When perceptron-like machines came on the scene, we found that in order to understand their capabilities we needed some new ideas. It was not enough simply to examine the machines themselves or the procedures used to make them learn. Instead, we had to find new ways to understand the problems they would be asked to solve. This is why our book turned out to be concerned less with perceptrons per se than with concepts that could help us see the relation between patterns and the types of parallel-machine architectures that might or might not be able to recognize them.

Why was it so important to develop theories about parallel machines? One reason was that the emergence of serial computers quickly led to a very respectable body of useful ideas about algorithms and algorithmic languages, many of them based on a half-century's previous theories about logic and effective computability. But similarly powerful ideas about parallel computation did not develop nearly so rapidly—partly because massively parallel hardware did not become available until much later and partly because much less knowledge that might be relevant had been accumulated in the mathematical past. Today, however, it is feasible either to simulate or to actually assemble huge and complex arrangements of interacting elements. Consequently, theories about parallel computation have now become of immediate and intense concern to workers in physics, engineering, management, and many other disciplines—and especially to workers involved with brain science, psychology, and artificial intelligence.

Perhaps this is why the past few years have seen new and heated discussions of network machines as part of an intellectually aggressive movement to establish a paradigm for artificial intelligence and cognitive modeling. Indeed, this growth of activity and interest has been so swift that people talk about a "connectionist revolution." The purpose of this epilogue, added in 1988, is to help present-day students to use the ideas presented in *Perceptrons* to put the new results into perspective and to formulate more clearly the research questions suggested by them. To do this succinctly, we adopt the strategy of focusing on one particular example of modern connectionist writing. Recently, David Rumelhart, James McClelland, and fourteen collaborators published a two-volume work that has become something of a connectionist manifesto: *Parallel Distributed Processing* (MIT Press, 1986). We shall take this work (hence-

forth referred to as *PDP*) as our connectionist text. What we say about this particular text will not, of course, apply literally to other writings on this subject, but thoughtful readers will seize the general point through the particular case. In most of this epilogue we shall discuss the examples in *PDP* from inside the connectionist perspective, in order to flag certain problems that we do not expect to be solvable within the framework of any single, homogeneous machine. At the end, however, we shall consider the same problems from the perspective of the overview we call "society of mind," a conceptual framework that makes it much more feasible to exploit collections of specialized accomplishments.

PDP describes *Perceptrons* as pessimistic about the prospects for connectionist machinery:

". . . even though multilayer linear threshold networks are potentially much more powerful . . . it was the limitations on what perceptrons could possibly learn that led to Minsky and Papert's (1969) pessimistic evaluation of the perceptron. Unfortunately, that evaluation has incorrectly tainted more interesting and powerful networks of linear threshold and other nonlinear units. As we shall see, the limitations of the one-step perceptrons in no way apply to the more complex networks." (vol. 1, p. 65)

We scarcely recognize ourselves in this description, and we recommend rereading the remarks in section 0.3 about romanticism and rigor. We reiterate our belief that the romantic claims have been less wrong than the pompous criticisms. But we also reiterate that the discipline can grow only when it makes a parallel effort to critically evaluate its apparent accomplishments. Our own work in *Perceptrons* is based on the interaction between an enthusiastic pursuit of models of new phenomena and a rigorous search for ways to understand the limitations of these models.

In any case, such citations have given our book the reputation of being mainly concerned with what perceptrons cannot do, and of having concluded with a qualitative evaluation that the subject was not important. Certainly, some chapters prove that various important predicates have perceptron coefficients that grow unmanageably large. But many chapters show that other predicates can be surprisingly tractable. It is no more apt to describe our mathematical theorems as pessimistic than it would be to say the same about

deducing the conservation of momentum from the laws of mechanics. Theorems are theorems, and the history of science amply demonstrates how discovering limiting principles can lead to deeper understanding. But this happens only when those principles are taken seriously, so we exhort contemporary connectionist researchers to consider our results seriously as sources of research questions instead of maintaining that they "in no way apply."

What Perceptrons Can't Do

To put our results into perspective, let us recall the situation in the early 1960s: Many people were impressed by the fact that initially unstructured networks composed of very simple devices could be made to perform many interesting tasks—by processes that could be seen as remarkably like some forms of learning.

A different fact seemed to have impressed only a few people: While those networks did well on certain tasks and failed on certain other tasks, there was no theory to explain what made the difference—particularly when they seemed to work well on small ("toy") problems but broke down with larger problems of the same kind.

Our goal was to develop analytic tools to give us better ideas about what made the difference. But finding a comprehensive theory of parallel computation seemed infeasible, because the subject was simply too general. What we had to do was sharpen our ideas by working with some subclass of parallel machines that would be sufficiently powerful to perform significant computations, that would also share at least some of the features that made such networks attractive to those who sought a deeper understanding of the brain, and that would also be mathematically simple enough to permit theoretical analysis. This why we used the abstract definition of *perceptron* given in this book. The perceptron seemed powerful enough in function, suggestive enough in architecture, and simple enough in its mathematical definition, yet understanding the range and character of its capabilities presented challenging puzzles.

Our prime example of such a puzzle was the recognition of connectedness. It took us many months of work to capture in a formal proof our strong intuition that perceptrons were unable to

represent that predicate. Perhaps the most instructive aspect of that whole process was that we were guided by a flawed intuition to the proof that perceptrons cannot recognize the connectivity in any general or practical sense. We had assumed that perceptrons could not even detect the connectivity of hole-free blobs—because, as we supposed, no local forms of evidence like those in figure 5.7 could correlate with the correct decision. Yet, as we saw in subsection 5.8.1, if a figure is known to have no holes, then a low-order perceptron can decide on its connectivity; this we had not initially believed to be possible. It is hard to imagine better evidence to show how artificial it is to separate "negative" from "positive" results in this kind of investigation. To explain how this experience affected us, we must abstract what we learned from it.

First we learned to reformulate questions like "Can perceptrons perform a certain task?" Strictly speaking, it is misleading to say that perceptrons cannot recognize connectedness, since for any particular size of retina we can make a perceptron that will recognize any predicate by providing it with enough φs of sufficiently high order. What we did show was that the general predicate requires perceptrons of unbounded order. More generally, we learned to replace globally qualitative questions about what perceptrons cannot do with questions in the spirit of what is now called computational complexity. Many of our results are of the form $M = f(R)$, where R is a measure of the size of the problem and M is the magnitude of some parameter of a perceptron (such as the order of its predicates, how many of them might be required, the information content of the coefficients, or the number of cycles needed for learning to converge). The study of such relationships gave us a better sense of what is likely to go wrong when one tries to enlarge the scale of a perceptron-like computation. In serial computing it was already well known that certain algorithms depending on search processes would require numbers of steps of computation that increased exponentially with the size of the problem. Much less was known about such matters in the case of parallel machines.

The second lesson was that in order to understand what perceptrons can do we would have to develop some theories of "problem domains" and not simply a "theory of perceptrons." In previous

work on networks, from McCulloch and Pitts to Rosenblatt, even the best theorists had tried to formulate general-purpose theories about the kinds of networks they were interested in. Rosenblatt's convergence theorem is an example of how such investigations can lead to powerful results. But something qualitatively different was needed to explain why perceptrons could recognize the connectedness of hole-free figures yet be unable to recognize connectedness in general. For this we needed a bridge between a theory about the computing device and a theory about the content of the computation. The reason why our group-invariance theorem was so useful here was that it had one foot on the geometric side and one on the computational side.

Our study of the perceptron was an attempt to understand general principles through the study of a special case. Even today, we still know very little, in general, about how the costs of parallel computation are affected by increases in the scale of problems. Only the cases we understand can serve as bases for conjectures about what will happen in other situations. Thus, until there is evidence to the contrary, we are inclined to project the significance of our results to other networks related to perceptrons. In the past few years, many experiments have demonstrated that various new types of learning machines, composed of multiple layers of perceptron-like elements, can be made to solve many kinds of small-scale problems. Some of those experimenters believe that these performances can be economically extended to larger problems without encountering the limitations we have shown to apply to single-layer perceptrons. Shortly, we shall take a closer look at some of those results and see that much of what we learned about simple perceptrons will still remain quite pertinent. It certainly is true that most of the theorems in this book are explicitly about machines with a single layer of adjustable connection weights. But this does not imply (as many modern connectionists assume) that our conclusions don't apply to multilayered machines. To be sure, those proofs no longer apply unchanged, because their antecedent conditions have changed. But the phenomena they describe will often still persist. One must examine them, case by case. For example, all our conclusions about order-limited predicates (see section 0.7) continue to apply to networks with multiple layers, because the order of any unit in a given layer is bounded by the *product* of the

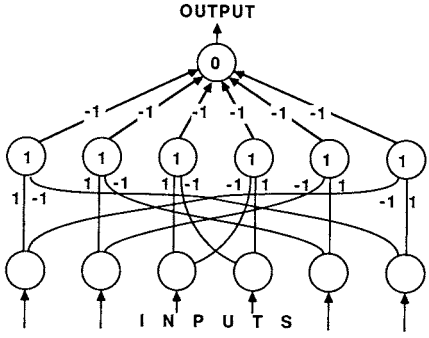

Figure 1 Symmetry using order-2 disjunction.

orders of the units in earlier layers. Since many of our arguments about order constrain the representations of group-invariant predicates, we suspect that many of those conclusions, too, will apply to multilayer nets. For example, multilayer networks will be no more able to recognize connectedness than are perceptrons. (This is not to say that multilayer networks do not have advantages. For example, the product rule can yield logarithmic reductions in the orders and numbers of units required to compute certain high-order predicates. Furthermore, units that are arranged in loops can be of effectively unbounded order; hence, some such networks *will* be able to recognize connectedness by using internal serial processing.)

