptions: that the contents of the urn do not change (invariance), and that there are a fixed number of marble types represented in the urn (which provides a sufficient learnable regularity). From this it was possible to deduce totally rigorously that from a small sample one can make meaningful predictions about future draws from the urn.

5.4 Error Control

Clearly, then, induction with our minimal assumptions is powerful. Equally clear is that we cannot avoid errors, which can come from two sources. The first source is that of rarity errors. There may exist in the urn some rare types of marbles that are unlikely to be drawn in any small sample. Their existence therefore will be predicted with correspondingly small probability. The second source is that of misfortune errors. With some small probability the sample drawn will be unrepresentative of the contents of the urn because it missed some common marble types. In extreme enough cases of such misfortune, as when all the common marble types are missed, the sample will not support any useful claims about what is in the urn.

The interesting thing is that while neither of these two sources of error can be totally eliminated, both can be controlled. By this I mean that the probability of both sources of error can be driven down to arbitrarily small

(but nonzero) quantities by increasing the number of marbles drawn. Most significantly, controlling error is affordable: The cost in increased sample size that needs to be paid will depend only modestly, in fact polynomially, on the predictive error that is to be tolerated. That affordability is crucial—if good predictions could be achieved only in an idealistic limit requiring exponentially much effort, then explanations of real-world phenomena would not follow.

In this urn model the only action is to pick marbles and there is no other computation. Hence the computational cost may be regarded as the number of marbles picked from the urn. I shall call this number S. By feasible I have a concrete quantitative notion in mind. I want S to increase only polynomially both with n, the number of different marble types in the urn, and with $1/error$, the inverse of the maximum error one is choosing to tolerate. Here *error* will be either the rarity error or the misfortune error, whichever is smaller. An instance of a polynomial bound for the error is the quadratic bound $(1/error)^2$. Then if one is willing to tolerate 10 percent errors (i.e., $error = 0.1$), then the number of examples and total number of steps needed to achieve that level of accuracy should be proportional to $(1/error)^2 = 10^2 = 100$.

We can find out just how affordable our predictions can be. If we assume, for example, that all n types occur in the urn with the same probability, to draw a set representing half the available types, we will need to choose at least $n/2$. In this case the cost of a good induction will be proportional to the number of available types. It can also be shown that for some distributions the dependence on the inverse error can be similarly linear. This shows that some minimum price may have to be paid for good generalization. Fortunately, it can be shown that for no distribution is the actual price ever much higher than these bounds, which are linear in n and in $1/error$.[5] The argument needed to prove this is a generalization of the one just given for the particular case of five marble types, where it was shown that a sample of 100 marbles was sufficient to get a confidence of 97 percent of having a prediction error less than 20 percent.

5.5 Toward PAC Learnability

The urn example establishes the feasibility of induction in that setting, but it is only a restricted setting; the number of distinguishable objects is small, in fact small enough that it is practicable for the learner to witness a big fraction of them. We have already seen Sextus Empiricus's objections to

such an assumption. In the real world the number of distinguishable objects is so large that no learner can expect to see more than a minute fraction, even in a lifetime. A child will have seen only a small fraction of the millions of distinguishable individual animals and yet be able to classify examples according to species. The onerous requirement on human induction, which the urn example does not satisfy, is therefore that after seeing S examples, it has to be able to generalize over sets that have far more than S distinguishable individuals.

Hence the question we have to ask is this: What kind of induction is feasible that matches the urn example in rigor and practical feasibility, but can induce over sets of as many as, say, 2^S distinguishable types, rather than merely S, but with a cost that is polynomial in S, and not exponential?

Fortunately we can adapt the urn example for learning in such a more expressive and exacting setting. Suppose, for simplicity, that the features we detect when trying to distinguish an animal from a plant on a certain planet all have just yes/no values. Suppose also that there are just twenty such features, including the following: is grey, is red, is green, is brown, is small, is big, has eyes, has legs, has leaves, has long ears, can move, can breathe, and so on. Assuming a criterion in terms of these particular features exists, then for each possible combination of yes/no values of the twenty features that applies to some creature it is completely determined whether that creature is an animal or a plant.

The problem is that twenty features, each being present or absent, can be combined in $2^{20} = 1{,}048{,}576$ ways. Now if we have the opportunity to observe many millions of creatures, each identified as animal or plant, we are essentially back to the urn model. We will see all commonly occurring combinations of features, view each different combination as a type of marble, and will then be able to classify future configurations with the same confidence that the calculation for the urn model justifies. However, a scenario where it is necessary to see exponentially many examples in terms of the natural parameter, here the number of features, is simply unrealistic. Even if not all the 2^{20} different combinations of features occur in nature, a large number may, perhaps in the thousands. Humans can learn from far fewer examples even in cases such as this where the number of distinguishable individuals is really enormous. Our algorithm should be able to, as well.

