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

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 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. Copyright 1985 by the American Psychological Association, Inc. Adapted with permission.

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 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.

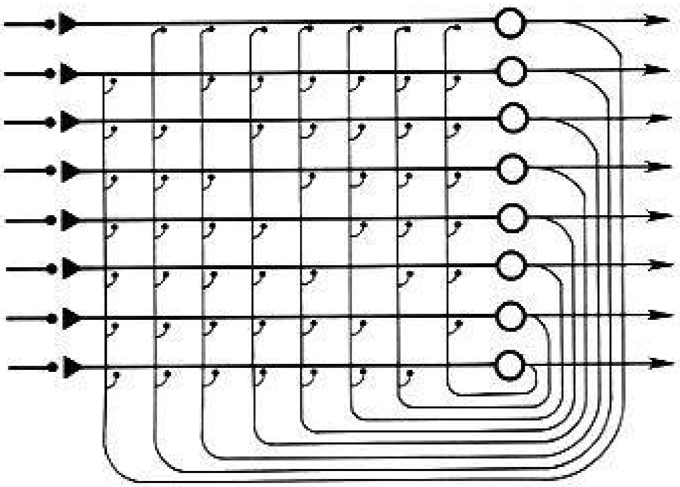

FIGURE 1. A simple information processing module, consisting of a small ensemble of eight processing units. Each unit receives inputs from other modules (indicated by the single input impinging on the input line of the unit from the left; this can stand for a number of converging input signals from several 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.

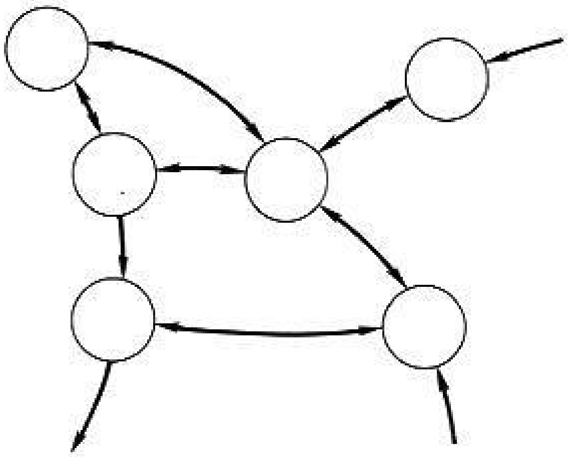

FIGURE 2. An illustrative diagram showing several modules and interconnections among them. Arrows between modules simply indicate that some of the units in one module send inputs to some of the units in the other. The exact number and organization of modules is, of course, unknown; the figure is simply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

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 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

[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.

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 +1s 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

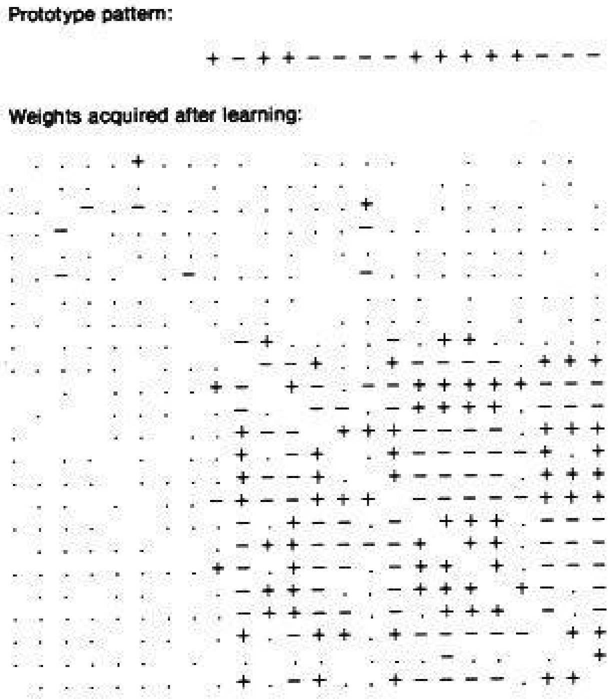

FIGURE 3. Weights acquired in learning from distorted exemplars of a prototype. (The prototype pattern is shown above the weight matrix. Blank entries correspond to weights with absolute values less than .01; dots correspond to absolute values less than .06; plusses or minuses are used for weights with 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 Figure 4, 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.

Prototypes

Dog:

+ − + − + − + −   + − + + − − − − + + + + + − − −

Cat:

+ + − − + + − −   + − + + − − − − + − + − + + − +

Bagel:

+ − − + + − − +   + + − + − + + − + − − + + + + −

Weights

	−1	−1		+2*	−1		−1	+3			+5	−2	−1		−3	+2	−1			+2			−1
		−2		−1	+4*			+1	−1				−1	−1	+1	−1	−1		−3	−1		−3	+3
−1	−2		−1			+5				+1		+1	−1	−1		−1	+4		+1		−5		−1
		−1		−1		−1	+2	−1	+3	−3			+2	+3			−1	−2		−1	+1	+3	
+3	−1	−1					+1		−1		+3	−3			−2	+2			+1	+3	+1	−1	−1
−1	+5	−3				−2		−1	−1		−1		−1			−1	−2	+1	−5	−1	+2	−1	+5
−1		+3	−1	−1	−2					+1			−1		+1		+4			−1	−2		
−1	−1	−1	+2			−1		−1	+3	−2	−1		+3	+3		−1	−1	−3	+1	−1	+3	+3	
+3		−1	−1	+2	−1	−1					+3	−2			−3	+2		+1		+3	+1		−1
−1	−1	−1	+3		−1		+3			−3	−1		+3	+3	+1	−1	−2	−2	+1	−1		+2	
+1	+1	+1	−3				−2	+1	−3			−1	−3	−3			+1	+3	−1		−1	−2	+2
+2	−1	−1	−1	+3		−1		+3	−1	+1		−2	−1		−2	+2	−1			+2	+1	−1	
−2	+1	+1		−2	+1	+1	+1	−3			−3			+1	+2	−3			−1	−2	−1	+1	
	−1		+2	−1			+2	−1	+2	−2		+1		+2	+1	−1		−3		−1		+2	
−1	−1	−1	+2		−1	−1	+2		+2	−3			+3		+1			−3		−1		+3	
−2	+1	+1	+1	−3		+1	+1	−2	+1	−1	−3	+3				−2	+1	−1	−1	−3	−1	+1	
+3		−1		+2		−1		+2			+3	−3			−3		−1	+1		+3	+1	−1	−1
		+5	−1	−1	−1	+4	−1					+1		−3	+1			+1		−1	−5	−1	−1
	+1		−2		+1	+1	−3	+1	−2	+2	+1	−1	−3	−3	−1		+1		−2			−3	+1
+1	−5	+2	+1	+1	−5	+3				−2	+1	−1	+1			+1	+1	−2		+1	−2	+1	−3
+2	−1	−1		+3	−1			+2		+1	+2	−2			−3	+2		+1					
	+3	−4	+1		+3	−4	+1	+1		−1			+1	+1		+1	−5	−1	−3			+1	+2
			+2	−1		−1		−1	+2	−3			+2	+2			−1	−3			+1		
−1	+3	−2	−1	−1	+5	−3			−1	+3		+1	−1			−1		+1	−2		+3	−1	

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.