Thus, in some cases our conclusions will remain provably true and in some cases they will be clearly false. In the middle there are many results that we still think may hold, but we do not know any formal proofs. In the next section we shall show how some of the experiments reported in *PDP* lend credence to some such conjectures.

Recognizing Symmetry

In this section we contrast two different networks, both of which recognize symmetrical patterns defined on a six-point linear retina. To be precise, we would like to recognize the predicate *X is symmetric about the midpoint of R*. Figure 1 shows a simple way to represent this is as a perceptron that uses $R\ \varphi$ units, each of order 2. Each one of them will locally detect a deviation from symmetry

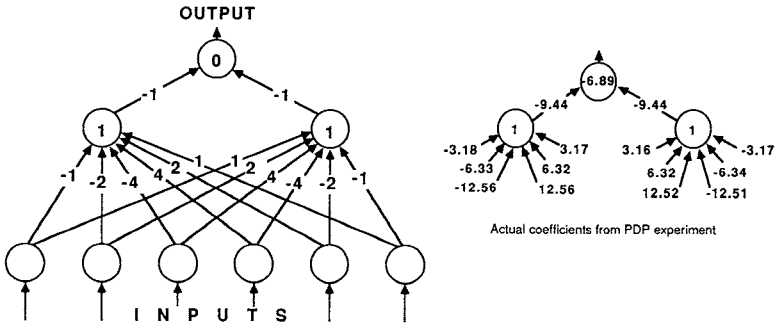

Figure 2 Symmetry using order-R stratification.

at two particular retinal points. Figure 2 shows the results of an experiment from *PDP*. It depicts a network that represents ψ_{SYMMETRY} in quite a different way. Amazingly, this network uses only two φ functions—albeit ones of order R.

The weights displayed in figure 2 were produced by a learning procedure that we shall describe shortly. For the moment, we want to focus not on the learning problem but on the character of the coefficients. We share the sense of excitement the *PDP* experimenters must have experienced as their machine converged to this strange solution, in which this predicate seems to be portrayed as having a more holistic character than would be suggested by its conjunctively local representation. However, one must ask certain questions before celebrating this as a significant discovery. In *PDP* it is recognized that the lower-level coefficients appear to be growing exponentially, yet no alarm is expressed about this. In fact, anyone who reads section 7.3 should recognize such a network as employing precisely the type of computational structure that we called stratification. Also, in the case of network 2, the learning procedure required 1,208 cycles through each of the 64 possible examples—a total of 77,312 trials (enough to make us wonder if the time for this procedure to determine suitable coefficients increases exponentially with the size of the retina). *PDP* does not address this question. What happens when the retina has 100 elements? If such a network required on the order of 2^{200} trials to learn, most observers would lose interest.

This observation shows most starkly how we and the authors of *PDP* differ in interpreting the implications of our theory. Our "pes-

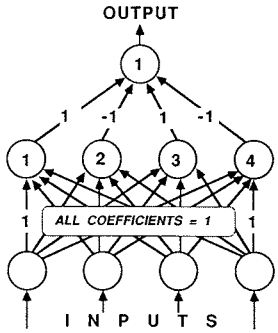

Figure 3 Parity using Gamba masks.

simistic evaluation of the perceptron" was the assertion that, although certain problems can easily by solved by perceptrons on small scales, the computational costs become prohibitive when the problem is scaled up. The authors of *PDP* seem not to recognize that the coefficients of this symmetry machine confirm that thesis, and celebrate this performance on a toy problem as a success rather than asking whether it could become a profoundly "bad" form of behavior when scaled up to problems of larger size.

Both of these networks are in the class of what we called Gamba perceptrons in section 13.1—that is, ordinary perceptrons whose φ functions are themselves perceptrons of order 1. Accordingly, we are uncomfortable about the remark in *PDP* that "multilayer linear threshold networks are potentially much more powerful than single-layer perceptrons." Of course they are, in various ways—and chapter 8 of *PDP* describes several studies of multilayer perceptron-like devices. However, most of them—like figure 2 above—still belong to the class of networks discussed in *Perceptrons*.

Also in chapter 8 of *PDP*, similar methods are applied to the problem of recognizing parity—and the very construction described in our section 13.1, through which a Gamba perceptron can recognize parity, is rediscovered. Figure 3 here shows the results. To learn these coefficients, the procedure described in *PDP* required 2,825 cycles through the 16 possible input patterns, thus consuming 45,200 trials for the network to learn to compute the parity predicate for only four inputs. Is this a good result or a bad result? We cannot tell without more knowledge about why the procedure requires so many trials. Until one has some theory of that, there is no

way to assess the significance of any such experimental result; all one can say is that 45,200 = 45,200. In section 10.1 we saw that if a perceptron's φ functions include only masks, the parity predicate requires doubly exponential coefficients. If we were sure that *that* was happening, this would suggest to us that we should represent 45,200 (approximately) as 2^{2^4} rather than, say, as 2^{16}. However, here we suspect that this would be wrong, because the input units aren't masks but predicates—apparently provided from the start—that already know how to "count." These make the problem much easier. In any case, the lesson of *Perceptrons* is that one cannot interpret the meaning of such an experimental report without first making further probes.

Learning

We haven't yet said how those networks learned. The authors of *PDP* describe a learning procedure called the "Generalized Delta Rule"—we'll call it GD—as a new breakthrough in connectionist research. To explain its importance, they depict as follows the theoretical situation they inherited:

"A further argument advanced by Minsky and Papert against perceptron-like models with hidden units is that there was no indication how such multilayer networks were to be trained. One of the appealing features of the one-layer perceptron is the existence of a powerful learning procedure, the perceptron convergence procedure of Rosenblatt. In Minsky and Papert's day, there was no such powerful learning procedure for the more complex multilayer systems. This is no longer true. . . . The GD procedure provides a direct generalization of the perceptron learning procedure which can be applied to arbitrary networks with multiple layers and feedback among layers. This procedure can, in principle, learn arbitrary functions including, of course, parity and connectedness." (vol. 1, p. 113)

In Minsky and Papert's day, indeed! In this section we shall explain why, although the GD learning procedure embodies some useful ideas, it does not justify such sweeping claims. But in order to explain why, and to see how the approach in the current wave of connectionism differs from that in *Perceptrons,* we must first examine with some care the relationship between two branches of perceptron theory which could be called "theory of learning" and "theory of representation." To begin with, one might paraphrase

the above quotation as saying that, until recently, connectionism had been paralyzed by the following dilemma:

Perceptrons could learn anything that they could represent, but they were too limited in what they could represent.

Multilayered networks were less limited in what they could represent, but they had no reliable learning procedure.

According to the classical theory of perceptrons, those limitations on representability depend on such issues as whether a given predicate P can be represented as a perceptron defined by a given set Φ on a given retina, whether P is of finite order, whether P can be realized with coefficients of bounded size, whether properties of several representable predicates are inherited by combinations of those predicates, and so forth. All the results in the first half of our book are involved with these sorts of representational issues. Now, when one speaks about "powerful learning procedures," the situation is complicated by the fact that, given enough input units of sufficiently high order, even simple perceptrons can represent—and therefore learn—arbitrary functions. Consequently, it makes no sense to speak about "power" in absolute terms. Such statements must refer to relative measures of sizes and scales.

As for learning, the dependability of Rosenblatt's Perceptron Convergence theorem of section 11.1—let's call it PC for short—is very impressive: If it is possible at all to represent a predicate P as a linear threshold function of a given set of predicates Φ, then the PC procedure will eventually discover some particular set of coefficients that actually represents P. However, this is not, in itself, a sufficient reason to consider PC interesting and important, because that theorem says nothing about the crucial issue of efficiency. PC is not interesting merely because it provides a systematic way to find suitable coefficients. One could always take recourse, instead, to simple, brute-force search—because, given that some solution exists, one could simply search through all possible integer coefficient vectors, in order of increasing magnitude, until no further "errors" occurred. But no one would consider such an exhaustive process to be an interesting foundation for a learning theory.