This brings me to the definition of learnability, or the requirement that we can reasonably impose on a learning algorithm before we declare it to be

successful. In this definition we shall demand that an algorithm should learn from a number of examples that is polynomial in the number of the features n. The need for n or n^2 or even n^3 examples may be acceptable, but exponentially many, such as 2^n, would not. (The number of distinguishable types, however, may be as many as 2^n.) We also want to control the error, and insist that this control be again polynomial. In order to achieve all this, we therefore insist that the computational cost of the process of deriving the induced generalization from the examples is polynomial not only in n and but also in 1/*error*. Note that this computational complexity criterion already implies the polynomial limitation on the number of examples drawn, or the sample complexity, since it takes at least one operation to process each example.

The next question to ask is whether this notion of induction is not so onerous that it is unachievable. We can show that this is not the case—induction in the sense of this definition can be attained for certain useful classes of concepts. One such class is that of conjunctions. A conjunction is an expression that specifies for each feature whether it must hold, it must not hold, or it does not matter whether it holds. An example of a conjunction in the present case is

(can move = **true**) **and** (has eyes = **true**) **and** (is green = **false**).

This expresses the criterion that "can move" and "has eyes" must hold, "is green" must not hold, and the remaining seventeen feature values do not matter. Such a conjunction, in turn, defines the concept of an animal on a certain planet if and only if every animal there satisfies the conjunction (i.e., satisfies all three components) and every nonanimal fails to satisfy it (i.e., fails to satisfy at least one of the three components.)

Now let us assume, for the sake of argument, that the concept we are trying to learn can be expressed by exactly this conjunction. In other words, we are assuming that everything on that planet that can move and has eyes and is not green is an animal, while anything that fails to have at least one of these three properties is not. How would we learn the conjunction efficiently from a modest number of examples? It turns out that the marbles and urn analysis of the previous section can be adapted to apply here also, but with the polynomial bound now in terms of the number of features n, rather than only in terms of the number of distinguishable types, which may be 2^n or exponential in that quantity.

To see this, we can treat each example of an animal or nonanimal as a marble that has written on it for each of the twenty features whether the feature holds for the example or not. We only put marbles corresponding to positive examples of animals in the urn. The marbles representing animals will all be labeled by the statement: can move = **true**, has eyes = **true,** is green = **false**, while the other seventeen features between them can take on any of the 2^{17} different combinations of the remaining feature values in any arbitrary ratios of relative frequency.

Our learning algorithm works like this. It forms a list of the $2n$ possible properties, for each feature one property asserts that the feature is true and the other that it is false. We call this list L. The list L initially contains all $2n$ properties, or forty in our running example. Marbles are then drawn one by one. If a marble is drawn that misses some properties that are in L, those properties will be deleted from L, because they are properties that not all animals share. For example, if one animal is not grey, then greyness cannot be a necessary property for all animals, and this property should not be in the conjunction. After 100 marbles have been drawn, the conjunction of the properties remaining in L is declared to be the hypothesis or criterion for animals.

This procedure, known as the elimination algorithm, induces an accurate criterion for recognizing whether something is an animal. The reason is the following. First, all the properties in the correct conjunction for animals must be present in the final L since every marble in the urn had all the properties that all animals share, and the only properties deleted from L were those that were missing in at least one animal. So the only possible source of error is that some property, such as "has long ears," remained in this final L, while it should not have. This would mean that in 100 trials every animal drawn had long ears. If this property is not essential to animals, then the ones that *lack* this property must be truly rare (rarity error) or we were unlucky in our pick of 100 animals (misfortune error). Exactly as in the urn argument, we can argue here also that, with high probability, the properties that falsely remain in L (e.g., having long ears) must be those that between them exclude no more than a small percentage of animals. In fact, essentially the same polynomial bound can be proved in terms of n (the number of features, not animals) and $1/error$ as for the urn problem.[6]

That correctly classifying animals via conjunctions is no more difficult than the urn problem is perhaps surprising, since there were just n different

marble types in the first scenario, while now there are 2^n different types of animals. Let us therefore reexamine our assumptions.

First, we made the Invariance Assumption that the examples encountered in the testing phase come from the same source as in the learning phase. Examples rarely seen during learning will be equally rare during testing, and therefore less important for the learner to know about.

Second, we made the Learnable Regularity Assumption. In this case we assumed that for the given features a criterion for distinguishing animals from plants could be expressed as a conjunction. This was sufficient because conjunctions can be shown to be learnable, as we have just seen.

As we shall see later, many function classes, because they seem not to have learnable regularity, appear not to be learnable even when the Invariance Assumption holds. In other words, the Learnable Regularity Assumption substantively constrains what learning algorithms can do. That may seem a problem, but in fact such constraints are needed to make learning possible. The fact that the elimination algorithm for conjunctions used only positive examples can help us see why! It may seem impossible to learn to classify animals and plants by looking at only animals. Nevertheless, as we have seen, it is both possible and even rigorously justifiable. The reason is that the constraint that there exists a conjunction that distinguishes one type from another is in itself highly informative. In this case it permits learning from positive examples alone. (As a very loose analogy for how information can be conveyed by constraints, suppose I challenged you to solve a puzzle that I claim to have solved myself. In your search for a solution you would find it helpful to know whether it had taken me ten seconds, ten hours, or ten days, even though this information sheds little light specific to the problem.)

