CHAPTER 17

A Distributed Model of Human Learning and Memory

J. L. McCLELLAND and D. E. RUMELHART

The view that human memory is distributed has come and gone several times over the years. Hughlings-Jackson, the 19th century neurologist; Kurt Goldstein, the Gestalt neurologist of the early 20th century; and Karl Lashley, the physiological psychologist of the same era, all held to variants of a distributed model of the physiology of learning and memory.

While Lashley's and Goldstein's more radical views have not been borne out, the notion that memory is physiologically distributed within circumscribed regions of the brain seems to be quite a reasonable and plausible assumption (see Chapters 20 and 21). But given the rather loose coupling between a psychological or cognitive theory and a theory of physiological implementation, we can ask, does the notion of distributed memory have anything to offer us in terms of an understanding of human cognition?

In Chapter 3, several of the attractive properties of distributed models are described, primarily from a computational point of view. This chapter addresses the relevance of distributed models from the point of view of the psychology of memory and learning. We begin by

---

This chapter is a modified version of the article "Distributed Memory and the Representation of General and Specific Information" by J. L. McClelland and D. E. Rumelhart, which appeared in *Journal of Experimental Psychology: General*, 1985, *114*, 159-188.

considering a dilemma that faces cognitive theories of learning and memory, concerning the representations of general and specific information. Then we show how a simple version of a distributed model circumvents the dilemma. We go on to show how this model can account for a number of recent findings about the learning and representation of general and specific information. We end by considering some other phenomena that can be accounted for using a distributed model, and we consider a limitation of the model and ways this limitation might be overcome.

The Dilemma

One central dilemma for theories of memory has to do with the choice to represent *general* or *specific* information. On the one hand, human memory and human learning seem to rely on the formation of summary representations that generalize from the details of the specific experiences that gave rise to them. A large number of experiments, going back to the seminal findings of Posner and Keele (1968) indicate that we appear to extract what is common to a set of experiences. On the basis of this sort of evidence, a number of theorists have proposed that memory is largely a matter of generalized representations, either abstracted representations of concepts discarding irrelevant features, or prototypes—representations of typical exemplars (Rosch, 1975). On the other hand, specific events and experiences play a prominent role in memory. Experimental demonstrations of the importance of specific stimulus events even in tasks which have been thought to involve abstraction of a concept or rule are now legion. Responses in categorization tasks (Brooks, 1978; Brooks, Jacoby, & Whittlesea, 1985; Medin & Schaffer, 1978), perceptual identification tasks (Jacoby, 1983a, 1983b; Whittlesea, 1983), and pronunciation tasks (Glushko, 1979) all seem to be quite sensitive to the congruity between particular training stimuli and particular test stimuli, in ways which most abstraction or prototype formation models would not expect.

One response to this dual situation has been to propose models in which apparent rule-based or concept-based behavior is attributed to a process that makes use of stored traces of specific events or specific exemplars of the concepts or rules. According to this class of models, the apparently rule-based or concept-based behavior emerges from what might be called a conspiracy of individual memory traces or from a sampling of one from the set of such traces. Models of this class include the Medin and Schaffer (1978) context model, the Hintzman (1983) multiple trace model, and the Whittlesea (1983) episode model.

This trend is also exemplified by our interactive activation model of word perception and an extension of the interactive activation model to generalization from exemplars (McClelland, 1981). Both of these models were described in Chapter 1.

One feature of some of these exemplar models is very troublesome. Many of them are internally inconsistent with respect to the issue of abstraction. Thus, though our word perception model assumes that linguistic rules emerge from a conspiracy of partial activations of detectors for particular words, thereby eliminating the need for abstraction of rules, the assumption that there is a single detector for each word implicitly assumes that there is an abstraction process that lumps each occurrence of the same word into the same single detector unit. Thus, the model has its abstraction and creates it too, though at somewhat different levels.

One logically coherent response to this inconsistency is to simply say that each word or other representational object is itself the result of a conspiracy of the entire ensemble of memory traces of the different individual experiences we have had with the object. We will call this view the *enumeration of specific experiences* view. It is exemplified most clearly by Jacoby (1983a, 1983b), Hintzman (1983), and Whittlesea (1983).

As the papers just mentioned demonstrate, enumeration models can work quite well as an account of quite a number of empirical findings. However, there still seems to be a drawback. Enumeration models seem to require an unlimited amount of storage capacity, as well as mechanisms for searching an almost unlimited mass of data. This is especially true when we consider that the primitives out of which we normally assume each experience is built are themselves based on generalizations. For example, a word is a sequence of letters, and a sentence is a sequence of words. Are we to believe that all these units are mere notational conveniences for the theorist, and that every event is stored separately as an extremely rich (obviously structured) representation of the event, with no condensation of details?

Thus, the dilemma remains. It would appear that enumeration models cannot completely eliminate the need for some kind of abstractive representation in memory. Yet the evidence that the characteristics of particular events influence memory performance cannot be disposed of completely.

One response to this dilemma is to propose the explicit storage of both general and specific information. For example, Elio and J. R. Anderson (1981) have proposed just such a model. In this model, memory traces are stored in the form of productions, and mechanisms of production generalization and differentiation form summary representations and refine them as necessary during learning.

Distributed models suggest another alternative. As we shall demonstrate in this chapter, a very simple distributed model, relying on a very simple learning rule—the delta rule—is capable of accounting for empirical data that has been taken as suggesting that we store summary representations (e.g., prototypes) as well as data that has been taken as suggesting that memory consists of an enumeration of specific experiences.

