CHAPTER 22

Neural and Conceptual Interpretation of PDP Models

P. SMOLENSKY

Mind and brain provide two quite different perspectives for viewing cognition. Yet both perspectives are informed by the study of parallel distributed processing. This duality creates a certain ambiguity about the interpretation of a particular PDP model of a cognitive process: Is each processing unit to be interpreted as a neuron? Is the model supposed to relate to the neural implementation of the process in some less direct way?

A closely related set of questions arises when it is observed that PDP models of cognitive processing divide broadly into two classes. In *local models*, the activity of a single unit represents the degree of participation in the processing of a known conceptual entity—a word, a word sense, a phoneme, a motor program. In *distributed models*, the strength of *patterns of activity over many units* determine the degree of participation of these conceptual entities. In some models, these patterns are chosen in a deliberately arbitrary way, so that the activity of a single unit has no apparent "meaning" whatever—no discernible relation to the conceptual entities involved in the cognitive process. On the surface, at least, these two types of models seem quite different. Are they as different as they seem? How are they related?

This chapter begins with a brief consideration of the neural interpretation of PDP models of cognition. These considerations serve mostly to lay out a certain perspective on the PDP modeling world, to make some distinctions I have found to be valuable, to introduce some terminology, and to lead into the main question of this chapter: How are distributed and local PDP models related? The chapter ends with a discussion of how, using the framework of PDP models, we might forge a mathematical relationship between the principles of mind and brain.

The following technique will be used to relate distributed to local models. We take a distributed model of some cognitive process, and mathematically formulate a *conceptual description* of that model, a description in terms of the conceptual entities themselves rather than the activity patterns that represent them. From some perspectives, this amounts to taking an account of cognition in terms of neural processing and transforming it mathematically into an account of cognition in terms of conceptual processing. The conceptual account has a direct relation to a local model of the cognitive process, so a distributed model has been mapped onto a local model.

The mathematical formulation of the conceptual description of a distributed model is straightforward, and the mathematical results reported in this chapter are all quite elementary, *once the appropriate mathematical perspective is adopted.* The major portion of this chapter is therefore devoted to an exposition of this abstract perspective on PDP modeling, and to bringing the consequent mathematical observations to bear on the cognitive issues under consideration.

The abstract viewpoint presented in this chapter treats PDP models as *dynamical systems* like those studied in mathematical physics. The mathematical concepts and techniques that will be used are those of linear algebra, the study of vector spaces; these techniques are discussed in some detail in Chapter 9. The formal parts of the discussion will be confined to footnotes and italicized passages; these may be skipped or skimmed as all results are discussed *conceptually* in the main portion of the text, which is self-contained.

NEURAL AND CONCEPTUAL INTERPRETATIONS

The interpretation of any mathematical model involves the mapping of a mathematical world into some observable part of the real world. The ambiguity in the interpretation of PDP models arises because the mathematical world of the model can be mapped into *two* observable worlds: the neural world and the world of cognitive behavior.

The neural world relevant here is discussed in Chapter 20—a world of receptor cells, neurons, synaptic contacts, membrane depolarizations, neural firings, and other features of the nervous system viewed at this level of description. The mathematical world is that described in Chapter 2—a world of processing units, weighted connections, threshold and sigmoidal functions, and activation.

Least precisely prescribed is the world of cognitive behavior. This world is mapped by experiments that probe perceptual behavior, reasoning and problem solving behavior, skilled motor behavior, linguistic behavior, memory task behavior, and the like. PDP models have been interpreted in terms of all these aspects of behavior, and to understand the common basis for these interpretations we must adopt a fairly general—and rather imprecise—way of speaking about cognition.

The connection between the formal structure of PDP models and cognitive behavior rests on theoretical knowledge constructs hypothesized to underlie this behavior. Consider perception first. Interpreting sensory input can be thought of as consideration of many *hypotheses* about possible interpretations and assigning degrees of confidence in these hypotheses. Perceptual hypotheses like "a word is being displayed the first letter of which is *A*," "the word *ABLE* is being displayed," and "the word *MOVE* is being displayed" are tightly interconnected; confidence in the first supports confidence in the second and undercuts confidence in the third. Thus assignment of confidence—*inference*—is supported by knowledge about the positive and negative evidential relations among hypotheses. This same kind of knowledge underlies other cognitive abilities; this kind of inference can support problem solving, the interpretation of speech and stories, and also motor control. The act of typing *ABLE* can be achieved by letting "the word *ABLE* is to be typed" support "the first letter to be typed is *A*" and inhibit "the first letter to be typed is *M*."

This way of thinking about cognition can be summarized by saying that behavior rests on a set of internal entities called hypotheses that are positively and negatively related in a knowledge base that is used for inference, the propagation of confidence. The hypotheses relate directly to our way of thinking about the given cognitive process; e.g., for language processing the hypotheses relate to words, syntactic categories, phonemes, meanings. To emphasize that these hypotheses are defined in terms of our concepts about the cognitive domain, I will call them *conceptual hypotheses* or the *conceptual entities*, or simply *concepts*; they are to be distinguished from the mathematical and neural entities—units and neurons—of the other two worlds.

The internal structure of the neural, mathematical, and conceptual worlds we have described are quite similar. Table 1 displays mappings that directly relate the features of the three worlds. Included are all the central defining features of the PDP models—the mathematical world—and the corresponding features of the portion of the neural and conceptual worlds that are directly idealized in PDP models.