What, then, makes PC seem significant? That it discovers those coefficients in ways that are intriguing in several other important respects. The PC procedure seems to satisfy many of the intuitive requirements of those who are concerned with modeling what really happens in a biological nervous system. It also appeals to both our engineering aesthetic and our psychological aesthetic by serving simultaneously as both a form of guidance by error correction and a form of hill-climbing. In terms of computational efficiency, PC seems much more efficient than brute-force procedures (although we have no rigorous and general theory of the conditions under which that will be true). Finally, PC is so simple mathematically as to make one wish to believe that it reflects something real.

Hill-Climbing and the Generalized Delta Procedure

Suppose we want to find the maximum value of a given function $F(x,y,z, \ldots)$ of n variables. The extreme brute-force solution is to calculate the function for all sets of values for the variables and then select the point for which F had the largest value. The approach we called hill-climbing in section 11.3 is a local procedure designed to attempt to find that global maximum. To make this subject more concrete, it is useful to think of the two-dimensional case in which the x–y plane is the ground and $z = F(x,y)$ is the elevation of the point (x,y,z) on the surface of a real physical hill. Now, imagine standing on the hill in a fog so dense that only the immediate vicinity is visible. Then the only resort is to use some diameter-limited local process. The best-known method is the method known as "steepest ascent," discussed in section 11.6: First determine the slope of the surface in various directions from the point where you are standing, then choose the direction that most rapidly increases your altitude and take a step of a certain size in that direction. The hope is that, by thus climbing the slope, you will eventually reach the highest point.

It is both well known and obvious that hill-climbing does not always work. The simplest way to fail is to get stuck on a local maximum—an isolated peak whose altitude is relatively insignificant. There simply is no local way for a hill-climbing procedure to be sure that it has reached a global maximum rather than some local feature of topography (such as a peak, a ridge, or a plain) on which it may get trapped. We showed in section 11.6 that PC is equivalent (in a peculiar sense) to a hill-climbing procedure that works its way to the top of a hill whose geometry can actually

be proved not to have any such troublesome local features—provided that there actually exists some perceptron-weight vector solution A^* to the problem. Thus, one could argue that perceptrons "work" on those problems not because of any particular virtue of the perceptrons or of their hill-climbing procedures but because the hills for those soluble problems have clean topographies. What are the prospects of finding a learning procedure that works equally well on all problems, and not merely on those that have linearly separable decision functions? The authors of *PDP* maintain that they have indeed discovered one:

"Although our learning results do not guarantee that we can find a solution for all solvable problems, our analyses and results have shown that, as a practical matter, the error propagation scheme leads to solutions in virtually every case. In short, we believe that we have answered Minsky and Papert's challenge and have found a learning result sufficiently powerful to demonstrate that their pessimism about learning in multilayer machines was misplaced." (vol. 1, p. 361)

But the experiments in *PDP*, though interesting and ingenious, do not actually demonstrate any such thing. In fact, the "powerful new learning result" is nothing other than a straightforward hill-climbing algorithm, with all the problems that entails. To see how GD works, assume we are given a network of units interconnected by weighted, unidirectional links. Certain of these units are connected to input terminals, and certain others are regarded as output units. We want to teach this network to respond to each (vector) input pattern $\mathbf{X}_p$ with a specified output vector $\mathbf{Y}_p$. How can we find a set of weights $w = \{w_{ij}\}$ that will accomplish this? We could try to do it by hill-climbing on the space of $\mathbf{W}$s, provided that we could define a suitable measure of relative altitude or "success." One problem is that there cannot be any standard, universal way to measure errors, because each type of error has different costs in different situations. But let us set that issue aside and do what scientists often do when they can't think of anything better: sum the squares of the differences. So, if $\mathbf{X}(\mathbf{W},\mathbf{X})$ is the network's output vector for internal weights $\mathbf{W}$ and inputs $\mathbf{X}$, define the altitude function $E(\mathbf{W})$ to be this sum:

$$E(\mathbf{W}) = -\sum_{\substack{\text{all input} \\ \text{patterns } p}} [\mathbf{Y}_p - \mathbf{Y}(\mathbf{W}, \mathbf{X}_p)]^2.$$

In other words, we compute our measure of success by presenting successively each stimulus $\mathbf{X}_p$ to the network. Then we compute the (vector) difference between the actual output and the desired output. Finally, we add up the squares of the magnitudes of those differences. (The minus sign is simply for thinking of climbing up instead of down.) The error function E will then have a maximum possible value of zero, which will be achieved if and only if the machine performs perfectly. Otherwise there will be at least one error and $E(\mathbf{W})$ will be negative. Then all we have to is climb the hill $E(\mathbf{W})$ defined over the (high-dimensional) space of weight vectors $\mathbf{W}$. If our paths reaches a $\mathbf{W}$ for which $E(\mathbf{W})$ is zero, our problem will be solved and we will be able to say that our machine has "learned from its experience."

We'll use a process that climbs this hill by the method of steepest ascent. We can do this by estimating, at every step, the partial derivatives $\partial E/\partial w_{ij}$ of the total error with respect to each component of the weight vector. This tells us the direction of the gradient vector $dE/d\mathbf{W}$, and we then proceed to move a certain distance in that direction. This is the mathematical character of the Generalized Delta procedure, and it differs in no significant way from older forms of diameter-limited gradient followers.

Before such a procedure can be employed, there is an obstacle to overcome. One cannot directly apply the method of gradient ascent to networks that contain threshold units. This is because the derivative of a step-function is zero, whenever it exists, and hence no gradient is defined. To get around this, *PDP* applies a smoothing function to make those threshold functions differentiable. The trick is to replace the threshold function for each unit with a monotonic and differentiable function of the sum of that unit's inputs. This permits the output of each unit to encode information about the sum of its inputs while still retaining an approximation to the perceptron's decision-making ability. Then gradient ascent becomes more feasible. However, we suspect that this smoothing trick may entail a large (and avoidable) cost when the predicate to be learned is actually a composition of linear threshold functions. There ought to be a more efficient alternative based on how much each weight must be changed, for each stimulus, to make the local input sum cross the threshold.

In what sense is the particular hill-climbing procedure GD more powerful than the perceptron's PC? Certainly GD can be applied to more networks than PC can, because PC can operate only on the connections between one layer of φ units and a single output unit. GD, however, can modify the weights in an arbitrary multilayered network, including nets containing loops. Thus, in contrast to the perceptron (which is equipped with some fixed set of φs that can never be changed), GP can be regarded as able to change the weights inside the φs. Thus GD promises, in effect, to be able discover useful *new* φ functions—and many of the experiments reported in *PDP* demonstrate that this often works.

A natural way to estimate the gradient of $E(W)$ is to estimate $\partial E/\partial w_{ij}$ by running through the entire set of inputs for each weight. However, for large networks and large problems that could be a horrendous computation. Fortunately, in a highly connected network, all those many components of the gradient are not independent of one another, but are constrained by the algebraic "chain rule" for the derivatives of composite functions. One can exploit those constraints to reduce the amount of computation by applying the chain-rule formula, recursively, to the mathematical description of the network. This recursive computation is called "back-propagation" in *PDP*. It can substantially reduce the amount of calculation for each hill-climbing step in networks with many connections. We have the impression that many people in the connectionist community do not understand that this is merely a particular way to compute a gradient and have assumed instead that back-propagation is a new learning scheme that somehow gets around the basic limitations of hill-climbing.

Clearly GD would be far more valuable than PC if it could be made to be both efficient and dependable. But virtually nothing has been proved about the range of problems upon which GD works both efficiently and dependably. Indeed, GD can fail to find a solution when one exists, so in that narrow sense it could be considered *less* powerful than PC.

In the early years of cybernetics, everyone understood that hill-climbing was always available for working easy problems, but that it almost always became impractical for problems of larger sizes

and complexities. We were very pleased to discover (see section 11.6) that PC could be represented as hill-climbing; however, that very fact led us to wonder whether such procedures could *dependably* be generalized, even to the limited class of multilayer machines that we named Gamba perceptrons. The situation seems not to have changed much—we have seen no contemporary connectionist publication that casts much new theoretical light on the situation. Then why has GD become so popular in recent years? In part this is because it is so widely applicable, and because it does indeed yield new results (at least on problems of rather small scale). Its reputation also gains, we think, from its being presented in forms that share, albeit to a lesser degree, the biological plausibility of PC. But we fear that its reputation also stems from unfamiliarity with the manner in which hill-climbing methods deteriorate when confronted with larger-scale problems.

In any case, little good can come from statements like "as a practical matter, GD leads to solutions in virtually every case" or "GD can, in principle, learn arbitrary functions." Such pronouncements are not merely technically wrong; more significantly, the pretense that problems do not exist can deflect us from valuable insights that could come from examining things more carefully. As the field of connectionism becomes more mature, the quest for a general solution to all learning problems will evolve into an understanding of which types of learning processes are likely to work on which classes of problems. And this means that, past a certain point, we won't be able to get by with vacuous generalities about hill-climbing. We will really need to know a great deal more about the nature of those surfaces for each specific realm of problems that we want to solve.

On the positive side, we applaud those who bravely and romantically are empirically applying hill-climbing methods to many new domains for the first time, and we expect such work to result in important advances. Certainly these researchers are exploring networks with architectures far more complex than those of perceptrons, and some of their experiments already have shown indications of new phenomena that are well worth trying to understand.