These facts might easily be taken to support the enumeration of specific experiences view, in which the logogen is replaced by the entire ensemble of experiences with the word, with each experience capturing aspects of the specific context in which it occurred. Such a view has been championed most strongly by Jacoby (1983a, 1983b).

Our distributed model offers an alternative interpretation. We see the traces laid down by the processing of each input as contributing to the composite, superimposed memory representation. Each time a stimulus is processed, it gives rise to a slightly different memory trace—either because the item itself is different or because it occurs in a different context that conditions its representation. The logogen is replaced by the set of specific traces, but *the traces are not kept separate.* Each trace contributes to the composite, but the characteristics of particular experiences tend nevertheless to be preserved, at least until they are overridden by canceling characteristics of other traces. Also, the traces of one stimulus pattern can coexist with the traces of other stimuli, within the same composite memory trace.

It should be noted that we are not faulting either the logogen model or models based on the enumeration of specific experiences for their physiological implausibility here, since these models are generally not stated in physiological terms, and their authors might reasonably argue that nothing in their models precludes distributed storage at a physiological level. What we are suggesting is that a model which proposes explicitly distributed, superpositional storage can account for the kinds of findings that logogen models have been proposed to account for, as well as other findings which strain the utility of the concept of the logogen as a psychological construct.

To illustrate the distributed model's account of repetition priming effects, we carried out the following simulation experiment. We made up a set of eight random vectors, each 24 elements long, each one to be thought of as the prototype of a different recurring stimulus pattern. Through a series of 10 training cycles using the set of eight vectors, we constructed a composite memory trace. During training, the model did not actually see the prototypes, however. On each training presentation it saw a new random distortion of one of the eight prototypes. In each of the distortions, each of the 24 elements had its value flipped with a probability of .1. Weights were adjusted after every presentation and were then allowed to decay to a fixed residual before the presentation of the next pattern.

The composite memory trace formed as a result of this experience plays the same role in our model that the set of logogens or detectors play in a model like Morton's or, indeed, the interactive activation model of word perception. That is, the trace contains information which allows the model to enhance perception of familiar patterns, relative to unfamiliar ones. We demonstrate this by comparing the activations resulting from the processing of subsequent presentations of new distortions of our eight familiar patterns, compared to other random patterns with which the model is not familiar. The pattern of activation that is the model's response to the input is stronger and grows to a particular level more quickly if the stimulus is a new distortion of an old pattern than if it is a new pattern. This effect is illustrated in Figure 5.

Pattern activation and response strength. The measure of activation shown in the figure is the dot product of the pattern of activation over the units of the module with the stimulus pattern itself, normalized for the number n of elements in the pattern: For the pattern p we call this expression α_p. In mathematical notation it is just

$$\alpha_p = \frac{1}{n}\sum_i a_i e_{pi}(t)$$

where i indexes the units in the module, and e_{pi} indexes the external input to unit i in pattern p. Essentially, α represents the degree to which the actual pattern of activation on the units captures the input pattern. It is an approximate analog of the activation of an individual unit in models that allocate a single unit to each whole pattern.

To relate these pattern activations to response probabilities, we must assume that mechanisms exist for translating patterns of activation into

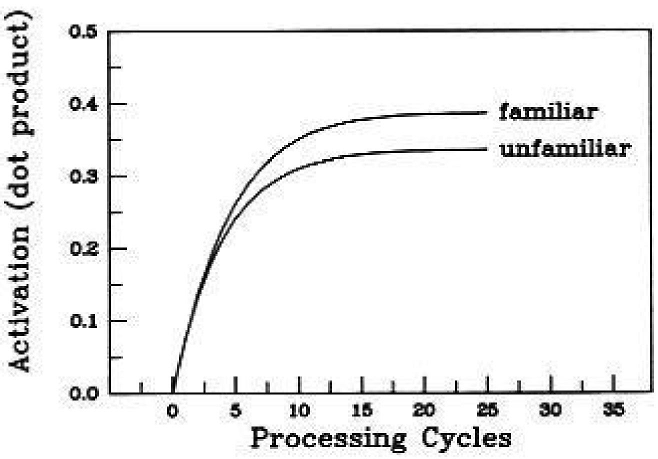

FIGURE 5. Growth of the pattern of activation for new distortions of familiar and unfamiliar patterns. The measure of the strength of the pattern of activation is the dot product of the response pattern with the input vector. See text for an explanation.

overt responses measurable by an experimenter. We will assume that these mechanisms obey the same principles discussed in Chapter 15 for relating activations to response probabilities, simply replacing the activations of particular units with the α measure of pattern activation.[3]

These assumptions finesse an important issue, namely, the mechanism by which a pattern of activation gives rise to a particular response. A specific mechanism for response generation is described in Chapter 18. For now, we wish only to capture basic properties any actual response selection mechanism must have: It must be sensitive to the input pattern, and it must approximate other basic aspects of response selection behavior captured by the Luce (1963) choice model.

Effects of experimental variables on time-accuracy curves. Applying the assumptions described above, we can calculate probability of correct response as a function of processing cycles for familiar and unfamiliar patterns. The result for a particular choice of scaling parameters is shown in Figure 6. If we assume that performance in a perceptual identification task is based on the height of the curve at the point where processing is cut off by masking (McClelland & Rumelhart, 1981), then familiarity would lead to greater accuracy of perceptual identification at a given exposure duration. In a reaction time task, if the response is emitted when its probability reaches a particular threshold activation value, familiarity would lead to speeded responses. Thus, the model is consistent with the ubiquitous influence of familiarity both on response accuracy and speed, in spite of the fact that it has no detectors for familiar stimuli.

