Chapter 2

Computational approaches to induction

2.1 Introduction

This chapter brings together all of the technical apparatus that we will need over the chapters that will follow. It focuses on the notion of *probability*, and the application of probability to *information* and *algorithmic complexity*. From the perspective of researchers working in this area, the chapter is relatively informal, and the reader who knows all of the material here is likely to notice places where we have glossed over technical niceties. On the other hand, the reader for whom this chapter is written, and who does not know the material already, may find the chapter quite technical and perhaps challenging. We have tried to steer a middle course between the expectations of these different sorts of readers.

The models that we look at here are in some respects much simpler—indeed, impoverished—with respect to some of the assumptions made in contemporary linguistics. But there is an impoverished side to contemporary linguistic models as well that is captured more richly in these models, involving principally the elaboration of distributions that are not uniform. We will explain that a bit more clearly at the end of the following section.

2.2 Probability

2.2.1 Probability distributions

The most important theme in the development of modern empiricism is the meaning and use of probability, and its essential role in understanding the nature of knowledge and learning. Probability plays a role in all of the themes that we deal with in this book, and in this section we give an introduction to some of the formal conditions and properties of a probabilistic analysis.

Let us begin by asking the reader to put to one side his everyday notion of probability, or the one based on frequencies and the rolling of dice: it is true that the probability of rolling a two is one in six each time we roll a fair die, but we will not get there for a while yet. We begin, rather, with a simpler, more basic, and more mathematical perspective on what probability is.

We will say that we have a probabilistic account of some domain when we can assign a distribution over its members: a distribution is an assignment of a nonnegative number to each member, subject to the condition that all of these nonnegative numbers add up to exactly 1.0 (though see the footnote for some brief remarks about probabilities over continuous domains.[1]) The set over which the distribution is defined is called the *sample space*. We may as well use some mathematical notation: a distribution is a function that maps from a particular domain D, which is the sample space, to the real numbers in the closed interval [0,1]. In the vast majority of cases that interest us, the domain D is infinite: it might consist, for example, of all possible strings of English words, and there is no upper limit to the number of such strings. It is important to be clear on the following fact: it is not difficult to establish a function that assigns not simply a nonnegative but in fact a positive number to an infinite set of elements and still have it sum to exactly 1.0. That is, it is easy to display functions where $\sum_{x \in D} p(x) = 1.0$. If we can assign a strict ordering to the set D (we call that an *enumeration* of D), then it is very easy to give an example of such a distribution. One such example—but by no means the only such example—is the following: to the i^{th} element, we assign the probability 2^{-i}. The "amount" of probability assigned to elements gets very small very fast as i increases, but it always remains positive, and it sums to exactly 1.0 in the limit.[2]

We have thus begun with a very abstract characterization of probability. We began this way in part to counter the intuitions which might otherwise arise: for instance, that probabilities are just a refined way of talking about the frequency with which events occur in the world, like the frequency with which a

[1] This is a simplification of the real mathematical situation, but the simplification should not matter except to the reader who already understands it. In more general cases, where the domain that we are looking at contains not simply discrete elements but variables taking on real values (that is, any value in some real interval), then we draw a distinction between the underlying domain of outcomes and various sets of those outcomes, called events, and in this case, events roughly consist of *open intervals* of real numbers, and no sets of real numbers that cannot be built up out of such intervals. The outcomes are not assigned probabilities as such, but the events are. In addition, we draw a distinction between the function that assigns probability to the outcomes—which is a probability mass function—and the function that assigns probability to events, which is the real probability.

[2] We caution the reader that there is no need to worry about whether the sum ever gets to 1.0 or not: this worry has been settled by the mathematics that lies behind saying that the sum is taken in the limit; there is no other meaning that would attach to an infinite sum, for what that's worth.

die comes up two. We do not adopt this *frequentist* view of probability in this book. We take instead a *Bayesian* point of view, in which part of the reason we compute probability distributions is in order to be able to state explicitly what the probability is that a parameter we care about falls within a certain interval, given what evidence we have seen so far. A frequentist approach to modeling some English text assumes that the grammar generating these texts has parameters which take on certain values that we are trying to discover: the parameters are in the reality, so to speak, and a mastery of statistical methods will allow us to judge what the odds are that the true value is within a close window to the value that we have inferred from the data. In a Bayesian approach to modeling, the same text is a succession of refinements of distributions which express what we can rightfully infer the value of the parameter (or parameters) to be. This is sometimes referred to as a *subjective* perspective, and that is fair enough, especially if that is understood as a reminder that we are by no means obliged to think that the parameters we compute would correspond to a value that the Omnipotent has set (or can look at). It is not subjective in the sense of being a matter of opinion or personal preference, however.

A crucial implication of this view is that it is not meaningful to ask what the probability of an event e (or an outcome) is in any absolute sense: it only makes sense to ask that question, given a particular distribution over the domain in which the event e occurs. If the distribution is called D, then we may meaningfully write $p_D(e)$.

The fundamental goal of a probabilistic approach to modeling reality is to construct a model in which probabilities are assigned to interesting phenomena in ways that match up with reality. By recognizing that what we build is a *model*, we acknowledge that there is an element of simplification at work; we hope that the degree of simplification will not be fatal. We recognize that we need to clarify exactly how the quantitative aspects of our model "match up" with reality: that phrase is a little too vague to be left as it stands. And finally, we add this: because we know that the subject of our study is language, we know that we will be considering many sequences of word choices. Since we know that the vocabulary of a language is typically on the order of 10^5 or more, we know that there are many possible sequences of words of length n: there are on the order of 10^{5n}, which is a seriously large number. Our goal is to figure out how we can construct quantitative—here, probabilistic—models that remain tractable even when we allow them to generate long strings of words. We must do this by building up the larger model out of smaller parts that can be more simply described. So we will first develop some simple tools to describe choices from a finite set and then build up to models that allow sequences of such choices.

By way of contrast, non-probabilistic models—generative models in linguistics, for example—steer clear of such straightforward questions as what the probability is of choosing the word *dog* as the expansion of a terminal category *noun* in English. Ask a syntactician who is offering an analysis of a grammatical English sentence why the word chosen in expanding a given noun node was *dog*, say, and you will be told that the choice was just a *for instance*, that we could have chosen a different word—it does not matter what word we chose, the point is still the same. The probabilist wants to reply: yes *of course* it matters what assumptions we make about choice of noun (or of any other category), and if we chose to assign a uniform distribution over all nouns in the lexicon, we can certainly do that (no one is stopping us from doing so), but we will provide a less enlightening analysis of English if we use a uniform distribution rather than one that reflects what people actually say.

2.2.2 Conditional probability and Bayes' rule

We sometimes think of a distribution as assigning a nearly tangible substance called probability mass over its domain. There is a total amount of probability mass equal to 1.0 units—we might imagine that the units are kilograms. This amount is divided up and distributed over the domain of the distribution.

A conditional probability is what we get if we focus our attention on just one subpart S of the domain of the distribution and ask about the probability of an event e that is in S on the condition that all we care about is events inside of S. If we have a probability distribution over all the words of English, then the probability assigned to the word *dog* might be 0.000631712 (as it is in the Brown corpus), while the conditional probability of the word *dog*, conditioned by the word being a singular noun, might be 0.001329677. If we know that a word is a singular noun, what is the probability that it is *dog*? In its more central form, the notion of conditional probability is based on the idea that we consider not the entire universe of possible outcomes but only some subset—and we call that subset "what is given." Formally speaking, the conditional probability of A given B is the probability that both A and B hold, divided by the total probability mass assigned to the condition B. Stated using more general symbols, we define conditional probability in this way:

$$p(A|B) \equiv \frac{p(A\&B)}{p(B)} \tag{2.1}$$

This is just a definition, but it is all we need in order to show what Bayes' rule is and where it comes from. The definition of $p(A|B)$ above immediately leads

to the following statement:

$$p(A|B)p(B) = p(A\&B) \tag{2.2}$$

and the very same definition tells us that

$$p(B|A)p(A) = p(B\&A) \tag{2.3}$$

But since $p(A\&B)$ is the same as $p(B\&A)$— it is the probability of the event of both A and B occurring—it follows that

$$p(B|A)p(A) = p(A|B)p(B) \tag{2.4}$$

and hence that

$$p(B|A) = \frac{p(A|B)p(B)}{p(A)} \tag{2.5}$$

This is Bayes' rule (sometimes called Bayes' law or Bayes' theorem), and as one can see, it is nothing more than a simple algebraic manipulation of the definition of conditional probability. Its function is to give an explicit account of how we reverse the conditioning of two events.

