PART II

BASIC MECHANISMS

The chapters of Part II represent explorations into specific architectures and learning mechanisms for PDP models. These explorations proceed through mathematical analysis coupled with results from simulations. The major theme which runs through all of these explorations is a focus on the learning problem. How can PDP networks evolve to perform the kinds of tasks we require of them? Since one of the primary features of PDP models in general is their ability to self-modify, these studies form an important base for the application of these models to specific psychological and biological phenomena.

In Chapter 5, Rumelhart and Zipser begin with a summary of the history of early work on learning in parallel distributed processing systems. They then study an unsupervised learning procedure called *competitive learning*. This is a procedure whereby feature detectors capable of discriminating among the members of a set of stimulus input patterns evolve without a specific teacher guiding the learning. The basic idea is to let pools of potential feature detector units *compete* among themselves to respond to each stimulus pattern. The winner within each pool—the one whose connections make it respond most strongly to the pattern—then adjusts its connections slightly toward the pattern that it won. Several earlier investigators have considered variants of the competitive learning idea (e.g., Grossberg, 1976; von der Malsberg, 1973). Rumelhart and Zipser show that when a competitive network is trained through repeated presentations of members of a set of patterns, each unit in a pool comes to respond when patterns with a particular

attribute or property are presented. If there are two units in a pool, each comes to respond to opposite values of a binary feature which is useful in describing the stimulus set. If there are three units in the pool, each unit comes to respond to a value of a trinary feature, etc. It is shown through simulations and mathematical analysis that the competitive learning system can serve as a basis for the development of useful pattern descriptions.

Chapters 6 and 7 describe Smolensky's *harmony theory* and Hinton and Sejnowski's *Boltzmann machine*, respectively. These approaches were developed at the same time, and they have much in common. Both harmony theory and Boltzmann machines employ binary units whose values are determined probabilistically according to the Boltzmann equation. Each employs *simulated annealing* in which the *temperature* of the Boltzmann equation is moved slowly to zero as the system relaxes into its solution state where it finally freezes. Both systems apply mathematical formulations borrowed from physics to their systems to describe and analyze their behavior.

In spite of these similarities, the two systems were developed from very different perspectives. The similarities arose largely because both systems tapped mathematical physics as a tool for formalizing their ideas. Smolensky's harmony theory grew from an attempt to formalize the notion of *schema* and the ideas of schema theory. Hinton and Sejnowski's Boltzmann machine is based on the idea that stochastic units can be used as a mechanism of search—for finding globally good states of networks through simulated annealing. It combines insights on simulated annealing from Kirkpatrick, Gelatt, and Vecchi (1983) with the proof by Hopfield (1982) that there is a global energy function that can be locally minimized through a process of asynchronously updating individual units.

Chapter 6 provides a mathematical development of harmony theory and shows how a symbolic level of description can be seen as emerging from interactions among the individual processing units in harmony theory. It shows how harmony theory can be applied to a variety of phenomena, including intuitive problem solving and aspects of perception. It also provides a useful description of the mathematical relationships among harmony theory, Boltzmann machines, and the related mechanisms studied by S. Geman and D. Geman (1984).

Chapter 7 focuses on the issue of learning in Boltzmann machines. One of the most important contributions of the work on Boltzmann machines is the development of the two phase (wake/sleep) learning procedure. Hinton and Sejnowski show that if a Boltzmann machine runs under the influence of environmental inputs for a while and then runs "freely"—without inputs from the environment—there is a very simple learning rule which will allow the Boltzmann machine to pick up

environmental regularities and develop its own internal representations for describing those regularities. The major part of Chapter 7 is an analysis of this learning procedure.

Chapter 8 is the study of still another learning procedure. In this chapter, Rumelhart, Hinton, and Williams show that it is possible to develop a generalization of the *delta rule* described in Chapter 2 so that arbitrary multilayered networks of units can be can be trained to do interesting tasks. Using this learning rule, the system can learn to associate arbitrary input/output pairs and in this way can learn to compute arbitrary input/output functions. The generalized delta rule is shown to provide a method of modifying any weight in any network, based on locally available information, so as to implement a gradient descent process that searches for those weights that minimize the error at the output units. Further, simulation work presented in the chapter shows that the problems of local minima often associated with gradient descent and other hill-climbing methods are suprisingly rare.

In general, the chapters in this section demonstrate that the barriers to progress in understanding learning in networks of simple neuron-like units have begun to crumble. There are still deep problems that remain unsolved, but the learning mechanisms described in these chapters make several inroads into some of the most challenging aspects of the theory of parallel distributed processing.

CHAPTER 5

Feature Discovery by Competitive Learning

D. E. RUMELHART and D. ZIPSER

This chapter reports the results of our studies with an unsupervised learning paradigm that we call *competitive learning.* We have examined competitive learning using both computer simulation and formal analysis and have found that when it is applied to parallel networks of neuron-like elements, many potentially useful learning tasks can be accomplished. We were attracted to competitive learning because it seems to provide a way to discover the salient, general features which can be used to classify a set of patterns. The basic components of the competitive learning scheme are:

- Start with a set of units that are all the same except for some randomly distributed parameter which makes each of them respond slightly differently to a set of input patterns.
- Limit the "strength" of each unit.
- Allow the units to compete in some way for the right to respond to a given subset of inputs.

The net result of correctly applying these three components to a learning paradigm is that individual units learn to specialize on sets of

This chapter originally appeared in *Cognitive Science*, 1985, 9, 75-112.

similar patterns and thus become "feature detectors" or "pattern classifiers." In addition to Frank Rosenblatt, whose work will be discussed below, several others have exploited competitive learning in one form or another over the years. These include von der Malsburg (1973), Grossberg (1976), Fukushima (1975), and Kohonen (1982). Our analyses differ from many of these in that we focus on the development of feature detectors rather than pattern classification. We address these issues further below.

One of the central issues in the study of the processing capacities of neuron-like elements concerns the limitations inherent in a one-level system and the difficulty of developing learning schemes for multilayered systems. Competitive learning is a scheme in which important features can be discovered at one level that a multilayer system can use to classify pattern sets which cannot be classified with a single level system.