The specific model we consider does have a limitation that it shares with enumeration models: It relies on a fixed set of representational primitives. However, we shall argue that natural extensions of the model which overcome this limitation are now possible. Though we do not explore these extensions in detail, we do consider some issues that will arise if these extensions are incorporated.

A DISTRIBUTED MODEL OF MEMORY

The model we shall describe is a member of the class of models discussed in Chapter 3. In developing the ideas presented here, we were strongly influenced by the work of J. A. Anderson (e.g., 1977, 1983; Anderson, Silverstein, Ritz, & Jones, 1977; Knapp & Anderson, 1984) and Hinton (1981a). We have adopted and synthesized what we found to be the most useful aspects of their distinct but related models, preserving (we hope) the basic spirit of both.

In keeping with the overall enterprise of the book, our distributed model is a model of the microstructure of memory. It specifies the internal workings of some of the components of information processing and memory, in particular those concerned with the retrieval and use of prior experience. The model does not specify in and of itself how these acts of retrieval and use are planned, sequenced, and organized into coherent patterns of behavior. Some thoughts about ways this might be done may be found in Chapters 8 and 14.

General Properties

Our model shares a number of basic assumptions about the nature of the processing and memory system with most other distributed models. In particular, the processing system is assumed to consist of a highly interconnected network of units that take on activation values and communicate with other units by sending signals modulated by weights associated with the connections between the units, according to the principles laid out in Chapter 2. Sometimes we may think of the units

as corresponding to particular representational primitives, but they need not. For example, even what we might consider to be a primitive feature of something, like having a particular color, might be a pattern of activation over a collection of units.

Modular structure. We assume that the units are organized into modules. Each module receives inputs from other modules; the units within the module are richly interconnected with each other; and they send outputs to other modules. Figure 1 illustrates the internal structure of a very simple module, and Figure 2 illustrates some hypothetical interconnections between a number of modules. Both figures grossly under-represent our view of the numbers of units per module and the number of modules. We would imagine that there would be thousands to millions of units per module and many hundred or perhaps many thousand partially redundant modules in anything close to a complete memory system.

The state of each module represents a synthesis of the states of all of the modules it receives inputs from. Some of the inputs will be from relatively more sensory modules, closer to the sensory end-organs of

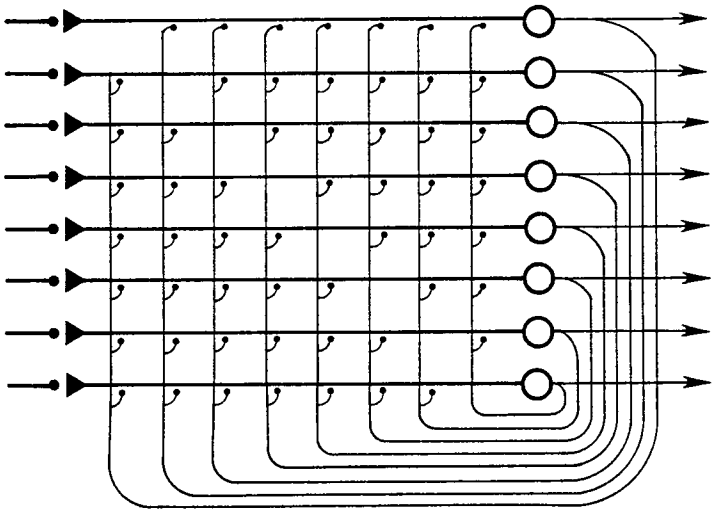

FIGURE 1. A simple information processing module, consisting of a small ensemble of eight processing units. Each unit receives inputs from other modules (indicated by the single input impinging on the input line of the unit from the left; this can stand for a number of converging input signals from several units outside the module) and sends outputs to other modules (indicated by the rightward output line from each unit). Each unit also has a modifiable connection to all the other units in the module, as indicated by the branches of the output lines that loop back onto the input lines leading into each unit.

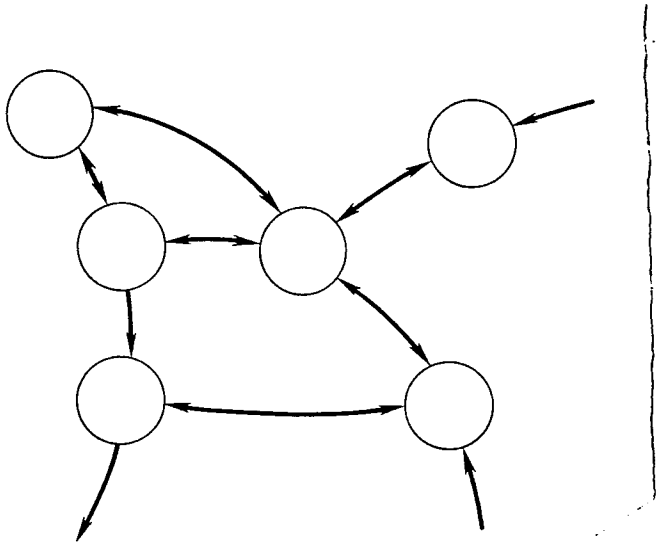

FIGURE 2. An illustrative diagram showing several modules and interconnections among them. Arrows between modules simply indicate that some of the units in one module send inputs to some of the units in the other. The exact number and organization of modules is, of course, unknown; the figure is simply intended to be suggestive.

one modality or another. Others will come from relatively more abstract modules, which themselves receive inputs from and send outputs to other modules placed at the abstract end of several different modalities. Thus, each module combines a number of different sources of information.

Units play specific roles within patterns. A pattern of activation only counts as the same as another if the same units are involved. The reason for this is that the knowledge built into the system for re-creating the patterns is built into the set of interconnections among the units, as we will explain below. For a pattern to access the right knowledge, it must arise on the appropriate units. In this sense the units play specific roles in the patterns.

