1

The logic of induction

RICHARD FORSYTH

> Our only hope therefore lies in a true induction.
> Francis Bacon, *First Book of Aphorisms*

> Induction simply does not exist, and the opposite view is a straightforward mistake.
> Karl Popper, *Conjectures and Refutations*

Most of today's so-called 'intelligent' computer systems would qualify as brain-damaged if they were people for the simple reason that they can be relied on to repeat the same behaviour in the same situation again and again and again. Predictability and reliability are desirable features of computer systems, but they are not the hallmarks of intelligence. Indeed a system that keeps repeating the same mistake over and over is, to the lay observer, demonstrably stupid. Yet this is exactly what today's most advanced software systems do (including master-level chess programs, commercially viable expert systems, etc.) – unless they are reprogrammed.

It is a theme of this book that artificial intelligence (AI) is and will remain a misnomer until systems that learn to adapt become commonplace. To put it another way: machine learning is the key to machine intelligence. This fact is still not widely recognized, even though the well-documented difficulties of knowledge elicitation for expert systems have led to a revival of interest in automatic rule-induction techniques over the last few years (Michie, 1986).

It seems clear that this revival of interest in machine learning is overdue, is important, and will continue until systems capable of self-improvement become the norm rather than the exception. Indeed it is likely that many AI problems (e.g. speech understanding) are so difficult that they can only be solved by systems that go through a 'childlike' phase.

Putting it simply, learning is fundamental to intelligent behaviour – and that is a lesson that the AI community will have to learn if it is to achieve its objectives. Machine learning is far more than just a short cut in the 'knowledge acquisition' phase of constructing an expert system.

Furthermore, machine induction opens up the possibility of synthesizing totally new knowledge – of automating the process of scientific discovery and creating patterns and concepts that no one had ever thought of.

1.1 INDUCTIVE INFERENCE

Before discussing how, and in what sense, computers may be made to *learn*, we begin with a brief look at what philosophers and psychologists have said about the subject. We, too, may be able to learn from the past.

Induction – i.e. the derivation of general laws by examining particular instances – has long fascinated philosophers very much as a cobra fascinates its victims. It is both important and intractable, at least to a mind schooled in syllogistic reasoning: for no one has yet found a way to make induction truth-preserving, except in a closed system.

Clearly induction not only forms the basis for much of our day-to-day learning; it is at the foundation of the whole edifice of scientific discovery as well. Thus it is a cornerstone of human reasoning, but 'though science and indeed our daily life could not go forward without the inductive method, there has never been a proof of it' (Bronowski and Mazlish, 1960).

1.1.1 Inductive inference in philosophy

The problem is that inductively derived rules (whether created by persons or machines) cannot be proved correct. This age-old philosophical problem has broken the minds of the greatest thinkers since ancient times; and the doomed quest for a formal proof of the inductive method has hampered understanding of how it actually works. Only in the computer age have we come close to a systematic understanding of the act of induction. In short, the question of induction is a good example of how an intractable philosophical problem loses its sting (without actually being solved) when technological advance turns it into an engineering question.

A typical inductive inference goes something like this:

I have seen lots of black crows.
I have never seen a white crow.
Therefore, no crows are white.

Another old favourite is the sunrise 'problem':

Yesterday the sun rose in the East and set in the West.
Every day of my life it has risen in the East and set in the West.
Never in living memory has anyone seen it do anything else.
Therefore, it will rise in the East tomorrow too.

These innocuous acts of common-sense inference are in fact logically invalid, and philosophers have spent many sleepless nights attempting to find rational

grounds for validating them – not so much because they expect the sun to rise in the West tomorrow, but because they would like to put such conclusions on firmer footing. After all, albino crows are in fact white.

Let us look at what philosophers (chiefly Francis Bacon and John Stuart Mill) have said about induction. They have devoted their attention primarily to its role in the scientific method.

Unfortunately, from a software designer's viewpoint, they have been less interested in how to do it than in how to justify it. Their objective was to frame rules governing inductive argument just as logicians, from Aristotle to Boole, have framed rules for deductive argument. As J. S. Mill put it (in his *System of Logic*): 'what induction is, and what conditions render it legitimate, cannot but be deemed the main question of the science of logic'.

In this endeavour they have been unsuccessful. Nevertheless, the AI practitioner who is chiefly interested in how to mechanize the process of induction can glean a number of hints from the work of the philosophers.

Francis Bacon had a very modern view of the deficiencies of any purely deductive science. He was not content with the scholastic approach to knowledge that had held sway in Europe since before the Christian era, and which had been renovated and refurbished in his own day by Descartes in particular. In his *First Book of Aphorisms*, he expressed his discontent as follows: 'The sciences we now possess are merely systems for the nice ordering of things already invented: not methods of invention or directions for new works.' Also in the *First Book of Aphorisms*, he stressed the importance of negative evidence, and the tendency of the human mind to overlook it (see Hampshire, 1956):

> It is the peculiar and perpetual error of human intellect to be more excited by affirmatives than by negatives; whereas it ought properly to hold itself indifferently disposed towards both alike. Indeed in the establishment of any true axiom, the negative instance is the more forcible of the two.