TABLE 1

THE MAPPINGS FROM THE MATHEMATICAL WORLD TO THE NEURAL AND CONCEPTUAL WORLDS

Neural	Mathematical	Conceptual
neurons	units	hypotheses
spiking frequency	activation	degree of confidence
spread of depolarization	spread of activation	propagation of confidence: inference
synaptic contact	connection	conceptual - inferential - interrelations
excitation/inhibition	positive/negative weight	positive/negative inferential relations
approximate additivity of depolarizations	summation of inputs	approximate additivity of evidence
spiking thresholds	activation spread threshold G	independence from irrelevant information
limited dynamic range	sigmoidal function F	limited range of processing strength

In Table 1, individual units in the mathematical world are mapped on the one hand into individual neurons and on the other into individual conceptual hypotheses.[1] These two mappings will be taken to define the *local neural interpretation* and the *local conceptual interpretation* of PDP models, respectively. These are *two separate* mappings, and a particular PDP model may in fact be intended to be interpreted with only one of these mappings. Using both mappings for a single model would imply that individual concepts, being identified with individual units, would also be identified with individual neurons.

In addition to the local mappings there are also *distributed* mappings of the mathematical world into each of the neural and conceptual worlds. In a distributed *conceptual* interpretation, the confidence in a

[1] What is relevant to this chapter are the general features, not the details, of Table 1. The precision with which the mappings are described is sufficient for present purposes. For a more precise account of the relation between the mathematics of certain PDP models and the mathematics of inference, see Hinton, Sejnowski, and Ackley (1984).

conceptual hypothesis is represented by the strength of a pattern of activation over a set of mathematical units. In a distributed *neural* interpretation, the activation of a unit is represented by the strength of a pattern of neural activity of a set of neurons.

It must be emphasized that the choices of neural and conceptual interpretations are truly independent. Some neural models (e.g., Hopfield, 1982) may have no direct conceptual interpretation at all; they are intended as abstract models of information processing, with no cognitive domain implied and therefore no direct connection with a conceptual world. The reading model of McClelland and Rumelhart (1981; Rumelhart & McClelland, 1982; see Chapter 1) has an explicit local conceptual interpretation; we can choose to give it no neural interpretation, a local neural interpretation (implying individual neurons for individual words), or a distributed neural interpretation. The Hinton (1981a) and J. A. Anderson (1983) models of semantic networks have explicitly distributed conceptual interpretations; they *can* be given a local neural interpretation, so that the patterns over units used in the models are directly interpreted as patterns over neurons. They can also be given a distributed neural interpretation, in which the units in the model are represented by activity patterns over neurons so that the concepts—patterns over units—correspond to new patterns over neurons.

The nature of the patterns chosen for a distributed interpretation—either neural or conceptual—can be important (although it is not always; this is one of the results discussed later). A distributed interpretation will be called *quasi-local* if none of the patterns overlap, that is, if every pattern is defined over its own set of units. Quasi-local distributed interpretation, as the name implies, forms a bridge between local and distributed interpretation: a quasi-local neural interpretation associates several neurons with a single mathematical unit, but only a single unit with each neuron.

Since quasi-local interpretations are special cases of distributed representations, the methods applied in this chapter to the general case of distributed representations could also be applied to quasi-local representations. Certain similarities to local representations can be expected to emerge, but the general results to be discussed suggest that it is a mistake to assume that the mathematical properties of the local and quasi-local cases will be essentially the same.

The primary reason for displaying Table 1 is to emphasize that the neural and conceptual bases for interest in PDP models are *completely* independent. Even if all neural interpretations are eliminated empirically, or if no neural interpretation is given a model at all, a conceptual interpretation remains a strong independent source of cognitive relevance for a PDP model.

At the same time, the independence of the neural and conceptual mappings makes it quite striking that *both* contact exactly the same class of mathematical models. Why should this be? Is it that we have ignored crucial features of the two worlds, features which would lead to quite different mathematical abstractions? A more encouraging possibility is this: Perhaps we *have* captured the essence of neural processing in PDP models. Perhaps implicit in the processing of neural firing patterns is another mathematical description, a description in terms of the *concepts* represented in those patterns. Perhaps when we analyze the mathematics of this conceptual description, we will find that it has the mathematical structure of a PDP model—that because of special properties of PDP models, at both the neural *and* conceptual levels of description, *the mathematical structure is the same.*

This wildly optimistic scenario (depicted schematically in Figure 1) will be called *the hypothesis of the isomorphism of the conceptual and neural levels*—the *isomorphism hypothesis* for short. We will find that it is in fact exactly true of the simplest—one might say the purest—PDP models, those without the nonlinear threshold and sigmoidal functions. For models with these nonlinearities, we shall see how the isomorphism