Scaling Problems Up in Size

Experiments with toy-scale problems have proved as fruitful in artificial intelligence as in other areas of science and engineering.

Many techniques and principles that ultimately found real applications were discovered and honed in microworlds small enough to comprehend yet rich enough to challenge our thinking. But not every phenomenon encountered in dealing with small models can be usefully scaled up. Looking at the relative thickness of the legs of an ant and an elephant reminds us that physical structures do not always scale linearly: an ant magnified a thousand times would collapse under its own weight. Much of the theory of computational complexity is concerned with questions of scale. If it takes 100 steps to solve a certain kind of equation with four terms, how many steps will it take to solve the same kind of equation with eight terms? Only 200, if the problem scales linearly. But for other problems it will take not twice 100 but 100 squared.

For example, the Gamba perceptron of figure 2 needs only two φ functions rather than the six required in figure 1. In neither of these two toy-sized networks does the number seem alarmingly large. One network has fewer units; the other has smaller coefficients. But when we examine how those numbers grow with retinas of increasing size, we discover that whereas the coefficients of figure 1 remain constant, those of figure 2 grow exponentially. And, presumably, a similar price must be paid again in the number of repetitions required in order to learn.

In the examination of theories of learning and problem solving, the study of such growths in cost is not merely one more aspect to be taken into account; in a sense, it is the only aspect worth considering. This is because so many problems can be solved "in principle" by exhaustive search through a suitable space of states. Of course, the trouble with that in practice is that there is usually an exponential increase in the number of steps required for an exhaustive search when the scale of the problem is enlarged. Consequently, solving toy problems by methods related to exhaustive search rarely leads to practical solutions to larger problems. For example, though it is easy to make an exhaustive-search machine that never loses a game of noughts and crosses, it is infeasible to do the same for chess. We do not know if this fact is significant, but many of the small examples described in *PDP* could have been solved as quickly by means of exhaustive search—that is, by systematically assigning and testing all combinations of small integer weights.

When we started our research on perceptrons, we had seen many interesting demonstrations of perceptrons solving problems of very small scale but not doing so well when those problems were scaled up. We wondered what was going wrong. Our first "handle" on how to think about scaling came with the concept of the order of a predicate. If a problem is of order N, then the number of φs for the corresponding perceptron need not increase any faster than as the Nth power of R. Then, whenever we could show that a given problem was of low order, we usually could demonstrate that perceptron-like networks could do surprisingly well on that problem. On the other hand, once we developed the more difficult techniques for showing that certain other problems have unbounded order, this raised alarming warning flags about extending *their* solutions to larger domains.

Unbounded order was not the only source of scaling failures. Another source—one we had not anticipated until the later stages of our work—involved the size, or rather the information content, of the coefficients. The information stored in connectionist systems is embodied in the strengths of weights of the connections between units. The idea that learning can take place by changing such strengths has a ring of biological plausibility, but that plausibility fades away if those strengths are to be represented by numbers that must be accurate to ten or twenty decimal orders of significance.

The Problem of Sampling Variance

Our description of the Generalized Delta Rule assumes that it is feasible to compute the new value of $E(\mathbf{W})$ at every step of the climb. The processes discussed in chapter 8 of *PDP* typically require only on the order of 100,000 iterations, a range that is easily accessible to computers (but that might in some cases strain our sense of biological plausibility). However, it will not be practical, with larger problems, to cycle through all possible input patterns. This means that when precise measures of $E(\mathbf{W})$ are unavailable, we will be forced to act, instead, on the basis of incomplete samples—for example, by making a small hill-climbing step after each reaction to a stimulus. (See the discussion of complete versus incremental methods in subsection 12.1.1.) When we can no longer compute $dE/d\mathbf{W}$ precisely but can only estimate its components, then the actual derivative will be masked by a certain amount of

sampling noise. The text of *PDP* argues that using sufficiently small steps can force the resulting trajectory to come arbitrarily close to that which would result from knowing $dE/d\mathbf{W}$ precisely. When we tried to prove this, we were led to suspect that the choice of step size may depend so much on the higher derivatives of the smoothing functions that large-scale problems could require too many steps for such methods to be practical.

So far as we could tell, every experiment described in chapter 8 of *PDP* involved making a complete cycle through all possible input situations before making any change in weights. Whenever this is feasible, it completely eliminates sampling noise—and then even the most minute correlations can become reliably detectable, because the variance is zero. But no person or animal ever faces situations that are so simple and arranged in so orderly a manner as to provide such cycles of teaching examples. Moving from small to large problems will often demand this transition from exhaustive to statistical sampling, and we suspect that in many realistic situations the resulting sampling noise would mask the signal completely. We suspect that many who read the connectionist literature are not aware of this phenomenon, which dims some of the prospects of successfully applying certain learning procedures to large-scale problems.

Problems of Scaling

In principle, connectionist networks offer all the potential of universal computing devices. However, our examples of order and coefficient size suggest that various kinds of scaling problems are likely to become obstacles to attempts to exploit that potential. Fortunately, our analysis of perceptrons does not suggest that connectionist networks need always encounter these obstacles. Indeed, our book is rich in surprising examples of tasks that simple perceptrons can perform using relatively low-order units and small coefficients. However, our analysis does show that parallel networks are, in general, subject to serious scaling phenomena. Consequently, researchers who propose such models must show that, in their context, those phenomena do not occur.

The authors of *PDP* seem disinclined to face such problems. They seem content to argue that, although we showed that single-layer networks cannot solve certain problems, we did not know that

there could exist a powerful learning procedure for multilayer networks—to which our theorems no longer apply. However, strictly speaking, it is wrong to formulate our findings in terms of what perceptrons can and cannot do. As we pointed out above, perceptrons of sufficiently large order can represent *any* finite predicate. A better description of what we did is that, in certain cases, we established the computational costs of what perceptrons can do as a function of increasing problem size. The authors of *PDP* show little concern for such issues, and usually seem content with experiments in which small multilayer networks solve particular instances of small problems.

What should one conclude from such examples? A person who thinks in terms of *can* versus *can't* will be tempted to suppose that if toy machines can do something, then larger machines may well do it better. One must always probe into the practicality of a proposed learning algorithm. It is no use to say that "procedure P is capable of learning to recognize pattern X" unless one can show that this can be done in less time and at less cost than with exhaustive search. Thus, as we noted, in the case of symmetry, the authors of *PDP* actually recognized that the coefficients were growing as powers of 2, yet they did not seem to regard this as suggesting that the experiment worked only because of its very small size. But scientists who exploit the insights gained from studying the single-layer case might draw quite different conclusions.

The authors of *PDP* recognize that GD is a form of hill-climber, but they speak as though becoming trapped on local maxima were rarely a serious problem. In reporting their experiments with learning the XOR predicate, they remark that this occurred "in only two cases . . . in hundreds of times." However, that experiment involved only the toy problem of learning to compute the XOR of two arguments. We conjecture that learning XOR for larger numbers of variables will become increasingly intractable as we increase the numbers of input variables, because by its nature the underlying parity function is absolutely uncorrelated with any function of fewer variables. Therefore, there can exist no useful correlations among the outputs of the lower-order units involved in computing it, and that leads us to suspect that there is little to gain from following whatever paths are indicated by the artificial introduction of smoothing functions that cause partial derivatives to exist.

The *PDP* experimenters encountered a more serious local-maximum problem when trying to make a network learn to add two binary numbers—a problem that contains an embedded XOR problem. When working with certain small networks, the system got stuck reliably. However, the experimenters discovered an interesting way to get around this difficulty by introducing longer chains of intermediate units. We encourage the reader to study the discussion starting on page 341 of *PDP* and try to make a more complete theoretical analysis of this problem. We suspect that further study of this case will show that hill-climbing procedures can indeed get multilayer networks to learn to do multidigit addition. However, such a study should be carried out not to show that "networks are good" but to see which network architectures are most suitable for enabling the information required for "carrying" to flow easily from the smaller to the larger digits. In the *PDP* experiment, the network appears to us to have started on the road toward inventing the technique known to computer engineers as "carry jumping."

To what extent can hill-climbing systems be made to solve hard problems? One might object that this is a wrong question because "hard" is so ill defined. The lesson of *Perceptrons* is that we must find ways to make such questions meaningful. In the case of hill-climbing, we need to find ways to characterize the types of problems that lead to the various obstacles to climbing hills, instead of ignoring those difficulties or trying to find universal ways to get around them.

The Society of Mind

The preceding section was written as though it ought to be the principal goal of research on network models to determine in which situations it will be feasible to scale their operations up to deal with increasingly complicated problems. But now we propose a somewhat shocking alternative: Perhaps the scale of the toy problem is that on which, in physiological actuality, much of the functioning of intelligence operates. Accepting this thesis leads into a way of thinking very different from that of the connectionist movement. We have used the phrase "society of mind" to refer to the idea that mind is made up of a large number of components, or "agents," each of which would operate on the scale of what, if taken in

isolation, would be little more than a toy problem. [See Marvin Minsky, *The Society of Mind* (Simon and Schuster, 1987) and Seymour Papert, *Mindstorms* (Basic Books, 1982).]