Bacon also pointed out that an inductive leap, to be of value, must go beyond the observations on which it is based. When it does, and is subsequently confirmed empirically, our confidence in it is strengthened:

> But in establishing axioms by this kind of induction, we must also examine and try whether the axiom so established be frame to the measure of those particulars only from which it is derived, or whether it be larger and wider. And if it be larger and wider, we must observe whether by indicating to us new particulars it confirm that wideness and largeness as by a collateral security; that we may not either stick fast in things already known, or loosely grasp at shadows and abstract forms; not at things solid and realized in matter.

Users of modern-day inductive systems, or indeed of statistical classification programs, will recognize this as a warning about the tendency of such systems to underestimate the error rate when the rules come out of the lab and go into service in the field.

Bacon had great faith in the method of induction, and was surprisingly modern

(anticipating Babbage by two hundred years) in his belief that the work of the understanding could 'be done as if by machinery' (*Novum Organum*). Nevertheless he fully realized the danger of over-generalization. One of his cautionary remarks could serve as a slogan for the entire enterprise: 'the understanding must not therefore be supplied with wings, but rather hung with weights, to keep it from leaping and flying'.

Bacon was not in fact very influential in the progress of science, compared to Descartes or Newton for example, who popularized an axiomatic approach, and so there was little attempt to develop his systematization of induction for a long time after his death. It was not until John Stuart Mill laid down four primary 'experimental methods' for inducing general laws from particular cases that there was a significant advance on Baconian ideas about induction.

Mill considered that 'the business of inductive logic is to provide rules and models (such as the syllogism and its rules are for ratiocination) to which if inductive arguments conform those arguments are conclusive and not otherwise'; and he put forward four such rules and models himself. These were:

1. the method of agreement
2. the method of differences
3. the method of residues
4. the method of concomitant variation

These methods can be summarized as follows (see also Luce, 1958; Passmore, 1968):

1. *The method of agreement:* If two or more examples of a phenomenon under investigation have only one factor in common, the factor in which alone all the examples agree is the cause or effect of the given phenomenon.
2. *The method of differences:* If a positive instance of the phenomenon under investigation and a negative instance of the phenomenon have every circumstance in common except one, the single circumstance in which the two examples differ is the effect or the cause, or an indispensable part of the cause, of the phenomenon in question.
3. *The method of residues:* Remove from any phenomenon any part of it known to be the effect of certain antecedents and the remainder of the phenomenon is the effect of the remaining antecedents.
4. *The method of concomitant variation:* If one phenomenon varies regularly in some manner whenever another phenomenon varies in some particular way the first is connected with the second through some chain of causation.

Mill's four cardinal rules of induction suffer to some extent from antiquated terminology and from a preoccupation with causal determinism as the key to scientific investigation (which is no longer a fashionable view); nevertheless it is instructive when considering a computer-based learning program to ask oneself which of his four rules it uses. Normally one or more of his methods will be found at the heart of its induction strategy (see also Chapter 11).

Mill's first two methods (which are meant to be employed together) work best if there is no uncertainty in the causal chain which the scientist is attempting to explicate. The third method is best interpreted as heuristic advice to the scientific investigator. The fourth rule is the only one that makes much sense if one admits numerical data and uncertain information. It seeks correlation between two aspects or attributes of the phenomenon, and to that extent is the most general of the four rules, effectively subsuming the methods of agreement and differences. The method of concomitant variation involves looking for factors that vary together, or in inverse proportions; for example, the height and momentum of a weight when it is dropped to the ground, or the death-rate from cholera in a district and the distance from a particular well. Once a correlation between two quantities has been established, a law relating them can be proposed, and further tested (see also Chapter 7).

Mill's four methods, however, were specified before the computer age, and at first glance they appear to offer only very sketchy guidance for a program designer. When I first read them I thought they were merely interesting historical curiosities. However, when recast in a more modern format, they still serve as useful reference points for clarifying the whole business of induction by machine.

In Table 1.1 I have attempted to reformulate Mill's laws using more modern notations. One is a logical notation; the other is a probabilistic (Bayesian) notation. For example, the line

$$\text{not-C} \rightarrow \text{not-A} \Rightarrow \text{A} \rightarrow \text{C}$$

can be read as 'when the absence of C implies the absence of A there are grounds for believing that the presence of A implies the presence of C'. (Note that the inductive implication arrow $\Rightarrow$ is *not* a logically valid inference.) An alternative rendering of the same inductive implication is given on the line below in

Table 1.1 Mill's four laws (modernized)

1. *Method of agreement:*
$$\text{C} \rightarrow \text{A} \Rightarrow \text{A} \rightarrow \text{C}$$
$$\text{P(A}|\text{C)} = 1 \Rightarrow \text{P(C}|\text{A)} \gg 0$$
2. *Method of differences:*
$$\text{not-C} \rightarrow \text{not-A} \Rightarrow \text{A} \rightarrow \text{C}$$
$$\text{P(not-A}|\text{not-C)} = 1 \Rightarrow \text{P(C}|\text{A)} \gg 0$$
3. *Method of residues:*
$$\text{not(C} \rightarrow \text{not-A) OR not(not-C} \rightarrow \text{A)} \Rightarrow \text{A} \rightarrow \text{C}$$
$$\text{P(not-A}|\text{C)} = 0 \text{ OR P(A}|\text{not-C)} = 0 \Rightarrow \text{P(C}|\text{A)} \gg 0$$
4. *Method of concomitant variation:*
$$\text{A} = \text{f(C)} \Rightarrow \text{C} = \text{f(A)}$$

probabilistic terms as

$$P(\text{not-A}|\text{not-C}) = 1 \Rightarrow P(C|A) \gg 0$$

which states that 'when the probability of not-A given not-C is 1 (certainty) there a grounds for believing that the probability of C given A is very much greater than zero'.