2.2.3 Sequences of random variables

Because the study of language is the study of large numbers of strings of words, we need to employ the appropriate mathematical tool for the job at hand. Here, the right tool to use is a sequence of random variables. The term *random variable* is a bit misleading, though it fits with the intuition that is often offered for it: a random variable can typically be given a human meaning as a measurement of something happening. For example, we note the number of letters in successive words, or the number of births in successive years, or in each of the countries around the world—or simply the choice made of the word in successive positions of a sentence.

Nobody denies that are there are dependencies between the words in a sentence, and describing the sequence of words in this way does not commit us to the idea that these dependencies do not exist. Random does not mean that there is no order or no relationship between the words, and much of what follows are ways of fleshing out more or less precisely what sorts of dependencies there are. Using sequences of random variables allows us to talk about not just the probability of a *word* but the probability of a *sentence* (or sequence of sentences) and thus to talk about grammar, but using a probabilistic vocabulary rather than the categorical vocabulary of sets.

2.2.4 Finite-state automata

We will explain first what has come to be known as a *Moore* machine, a particular type of finite-state automaton. We imagine that there is a finite *set* of different states, each capable of outputing various words: each state knows about itself the probability with which it will generate any particular word, and each state knows what the probability is that it will transition to any of the other states at the following moment of time. In symbols, we say that for each state i, there is a probability $\tau_i(w)$ that it will emit word w, and these sum to 1: $\sum_w \tau_i(w) = 1$; and the transition probabilities from each state must sum to 1 as well: for each i, $\sum_j s_{i,j} = 1$.

Graphically, there are two ways to think of the evolution in time of such a system. The first is as a path through a graph, moving around through the permitted paths indicated by edges between the states. See Figure 2.1. Associated with each step through the graph is a word that is generated by the system when it is in (or at, if you prefer) a particular state.

The second way of thinking about this is by imagining that all of the states are stacked up on top of one another, and there are as many copies of this stack as there are words emitted by the system. Then a path through the system is one that moves from left to right, with only one state chosen per moment of time; as before, a word is emitted by each state that is passed through (see Figure 2.2).

These models are generally described as systems which evolve only with knowledge of the previous state that they were in. But the way we have set

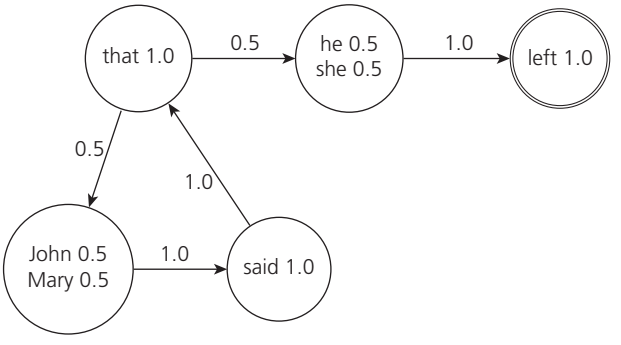

Fig. 2.1 A graph representing a finite-state automaton. The process starts at the bottom left state and ends at the top right state. This will generate, for example, the strings "John said that he left" and "John said that Mary said that she left," as well as infinitely many others. In order to compute the probability of each string, we multiply the probabilities of each transition, and each output. Thus "John said that he left" has the probability $0.5 \times 1 \times 1 \times 1 \times 1 \times 0.5 \times 0.5 \times 1 \times 1$, which is 0.125.

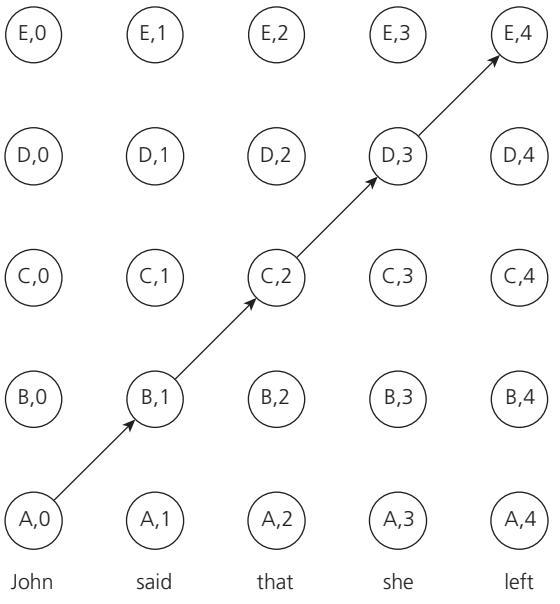

Fig. 2.2 A diagram where each moment in time is represented by a separate vertical stack of states. We show a single path that outputs one sentence.

things up graphically, when it is emitting a word, it really only knows the state it is in at that moment; the state that it is in determines, by definition, what the probability is that it will emit any particular word. Nonetheless, it is reasonable to say that the system as a whole knows what the immediately preceding state was, because the determination of what state the system is in at time t depends solely and entirely on the state that the system was in at the immediately preceding time $t - 1$.

Now, with a Moore machine of the sort we have described, it is not unreasonable to think of the system, when it is in a particular state, as thinking of the words it can generate as being of the same part of speech in some sense or other. But we should be clear that there is nothing at all wrong with having a Moore machine which has several states that generate the same words (with similar or different probabilities). This may be because a word is ambiguous in its category (*content* or *can*), or it may be for any other reason: the probability that is assigned to a noun determiner (such as *the*, *his*, or *a*) might well be very different depending on whether the determiner appears in the sentence initially or not, to take just one simple example.

The easiest mistake to make at this point is to think that one really knows what *being in state i* at a particular moment means. We could imagine an extreme case where there are exactly as many states as there are words in the vocabulary and that each state generated one and only word—its

particular word. In that case, generating a certain word is equivalent to actually *being* in a particular state, and so it would not be terribly wrong to say of such a system that it models the production of a word as being conditioned on, and only on, the immediately preceding word. But that is an entirely artificial example. We typically consider models with a far smaller number of states than there are words in the lexicon.

In some such cases, it is reasonable to think of the states as roughly, but only roughly, corresponding to parts of speech. A system like that is illustrated in Figure 2.3.

But that need not be the case. A state of such a model could correspond to a more abstract point in a derivation (or partially constructed tree, as a linguist would view it). For example, the initial state of a model might transition to some state which can generate not just a single word, but a phrase, and the generation process of this phrase might consist of generating a sequence of words through some states, specialized for this particular task. This mild extension of the formalism leads to what were called Recursive Transition Networks [Woods, 1970]; now, they have been assimilated into the theory of probabilistic phrase structure grammars, which is the approach we explore below. We call this a first-order Markov model, which is to say, a model whose properties depend only on what state the system was in at the previous moment. Second-order models are those where the probabilities can depend not just on the immediately preceding state but also on the previous one as well.

We call such a system *deterministic*, since knowledge of the output allows us to infer what states the system passed through. We can also say that the system is a first-order Markov model, in the sense that it only needs to look back one state to be fully described. You may object to that statement and say that if the system has selected state S_4, for example, then it has no further need to know what state preceded. That is true, but breaking up the options a system has into states is just a way of talking about how the system is conditioned by its past.

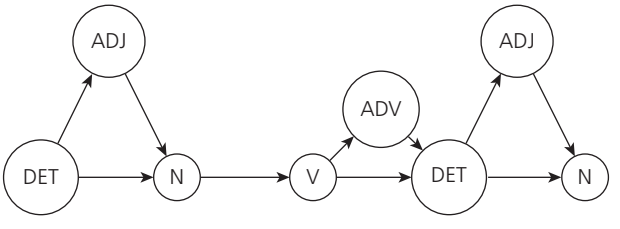

Fig. 2.3 A diagram showing states that correspond to parts of speech. We omit the transition probabilities and the outputs. The state labelled DET outputs determiners, ADJ adjectives, and so on.

These simple *finite state* models are too weak to capture the sorts of dependencies that we see in natural languages; indeed, these weaknesses were, historically, one of the primary motivations for generative grammar. In Section 2.8 we will look at how these models can be enriched to formalisms which seem to be sufficiently powerful to describe natural language syntax.

2.3 The probability of the data

Probabilistic grammars open up an entirely new way to think about the connection between the predictions made by a grammar and the data supplied by reality. This new way is to seek the grammar that maximizes the probability of the data. In this regard, it is an *alternative* to the view that the goal of a model of language is to generate all grammatical sentences and *not* to generate the ungrammatical sentences. This point is very much worth emphasizing, because that understanding of what a grammar is intended to accomplish has become so well established in the field that it may be hard at first to imagine that anyone might disagree with it.