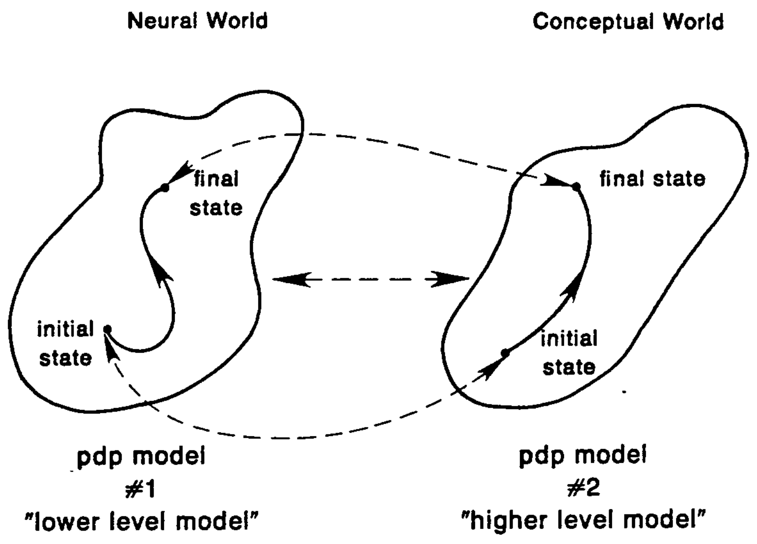

FIGURE 1. The dashed line indicates a mapping from a PDP model representing neural states to a PDP model representing conceptual states. The mapping is an isomorphism if, when the two models start in corresponding states and run for the same length of time, they always end up in corresponding states.

hypothesis fails, and explore a phenomenon within these models that distinguishes between the neural and conceptual levels.[2]

It is an open empirical question whether the conceptual entities with which we understand most cognitive processes are represented by the firing of a single neuron, by the firing of a group of neurons dedicated to that concept, by a pattern of firing over a group of neurons involved in representing many other concepts, by neural features other than firing, or by no neural features at all. The purpose of this chapter is to show how to use PDP models as a mathematical framework in which to *compare the implications* of assumptions like those of local and distributed models.

The plan of attack in this chapter is to compare two related mathematical models, each of which can be given either neural or conceptual interpretations. They can be thought of as describing a single neural net with both a local and a distributed model, or as implementing inference over a single set of conceptual hypotheses using both a local and a distributed model. The comparison of these two models will thus tell us two things. It will show how a description of a neural net in terms of its patterns compares with a description in terms of its individual neurons. It will also provide information about how behavioral predictions change when a local model of a set of concepts is converted to a distributed model.

The comparison between the two mathematical models will constitute an investigation of the isomorphism of levels hypothesis. Model 2 will be a description of the *dynamics of the patterns of activation* of Model 1; the hypothesis is that the description at the higher level of patterns (Model 2) obeys the same laws as—is isomorphic to—the description at the lower level of individual units (Model 1). To permit all the relevant interpretations to apply, I shall call Model 1 simply the *lower-level model* and Model 2 the *higher-level model.*

[2] Strictly speaking, an *isomorphism* insists not only that there be a mapping between the neural and conceptual world that preserves the dynamics, as indicated by Figure 1, but also that the map establish a one-to-one correspondence between states in the two worlds. Actually, it seems reasonable to assume that the neural world is a larger space than the conceptual world: that a conceptual state lumps together many neural states, or that the set of possible neural states includes many that have no conceptual counterpart. This would render the "conceptual level" a "higher" level of description. These considerations will manifest themselves formally in the section "Pattern Coordinates," but for now the relation of isomorphism, despite its symmetry, serves to emphasize the true strength of the hypothesis under consideration.

FROM NETWORK OF PROCESSORS TO DYNAMICAL SYSTEM

This section introduces the perspective of PDP models as *dynamical systems*. Traditionally, a PDP model is viewed as a network of processors communicating across links; I will call this the *computational viewpoint*. To illustrate the difference between the computational and dynamical system perspectives, it is useful to consider a prototypical dynamical system: a collection of billiard balls bouncing around on a table.

A common exercise in object-oriented computer programming is to fill a video screen with billiard balls bouncing off each other. Such a program creates a conceptual processor for each billiard ball. Each "ball" processor contains variables for its "position" and "velocity," which it updates once for each tick of a conceptual clock. These processors must exchange messages about the current values of their variables to inform each other when "bounces" are necessary.

Billiard balls *can* be seen from a computational viewpoint as processors changing their local variables through communication with other processors. Physics, however, treats the position and velocity values simply as real variables that are mutually constrained *mathematically* through the appropriate "laws of physics." This is characteristic of the dynamical system viewpoint.

To view a PDP model as a dynamical system, we separate the data and process features of the units. The activation values of the units are seen merely as variables that assume various values at various times, like the positions and velocities of billiard balls. The changes in these variables over time are not conceptualized as the result of prescribed computational processes localized in the units. In fact the processes by which such changes occur are unanalyzed; instead, mathematical equations that constrain these changes are analyzed.

The equations that determine activation value changes are the analogs of the laws of physics that apply to billiard balls; they are the "laws of parallel distributed processing" that have been described in Chapter 2. A version of these equations can be written as follows. Let $u_\nu(t)$ denote the activation of unit ν at time t. Then its new value one unit of time later is given by

$$\mathrm{u}_\nu(t+1) = \mathrm{F}[\sum_{\mu} \mathrm{W}_{\nu\mu} \mathrm{G}(\mathrm{u}_\mu(t))]. \qquad (1\mathrm{A})$$

Here F is a particular nonlinear sigmoid function, an increasing S-shaped function that takes a real number as input (the net activation flowing into a unit) and gives as output a number in the range $[-m, M]$

(the new activation of the unit). G is a nonlinear threshold function: $G(x) = x$ unless x is less than a threshold, in which case $G(x) = 0$. $W_{\nu\mu}$ is the strength of the connection *from* unit μ *to* unit ν.