Actually a probability of 1 is unheard of outside textbook examples, so it would be safer to replace

$$= 1 \quad \text{and} \quad = 0$$

by

$$> 1 - e \quad \text{and} \quad < e$$

where e is some small error tolerance (whose exact value would be dependent on the domain) in all the above examples. The spirit of the rules, however, would not be affected.

This translation is not rigidly faithful to J. S. Mill's original meaning, but it is close to his intention, and I believe is more useful in the present context. To make the rules intelligible, it is instructive to replace the letters A and C by words. I offer three choices:

Antecedent	Consequent
Attribute	Category
Alcoholic	Cirrhosis

The first pair of terms is closest to Mill's original formulation. Thus

$$C \rightarrow A \Rightarrow A \rightarrow C$$

becomes

> If the Consequent always follows the Antecedent then there are grounds for believing that the Antecendent causes the Consequent.

but it is rather bare and abstract. (It also suffers from Mill's preoccupation with causal linkage, which is absent from the purely symbolic rendering.) The second pair of terms fits in best with current terminology in the field of computerized induction:

> If examples of Category C (almost) always have Attribute A then propose a rule that Attribute A implies Category C.

The third pair gives a more memorable mental image, and also highlights the practical dangers of leaping to conclusions via induction:

> If (almost) all Cirrhosis patients have a history of Alcohol abuse then hypothesize that Alcohol abuse is a cause of Cirrhosis of the liver.

This kind of reasoning step is in fact how medical (and other) researchers think;

but – unlike machines – they have a background of common sense and expert knowledge against which the limitations of such a potential over-generalization can be tested. After all, there are other causative factors besides excessive alcohol consumption in the development of this particular liver disease.

(The above example illustrates the fact that many very useful inductively derived rules are not certain: they usually work, and we hedge them with phrases like 'in most cases', 'highly likely' and so on. Ideally, we would like our inductive computer programs to be able to operate in the same way.)

One thing that becomes apparent from the above recoding exercise is that Mill's first three laws are only relevant for nominal or categorical data, while the fourth applies principally to data measured on a numeric scale. Thus the first three laws form a group (which we might term **non-parametric methods**) distinct from the fourth law.

Another point worth noting is that the fourth of Mill's four rules is the most general: in effect it subsumes the other three, since they are all special kinds of concomitant variation. For that reason the fourth rule is the most important. And once you accept that covariation need not be perfect to be informative, you can say that all today's induction programs are based on a procedure for detecting concomitant variation. But while noticing concomitant variation is essential to machine (and human) learning, it is not the whole story.

Bertrand Russell was another philosopher who wrestled with the problem of justifying inductive reasoning and eventually admitted defeat. As he says in *The Problems of Philosophy*, the inductive principle is 'incapable of being proved by an appeal to experience'. Its role in human thought, however, is so fundamental that 'we must either accept the inductive principle on the grounds of its intrinsic evidence, or forgo all justification of our expectations about the future' (Russell, 1912). In other words, if you do not believe in induction, you cannot believe anything (see also Russell, 1961).

He did, however, add a concern with statistical reasoning which was largely absent from Mill's treatment of the topic. The inductive principle, as he saw it, was essentially probabilistic. In brief, when two things, such as thunder and lightning, have been found to go together many times and never found apart, then 'a sufficient number of cases of association will make the probability of a fresh association nearly a certainty, and will make it approach certainty without limit'. The more common the joint occurrence (with no exceptions) the more secure the inductive generalization. In practice, we extend this principle – the weight of numbers – to cases of (imperfect) statistical association. For instance, as more and more studies piled up in which cigarette smoking was associated with a raised incidence of lung cancer, but not with the certainty of the disease, more and more people became convinced that smoking was a cause of lung cancer; but even today the evidence is not conclusive, and a few die-hards remain to be persuaded of the causative link.

Wittgenstein's contribution to the philosophy of induction was to emphasize, or re-emphasize, simplicity. He asserted in the *Tractatus Logico-Philosophicus*,

paragraph 6.363, that 'the procedure of induction consists in accepting as true the simplest law that can be reconciled with our experiences' (Wittgenstein, 1961). Thus he recognized that many generalizations could be consistent with the evidence, and resurrected Occam's Razor as a means of choosing between them.

Over-elaborate explanations are difficult to disprove on the grounds of evidence alone, as the resilience of numerous superstitions testifies. Often such theories are only rejected because of internal contradictions. Indeed the history of science is littered with discredited hypotheses that were embellished with so many baroque flourishes that they collapsed under the weight of their own implausibility, even though they coincided with the known facts. A good case in point is Ptolemy's cosmology, which can be made to fit the facts of astronomical observation even today, by addition of a sufficient number of epicycles, but which was overturned by the Copernican theory because the latter had the virtue of simplicity.

Karl Popper (in Miller, 1987) is the living philosopher who has had most to say about the problem of induction, and his view, expressed trenchantly at the head of this chapter, is that the process does not exist:

> I hold with Hume that there simply is no such logical entity as an inductive inference; or, that all so-called inductive inferences are logically invalid – and even *inductively* invalid, to put it more sharply. We have many examples of deductively valid inferences, and even some partial criteria of deductive validity; but no example of an inductively valid inference exists (author's italics).

