

## Distributed Memory and the Representation of General and Specific Information

James L. McClelland and David E. Rumelhart  
University of California, San Diego

We describe a distributed model of information processing and memory and apply it to the representation of general and specific information. The model consists of a large number of simple processing elements which send excitatory and inhibitory signals to each other via modifiable connections. Information processing is thought of as the process whereby patterns of activation are formed over the units in the model through their excitatory and inhibitory interactions. The memory trace of a processing event is the change or increment to the strengths of the interconnections that results from the processing event. The traces of separate events are superimposed on each other in the values of the connection strengths that result from the entire set of traces stored in the memory. The model is applied to a number of findings related to the question of whether we store abstract representations or an enumeration of specific experiences in memory. The model simulates the results of a number of important experiments which have been taken as evidence for the enumeration of specific experiences. At the same time, it shows how the functional equivalent of abstract representations—prototypes, logogens, and even rules—can emerge from the superposition of traces of specific experiences, when the conditions are right for this to happen. In essence, the model captures the structure present in a set of input patterns; thus, it behaves as though it had learned prototypes or rules, to the extent that the structure of the environment it has learned about can be captured by describing it in terms of these abstractions.

In the late 1960s and early 1970s a number of experimenters, using a variety of different tasks, demonstrated that subjects could learn through experience with exemplars of a category to respond better—more accurately, or more rapidly—to the prototype than to any of the particular exemplars. The seminal

demonstration of this basic point comes from the work of Posner and Keele (1968, 1970). Using a categorization task, they found that there were some conditions in which subjects categorized the prototype of a category more accurately than the particular exemplars of the category that they had previously seen. This work, and many other related experiments, supported the development of the view that memory by its basic nature somehow abstracts the central tendency of a set of disparate experiences, and gives relatively little weight to the specific experiences that gave rise to these abstractions.

---

Preparation of this article was supported in part by a grant from the Systems Development Foundation and in part by a National Science Foundation Grant BNS-79-24062. The first author is a recipient of a Career Development Award from the National Institute of Mental Health (5-K01-MH00385).

This article was originally presented at a conference organized by Lee Brooks and Larry Jacoby on "The Priority of the Specific." We would like to thank the organizers, as well as several of the participants, particularly Doug Medin and Rich Shiffrin, for stimulating discussion and for empirical input to the development of this article.

Requests for reprints should be sent to James L. McClelland, Department of Psychology, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213 or to David E. Rumelhart, Institute for Cognitive Science, C-015, University of California—San Diego, La Jolla, California 92093.

Recently, however, some have come to question this "abstractive" point of view, for two reasons. First, specific events and experiences clearly play a prominent role in memory and learning. 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; Medin & Shaffer, 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 models would not expect.

At the same time, a number of models have been proposed in which behavior which has often been characterized as *rule-based* or *concept-based* 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 Shaffer (1978) context model, Hintzman's (1983) multiple trace model, and Whittlesea's (1983) episode model. This trend is also exemplified by our interactive activation model of word perception (McClelland & Rumelhart, 1981; Rumelhart & McClelland, 1981, 1982), and an extension of the interactive activation model to generalization from exemplars (McClelland, 1981).

One feature of some of these exemplar-based models troubles us. 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 slightly different levels.

One logically coherent response to this inconsistency is to simply say that each word or other representational object is itself a conspiracy of the entire ensemble of memory traces of the different individual experiences we have had with that unit. 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 of specific experiences can work quite well as an account of quite a number of empirical findings. However, there still seems to be one drawback. Such 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 one experience is built are themselves abstractions. For example, a word is a sequence of letters, or a sentence is a sequence of words. Are we to believe that all of these abstractions are mere notational conveniences for the theorist, and that every event is stored as an extremely rich (obviously structured) representation of the event, with no abstraction?

In this article, we consider an alternative conceptualization: a distributed, superpositional approach to memory. This view is similar to the separate enumeration of experiences view in some respects, but not in all. On both views, memory consists of traces resulting from specific experiences; and on both views, generalizations emerge from the superposition of these specific memory traces. Our model differs, though, from the enumeration of specific experiences in assuming that the superposition of traces occurs at the time of storage. We do not keep each trace in a separate place, but rather we superimpose them so that what the memory contains is a composite.

Our theme will be to show that distributed models provide a way to resolve the abstraction-representation of specifics dilemma. With a distributed model, the superposition of traces automatically results in abstraction though it can still preserve to some extent the idiosyncrasies of specific events and experiences, or of specific recurring subclasses of events and experiences.