Equally learnable are disjunctions, which are expressions of exactly the same form as conjunctions, except that each **and** is replaced by an **or**, requiring only that at least one of the listed set of the properties holds, rather than that every one of them holds. We would then assume that the concept of an animal is expressible as a disjunction such as x_3 **or** x_7 **or** x'_9, where x'_9 denotes **not** x_9. Like conjunctions, disjunctions are learnable, albeit by relying entirely on negative examples, and again using the elimination algorithm. So, if the algorithm encounters a brown plant, "is brown" gets removed from the list of traits that each guarantee something being an animal.

Eliminating what has to be eliminated, as we do here, is, of course, a long-recognized principle of reasoning. Francis Bacon, in the early seventeenth century, and John Stuart Mill, in the nineteenth, both emphasized its importance.[7] Sherlock Holmes was even more categorical: "When you have eliminated the impossible, whatever remains, however improbable, must be the truth. " Unfortunately for Mr. Holmes, exploiting the elimination method in a foolproof way is rarely practicable. Some cases will remain on the list that are not true. The PAC framework offers the needed analysis of the error that will result.

It is natural to ask whether conjunctions and disjunctions are expressive enough to account for human concepts. One way of phrasing this is to let the features be words in a certain dictionary, and ask whether each word in the dictionary can be expressed in terms of, say, a conjunction of the others. The philosopher Ludwig Wittgenstein argued that the notion of a game has no feature that is common to all instances of it. For example, not every game is won or lost, or played by two people, and so on. This implies that conjunctions are not enough for expressing everyday words in terms of each other, since such conjunctions would have to contain exactly the features that are essential to all instances. A similar argument can be made for disjunctions.

Circuits that consist of **and** and **or** statements, composed in an arbitrary way, rather than in a single layer as in conjunctions or disjunctions, are much more expressive. If one has to learn such a circuit, and intermediate nodes in it do not correspond to natural concepts for which labeled examples are available, then the ability to learn conjunctions and disjunctions is not enough. Indeed, as we shall see later, it is widely believed that some function classes that are more expressive than conjunctions or disjunctions are *not* learnable. The questions of determining the most expressive classes of functions for which learning is still possible are the most fundamental questions of learning theory.

5.6 PAC Learnability

What we have been describing is a notion of probably approximately correct (PAC) learning. When first introduced, the corresponding class was simply called learnable, in analogy with Turing's notion of computable, to indicate that what was being sought was a robust characterization of what was practically learnable by explicit computational means. The "probably" acknowledges misfortune errors, and the "approximately" rarity errors.

Many of the concept classes we have seen so far are PAC learnable: conjunctions and disjunctions are, as well as the class of linear separators discussed in Chapter 3. The perceptron algorithm described there is not quite sufficient to establish this. One impediment is that the number of iterations of the perceptron algorithm will be exponential if some examples are exponentially close to the separator. Fortunately, this impediment can be overcome by using a different algorithm, one based on linear programming.

The critical idea in PAC learning is that both statistical as well as computational phenomena are acknowledged, and both are quantified. There have been earlier attempts to model induction using purely computational or purely statistical notions.[8] However, I believe that combining the computational *and* the statistical provides the key to understanding the rich variety of learning phenomena that exist. The notion of PAC learning is concerned with describing what needs to be achieved in order to constitute induction. It is neutral on both which concept class should be learned and which algorithm is to be used to learn it. What it does do is to offer a quantitative analysis of learning. Which algorithms the human nervous system uses and which classes are being learned are not currently known. But at least we have a way of making these questions concrete.

By now it may have occurred to the reader that the model described is undoubtedly a simplification of the broad range of phenomena that humans manifest in relation to learning. In itself the model addresses the core phenomenon of computationally feasible induction from examples. The model can be and has been extended in numerous directions so as to capture many additional aspects of learning.[9] These directions include allowing for some kind of noise in the data, the concepts changing slowly rather than staying invariant, the learner asking certain questions, or the algorithm working only for specific distributions. Having a definite mathematical model of the inductive process gives a vantage point for investigating these important facets of learning.

5.7 Occam: When to Trust a Hypothesis

The great advantage of a hypothesis generated by a PAC learning algorithm is that it comes with a reliability guarantee. However, it is often the case that we are confronted with a hypothesis about the provenance of which we know nothing. A hedge fund manager, for example, might be told that some specific pattern of price fluctuation has been present in market activity for

some time. He must decide if he should start using this pattern to make investment decisions, even if no information is available about how the pattern was identified or who had identified it. This kind of question also arises routinely in the practice of machine learning, where many algorithms are employed that have not been proved to be PAC learning algorithms but are useful nevertheless. Happily there are some entirely rigorous criteria to apply to such situations also.

The answer lies with Occam algorithms: They provide a rigorous approach, even in such cases of total ignorance about the origins of a hypothesis, and exemplify the role of purely statistical arguments in machine learning.[10] What this approach provides are some conditions under which an unfamiliar hypothesis can be trusted. These conditions make concrete and rigorous the intuition sometimes attributed to the fourteenth-century logician William of Ockham that all things being equal, simpler hypotheses are more likely to be valid than complex ones.