So what does go on in our heads when we encounter specific events and jump to general conclusions (such as reading the extracts cited in this chapter and deciding that philosophers live in a world of make-believe)? The answer, according to Popper (in Miller, 1987), is that all knowledge is, in the last analysis, guesswork:

> I hold that neither animals nor men use any procedure like induction, or any argument based on the repetition of instances. The belief that we use induction is simply a mistake. It is a kind of optical illusion. What we do use is a method of trial and elimination of error; however misleadingly this method may look like induction, its logical structure, it we examine it closely, totally differs from that of induction. Moreover, it . . . does not give rise to any of the difficulties connected with the problem of induction.

Although Popper's views appear nihilistic and downright perverse, he is actually closer to the spirit of the machine learning enterprise than any of the sages quoted earlier. His position is that knowledge is gained by making conjectures and refuting the ones that do not fit the facts, in science as in daily life. Therefore it is still rational to prefer, among competing hypotheses, those that have stood up to harsh criticism and multiple empirical trials. (This is how he smuggles an 'argument based on repetition of instances' back into his scientific methodology.) And it turns out that forming novel conjectures, most of which have to be weeded out after comparison with the data, is exactly what computers are good at – better, in certain circumstances, than people.

Where Popper can be criticized is: (1) in his blithe indifference to where hypotheses come from, i.e. the mechanics of conjecturing; and (2) in his one-sided view that all scientific hypotheses are universal laws.

The former criticism means that he has no practical help to offer in the actual building of an inductive engine; the latter means that he neglects statistical reasoning almost completely, due to an insistence that a single counter-example disproves a law (whereas no finite number of positive examples can prove a law). Not all scientific propositions, however, are universal. Some singular statements count as perfectly well-formed scientific hypotheses, for example: There are carbon-based life-forms on other planets; It is physiologically possible for a woman to run one mile in less than four minutes; Extra-terrestrial civilizations will arise before the end of the universe; Black holes exist; . . . and so on. Such assertions can be proved by a single instance; they cannot be disproved by accumulation of counter-examples. One could search all the planets circling all the stars in all the galaxies throughout the observable universe without fully disproving the conjecture that carbon-based life-forms exist elsewhere than earth. Yet one example would prove the conjecture correct. Indeed, the whole of AI is founded on a singular premiss: that it is possible to construct a thinking being without breeding it. So far all the evidence is contrary, but that has not disproved AI's fundamental principle, which could be vindicated (perhaps centuries in the future) by a single successful thinking machine. (This is not to deny that general laws are, broadly speaking, more useful than existential hypotheses.)

As a matter of fact the most useful kinds of hypotheses in real life are fuzzy rules of thumb, *already contradicted* by a small number of counter-examples, such as: falling out of aeroplanes at 8000 feet without a parachute is fatal. The statistical dimension is something that practical inductive programs cannot ignore even if the 'weight of numbers' has no grounding in pure logic.

Thus, although philosophy has not laid the blueprint for an inductive engine, it does provide some guidelines for people wishing to build one:

1. Use negative as well as positive evidence (Bacon);
2. Look for concomitant variation in the causal factors and the result (Mill);
3. The more frequently an association is observed, the more likely the association is to be generally true (Russell, *pace* Popper);
4. Prefer simple to complex generalizations (Wittgenstein).

This may seem no more than common sense, but that is a good sign; for induction is a common-sense process which philosophers seek to clarify and AI workers to mechanize.

1.1.2 Inductive learning in psychology

Induction in science is a public procedure. In daily life, however, induction goes on in private whenever we learn from experience. As such, it has been extensively studied by psychologists for over a century. So we might expect to find some useful

hints on the mechanisms that underlie human learning in the psychological research literature. If we did, however, we would be in for a disappointment.

Broadly speaking, psychological theories of learning fall into two groups, stimulus–response (S–R) theories and cognitive theories. The S–R theorists regard the organism as a black box. They are interested in relating inputs (stimuli) with outputs (responses), but do not claim to model what is going on inside the animal's brain.

For example, mathematically inclined S–R theorists have proposed a number of mathematical formulae that might account for the notorious 'learning curve', depicted in hundreds of introductory psychology textbooks, which rises (or falls if a habit is being extinguished) smoothly and exponentially towards an asymptote. The shape and slope of this curve depend on parameters in the equations describing the animals' response patterns. But, until recently, the S–R psychologists who tested and refined such equations were interested purely in fitting the data. They would have been horrified if anyone tried to give the parameters they used the status of mental constructs. This attitude – despite vigorous attacks on it from inside psychology (e.g. Koestler, 1964) – was only broken down when psychologists from both camps became familar with computers and realized that algorithms (i.e. rules of behaviour) were quite as rigorous, and therefore quite as respectable scientifically, as mathematical equations.

Cognitive theorists, on the other hand, do attempt to describe the mental structures which are (presumably) created and destroyed within the nervous system as an organism (especially a human) learns from experience. Interestingly enough, they resort to extensive borrowing from the field of computer science to describe what they think is going on inside the skull. They have no prejudice against mentalistic models, they just have not found any that work very well.