Thirty-five years of experience have shown that getting neuron-like elements to learn some easy things is often quite straightforward, but designing systems with powerful general learning properties is a difficult problem, and the competitive learning paradigm does not change this fact. What we hope to show is that competitive learning is a powerful strategy which, when used in a variety of situations, greatly expedites some difficult tasks. Since the competitive learning paradigm has roots which go back to the very beginnings of the study of artificial learning devices, it seems reasonable to put the whole issue into historical perspective. This is even more to the point, since one of the first simple learning devices, the perceptron, caused great furor and debate, the reverberations of which are still with us.

In the beginning, thirty-five or forty years ago, it was very hard to see how anything resembling a neural network could learn at all, so any example of learning was immensely interesting. Learning was elevated to a status of great importance in those days because it was somehow uniquely associated with the properties of animal brains. After McCulloch and Pitts (1943) showed how neural-like networks could compute, the main problem then facing workers in this area was to understand how such networks could learn.

The first set of ideas that really got the enterprise going were contained in Donald Hebb's *Organization of Behavior* (1949). Before Hebb's work, it was believed that some physical change must occur in a network to support learning, but it was not clear what this change could be. Hebb proposed that a reasonable and biologically plausible change would be to strengthen the connections between elements of the network only when both the presynaptic and postsynaptic units were active simultaneously. The essence of Hebb's ideas still persists today in many learning paradigms. The details of the rules for changing weight

may be different, but the essential notion that the strength of connections between the units must change in response to some function of the correlated activity of the connected units still dominates learning models.

Hebb's ideas remained untested speculations about the nervous system until it became possible to build some form of simulated network to test learning theories. Probably the first such attempt occurred in 1951 when Dean Edmonds and Marvin Minsky built their learning machine. The flavor of this machine and the milieu in which it operated is captured in Minsky's own words which appeared in a wonderful *New Yorker* profile of him by Jeremy Bernstein (1981):

> In the summer of 1951 Dean Edmonds and I went up to Harvard and built our machine. It had three hundred tubes and a lot of motors. It needed some automatic electric clutches, which we machined ourselves. The memory of the machine was stored in the positions of its control knobs, 40 of them, and when the machine was learning, it used the clutches to adjust its own knobs. We used a surplus gyropilot from a B24 bomber to move the clutches. (p. 69)

This machine actually worked and was so fascinating to watch that Minsky remembers:

> We sort of quit science for awhile to watch the machine. We were amazed that it could have several activities going on at once in this little nervous system. Because of the random wiring it had a sort of fail safe characteristic. If one of the neurons wasn't working, it wouldn't make much difference and with nearly three hundred tubes, and the thousands of connections we had soldered there would usually be something wrong somewhere. . . . I don't think we ever debugged our machine completely, but that didn't matter. By having this crazy random design it was almost sure to work no matter how you built it. (p. 69)

In fact, the functioning of this machine apparently stimulated Minsky sufficiently to write his PhD thesis on a problem related to learning (Minsky, 1954). The whole idea must have generated rather wide interest; von Neumann, for example, was on Minsky's PhD committee and gave him encouragement. Although Minsky was perhaps the first on the scene with a learning machine, the real beginnings of meaningful neuron-like network learning can probably be traced to the work of Frank Rosenblatt, a Bronx High School of Science classmate of

Minsky's. Rosenblatt invented a class of simple neuron-like learning networks which he called perceptrons. In his book, *Principles of Neurodynamics* (1962), Rosenblatt brought together all of his results on perceptrons. In that book he gives a particularly clear description of what he thought he was doing:

> Perceptrons are not intended to serve as detailed copies of any actual nervous system. They're simplified networks, designed to permit the study of lawful relationships between the organization of a nerve net, the organization of its environment, and the "psychological" performances of which it is capable. Perceptrons might actually correspond to parts of more extended networks and biological systems; in this case, the results obtained will be directly applicable. More likely they represent extreme simplifications of the central nervous system, in which some properties are exaggerated and others suppressed. In this case, successive perturbations and refinements of the system may yield a closer approximation.
>
> The main strength of this approach is that it permits meaningful questions to be asked and answered about particular types of organizations, hypothetical memory mechanisms, and neural models. When exact analytical answers are unobtainable, experimental methods, either with digital simulation or hardware models, are employed. The model is not the terminal result, but a starting point for exploratory analysis of its behavior. (p. 28)

Rosenblatt pioneered two techniques of fundamental importance to the study of learning in neural-like networks: digital computer simulation and formal mathematical analysis, although he was not the first to simulate neural networks that could learn on digital computers (cf. Farley & Clark, 1954).

Since the paradigm of competitive learning uses concepts that appear in the work of Rosenblatt, it is worthwhile reviewing some of his ideas in this area. His most influential result was the "perceptron learning theorem" which boldly asserts:

> Given an elementary α-perceptron, a stimulus world W, and any classification $C(W)$ for which a solution exists; let all stimuli in W occur in any sequence, provided that each stimulus must reoccur in finite time; then beginning from an arbitrary initial state, an error correction procedure will always yield a solution to $C(W)$ in finite time, . . . (p. 596)

As it turned out, the real problems arose out of the phrase "for which a solution exists"—more about this later.

Less widely known is Rosenblatt's work on what he called "spontaneous learning." All network learning models require rules which tell how to present the stimuli and change the values of the weights in accordance with the response of the model. These rules can be characterized as forming a spectrum, at one end of which is learning with an error-correcting teacher, and at the other is completely spontaneous, unsupervised discovery. In between is a continuum of rules that depend on manipulating the content of the input stimulus stream to bring about learning. These intermediate rules are often referred to as "forced learning." Here we are concerned primarily with attempts to design a perceptron that would discover something interesting without a teacher because this is similar to what happens in the competitive learning case. In fact, Rosenblatt was able to build a perceptron that was able to spontaneously dichotomize a random sequence of input patterns into classes such that the members of a single class were similar to each other, and different from the members of the other class. Rosenblatt realized that any randomly initialized perceptron would have to dichotomize an arbitrary input pattern stream into a "1-set," consisting of those patterns that happened to produce a response of 1, and a "0-set," consisting of those that produced a response of 0. Of course one of these sets could be empty by chance and neither would be of much interest in general. He reasoned that if a perceptron could reinforce these sets by an appropriate rule based only on the perceptron's spontaneous response and not on a teacher's error correction, it might eventually end up with a dichotomization in which the members of each set were more like each other than like the members of the opposite set. What was the appropriate rule to use to achieve the desired dicotomization? The first rule he tried for these perceptrons, which he called C-type, was to increment weights on lines active with patterns in the 1-set, and decrement weights on lines active with patterns in the 0-set. The idea was to force a dichotomization into sets whose members were similar in the sense that they activated overlapping subsets of lines. The results were disastrous. Sooner or later all the input patterns were classified in one set. There was no dichotomy but there was stability. Once one of the sets won, it remained the victor forever.