Suppose that you are trying to predict horse races, and that someone gives you data from a hundred past races in which every time the heaviest horse won. Discerning whether the heaviest horse is the sure thing it might appear to be requires several steps. First, you will need to be convinced that the 100 races you are shown were not maliciously selected from among many that overall showed no such clear pattern. Second, you would need further reassurance that the data was not from a different planet. These two requirements are roughly equivalent to the Invariance Assumption of the definition of learnability, that future events will be drawn independently from the same probability distribution as the 100 events in the dataset. Third, you would need to assess the complexity of the hypothesis. It is tempting to bet on the heaviest horse because of the simplicity of the rule "the heaviest will win." It seems unlikely that 100 races would all satisfy such a simple rule just by accident. If the rule were much more complex, for example that the horse's height, the owner's weight, and the trainer's age (all in appropriate units) added up to a prime number, then you would be a little more skeptical, and justifiably so. Even if the winners were totally unpredictable and arbitrary, some prediction rule could always be engineered to match them after the fact, if the rule is allowed to be complicated enough.

This intuition can be made rigorous as follows. Suppose that you have a well-defined language for expressing hypotheses, and suppose that at most N distinct hypotheses can be expressed in this language. Suppose also that

someone gives you a dataset of S examples that all agree with h, one of the N permitted hypotheses. To decide whether to accept this hypothesis h, we need only the Invariance Assumption. Suppose that a fixed rule $h^\star$ is bad, in the sense that it predicts correctly only a fraction p of the examples from the distribution. Then the probability that it will be accurate on every one of the S independently drawn examples will be p^S. This will be an extremely small fraction if S is large and p is substantially less than 1. For example, if $S = 100$, and p is less than 0.8, then the probability that this bad rule will predict correctly every one of 100 random examples is about 0.0000000002, or 2×10^{-10}. This is smaller than the probability that thirty-two successive tosses of a fair coin will all come up heads, an extremely unlikely event.

This argument applies to any one fixed rule. But what if an adversary tried to trick us by choosing the rule to fit the data after having looked at the data set? We know there are certain limits to what the trickster can do. There are a limited number N of hypotheses from which he can choose; for example, if the hypotheses are conjunctions over n variables then $N = 3^n$, since each of the variables has three options, present in the conjunction, present in negated form, or absent. In particular, if the probability that any one bad hypothesis looks good is no more than 2×10^{-10}, then the probability that at least one of the up to N bad hypotheses looks good is no more than N times this, or $N \times 2 \times 10^{-10}$. Even if the trickster had a million different rules to choose from, the odds of him finding one that would be both bad and an effective trick is only 1 in 5,000. In fact, as long as N is much smaller than 5 billion, we can be confident that there will be no rule among the N that classifies all the 100 examples correctly but is in fact less than 80 percent accurate on the distribution D in general.

So, as an example, imagine race horses are coded according to a list of 1,000 traits, and that each hypothesis is a conjunction of three traits, such as "largest **and** darkest **and** oldest." There will be about 166 million distinct hypotheses. (This is because if one makes a sequence of three choices, each with a thousand outcomes, there will be 1 billion outcomes overall. Because the order of the traits, if distinct, doesn't matter, only about one in six such hypotheses is unique.) Hence the probability that even one bad hypothesis among them agrees with all the 100 examples will be slightly more than .03. So if "largest **and** darkest **and** oldest" correctly predicted the winner of 100 randomly chosen races, you would have to be irrational not to bet that way

on the next one. We can make similar calculations, and determine what level of confidence in a given rule is justified, even with fewer than 100 examples, and even when the rule predicts not all but only most examples correctly.

What we have shown is that we can depend on the predictive power of a rule someone has given us if we are sure of three conditions. First, we need to be given a data set of past examples that have high agreement with the rule. Second, we need to know that the rule is from some small class of rules that was fixed before the examples were selected. And third, we must have convinced ourselves that the examples presented to us were chosen randomly and independently from the same distribution from which we will wish to make future predictions—that is, only rules trained on thoroughbreds should be used to bet on thoroughbreds.

Now, what is the relationship between PAC learning and these Occam algorithms? In the case of PAC learning we have a guarantee, ahead of seeing any examples, that the learning algorithm we have in hand, such as the elimination algorithm for conjunctions, will yield a good predictor whenever the hidden function to be learned lies in a certain class of concepts. In the case of Occam algorithms, the PAC-like guarantee of predictive accuracy does not depend on the process by which the hypothesis was generated; it is provided only for one specific hypothesis at a time. But that can be liberating, as the Occam argument then gives us a rational justification for trying learning algorithms that do not always work. If we are lucky, and find a hypothesis that explains the data and is short enough to have predictive power, then we can go ahead and use that hypothesis for making predictions. If we are unlucky, and the hypothesis obtained is too long or does not fit the data sufficiently, then we will recognize this failure immediately and not use it to predict, so that no harm will have been done.