We will begin by introducing a specific version of a distributed model of memory. We will show how it works and describe some of its basic properties. We will show how our model can account for several recent findings (Salasoo, Shiffrin, & Feustel, 1985; Whittlesea, 1983), on the effects of specific experiences on later performance, and the conditions

under which functional equivalents of abstract representations such as prototypes or logogens emerge. The discussion considers generalizations of the approach to the semantic-episodic distinction and the acquisition of linguistic rule systems, and considers reasons for preferring a distributed-superpositional memory over other models.

*Previous, related models.* Before we get down to work, some important credits are in order. Our distributed model draws heavily from the work of 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. We view our model as an exemplar of a class of existing models whose exploration Hinton, Anderson, Kohonen (e.g., Kohonen, 1977; Kohonen, Oja, & Lehtio, 1981), and others have pioneered. A useful review of prior work in this area can be obtained from Anderson and Hinton (1981) and other articles in the volume edited by Hinton and Anderson (1981). Some points similar to some of these we will be making have recently been covered in the papers of Murdock (1982) and Eich (1982), though the distributed representations we use are different in important ways from the representations used by these other authors.

Our distributed model is not a complete theory of human information processing and memory. It is a model of the internal structure of some components of information processing, 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.

## A Distributed Model of Memory

### *General Properties*

Our model adheres to the following general assumptions, some of which are shared with several other distributed models of processing and memory.

*Simple, highly interconnected units.* The processing system consists of a collection of simple processing units, each interconnected with many other units. The units take on activation values, and communicate with other units by sending signals modulated by weights associated with the connections between the units. 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 underrepresent 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 hundreds or perhaps many thousands of 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.

*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 partially overlapping fashion. Alternative mental states are simply alternative patterns of activation over the modules. Information processing is

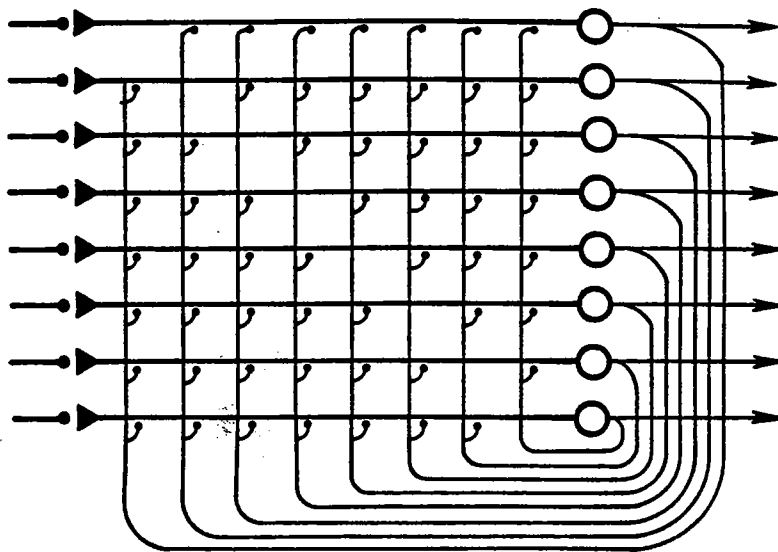


Figure 1. A simple information processing module, consisting of a small ensemble of eight processing units. [Each unit receives inputs from other modules (indicated by the single input impinging on the input line of the node from the left; this can stand for a number of converging input signals from several nodes outside the module) and sends outputs to other modules (indicated by the output line proceeding to the right from each unit). Each unit also has a modifiable connection to all the other units in the same module, as indicated by the branches of the output lines that loop back onto the input lines leading into each unit. All connections, which may be positive or negative, are represented by dots.]

the process of evolution in time of mental states.

*Units play specific roles within patterns.* A pattern of activation only counts as the same as another if the same units are involved.

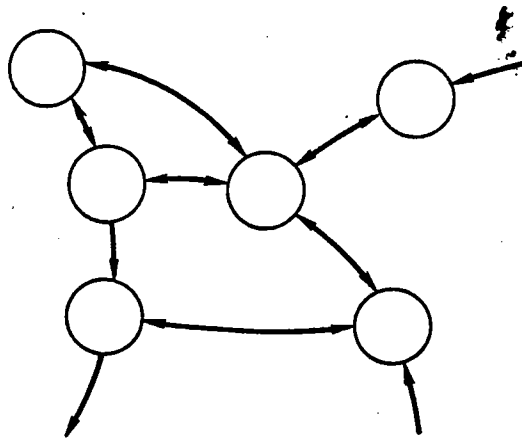


Figure 2. An illustrative diagram showing several modules and interconnections among them. (Arrows between modules simply indicate that some of the nodes in one module send inputs to some of the nodes in the other. The exact number and organization of modules is of course unknown; the figure is simply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

The reason for this is that the knowledge built into the system for recreating the patterns is built into the set of interconnections among the units, as we will explain later. 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 similar input patterns arising at different locations in the world can be mapped into the same set of units internally. Such mechanisms are outside the scope of this article (but see Hinton, 1981b; McClelland, 1985).