Not to be daunted, he examined why this undesirable result occurred and realized that the problem lay in the fact that since the weights could grow without limit, the set that initially had a majority of the patterns would receive the majority of the reinforcement. This meant that weights on lines which could be activated by patterns in both sets would grow to infinite magnitudes in favor of the majority set, which in turn would lead to the capture of minority patterns by the majority set and

ultimate total victory for the majority. Even where there was initial equality between the sets, inevitable fluctuations in the random presentation of patterns would create a majority set that would then go on to win. Rosenblatt overcame this problem by introducing mechanisms to limit weight growth in such a way that the set that was to be positively reinforced at active lines would compensate the other set by giving up some weight from all its lines. He called the modified perceptrons C'. An example of a C' rule is to lower the magnitude of all weights by a fixed fraction of their current value before specifically incrementing the magnitude of some of the weights on the basis of the response to an input pattern. This type of rule had the desired result of making an equal dichotomy of patterns a stable rather than an unstable state. Patterns in each of the sets were similar to each other in the sense that they depended on similar sets of input lines to produce a response. In Rosenblatt's initial experiment, the main feature of similarity was not so much the shape of the patterns involved, but their location on the retina. That is, his system was able to spontaneously learn something about the geometry of its input line arrangement. Later, we will examine this important property of spontaneous geometry learning in considerable detail. Depending on the desired learning task, it can be either a boon or a nuisance.

Rosenblatt was extremely enthusiastic about his spontaneous learning results. In fact, his response can be described as sheer ecstasy. To see what he thought about his achievements, consider his claim (Rosenblatt, 1959):

> It seems clear that the class C' perceptron introduces a new kind of information processing automaton: For the first time, we have a machine which is capable of having original ideas. As an analogue of the biological brain, the perceptron, more precisely, the theory of statistical separability, seems to come closer to meeting the requirements of a functional explanation of the nervous system than any system previously proposed. (p. 449)

Although Rosenblatt's results were both interesting and significant, the claims implied in the above quote struck his contemporaries as unfounded. What was also significant was that Rosenblatt appeared to be saying that the type of spontaneous learning he had demonstrated was a property of perceptrons, which could not be replicated by ordinary computers. Consider the following quote from the same source:

> As a concept, it would seem that the perceptron has established, beyond doubt, the feasibility and principle of

> non-human systems which may embody human cognitive functions at a level far beyond that which can be achieved through present day automatons. The future of information processing devices which operate on statistical, rather then logical principles seems to be clearly indicated. (p. 449)

It is this notion of Rosenblatt's—that perceptrons are in some way superior to computers—that ignited a debate in artificial intelligence that had significant effects on the development of neural-like network models for both learning and other cognitive processes. Elements of the debate are still with us today in arguments about what the brain can do that computers can't do. There is no doubt that this was an important issue in Rosenblatt's mind, and almost certainly contributed to the acrimonious debate at that time. Consider the following statement by Rosenblatt made at the important conference on Mechanization of Thought Processes back in 1959:

> Computers seem to share two main functions with the brain: (a) Decision making, based on logical rule, and (b) control, again based on logical rules. The human brain performs these functions, together with a third: interpretation of the environment. Why do we hold interpretation of the environment to be so important? The answer, I think, is to be found in the laws of thermodynamics. A system with a completely self contained logic can never spontaneously improve its ability to organize, and to draw valid conclusions from information. (Rosenblatt, 1959, p. 423)

Clearly in some sense, Rosenblatt was saying that there were things that the brain and perceptrons, because of their statistical properties, could do which computers could not do. Now this may seem strange since Rosenblatt knew that a computer program could be written that would simulate the behavior of statistical perceptrons to any arbitrary degree of accuracy. Indeed, he was one of the pioneers in the application of digital simulation to this type of problem. What he was actually referring to is made clear when we examine the comments of other participants at the conference, such as Minsky (1959) and McCarthy (1959), who were using the symbol manipulating capabilities of the computer to directly simulate the logical processes involved in decision making, theorem proving, and other intellectual activities of this sort. Rosenblatt believed the computer used in this way would be inadequate to mimic the brain's true intellectual powers. This task, he thought, could only be accomplished if the computer or other electronic devices were used to simulate perceptrons. We can summarize these divergent

points of view by saying that Rosenblatt was concerned not only with what the brain did, but with how it did it, whereas others, such as Minsky and McCarthy, were concerned with simulating what the brain did, and didn't really care how it was done. The subsequent history of AI has shown both the successes and failures of the standard AI approach. We still have the problems today, and it's still not clear to what degree computational strategies similar to the ones used by the brain must be employed in order to simulate its performance.