One might say that the most reliable way of testing a hypothesis is to do a test on new data—that is, data that has not been used in deriving the hypothesis. While this is true, it is not free of cost, since it requires that we retain some data that does not inform the hypothesis. Using that extra data, we might be able to produce a better hypothesis. When one does go live with a hypothesis—whether betting on a horse or recommending an investment or a medical treatment—one inevitably has to make an Occam-like decision: Given all the available data what exactly is the best hypothesis that can be deployed?

5.8 Are There Limits to Learnability?

We have seen a variety of learning algorithms. The perceptron algorithm is one; the elimination algorithm is another. Obtaining a succinct hypothesis and appealing to an Occam argument is a third approach. As we shall see in Chapter 9, many learning algorithms are now known and some are already widely deployed. Also, there are no doubt many more algorithms that no one has yet conceived. But where do we look to for ultimate limits? As pointed out earlier, learning is based on a deep interplay of computational and statistical phenomena. If there are limits to learning, then these are the directions in which they will be found.

First consider statistical limits, which, though weak, are significant. These impose a condition on the minimum number of training examples needed in order to learn reliably. This number does depend on the distribution. For easy distributions high accuracy can be reached with few examples. An extreme case of an easy distribution is one that has only one positive example that ever occurs, in which case seeing that once will give all the information that is available or needed. The bounds obtained for worst-case distributions often provide useful guidance on how many examples to use in practice. One can show that, for certain distributions of examples, the minimum number of examples required is proportional to the ratio of the number of variables n and the error to be tolerated.[11]

Computational limits are more severe. The definition of PAC learning requires that the learning process be a polynomial time computation—learning must be achievable with realistic computational resources. It turns out that only certain simple polynomial time computable classes, such as conjunctions and linear separators, are known to be learnable, and it is currently widely conjectured that most of the rest is not.

This intuition can be expressed more precisely with an Occam-style argument. Suppose that a function of n variables that describes some regularity is detectable or computable in, say, n^2 steps. (Note that conjunctions can be detected in about n steps, but some useful regularities may be more complex.) Then the behavior of the function on a little more than n^2 randomly chosen inputs will determine the behavior on most of the exponentially many possible inputs in a PAC sense. In other words, the hidden function will be largely determined, for any distribution D in question, by its values at a polynomial number of inputs. The reason that most such functions do not appear to be learnable with polynomial effort is not that in a polynomial

amount of data the function is not already implicit, but that this implicit information cannot be extracted from that data with polynomial effort.

I believe that the primary stumbling block that prevents humans from being able to learn more complex concepts at a time than they can, is the computational difficulty of extracting regularities from moderate amounts of data, rather than the need for inordinate amounts of data. For example, the difficulty of discovering the elliptical nature of the orbits of the planets was not that the amount of data needed took hundreds of generations to compile, but that elliptical orbits as seen from Earth did not constitute a regularity that humans found easy to extract.

Yet another way of stating the relative roles in learning of computation and statistics is to observe that if the assertion that P = NP (or equivalently, that the NP-complete problems are computable in polynomial time) is true, then all of P would be PAC learnable. Recall that NP is the class corresponding to mental searches. If P is equal to NP, then one can take a polynomial number of random, labeled examples and then simulate in P a machine that realizes the necessary mental search for a hypothesis that agrees with these labeled examples. This would give a hypothesis with an Occam guarantee of good predictive accuracy on future examples. In other words, the truth of the computational assertion that P = NP would imply that all polynomial computable functions would be learnable.

We conclude from this that if we are to understand the limitations of learning, we need to look at computational limitations. Unless it turns out that P = NP, or some other unexpectedly strong enough positive result is true, the notion of knowledge being implicit in data is not sufficient to mark the boundaries of what is learnable.

So what are the computational limitations on learning?

We get a first clue from the study of cryptography. This field is concerned with designing algorithms for encoding a message so that an intended recipient can decode it, but any unwelcome eavesdropper who intercepts the message cannot. To make this decoding possible for the intended recipient, but not for the eavesdropper, the intended recipient must have as a key information that is not available to the eavesdropper.

In traditional cryptography the key is conveyed to the recipient by some secure process, such as inside an actual physically locked box or by a whisper in the ear from a trusted emissary. In public-key cryptography no such secure physical process is needed for transferring the key.[12] Instead, the

recipient generates a "public key" and transfers it to the sender through public means, but the recipient privately retains some information, the "private key." In the RSA system the public key is the product of two large prime numbers p and q, and the private key consists of the prime numbers p and q themselves. Anyone can encrypt a message intended for the recipient with the public key, but only the recipient with the aid of the retained secret private key will be able to decode it in polynomial time. To ensure that the number of potential private keys is so large that it is not practicable for the eavesdropper to simply try them all in turn, the key should be, say, a thousand-digit number, of which there are too many to enumerate.