Forsyth and Rada (1986) sum the situation up as follows:

> S–R theorists speak of "learning" and experiment mainly with rats and pigeons. Cognitive psychologists usually talk of "memory" and do most of their experiments on human beings. At present cognitive theorizing is in the ascendant (partly at least because cognitive theories lend themselves to computer simulation) but the debate continues between adherents of the two approaches. It will be a very long time before a unified psychological theory of learning emerges that fits the multiplicity of experimental data concerning human and animal learning.

In the meantime the AI worker looking to psychology for ideas on how to build learning systems will be disappointed. To be blunt, the psychologists do not know how it is done. For many years behaviourism was the prevailing orthodoxy in experimental psychology, and behaviourists eschewed mentalist concepts. The idea that a rat is forming and testing hypotheses when it runs a maze or that a pigeon is amending its own rule base as it pecks for food was a heresy for behavioural scientists from about 1920 till the late 1960s. It is scarcely surprising, therefore, that psychologists cannot explain something they have only recently admitted to exist.

Essentially the S–R theorists do not want to explain how learning actually works, and the cognitive theorists are unable to do so. As Holland *et al.* (1986) put it: 'Induction, which had been called the "scandal of philosophy", had become the scandal of psychology and artificial intelligence as well.' Part of the explanation for this dismal state of affairs lies in the disparity between neurons and notions. The modern psychologist who trains a rat to run a maze believes that it has developed some sort of **cognitive map** of the maze. The physiologist who cuts the creature up afterwards knows a good deal about how its nervous system works. But nobody yet knows how to reconcile these two levels of description. How is the software (the cognitive map) implemented in terms of the hardware (the neural interconnections)? The question has yet to be answered.

(There is an interesting parallel here, which there is insufficient space to explore as it deserves, between the SR vs cognitive opposition in psychology and the Connectionist vs symbolic approach to machine learning in AI. Very roughly, Connectionist systems (Hinton and Anderson, 1981) aim to model the brain, while symbolic systems – the mainstream of AI – aim to model the mind. As in psychology, so in AI: no one has yet unified the two approaches. It may be that the two types of description are incompatible. See also Chapters 11 and 12.)

Perhaps the only conclusion that would command a near-universal consensus among psychological researchers and at the same time make sense from the system-designer's point of view is that feedback, or knowledge of results, is absolutely crucial to the acquisition of novel skills, and must be provided as promptly as possible (see Hilgard and Bower, 1966; Bolles, 1979).

1.2 THE ACT OF INDUCTION

If learning is so poorly understood by philosophers and psychologists, how can upstart AI programmers hope to computerize the process? After all, they cannot even give a watertight definition of learning. The answer is that AI researchers are posing themselves a rather different problem. The philosphers wanted to know 'how can induction be rigorously validated?'. The psychologists want to know 'how does the brain store the results of experience?'. The AI programmer has a more modest goal, which involves answering the question 'in what ways can a machine develop general rules from specific examples, and how reliable are those rules in practice?' As the history of science makes abundantly clear, it is not answering but asking the right questions which is the key to progress.

Indeed it already appears that the results coming in from AI workers on induction by computer will throw new light on the unanswered questions of philosophy and psychology.

For the rest of this book we will say that learning is a phenomenon exhibited when a system improves its performance at a given task without being reprogrammed. Improvement can mean various things, including: a higher proportion of correct decisions, faster response, lower-cost solutions and wider range of applicability. We use the term **induction** (which means reasoning from specific cases to general principles) to describe the method by which learning is

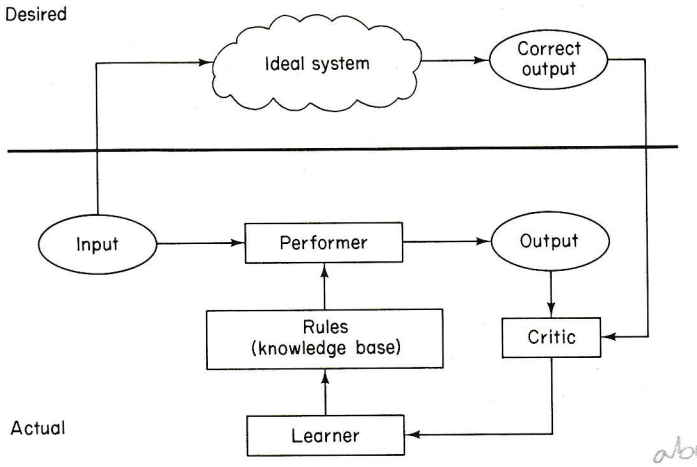

Fig. 1.1. A framework for learning. According to this diagram, learning is 99% pattern recognition and 1% reasoning.

effected. **Rule induction** covers a special, and prevalent, case of induction, in which the results of induction are expressible as condition–action rules.

It is important to note that, from this standpoint, a machine cannot learn anything unless its performance can be assessed objectively. For learning to take place, there must be a criterion according to which decisions or partial solutions can be scored. Otherwise no one can say whether the system has changed for the better or for the worse. Agreeing on a suitable measure, however, may not be a simple matter, as we shall see.

1.2.1 A framework for induction

Looked at from sufficiently far away, all systems designed to modify and improve their performance share certain important common features. Figure 1.1 is a diagram of the four major components of a typical learning system. Essentially this sketches a pattern recognizer which learns to associate input descriptions with output categories; but, as we shall see, many systems that are not overtly concerned with pattern recognition also fit into this general framework (Forsyth, 1984). Note that the system contains a **feedback** loop. We can briefly describe its main components in turn, by going round this feedback loop.