In addition to producing fertilizer, as all debates do, this one also stimulated the growth of some new results on perceptrons, some of which came from Minsky. Rosenblatt had shown that a two layer perceptron could carry out any of the 2^{2^N} possible classifications of N binary inputs; that is, a solution to the classification problem had always existed in principle. This result was of no practical value however, because 2^N units were required to accomplish the task in the completely general case. Rosenblatt's approach to this problem was to use a much smaller number of units in the first layer with each unit connected to a small subset of the N inputs at random. His hope was that this would give the perceptron a high probability of learning to carry out classifications of interest. Experiments and formal analysis showed that these random devices could learn to recognize patterns to a significant degree but that they had severe limitations. Rosenblatt (1962) characterized his random perceptron as follows:

> It does not generalize well to similar forms occurring in new positions in the retinal field, and its performance in detection experiments, where a familiar figure appears against an unfamiliar background, is apt to be weak. More sophisticated psychological capabilities, which depend on the recognition of topological properties of the stimulus field, or on abstract relations between the components of a complex image, are lacking. (pp. 191-192)

Minsky and Papert worked through most of the sixties on a mathematical analysis of the computing powers of perceptrons with the goal of understanding these limitations. The results of their work are available in a book called *Perceptrons* (Minsky & Papert, 1969). The central theme of this work is that parallel recognizing elements, such as perceptrons, are beset by the same problems of scale as serial pattern recognizers. Combinatorial explosion catches you sooner or later, although sometimes in different ways in parallel than in serial. Minsky and Papert's book had a very dampening effect on the study of neuron-like networks as computational devices. Minsky has recently come to reconsider this negative effect:

> I now believe the book was overkill. . . . So after being irritated with Rosenblatt for overclaiming and diverting all those people along a false path, I started to realize that for what you get out of it — the kind of recognition it can do—it is such a simple machine that it would be astonishing if nature did not make use of it somewhere. (Bernstein, 1981, p. 103)

Perhaps the real lesson from all this is that it really is worthwhile trying to put things in perspective.

Once the problem of scale has been understood, networks of neuron-like elements are often very useful in practical problems of recognition and classification. These networks are somewhat analogous to computers, in that they won't do much unless programmed by a clever person; networks, of course, are not so much programmed as designed. The problem of finding networks of practical size to solve a particular problem is challenging because relatively small changes in network design can have very large effects on the scale of a problem. Consider networks of neuron-like units that determine the parity of their N binary inputs (see Figure 1). In the simple perceptrons studied by Minsky and Papert, units in the first layer output 1 only if all their inputs are 1 and output 0 otherwise. This takes 2^N units in the first layer, and a single linear threshold unit with a fan-in of 2^N in the second layer, to determine parity. If the units in the first layer are changed to linear threshold elements, then only N of them are required, but all must have a fan-in of N. If we allow a multilayer network to do the job, then about $3N$ units are needed, but none needs a fan-in of more than 2. The number of layers is of order $\log_2 N$. The importance of all this to the competitive learning paradigm, or any other for that matter, is that no network can learn what it is not capable of doing in principle. What any particular network can do is dependent on its structure and the computational properties of its component elements. Unfortunately, there is no canonical way to find the best network or to determine what it will learn, so the whole enterprise still has much of the flavor of an experimental science.

THE COMPETITIVE LEARNING MECHANISM

Paradigms of Learning

It is possible to classify learning mechanisms in several ways. One useful classification is in terms of the learning paradigm in which the

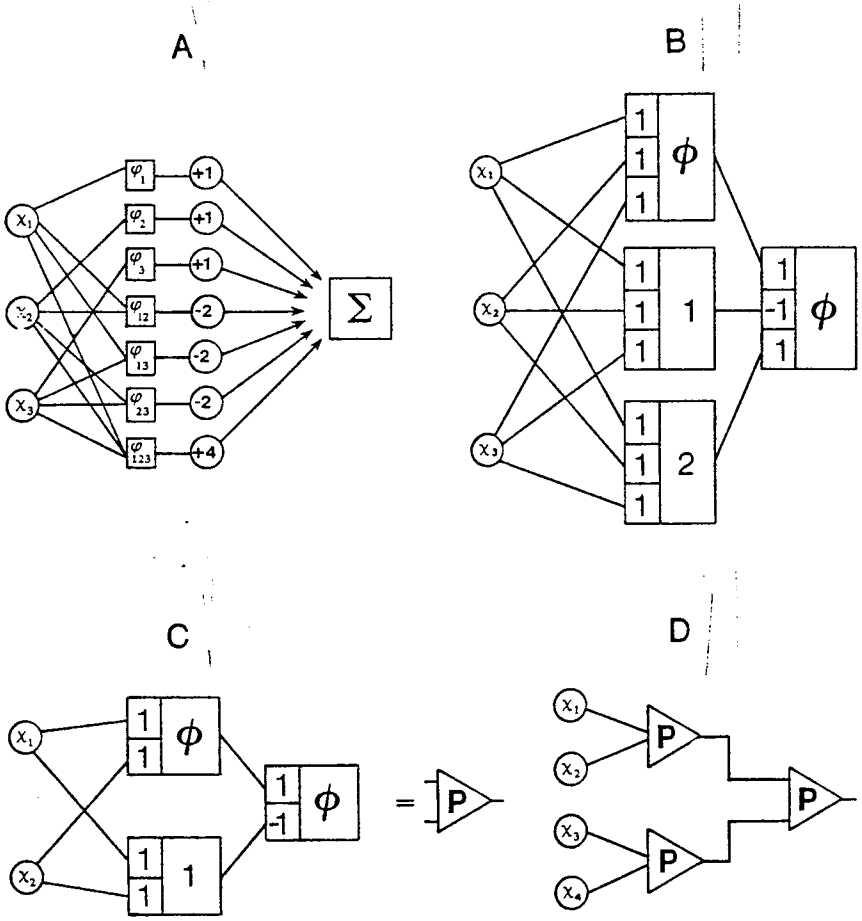

FIGURE 1. *A*: Parity network from Minsky and Papert (1969). Each ϕ unit has an output of 1 only if all of its inputs are 1. Σ is a linear threshold unit with threshold of 0, i.e., like all the other linear threshold units in the figure, it fires only when the sum of its weighted inputs is greater than the threshold. This and all the other networks signal odd parity with a 1 in the rightmost unit of the network. *B*: Parity network made from two layers of linear threshold units. *C*: Three-unit network for determining the parity of a pair of inputs. *D*: Two-layer network using the subnetwork described in *(C)*. In general, the number of *P*-units is of order N and the number of layers is of order $\log_2 N$.

model is supposed to work. There are at least four common learning paradigms in neural-like processing systems:

- *Auto Associator.* In this paradigm a set of patterns are repeatedly presented and the system is supposed to "store" the patterns. Then, later, parts of one of the original patterns or possibly a pattern similar to one of the original patterns is presented, and the task is to "retrieve" the original pattern through a kind of pattern completion procedure. This is an auto-association process in which a pattern is associated with itself so that a degraded version of the original pattern can act as a retrieval cue.