In some respects, the idea that a grammar should generate all and only the sentences of a language is an idea strongly associated with generative grammar, and hence with the work of Noam Chomsky. It certainly is true that this conception of formal grammar emerged clearly only in the 1950s, but it was emerging before generative grammar; see, for example, Harwood [1955], for a clear statement on evaluating a grammar on the basis of what it does and does not generate. The alternative view—that a grammar should be probabilistic and that we evaluate a grammar by virtue of its ability to assign a high probability to the data—has a comparable history. Strongly influenced by Rudolf Carnap, Marvin Minsky, and Chomsky, Solomonoff developed a probabilistic framework in which the grammar for a given set of data is selected by a process that maximizes the probability of the data (taking into account the probability of the grammar as well). Solomonoff [1997] provides an overview of the evolution of Solomonoff's work on this, going back to the mid to late 1950s.

2.4 Bayesian analysis: priors and likelihood

We turn now to what is called Bayesian reasoning. Though we have discussed Bayes' rule, which is essential to Bayesian reasoning, there is more to it than just a simple algebraic manipulation. It is essentially based on the realization that any assignment of probability to an event e is conditioned by the distribution f being used, and on the principle that it makes sense to speak of the probability of using a particular distribution. It is this latter step that is special—and not uncontroversial. Let us look at it more closely.

Bayesian models usually include parameters, that is, variables that take on a specific value (even if that value is not explicitly known in some contexts). If we have a loaded die which can come up on any of its six faces, but we know ahead of time that the probabilities of each face coming up are not all equal, then we might well develop a model in which there are five parameters, one for each of the probabilities that the die will come up 1, 2, 3, 4, or 5; the other value (the probability that it comes up 6) does not need to be specified, since a distribution must add up to 1.0.

We thus have two distinct ways in which we may think of the probability of an event as being conditioned: the value that one random variable takes on can be conditioned by the value that another random variable takes on; it can also be conditioned (in a different sense) by the choice of the values of the parameters adopted in the model. Notationally, this is often written by separating the two types of variables with a semicolon: $p(X) = f(X_{t-1}; \lambda)$.

Let's consider a simple example of this, such as a model for flipping a coin which we have no reason to believe is fair: we believe that there is a probability p that it will come up heads, but we have no prior knowledge at all regarding what that probability is. If we knew p and we decided to flip the coin 100 times, we could assert with what probabilities we would expect the coin to come up 40 times as heads, 50 times as heads, 60 times as heads, and so on. More generally, if we flip a coin n times, then the probability that we will get m heads is the familiar binomial distribution, which can be written as

$$p(X = m) = \binom{n}{m} p^m (1 - p)^{n-m} \tag{2.6}$$

But the question of how we approach the inverse problem remains: how do we estimate p for this coin, if we have flipped it 100 times and it has come up heads 45 of these 100 times? One reasonable answer is to consider all possible values for p and then to choose the one which assigns the highest probability to the observed data. Bear in mind two things: first, the parameter p can take on an uncountable number of values (we said earlier that we would consider only sample spaces with a countable number of objects—which is still true here), and second of all, we may let p varying over all its possible values but that does not create a distribution: there is no sense in which all of the probability values sum (or integrate) to a value of 1.0 as we let p vary all the way from 0 to 1.

It for this reason, in essence, that we distinguish between *probability* and *likelihood*: unlike common English usage, technical usage keeps these two notions completely separate. A likelihood function describes the probability that a model would assign to a particular state of affairs as we consider all the different values that the model's parameters may take on, while a probability

function describes the probability that is assigned by the model (with its parameters fully specified) to all of the events in the sample space.

There is another, closely related aspect of Bayesian analysis that makes it a good model for thinking about systems that learn. If we are prepared to think about knowledge as knowledge of distributions, then we are drawn to asking how this knowledge is updated—in a word, changed—when the systems makes additional observations. Bayes' rule can be understood as a very concrete way to model the update of knowledge in the light of new evidence.

2.4.1 A psychological view of Bayes' rule

When the cognitive system acquires new information, such as hearing some particular linguistic input or observing some aspect of the social or physical environment, it has to update its state of knowledge, in the light of this new information. How can this process of updating be understood? In general, this problem is extremely difficult—it is the notorious "frame problem" of artificial intelligence [McCarthy and Hayes, 1969; Pylyshyn, 1987]—the problem of tracing the consequences for one's overall knowledge of the world, in the light of a specific piece of information. The problem is difficult, because, in general, one's overall state of knowledge can typically change in many ways in order to accommodate new data—and it is not straightforward to decide which modifications should be preferred.

The problem can, however, be addressed head-on for problems in which probabilistic methods can be brought to bear: specifically, where the state of knowledge of the agent, about some particular domain, can be captured by specifying a probability distribution. Suppose we are considering the problem of learning a grammar from experience. The probabilistic approach requires that we begin by specifying an initial probability distribution over these grammars, representing the learner's initial state of knowledge (or, rather, state of ignorance). This is known as the prior distribution over grammars. There is a further step, though, which is require to connect these grammars to actual linguistic data, d—each grammar g_i must itself be associated with a probability distribution, which specifies the probability distribution of each sentence, if g_i is right. In the jargon of the Bayesian approach, we must specify a "prior distribution" over grammars $G = \{g_i\}$, and in this book we will use the symbol $\pi()$ to represent a distribution over grammars and a "likelihood function" $p(d|g)$, which spells out the probability of each piece of linguistic data d in the light of each grammar g.

Now, suppose some linguistic data d is encountered—that is, a particular string of words is heard and assumed to be part of the language being learned. How do we update our beliefs about the various possible grammars? A key

idea in the Bayesian approach is to follow so-called Bayesian updating: that is, we replace the prior probabilities $p(g)$ with so called "posterior" probabilities $p(g|d)$—that is, the probability of the grammars given that we know that the data d has been encountered. And it turns out that this posterior distribution is completely determined by the priors $p(g)$ over the grammar and the likelihoods $p(d|g)$, that is, how likely the data d is, according to each grammar. We'll come to the specifics of how this works later on, but the intuition is straightforward: the probability of a grammar after the data has arrived is proportional to the product of the relevant prior and likelihood terms. Roughly speaking, our updated probability for a grammar, once the linguistic data has arrived, is determined by how probable it was beforehand and how well it "predicted" the observed data.

But where, you may ask, does the prior distribution, from which the posterior is derived, come from (similar questions, with similar potential answers, may be asked about how we flesh out the likelihood term—how do we, say, get from all-or-nothing grammatical rules to probabilities over sentences)? It may, in turn, have been the posterior distribution computed after the observation of some earlier data, but somewhere along the line, as we go up the logical ladder, there must be an end—or rather, a beginning. Somewhere there is a probability distribution that was employed before data was encountered. What do we say about that?

There are three things that the Bayesian analyst is inclined to say at this point. The first is that if enough data has been observed, then it sometimes does not matter very much what the initial hypothesis was, at least if the initial hypothesis is not wholly "unreasonable": under many circumstances, Bayesian reasoning will drive a learner towards a hypothesis that had a very low probability before any data at all was seen. In some practical contexts, particularly where the quantity of the data is large in relation to the complexity of the pattern being learned, this provides suitable reassurance.

A second thing the Bayesian analyst will say is that we can indeed say something about the prior probability distribution that we wish to employ before any data whatsoever has been seen: we can say something, that is, about the universal prior distribution, or some related construct. This is the second sort of general Bayesian reasoning: it consists of an attempt to calculate probabilities with as few givens as is humanly (or superhumanly) possible—roughly, we give high priors to patterns, grammars, or hypotheses which are simple, in a well-defined sense.

A third thing the Bayesian can say is that inductive inference never starts from a blank slate. In any system of representation (even one that is universal and can express any data whatever), some data will be more easily expressed than other data—every system of representation comes with an implicit prior.

Sometimes this viewpoint is expressed by saying that every learner has some kind of inductive bias—that is, there is no such thing as a completely neutral learner.

In sum: Bayesian reasoning is inherently dynamic, in the sense that it always involves the relationship between two probability distributions, a first one which we have access to before a set of observations, and a second which we have access to after the observations and which constitutes a rational update of the first. The Bayesian hopes to shed light on the first probability distribution, before all data, and on how the presentation of succeeding encounters with the world lead to an update of the relevant distributions, which is to say, a better understanding of the world and a more accurate model of it.

2.5 Compression and complexity

In this section, we turn to the problem of finding the right balance detail and generalization in the formulation of human and scientific generalizations: the idea that this question can be frontally assaulted through quantitative means is perhaps the single most important idea that lies behind what we have termed in this book the new empiricism. Until the late 19th century, empiricists were expected to view generalizations as poor cousins of complete descriptions: generalizations are what you get when you leave out the details, what it takes to be fully and completely accurate.