To illustrate this idea, let's try to compare the performance of the symmetry perceptron in *PDP* with human behavior. An adult human can usually recognize and appreciate the symmetries of a kaleidoscope, and that sort of example leads one to imagine that people do very much better than simple perceptrons. But how much can people actually do? Most people would be hard put to be certain about the symmetry of a large pattern. For example, how long does it take you to decide whether or not the following pattern is symmetrical?

DB4HWUK85HCNZEWJKRKJWEZNCH58KUWH4BD

In many situations, humans clearly show abilities far in excess of what could be learned by simple, uniform networks. But when we take those skills apart, or try to find out how they were learned, we expect to find that they were made by processes that somehow combined the work (already done in the past) of many smaller agencies, none of which, separately, need to work on scales much larger than do those in *PDP*. Is this hypothesis consistent with the *PDP* style of connectionism? Yes, insofar as the computations of the nervous system can be represented as the operation of societies of networks. But no, insofar as the mode of operation of those societies of networks (as we imagine them) raises theoretical issues of a different kind. We do not expect procedures such as GD to be able to produce such societies. Something else is needed.

What that something must be depends on how we try to extend the range of small connectionist models. We see two principal alternatives. We could extend them either by scaling up small connectionist models or by combining small-scale networks into some larger organization. In the first case, we would expect to encounter theoretical obstacles to maintaining GD's effectiveness on larger, deeper nets. And despite the reputed efficacy of other alleged remedies for the deficiencies of hill-climbing, such as "annealing," we stay with our research conjecture that no such procedures will work very well on large-scale nets, except in the case of problems that turn out to be of low order in some appropriate sense. The

second alternative is to employ a variety of smaller networks rather than try to scale up a single one. And if we choose (as we do) to move in that direction, then our focus of concern as theoretical psychologists must turn toward the organizing of small nets into effective large systems. The idea that the lowest levels of thinking and learning may operate on toy-like scales fits many of our common-sense impressions of psychology. For example, in the realm of language, any normal person can parse a great many kinds of sentences, but none of them past a certain bound of involuted complexity. We all fall down on expressions like "the cheese that the rat that the cat that the dog bit chased ate." In the realm of vision, no one can count great numbers of things, in parallel, at a single glance. Instead, we learn to "estimate." Indeed, the visual joke in figure 0.1 shows clearly how humans share perceptrons' inability to easily count and match, and a similar example is embodied in the twin spirals of figure 5.1. The spiral example was intended to emphasize not only that low-order perceptrons cannot perceive connectedness but also that humans have similar limitations. However, a determined person can solve the problem, given enough time, by switching to the use of certain sorts of serial mental processes.

Beyond Perceptrons

No single-method learning scheme can operate efficiently for every possible task; we cannot expect any one type of machine to account for any large portion of human psychology. For example, in certain situations it is best to carefully accumulate experience; however, when time is limited, it is necessary to make hasty generalizations and act accordingly. No single scheme can do all things. Our human semblance of intelligence emerged from how the brain evolved a multiplicity of ways to deal with different problem realms. We see this as a principle that underlies the mind's reality, and we interpret the need for many kinds of mechanisms not as a pessimistic and scientifically constraining limitation but as the fundamental source of many of the phenomena that artificial intelligence and psychology have always sought to understand. The power of the brain stems not from any single, fixed, universal principle. Instead it comes from the evolution (in both the individual sense and the Darwinian sense) of a variety of ways to develop new mechanisms and to adapt older ones to perform new functions. Instead of seeking a way to get around that need for diversity, we

have come to try to develop "society of mind" theories that will recognize and exploit the idea that brains are based on many different kinds of interacting mechanisms.

Several kinds of evidence impel us toward this view. One is the great variety of different and specific functions embodied in the brain's biology. Another is the similarly great variety of phenomena in the psychology of intelligence. And from a much more abstract viewpoint, we cannot help but be impressed with the practical limitations of each "general" scheme that has been proposed—and with the theoretical opacity of questions about how they behave when we try to scale their applications past the toy problems for which they were first conceived.

Our research on perceptrons and on other computational schemes has left us with a pervasive bias against seeking a general, domain-independent theory of "how neural networks work." Instead, we ought to look for ways in which particular types of network models can support the development of models of particular domains of mental function—and vice versa. Thus, our understanding of the perceptron's ability to perform geometric tasks was actually based on theories that were more concerned with geometry than with networks. And this example is supported by a broad body of experience in other areas of artificial intelligence. Perhaps this is why the current preoccupation of connectionist theorists with the search for general learning algorithms evokes for us two aspects of the early history of computation.

First, we are reminded of the long line of theoretical work that culminated in the "pessimistic" theories of Gödel and Turing about the limitations on effective computability. Yet the realization that there can be no general-purpose decision procedure for mathematics had not the slightest dampening effect on research in mathematics or in computer science. On the contrary, awareness of those limiting discoveries helped motivate the growth of rich cultures involved with classifying and understanding more specialized algorithmic methods. In other words, it was the realization that seeking overgeneral solution methods would be as fruitless as—and equivalent to—trying to solve the unsolvable halting problem for Turing machines. Abandoning this then led to seeking progress in more productive directions.

Our second thought is about how the early research in artificial intelligence tended to focus on general-purpose algorithms for reasoning and problem solving. Those general methods will always play their roles, but the most successful applications of AI research gained much of their practical power from applying specific knowledge to specific domains. Perhaps that work has now moved too far toward ignoring general theoretical considerations, but by now we have learned to be skeptical about the practical power of unrestrained generality.

Interaction and Insulation

Evolution seems to have anticipated these discoveries. Although the nervous system appears to be a network, it is very far from being a single, uniform, highly interconnected assembly of units that each have similar relationships to the others. Nor are all brain cells similarly affected by the same processes. It would be better to think of the brain not as a single network whose elements operate in accord with a uniform set of principles but as a network whose components are themselves networks having a large variety of different architectures and control systems. This "society of mind" idea has led our research perspective away from the search for algorithms, such as GD, that were hoped to work across many domains. Instead, we were led into trying to understand what specific kinds of processing would serve specific domains.

We recognize that the idea of distributed, cooperative processing has a powerful appeal to common sense as well to computational and biological science. Our research instincts tell us to discover as much as we can about distributed processes. But there is another concept, complementary to distribution, that is no less strongly supported by the same sources of intuition. We'll call it *insulation*.

Certain parallel computations are by their nature synergistic and cooperative: each part makes the others easier. But the And/Or of theorem 4.0 shows that under other circumstances, attempting to make the same network perform two simple tasks at the same time leads to a task that has a far greater order of difficulty. In those sorts of circumstances, there will be a clear advantage to having mechanisms, not to connect things together, but to keep such tasks apart. How can this be done in a connectionist net? Some recent work hints that even simple multilayer perceptron-like nets can

learn to segregate themselves into quasi-separate components—and that suggests (at least in principle) research on uniform learning procedures. But it also raises the question of how to relate those almost separate parts. In fact, research on networks in which different parts do different things and learn those things in different ways has become our principal concern. And that leads us to ask how such systems could develop *managers* for deciding, in different circumstances, which of those diverse procedures to use.

For example, consider all the specialized agencies that the human brain employs to deal with the visual perception of spatial scenes. Although we still know little about how all those different agencies work, the end result is surely even more complex than what we described in section 13.4. Beyond that, human scene analysis also engages our memories and goals. Furthermore, in addition to all the systems we humans use to dissect two-dimensional scenes into objects and relationships, we also possess machinery for exploiting stereoscopic vision. Indeed, there appear to be many such agencies—distinct ones that employ, for example, motion cues, disparities, central correlations of the Julesz type, and memory-based frame-array-like systems that enable us to imagine and virtually "see" the occluded sides of familiar objects. Beyond those, we seem also to have been supplied with many other visual agencies—for example, ones that are destined to learn to recognize faces and expressions, visual cliffs, threatening movements, sexual attractants, and who knows how many others that have not been discovered yet. What mechanisms manage and control the use of all those diverse agencies? And from where do those managers come?

Stages of Development

In *Mindstorms* and in *The Society of Mind,* we explained how the idea of intermediate, hidden processes might well account for some phenomena discovered by Piaget in his experiments on how children develop their concepts about the "conservation of quantity." We introduced a theory of mental growth based on inserting, at various times, new inner layers of "management" into already existing networks. In particular, we argued that, to learn to make certain types of comparisons, a child's mind must construct a multilayer structure that we call a "society-of-more." The lower levels of that net contain agents specialized to make a variety of spatial

and temporal observations. Then the higher-level agents learn to classify, and then control, the activities of the lower ones. We certainly would like to see a demonstration of a learning process that could spontaneously produce the several levels of agents needed to embody a concept as complex as that. Chapter 17 of *The Society of Mind* offers several different reasons why this might be very difficult to do except in systems under systematic controls, both temporal and architectural. We suspect that it would require far too long, in comparison with an infant's months of life, to create sophisticated agencies entirely by undirected, spontaneous learning. Each specialized network must begin with promising ingredients that come either from prior stages of development or from some structural endowment that emerged in the course of organic evolution.