*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.

This view of the nature of the memory trace clearly sets these kinds of models apart from traditional models of memory in which some copy of the "active" pattern is generally thought of as being stored directly. Instead

of this, what is actually stored in our model is changes in the connection strengths. These changes are derived from the presented pattern, and are arranged in such a way that, when a part of a known pattern is presented for processing, the interconnection strengths cause the rest of the pattern to be reinstated. Thus, although the memory trace is not a copy of the learned pattern, it is something from which a replica of that pattern can be recreated. 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. Surprisingly enough, as we will see in several examples, the connections between the units in a single module can store the information needed to complete many different familiar patterns.

*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 because the modules are assumed to be in continuous interaction. The simplification is justified, however, in that it allows us to focus on some of the basic properties of distributed memory that 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 real system 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.

*Activation values.* The units take on activation values which range from  $-1$  to  $+1$ . Zero represents in this case a neutral resting value, toward which the activations of the units tend to decay.

*Inputs, outputs, and internal connections.* Each unit receives input from other modules and sends output to other modules. For the present, we assume that the inputs from other modules occur at connections whose weights are fixed. In the simulations, we treat the input from outside the module as a fixed pattern, ignoring (for simplicity) the fact that the input pattern evolves in time and might be affected by feedback from the module under study. Although 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 extra-modular 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 later. The weights can take on any real values, positive, negative, or 0. There is no connection from a unit onto itself.

*The processing cycle.* Processing within a module takes place as follows. Time is divided into discrete ticks. 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.

Each tick is divided into two phases. In the first phase, each unit determines its net input, based on the external input to the unit and activations of all of the units at the end of the preceding tick modulated by the weight coefficients which determine the strength and direction of each unit's effect on every other.

For mathematical precision, consider two units in our module, and call one of them unit  $i$ , and the other unit  $j$ . The input to unit  $i$  from unit  $j$ , written  $i_{ij}$  is just

$$i_{ij} = a_j w_{ij},$$

where  $a_j$  is the activation of unit  $j$ , and  $w_{ij}$  is the weight constant modulating the effect of unit  $j$  on unit  $i$ . The total input to unit  $i$  from all other units internal to the module,  $i_i$ , is then just the sum of all of these separate inputs:

$$i_i = \sum_j i_{ij}.$$

Here,  $j$  ranges over all units in the module other than  $i$ . This sum is then added to the external input to the unit, arising from outside the module, to obtain the net input to unit  $i$ ,  $n_i$ :

$$n_i = i_i + e_i,$$

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

In the second phase, the activations of the units are updated. 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.

Mathematically, we can express these assumptions as follows: For unit  $i$ , if  $n_i > 0$ ,

$$\dot{a}_i = En_i(1 - a_i) - Da_i.$$

If  $n_i \leq 0$ ,

$$\dot{a}_i = En_i[a_i - (-1)] - Da_i.$$

In these equations,  $E$  and  $D$  are global parameters which 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  $\dot{a}_i$  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.

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.

*The Delta rule.* The rule that determines the size and direction (up or down) of the change at each connection is the crux of the model. The idea is often difficult to grasp on first reading, but once it is understood it seems very simple, and it directly captures the goal of facilitating the completion of the pattern, given some part of the pattern as a retrieval or completion cue.

To allow each part of a pattern to reconstruct the rest of the pattern, we simply want to set up the internal connections among the units in the module so that when part of the pattern is presented, activating some of the units in the module, the internal connections will lead the active units to tend to reproduce the rest. To do this, we want to make the internal input to each unit have the same effect on the unit that the external input has on the unit. That is, given a particular pattern to be stored, we want to find a set of connections such that the internal input to each unit from all of the other units matches the external input to that unit. The connection change procedure we will describe has the effect of moving the weights of all the connections in the direction of achieving this goal.

The first step in weight adjustment is to see how well the module is already doing. If the network is already matching the external input to each unit with the internal input from the other units, the weights do not need to be changed. To get an index of how well the network is already doing at matching its excitatory input, we assume that each unit  $i$  computes the difference  $\Delta_i$  between its exter-