The **Critic** compares the actual with the desired output. In order to do so, there must be an 'ideal system', as we call it, against which the system's behaviour is measured. In practice this may be a human expert, or teacher. For instance, if the task is medical diagnosis, the ideal system may be the diagnosis given by a top consultant when faced with the patient whose history is being presented to the

computer as input. The job of the critic is to apportion credit and/or blame for the system's responses. It must assess deviations from correct performance.

This can be simple or complex. In a simple case the Critic might compare (for example) rainfall as forecast with actual rainfall. If 0.6 mm of rain fell and the system predicted 1.7 mm then it is only a matter of subtracting one number from another and passing the difference on as feedback to the learning module. In other circumstances there may be more work for the critic to do to ascertain what went wrong. For example, after losing a game of chess, it may not be at all obvious where the computer made its mistakes. But however simple or complex the task of the critic, evaluative feedback is absolutely fundamental to the learning process. There are 'unsupervised learning' programs in existence, which do roughly the job that statisticians know as **cluster analysis**, i.e. tidying up a cluttered conceptual space, but this facility, though a component of learning, is not learning in the sense of improving performance at an assigned task. What such programs do is a precursor to the learning process proper, but it does not count as learning until the revised conceptual vocabulary is actually put to use.

To continue with the diagram of Fig. 1.1: the **Learner** is the heart of the system. This is the part of the system which has responsibility for amending the knowledge base to correct erroneous performance. A large number of learning strategies have been proposed since work on machine learning got under way in the 1950s, some of which will be examined in subsequent chapters.

The **Rules** are the data structures that encode the system's current level of expertise. They guide the activity of the performance module. The crucial point is that they can be amended. Instead of a read-only knowledge base (as in most current expert systems) the rules constitute a programmable–erasable knowledge base. Obviously they must only be modified under strictly defined conditions or chaos will result. Other forms of knowledge representation than condition–action rules have been used successfully, but we use the term 'rules' as a convenient shorthand.

Finally, the **Performer** is the part of the system that carries out the task. This uses the rules in some way to guide its actions. In other words, it is a kind of interpreter. Thus when the rules are updated, the behaviour of the system as a whole changes (for the better, if all goes according to plan). Naturally the Performer is the part of the system which varies most from one task to another, and is therefore the part of the system about which least can be said in general terms.

In discussions of machine learning systems, most attention is typically focused on the Learner and the Performer modules. But it is worth pointing out that the two commonest reasons for failure of machine learning systems stem from inadequacies in the other two functions, the Critic and the Rules:

1. Failure of the Critic: the chosen evaluation measure is misleading;
2. Deficiency in the Rules: the chosen representation is not capable of expressing the knowledge needed for satisfactory performance.

1.2.2 A taxonomy for learning

Two other terms need to be defined before our examination of practical learning methods – description language and training set.

The **description language** (or **rule language**) is the notation or formalism in which the knowledge of the system is expressed. There are two kinds of description language which are important. The first is the formalism used to represent the input examples. The second kind of description language is that chosen to represent the rules themselves. It should be noted that the expressiveness of the description language in which the rules are formulated is crucial to the success of any learning algorithm. It also has a bearing on how readily the knowledge can be understood, and hence on whether it can be transferred to people.

The notion of a **training set** is important in understanding how a machine learning system is tested. Typically there is a database of examples for which the solutions are known. The system works through these instances and develops a rule or set of rules for associating input descriptions with output decisions (e.g. disease symptoms with diagnoses). But, as Bacon pointed out, rules must be tested on cases other than those from which they were derived. Therefore there should be another database, the **test set**, of the same kind but containing unseen data. If the rules also apply successfully to these fresh cases, our confidence in them is increased (this issue is considered again in Chapter 2).

This preliminary definition of terms enables us to compare learning systems in a consistent manner.

Machine learning can be viewed from a bewildering number of perspectives – as optimization, concept formation, pattern recognition, automatic classification, scientific discovery, programming by example and many more. But the simple common theme of generate + test (which is the dynamic underlying the static block diagram of Fig. 1.1) provides a unifying principle. All learning systems, however clever they are about avoiding brute-force exhaustive search, propose new potential solutions and test those potential solutions (see Section 1.1.1 on Karl Popper). This means that there are two fundamental questions we can ask about an automatic induction system:

1. How are new knowledge structures generated from old ones?
2. How are candidate knowledge structures evaluated?

There are many possible taxonomies of the field of rule induction; but the answers to these two simple questions help to organize an area that up till now has been characterized by somewhat anarchic eclecticism. If we know the answers to those two questions for a particular system, we have grasped the essence of its operation.

A more detailed classification scheme divides up learning systems into 256 types based on the answers to eight two-way questions. Strictly speaking, these dichotomies are mostly polarities, i.e. a real system will often fall between the two

extremes, but lumping all systems to one side or the other of an imaginary midpoint simplifies the scheme and is useful for two reasons: (1) it provides the reader with a standard checklist of questions to ask (attributes to look for) in attempting to understand a novel learning system; (2) it relates learning systems together, and may even suggest new meta-rules or PhD projects (e.g. Why are there no existing systems with such and such a combination of attributes?)