When should new layers of control be introduced? If managers are empowered too soon, when their workers still are too immature, they won't be able to accomplish enough. (If every agent could learn from birth, they would all be overwhelmed by infantile ideas.) But if the managers arrive too late, that will retard all further growth. Ideally, every agency's development would be controlled by yet another agency equipped to introduce new agents just when they are needed—that is, when enough has been learned to justify the start of another stage. However, that would require a good deal of expertise on the controlling agency's part. Another way—much easier to evolve—would simply enable various agencies to establish new connections at genetically predetermined times (perhaps while also causing lower-level parts to slow further growth). Such a scheme could benefit a human population on the whole, although it might handicap individuals who, for one reason or another, happen to move ahead of or behind that inborn "schedule." In any case, there are many reasons to suspect that the parts of any system as complex as a human mind must grow through sequences of stage-like episodes.

Architecture and Specialization

The tradition of connectionism has always tried to establish two claims: that connectionist networks can accomplish interesting tasks and that they can learn to do so with no explicit programming. But a closer look reveals that rarely are those two virtues

present in the same device. It is true that networks, taken as a class, can do virtually anything. However, each particular type of network can best learn only certain types of things. Each particular network we have seen seems relatively limited. Yet our wondrous brains are themselves composed of connected networks of cells.

We think that the difference in abilities comes from the fact that a brain is not a single, uniformly structured network. Instead, each brain contains hundreds of different types of machines, interconnected in specific ways which predestine that brain to become a large, diverse society of partially specialized agencies. We are born with specific parts of our brains to serve every sense and muscle group, and with perhaps separate sections for physical and social matters (e.g., natural sounds versus social speech, inanimate scenes versus facial expressions, mechanical contacts versus social caresses). Our brains also embody proto-specialists involved with hunger, laughter, anger, fear, and perhaps hundreds of other functions that scientists have not yet isolated. Many thousands of genes must be involved in constructing specific internal architectures for each of those highly evolved brain centers and in laying out the nerve bundles that interconnect them. And although each such system is embodied in the form of a network-based learning system, each almost surely also learns in accord with somewhat different principles.

Why did our brains evolve so as to contain so many specialized parts? Could not a single, uniform network learn to structure itself into divisions with appropriate architectures and processes? We think that this would be impractical because of the problem of representing knowledge. In order for a machine to learn to recognize or perform X, be it a pattern or a process, that machine must in one sense or another learn to represent or embody X. Doing that efficiently must exploit some happy triadic relationship between the structure of X, the learning procedure, and the initial architecture of the network. It makes no sense to seek the "best" network architecture or learning procedure because it makes no sense to say that *any* network is efficient by itself: that makes sense only in the context of some class of problems to be solved. Different kinds of networks lend themselves best to different kinds of representations and to different sorts of generalizations. This means that the study of networks in general must include attempts, like those in

this book, to classify problems and learning processes; but it must also include attempts to classify the network architectures. This is why we maintain that the scientific future of connectionism is tied not to the search for some single, universal scheme to solve all problems at once but to the evolution of a many-faceted technology of "brain design" that encompasses good technical theories about the analysis of learning procedures, of useful architectures, and of organizational principles to use when assembling those components into larger systems.

Symbolic versus Distributed

Let us now return to the conflict posed in our prologue: the war between the connectionists and the symbolists. We hope to make peace by exploiting both sides.

There are important virtues in the use of parallel distributed networks. They certainly often offer advantages in simplicity and in speed. And above all else they offer us ways to learn new skills without the pain and suffering that might come from comprehending how. On the darker side, they can limit large-scale growth because what any distributed network learns is likely to be quite opaque to other networks connected to it.

Symbolic systems yield gains of their own, in versatility and unlimited growth. Above all else they offer us the prospect that computers share: of not being bound by the small details of the parts of which they are composed. But that, too, has its darker side: symbolic processes can evolve worlds of their own, utterly divorced from their origins. Perceptrons can never go insane—but the same cannot be said of a brain.

Now, what are symbols, anyway? We usually conceive of them as compact things that represent more complex things. But what, then, do we mean by *represent*? It simply makes no sense, by itself, to say that "S represents T," because the significance of a symbol depends on at least three participants: on S, on T, and on the context of some process or user U. What, for example, connects the word *table* to any actual, physical table? Since the words people use are the words people learn, clearly the answer must be that there is no direct relationship between S and T, but that there is a more complex triadic relationship that connects a symbol, a thing,

and a process that is active in some person's mind. Furthermore, when the term *symbol* is used in the context of network psychology, it usually refers to something that is reassignable so that it can be made to represent different things and so that the symbol-using processes can learn to deal with different symbols.

What do we mean by *distributed*? This usually refers to a system in which each end-effect comes not from any single, localized element-part, but from the interactions of many contributors, all working at the same time. Accordingly, in order to make a desired change in the output of a distributed system, one must usually alter a great many components. And changing the output of any particular component will rarely have a large effect in any particular circumstance; instead, such changes will tend to have small effects in many different circumstances.

Symbols are tokens or handles with which one specialist can manipulate representations within another specialist. But now, suppose that we want one agency to be able to exploit the knowledge in another agency. So long as we stay inside a particular agency, it may be feasible to use representations that involve great hosts of internal interactions and dependencies. But the fine details of such a representation would be meaningless to any outside agency that lacks access to, or the capacity to deal with, all that fine detail. Indeed, if each representation in the first agency involves activities that are uniformly distributed over a very large network, then direct communication to the other agency would require so many connection paths that both agencies would end up enmeshed together into a single, uniform net—and then all the units of both would interact.

How, then, could networks support symbolic forms of activities? We conjecture that, inside the brain, agencies with different jobs are usually constrained to communicate with one another only through neurological bottlenecks (i.e., connections between relatively small numbers of units that are specialized to serve as symbolic recognizers and memorizers). The recognizers learn to encode significant features of the representation active in the first network, and the memorizers learn to evoke an activity that can serve a corresponding function in the receiving network. But in order to prevent those features from interfering too much with one

another, there must be an adequate degree of insulation between the units that serve these purposes. And that need for insulation can lead to genuine conflicts between the use of symbolic and distributed representations. This is because distributed representations make it hard to combine (in arbitrary, learnable ways) the different fragments of knowledge embodied in different representations. The difficulty arises because the more distributed is the representation of each fragment, the fewer fragments can be simultaneously active without interfering with one another. Sometimes those interactions can be useful, but in general they will be destructive. This is discussed briefly in section 8.2 of *The Society of Mind:*

"The advantages of distributed systems are not alternatives to the advantages of insulated systems: the two are complementary. To say that the brain may be composed of distributed systems is not the same as saying that it *is* a distributed system—that is, a single network in which all functions are uniformly distributed. We do not believe that any brain of that sort could work, because the interactions would be uncontrollable. To be sure, we have to explain how different ideas can become connected to one another—but we must also explain what keeps our separate memories intact. For example, we praised the power of metaphors that allow us to mix the ideas we have in different realms—but all that power would be lost if all our metaphors got mixed! Similarly, the architecture of a mind-society must encourage the formation and maintenance of distinct levels of management by preventing the formation of connections between agencies whose messages have no mutual significance. Some theorists have assumed that distributed systems are inherently both robust and versatile but, actually, those attributes are more likely to conflict. Systems with too many interactions of different types will tend to be fragile, while systems with too many interactions of similar types will tend to be too redundant to adapt to novel situations and requirements."

A larger-scale problem is that the use of widely distributed representations will tend to oppose the formulation of knowledge about knowledge. This is because information embodied in distributed form will tend to be relatively inaccessible for use as a subject upon which other knowledge-based processes can operate. Consequently (we conjecture), systems that use highly distributed representations will tend to become conceptual dead ends as a result of their putting performance so far ahead of comprehension as to

retard the growth of reflective thought. Too much diffusing of information can make it virtually impossible (for other portions of the brain) to find out how results, however useful, are obtained. This would make it very difficult to dissect out the components that might otherwise be used to construct meaningful variations and generalizations. Of course such problems won't become evident in experiments with systems that do only simple things, but we can expect to see such problems grow when systems try to learn to do more complex things. With highly distributed systems, we should anticipate that the accumulation of internal interactions may eventually lead to intractable credit-assignment problems. Perhaps the only ultimate escape from the limitations of internal interactions is to evolve toward organizations in which each network affects others primarily through the use of *serial* operations and specialized short-term-memory systems, for although seriality is relatively slow, its uses makes it possible to produce and control interactions between activities that occur at different and separate places and times.