The balance began to tip with the work of Ernst Mach, who emphasized the power of the role of science in organizing enormous amounts of data: the idea began to dawn that compressing a large amount of data in a fashion that allowed the details to be fully recoverable was no trivial matter and no mean feat.

With the dawn of the computer age in the mid 20th century, a crucial notion emerged: that of lossless compression. Although not a household word, and not well-known in either linguistics or psychology, lossless compression is an important concept, one that is closely related to redundancy. Suppose one has to describe a digital image that consists of twenty copies of the Mona Lisa, distributed randomly over a page, each the same. One could retain a bit-by-bit description of that page—or one could describe just one copy of the image, and then specify the x and y coordinates of each of the twenty copies of the image, thereby saving close to 95% of the memory needed for the complete description of the page. Saving the image in this way is an example of a lossless compression, since the shorter description of the image can be used to completely and accurately reproduce the original image.

What if there were a few differences here and there in each of the copies of the Mona Lisa? If the differences were minor, it would still lead to a considerable savings to describe the whole image as twenty copies of the same

basic image and then describe, for each image, how the basic image was slightly distorted in each case. The basic image serves as the basis of the generalization, but room is still provided in the description to allow details to come in, to indicate precisely where the simple generalization is not quite good enough to reproduce the original data.

The goal of lossless compression is to find a way to use regularities that exist in the data in such a way that the data can be reconstructed from a simpler description, where by the word "simpler" we mean "shorter" in some measurable sense, not simpler conceptually. After all, the description of a large image in terms of a very large number of pixels is conceptually very simple, but if there is structure—which amounts to redundancy—in the image, then a shorter overall description can be achieved by using, or extracting, that structure.

There is such a thing as lossy compression; this term is used to describe methods of digital analysis that allow a much more compact description of some data (such as a music recording) which can be used to reconstruct the original recording (or image) in a way that is good enough for practical purposes; mp3 recordings are familiar examples of such compressed formats, and needless to say, lossy compression creates compressed descriptions that are typically much smaller than lossless compression does. But bear in mind that we are only interested in lossless compression.

Is there a right way and a wrong way to compress data? This turns out to be a delicate question. At first blush, the answer would seem to be "no". There are various ways of compressing sound files, video files, and textual files of all sorts. The technology behind WinZip (the family of compression methods called Lempel-Ziv-Welch) identifies substrings of data that occur frequently in a computer file, and some methods may actually work better on certain kinds of data than others; some types of data may be reasonably well compressed by several different methods. So what would "right" and "wrong" mean here? If the goal is to reconstruct the original data that comprised the image, and each method can be guaranteed to completely reconstruct the original, who is to say what is better and what is worse?

But the question, correctly posed, is more nuanced than such an answer would suggest. Let us take a look at the way in which this notion of lossless compression has been used to shed light on the theoretical notion of a random sequence of numbers. We may all think that we know what is meant by a finite sequence of randomly chosen integers, but how can we make that notion clear and firm? The answer turns out to be that we can define what is meant by a sequence that is not random: it is one for which one can supply a lossless description in fewer symbols than it takes to enumerate the numbers separately. A simple example: consider

$$39, 44, 49, 54, 59, 64, 69, 74, 79, 84, 89, 94 \tag{2.7}$$

It is easy to see that this can be described as $39 + 5k$, $k = 0 \ldots 11$, that is, as k goes from 0 to 11. If we allow ourselves to use the basic properties of arithmetic, then we agree that the formula "$39 + 5k$, $k = 0 \ldots 11$" is indeed shorter than the original sequence, and it is the existence of that shorter formula that is what we mean when we say that the sequence is not random.

But the reader may quite rightly object that part of the reason that this sequence of numbers can be compressed to the shorter formula is that there is a good deal of knowledge and structure lurking behind the fact that we allowed ourselves to use the basic properties of arithmetic. Isn't something wrong when you don't seem to have to pay anything (so to speak) for multiplication of numbers, and all that goes into it, when we use the simple formulate "$5k$"?

That is a fair objection, when all is said and done. And so the right definition of what makes a number not random is based on finding a formula that is expressed not in the language of everyday arithmetic but in the language of a universal computer—a Universal Turing Machine, for example. This condition keeps us honest: it forces us to bring our formula down to the very most basic bits that define an arithmetic expression.

So we have summarized the sequence in (2.7) as a formula, but what exactly is that formula? There are at least three obvious candidates (and certainly many, many more):

a. Function $f(k)$, where $k \geq 0$: the sequence of numbers starting at 39, and consisting of k more numbers, each one 5 more than the preceding number; f(11).

b. Function $g(j, k)$, where $j \geq 0$, $k \geq 0$: the sequence of numbers starting at j and consisting of k more numbers, each one 5 more than the preceding number; g(39, 11).

c. Function $h(i, j, k)$, where $i \geq 0$, $j \geq 0$, $k \geq 0$; the sequence of numbers starting at j and consisting of k more letters, each one i more than the preceding number; h(5, 39, 11).

There are two ways of deciding what we want to call the right answer: either we get down and do the hard work of calculating exactly how long each of these programs is and select the shortest one (this is called taking algorithmic complexity seriously); or we look to see what other sequences of numbers we are going to need to compress, if whoever gave us (2.7) has more work for us to do. If the next sequence is (2.8), then we can be pretty sure that (c) is better than (a) or (b)

$$21, 24, 27, 30, 33, 36, 39 = h(3, 21, 6) \tag{2.8}$$

So what is the right generalization, then? Sometimes we learn about the nature of the generalization by seeing more data—(2.8) is helpful after seeing (2.7), but we don' t always have that opportunity; sometimes the data is limited. Sometimes what we need to figure out is what the parameters are that will not be explained but will rather be taken to be arbitrary, that is, as part of the specification sent to the function. In the case of (2.8), it is apparently arbitrary that the interval between the numbers is 3—in (2.7), it was 5; we do not have a way to guess what it might be the next time.

But just because we can't explain why the interval is 5 in one case and 3 in another does not mean we have not explained anything: far from it. We simply are not in a position to predict it.

Let's consider a different sort of variant on these number sequences. Suppose we consider the sequence in (2.9).

$$39, 44, 49, 54, 59, 63, 69, 74, 79, 84, 89, 94 \tag{2.9}$$

The sequence in (2.9) differs from that in (2.7) in just one respect: it has a 63 instead of a 64. So what should we do? What is the best way to compress the data? There are three things we could do. We could throw up our hands and say that there is no way to compress it. Or we could compress (4) just like we compressed (1) and decide that just being off by 1 on one of the digits was not enough to worry about and that if we get in trouble, we'll say that we suspected that there was an error in the data and we tried to clean it up. Or—and this is the right answer—we define a new function $m(k)$ in terms of the function $f(k)$: we say if $k = 5$, then $m(k) = f(k) - 1$; in all other cases, $m(k) = f(k)$.

This formulation makes the empiricist happy. It is a lossless compression of the sequence, it recovers the odd number in the sequence (63 instead of 64), and it wears on its sleeve the fact that the description would be simpler if the correct value were 64—but it isn't.

But the most important aspect of the example is the way in which it illustrates how data and generalization cooperate in a quantitative sense. The more data there is, the more work there is that a good generalization can accomplish: the work it can accomplish is to reduce—which is to say, to compress—the data by extracting the generalizations and leaving only what is unpredictable.

2.6 **The problem of induction**

The idea that there is a deep connection between data compression and the fundamental problem of scientific induction evolved over the eighty years that spanned the period from Ernst Mach's work on the philosophy of science up

to the work by Ray Solomonoff on induction and probability. The problem of induction is the fundamental philosophical problem of science: what justifies us in passing from a finite set of observations about the world to a generalization that covers an infinite number of cases? That is what science is about, after all: finding generalizations that are suggested by, but which go way beyond, the data that serves as the foundation of the proposal. A critical aspect of this passing-beyond-the-finite-to-the-scientific-generalization is the realization that the generalization is always *simpler* than the conjunction of the original observations. To count as a real case of induction—empirical induction, not mathematical induction!—it must be the case that the generalization is simpler. That is not a sufficient condition; any observation, or set of observations, can be made simpler and serve as a set of predictions going beyond the observed data, and most of those generalizations will be wrong; but it is a necessary condition.