The "knowledge" contained in a PDP model lies in the connection strengths $\{W_{\nu\mu}\}$ or *weight matrix*, **W**. The nonlinear functions F and G encode no knowledge about the cognitive domain of the model, and serve to control the activation spread in non-domain-specific ways.

Thus the heart of a PDP model is its weight matrix; the rest of the machinery can be viewed as bells and whistles added to get the weight matrix to be "used properly" during inference. The purest PDP models, from this point of view, consist only of the weight matrix; from the preceding equation, F and G are removed:

$$u_\nu(t+1) = \sum_\mu W_{\nu\mu} u_\mu(t).$$

The absence of the controlling nonlinearities make these models difficult to use for real modeling, but for our analytic purposes, they are extremely convenient. The main point is that even for nonlinear models with F and G present, it remains true that at *the heart of the model is the linear core,* **W**. For this reason, I will call the dynamical systems governed by the preceding equation *quasi-linear dynamical systems.* In this chapter, the analysis of PDP models becomes the analysis of quasi-linear dynamical systems. These systems will be viewed as elaborations of linear dynamical systems.

KINEMATICS AND DYNAMICS

Investigation of the hypothesis of the isomorphism of levels is a purely mathematical enterprise that bears on the questions of interpreting PDP models. This section provides an introduction to the mathematical structure of dynamical systems.

There are two essential components of any dynamical system. First, there is the *state space* S, the set of all possible states of the system. In our case, each state $\mathbf{s}$ in S is a pattern of activation, i.e., a vector of activation values for all of the units. The second component of a dynamical system is a set of *trajectories* $\mathbf{s}_t$, the paths through S that obey the *evolution equations* of the system. These trajectories can start at any point $\mathbf{s}_0$ in S. For activation models, $\mathbf{s}_0$ is the initial activation pattern determined by the input given to the model (or its history prior to our observation of it), and the corresponding trajectory $\mathbf{s}_t$ is the ensuing sequence of activation values for all later times, viewed as a path in the state space S. (This "path" is a discrete set of points because the values of t are; this fact is not significant in our considerations.) The evolution equations for our quasi-linear dynamical systems are Equations 1A of the previous section.

Corresponding to these two components of dynamical systems are two sets of questions. What is the state space S? Is it finite or infinite? Bounded or unbounded? Continuous or discrete? How can the points in S be uniquely labeled? What structures relate points to one another? These properties define the *geometry* of state space, what I will call the *kinematics* of the dynamical system. In kinematics, the evolution equations and trajectories are ignored; time plays no role. Only the properties of the points in state space themselves are considered.

The questions in the second set pertain to the trajectories. Are they repetitive (periodic)? Do they tend to approach certain special states (limit points)? Can we define quantities over S that differ from trajectory to trajectory but are constant along a trajectory (conserved quantities)? These are the questions about the system's *dynamics*, and their answers depend strongly on the details of the evolution equations.

It may seem that the questions of dynamics are the real ones of interest. However, it is useful to consider kinematics separately from dynamics for two reasons. First, the link between kinematics and dynamics is strong: The kinds of evolutionary equations that can sensibly be assumed to operate in a dynamical system are limited by the geometry of its state space. For example, geometrical structures expressing the *symmetries* of spacetime or elementary-particle state spaces restrict *severely* the possible evolutionary equations: This is the basis of the theory of relativity and gauge field theories of elementary particles. In our case, imposing boundedness on the state space will eventually lead to the breakdown of the isomorphism of levels. The second reason to emphasize kinematics in its own right is that the questions of *interpreting* the dynamical system have mostly to do with interpreting the states, i.e., with kinematics alone.

In this chapter we are concerned primarily with interpretation, and the discussion will therefore center on kinematics; only those aspects of dynamics that are related to kinematics will be considered. These aspects involve mainly the general features of the evolution equations—the linearity of one component (**W**) and the nonlinearity of the remaining components (F,G). More detailed questions about the trajectories (such as those mentioned above) address the *behavior* of the system rather than its interpretation, and will not be considered in this chapter.[3]

[3] J. A. Anderson, Silverstein, Ritz, and Jones (1977) show the power of concepts from linear algebra for studying the dynamics of PDP models. Some of their results are closely related to the general observations about nonlinear models I will make in the last portion of this chapter.

KINEMATICS

The first question to ask about a state space is: How can the points—states—be labeled? That is, we must specify a *coordinate system* for S, an assignment to each activation state $\mathbf{s}$ of a unique set of numbers.

Each such $\mathbf{s}$ denotes a pattern of activation over the units. Let us denote the activation value of the νth unit in state $\mathbf{s}$ by $u_\nu(\mathbf{s})$; this was formerly just denoted u_ν. These functions $\{u_\nu\}$ form the *unit coordinates* for S. Each function u_ν takes values in the set of allowed activation values for unit ν. In the standard PDP model, this is the interval $[-m, M]$. One could also consider binary units, in which case the functions u_ν would take values in the set $\{0,1\}$ (or $\{-1,1\}$). In any event, if all units are of the same type, or, more specifically, if all have the same set of allowed values, then all unit coordinates take values in a single set.