Obviously, a system of this sort is useless without sophisticated perceptual processing mechanisms at the interface between memory and the outside world, so that input patterns arising at different locations in the world can be mapped into the same set of units internally. The scheme proposed by Hinton (1981b; see Chapter 4) is one such mechanism. Chapter 16 describes mechanisms that allow several different patterns to access the same set of units at the same time; in the present chapter, we restrict attention to the processing of one pattern at a time.

Relation to Basic Concepts in Memory

Several standard concepts in the memory literature map easily onto corresponding aspects of distributed memory models. Here we consider three key concepts in memory and their relation to our distributed memory model.

Mental state as pattern of activation. In a distributed memory system, a mental state is a pattern of activation over the units in some subset of the modules. The patterns in the different modules capture different aspects of the content of the mental states in a kind of a partially overlapping fashion. Alternative mental states are simply alternative patterns of activation over the modules. Information processing is the process of evolution in time of mental states.

Memory traces as changes in the weights. Patterns of activation come and go, leaving traces behind when they have passed. What are the traces? They are changes in the strengths or weights of the connections between the units in the modules.[1] As we already said, each memory trace is distributed over many different connections, and each connection participates in many different memory traces. The traces of different mental states are therefore superimposed in the same set of weights.

Retrieval as reinstatement of prior pattern of activation. Retrieval amounts to partial reinstatement of a mental state, using a cue which is a fragment of the original state. For any given module, we can see the cues as originating from outside of it. Some cues could arise ultimately from sensory input. Others would arise from the results of previous retrieval operations, fed back to the memory system under the control of a search or retrieval plan. It would be premature to speculate on how such schemes would be implemented in this kind of a model, but it is clear that they must exist.

Detailed Assumptions

In the rest of our presentation, we will be focusing on operations that take place within a single module. This obviously oversimplifies the behavior of a complete memory system since the modules are assumed

[1] These traces are not, of course, to be confused with the continuing pattern of activation stored in the Trace processing structure discussed in Chapter 15.

to be in continuous interaction. The simplification is justified, however, in that it allows us to examine some of the basic properties of distributed memory systems which are visible even without these interactions with other modules.

Let us look, therefore, at the internal structure of one very simple module, as shown in Figure 1. Again, our image is that in a system sufficient for real memories, there would be much larger numbers of units. We have restricted our analysis to small numbers simply to illustrate basic principles as clearly as possible; this also helps to keep the running time of simulations in bounds.

Continuous activation values. The units used in the present model are fairly standard units, taking on continuous activation values in the range from -1 to $+1$. Zero is in this case a neutral resting value, toward which the activations of the units tend to decay.

Unlike the models described in some of the previous chapters in this section, there is no threshold activation value. This means that both positive and negative activations influence other units.

Inputs, outputs, and internal connections. Each unit receives input from other modules and sends output to other modules. For simplicity, we assume that the inputs from other modules occur at connections whose weights are fixed. In the simulations, we also treat the input from outside the module as a static pattern that comes on suddenly, ignoring for simplicity the fact that the input pattern evolves in time and might be affected by feedback from the module under study. While the input to each unit might arise from a combination of sources in other modules, we can lump the external input to each unit into a single real valued number representing the combined effects of all components of the external input. In addition to extramodular connections, each unit is connected to all other units in the module via a weighted connection. The weights on these connections are modifiable, as described below. The weights can take on any real values: positive, negative, or zero. There is no connection from a unit onto itself.

An input pattern is presented at some point in time over some or all of the input lines to the module and is then left on for several ticks, until the pattern of activation it produces settles down and stops changing.

The processing cycle. The model is a synchronous model, like all of the other models in this section. On each processing cycle, each unit determines its net input, based on the external input to the unit and the activations of all of the units at the end of the preceding cycle modulated by the weight coefficients which determine the strength and

direction of each unit's effect on every other unit. Then the activations of all of the units are updated simultaneously.

The net input to a unit consists of two parts, the *external* input, arising from outside the module, and the *internal* input, arising from the other units in the module. The internal input to unit i, i_i, is then just the sum of all of these separate inputs, each weighted by the appropriate connection strength:

$$i_i = \sum_j a_j w_{ij}.$$

Here, j ranges over all units in the module other than i, a_j is the activation of unit j at the end of the previous cycle, and w_{ij} is the weight of the connection to unit i from unit j. This sum is then added to the *external* input to the unit, arising from outside the module, to obtain the net input to unit i, net_i:

$$net_i = i_i + e_i$$

where e_i is just the lumped external input to unit i.

Activations are updated according to a simple nonlinear "squashing" function like the one we have used in several other chapters. If the net input is positive, the activation of the unit is incremented by an amount proportional to the distance left to the ceiling activation level of +1.0. If the net input is negative, the activation is decremented by an amount proportional to the distance left to the floor activation level of −1.0. There is also a decay factor which tends to pull the activation of the unit back toward the resting level of 0. For unit i, if $net_i > 0$,

$$\Delta a_i = Enet_i(1 - a_i) - Da_i. \tag{1}$$

If $net_i \leqslant 0$,

$$\Delta a_i = Enet_i(a_i - (-1)) - Da_i. \tag{2}$$

In these equations, E and D are global parameters that apply to all units and set the rates of excitation and decay, respectively. The term a_i is the activation of unit i at the end of the previous cycle, and Δa_i is the change in a_i; that is, it is the amount added to (or, if negative, subtracted from) the old value a_i to determine its new value for the next cycle.