2.7 Algorithmic complexity

One of the great ideas developed during the 20th century is that of algorithmic complexity, a notion that grew out of the work of many people, and for which Ray Solomonoff, Andrey Kolmogorov, and Gregory Chaitin are generally recognized as the most important contributors, involving work accomplished largely during the 1950s and 1960s. Some of the central ideas of algorithmic complexity have already emerged in the discussion so far in this book. Part of our goal in this book is to encourage cognitive scientists to take the opportunity to learn more about algorithmic complexity, because we think that this is an area of work—mathematical, and not just mathematical—that can have, and will have, an enormous impact on how we understand the nature of learning and of knowledge. One widely cited book in this area, and one that we recommend to our reader, is *An Introduction to Kolmogorov Complexity and its Applications* [Li and Vitányi, 1997], and we will make quite a few references to it over the course of this book.

Solomonoff [1964a, p. 3] writes

> The "solutions" that are proposed involve Bayes' Theorem. A priori probabilities are assigned to strings of symbols by examining the manner in which these strings might be produced by a universal Turing machine. Strings with short and/or numerous "descriptions" . . . are assigned high a priori probabilities. Strings with long, and/or few descriptions are assigned small a priori probabilities. . . . Turing machines are . . . used to explicate the concepts of "simplicity" or "economy"—the most "simple" hypothesis being that with the shortest "description."

2.8 Grammars as algorithms, and grammatical complexity as algorithmic complexity

From generative grammar, we adopt the notion that a grammar is an algorithm, and from the study of algorithmic complexity, we adopt the notion that there is a well-defined notion of algorithmic complexity: it follows that we have access to a well-defined notion of grammatical complexity. The devil remains in making that explicit and precise, and then in working out whether the notion of grammatical complexity that flows from that source is one that will serve us in some way in the study of grammar, of language use, and of language learning. This book is written, as the reader can see, in the hope and belief that this can be accomplished. The relevant notion of grammatical complexity is sometimes called generative capacity—often subdivided into *weak* and *strong* generative capacity, where *weak* refers to the set of strings that can be generated by the grammar, and *strong* refers to the sets of structures that can be generated.

Languages, uncontroversially, are pairings of sounds (typically in the form of sequences of words) together with meanings. The relation between the two is clearly complex; far from being a simple one-to-one mapping, some sentences can have multiple meanings, whether this is caused by simple lexical ambiguity or by syntactic ambiguity, and the same meaning can be expressed through different sequences of words. A standard, and reasonable, assumption is that behind this mapping lies some latent hierarchical structure—a structural description [Chomsky, 1957]. In this book however we shall have little to say about the notion of meaning, nor about the hierarchical structures that underlie the sound/meaning relation. This is not because we think that these factors are not important or interesting, but rather that they are *unobservable*. We know, exactly, what the sequence of words in a sentence is but what the meaning of a sentence is precisely is still a matter of dispute after centuries indeed millennia of philosophical argument [Quine, 1960, Lewis, 1970, Partee, 2010]. While we do have some knowledge about the possible sets of meanings, their truth conditions and their entailment relations, the nature of the structural descriptions is still less clear.

From a mathematical point of view, therefore, we tend to view grammars as devices that generate merely the strings: sequences of words or phonemes. The trace of the derivational process that the grammar follows in the course of generation can be taken as a structural description.[3] The same sequence of words can be generated by two distinct processes—this gives a natural treatment of

[3] The distinction between derivation tree and derived tree can be important in this context.

ambiguity: we attach the meanings to the derivation trees rather than to the sequences of words.

One direction from which one can study this notion of complexity is the classic Chomsky hierarchy of phrase structure grammars (PSGs): the regular, context-free and context-sensitive languages. From a modern perspective, however, the original characterization of phrase structure grammars as string-rewriting systems and in particular the resulting class of context-sensitive languages seem to be not the best solution. Chomsky early on recognized the limitations of the class of context-free grammars but his proposed extension, context-sensitive grammars, turned out to be far too powerful, and as a result has not been used extensively. Using rewriting systems it is indeed hard to find a natural class that is more powerful than the context-free grammars without going all the way to the context-sensitive grammars. Accordingly, attention shifted towards context-free grammars augmented with transformations, which again unfortunately turned out to be too powerful [Peters and Ritchie, 1973].

While the original PSGs had a number of flaws which made them unsuitable as models for natural language syntax (while still important in other areas of computer science), more modern varieties of PSGs such as Generalized Phrase Structure Grammar (GPSG), Head-Driven Phrase Structure Grammar (HPSG), and the like no longer had those flaws [Borsley, 1996]. The rehabilitation of PSGs became complete with the discovery that Minimalist Grammars [Stabler, 1997], an attempt by Ed Stabler to formalize the ideas of the Minimalist Program, were weakly and strongly equivalent to a PSG formalism, the class of Multiple Context-Free Grammars (MCFGs) [Michaelis, 2001]. This revealed that one of the great divides in syntactic theory—between models that use movement and those that did not—turned out to be, from one perspective, merely a notational difference.

There is now a fairly broad consensus that from a technical point of view, some subclass of the class of MCFGs is adequate for the structural description of natural languages. There are, however, some phenomena which indicate that it might be necessary to augment this formalism with some additional operations, such as copying [Kobele, 2006], and there is in addition some debate about which precise subclass is necessary: one view is that the class of well-nested 2-MCFGs, which defines the same set of languages as the Tree-Adjoining Grammars and various other equivalent formalisms, is adequate—another is that a somewhat larger set of grammars is required.

This is, then, one notion of the complexity of grammar; for each language, we can try to place the set of strings generated by the language at some position

in this hierarchy, in some class of grammars that have sufficient computational resources to generate the set of strings. Of course, if a language is in some class, then it is also in any superclass of that class, and so the most one can hope here is to have some sort of lower bound on the appropriate complexity class.

This is harder than it seems—to show that a language is for example a context-free language would require one to show that there is a context-free grammar that generates all and only the grammatical strings of a language. This is a formidable task for at least two reasons: first, to draw a sharp distinction between the grammatical and ungrammatical seems impossible in the light of the pervasive gradience in natural language. Second, in spite of the best efforts of linguists over several decades, it turns out to be extremely hard to pin down a precise grammar that will draw the grammatical/ungrammatical boundary in a reasonable place. Indeed, the failure of this methodology of manual construction of grammars motivates a shift towards a focus on computational procedures for learning these grammars automatically.

Another notion of complexity is more primitive: how big the grammar is. We can have various more precise measures of the size of the grammar that depend on how exactly we count the length [Chomsky and Halle, 1968], but broadly speaking, the number of symbols we use to write down the grammar, assuming that we have some fixed finite set of symbols that we use in the grammar, will be a sufficiently precise measure, perhaps scaled by the logarithm of the number of symbols that are used.

It is important then to distinguish these two different notions of complexity—one related to the type of the grammar, in the sense of its position in the hierarchy, and one related to its size. One can have small context-sensitive grammars and large regular grammars, and indeed there is often a trade-off between these two ideas. Given a language which is regular, we know that there will be a finite regular grammar which describes it, but there might also be a much smaller context-free grammar that generates exactly the same set of strings. Indeed, given any nontrivial regularity in a language, one could in principle add a component to the grammar which would represent that regularity and thus reduce by some small amount the size of the grammar. However, as Zwicky and Pullum [1987] point out,

> Not every regularity in the use of language is a matter of grammar.

We should bear in mind that while we might want to prefer, all else being equal, the smaller grammar, on occasion there might be regularities which we do not need to extract in order to have an adequate grammar.

2.8.1 Grammars as probabilistic generative models

Grammars, then, in our parlance are generative devices that generate languages as sets of strings together with their associated sets of structures. Such a grammar can generate more than one string and so must have choice points at which it will choose to generate one string rather than another. This uncertainty in the generative process lends itself naturally to a probabilistic treatment; at each point where the process could generate one syntactic object rather than another, we can attach some parameters to our model which determine the probability that it will choose one over the other. A context-free grammar can thus be converted into a probabilistic context-free grammar (PCFG) by adding a set of suitable parameters to the grammar. The most straightforward way of doing this is to attach, for each nonterminal in the grammar a collection of parameters, one for each production, with that nonterminal on the left-hand side; that parameter gives the probability that the nonterminal will be expanded by that production. So if we have a nonterminal that corresponds to a lexical category such as noun, and in the grammar we have 5000 words that can be nouns, then we will have a vector of 5000 parameters, one that specifies the relative probability of each noun.

We illustrate this with a toy PCFG which only has a few productions, as shown in Table 2.1.

Just as in the finite-state models earlier, we make some independence assumptions: instead of a state, we have a nonterminal, but the probability of expanding the nonterminal does not depend on what has happened higher in the tree but on the context in which the nonterminal appears. In this way,

Table 2.1 A very simple PCFG. The sum of the probabilities of all rules with the same left-hand side is 1.