But what about priming and the role of congruity between the prime event and the test event? To examine this issue, we carried out a second experiment. Following learning of eight patterns as in the previous experiment, new distortions of half of the random vectors previously learned by the model were presented as primes. For each of these primes, the pattern of activation was allowed to stabilize, and changes in the strengths of the connections in the model were then made. We then tested the model's response to (a) the same four distortions, (b) four new distortions of the same patterns, and (c) distortions of the four previously learned patterns that had not been

[3] One complication arises due to the fact that it is not, in general, possible to specify exactly what the set of alternative responses might be for the denominator of the Luce choice rule used in the word perception model. For this reason, the strengths of other responses are represented by a constant C (which stands for the competition). Thus, the expression for probability of choosing the response appropriate to pattern p is just $p(r_p) = (e^{k\bar{\alpha}_p})/(C + e^{k\bar{\alpha}_p})$ where $\bar{\alpha}_p$ represents the time average of α_p, and k is a scaling constant.

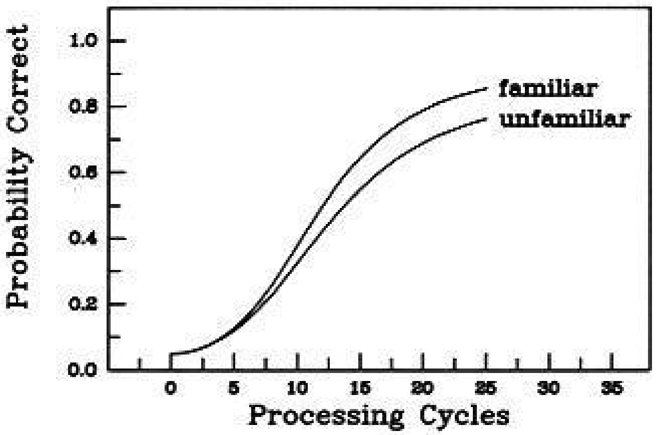

FIGURE 6. Simulated growth of response accuracy over the units in a 24-unit module, as a function of processing cycles, for new distortions of previously learned patterns compared to new distortions of patterns not previously learned.

presented as primes. There was no decay in the weights over the course of the priming experiment; if decay had been included, its main effect would have been to reduce the magnitude of the priming effects.

The results of the experiment are shown in Figure 7. The response of the model is greatest for the patterns preceded by identical primes, intermediate for patterns preceded by similar primes, and weakest for patterns not preceded by any related prime.

Our model, then, appears to provide an account, not only for the basic existence of priming effects, but also for the graded nature of priming effects as a function of congruity between prime event and test event. It avoids the problem of multiplication of context-specific detectors which logogen theories fall prey to, while at the same time avoiding enumeration of specific experiences. Congruity effects are captured in the composite memory trace.

The model also has another advantage over the logogen view. It accounts for repetition priming effects for unfamiliar as well as familiar stimuli. When a pattern is presented for the first time, a trace is produced just as it would be for stimuli that had previously been presented. The result is that, on a second presentation of the same pattern or a new distortion of it, processing is facilitated. The functional equivalent of a logogen begins to be established from the very first presentation.

To illustrate the repetition priming of unfamiliar patterns and to compare the results with the repetition priming we have already observed for familiar patterns, we carried out a third experiment. This time, after learning eight patterns as before, a priming session was run, in which new distortions of four of the familiar patterns and distortions of four new patterns were presented. Then, in the test phase, 16 stimuli were presented: New distortions of the primed, familiar patterns; new distortions of the unprimed, familiar patterns; new distortions of the primed, previously unfamiliar patterns; and finally, new distortions of four patterns that were neither primed nor familiar. The results are shown in Figure 8. What we find is that long-term familiarity and recent priming have approximately additive effects on the asymptotes of the time-accuracy curves. The time to reach any given activation level shows a mild interaction, with priming having slightly more of an effect for unfamiliar than for familiar stimuli.

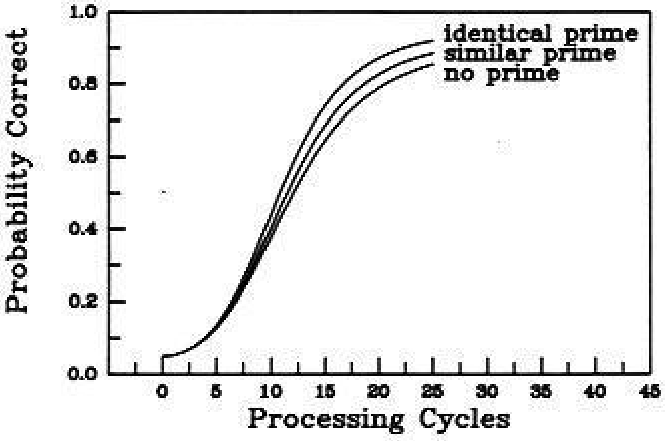

FIGURE 7. Response probability as a function of exposure time, for patterns preceded by identical primes, similar primes, or no related prime.

These results are consistent with the bulk of the findings concerning the effects of pre-experimental familiarity and repetition in a recent series of experiments by Feustel, Shiffrin, and Salasoo (1983) and Salasoo, Shiffrin, and Feustel (1985). They found that pre-experimental familiarity of an item (word vs. nonword) and prior exposure had this very kind of interactive effect on exposure time required for accurate identification of all the letters of a string, at least when words and nonwords were mixed together in the same lists of materials.

A further aspect of the results reported by Salasoo, Shiffrin, and Feustel is also consistent with our approach. In one of their experiments, they examined the threshold for accurate identification as a function of number of prior presentations, for both words and

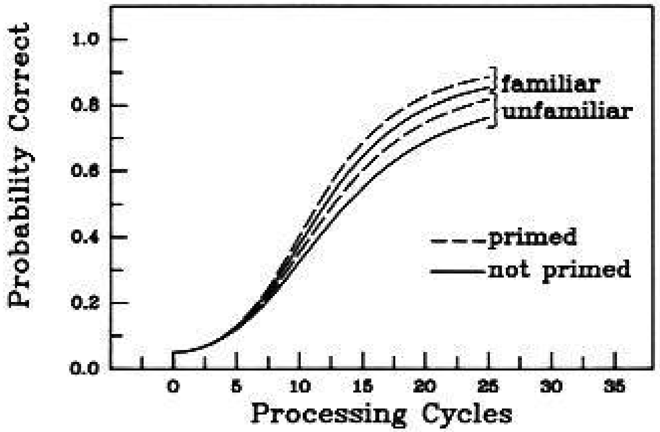

FIGURE 8. Response to new distortions of primed, familiar patterns; unprimed, familiar patterns; primed, unfamiliar patterns; and unprimed, unfamiliar patterns.

pseudowords. While thresholds were initially elevated for pseudowords, relative to words, there was a rather rapid convergence of the thresholds over repeated presentations, with the point of convergence coming at about the same place on the curve for two different versions of their perceptual identification task (Salasoo et al., 1985). Our model, likewise, shows this kind of convergence effect, as illustrated in Figure 9.