The details of these assumptions are quite unimportant to the behavior of the model. Many of the same results have been obtained using other sigmoid squashing functions. The fact that activations range from −1 to +1 is, likewise, not very relevant, though if activations ranged from 0 to 1 instead, it would be necessary to incorporate threshold terms in addition to the connection strengths to get the model to produce roughly the same results.

Given a fixed set of inputs to a particular unit, its activation level will be driven up or down in response until the activation reaches the point where the incremental effects of the input are balanced by the decay. In practice, of course, the situation is complicated by the fact that as each unit's activation is changing it alters the input to the others. Thus, it is necessary to run the simulation to see how the system will behave for any given set of inputs and any given set of weights. In all the simulations reported here, the model is allowed to run for 50 cycles, which is considerably more than enough for it to achieve a stable pattern of activation over all the units.

Memory traces. The memory trace of a particular pattern of activation is a set of changes in the entire set of weights in the module. We call the whole set of changes an *increment* to the weights. After a stable pattern of activation is achieved, weight adjustment takes place. This is thought of as occurring simultaneously for all of the connections in the module.

We use the delta rule to determine the size and direction of the change at each connection. This learning rule is explored extensively in Chapters 2, 8, and 11. Here, we consider it from the point of view of indicating how it implements, in a direct and simple way, the storage of the connection information that will allow a module to re-create complete patterns of activation originally produced by external input when only a part of the pattern is presented as a "retrieval" cue.

To achieve pattern completion, we want to set up the internal connections among the units in the module so that when part of the pattern is presented, the internal connections will tend to reproduce the rest. Consider, in this light, a particular pattern of activation, and assume for the moment that in this pattern the external input to a particular unit j is excitatory. Then we will want the internal input from other units in the module to tend to excite unit j when they take on the activation values appropriate for this particular pattern. In general, what we need to do for each unit is to adjust the weights so that the internal connections in the module will tend to reproduce the external input to that unit, given that the rest of the units have the activation values appropriate for this particular pattern.

The first step in weight adjustment is to see how well we are already doing. If the network is already doing what it should, the weights do not need to be changed. Therefore, for each unit i, we compute the difference δ_i between the external input to the unit and the net internal input to the unit from other units in the module:

$$\delta_i = e_i - i_i.$$

In determining the activation value of the unit, we added the external input together with the internal input. Now, in adjusting the weights, we are taking the difference between these two terms. This implies that the unit must be able to aggregate all inputs for purposes of determining its activation, but it must be able to distinguish between external and internal inputs for purposes of adjusting its weights.

Let us consider the term δ_i for a moment. If it is positive, the internal input is not activating the unit enough. If negative, it is activating the unit too much. If it is zero, everything is fine and we do not want to change anything. Thus, δ_i determines the magnitude and direction of the overall change that needs to be made in the internal input to unit i. To achieve this overall effect, the individual weights are then adjusted according to the following formula:

$$\Delta w_{ij} = \eta \delta_i a_j. \tag{3}$$

The parameter η is just a global strength parameter which regulates the overall magnitude of the adjustments of the weights; Δw_{ij} is the *change* in the weight to i from j.

The delta rule, given by Equation 3, has all the intended consequences. That is, it tends to drive the weights to the right values to make the internal inputs to a unit match the external inputs. For example, consider the case in which δ_i is positive and a_j is positive. In this case, the value of δ_i tells us that unit i is not receiving enough excitatory input, and the value of a_j tells us that unit j has positive activation. In this case, the delta rule will increase the weight from j to i. The result will be that the next time unit j has a positive activation, its excitatory effect on unit i will be increased, thereby reducing δ_i. Similar reasoning applies to cases where δ_i is negative, a_j is negative, or both are negative. Of course, when either δ_i or a_j is zero, w_{ij} is not changed. In the first case, there is no error to compensate for; in the second case, a change in the weight will have no effect the next time unit j has the same activation value.

What the delta rule can and cannot do. Several important points about the delta rule have been discussed in Chapters 2, 8, and 11. Here we consider just those that are most relevant to our present purposes. Basically, the most important result is that, for a set of patterns that we present repeatedly to a module, if there is a set of weights that will allow the system to reduce δ to 0 for each unit in each pattern, this rule will find it through repeated exposure to all of the members of the set of patterns. However, it is important to note that the existence of a set of weights that will allow δ to be reduced to 0 is not guaranteed, but depends on the structure inherent in the set of patterns that the model

is given to learn. To be perfectly learnable by our model, the patterns must conform to the following *linear predictability constraint*:

> Over the entire set of patterns, the external input to each unit must be predictable from a linear combination of the activations of every other unit.

While this limitation is severe, it is important to realize that the extent to which a set of patterns satisfy the linear predictability constraint depends on the way the set of patterns is represented. From this point of view, we must distinguish clearly between the theoretical description of a set of stimuli presented to a human subject for processing and the patterns of activation produced in some module deeply embedded in the cognitive system. As a rule of thumb, an encoding which treats each dimension or aspect of a stimulus separately is unlikely to be sufficient; what is required is a *context sensitive* or *conjunctive* encoding, such that the representation of each aspect is colored by other aspects. A fuller discussion of this issue is presented in Chapter 3; we also return to it below in considering some recent evidence obtained by Medin and Schwanenflugel (1981).

No hidden units. As explained in Chapters 7 and 8, the linear predictability constraint arises from the fact that the present model contains no hidden units. If hidden units were incorporated, the generalized delta rule described in Chapter 8 could be used to train the connections into these units, thereby allowing the model to form new representational primitives to overcome the linear predictability constraint when it arises. At the end of the chapter, we consider the introduction of hidden units and the effects that this would have on the behavior of the model.