Domain:

1. General purpose vs specific
   Does the system apply to a variety of application areas or is it specialized to one field?
   General-purpose example: ID3
   Specific example: Meta-dendral

Induction method:

2. Incremental vs one-shot learning
   Does the system maintain a best-so-far rule or description which is amended as new examples arrive one by one or does it look at the entire training set before forming its rule(s)?
   Incremental example: UNIMEM
   Single-shot example: CN2
3. Subsumption-based vs non-hierarchic
   Does the induction process rely on an inheritance lattice that orders concepts from general to specific, or does it work without reference to the generality of the descriptions it generates?
   Subsumption-based example: LEX
   Non-hierarchic example: Holland's genetic algorithm
4. Deterministic vs non-deterministic
   Does the system always give the same results from the same data or is there a random element in its rule generation?
   Deterministic example: BACON.4
   Randomized example: BEAGLE

Critic:

5. Logical evaluation vs quantitative evaluation
   Does the system classify trials only as successes or failures, or does it measure the distance from the correct answer according to a metric?
   Logical example: Perceptron
   Quantitative example: EURISKO

Representation:

6. Unary features vs structural predicates
   Are examples presented to the system as feature vectors, or can the system accept examples which have internal structure described by multi-term predicates?
   Unary example: Assistant
   Structural example: Winston's program

7. Humanly intelligible vs black box
   Is the rule language readable by people or is it in an opaque internal code?
   Intelligible example: Induce 1.2
   Black box example: Boltzmann machine
8. Fixed language vs extensible language
   Is the system restricted to a description language given by its designer or can it extend its own vocabulary by defining new concepts and functions? (This is very much a matter of degree.)
   Fixed language example: AQ11
   Extensible language example: CYRANO

Do not worry if you have not (yet) come across most of the systems cited above as examples. They merely serve as a preliminary demonstration of the feasibility of the eight attributes, by showing at least one example from both ends of the spectrum in each case. Many of them will be discussed in subsequent chapters.

This list of attributes is not presented as the ultimate classification scheme for machine learning, but is offered as an aid for the reader who is new to the subject. It should help to impose order on the variety of systems described in the remainder of this book. We can put it to the test, and at the same time introduce one final important concept (the **search space**) by applying it to a well-known induction system.

1.3 IN SEARCH OF KNOWLEDGE

It is a fundamental notion of machine learning research that the process of induction can be viewed as a search through an abstract space of potential rules or descriptions. Tom Mitchell (1982) was the first to make this important point in print.

A rule language defines a vast (possibly infinite) set of potential rules or concepts. The job of a learning algorithm is to hunt through that enormous space for useful descriptions in an efficient manner. In nearly all realistic situations, the proportion of valuable descriptions to syntactically valid descriptions is extremely low: at needle-in-haystack level or worse. The main problem (assuming that the rule language is capable of describing any correct rules at all) is how to ignore the great majority of useless descriptions without missing the useful one(s).

Mitchell's insight was that the voluminous AI literature on search as a problem-solving technique could be harnessed in the quest for efficient learning algorithms. Before then, the two topics of learning and search had been treated as separate. So it is appropriate to round off this introductory chapter by describing, as our first concrete example, the algorithm that Mitchell devised for searching a space of possible concepts (which he called the **version space**). The method is known as the **candidate-elimination** algorithm.

For computer programmers it can be described as a cross between the Sieve of Eratosthenes and the Binary chop. The sieve of Eratosthenes is a way of finding

prime numbers by setting up an array of odd integers and eliminating those which are divisible by numbers lower down in the array. The binary chop, or binary search, is a technique for converging on a solution, or an approximation to a solution, by repeatedly readjusting inwards the two extreme points between which the solution is known to lie. Both are well-known programming techniques.

The basic idea behind the candidate-elimination algorithm appears absurd at first sight: list all possible descriptions, then cross off the ones that do not apply to the training data. Any that are left after the crossing-out process must be correct descriptions. This procedure constitutes an elegant method (at least with noise-free training instances) of converging on the description or descriptions of a concept (Mitchell, 1977, 1982).

What could be simpler than taking all possible descriptions and eliminating the ones that conflict with the training set? The catch is, of course, that the set of all possible descriptions is (except in trivial cases) far too large to enumerate individually. But despite the enormity of the search space, the method becomes practical by taking account of the fact that a partial ordering exists among the concept descriptions, from general to specific. Thus any two descriptions can be compared and ranked according to which is the more general. For instance, 'deaf white cat' is more specific than 'white cat'. (This remains true in principle even if, empirically, all white cats happen to be deaf.)

This realization allows the system to maintain two boundary sets – S, the set of the most specific possible descriptions compatible with the training data, and G, the set of the most general possible descriptions still compatible with the training data. Between them, these two finite sets define the edges of a far larger set in a compact form.

The S and G sets are gradually made to converge as more and more training instances are processed. The convergence is achieved as follows:

1. When a positive instance is encountered, any description in G that does not cover it is eliminated, and all elements of S are generalized as little as possible so that they do cover it;
2. When a negative instance is encountered, any description in S that covers it is deleted, and all elements of G are specialized as little as possible so that they no longer cover it.

The S set can be initialized with the description of a 'seed' training example (which is a positive instance of the concept to be discovered) or alternatively with all possible maximally specific descriptions, and the G set is initialized with the null description. (Thus the G set starts by stating that all examples are positive examples.) In effect the S set describes sufficient conditions for belonging to the concept being learned and the G set describes necessary conditions. When the two sets converge, the concept is fully defined. An illustration of this procedure appears as Fig. 1.2.