It is sometimes helpful to draw a very simplified example of a state space S. Using unit coordinates, we can plot the points of S with respect to some Cartesian axes. We need one such axis for each u_ν, i.e., each unit. Since three axes are all we can easily represent, we imagine a very simple network of only three units. The state space for such a network is shown in Figure 2. In Figure 2A, the case of activation values in $[-m, M]$ is depicted. In this case, S is a solid cube (with side of length $m+M$). Figure 2B depicts the case of binary activation values $\{0,1\}$; in this case, S is the eight vertices (corner points) of a cube. Except where specified otherwise, in the remainder of this chapter the standard case of continuous activation values will be assumed.

Thus if the network contains N units, S is an N-dimensional space that can be thought of as a solid hypercube. Any other point of N-space outside this hypercube is excluded from S because it corresponds to activation for at least one unit outside the allowed range. Thus, states with "too much" activation have been excluded from S.

"Too much" has so far been defined according to each unit individually; *it is interesting to consider whether states should be excluded if they correspond to "too much" activation among the units* collectively. *This would amount to excluding from S even some of the points in the hypercube.*

Here are two ways one might eliminate states with too much activation in total. The first way is to require that of the complete set of N units, only $N_{\max}$ can be active (i.e., have nonzero activation) at any one time. As Figure 3 shows, this removes much of the hypercube and leaves S with a rather bizarre shape. This S is topologically quite different from the original hypercube.

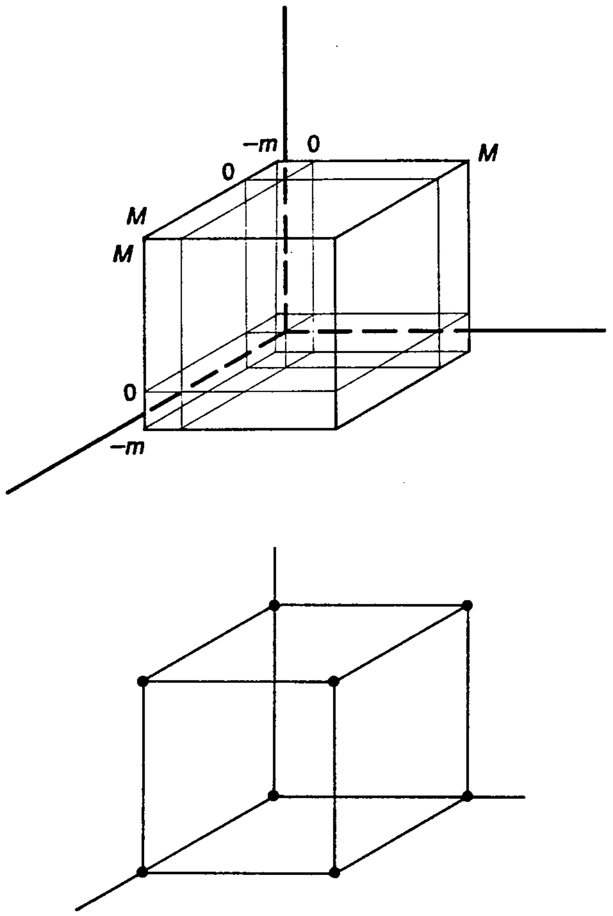

FIGURE 2. *A*: The solid cube bounded by $-m$ and M forms the standard state space for PDP models containing continuous units. *B*: The eight corners of the cube bounded by 0 and 1 forms a modified state space, corresponding to PDP models containing binary units.

A less unusual approach is to define "too much activation in total" by the condition

$$\sum_{\nu} |u_\nu| > a_{\max}.$$

This results in an S *that is depicted in Figure 4. Unlike the bizarre* S *of Figure 3, this new* S *is* not *topologically distinct from the original hypercube.*

Imposing this kind of limitation on total activation would turn out to have much less of an effect on the dynamics of the model than would the limitation on the number of active units.

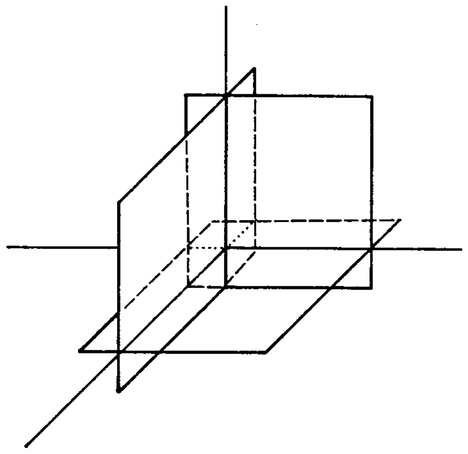

FIGURE 3. If only two of three units are allowed to have nonzero activation, the state space is formed from three intersecting planes.

Redefining the "total activation" to be the Euclidean distance of the plotted point from the origin would not change the conclusion that S is not topologically distinct from the hypercube. In fact, limiting the Euclidean distance or the sum of activations, or using the original hypercube, are all special cases of defining S to be a "ball" with respect to some metric in N-space. It is a fact that all such balls are topologically equivalent (e.g., Loomis & Sternberg, 1968, p. 132).

In the remainder of this chapter, S will denote the standard hypercube as depicted in Figure 2A, the state space of the general nonlinear activation model. The state space of the simplified, linear, activation model will be denoted S_L. This space, as we shall see in the next section, is simply all of N-dimensional Euclidean space, where N is again the number of units in the network. (For example, there is no need to draw S_L because, for $N = 2$, it is the entire plane of the paper!) S is clearly a subset of S_L; in S, the unit coordinates of any state fall within the restricted range $[-m, M]$, while in S_L the unit coordinates can be any real numbers.