Decay in the increments to the weights. We assume that each trace or increment undergoes a decay process, though the rate of decay of the increments is assumed to be much slower than the rate of decay of patterns of activation. Following a number of theorists (e.g., Wickelgren, 1979), we imagine that traces at first decay rapidly, but then the remaining portion becomes more and more resistant to further decay. Whether it ever reaches a point where it is no longer decaying at all, we do not know. The basic effect of this assumption is that individual inputs exert large short-term effects on the weights, but, after they decay, the residual effect is considerably smaller. The fact that each increment has its own temporal history increases the complexity of computer simulations enormously. In many of the simulations, therefore, we will specify simpler assumptions to keep the simulations tractable.

Parameters and details. In simulations using the model, it is important to keep the net input to each unit on each processing cycle relatively small, so that activations do not jump too suddenly and go out of range. For this purpose we set the parameters E and D equal to .15. It is also important to reduce η in proportion to the number of internal inputs to each unit. The number of internal inputs to each unit is just 1 minus the number of units, since each unit receives an input from every other unit. Thus, if we define $\eta = S/(n-1)$, where n is the number of units, then the rate of learning, in terms of the reduction in δ, will be about the same for all values of n. Instability can result if S is set greater than 1.0. Generally a value of .85 was used in the following simulations.

Illustrative Examples

In this section, we describe a number of simulations to illustrate several key aspects of the model's behavior. We wish to demonstrate several points:

1. The model can extract what appears to be the prototype or central tendency of a set of patterns, if the patterns are in fact random distortions of the same base or prototype pattern.

2. The model can do this for several *different* patterns, using the same set of connections to store its knowledge of all the prototypes.

3. This ability does not depend on the exemplars being presented with labels so that the model is given the where-with-all to keep them straight.

4. Representations of specific, repeated exemplars can coexist in the same set of connections with knowledge of the prototype.

Learning a prototype from exemplars. To illustrate the first point, we consider the following hypothetical situation. A little boy sees many different dogs, each only once and each with a different name. All the dogs are a little different from each other, but in general there is a pattern which represents the typical dog—each one is just a different distortion of this prototype. (We are not claiming that the dogs in the world have no more structure than this; we make this assumption for

purposes of illustration only). For now we will assume that the names of the dogs are all completely different. We would expect, given this experience, that the boy would learn the prototype of the category, even without ever seeing any particular dog that matches the prototype directly (Posner & Keele, 1968, 1970; J. A. Anderson, 1977, applies an earlier version of a distributed model to this case). That is, the prototype will seem as familiar as any of the exemplars, and the boy will be able to complete the pattern corresponding to the prototype from any part of it. He will not, however, be very likely to remember the names of each of the individual dogs, though he may remember the most recent ones.

We model this situation with a module consisting of 24 units. We assume that the presentation of a dog produces a visual pattern of activation over 16 of the units in the hypothetical module (the 9th through 24th, counting from left to right). The name of the dog produces a pattern of activation over the other 8 units (Units 1 to 8, counting from left to right).

Each visual pattern, by assumption, is a distortion of a single prototype. The prototype used for the simulation simply had a random series of $+1$ and -1 values. Each distortion of the prototype was made by probabilistically flipping the sign of randomly selected elements of the prototype pattern. For each new distorted pattern, each element has an independent chance of being flipped, with probability .2. Each name pattern was simply a random sequence of $+1$s and -1s for the eight name units. Each encounter with a new dog is modeled as a presentation of a new name pattern with a new distortion of the prototype visual pattern. Fifty different trials were run, each with a new name-pattern/visual-pattern pair.

For each presentation, the pattern of activation is allowed to stabilize, and then the weights are adjusted as described above. The increment to the weights is then allowed to decay considerably before the next input is presented. For simplicity, we assume that before the next pattern is presented, the last increment decays to a fixed small proportion (5%) of its initial value and thereafter undergoes no further decay.

What does the model learn? The module acquires a set of weights which is continually buffeted about by the latest dog exemplar, but which captures the prototype dog quite well. Waiting for the last increment to decay to the fixed residual yields the weights shown in Figure 3.

These weights capture the correlations among the values in the prototype dog pattern quite well. The lack of exact uniformity is due to the more recent distortions presented, whose effects have not been corrected by subsequent distortions. This is one way in which the model gives priority to specific exemplars, especially recent ones. The

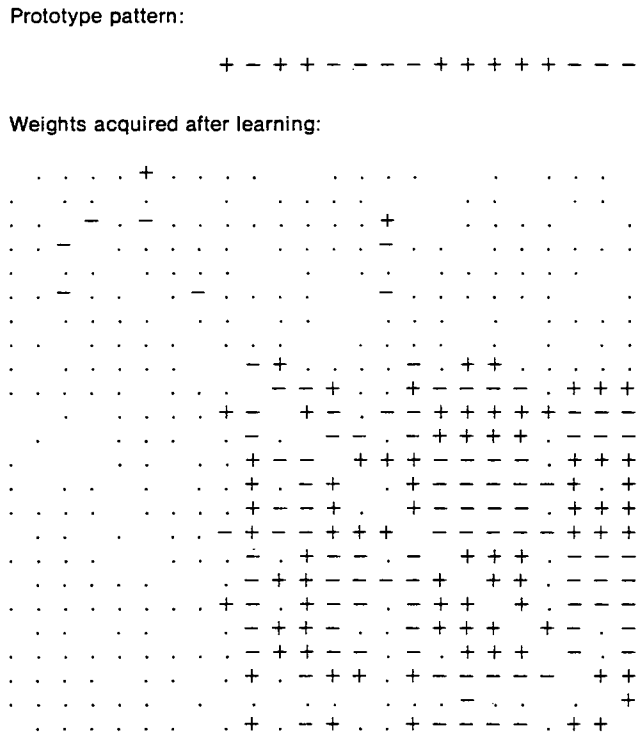

FIGURE 3. Weights acquired in learning from distorted exemplars of a prototype. (The prototype pattern is shown above the weight matrix. Blank entries correspond to weights with absolute values less than .01; dots correspond to absolute values less than .06; plusses or minuses are used for weights with larger absolute values.)