Rule	Probability
S → *NP VP*	1
NP → *Det N*	1
N → cat	0.6
N → truck	0.4
Det → the	0.9
Det → a	0.1
VP → left	0.7
VP → died	0.3

PCFGs are just like their nonprobabilistic ancestor—they are "context-free" rather than context-sensitive. But again, these limitations are not intrinsic to the approach but represent a weakness merely of this particular naive model. More sophisticated models such as tree-substitution grammars make weaker independence assumptions, though this comes at the cost of some additional computational complexity.

Given a probabilistic grammar of this type, we can use it to generate trees and thus sequences of words. Figure 2.4 shows one step in this generation process. It is best not to think of these models as being string-rewriting systems, as they were originally defined, but instead as systems that generate a tree: a derivation tree that records the sequences of steps used in the construction of the string that is its yield. We can also use it as a probabilistic model; and in this case the probability of each tree can be calculated as the product of the probabilities used at each step in its generation. The probability of a string is then the sum of all of the probabilities of the trees that could have given rise to it. If there are none, then the probability of the string is 0; if the string is unambiguous, then there will be only one such probability to be calculated, but in general, for a given string, there may be many possible trees that may each reflect a different interpretation of the string. These probabilities can then be used to calculate the goodness of fit of a corpus to the model; the likelihood of the grammar. We will see in Chapter 6 an example of how such a likelihood is calculated for specific grammars, given a specific corpus.

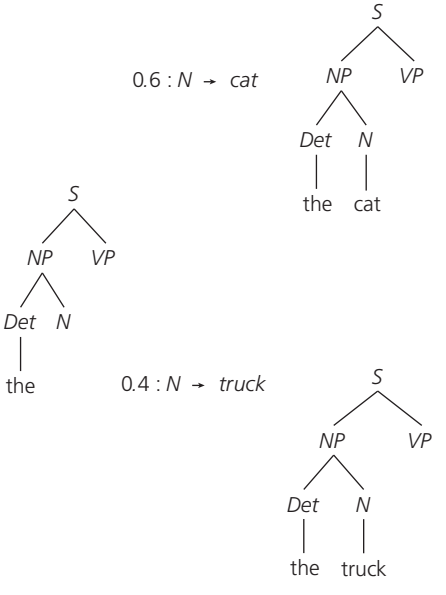

Fig. 2.4 Generating from a PCFG. Here we have a partial tree, on the left, which is expanded probabilistically in two different ways by expanding the *N* node using two different productions.

2.9 Learning and search

The reader must be prepared to deal with statements like "Consider the space of all possible grammars." For most linguists and psychologists, this sounds a bit odd at first. But for the person interested in the abstract task of learning, it is a necessary first step. In any particular discussion, we need to begin with a common understanding as to what the class, or space, of all grammars is that we wish to consider. Learning is a process by which we can accept a certain amount of data and then select one (or perhaps more than one) grammar from this class of all possible grammars.

Learning can thus be thought of as a process of *search*; a process where we search through candidate hypotheses about the grammar in order to find one that has the best chance of being correct. Searching requires two ingredients—a goal, that is, a property that will pick out the object you are looking for, and a search strategy that enables you to find it without exhaustively searching through every possible option. The property we look for is normally specified in terms of an *objective function*, a mathematical formula that specifies how good a candidate is. We then look for the grammar that is best according to this property—the grammar that maximizes the objective function.

The typical objective function in Bayesian inference is the *posterior probability*, the probability of the grammar (h for *hypothesis*) given the data $p(h|d)$. It can make sense to talk about the probability of a hypothesis regarding some set of data only if we can speak sensibly about the class of all possible hypotheses that deal with that data. The reason for this is that a probability can only be defined if we can be certain that as we sum over all possible elements, the sum of the probabilities is 1.0 in the limit. This requires that we have a prior distribution $p(h)$ that defines what the initial probability of a grammar is before we have seen any data.[4]

The second ingredient is a computational strategy for finding the best hypothesis—and defining such a strategy depends crucially on the properties of the search space itself, which depend in turn on properties of the grammars. The metaphor of the space of grammars makes this clear: it relies on the intuition that some grammars are close to each other with respect to some measure of similarity. In the case of probabilistic grammars, we might have two grammars which have the same set of rules and very similar parameters.

The smaller the search space, the easier the search problem. This much seems to be a truism, but even the smallest plausible grammar spaces are so large that

[4] It is possible to have what are called *improper priors*, which do not sum to one.

an exhaustive search is computationally impossible or implausible; this means that even if we have an objective function that picks out a suitable grammar, it may be very hard indeed to find it in the space of all possible grammars.

We have heard from colleagues the complaint that searching for hypotheses in a well-defined space hardly feels like *learning*—after all, the solution is "built in" via the specification of the hypothesis space. Searching through a defined hypothesis space to find hypotheses and calculate their relative probability does not encompass the sort of spirit of discovery that learning it feels like from the inside, or that discovering the hypothesis space itself would constitute. This is an understandable intuition to have; it does feel like a key part of the solution has been built in by the specification of the problem and that discovering that specification is "the hard part." However, it's important to note that a model that discovered that specification would still not feel like learning, according to this very same intuition. This is because such a specification can itself be seen as a specific hypothesis in a more abstract hypothesis space—a hypothesis space of possible specifications; and finding *that* specific hypothesis could only be accomplished in one of two ways. Either the hypothesis would have appeared there completely randomly and arbitrarily, or there would have been some understandable process or set of rules by which it was added. As we have seen, however, that process is what implicitly defines a space of its own (in this case, the space of possible specifications). And because that space is specified (implicitly, by that process or set of rules) in the exact same way the original hypothesis space was specified (implicitly, by the original generative process), the hypotheses within it are "built in" in the same way that the original hypotheses were. In general, the only way for something to be learned that *doesn't* amount to finding it out of a defined hypothesis space is for it to be able to spring into the hypothesis space in such a way that is essentially random (i.e., unexplainable via some process or rule). If this is truly what learning is, it seems to preclude the possibility of studying it scientifically; but luckily, this is not what most of us generally mean by learning [Perfors, 2012].

No one will disagree strongly if we acknowledge from the outset that it is convenient to break down the ways in which grammars can vary into relatively distinct subparts. For example, it is often convenient to think of a language as consisting of a lexicon (finite, or even infinite) of words, and a set of grammatical principles determining how words can be strung together to make grammatical and meaningful sentences. And again there would be little disagreement if we were to say that each language assembles its lexicon from a set of concatenative processes involving an inventory of phonemes, or something

like phonemes, that is particular to the language.[5] Computational thinkers often use the symbol Σ to represent the set of phonemes, and the set of all finite strings formed from Σ is written Σ^*; hence, we can say that a language chooses its lexicon as a subset of Σ^* if Σ is its inventory of phonemes. If there is an upper bound to the length of a possible word, and an upper bound to the size of a language's lexicon, then there is only a finite number of lexicons that can be selected from a given inventory of phonemes Σ.

Reasoning along these lines, one might try to claim that there are as a result only a finite number of possible languages; again taking the assumption that there are a finite number of lexical and syntactic categories, and placing some bounds on the size of the rules in the grammar, perhaps limiting them to be binary branching, we can arrive at the conclusion that there are only a finite number of possible languages. While this reasoning is arbitrary and defective in a number of ways, as has been noted before [Pullum, 1983], let us assume for the moment that we accept the conclusion. Surely, in this case, the fact that the space of possible grammars is now finite means that the learnability problem is trivially easy? For a while it seemed that this might be the case, but it rapidly became clear that this view was too optimistic and that even in this somewhat implausible situation, the search problem remains hard. We discuss the computational problems involved later on, in Chapter 4.

2.10 Generalization, reasoning, and learning

A fundamental question in understanding how people learn and reason about the world is why we generalize beyond the input we receive at all—why don't we simply memorize everything we encounter? The decision to generalize (so to speak) runs into a logical problem when there are an infinite number of possible features along which that generalization could be formed. Nevertheless, we *must* generalize because the ability to make inferences and predict data we have not previously observed relies on our ability to extract structure from our observations of the world. If we do not generalize, we cannot learn, even though any act of generalization is, by definition, a simplification of the data in the world, and even though it can result in error. What is critical is to simplify the data in such a way as to find the optimal balance between the gain in generalization and the cost due to error.

[5] We need a fairly broad definition of concatenation to allow, for example, for the intercalation necessary for Semitic morphology.

Achieving this balance is one of the fundamental goals of any learner, and indeed of any scientific theory, or of any computational or statistical framework. Too much emphasis on simplicity means the learner is unable to learn from data, producing a high degree of error; too much emphasis on precisely memorizing the data means that the learner overfits, unable to capture the correct underlying generalizations. Bayesian models capture the trade-off between simplicity and goodness-of-fit in an optimal way—any learner performing the trade-off in this way would be guaranteed to predict future events more accurately than a learner calculating the trade-off differently. Because human learners—including children—care primarily about predicting future events, they too must adopt some version of this trade-off. But how are simplicity (prior probability) and goodness-of-fit (likelihood) actually calculated and used?