There is one finding by Salasoo et al. (1985) that appears at first glance to support the view that there is some special process of unit formation that is distinct from the priming of old units. This is the fact that after a year between training and testing, performance with pseudowords used during training is indistinguishable from performance with words, but performance with words used during training shows no residual benefit compared to words not previously used. The data certainly are consistent with the view that training experience made the pseudowords into lasting perceptual units at the same time that it produced transitory priming of existing units. We have not attempted to account for this finding in detail, but we doubt that it is inconsistent with a distributed model. In support of this, we offer one reason why repetition effects might seem to persist longer for pseudowords rather than for words in the Salasoo et al. experiment. For pseudowords, a strong association would be built up between the item and the learning context during initial training. Such associations would be formed for words, but because these stimuli have been experienced many times before and have already been well learned, smaller increments in connection strengths are formed for these stimuli during training, and thus the strength of the association between the item and the learning context would be less.

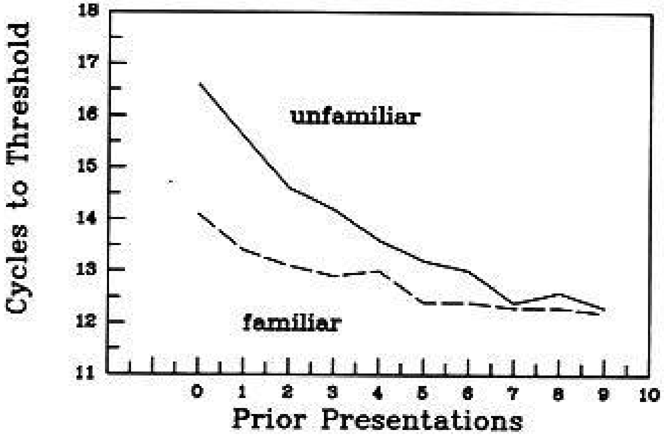

FIGURE 9. Time to reach a fixed accuracy criterion (60% correct) for previously familiar and unfamiliar patterns, as a function of repetitions.

If our interpretation is correct, we would expect to see a disadvantage for pseudowords relative to words if the testing were carried out in a situation which did not reinstate the mental state associated with the original learning experience since for these stimuli much of what was learned would be tied to the specific learning context. Such a prediction would appear to differentiate our account from any view that postulated the formation of an abstract, context-independent logogen as the basis for the absence of a pseudoword decrement effect.

Representation of General and Specific Information

In the previous section, we cast our distributed model as an alternative to the view that familiar patterns are represented in memory either by separate detectors or by an enumeration of specific experiences. In this section, we show that the model provides alternatives to both abstraction and enumeration models of learning from exemplars of prototypes.

Abstraction models were originally motivated by the finding that subjects occasionally appeared to have learned better how to categorize the prototype of a set of distorted exemplars than the specific exemplars they experienced during learning (Posner & Keele, 1968). However, pure abstraction models have never fared very well since there is nearly always evidence of some superiority of the particular training stimuli over other stimuli equally far removed from the prototype. A favored model, then, is one in which there is both abstraction and memory for particular training stimuli.

Recently, proponents of models involving only enumeration of specific experiences have noted that such models can account for the basic fact that abstraction models are primarily designed to account for—enhanced response to the prototype, relative to particular previously seen exemplars, under some conditions—as well as failures to obtain such effects under other conditions (Hintzman, 1983; Medin & Schaffer, 1978). In evaluating distributed models, then, it is important to see if they can do as well. J. A. Anderson (1977) has made important steps in this direction, and Knapp and J. A. Anderson (1984) have shown how their distributed model can account for many of the details of the Posner-Keele experiments. Recently, however, two sets of findings have been put forward that appear to strongly favor the enumeration of specific experiences view, at least relative to pure abstraction models. It is important, therefore, to see how well our distributed model can do in accounting for these effects.

The first set of findings comes from a set of studies by Whittlesea (1983). In a large number of studies, Whittlesea demonstrated a role for specific exemplars in guiding performance on a perceptual identification task. We wanted to see whether our model would demonstrate a similar sensitivity to specific exemplars. We also wanted to see whether our model would account for the conditions under which such effects are not obtained.

Whittlesea used letter strings as stimuli. The learning experiences subjects received involved simply looking at the stimuli one at a time on a visual display and writing down the sequence of letters presented. Subjects were subsequently tested for the effect of this training on their ability to identify letter strings bearing various relationships to the training stimuli and to the prototypes from which the training stimuli were derived. The test was a perceptual identification task; the subject was simply required to try to identify the letters from a brief flash.

The stimuli Whittlesea used were all distortions of one of two prototype letter strings. Table 4 illustrates the essential properties of the sets of training and test stimuli he used. The stimuli in Set Ia were each one step away from the prototype. The Ib items were also one step from the prototype and one step from one of the Ia distortions. The Set IIa stimuli were each two steps from the prototype and one step from a particular Ia distortion. The Set IIb items were also two steps from the prototype, and each was one step from one of the IIa distortions. The Set IIc distortions were two steps from the prototype also, and each was two steps from the closest IIa distortion. Over the set of five IIc distortions, the A and B subpatterns each occurred once in each position, as they did in the case of the IIa distortions. The distortions in Set III were three steps from the prototype and one step from the closest member of Set IIa. The distortions in Set V were each five steps from the prototype.

TABLE 4

SCHEMATIC DESCRIPTION OF STIMULUS SETS
USED IN SIMULATIONS OF WHITTLESEA'S EXPERIMENTS

Prototype	Ia	Ib	IIa	IIb	IIc	III	V
PPPPP	APPPP	BPPPP	ABPPP	ACPPP	APCPP	ABCPP	CCCCC
	PAPPP	PBPPP	PABPP	PACPP	PAPCP	PABCP	CBCBC
	PPAPP	PPBPP	PPABP	PPACP	PPAPC	PPABC	BCACB
	PPPAP	PPPBP	PPPAB	PPPAC	CPPAP	CPPAB	ABCBA
	PPPPA	PPPPB	BPPPA	CPPPA	PCPPA	BCPPA	CACAC

Note: The actual stimuli used can be filled in by replacing P with +—+—; A with ++——; B with +——+; and C with ++++. The model is not sensitive to the fact the same subpattern was used in each of the 5 slots.

Whittlesea ran seven experiments using different combinations of training and test stimuli. We carried out simulation analogs of all of these experiments plus one additional experiment that Whittlesea did not run. The main difference between the simulation experiments and Whittlesea's actual experiments was that he used two different prototypes in each experiment, while we only used one.