effects of recent exemplars are particularly strong, of course, before they have had a chance to decay. The module can complete the prototype quite well, and it will respond more strongly to the prototype than to any distortion of it. It has, however, learned no particular relation between this prototype and any name pattern since a totally different random association was presented on each trial. If the pattern of activation on the name units had been the same in every case (say, each dog was just called "dog"), or even in just a reasonable fraction of the cases, then the module would have been able to retrieve this shared name pattern from the prototype of the visual pattern and the prototype pattern from the name; we will see cases of this kind of behavior in the next section.

Multiple, nonorthogonal prototypes. In the preceding simulation, we have seen how the distributed model acts as a sort of signal averager,

finding the central tendency of a set of related patterns. In and of itself this is an important property of the model, but the importance of this property increases when we realize that the model can average several *different* patterns in the same composite memory trace. Thus, several different prototypes can be stored in the same set of weights. This property is important because it means that the model does not fall into the trap of needing to decide which category to put a pattern into before knowing which prototype to average it with. The acquisition of the different prototypes proceeds without any sort of explicit categorization. If the patterns are sufficiently dissimilar (i.e., orthogonal), there is no interference among them at all. Increasing similarity leads to increased confusability during learning, but eventually the delta rule finds a set of connection strengths that minimizes the confusability of similar patterns. These points are discussed mathematically in Chapter 11; here we illustrate this through a simulation of the following hypothetical situation.

Let us suppose that our little boy sees different dogs, different cats, and different bagels in the course of his day-to-day experience. First, let's consider the case in which each experience with a dog, a cat, or a bagel is accompanied by someone saying "dog," "cat," or "bagel," as appropriate..

The simulation analog of this situation involved forming three "visual" prototype patterns of 16 elements: two of them (the one for dog and the one for cat) somewhat similar to each other ($r = .5$) and the third (for the bagel) orthogonal to both of the other two. Paired with each visual pattern was a name pattern of eight elements. Each name pattern was orthogonal to both of the others. Thus, the prototype visual pattern for cat and the prototype visual pattern for dog were similar to each other, but their names were not related.

Stimulus presentations involved presentations of distorted exemplars of the name/visual pattern pairs to a module like the one used in the previous simulation. This time, both the name pattern and the visual pattern were distorted, with each element having its sign flipped with an independent probability of .1 on each presentation. Fifty different distortions of each name/visual pattern pair were presented in groups of three, consisting of one distortion of the dog pair, one distortion of the cat pair, and one distortion of the bagel pair. Weight adjustment occurred after each presentation, with decay to a fixed residual before each new presentation.

At the end of training, the module was tested by presenting each name pattern and observing the resulting pattern of activation over the visual units, and by presenting each visual pattern and observing the pattern of activation over the name units. The results are shown in Table 1. In each case, the model reproduces the correct completion for

TABLE 1

RESULTS OF TESTS AFTER LEARNING THE DOG, CAT, AND BAGEL PATTERNS

	Name Pattern	Visual Pattern
Pattern for dog prototype	+ − + − + − + −	+ − + + − − − − + + + + + − − −
Response to dog name		+3 −4 +4 +4 −4 −4 −4 −4 +4 +4 +4 +3 +4 −4 −4 −3
Response to dog visual pattern	+5 −4 +4 −5 +5 −4 +4 −4	
Pattern for cat prototype	+ + − − + + − −	+ − + + − − − − + − + − + + − +
Response to cat name		+4 −3 +4 +4 −4 −3 −3 −4 +4 −4 +4 −4 +4 +4 −4 +4
Response to cat visual pattern	+5 +4 −4 −5 +4 +4 −4 −4	
Pattern for bagel prototype	+ − − + + − − +	+ + − + − + + − + − − + + + + −
Response to bagel name		+3 +4 −4 +4 −4 +4 +4 −4 +4 −4 −4 +4 +3 +4 +4 −4
Response to bagel visual pattern	+4 −4 −4 +4 +4 −4 −4 +4	

Note: Decimal points have been suppressed for clarity; thus, an entry of +4 represents an activation value of +.4.

the probe, and there is no apparent contamination of the "cat" pattern by the "dog" pattern, even though the visual patterns are both similar.

In general, pattern completion is a matter of degree. Below, in simulating particular experimental results, we will introduce an explicit measure of the degree to which a particular pattern is active in the units of a module. For now, it is sufficient simply to note that the sign of all of the elements is correct; given this, the average magnitude of the elements gives an approximate measure of the "degree" of pattern reinstatement.