The definition of simplicity and the corresponding calculation of $p(h)$ are not the result of externally imposed ad hoc mechanisms; rather, they emerge naturally from the assumption that hypotheses (which can be grammars or any other type of linguistic representation) themselves are generated from a space of candidate hypotheses. To illustrate this schematically, we can imagine that the hypotheses in Figure 2.5 correspond to different sets of ellipses within a two-dimensional space.[6] Simpler hypotheses require fewer choice points during the generation process. Hypothesis A can be fully captured by making only four choices: two for the coordinates of the center of the ellipse (x and y), one for its major axis (a), and one for its minor axis (b). By contrast, Hypothesis C contains thirty distinct ellipses and therefore requires 120 separate choices to

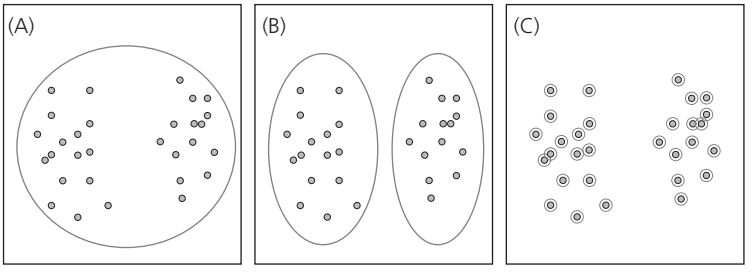

Fig. 2.5 Hypothesis A is too simple, C is too complex, and B is "just right." Hypothesis A is quite simple but fits the observed data poorly; C fits closely but is highly complicated. The best description of the data should optimize a trade-off between complexity and fit, as in B.

[6] This example is explored more fully in Perfors et al. [2011a], which presents a tutorial introduction to Bayesian modeling in cognitive science.

specify, four for each one. This notion of calculating complexity as a function of the number of choice points is a reflection of the idea that the more complicated something is, the more alternatives to it there are that might have been generated instead of it during a generation process. The more choices a hypothesis resulted from, the more likely it is that those choices could have been made in a different way, resulting in a different hypothesis.

The precise prior probability of a hypothesis is therefore not arbitrarily assigned but rather falls out in a principled way from how the hypotheses are generated. The generative model for the hypotheses in Figure 2.5 is one that can result in any possible combination of ellipses within the space. A different generative model would result in a different—but no less principled—assignment of prior probabilities. For instance, if we assumed that the regions could be squares rather than ellipses, then each region would require three choice points rather than four: the x and y coordinates of the center of the square, plus its width. The logic favoring simple hypotheses would be the same: multiple regions will still be a priori less likely than a few. The precise generative model therefore matters for determining exactly what the relative probability of a hypothesis would be, but most reasonable models would give qualitatively similar relative probabilities to qualitatively similar hypotheses.

How well data is predicted by the hypothesis is captured by the likelihood, given by $p(d|h)$. Although the likelihood can sometimes be difficult to calculate in practice, it is straightforward to understand intuitively. For instance, Hypothesis C in Figure 2.5 clearly has a high likelihood: if the hypothesis is true—that is, if the data is truly generated by thirty distinct underlying processes corresponding to the thirty ellipses of C—then the data points could hardly be anywhere else. Hypothesis C therefore fits the data extremely well. By contrast, Hypothesis A has a relatively low likelihood: it does not explain why the data points are found where they are. After all, according to A, the thirty data points would be just as likely if they were each randomly located in other places within the blue ellipse. The ratio of the observed data points to the area for predicted data is low for A, since the data could easily have been elsewhere, but high for C, since it couldn't. Likelihood is, essentially, this ratio; thus, hypotheses that make specific predictions—those with more explanatory power—are favored in the likelihood.

The Bayesian framework, then, offers a natural way to both calculate the simplicity of different hypotheses or theories and then evaluate those theories on the basis of how well they account for the observed data. Bayes' rule offers a principled way to evaluate the trade-off between simplicity (prior probability) and goodness-of-fit (likelihood). Thus, as in Figure 2.5, it will naturally tend to prefer hypotheses (like Hypothesis B) that—like Goldilocks in the famous

story—are neither too weak nor too strong but are "just right." Hypothesis C, for instance, clearly has a high degree of goodness-of-fit (likelihood), while Hypothesis A has a relatively low likelihood. However, Hypothesis A is simple, while C is quite complex. The best description of the data would be a hypothesis that optimizes the trade-off between complexity and fit, as in Hypothesis B.

Bayes' rule and the mathematics of probability theory thus provide a principled way to combine these two factors in such a way to guarantee optimal inductive reasoning ability. As we will see, an ideal learner incorporating a simplicity metric will be able to predict the sentences of the language with an error that approaches 0 as the size of the corpus goes to infinity [Solomonoff, 1978; Chater and Vitányi, 2007]. It is therefore reasonable to think that the Bayesian approach may be well suited to providing an objective way to compare different grammatical theories and formalisms within linguistics—and is thus another method for addressing many of the questions that have occupied linguists for years. As a result of performing this trade-off, the amount and type of data can have a profound effect on the inferred theory. Especially when the representations involved are richly structured, what look like discrete qualitative shifts emerge simply because the trade-off favors different theories as the data changes. In Chapter 6 we will see how shifts in behavior that qualitatively parallel human learning are a natural by-product of Bayesian learning of realistic data.

One striking example of this, which we will explore more fully in subsequent chapters, is that Bayesian models can naturally handle situations in which there is no negative evidence. As in Figure 2.5, Bayesian inference recognizes that a hypothesis that is too complex for the observed data will overfit, missing important generalizations, while one that is insufficiently complex will not be explanatory enough. Because of this, a distinctive pattern of reasoning naturally emerges as the amount of data changes. When there are few data points, the simpler theories are favored, resulting in a tendency towards overgeneralization. As the number of data points increases, the likelihood increasingly favors the theory that most closely matches the observed data, and overgeneralization decreases. This captures the notion of a suspicious coincidence, since hypotheses that predict the observation of data points that in fact never occur tend to be increasingly disfavored. It also provides a natural solution to the problem of deciding among hypotheses given positive-only examples. As the size of the dataset approaches infinity, a Bayesian learner rejects larger or more overgeneral hypotheses in favor of more precise ones. But with limited amounts of data, the Bayesian approach can make more subtle predictions, as the graded size-based likelihood trades off against the preference for simplicity in the prior. The likelihood in Bayesian learning can thus

be seen as a principled quantitative measure of the weight of implicit negative evidence—one that explains how and when overgeneralization should occur.

2.11 Gold's work

The oldest and perhaps most influential formal model of learnability in the context of language learning is Gold's paradigm of *identification in the limit* introduced in his seminal paper [Gold, 1967]. Indeed, for many linguists, learnability begins and ends with the models presented in that paper. Gold considers several different models, but the most relevant for our purposes is the model where the learner receives only positive examples. We will briefly outline the model here; for more detailed discussion see Johnson [2004] and Clark and Lappin [2011]. One of the important properties of this learning model is that it is not probabilistic, though probabilistic variants of it have been proposed.

We assume that some language L has been chosen in some way; we call this the target language. The learning model proceeds sequentially: the learner is provided with an infinite sequence of examples drawn from the target language. This sequence is guaranteed to contain all the elements of the language at some point and not to contain any elements that are not in the language; but other than these necessary constraints, the sequence does not need to satisfy any other limitations. A sequence of this type is called a *presentation* of the language L. We can write the sequence as $w_1, w_2, \ldots$.

After receiving each example, the learner will produce a hypothesis; thus, the learner will produce an infinite sequence of hypotheses H_1, H_2, and so on. The learning criterion is quite simple: the learner must converge to a single correct hypothesis. Formally, there are several different ways of expressing this. The most explicit is this: there must be some point at which the learner has converged on a particular hypothesis, and this hypothesis is correct. There must be some N such that $L(H_N) = L_0$ and for all $n > N, H_N = H_n$. The learner need not be able to tell when it has converged.

We say that a learner learns the language L if for every presentation of L, the learner will converge in the sense just defined to a representation for L. It is worth pausing here to note how this differs from a probabilistic learning model.

Gold's paper is notable for the very strong negative results that he obtained in this model. He defined the idea of a superfinite language class, which is any class that contains all finite languages and at least one infinite one, and showed that no superfinite class could be identified in the limit from positive data alone. Since the classes of regular languages, and a fortiori context-free

and context-sensitive languages are clearly superfinite, they cannot therefore be learned. This result has been extremely influential in the development of learnability theory as it is applied to language acquisition; it motivates many foundational assumptions such as the Subset Principle.