The simulation employed a simple 20-unit module. The set of 20 units was divided into five submodules, one for each letter in Whittlesea's letter strings. The prototype pattern and the different distortions used can be derived from the information provided in Table 4.

Each simulation experiment began with null connections between the units. The training phase involved presenting the set or sets of training stimuli analogous to those Whittlesea used, for the same number of presentations. To avoid idiosyncratic effects of particular orders of training stimuli, each experiment was run six times, each with a different random order of training stimuli. On each trial, activations were allowed to settle down through 50 processing cycles, and then connection strengths were adjusted. There was no decay of the increments to the weights over the course of an experiment.

In the test phase, the model was tested with the sets of test items analogous to the sets Whittlesea used. As a precaution against effects of prior test items on performance, we simply turned off the adjustment of weights during the test phase.

A summary of the training and test stimuli used in each of the experiments, of Whittlesea's findings, and of the simulation results are shown in Table 5. The numbers represent relative amounts of enhancement in performance as a result of the training experience, relative to a pretest baseline. For Whittlesea's data, this is the per-letter increase in letter identification probability between a pre- and posttest. For the simulation, it is the increase in the size of the dot product for a pretest with null weights and a posttest after training. For comparability to the data, the dot-product difference scores have been doubled. This is simply a scaling operation to facilitate qualitative comparison of experimental and simulation results.

TABLE 5

SUMMARY OF PERCEPTUAL IDENTIFICATION EXPERIMENTS WITH EXPERIMENTAL AND SIMULATION RESULTS

Whittlesea's Experiment Number	Training Stimulus Set(s)	Test Stimulus Sets	Experimental Results	Simulation Results
1	Ia	Ia	.27	.24
		Ib	.16	.15
		V	.03	-.05
2	IIa	IIa	.30	.29
		IIc	.15	.12
		V	.03	-.08
3	IIa	IIa	.21	.29
		IIb	.16	.14
		IIc	.10	.12
4	IIa	P	-	.24
		Ia	.19	.21
		IIa	.23	.29
		III	.15	.15
4'	Ia	P	-	.28
		Ia	-	.24
		IIa	-	.12
5	IIa,b,c	P	-	.25
		Ia	.16	.21
		IIa	.16	.18
		III	.10	.09
6	III	Ia	.16	.14
		IIa	.16	.19
		III	.19	.30
7	IIa	IIa	.24	.29
		IIc	.13	.12
		III	.17	.15

A comparison of the experimental and simulation results shows that wherever there is a within-experiment difference in Whittlesea's data, the simulation produced a difference in the same direction. (Between-experiment comparisons are not considered because of subject and material differences which render such differences unreliable). The next several paragraphs review some of the major findings in detail.

Some of the comparisons bring out the importance of congruity between particular test and training experiences. Experiments 1, 2, and 3 show that when distance of test stimuli from the prototype is controlled, similarity to particular training exemplars makes a difference both for the human subject and in the model. In Experiment 1, the relevant contrast was between Ia and Ib items. In Experiment 2, it was between IIa and IIc items. Experiment 3 shows that the subjects and the model both show a gradient in performance with increasing distance of the test items from the nearest old exemplar.

Experiments 4, 4', and 5 explore the status of the prototype and other test stimuli closer to the prototype than any stimuli actually shown during training. In Experiment 4, the training stimuli were fairly far away from the prototype, and there were only five different training stimuli (the members of the IIa set). In this case, controlling for distance from the nearest training stimuli, test stimuli closer to the prototype showed more enhancement than those farther away. (Ia vs. III comparison). However, the actual training stimuli nevertheless had an advantage over both other sets of test stimuli, including those that were closer to the prototype than the training stimuli themselves (IIa vs. Ia comparison).

In experiment 4' (not run by Whittlesea), the same number of training stimuli were used as in Experiment 4, but these were closer to the prototype. The result is that the simulation shows an advantage for the prototype over the old exemplars. The specific training stimuli used, even in this experiment, do influence performance, however, as Whittlesea's first experiment (which used the same training set) shows (Ia-Ib contrast). This effect holds both for the subjects and for the simulation. The pattern of results is similar to the findings of Posner and Keele (1968), in the condition where subjects learned six exemplars which were rather close to the prototype. In this condition, their subjects' categorization performance was most accurate for the prototype, but more accurate for old than for new distortions, just as in this simulation experiment.

In Experiment 5, Whittlesea demonstrated that a slight advantage for stimuli closer to the prototype than the training stimuli would emerge, even with high-level distortions, when a large number of different distortions were used once each in training, instead of a smaller number of distortions presented three times each. The effect was rather small in Whittlesea's case (falling in the third decimal place in the per-letter enhancement effect measure) but other experiments have produced similar results, and so does the simulation. In fact, since the prototype was tested in the simulation, we were able to demonstrate a monotonic drop in performance with distance from the prototype in this experiment.

Experiments 6 and 7, which used small numbers of training exemplars rather far from the prototype, both explore in different ways the relative influence of similarity to the prototype and similarity to the set of training exemplars. Both in the data and in the model, similarity to particular training stimuli is more important than similarity to the prototype, given the sets of training stimuli used in these experiments.

Taken together with other findings, Whittlesea's results show clearly that similarity of test items to particular stored exemplars is of paramount importance in predicting perceptual performance. Other experiments show the relevance of these same factors in other tasks, such as recognition memory, classification learning, etc. It is interesting to note, though, that performance does not honor the specific exemplars so strongly when the training items are closer to the prototype. Under such conditions, performance is superior on the prototype or stimuli closer to the prototype than the training stimuli. Even when the training stimuli are rather distant from the prototype, they produce a benefit for stimuli closer to the prototype if there are a large number of distinct training stimuli each shown only once. Thus, the dominance of specific training experiences is honored only when the training experiences are few and far between. Otherwise, an apparent advantage for the prototype, though with some residual benefit for particular training stimuli, is the result.

The congruity of the results of these simulations with experimental findings underscores the applicability of distributed models to the question of the nature of the representation of general and specific information. In fact, we were somewhat surprised by the ability of the model to account for Whittlesea's results, given the fact that we did not rely on context-sensitive encoding of the letter-string stimuli. That is, the distributed representation we assigned to each letter was independent of the other letters in the string. A context-sensitive encoding would, however, prove necessary to capture a larger ensemble of stimuli.