The Parallel Paradox

It is often argued that the use of distributed representations enables a system to exploit the advantages of parallel processing. But what *are* the advantages of parallel processing? Suppose that a certain task involves two unrelated parts. To deal with both concurrently, we would have to maintain their representations in two decoupled agencies, both active at the same time. Then, should either of those agencies become involved with two or more subtasks, we would have to deal with each of them with no more than a quarter of the available resources. If that proceeded on and on, the system would become so fragmented that each job would end up with virtually no resources assigned to it. In this regard, distribution may oppose parallelism: the more distributed a system is—that is, the more intimately its parts interact—the fewer *different* things it can do at the same time. On the other side, the more we do *separately* in parallel, the less machinery can be assigned to each element of what we do, and that ultimately leads to increasing fragmentation and incompetence.

This is not to say that distributed representations and parallel processing are always incompatible. When we simultaneously activate

two distributed representations in the same network, they will be forced to interact. In favorable circumstances, those interactions can lead to useful parallel computations, such as the satisfaction of simultaneous constraints. But that will not happen in general; it will occur only when the representations happen to mesh in suitably fortunate ways. Such problems will be especially serious when we try to train distributed systems to deal with problems that require any sort of structural analysis in which the system must represent relationships between substructures of related types—that is, problems that are likely to compete for the same limited resources.

On the positive side, there are potential virtues to embodying knowledge in the form of networks of units with weighted interconnections. For example, distributed representations can sometimes be used to gain the robustness of redundancy, to make machines that continue to work despite having injured, damaged, or unreliable components. They can embody extremely simple learning algorithms, which operate in parallel with great speed.

Representations and Generalizations

It is often said that distributed representations are inherently possessed of useful holistic qualities; for example, that they have innate tendencies to recognize wholes from partial cues—even for patterns they have not encountered before. Phenomena of that sort are often described with such words as *generalization, induction,* or *gestalt.* Such phenomena certainly *can* emerge from connectionist assemblies. The problem is that, for any body of experience, there are always *many* kinds of generalizations that can be made. The ones made by any particular network are likely to be inappropriate unless there happens to be an appropriate relationship between the network's architecture and the manner in which the problem is represented. What makes architectures and representations appropriate? One way to answer that is to study how they affect which signals will be treated as similar.

Consider the problem of comparing an arbitrary input pattern with a collection of patterns in memory, to find which memory is most similar to that stimulus. In section 12.7 we conjectured that solving best-match problems will always be very tedious when serial hard-

ware is used. *PDP* suggests another view in regard to parallel, distributed machines: "This is precisely the kind of problem that is readily implemented using highly parallel algorithms of the kind we consider." This is, in some ways, plausible, since a sufficiently parallel machine could simultaneously match an input pattern against every pattern in its memory. And yet the assertion is quaintly naive, since *best match* means different things in different circumstances. Which answers should be accepted as best always depends on the domain of application. The very same stimulus may signify food to one animal, companionship to another, and a dangerous predator to a third. Thus, there can be no single, universal measure of how well two descriptions match; every context requires appropriate schemes. Because of this, distributed networks do not magically provide solutions to such best-match problems. Instead, the functional architecture of each particular network imposes its own particular sort of metrical structure on the space of stimuli. Such structures may often be useful. Yet, that can give us no assurance that the outcome will correspond to what an expert observer would consider to be the very best match, given that observer's view of what would be the most appropriate response in the current context or problem realm.

We certainly do not mean to suggest that networks cannot perform useful matching functions. We merely mean to emphasize that different problems entail different matching criteria, and that hence no particular type of network can induce a topology of similarity or nearness that is appropriate for every realm. Instead, we must assume that, over the course of time, each specialized portion of the brain has evolved a particular type of architecture that is reasonably likely to induce similarity relationships that are useful in performing the functions to which that organ is likely (or destined) to be assigned. Perhaps an important activity of future connectionist research will be to develop networks that can *learn* to embody wide ranges of different, context-dependent types of matching functions.

We have also often heard the view that machines that employ localized or symbolic representations must be inherently less capable than are distributed machines of insight, consciousness, or sense of self. We think this stands things on their heads. It is *because* our brains primarily exploit connectionist schemes that

we possess such *small* degrees of consciousness, in the sense that we have so little insight into the nature of our own conceptual machinery. We agree that distributed representations probably are used in virtually every part of the brain. Consequently, each agency must learn to exploit the abilities of the others without having direct access to compact representations of what happens inside those other agencies. This makes direct insight infeasible; the best such agencies can do is attempt to construct their own models of the others on the basis of approximate, pragmatic models based on presuppositions and concepts already embodied in the observing agency. Because of this, what appear to us to be direct insights into ourselves must be rarely genuine and usually conjectural. Accordingly, we expect distributed representations to tend to produce systems with only limited abilities to reflect accurately on how they do what they do. Thinking about thinking, we maintain, requires the use of representations that are localized enough that they can be dissected and rearranged. Besides, distributed representations spread out the information that goes into them. The result of this is to mix and obscure the effects of their separate elements. Thus their use must entail a heavy price; surely, many of them must become "conceptual dead ends" because the performances that they produce emerge from processes that other agencies cannot comprehend. In other words, when the representations of concepts are distributed, this will tend to frustrate attempts of other agencies to adapt and transfer those concepts to other contexts.

How much, then, can we expect from connectionist systems? Much more than the above remarks might suggest, since reflective thought is the lesser part of what our minds do. Most probably, we think, the human brain is, in the main, composed of large numbers of relatively small distributed systems, arranged by embryology into a complex society that is controlled in part (but only in part) by serial, symbolic systems that are added later. But the subsymbolic systems that do most of the work from underneath must, by their very character, block all the other parts of the brain from knowing much about how they work. And this, itself, could help explain how people do so many things yet have such incomplete ideas of how those things are actually done.

The following remarks are intended to introduce the literature of this field. This is not to be considered an attempt at historical scholarship, for we have made no serious venture in that direction.

In a decade of work on the family of machines loosely called perceptrons, we find an interacting evolution and refinement of two ideas: first, the concept of realizing a predicate as a linear threshold function of much more local predicates; second, the idea of a convergence or "learning" theorem. The most commonly held version of this history sees the perceptron invented by Rosenblatt in a single act, with the final proof of the convergence theorem vindicating his insight in the face of skepticism from the scientific world. This is an oversimplification, especially in its taking the concept of perceptron as static. For in fact a key part of the process leading to the convergence theorem was the molding of the concept of the machine to the appropriate form. (Indeed, how often does "finding the proof" of a conjecture involve giving the conjecture a more provable form?)

In the early papers one sees a variety, both of machines and of "training" procedures, converging in the course of accumulation of mathematical insight toward the simple concepts we have used in this book. Students interested in this evolution can read:

Rosenblatt, Frank (1959), "Two theorems of statistical separability in the perceptron," *Proceedings of a Symposium on the Mechanization of Thought Processes*, Her Majesty's Stationary Office, London, pp. 421–456;

Rosenblatt, Frank (1962), *Principles of Neurodynamics*, Spartan Books, New York.

In a variety of contexts, other perceptronlike learning experiments had been described. Quite well-known was the paper of

Samuel, Arthur L. (1959), "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, Vol. 3, No. 3, pp. 210–223

who describes a variety of error-correcting vector addition procedures. In a later paper

Samuel, Arthur L. (1967), "Some studies in machine learning using the game of checkers, Part II," *IBM Journal of Research and Development*, Vol. 11, No. 4, pp. 601–618

he describes studies that lead toward detecting more complex

interactions between the partial predicates. The simple multilayer perceptronlike machines discussed in Chapter 13 were described in

Palmieri, G. and R. Sanna (1960), *Methodos*, Vol. 12, No. 48;

Gamba, A., L. Gamberini, G. Palmieri, and R. Sanna (1961), "Further experiments with PAPA," *Nuovo Cimento Suppl.* No. 2, Vol. 20, pp. 221–231.

Some earlier reward-modified machines, further from the final form of the perceptron, are described in

Ashby, W. Ross (1952), *Design for a Brain*, Wiley, New York;

Clark, Wesley A., and Farley, B. G. (1955), "Generalization of pattern-recognition in a self-organizing system," *Proceedings 1955 Western Joint Computer Conference*, pp. 85–111;

Minsky, M. (1954), "Neural nets and the brain-model problem," doctoral dissertation, Princeton University, Princeton, N.J.;

Uttley, A. M. (1956), "Conditional probability machines," in *Automata Studies*, Princeton University, Princeton, N.J., pp. 253–285.

The proof of the convergence theorem (Theorem 11.1) is another example of this sort of evolution. In an abstract mathematical sense, both theorem and proof already existed before the perceptron, for several people had considered the idea of solving a set of linear inequalities by "relaxation" methods—successive adjustments much like those used in the perceptron proceduce. An elegant early paper on this is

Agmon, S. (1954), "The relaxation method for linear inequalities," *Canadian Journal of Mathematics*, Vol. 6, No. 3, pp. 382–392.