The unit coordinates provide a convenient description of S for many purposes. However, it is important to realize that points of S can also be described with an infinitude of other coordinate systems. In a distributed interpretation, new coordinates that give a simple description of these patterns will turn out to be better than unit coordinates for interpreting states of the system. We shall construct these *pattern*

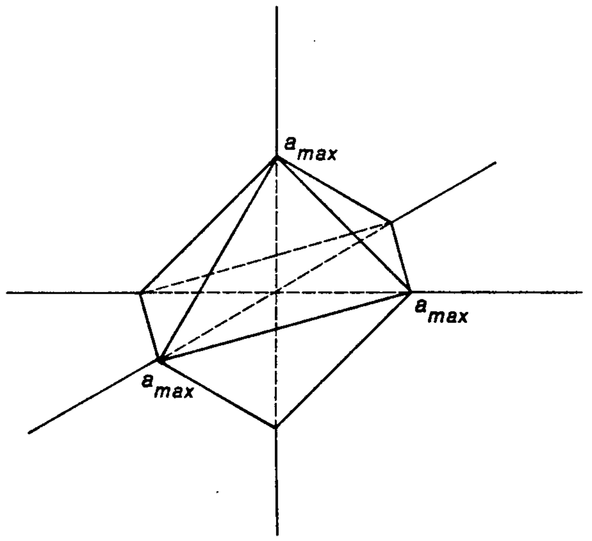

FIGURE 4. This solid octahedron is the state space obtained by restricting the sum of the magnitudes of the units' activations to be less than $a_{\max}$.

coordinates shortly, but this construction uses a property of S we must now discuss: its vector space structure.

VECTOR SPACE STRUCTURE

A central feature of parallel distributed processing is its exploitation of *superposition* of knowledge during computation. Each unit that becomes active exerts its influence in parallel with the others, superimposing its effects on those of the rest with a weight determined by its level of activation. As we shall see, in linear models, this is a mathematically precise and complete account of the processing. As emphasized in the section "From Network of Processors to Dynamical System," even nonlinear PDP models are quasi-linear systems, and knowledge is used in the same fashion as in linear models. Thus superposition plays a crucial role in all PDP models.

Superposition is naturally represented mathematically by *addition*. In the simplest case, addition of individual numbers represents superposition of unidimensional quantities. Adding multidimensional quantities (like states of activation models) is mathematically represented by *vector addition*. The concepts of vector and vector addition are best viewed together, as data and process: "Vector" formalizes the notion of "multidimensional quantity" specifically for the purpose of "vector addition."[4]

Actually, the notion of superposition corresponds to somewhat more than the operation of addition: Superposition entails the capability to form *weighted* sums. This is important for parallel distributed processing, where the complete state of the system typically corresponds to a *blend* of concepts, each *partially* activated. Such a state is mathematically constructed by *summing* the states corresponding to the partially activated concepts, each one *weighted* by its particular degree of activation.

Using unit coordinates, the operation of weighted summation is simply described. The activation of a unit in a state that is the weighted sum of two other states is simply the weighted sum of the activations of that unit in those two states. In other words, the unit coordinates of the weighted sum of states are the weighted sum of the unit coordinates of the states. Given two states $\mathbf{s}_1$ and $\mathbf{s}_2$, and two weights w_1 and w_2, the weighted sum $\mathbf{s}$ is written

$$\mathbf{s} = w_1\mathbf{s}_1 + w_2\mathbf{s}_2 .$$

What I have already said about the unit coordinates is then written

$$u_\nu(\mathbf{s}) = w_1 u_\nu(\mathbf{s}_1) + w_2 u_\nu(\mathbf{s}_2).$$

Using unit coordinates, the evolution equation of quasi-linear systems, Equation 1A, can be written

$$u_\nu(\mathbf{s}_{t+1}) = \mathrm{F}[\sum_\mu \mathrm{W}_{\nu\mu}\mathrm{G}(u_\mu(\mathbf{s}_t))]. \tag{1B}$$

The reason all the unit coordinates u_ν of the state $\mathbf{s}_{t+1}$ are guaranteed to lie in the allowed range $[-m, M]$ is that the function F takes all its values in that range. This nonlinear function is what ensures that the trajectories do not leave the bounded cube S of states. If F were absent, then the coordinate u_ν would just be the weighted sum (with weights $\mathrm{W}_{\nu\mu}$ for all values of μ) of the quantities $\mathrm{G}(u_\mu(t))$; this need

[4] It is common to use the term "vector" for any multidimensional quantity, that is, a quantity requiring more than one real number to characterize completely. This is, however, not faithful to the mathematical concept of vector unless the quantity is subject to superposition. The precise meaning of "superposition" is captured in the axioms for the operation of "addition" that defines it (see Chapter 9).

not lie in the allowed range.[5] So in the simplified linear theory in which F and G are absent, the evolution equation

$$u_\nu(\mathbf{s}_{t+1}) = \sum_\mu \mathrm{W}_{\nu\mu} u_\mu(\mathbf{s}_t) \tag{2A}$$

imposes no restriction on the range of values of the coordinates; trajectories may wander anywhere in N-space.