The conjectured infeasibility of computing p and q is believed to make RSA cryptography immune to attack. This conjectured immunity sheds light on why not all hypotheses in class P are learnable. One type of attack on any cryptographic scheme is known as chosen-plaintext attack. In this a would-be eavesdropper—perhaps an insider—is assumed to have access to the encryption device, and can feed the encryption device with many, possibly carefully chosen, pieces of text and then observe their encodings. From this information the eavesdropper extracts a decryption algorithm (equivalent to the key) that will decrypt any encoded message. In essence, then, a chosen-plaintext attack is similar to our learning scenario: the encoded messages are the examples, the bits of the original raw messages are their labels, and the decryption algorithm encapsulated by the private key is the concept to be learned. Assuming that RSA, or some other encryption scheme, is in fact resistant to chosen-plaintext attack, it follows that the class P is not

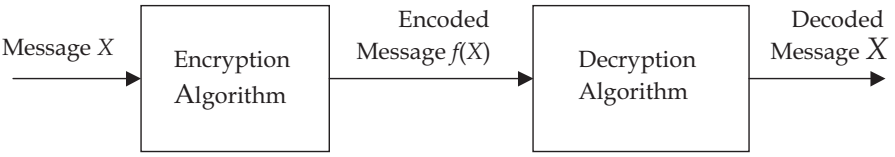

Figure 5.2 The existence of strong encryption methods implies the existence of functions that are computationally intractable to learn. The reason is that if all decryption algorithms were learnable from examples, then one could break any scheme by collecting enough pairs $(X, f(X))$ by feeding the encryption device with enough known messages X and intercepting their encodings $f(X)$. The pairs $(f(X), X)$ are then labeled examples for the decryption algorithm. Learning this algorithm amounts to breaking the code.

learnable, since if it were then we could learn all decryption functions and hence break all such cryptographic schemes.[13]

We do not need to look to just cryptography to find apparent impediments to learnability. Language gives us another domain.

In the 1950s the linguist Noam Chomsky considered various alternative classes of formal languages as the possible grammatical bases of natural languages, such as English.[14] The simplest in this so-called Chomsky hierarchy of formal languages are the regular languages. The mechanisms that generate them are finite-state automata, which are also among the simplest interesting computing mechanisms. An instance is illustrated in Figure 5.3.

Each such automaton can be represented as a diagram with a start node and a number of finish nodes. Each link in the diagram is labeled from a fixed set of symbols, such as *a* and *b*. The language accepted by the automaton is the set of all sequences that label paths in the diagram from the start node to some finish node.

Consider the finite automaton depicted in Figure 5.3. The language this accepts consists of all the sequences of *a*'s and *b*'s of length 5 that begin at the start node on the left and follow paths from left to right and terminate at the finish node. This automaton was generated by first drawing the lattice diagram shown, and then randomly labeling one of the two outgoing edges from each node with an *a* and the other with a *b*. Now, as you might observe, there are thirty-two possible sequences of five letters made from the set of *a* and *b,* but only sixteen distinct paths one can take from start to finish. Hence the language generated by this automaton only accepts one-half of the possible sequences as valid. This construction can be generalized so that if there are *l* letters (or other symbols, such as punctuation or numerals) in an alphabet,

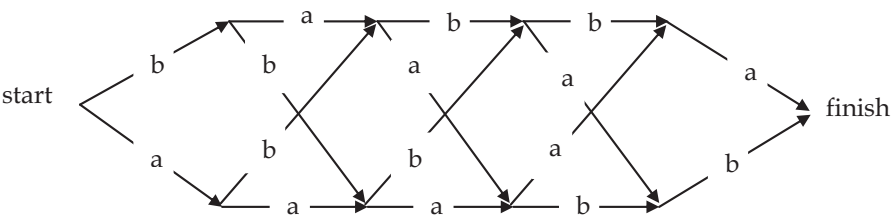

Figure 5.3 An example of a finite automaton of width 2 and depth 5. It accepts half of the sequences of *a*'s and *b*'s of length 5. For example, *bbbab* is accepted, but *aaaab* is not.

and if sentences are of length s, then there are l^s possible sequences and l^{s-1} of these, or a fraction $1/l$, would be accepted by the corresponding automaton.

If we could see all the l^s possible sequences and be told which ones are in the language and which are not, then we could predict new examples, simply because we have all the facts. This is just the urn model again. But this, of course, would be of exponential complexity in terms of the length of sentences.

The task of learning a regular language is that of taking a polynomial number (in terms of sentence length and alphabet size) of training sequences, each labeled according to whether it belongs to the language or not, and on that basis predicting for unseen sequences whether or not they are in the language. Inducing the hidden automaton or an approximation to it would be one possible approach. In principle, it is also possible that a learning algorithm would yield a hypothesis that can label new examples probably approximately correctly, but does not reveal an explicit description of the automaton. Several decades of research following Chomsky, however, failed to uncover such a learning algorithm. This failure was explained in the 1980s by a proof that any algorithm for PAC learning regular languages would imply a method for breaking the RSA cryptosystem.[15] Unless such systems are breakable, no computational learning algorithm for regular languages can exist.