In a case like the present one, in which the patterns known to the model are not all orthogonal, the values of the connection strengths that the model produces do not necessarily have a simple interpretation. Though their sign always corresponds to the sign of the correlation between the activations of the two patterns, their magnitude is not a simple reflection of the magnitude of their correlation, but is influenced by the degree to which the model is relying on this particular correlation to predict the activation of one unit from the others. Thus, in a case where two units (call them i and j) are perfectly correlated, the strength of the connection from i to j will depend on the number of other units whose activations are correlated with j. If i is the only unit correlated with j, it will have to do all the work of "predicting" j, so the weight will be very strong; on the other hand, if many units besides i are correlated with j, then the work of exciting j will be spread around, and the weight between i and j will be considerably smaller. The situation is exactly the same as the one that arises in linear regression: if several variables predict another, they share the weight. The weight matrix acquired as a result of learning the dog, cat, and bagel patterns (Figure 4) reflects these effects. For example, across the set of three prototypes, Units 1 and 5 are perfectly correlated, as are Units 2 and 6. Yet the connection from Unit 2 to Unit 6 is stronger than the connection from Unit 1 to Unit 5 (these connections are *d in the figure). The reason for the difference is that Unit 2 is one of only three units which correlate perfectly with Unit 6, while Unit 1 is one of seven units which correlate perfectly with Unit 5.[2]

Thus far we have seen that several prototypes, not necessarily orthogonal, can be stored in the same module without difficulty. It is true, though we do not illustrate it, that the model has more trouble with the cat and dog visual patterns earlier on in training, before learning has essentially reached asymptotic levels, as it has by the end of 50

[2] In Table 1, the weights do not reflect these contrasts perfectly in every case because the noise introduced into the learning happens, by chance, to alter some of the correlations present in the prototype patterns. Averaged over time, though, the weights will conform to their expected values.

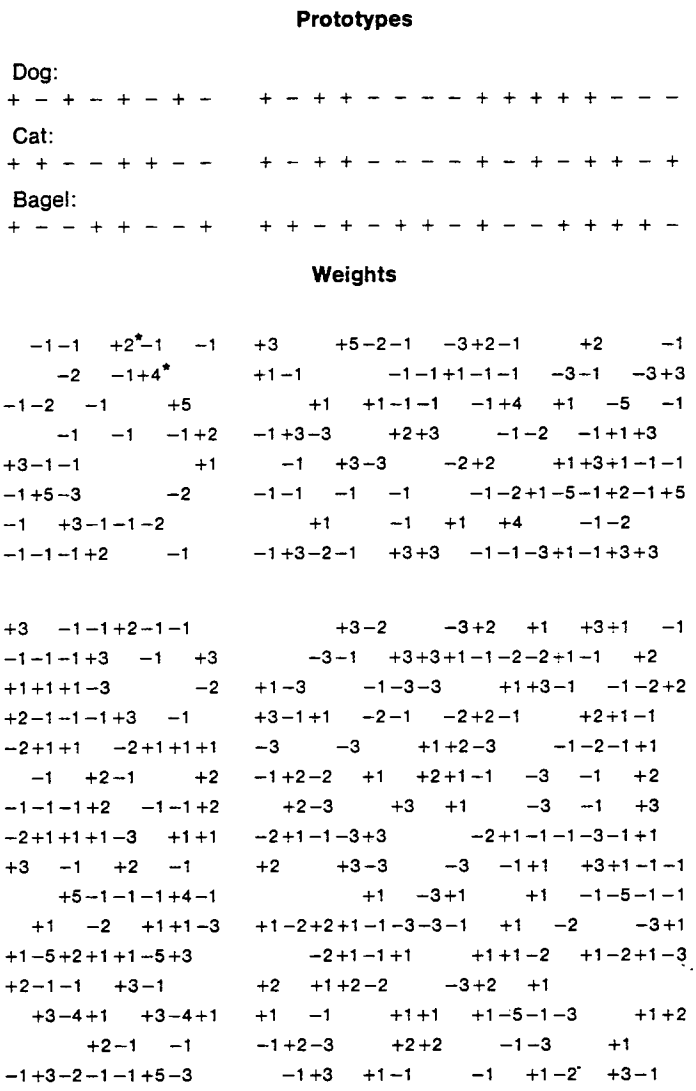

FIGURE 4. Weights acquired in learning the three prototype patterns shown. (Blanks in the matrix of weights correspond to weights with absolute values less than or equal to +5 stands for a weight of +.25. The gap in the horizontal and vertical dimensions is used to separate the name field from the visual pattern field.)

cycles through the full set of patterns. And, of course, even at the end of learning, if we present as a probe a part of the visual pattern that does not differentiate between the dog and the cat, the model will produce a blended response. Both of these aspects of the model seem generally consistent with what we should expect from human subjects.

Category learning without labels. An important further fact about the model is that it can learn several different visual patterns, even without the benefit of distinct identifying name patterns during learning. To demonstrate this we repeated the previous simulation, simply replacing the name patterns with 0s. The model still learns about the internal structure of the visual patterns so that, after 50 cycles through the stimuli, any unique subpart of any one of the patterns is sufficient to reinstate the rest of the corresponding pattern correctly. This aspect of the model's behavior is illustrated in Table 2. Thus, we have a model that can, in effect, acquire a number of distinct categories, simply through a process of incrementing connection strengths in response to each new stimulus presentation. Noise, in the form of distortions in the patterns, is filtered out. The model does not require a name or other guide to distinguish the patterns belonging to different categories.

Coexistence of the prototype and repeated exemplars. One aspect of our discussion up to this point may have been slightly misleading. We may have given the impression that the model is simply a prototype extraction device. It is more than this, however; it is a device that captures whatever structure is present in a set of patterns (subject, of course, to the linear predictability constraint). When the set of patterns has a prototype structure, the model will act as though it is extracting prototypes; but when it has a different structure, the model will do its best to accommodate this as well. For example, the model permits the coexistence of representations of prototypes with representations of particular, repeated exemplars.