- *Pattern Associator.* This paradigm is really a variant on the auto-association paradigm. A set of *pairs* of patterns are repeatedly presented. The system is to learn that when one member of the pair is presented it is supposed to produce the other. In this paradigm one seeks a mechanism in which an essentially arbitrary set of input patterns can be paired with an arbitrary set of output patterns.

- *Classification Paradigm.* The classification paradigm also can be considered as a variant on the previous learning paradigms, although the goals are sufficiently different and it is sufficiently common that it deserves separate mention. In this case, there is a fixed set of categories into which the stimulus patterns are to be classified. There is a training session in which the system is presented with the stimulus patterns along with the categories to which each stimulus belongs. The goal is to learn to correctly classify the stimuli so that in the future when a particular stimulus or a slightly distorted version of one of the stimuli is presented, the system will classify it properly. This is the typical paradigm in which the perceptron is designed to operate and in which the perceptron convergence theorem is proved.

- *Regularity Detector.* In this paradigm there is a population of stimulus patterns and each stimulus pattern, S_k, is presented with some probability p_k. The system is supposed to *discover* statistically salient features of the input *population*. Unlike the classification paradigm, there is no a priori set of categories into which the patterns are to be classified; rather, the system must develop its own featural representation of the input stimuli which captures the most salient features of the population of input patterns.

Competitive learning is a mechanism well-suited for regularity detection, as in the environment described in above.

Competitive Learning

The architecture of a competitive learning system (illustrated in Figure 2) is a common one. It consists of a set of hierarchically layered units in which each layer connects, via excitatory connections, with the layer immediately above it. In the most general case, each unit of a layer receives an input from each unit of the layer immediately below and projects output to each unit in the layer immediately above it. Moreover, within a layer, the units are broken into a set of inhibitory clusters in which all elements within a cluster inhibit all other elements in the cluster. Thus the elements within a cluster at one level compete with one another to respond to the pattern appearing on the layer below. The more strongly any particular unit responds to an incoming stimulus, the more it shuts down the other members of its cluster.

There are many variations on the competitive learning theme. A number of researchers have developed variants of competitive learning mechanisms and a number of results already exist in the literature. We have already mentioned the pioneering work of Rosenblatt. In addition, von der Malsburg (1973), Fukushima (1975), and Grossberg (1976), among others, have developed models which are competitive learning models, or which have many properties in common with competitive learning. We believe that the essential properties of the competitive learning mechanism are quite general. However, for the sake of concreteness, in this paper we have chosen to study, in some detail, the simplest of the systems which seem to be representative of the essential characteristics of competitive learning. Thus, the system we have analyzed has much in common with the previous work, but wherever possible we have simplified our assumptions. The system that we have studied most is described below:

- The units in a given layer are broken into a set of nonoverlapping clusters. Each unit within a cluster inhibits every other unit within a cluster. The clusters are winner-take-all, such that the unit receiving the largest input achieves its maximum value while all other units in the cluster are pushed to their minimum value.[1] We have arbitrarily set the maximum value to 1 and the minimum value to 0.

[1] A simple circuit for achieving this result is attained by having each unit activate itself and inhibit its neighbors. Grossberg (1976) employs just such a network to *choose* the maximum value of a set of units.

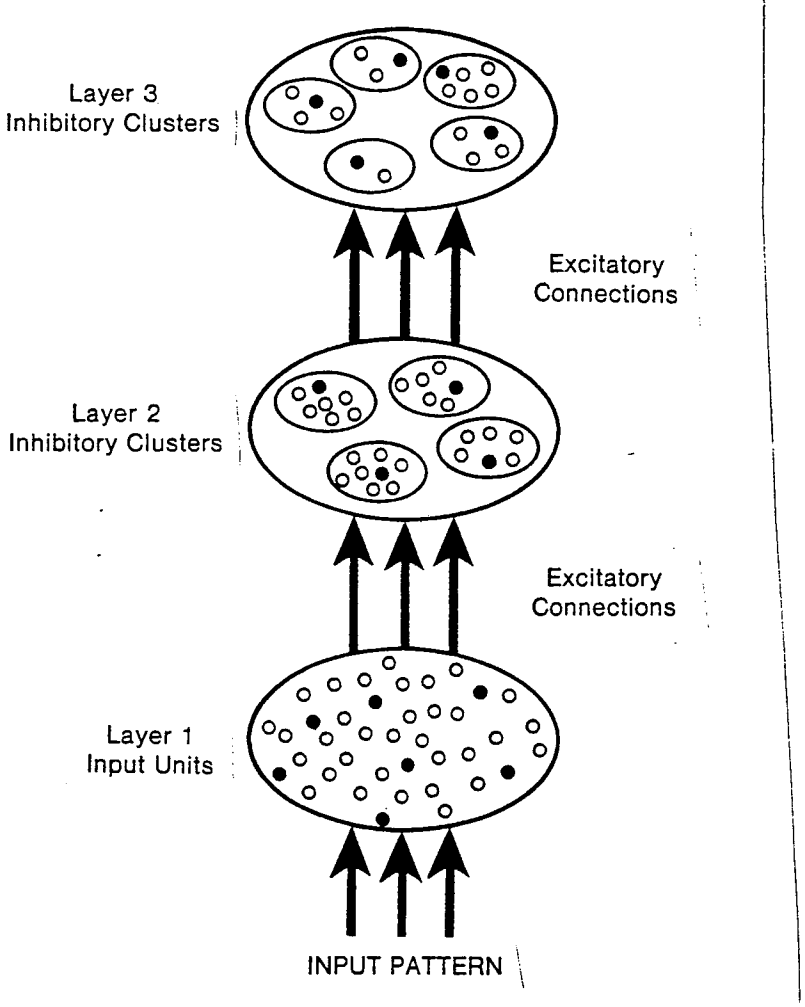

FIGURE 2. The architecture of the competitive learning mechanism. Competitive learning takes place in a context of sets of hierarchically layered units. Units are represented in the diagram as dots. Units may be active or inactive. Active units are represented by filled dots, inactive ones by open dots. In general, a unit in a given layer can receive inputs from all of the units in the next lower layer and can project outputs to all of the units in the next higher layer. Connections between layers are excitatory and connections within layers are inhibitory: Each layer consists of a set of clusters of mutually inhibitory units. The units within a cluster inhibit one another in such a way that only one unit per cluster may be active. We think of the configuration of active units on any given layer as representing the input pattern for the next higher level. There can be an arbitrary number of such layers. A given cluster contains a fixed number of units, but different clusters can have different numbers of units.