The problem can be understood through the frame of the Occam argument. A random set of polynomially many examples is sufficient to essentially determine the hidden automaton. The difficulty, however, is that this determination is only in the implicit sense that automata that are very different will be most likely inconsistent with the sample. It is in extracting the automaton from this sample by a feasible computation where the difficulty lies—no way is known of doing this extraction in polynomial time. Language learning is of some consequence. We cannot hope to understand what human languages are without understanding how they are learned. A formal language that cannot be learned, cannot be the basis of human language.

All known algorithms for learning regular languages appear to work in exponential time as a function of the length of the sequences of symbols. Whether this can be improved, even for the special class of automata generated uniformly at random using the lattice diagrams as illustrated, is currently an open problem. Any reader not convinced that there are real

computational impediments to generalization should take as a challenge this simpler goal of finding a PAC learning algorithm to categorize the output of a lattice automaton as words or not-words in polynomial time.

Finally, we note that there remain many natural classes of functions whose learnability is currently totally unresolved. For these it is not proven that they are PAC learnable, but equally there is no evidence that they are not. The prime example is the class of functions known as *disjunctive normal form*, often abbreviated as DNF. These are functions that can be expressed as polynomial-size expressions in terms of the number of variables, where the general form of the expression is an **or** statement joining subexpressions consisting of **and** statements of the variables, for example, (*X* **and** *Y*) **or** (*Z* **and** *T*). This can be viewed as a two-level representation, composed of conjunctions and disjunctions (which are themselves one-level representations). DNF is clearly more expressive than disjunctions and conjunctions separately. The best algorithm currently known for learning DNF has complexity exponential in the cube root of the expression size, which is still exponential but not of the worst kind and curiously similar to the best bound known for factoring integers.[16] DNF is an archetypal two-level representation, perhaps close to the boundaries of learnability. Resolving whether or not it can be PAC learned is a major open problem in learning theory.

5.9 Teaching and Learning

Learning a single concept from examples is already a striking natural phenomenon. But even more impressive is how humans can acquire expertise involving many complex interrelated concepts. Of course, humans often take many years to accomplish such feats. Our species invests up to two decades to educate its young, and there is no evidence that this is unnecessarily inefficient. But what view should we take of these more complex accomplishments?

In a college course one expects to learn more than just a set of concepts that could be learned equally well in any order. Rather, one expects to see a sequence of concepts, such that the later ones become accessible to the learner only after the earlier ones have been mastered. This is the paradigm of learning that we adopt here. To put it more mathematically, at any instant we are in a position to learn a new concept that is a member of a learnable concept class *C* with the set *X* of concepts that are already recognized as features. Once we have learned a new concept in this way, it becomes an

added recognizable feature for the purposes of learning additional concepts in subsequent phases of learning. Each such new concept becomes in one's brain an equal citizen with all the features X that were available as features in previous phases, and gets added to X for the next phase. For example, the term "data" has a certain meaning in any time and place. Once we have some familiarity with that meaning, we can learn to recognize derivative concepts, such as that of "big data," that may have been out of reach before "data" had been learned and became a feature. A basic PAC learning algorithm operates up to the level of complexity that is expressible by members of C. At higher levels we can think of the teacher as fulfilling a part of the role of a programmer in defining a sequence of concepts that can be PAC learned in that sequence, but perhaps in no other.

This analogy between a teacher and a programmer also highlights some essential differences between the two. When programming a computer, the programmer needs to understand what exactly the existing programs already implemented on the machine do. This is not the case for a teacher, who does not know exactly the meaning to the learner of each word the teacher is uttering. It is possible for a teacher to get across the notion of "dog" by showing examples without knowing precisely which features of dogs are recognized by the learner, and how exactly these are interpreted. Much more specific knowledge is required of a computer programmer.

This disconnection between the teacher and learner is not entirely deleterious. It offers unique advantages to learning systems that programmed systems lack. A teacher can convey information by suggestions and examples, without knowing the exact state of the learner. In contrast, the programmer has to know the exact state of the system, including the exact functionality of the previously programmed features, if the new program is to work as intended. One can go further and say that the inherent strength of the teacher-learner relationship is that it works even when the exact state of the learner is not known explicitly by anyone, including the learner. This incompleteness in both mutual knowledge and self-knowledge is inevitable among humans and may become increasingly relevant for machines also. It also, I believe, accounts for the difficulty of identifying any general panacea for improving human education. A second clear advantage of learning over being programmed is that it offers a limitless potential to recover from errors. A student who seriously misinterprets some concept is likely, at some

later time, to discover the inconsistency and recover from it. No omniscient external agent is needed to help.

In this formulation the most important role of a teacher is to point out the next good thing to learn. A second role is that of providing labeled examples. The actual labeling is less fundamental since many natural situations can be considered to be self-labeled. If a cat comes along we may recognize it as a cat from previous knowledge, and hence learn more about cats, without a teacher needing to label it as a cat.