TABLE 2

RESULTS OF TESTS AFTER LEARNING
THE DOG, CAT, AND BAGEL PATTERNS WITHOUT NAMES

Dog visual pattern:	+ − + + − − − − + + + + + − − −
Probe:	+ + + +
Response:	+3 −3 +3 +3 −3 −4 −3 −3 +6 +5 +6 +5 +3 −2 −3 −2
Cat visual pattern:	+ − + + − − − − + − + − + + − +
Probe:	+ − + −
Response:	+3 −3 +3 +3 −3 −3 −3 −3 +6 −5 +6 −5 +3 +2 −3 +2
Bagel visual pattern:	+ + − + − + + − + − − + + + + −
Probe:	+ − − +
Response:	+2 +3 −4 +3 −3 +3 +3 −3 +6 −6 −6 +6 +3 +3 +3 −3

As an illustration of this point, consider the following situation. Let us say that our little boy knows a dog next door named Rover and a dog at his grandma's house named Fido. And let's say that the little boy goes to the park from time to time and sees dogs, each of which his father tells him is a dog.

The simulation analog of this involved three different eight-element name patterns, one for Rover, one for Fido, and one for dog. The visual pattern for Rover was a particular randomly generated distortion of the dog prototype pattern, as was the visual pattern for Fido. For the dogs seen in the park, each one was simply a new random distortion of the prototype. The probability of flipping the sign of each element was again .2. The learning regime was otherwise the same as in the dog–cat–bagel example.

At the end of 50 learning cycles, the model was able to retrieve the visual pattern corresponding to either repeated exemplar (see Table 3) given the associated name as input. When given the dog name pattern as input, it retrieves the prototype visual pattern for dog. It can also retrieve the appropriate name from each of the three visual patterns. This is true, even though the visual pattern for Rover differs from the visual pattern for dog by only a single element. Because of the special importance of this particular element, the weights from this element to the units that distinguish Rover's name pattern from the the prototype name pattern are quite strong. Given part of a visual pattern, the model will complete it; if the part corresponds to the prototype, then that is what is completed, but if it corresponds to one of the repeated exemplars, then that exemplar is completed. The model, then, knows both the prototype and the repeated exemplars quite well. Several other sets of prototypes and their repeated exemplars could also be stored in the same module, as long as its capacity is not exceeded; given large numbers of units per module, a lot of different patterns can be stored.

Let us summarize the observations we have made in these several illustrative simulations. First, our distributed model is capable of storing not just one but a number of different patterns. It can pull the "central tendency" of a number of different patterns out of the noisy inputs; it can create the functional equivalent of perceptual categories with or without the benefit of labels; and it can allow representations of repeated exemplars to coexist with the representation of the prototype of the categories they exemplify in the same composite memory trace. The model is not simply a categorizer or "prototyping" device; rather, it captures the structure inherent in a set of patterns, whether it be characterizable by description in terms of prototypes or not, as long as the ensemble of patterns adheres to the linear predictability constraint.

TABLE 3

RESULTS OF TESTS WITH PROTOTYPE AND SPECIFIC EXEMPLAR PATTERNS

	Name Pattern	Visual Pattern
Pattern for dog prototype	+ − + − + − + −	+ − + + − − − − + + + + + − − −
Response to prototype name		+4 −5 +3 +3 −4 −3 −3 −3 +3 +3 +4 +3 +4 −3 −4 −4
Response to prototype visual pattern	+5 −4 +4 −4 +5 −4 +4 −4	
Pattern for Fido exemplar	+ − − − + − − −	+ − (−) + − − − − + + + + + (+) − −
Response to Fido name		+4 −4 −4 +4 −4 −4 −4 −4 +4 +4 +4 +4 +4 +4 −4 −4
Response to Fido visual pattern	+5 −5 −3 −5 +4 −5 −3 −5	
Pattern for Rover exemplar	+ − − + + + − +	+ (+) + + − − − − + + + + + − − −
Response to Rover name		+4 +5 +4 +4 −4 −4 −4 −4 +4 +4 +4 +4 +4 −4 −4 −4
Response to Rover visual pattern	+4 −4 −2 +4 +4 +4 −2 +4	

The ability to retrieve accurate completions of similar patterns is a property of the model that depends on the use of the delta learning rule. This allows both the storage of different prototypes that are not orthogonal and the coexistence of prototype representations and repeated exemplars.

SIMULATIONS OF EXPERIMENTAL RESULTS

Up to this point, we have discussed our distributed model in general terms and have outlined how it can accommodate both generalization and representation of specific information in the same network. We now consider, in the next two sections, how well the model does in accounting for some recent evidence about the details of the influence of specific experiences on performance and the conditions under which functional equivalents of summary representations such as logogens and prototypes emerge.

Repetition and Familiarity Effects

When we perceive an item—say a word, for example—this experience has effects on our later performance. If the word is presented again within a reasonable interval of time, the prior presentation makes it possible for us to recognize the word more quickly or from a briefer presentation.

Traditionally, this effect has been interpreted in terms of units that represent the presented items in memory. In the case of word perception, these units are called *word detectors* or *logogens* and a model of repetition effects for words has been constructed around the logogen concept (Morton, 1979). The idea is that the threshold for the logogen is reduced every time it "fires" (that is, every time the word is recognized), thereby making it easier to fire the logogen at a later time. There is supposed to be a decay of this priming effect, with time, so that eventually the effect of the first presentation wears off.

This traditional interpretation has come under serious question of late, for a number of reasons. Perhaps paramount among the reasons is the fact that the exact relation between the specific context in which the priming event occurs and the context in which the test event occurs makes a huge difference (Jacoby, 1983a, 1983b). Generally speaking, nearly any change in the stimulus—from spoken to printed, from male speaker to female speaker, etc.—tends to reduce the magnitude of the priming effect.