- Every element in every cluster receives inputs from the same lines.

- A unit learns if and only if it wins the competition with other units in its cluster.

- A stimulus pattern S_j consists of a binary pattern in which each element of the pattern is either *active* or *inactive*. An active element is assigned the value 1 and an inactive element is assigned the value 0.

- Each unit has a fixed amount of weight (all weights are positive) which is distributed among its input lines. The weight on the line connecting unit i on the lower (or input) layer to unit j on the upper layer, is designated w_{ij}. The fixed total amount of weight for unit j is designated $\sum_i w_{ij} = 1$. A unit learns by shifting weight from its inactive to its active input lines. If a unit does not respond to a particular pattern, no learning takes place in that unit. If a unit wins the competition, then each of its input lines give up some proportion g of its weight and that weight is then distributed equally among the active input lines.[2] More formally, the learning rule we have studied is:

$$\Delta w_{ij} = \begin{cases} 0 & \text{if unit } j \text{ loses on stimulus } k \\ g\dfrac{c_{ik}}{n_k} - gw_{ij} & \text{if unit } j \text{ wins on stimulus } k \end{cases}$$

where c_{ik} is equal to 1 if in stimulus pattern S_k, unit i in the lower layer is active and zero otherwise, and n_k is the number of active units in pattern S_k (thus $n_k = \sum_i c_{ik}$).

Figure 3 illustrates a useful geometric analogy to this system. We can consider each stimulus pattern as a vector. If all patterns contain the same number of active lines, then all vectors are the same length and each can be viewed as a point on an N-dimensional hypersphere,

---

[2] This learning rule was proposed by von der Malsburg (1973). As Grossberg (1976) points out, renormalization of the weights is not necessary. The same result can be obtained by normalizing the input patterns and then assuming that the weights approach the values on the input lines. Normalizing weights is simpler to implement than normalizing patterns, so we chose that option. For most of our experiments, however, it does not matter which of these two rules we chose since all patterns were of the same magnitude.

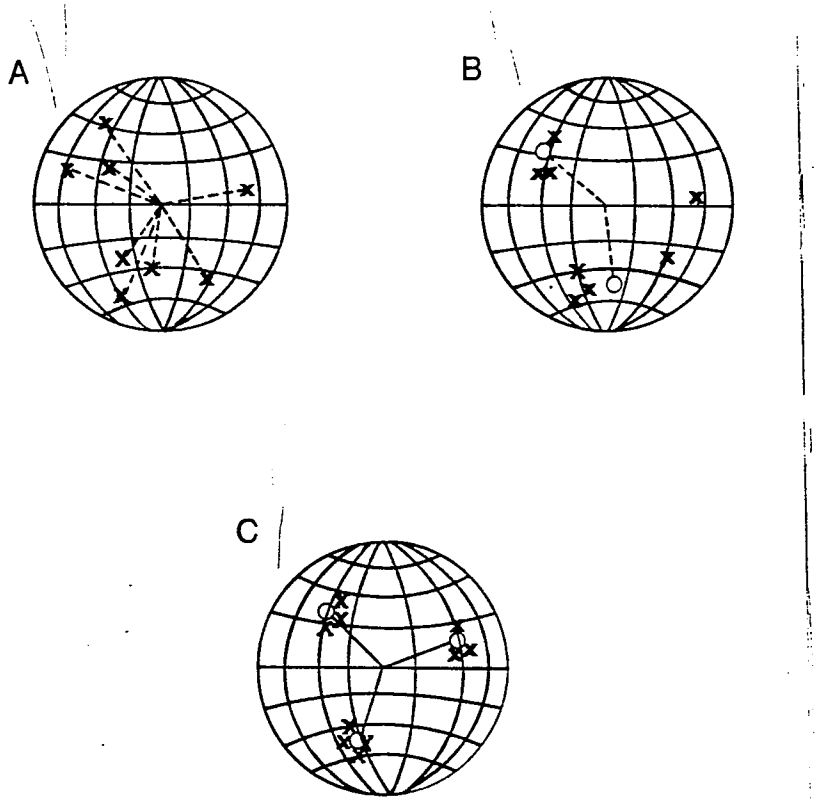

FIGURE 3. A geometric interpretation of competitive learning. *A*: It is useful to conceptualize stimulus patterns as vectors whose tips all lie on the surface of a hypersphere. We can then directly see the similarity among stimulus patterns as distance between the points on the sphere. In the figure, a stimulus pattern is represented as an ×. The figure represents a population of eight stimulus patterns. There are two clusters of three patterns and two stimulus patterns which are rather distinct from the others. *B*: It is also useful to represent the weights of units as vectors falling on the surface of the same hypersphere. Weight vectors are represented in the figure as ○'s. The figure illustrates the weights of two units falling on rather different parts of the sphere. The response rule of this model is equivalent to the rule that whenever a stimulus pattern is presented, the unit whose weight vector is closest to that stimulus pattern on the sphere wins the competition. In the figure, one unit would respond to the cluster in the northern hemisphere and the other unit would respond to the rest of the stimulus patterns. *C*: The learning rule of this model is roughly equivalent to the rule that whenever a unit wins the competition (i.e., is closest to the stimulus pattern), that weight vector is moved toward the presented stimulus. The figure shows a case in which there are three units in the cluster and three natural groupings of the stimulus patterns. In this case, the weight vectors for the three units will each migrate toward one of the stimulus groups.

where N is the number of units in the lower level, and therefore, also the number of input lines received by each unit in the upper level.