Mitchell's masterly exposition convinced some members of the AI community

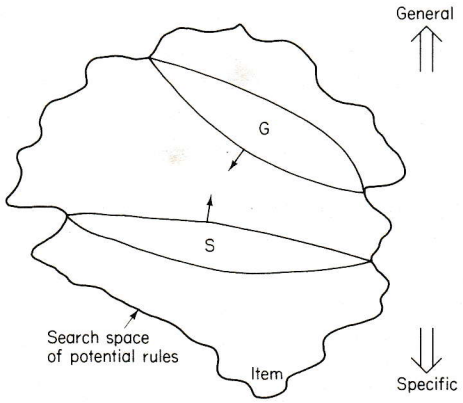

Fig. 1.2. Version space. Mitchell's algorithm is a kind of cross between the sieve of Eratosthenes and the binary chop.

that the induction problem had, in all essentials, been solved. However, like many theoretically optimal procedures, the candidate-elimination algorithm is very brittle in practice. It goes wrong with quite modest amounts of noise in the training data. Thus it is chiefly suited to domains, such as symbolic integration or chess problems, where the examples are themselves generated by formal rules; it is not suited to domains, such as weather forecasting or disease diagnosis, where the data is subject to random perturbations or observational errors. (Mitchell has extended the algorithm to cope with very small amounts of noise; but these remarks still apply.)

Thus Mitchell's general idea of learning as a search has survived and proved fruitful, though his particular search algorithm is merely one in a growing catalogue of induction techniques. We can describe it in terms of the attribute list of the previous section as follows:

1. Is it general purpose or specific? General purpose in so far as it is not tied to one subject-matter (though in practical terms it is limited to well-defined problem areas);
2. Is it incremental or single-shot? Incremental because the training examples are examined one at a time;
3. Is it subsumption-based or not? It is the archetypal subsumption-based system, since it relies entirely on the general-to-specific ordering of concepts;
4. Is it deterministic or non-deterministic? It is deterministic: given the same examples in the same order it will always produce the same results;
5. Does it use logical or quantitative evaluation? It uses logical evaluation: descriptions are either correct or not; there is no measure of near misses;

6. Does it use unary or multiple predicates? It uses unary predicates, although later work has extended it to deal with structural predicates;
7. Are its descriptions humanly intelligible or opaque? They are intended to be legible by people (although the basic method could work with other representations);
8. Does it use a fixed or extensible rule language? The language is fixed. (If it created new descriptors it would have to insert them in the generality lattice and start the G/S convergence process all over again.)

Thus the answers to all eight questions are Yes for the candidate-elimination algorithm. It defines an end-point of (11111111) in the induction-system typology outlined in the previous section. (Since you ask, I cannot think of a 00000000 system to contrast it with, though I have been involved personally with the creation of a financial rule generator that has not been reported but which would qualify as a 00000001 type.)

This chapter has now sketched a background against which to view the field of machine learning, from philosophical discussions to psychological theories of learning. It has outlined a conceptual framework within which modern efforts to computerize the act of induction can be understood, and it has introduced some important terminology. We can now proceed to explore some of the many aspects of machine learning more fully in the remaining chapters of the book.

1.4 REFERENCES

Bolles, R. (1979) *Learning Theory*, Holt, Rinehart and Winston, New York.

Bronowski, J. and Mazlish, B. (1960) *The Western Intellectual Tradition*, Hutchinson, London.

Forsyth, R. (ed.) (1984) *Expert Systems: Principles and Case Studies*, Chapman and Hall, London.

Forsyth, R. and Rada, R. (1986) *Machine Learning: Applications in Expert Systems and Information Retrieval*, Ellis Horwood, Chichester.

Hampshire, S. (ed.) (1956) *The Age of Reason*, Mentor Books, New York.

Hilgard, E. R. and Bower, G. H. (1966) *Theories of Learning*, Appleton-Century-Crofts, New York.

Hinton, G. E. and Anderson, J. (eds) (1981) *Parallel Models of Associative Memory*, Lawrence Erlbaum, Hillsdale, N.J.

Holland, J. H., Holyoak, K. J., Nisbett, R. E. and Thagard, P. R. (1986) *Induction: Processes of Inference, Learning and Discovery*, MIT Press, Cambridge, Mass.

Koestler, A. (1964) *The Act of Creation*, Hutchinson, London.

Luce, A. A. (1958) *Teach Yourself Logic*, English Universities Press, London.

Michie, D. (1986) *On Machine Intelligence*, Ellis Horwood, Chichester.

Miller, D. (ed.) (1987) *A Pocket Popper*, Fontana Press, Glasgow.

Mitchell, T. (1977) Version spaces: a candidate elimination approach to rule induction. *Internat. Joint Conf. on AI*, **5**, 305–10.

Mitchell, T. (1982) Generalization as search. *Artificial Intelligence*, **18**, 203–26.

Passmore, J. (1968) *A Hundred Years of Philosophy*, Penguin Books, Harmondsworth, Middx.

Popper, K. (1972) *Conjectures and Refutations* (4th edn), Routledge and Kegan Paul, London.

Russell, B. (1912) *The Problems of Philosophy*, Oxford University Press, Oxford.

Russell, B. (1961) *History of Western Philosophy*, Allen and Unwin, London.

Wittgenstein, L. (1961) *Tractatus Logico-Philosophicus*, tr. by Pears and McGuinness, Routledge and Kegan Paul, London.