Whether a context-sensitive encoding would produce the same or slightly different results depends on the exact encoding. The exact degree of overlap of the patterns of activation produced by different distortions of the same prototype determines the extent to which the model will tend to favor the prototype relative to particular old exemplars. The degree of overlap, in turn, depends upon the specific assumptions made about the encoding of the stimuli. However, the general form of the results of the simulation would be unchanged: When all the distortions are close to the prototype, or when there is a very large number of different distortions, the central tendency will produce the strongest response; but when the distortions are fewer and farther from the prototype, the training exemplars themselves will produce the strongest activations. What the encoding would effect is the similarity metric.

In this regard, it is worth mentioning another finding that may appear to challenge our distributed account of what is learned through repeated experiences with exemplars. This is the finding of Medin and Schwanenflugel (1981). Their experiment compared learning of two different sets of stimuli in a categorization task. One set of stimuli could be categorized by a linear combination of weights assigned to particular values on each of four dimensions considered independently. The other set of stimuli could not be categorized in this way. The experiment demonstrated clearly that linear separability was not necessary for categorization learning. Linearly separable stimuli were less easily learned than a set of stimuli that were not linearly separable but had a higher degree of similarity between exemplars within categories.

At first glance, it may seem that Medin and Schwanenflugel's experiment is devastating to our distributed approach since our distributed model can only learn linear combinations of weights. However, whether a linear combination of weights can suffice in the Medin and Schwanenflugel experiments depends on how patterns of activation are assigned to stimuli. If each stimulus dimension is encoded separately in the representation of the stimulus, then the Medin and Schwanenflugel stimuli cannot be learned by our model. But if each stimulus dimension is encoded in a context sensitive way, then the patterns of activation associated with the different stimuli become linearly separable again.

One way of achieving context sensitivity is via separate enumeration of traces. But it is well known that there are other ways as well. Several different kinds of context-sensitive encodings which do not require separate enumeration of traces or the allocation of separate units to individual experiences are considered in Chapter 3 and in Chapters 18 and 19.

It should be noted that the motivation for context-sensitive encoding in the use of distributed representations is captured by, but by no means limited to, the kinds of observations reported in the experiment by Medin and Schwanenflugel. Context-sensitive encoding is required if distributed models are to be able to overcome the linear predictability constraint, and this constraint pervades both computational and psychological applications of the idea of distributed representations.

EXTENSIONS OF THE MODEL

There are a number of phenomena in the learning and memory literature that lend themselves to accounts in terms of distributed models of memory. Here we will give a brief list. In some cases, the phenomena are addressed in other chapters, and we merely give pointers. In other cases, we have not yet had an opportunity to carry out detailed simulations; in these cases we describe briefly how we might envision encompassing the phenomena in terms of distributed models.

The Emergence of Semantic Memory From Episodic Traces

A distributed model leads naturally to the suggestion that semantic memory may be just the residue of the superposition of episodic traces. Consider, for example, representation of a proposition encountered in several different contexts, and assume for the moment that the context and content are represented in separate parts of the same module. Over repeated experience with the same proposition in different contexts, the proposition will remain in the interconnections of the units in the proposition submodule, but the particular associations to particular contexts will wash out. However, material that is only encountered in one particular context will tend to be somewhat contextually bound. So we may not be able to retrieve what we learn in one context when we need it in other situations. Other authors (e.g., J. R. Anderson & Ross, 1980) have recently argued against a separation between episodic and semantic memory, pointing out interactions between traditionally episodic and semantic memory tasks. Such findings are generally consistent with the view we have taken here.

Emergence of Regularities in Behavior and Their Coexistence With Exceptions

Distributed models also influence our thinking about how human behavior might come to exhibit the kind of regularity that often leads linguists to postulate systems of rules. In Chapter 18 we describe a distributed model of a system that can learn the past-tense system of English, given as inputs pairs of patterns corresponding to the phonological structure of the present and past tense forms of actual English verbs. The model accounts for several interesting phenomena which have been taken as evidence for the acquisition of "linguistic rules," such as the fact that children at a certain stage overregularize the pronunciation of irregular forms. It also accounts for the coexistence of irregular pronunciations with productive use of the regular past tense in the mature language user.

In general, then, distributed models appear to provide alternatives to a variety of models that postulate abstract, summary representations such as prototypes, logogens, semantic memory representations, or even linguistic rules.

Spared Learning in Amnesia

Distributed models also provide a natural way of accounting for spared learning by amnesics—persons who have diminished ability to learn new information after some traumatic event. There are generally two types of spared learning effects observed. First, in domains where amnesics show deficits, they nevertheless show a residual ability to learn gradually from repeated experiences. Indeed, they generally appear to be relatively more spared in their ability to extract what is common about an ensemble of experiences than in their ability to remember the details of individual events. Second, there are some domains in which most amnesics show no deficits at all. These phenomena are taken up from the point of view of a distributed model of learning and memory in Chapter 25. Here, we briefly consider the main points of the arguments made in that chapter.

First, distributed models provide a natural way of explaining why there should be residual ability to learn gradually from repeated experience even within those domains where amnesics are grossly deficient in their memory for specific episodic experiences. For if we simply imagine that the effective size of the increments to the connections is reduced in amnesia, then the general properties of distributed models—the fact that they extract the central tendency from a set of similar experiences and build up a trace of the prototype from a series of repeated exemplars—automatically provide an account of the gradual accumulation of repeated information in the face of a profound deficit in remembering any specific episode in which the information was presented. Indeed, in some cases reduced increments in the sizes of connections can actually be an advantage for pulling out the central tendency of a set of experiences since reduction in the size of the increments means that the idiosyncratic properties of recent experiences will not unduly dominate the composite memory trace.

The fact that large changes in connection strengths are not universally beneficial in distributed models may also provide a basis for explaining why certain aspects of learning and memory are completely unaffected in amnesia. The domains in which learning is almost completely spared are just the ones in which learning is very gradual in normals and is independent of awareness of having learned. These kinds of learning, we suggest, are acquired gradually by normals because large changes in the strengths of connections do not facilitate learning in such cases.

Memory Blends

Loftus (1977) has observed that subjects often combine information from two episodes when they attempt to describe some aspect of just one of these events. Many observations of this kind of effect have been observed in the context of legal testimony, but Loftus has now begun to explore the same effects in more basic experimental paradigms. The general result is that subjects do not simply report the correct property from one or the other memory trace; rather, they report a property that results from averaging or blending the properties of the individual traces. In one study, for example, many subjects reported that the color of a vehicle was green. However, no subject had actually seen a green truck; instead, they had seen a yellow truck in the original scene and a blue one in a distorted version of it. Such blend errors, obviously, are the beginnings of the formation of a summary representation and fall naturally out of distributed models.