Thus we see that the kinematical restriction of state space to the bounded region S has lead to the insertion of the bounded (and therefore nonlinear) function F into the dynamical equation. If state space is extended to all of N-space, i.e. S_L, then the linear dynamical equation above is permissible.

The linear evolution equation can be written more transparently in terms of state vectors rather than unit coordinates. Define N vectors $\mathbf{w}_\mu$ by

$$u_\nu(\mathbf{w}_\mu) = \mathrm{W}_{\nu\mu}.$$

$\mathbf{w}_\mu$ *is the vector of weights on connections* from *unit μ. It is also the activation vector that would exist at time $t+1$ if at time t, unit μ had activation 1 and all other units had activation 0. Now because the evolution is linear, the state at time $t+1$ produced by a general activation pattern at time t is just the weighted sum of the patterns that would be set up by individual unit activations at the units, with the weights equal to the actual activation of the units. That is,*

$$\mathbf{s}_{t+1} = \sum_\mu u_\mu(\mathbf{s}_t)\mathbf{w}_\mu.$$

(This vector can be seen to obey the linear evolution equation given above by evaluating its νth unit coordinate, using the rule for coordinates of weighted sums, and the defining coordinates of the vectors $\mathbf{w}_\mu$.)

This equation explicitly shows the blending of knowledge that characterizes parallel distributed processing. The vector $\mathbf{w}_\mu$ is the output of unit μ; it is the "knowledge" contained in that unit, encoded as a string of numbers. The state of the system at time $t+1$, $\mathbf{s}_{t+1}$, is created by forming a weighted superposition of all the pieces of knowledge stored in all the units. The weight for unit μ in this superposition determines how much influence is exerted by the knowledge encoded by that unit. This weight, according to the previous equation, is $u_\mu(\mathbf{s}_t)$. That is just the degree to which unit μ is active at time t.

Another useful form for the linear evolution equation uses matrix notation:

$$\mathbf{u}(t+1) = \mathbf{W}\,\mathbf{u}(\mathbf{t}). \tag{2B}$$

[5] By imposing special restrictions on the weights, it is possible to ensure that the weighted sum lies in S, and then the nonlinearity F can be eliminated. But like F, these restrictions would also have strong implications for the dynamics.

u is the $N \times 1$ column matrix of unit coordinates u_ν, and **W** is the $N \times N$ matrix of values $W_{\nu\mu}$. This matrix is relative to the unit coordinates; shortly we shall switch to other coordinates, changing the numbers in both the square matrix **W** and the column matrix **u**.

PATTERN COORDINATES

Now we consider activity patterns used in a distributed neural or conceptual interpretation. For concreteness, consider the pattern $<+1,-1,+1,-1>$ over the first four units. To denote this pattern, we define a vector **p** the first four unit coordinates of which are $<+1,-1,+1,-1>$; the remaining unit coordinates are zero. Now consider the state **s** with unit coordinates $<.3,-.3,.3,-.3,0,0, \ldots, 0>$. This state can be viewed in two ways. The first is as the superposition of four states: $\mathbf{u}_1$ with unit coordinates $<1,0,0, \ldots, 0>$, $\mathbf{u}_2$ with unit coordinates $<0,1,0, \ldots, 0>$, etc., with weights respectively $+.3, -.3, +.3$, and $-.3$. This is the *unit view* of **s**. The second view is simpler: **s** is simply .3 times **p**:

$$\mathbf{s} = .3\mathbf{p} .$$

This is the *pattern view* of **s**.

The general situation is comparable. If there is a whole set of distributed patterns, each can be represented by a vector $\mathbf{p}_i$. Any given state **s** can be represented in two ways: as the superposition of the vectors $\mathbf{u}_\nu$, with weights given by the unit coordinates of **s**, or as a superposition of pattern vectors $\mathbf{p}_i$. If the patterns comprise a distributed conceptual interpretation, the weights in this latter superposition indicate the system's confidence in the corresponding conceptual hypotheses.

Let's consider a slightly less simplified example. Let $\mathbf{p}_1$ be the vector **p** above, and let $\mathbf{p}_2$ correspond to the activation pattern $<+1,+1,+1,+1>$ over the first four units. Then the state **s** with unit coordinates $<.9,.6,.9,.6,0,0, \ldots, 0>$ can be viewed either as

$$\mathbf{s} = .9\mathbf{u}_1 + .6\mathbf{u}_2 + .9\mathbf{u}_3 + .6\mathbf{u}_4$$

or as

$$\mathbf{s} = .15\mathbf{p}_1 + .75\mathbf{p}_2 .$$

The first representation shows the activation pattern of units, while the second shows **s** to be a weighted sum of the two patterns with respective strengths .15 and .75.

It is useful to consider this example geometrically as well as algebraically. In Figure 5, **s** is drawn (only the first two unit dimensions are depicted). The projections of this vector onto the unit axes (defined by the vectors $\mathbf{u}_1$ and $\mathbf{u}_2$) are .9 and .6, while the projections onto the vectors $\mathbf{p}_1$ and $\mathbf{p}_2$ are .15 and .75. These conceptual vectors define the axes of the *pattern coordinate system* for state space.

In a PDP model with a distributed interpretation, the interpretation of the state of a system requires the use of the pattern coordinate system. The mathematics of linear algebra (discussed in Chapter 9) tells how to convert state descriptions from unit coordinates to pattern coordinates, and vice versa. All that is required is the specification of the patterns.