Each × in Figure 3A represents a particular pattern. Those patterns that are very similar are near one another on the sphere; those that are very different will be far from one another on the sphere. Now note that since there are N input lines to each unit in the upper layer, its weights can also be considered a vector in N-dimensional space. Since all units have the same total quantity of weight, we have N-dimensional vectors of approximately fixed length for each unit in the cluster. [3] Thus, properly scaled, the weights themselves form a set of vectors which (approximately) fall on the surface of the same hypersphere. In Figure 3B, the ○'s represent the weights of two units superimposed on the same sphere with the stimulus patterns. Now, whenever a stimulus pattern is presented, the unit which responds most strongly is simply the one whose weight vector is nearest that for the stimulus. The learning rule specifies that whenever a unit wins a competition for a stimulus pattern, it moves a percentage g of the way from its current location toward the location of the stimulus pattern on the hypersphere. Now, suppose that the input patterns fell into some number, M, "natural" groupings. Further, suppose that an inhibitory cluster receiving inputs from these stimuli contained exactly M units (as in Figure 3C). After sufficient training, and assuming that the stimulus groupings are sufficiently distinct, we expect to find one of the vectors for the M units placed roughly in the center of each of the stimulus groupings. In this case, the units have come to detect the grouping to which the input patterns belong. In this sense, they have "discovered" the structure of the input pattern sets.

Some Features of Competitive Learning

There are several characteristics of a competitive learning mechanism that make it an interesting candidate for further study, for example:

- Each cluster classifies the stimulus set into M groups, one for each unit in the cluster. Each of the units captures roughly an equal number of stimulus patterns. It is possible to consider a cluster as forming an M-ary feature in which every stimulus pattern is classified as having exactly one of the M possible

[3] It should be noted that this geometric interpretation is only approximate. We have used the constraint that $\sum_i w_{ij} = 1$ rather than the constraint that $\sum_i w_{ij}^2 = 1$. This latter constraint would ensure that all vectors are in fact the same length. Our assumption only assures that they will be approximately the same length.

values of this feature. Thus, a cluster containing 2 units acts as a binary feature detector. One element of the cluster responds when a particular feature is present in the stimulus pattern, otherwise the other element responds.

- If there is *structure* in the stimulus patterns, the units will break up the patterns along structurally relevant lines. Roughly speaking, this means that the system will find clusters if they are there. (A key problem, which we address below, is specifying the *nature* of the structure that this system discovers.)

- If the stimuli are highly structured, the classifications are highly stable. If the stimuli are less well-structured, the classifications are more variable, and a given stimulus pattern will be responded to first by one and then by another member of the cluster. In our experiments, we started the weight vectors in random directions and presented the stimuli randomly. In this case, there is rapid movement as the system reaches a relatively stable configuration (such as one with a unit roughly in the center of each cluster of stimulus patterns). These configurations can be more or less stable. For example, if the stimulus points don't actually fall into nice clusters, then the configurations will be relatively unstable, and the presentation of each stimulus will modify the pattern of responding so that the system will undergo continual evolution. On the other hand, if the stimulus patterns fall rather nicely into clusters, then the system will become very stable in the sense that the same units will always respond to the same stimuli.[4]

- The particular grouping done by a particular cluster depends on the starting value of the weights and the sequence of stimulus patterns actually presented. A large number of clusters, each receiving inputs from the same input lines can, in general, classify the inputs into a large number of different groupings, or alternatively, discover a variety of independent features present in the stimulus population. This can provide a kind of coarse coding of the stimulus patterns.[5]

---

[4] Grossberg (1976) has addressed this problem in his very similar system. He has proved that if the patterns are sufficiently sparse, and/or when there are enough units in the cluster, then a system such as this will find a perfectly stable classification. He also points out that when these conditions don't hold, the classification can be unstable. Most of our work is with cases in which there is *no* perfectly stable classification and the number of patterns is *much* larger than the number of units in the inhibitory clusters.

Formal Analysis

Perhaps the simplest mathematical analysis that can be given of the competitive learning model under discussion involves the determination of the sets of *equilibrium states* of the system—that is, states in which the average inflow of weight to a particular line is equal to the average outflow of weight on that line. Let p_k be the probability that stimulus S_k is presented on any trial. Let v_{jk} be the probability that unit j wins when stimulus S_k is presented. Now we want to consider the case in which $\sum_k \Delta w_{ij} v_{jk} p_k = 0$, that is, the case in which the average change in the weights is zero. We refer to such states as *equilibrium states*. Thus, using the learning rule and averaging over stimulus patterns we can write

$$0 = g\sum_k \frac{c_{ik}}{n_k} p_k v_{jk} - g\sum_k w_{ij} p_k v_{jk}$$

which implies that at equilibrium

$$w_{ij}\sum_k p_k v_{jk} = \sum_k \frac{p_k c_{ik} v_{jk}}{n_k}$$

and thus

$$w_{ij} = \frac{\sum_k \frac{p_k c_{ik} v_{jk}}{n_k}}{\sum_k p_k v_{jk}}.$$

There are a number of important observations to note about this equation. First, note that $\sum_k p_k v_{jk}$ is simply the probability that unit k wins averaged over all stimulus patterns. Note further that $\sum_k p_k c_{ik} v_{jk}$ is the probability that input line i is active and unit j wins. Thus, the ratio $\frac{\sum_k p_k c_{ik} v_{jk}}{\sum_k p_k v_{jk}}$ is the conditional probability that line i is active given unit j

---

[5] There is a problem in that one can't be certain that the different clusters will discover different features. A slight modification of the system in which clusters "repel" one another can insure that different clusters find different features. We shall not pursue that further in this paper.