Interference and Fan Effects

One of the oldest phenomena in the memory literature is interference. That similar memories or memories acquired in similar contexts interfere with each other was a central topic in memory research for years (see Crowder, 1976, for a review). Recently, this sort of interference effect has been explored by John Anderson, under the name of the *fan effect* (see J. R. Anderson, 1983, for a review). The fan effect refers to the fact that reaction times to indicate recognition of propositions goes up with the number of propositions studied that share arguments with the proposition under test. For example, *The boy kissed the girl* is harder to recognize if the subject has studied other sentences involving *the boy* or *the girl* than if he has not.

Distributed models like the one we have described here exhibit interference of just this kind. Singley (personal communication, 1985) has in fact simulated a simple fan-effect experiment. He trained a module of 48 units on "propositions" consisting of four parts. The parts correspond to a subject, a verb, an object, and a fourth part that served as a conjunctive representation of the three other constituents (following Hinton, 1981a). Some of the propositions shared arguments (subpatterns) with others. After several exposures to the set of patterns, Singley tested the module's response to each of the learned patterns and found that the response was generally weaker and took longer to reach a strength criterion for those patterns which shared arguments with other patterns. Obviously there is a lot more to fan effects than this; how well distributed models will be able to do in accounting for the details of the fan effect literature and other aspects of interference is a matter for further research.

AUGMENTING THE MODEL WITH HIDDEN UNITS

For all its successes, it must be said that our model does suffer from the fact that it can only learn to respond appropriately to sets of patterns that obey the linear predictability constraint. While this constraint can be overcome in specific cases by providing a conjunctive coding capable of serving as a basis for learning any prespecified set of patterns, it cannot be overcome, in general, without great cost in terms of units to cover the space of possibly necessary conjunctive representations. To overcome this limitation, it is necessary to have a model that can create the right conjunctive representation to solve the problems it faces in a way that is efficient in terms of units and connections.

Both the Boltzmann learning rule (Chapter 7) and the generalized delta rule (Chapter 8) provide ways of constructing just the required set of representations. Here we briefly sketch the application of the generalized delta rule to the case of an auto-associator with hidden units.

The basic idea is simple. A module consists of a large network of units. Some units are designated as *input/output* or simply *input* units and the remaining are designated as *hidden* units. As in the case without hidden units, the goal of the system is to have the net input from internal sources impinging on a particular input/output unit equal the external input to that unit. In this case, however, input from the hidden units is added as part of the total input from internal sources. The basic difference between a hidden unit and an input/output unit is that hidden units receive no inputs from outside the module, nor do they send outputs outside of the module. Rather, they receive inputs from other hidden units or from the input/output units. The generalized delta rule allows the system to develop a set of connections to and among the hidden units which, in principle, allows the system to retrieve the correct input pattern from *any* unique subportion of the input pattern—provided there are enough hidden units.

The "classical" example of a set of vectors for which hidden units are required is due to Hinton (1981a). This is the *one-same-one* problem. Suppose that the input vectors are divided into three portions. The first portion represents a number (either one or zero); the second portion represents a relation (either same or different); and the final portion represents a number (again either one or zero). Further, suppose that we want to store four vectors in the system representing the four propositions, *one-same-one*, *one-different-zero*, *zero-same-zero*, and *zero-different-one*. Now, we probe the system with two of the three sections of a stored input vector and hope that the memory system reconstructs the missing portion. For example, we present the system with *one-same-*? or *zero-?-one* and see what values get filled in for the unspecified portion of the vector.

For this example, learning done with the standard delta rule will not succeed. These vectors violate the linear predictability constraint. For example, suppose we present the system with *one-same-*?. In this case, the pattern associated with *one* in the first position are as often associated with *zero* in the third position as they are with *one* in that position. Thus, this pattern will try to establish a blend of *zero* and *one* in the unspecified portion of the input vector. Similarly, the pattern representing *same* in the middle portion are as often associated with the *zero* as with *one* in both other slots. Each portion will singly try to produce a blend of the two possibilities in the missing portion and both together will simply produce the same blend. The same argument can be made for each slot. However, with hidden units, the problem can be solved. A given hidden unit can relate bits from two or more slots together, and in that way it can fill in the appropriate pattern when it is missing from the input.

We have tested a network on a very simple version of this problem, using the generalized delta rule (Chapter 8) to learn the appropriate weights to and from the hidden units. In this case there were only three input/output units, one for each of the three components of the *one-same-one* propositions. The number *one* and the predicate *same* were represented by 1s in the appropriate positions, and the number *zero* and the *predicate* different were represented by 0s in the appropriate positions. Thus, the proposition *one-same-one* was represented by the input pattern 111 and the proposition *zero-different-one* was represented by pattern 001. The other two propositions were analogously represented as 010 and 100.

The basic auto-associative memory paradigm was modified slightly from the ones presented earlier in this chapter to accommodate the requirements of hidden units. Most of the modifications are not substantive, but were made to accommodate the technical requirements of the generalized delta rule. In particular, a *logistic* sigmoid "squashing" function was substituted for the one described in Equations 1 and 2. This was done because our original squashing function was not differentiable as required by the generalized delta rule (see Chapter 8 for details). This is a minor difference and has little effect on the performance of the system. Similarly, output values of units were allowed to range between [0,1] rather than [−1,1]. This change again is minor. It essentially involves a scale change and the addition of a *bias* or threshold term (again, see Chapter 8 for details). In this case 0.5 is the resting level rather than 0.0. One other somewhat more significant change of paradigm was also required. In the standard auto-associator we did not allow units to connect to themselves. The technical reason for this is that they can "learn" the input (i.e., they can make the internal input equal to the external input) simply by learning to "predict" themselves. However, this does not facilitate reconstruction of the whole pattern from a part. If the strongest connections are from a unit to itself, when that unit is not present in the probe pattern, the missing element will not be well predicted. We wished to impose the same restriction to the model with hidden units, but in this case the restriction becomes harder to enforce. That is, the system can learn to get around a constraint on direct connections by using hidden units to "shadow" each of the input units. When a particular input unit is turned on, it turns on its "shadow" hidden unit which in turn feeds activation back to the input unit. Thus we again have a unit predicting itself. Since we are most interested in the case of pattern completion, we want to avoid this situation. One way to avoid this is to require that the input from all units *except* the unit in question match the external input. To accomplish this, we used the following procedure: For each pattern to be learned, for example, 111, we actually trained the module to associate all of the patterns 11?, 1?1 and ?11 with the whole 111 pattern.