A computational theory of learning also provides the following further wrinkle on the role of a teacher. PAC learning guarantees that a concept class is learnable from random examples for any distribution that the environment may provide. The learner will, however, have a specific learning algorithm. A teacher with knowledge of the learner's algorithm will be in a position to accelerate learning. In particular, the teacher may be able to present a short sequence of well-designed examples that drive the learner's algorithm to the correct concept, after many fewer examples than would be needed if they had been random. For example, if we know that conjunctions are being learned, and the elimination algorithm described in Section 5.5 is being used, then a single stylized example that has only the essential features of the concept, and is devoid of any distracting features, would be effective. This is somewhat like bare bones illustrations in books for very young children. An elephant would be shown with prominent tusks and a trunk, but lacking any other detail. Such illustrations would then have the effect of driving the child's learning algorithm to the correct hypothesis after just the one example.

5.10 Learnable Target Pursuit

The ability of humans to acquire knowledge on a large scale is remarkable. At the basic level this must be based, I believe, on the execution of learning algorithms of the same nature as I have described. Supporting this there needs to be an additional capability that I call learnable target pursuit. At any instant any student, or any entity with a learning ability, has available all those features for which recognition algorithms have been previously acquired. The concepts that can be learned by the learning algorithm in terms of the available features are the accessible targets. They are the targets that will be learned by the learning algorithm, provided that labeled

examples of them present themselves. (For this discussion the question of how the recognition algorithms for the currently available features had been previously acquired is not relevant. In biology it would be a combination of evolution and learning. For computers it would be programming and learning.)

I suggest that humans are wired so as to be always ready to pursue any and all accessible targets. This provides a mechanism for continuous learning in any rich environment even in the absence of any teacher. It enables not only previously learned concepts to be fine tuned to higher accuracy, but also entirely new concepts to be learned.

How are examples labeled in the absence of a teacher? As already mentioned, many situations are self-labeled. If I can already recognize both swans and the color black, then I can start learning about black swans if I see one without needing a teacher to identify it. Similarly, if I meet a person I have not seen before, I can start learning about them without a teacher being necessary to identify them or their characteristics.

This capability for learnable target pursuit can operate in the absence of any teacher, but it also provides extra opportunities to be exploited by a teacher. These opportunities include the presentation of examples of concepts for which the student is ready. As already pointed out, progress can also be made without a teacher, but that is dependent more on the serendipity that examples of accessible concepts will somehow appear. Individuals can be smart and seek out experiences that enable them to pursue some useful targets that are accessible given their current knowledge. They can also make the mistake of devoting time to material for which they have insufficient preparation, in which case they may learn little.

5.11 PAC Learning as a Basis of Cognition

I have presented PAC learning as a mathematically rigorous and philosophically satisfactory notion of induction. But I think it is more than that: Its basic features are essential and unavoidable in any attempt to build a theory of cognition.

Humans presumably have some shared learning algorithm. I shall call this algorithm A, and the concept class it learns C. This observation already provides an account of how we can have shared concepts: Whatever I can learn from examples, I can pass on to you by pointing out examples to you, provided we have the same set of previously acquired features. This may also

work if our feature sets are different, as long as the target concept is accessible from both. To date, the algorithm *A* and concept class *C* used by humans have not been identified. But whatever they are, their very existence provides an assumption-free rigorous theory of induction for cognition. It makes no assumptions at all about the world or the distribution of objects in it. The frequencies with which objects have their exponentially many variants with different combinations of properties may be arbitrarily complex and need not be known to the learner. A human having algorithm *A* will manage to learn certain regularities in this complex world, and will miss some others. Other humans will have essentially the same capabilities. Anything you can learn I can learn too, at least in principle.

One much discussed issue in human learning is how some of it seems to occur from positive examples alone. In human behavior it is difficult to establish exactly when this phenomenon can be considered to have occurred. The fact that humans appear to be able to learn to identify members of an animal species from just examples of that species is not conclusive. We may have somehow figured out that each animal belongs to just one species and hence implicitly use positive examples of dogs as negative examples of cats when we are learning the latter. It is therefore not clear to what extent humans do really learn from positive examples alone. But even if we do, there is no inherent mystery in that. As we have seen there do exist learning algorithms, such as the elimination algorithm for conjunctions, that provably do exactly that.

Another issue is that of one-trial learning, or learning from very few examples. As I have previously mentioned, for a fixed algorithm there may be single examples that drive it to the correct hypothesis, and these may be the ones that good teachers provide. There also exist explanations of why a small number of examples may sometimes suffice even without a teacher. In cases where the true concept depends on only a few features among a much larger number of distracting features, so-called attribute-efficient learning is sometimes possible even without a teacher.[17] This means that the number of examples needed for PAC learning is proportional only to the small number of critical features, such as tusks, a trunk, and big ears, and depends only much more weakly on the possibly numerous irrelevant distracting ones. Another possibility is that some concepts may be easy to learn because instances of the category are separable by wide margins from noninstances, in the sense already discussed in the context of perceptrons.[18]

Everything in this chapter requires the Invariance Assumption. In practice, we can never be certain that the world will not change on us in an unexpected way, so that future examples will be from a very different distribution from those in the past. Past performance is not necessarily indicative of future results. Living organisms, however, need to make decisions all the time and take a view on what will happen next. The only course available is to learn as many of the world's regularities as we can, and allow them to guide our decision making. There is simply no alternative.