This very strong negative result arises out of a peculiarity of the Gold model—the learner must succeed for every possible presentation of the data, and thus for presentations even when they are constructed *adversarially*. Suppose we have some intelligent adversary whose goal is to stop the learner from converging in the Gold sense—this adversary will try to construct a presentation that will trick the learner into making an infinite number of errors. Gold shows that there is a strategy that the adversary can use that will work for any superfinite ~~language~~; indeed, the negative results have been made even stronger by subsequent researchers. All that is needed for an adversary to succeed in making the learner fail is for there to be an infinite increasing sequence of languages in the class—that is, a sequence $L_1 \subset L_2 \ldots$, and another language that contains all of these, $L_\infty = \bigcup_i L_i$. If the learner does in fact learn all of the L_i, then the adversary can construct a presentation for L_∞ where the learner will make an infinite number of errors. The adversary in this case can trick the learner into hypothesizing first L_1, then L_2, and so on, using a presentation for L_∞. Thus, the learner will always hypothesize successively larger sets from L_i and will never make the leap to the more general hypothesis L_∞.

The reason for this result is, broadly, that the presentation can indefinitely defer key examples from the language. As a result, it is hard for the learner to know what is *not* in the language, and the presence or absence of negative data becomes a crucial point in deciding learnability. This is misleading. One of the properties of natural language is that almost all sequences of words are ungrammatical: the set of grammatical strings is a very "small" subset of the set of all possible finite strings of English words. As a result, knowing that a particular sequence of words is grammatical tells you a lot, while learning that some sequence is not grammatical is in general of little use [Navarro and Perfors, 2011].

From one point of view, we are restating a problem mentioned earlier—why does the learner generalize beyond the input? How could a learner know, having seen, say, a few thousand sentences, that the correct grammar is an infinite set that contains those sentences rather than just the finite set of sentences itself? The Gold model casts this problem into particularly sharp relief because under this model the decision is impossible to make reliably; that is to say, the learner cannot always decide correctly when it is appropriate to generalize or not. Making this decision crucially depends on the frequency with which the individual items, sentences, appear. We can consider two extreme cases: one

where all of the sentences observed have occurred many times, and another where nearly all of them have occurred only once. In the first case one might be hesitant to conclude that there are any other examples; if we have seen everything several times, then the chance that there are unseen sentences is low. On the other hand, if many of the sentences have occurred only once, then it is highly likely that there are many other unseen sentences. If you press a button ten times and get ten different results, then one would expect the next time to get a different result; if you get the same result ten times, then the eleventh you expect to be the same. This much seems intuitively obvious and has sound statistical reasoning behind it [Good, 1953]. But it is less easy to reconcile this rather optimistic conclusion with the pessimistic results of Gold. The problem lies in the Gold model: the frequency information on which this decision relies is unreliable in this paradigm. The sequence of examples is not generated randomly, and therefore the number of times each example occurs does not contain useful information. Accordingly, learners in the Gold paradigm can't decide whether to generalize or not.

In reality, the frequency of examples does contain a great deal of information, and the fact that the majority of sentences that we hear are ones that we hear for the first time is one of the defining characteristics of natural language; its unbounded productivity constantly reminds us that the set of sentences we have already heard is not exhaustive. From the point of view of Bayesian learning, this frequency information is vital: the computation of the fit of the corpus to the grammar depends on the number of times each item occurs. If something occurs many times, then the optimal model will in general memorize that particular idiosyncratic example, whereas if it occurs only once, the lack of fit of the grammar is not highly penalized.

2.12 Biological plausibility, and the extent to which we care about it

We know that talk about learning as *performing a search through the space of all grammars*—or, indeed, simply the prospect of formalizing language and language learning—may leave some readers with the sense that we're headed in the wrong direction. We have colleagues who say that, after all, human cognition and human language capacity is ultimately realized in the brain, and a critical concern, then, ought to be whether the brain is capable of reasoning as our models and mathematical analyses do. And if we *must* model things, they add, why not an approach that is more neurally inspired? Some cognitive scientists prefer to look to a connectionist sort of model—like the Parallel Distributed Processing approach developed as a neurally inspired model of the

cognitive process [Rumelhart and McClelland, 1986a]. Like the brain, connectionist networks contain many highly interconnected, active processing units that communicate with each other by sending activation or inhibition through their connections. As in the brain, learning appears to involve modifying connections, and knowledge is represented in a distributed fashion over the connections. The result is that representations degrade gracefully with neural damage, and reasoning can be probabilistic and fuzzy rather than all-or-none.

By comparison with connectionist networks, our models and mathematical analyses may appear implausible neurologically. How could structured symbolic representations like grammars or logics be instantiated in our neural hardware? How could our cortex encode hypotheses and compare them based on a trade-off between their simplicity and goodness-of-fit? Perhaps most problematically, how could the brain approximate anything like optimal inference in a biologically realistic timeframe, when conventional algorithms for Bayesian inference running on conventional computing hardware take days or weeks to tackle problems that are vastly smaller than those the brain solves?

One important point is to realize that the sort of explanations and analyses we use almost all take place on what David Marr called the "computational level" [Marr, 1982]. Explanations on this level seek to understand cognition based on what its goal is, why that goal would be appropriate, and the constraints on achieving that goal, rather than precisely how it is implemented algorithmically. This is an important thing to do, because the nature of the reasoning may often depend more on the learner's goals and constraints than it does on the particular implementation. If one wants to understand a handheld calculator, it is more important to be aware of its function and intended purpose than it is to be able to follow the details of its wiring.

Being able to precisely specify and understand optimal reasoning is also useful for performing ideal learnability analysis of the sort that Gold did, and which we do in some of the chapters here. This is especially important if one wants to understand development: what must be built into the newborn mind in order to explain how infants eventually grow to be adult reasoners, with adult knowledge? Optimal learnability analyses, taking into account the goals and constraints of the organism, establish the bounds of the possible: if some knowledge could not possibly be learned by an optimal learner presented with the type of data children receive, it is probably safe to conclude either that actual children could not learn it either or that some of the assumptions about what the learner knows or can do are inaccurate.

More importantly, however, we are not currently too troubled by concerns about biological plausibility for two main reasons. First, we really know so little about the brain that it is hard to tell at this point what is biologically plausible

and what is not. It may seem to those used to working with serial computers that searching these enormous hypothesis spaces quickly enough is impossible; but the brain is a parallel computing machine made up of billions of highly interconnected neurons. The sorts of calculations that take a long time on a serial computer, like a sequential search of a hypothesis space, might be much more easily performed (or approximated) in parallel. They also might not; but whatever the future holds, the indications so far serve as a reminder of the danger of advancing from the "argument from incredulity" to any conclusions about biological plausibility.

Second, it is also important to note that more biologically inspired models like neural networks are themselves still unrealistic in important ways (see, e.g., Crick and Asanuna [1986] for a more thorough discussion). Single units in neural networks are assumed to have both excitatory and inhibitory connections, which is not neurally plausible. This is a problem because the primary learning mechanism, back-propagation, relies on the existence of such connections [Rumelhart and McClelland, 1986a]. A related problem is that errors do not propagate backwards in the brain, as assumed by the back-propagation algorithm. These issues are being overcome as the state of the art advances (see Rao et al. [2002] for some examples), but for the models most commonly used in cognitive science and linguistics—perceptrons, multilayered recurrent networks, and Boltzmann machines—they remain a relevant concern. Different techniques are therefore biologically plausible in some ways and perhaps less so in others. Knowing so little about the neurological mechanisms within the brain, it is difficult to characterize how plausible either approach is or how much the ways they fall short impact their utility.

Moreover, there are some indications that Bayesian-like reasoning may actually occur even on the neural level. Probability distributions can in fact be represented by neurons, and they can be combined according to a close approximation of Bayes' rule; posterior probability distributions may be encoded in populations of neurons in such a way that Bayesian inference is achieved simply by summing up firing rates [Ma et al., 2006]. Spiking neurons can be modeled as Bayesian integrators accumulating evidence over time [Deneve, 2004]. Recurrent neural circuits are capable of performing both hierarchical and sequential Bayesian inference [Deneve, 2004; Rao, 2004]. Even specific brain areas have been studied: for instance, there is evidence that the recurrent loops in the visual cortex integrate top–down priors and bottom–up data in such a way as to implement hierarchical Bayesian inference [Lee and Mumford, 2003]. This work, though still in its infancy, suggests that concerns about biological plausibility may not, in the end, prove to be particularly problematic.