In Agmon's procedure, one computes the Φ-vector that gives the largest numerical error in the satisfaction of the linear inequality, and uses a multiple of that vector for correction. (See §11.4.) We do not feel sufficiently scholarly to offer an opinion on whether this paper should deserve priority for the discovery of the convergence theorem. It is ~~quite clear~~ that the theorem would have been instantly obvious had the cyberneticists interested in perceptrons known about Agmon's work.

possible

In any case, the first proofs of the convergence theorem offered in cybernetic circles were quite independent of the work on linear inequalities. See, for example

Block, H. D. (1962), "The perceptron: a model for brain functioning," *Reviews of Modern Physics*, Vol. 34, No. 1, pp. 123–135.

This proof was quite complicated. The first use known to us of the simpler kind of analysis used in §11.1 is in

Papert, Seymour (1961), "Some Mathematical Models of Learning," *Proceedings of the Fourth London Symposium on Information Theory*, C. Cherry, Editor, Academic Press, New York.

Curiously, this paper is not mentioned by any later commentators (including the usually scholarly Nilsson) other than Rosenblatt in *Neurodynamics*. The convergence theorem is well discussed, in a variety of settings, by

Nilsson, Nils (1965), *Learning Machines*, McGraw-Hill, New York,

who includes a number of historical notes. Readers who consult the London Symposium volume might also read

Minsky, Marvin, and Oliver G. Selfridge (1961), "Learning in neural nets," *Proceedings of the Fourth London Symposium on Information Theory*, C. Cherry, Editor, Academic Press, New York.

for some discussion of the relations between convergence and hill-climbing. Although Minsky and Papert did not yet know one another, their papers in that volume overlap to the extent of proving the same theorem about the Bayesian optimality of linear separation. This coincidence had no obvious connection with their later collaboration.

As Agmon had clearly anticipated the learning aspect of the perceptron, so had Selfridge anticipated its quality of combining local properties to yield apparently global ones. This is seen, for example, in

Selfridge, Oliver G. (1956), "Pattern recognition and learning," *Proceedings of the Third London Symposium of Information Theory*, Academic Press, New York, p. 345.

Incidentally, we consider that there has been a strong influence of these cybernetic ideas on the trend of ideas and discoveries in the physiology of vision represented, for example, in

Lettvin, Jerome Y., H. Maturana, W. S. McCulloch, and W. Pitts (1959), "What the frog's eye tells the frog's brain," *Proceedings of the IRE*, Vol. 47, pp. 1940–1951

and

Hubel, D. H., and T. N. Wiesel (1959), "Receptive fields of single neurons in the cat's striate cortex," *Journal of Physiology*, Vol. 148, pp. 574–591.

Other ideas used in this book come from earlier models of physiological phenomena, notably the paper of

Pitts, W., and W. S. McCulloch (1947), "How we know universals," *Bulletin of Mathematical Biophysics*, Vol. 9, pp. 127–147

which is the first we know of that treats recognition invariant under a group by integrating or summing predicates over the group. This paper and that of Lettvin *et al.* are reprinted in

McCulloch, Warren S. (1965), *Embodiments of Mind*, The M.I.T. Press, Cambridge, Mass.,

and this book reprints other early attempts to pass from the local to the global with networks of individually simple devices. In a third paper reprinted in *Embodiments of Mind*

McCulloch, W. S., and Walter Pitts (1943), "A logical calculus of the ideas immanent in neural nets," *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115–137

will be found the prototypes of the linear threshold functions themselves. Readers who are unfamiliar with this theory, or that of Turing machines, are directed to the elementary exposition in

Minsky, Marvin (1967), *Computation: Finite and Infinite Machines*, Prentice-Hall, Englewood Cliffs, N.J.

The local-global transition has dominated several biological areas in recent years. A most striking example is the trend in analysis of animal behavior associated with the name of Tinbergen, as in his classic

Tinbergen, N. (1951), *The Study of Instinct*, Oxford, New York.

Returning to the technical aspects of perceptrons, we find that our main subject is not represented at all in the literature. We know of no papers that either prove that a nontrivial perceptron cannot accomplish a given task or else show by mathematical analysis that a perceptron can be made to compute any significant geometric predicate. There is a vast literature about experimental results but generally these are so inconclusive that we will refrain from citing particular papers. In most cases that seem to show

"success," it can be seen that the data permits an order-1 separation, or even a conjunctively local separation! In these cases, the authors do not mention this, though it seems inconceivable that they could not have noticed it!

The approach closest to ours, though still quite distant, is that of

Bledsoe, W. W., and I. Browning (1959), "Pattern recognition and reading by machine," *Proceedings of the Eastern Joint Computer Conference, 1959*, pp. 225–232.

Another early paper that recognizes the curiously neglected fact that partial predicates work better when realistically matched to the problem, is

Roberts, Lawrence G. (1960), "Pattern recognition with an adaptive network," *IRE International Convention Record*, Part II, pp. 66–70.

Rosenblatt made some studies (in *Neurodynamics*) concerning the probability that if a perceptron recognizes a certain class of figures it will also recognize other figures that are similar in certain ways. In another paper

Rosenblatt, Frank (1960), "Perceptual generalization over transformation groups," *Self-Organizing Systems*, Pergamon Press, New York, pp. 63–96.

he considers group-invariant patterns but does not come close enough to the group-invariance theorem to get decisive results.

The nearest approach to our negative results and methods is the analysis of ψ_{PARITY} found in

Dertouzos, Michael (1965), *Threshold Logic: A Synthesis Approach*, The M.I.T. Press, Cambridge, Mass.

This is also a good book in which to see how people who are *not* interested in geometric aspects of perceptrons deal with linear threshold functions. They had already been interested, for other reasons, in the size of coefficients of (first-order) threshold functions, and we made use of an idea described in

Myhill, John and W. H. Kautz (1961), "On the size of weights required for linear-input switching functions," *IRE Transactions on Electronic Computers*, Vol. 10, No. 2, pp. 288–290

to get our theorem in §10.1. A more recent result on order-1 coefficients is in

Muroga, Saburo, and I. Toda (1966), "Lower bounds on the number of

threshold functions," *IEEE Transactions on Electronic Computers*, Vol. EC-15, No. 5, pp. 805–806,

which improves upon an earlier result in

Muroga, Saburo (1965), "Lower bounds on the number of threshold functions and a maximum weight," *IEEE Transactions on Electronic Computers*, Vol. EC-14, No. 2, pp. 136–148.

These papers also discuss another question: the proportion of Boolean functions (of n-variables) that happens to be first-order. To our knowledge, there is no literature about the same question for higher-order functions.

The general area of artificial intelligence and heuristic programming was mentioned briefly in Chapter 13 as the direction we feel one should look for advanced ideas about pattern recognition and learning. No systematic treatment is available of what is known in this area, but we can recommend a few general references. The collection of papers in

Feigenbaum, Edward A., and Julian Feldman (1963), *Computers and Thought*, McGraw-Hill, New York.

shows the state of affairs in the area up to about 1962, while

Minsky, Marvin (1968), *Semantic Information Processing*, The M.I.T. Press, Cambridge, Mass., 1968

contains more recent papers—mainly doctoral dissertations—dealing with computer programs that manipulate verbal and symbolic descriptions. Anyone interested in this area should also know the classic paper

Newell, Allen, J. C. Shaw, and H. A. Simon (1959), "Report on a general problem-solving program," *Proceedings of International Conference on Information Processing*, UNESCO House, pp. 256–264.

The program mentioned in Chapter 13 is described in detail in

Guzman, Adolfo (1968), "Decomposition of a visual scene into bodies," *Proceedings Fall Joint Computer Conference, 1968.*

Finally, we mention two early works that had a rather broad influence on cybernetic thinking. The fearfully simple homeostat concept mentioned in §11.6 is described in

Ashby, W. Ross (1952), *Design for a brain*, Wiley, New York

which discussed only very simple machines, to be sure, but for the

first time with relentless clarity. At the other extreme, perhaps, was

Hebb, Donald O. (1949), *The Organization of Behavior*, Wiley, New York

which sketched a hierarchy of concepts proposed to account for global states in terms of highly local neuronal events. Although the details of such an enterprise have never been thoroughly worked out, Hebb's outline was for many workers a landmark in the shift from a search for a single, simple principle of brain organization toward more realistic attempts to construct hierarchies (or rather *heterarchies*, as McCulloch would insist) that could support the variety of computations evidently needed for thinking.

You might like to compare your reactions to this book with those of other readers. The following are serious discussions of the book and its theoretical approach:

Block, Herbert D: A Review of "Perceptrons". Information and Control vol. 17, 1970, pp. 501-522.

Newell, Allen: A step toward the understanding of Information Processes. Science vol 165, 22 August 1969, pp. 780-782.

Mycielski, Jan: Review of "Perceptrons". Bull. Amer. Math. Soc. vol 78, Jan 1972, pp. 12-15.

Minsky, M. and Papert, S: Re-View of Perceptrons. A.I. Memo 293, Artificial Intelligence Laboratory, M.I.T., Cambridge, Mass. 02139.

The Block review also contains an extensive bibliography.