There were three input units and three hidden units. A network was set up which allowed any input unit to connect to any other input unit or to any hidden unit. Hidden units were allowed to connect to any input unit and any hidden unit—including themselves. The model was repeatedly presented with each of the three training inputs of each of the four patterns as indicated above, and the network was allowed to learn until it became stable and correctly filled in the missing parts of the patterns. The system was taught to fill in the missing portion of the pattern in three cycles and then to hold this completed pattern. Table 6 shows the state achieved by the three hidden units for each of the four input patterns. We see a distributed pattern over the hidden units in which the first unit signals that the predicate was either different or the proposition was *one-same-one*. The second hidden unit indicates that the predicate was either *same* or the proposition was *one-different-zero*, and the final hidden unit indicates that the predicate was either *different* or the proposition was *one-same-one*. This provides a unique, linearly independent basis from which to reconstruct the required input patterns.

The final network is shown in Figure 10 and the weight matrix is shown in Table 7. A number of interesting features of the network should be observed. First, since none of the input units can predict any other of the input units, the potential connections among input units have all gone to zero. Second, the hidden units all connect positively to themselves and negatively to each other. This allows a unit once turned on to stay on and shut down any noise feeding into those units that were not turned on. Finally, the connections between the input units and the hidden units are roughly symmetric. That is, whenever a particular hidden unit is turned on by a given input unit, that hidden unit tries, in turn, to turn on that input unit. It is interesting that this general kind of architecture in which units are connected roughly symmetrically and in which there are pools of mutually inhibitory units which evolved following the presentation of these very difficult patterns is very much like the architecture we have assumed in a number of our models, including the interactive activation model of word perception and the models described in Chapters 15 and 16.

TABLE 6

HIDDEN UNIT REPRESENTATIONS FOR THE ONE-SAME-ONE PROBLEM

Propositions	Input Patterns		Hidden Unit Patterns
one-same-one	111	→	111
one-different-zero	100	→	011
zero-same-zero	010	→	110
zero-different-one	001	→	101

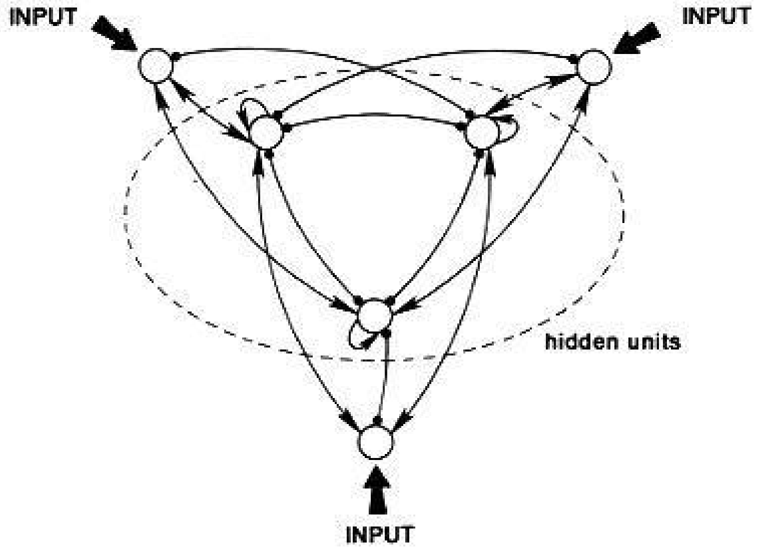

FIGURE 10. The network used in solving the *one-same-one* problem. See text for explanation.

Finally, before leaving this section, it should be noted that the hidden-units version of an auto-associator will behave very much like the system we have described throughout this chapter if the situation does not require hidden units. In this case, the connections among the

TABLE 7

WEIGHTS ACQUIRED IN LEARNING THE ONE-SAME-ONE PROBLEM

		FROM					
		Input Units			Hidden Units		
TO	Input Units	0	0	0	-6	4	4
		0	0	0	4	4	-6
		0	0	0	4	-6	4
	Hidden Units	-5	6	6	6	-3	-3
		6	6	-5	-3	6	-3
		6	-5	6	-3	-3	6

input units can capture the structure of the problem, and the connections to and from the hidden units will tend not to play an important role. This is because direct connections are learned more quickly than those which propagate information through hidden units. These will lag behind, and if they are not required, their strength will stay relatively small.

It should also be emphasized that the use of hidden units overcomes a problem that our present model shares with enumeration models: the problem of having a fixed set of representational primitives. Hidden units really are new representational primitives that grow and develop to meet the representational needs of the particular set of patterns that have been encountered. They allow the model to change its representation and thereby to change the pattern of similarity among the input patterns. We have only begun our exploration of the generalized delta rule and its application to the case of the auto-associator. We are very optimistic that it can serve as a mechanism for building the kinds of internal representations required by allowing distributed memories the flexibility of adapting to arbitrary sets of stimulus patterns.

CONCLUSION

In this chapter, we have argued that a distributed model of memory provides a natural way of accounting for the fact that memory appears to extract the central tendencies of a set of experiences and for the fact that memory is sensitive to the details of specific events and experiences. We have seen how a simple distributed model, using the delta rule, provides a fairly direct and homogeneous account for a large body of evidence relevant to the representation of general and specific information. The delta rule—and now the generalized delta rule—provide very simple mechanisms for extracting regularities from an ensemble of inputs without the aid of sophisticated generalization or rule-formulating mechanisms that oversee the performance of the processing system. These learning rules are completely local, in the sense that they change the connection between one unit and another on the basis of information that is locally available to the connection rather than on the basis of global information about overall performance. The model thus stands as an alternative to the view that learning in cognitive systems involves the explicit formulation of rules and abstractions under the guidance of some explicit overseer. We do not wish to suggest that explicit rule formation has no place in a cognitive theory of learning and memory; we only wish to suggest that such mechanisms need not

be invoked whenever behavior is observed that appears to be describable by some generalization or rule.

We emphasize that the basic properties of distributed models are largely independent of specific detailed assumptions about implementation, such as the dynamic range of the units involved or the exact form of the activation update rule. This points are underscored by the models considered in Chapters 18 and 19, in which some of the same general properties arise from models making use of somewhat different detailed assumptions.

ACKNOWLEDGMENTS

The work reported was supported in part by a grant from the System Development Foundation, in part by a contract from the Office of Naval Research (N00014-79-C-0323, NR667-437), and in part by a NIMH Career Development Award (MH00385) to the first author. Many of the ideas described here were developed and elaborated in the context of a conference organized by Lee Brooks and Larry Jacoby entitled "The Priority of the Specific," June, 1983, Elora, Ontario. The chapter owes a great deal to this conference and to stimulating discussions with the organizers and participants.