wins, $p(\text{line}_i = 1 \mid \text{unit}_j \text{ wins})$. Thus, if all patterns are of the same size, i.e., $n_k = n$ for all k, then the weight w_{ij} becomes proportional to the probability that line i is active given unit j wins. That is,

$$w_{ij} \rightarrow np\,(\text{line}_i = 1 \mid \text{unit}_j \text{ wins}).$$

We are now in a position to specify the response, at equilibrium, of unit j when stimulus S_l is presented. Let α_{jl} be the input to unit j in the face of stimulus S_l. This is simply the sum of weights on the active input lines. This can be written

$$\alpha_{jl} \rightarrow \sum_i w_{ij} c_{il} = \sum_i c_{il} \frac{\sum_k \frac{p_k c_{ik} v_{jk}}{n_k}}{\sum_k p_k v_{jk}}$$

which implies that at equilibrium

$$\alpha_{jl} = \frac{\sum_i p_i r_{li} v_{ji}}{\sum_i p_i v_{ji}}$$

where r_{li} represents the overlap between stimulus l and stimulus i, $r_{li} = \sum_k \frac{c_{ki} c_{kl}}{n_i}$. Thus, at equilibrium a unit responds most strongly to patterns that overlap other patterns to which the unit responds and responds most weakly to patterns that are far from patterns to which it responds. Finally, it should be noted that there is another set of restrictions on the value of v_{jk}—the probability that stimulus unit j responds to stimulus S_k. In fact, the competitive learning rule we have studied has the further restriction that

$$v_{jk} = \begin{cases} 1 & \alpha_{jk} > \alpha_{ik} \text{ for all } i \neq j \\ 0 & \text{otherwise.} \end{cases}$$

Thus, in general, there are many solutions to the equilibrium equations described above. The competitive learning mechanisms can only reach those equilibrium states in which the above-stated relationships between the v_{jk} and the α_{jk} also hold.

Whenever the system is in a state in which, on average, the weights are not changing, we say that the system has reached an *equilibrium state*. In such a state the values of α_{jk} become relatively stable, and therefore, the values of v_{ik} become stable. When this happens, the system always responds the same way to a particular stimulus pattern. However, it is possible that the weights will be pushed out of

equilibrium by an unfortunate sequence of stimuli. In this case, the system can move toward a new equilibrium state (or possibly back to a previous one). Some equilibrium states are more stable than others in the sense that the v_{ik} become very unlikely to change values for long periods of time. In particular, this will happen whenever the largest α_{jk} is much larger than any other α_{ik} for all stimulus patterns S_k. In this case, small movements in the weight vector of one of the units is very unlikely to change which unit responds to which stimulus pattern. Such equilibrium states are said to be highly *stable.* We should expect, then, that after it has been learning for a period of time, the system will spend most of its time in the most highly stable of the equilibrium states. One good measure of the stability of an equilibrium state is given by the average amount by which the input to the winning units is greater than the response of all of the other units averaged over all patterns and all units in a cluster. This measure is given by T below:

$$T = \sum_k p_k \sum_{j,i} v_{jk} (\alpha_{jk} - \alpha_{ik}).$$

The larger the value of T, the more stable the system can be expected to be and the more time we can expect the system to spend in that state. Roughly, if we assume that the system moves into states which maximize T, we can show that this amounts to maximizing the overlap among patterns within a group while minimizing the overlap among patterns between groups. In the geometric analogy above, this will occur when the weight vectors point toward maximally compact stimulus regions that are as distant as possible from other such regions.

SOME EXPERIMENTAL RESULTS

Dipole Experiments

The essential structure that a competitive learning mechanism can discover is represented in the overlap of stimulus patterns. The simplest stimulus population in which stimulus patterns can overlap with one another is one constructed out of *dipoles*—stimulus patterns consisting of exactly two active elements and the rest inactive. If we have a total of N input units there are $N(N-1)/2$ possible dipole stimuli. Of course, if the actual stimulus population consists of all $N(N-1)/2$ possibilities, there is no structure to be discovered. There are no clusters for our units to point at (unless we have one unit for each of the possible stimuli, in which case we can point a weight vector at each of the

possible input stimuli). If, however, we restrict the possible dipole stimuli in certain ways, then there can be meaningful groupings of the stimulus patterns that the system can find. Consider, as an example, a case in which the stimulus lines could be thought of as forming a two-dimensional grid in which the only possible stimulus patterns were those which formed adjacent pairs in the grid. If we have an $N \times M$ grid, there are $N(M-1) + M(N-1)$ possible stimuli. Figure 4 shows one of the 24 possible adjacent dipole patterns defined on a 4×4 grid. We carried out a number of experiments employing stimulus sets of this kind. In most of these experiments we employed a two-layer system with a single inhibitory cluster of size two. Figure 5 illustrates the architecture of one of our experiments. The results of three runs with this architecture are illustrated in Figure 6, which shows the relative values of the weights for the two units. The values are shown laid out on a 4×4 grid so that weights are next to one another if the units with which they connect are next to one another. The relative values of the weights are indicated by the filling of the circles. If a circle is filled, that indicates that Unit 1 had the largest weight on that line. If the circle is unfilled, that means that Unit 2 had the largest weight on that line. The grids on the left indicate the initial configurations of the weights. The grids on the right indicate the final configurations of weights. The lines connecting the circles represent the possible stimuli. For example, the dipole stimulus pattern consisting of the upper left input line and the one immediately to the right of it is represented by the line connecting the upper-left circle in the grid with its right neighbor. The unit that wins when this stimulus is presented is indicated by the width of the line connecting the two circles. The wide line indicates

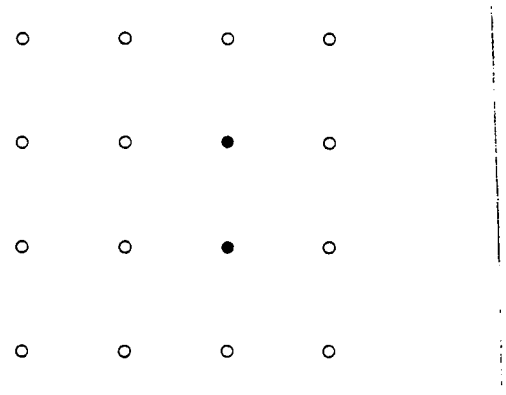

FIGURE 4. A dipole stimulus defined on a 4×4 matrix of input units. The rule for generating such stimuli is simply that any two adjacent units may be simultaneously active. Nonadjacent units may not be active and more than two units may not be simultaneously active. Active units are indicated by filled circles.