Before considering the conversion between unit and pattern coordinates, one observation needs to be made. Consider a distributed conceptual interpretation. (Exactly the same considerations apply to a distributed neural representation.) If confidence in a group of conceptual hypotheses are to be separable, then the pattern

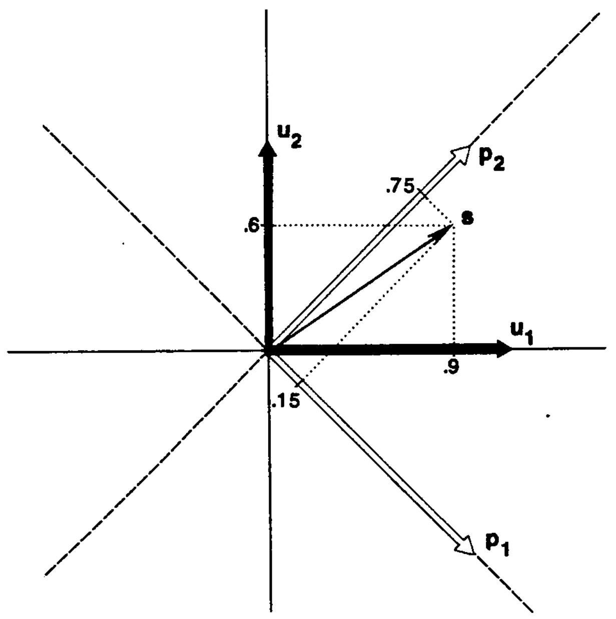

FIGURE 5. The unit basis vectors $\mathbf{u}_1$ and $\mathbf{u}_2$ define the axes for the unit coordinate system. The state s has unit coordinates $<.9,.6>$. The pattern vectors $\mathbf{p}_1$ and $\mathbf{p}_2$ define the axes for the pattern coordinate system. The state s has pattern coordinates $<.15,.75>$.

vectors representing them must be linearly independent. Suppose, for example, that $\mathbf{p}_3$ *is not independent of* $\mathbf{p}_1$ *and* $\mathbf{p}_2$*; say,*

$$\mathbf{p}_3 = .4\mathbf{p}_1 + .3\mathbf{p}_2$$

Then the state $\mathbf{s}$ *representing confidence .2 in Hypothesis 1, .15 in Hypothesis 2, and 0 confidence in Hypothesis 3,*

$$\mathbf{s} = .2\mathbf{p}_1 + .15\mathbf{p}_2 + 0\mathbf{p}_3 ,$$

will be identical to the state representing confidence .5 in Hypothesis 3, and 0 confidence in Hypotheses 1 and 2:

$$\mathbf{s} = 0\mathbf{p}_1 + 0\mathbf{p}_2 + .5\mathbf{p}_3 .$$

Thus Hypothesis 3 is not separable from Hypotheses 1 and 2; in fact any of the three hypotheses can be written as a superposition of the other two, so it is better to say that the Hypotheses 1, 2, and 3 are not independently *represented by the activation patterns described by the vectors* $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$.

Thus we must assume that the distributed representation involves a set of separable hypotheses that are represented by a linearly independent *set of vectors* $\{\mathbf{p}_i\}$. *Therefore if there are* N *units, so that state space is* N*-dimensional, there may be at most* N *conceptual vectors. If there are exactly* N *such vectors, then* $\{\mathbf{p}_i\}$ *forms a* basis *for* S_L*: Every state in* S_L *is uniquely expressible as a superposition of the patterns, and therefore interpretable in terms of the conceptual hypotheses. If there are fewer than* N *conceptual vectors, then there will be states of the model that are not conceptually interpretable, since no superposition of the vectors* $\{\mathbf{p}_i\}$ *will equal the state. This may be no problem if the* dynamics *of the model (i.e., W) tends to keep trajectories away from such states.*

When a distributed interpretation involves fewer than N *patterns, only the vectors in a* subspace *of the state space* S_L *are interpretable, and the pattern coordinates allow description only of this subspace. This reduction in expressivity is to be expected in a passage from a lower- to higher-level description. In any event, when there are fewer than* N *patterns, extra pattern vectors can be freely created to expand* $\{\mathbf{p}_i\}$ *to a complete basis. To simplify the analysis, we shall assume this to be done, noting that states involving these extra vectors are not completely interpretable.*

The unit and pattern coordinates of a state $\mathbf{s}$ are the components of the vector $\mathbf{s}$ with respect to the unit basis $\{\mathbf{u}_\nu\}$ and the pattern basis $\{\mathbf{p}_i\}$, respectively. To translate between these bases, we need the change-of-basis matrix $\mathbf{P}$ defined by the entries

$$\mathrm{P}_{\nu i} = u_\nu(\mathbf{p}_i).$$

The ith column of this matrix is simply the pattern of activation over the units defining the ith pattern. Then the unit coordinates of a state $\mathbf{s}$, the components of $\mathbf{s}$ with respect to the unit basis $\{\mathbf{u}_\nu\}$, can be computed from the conceptual components of $\mathbf{s}$, the components of $\mathbf{s}$ with respect to the pattern basis $\{\mathbf{p}_i\}$, by

$$\mathbf{u} = \mathbf{P}\mathbf{p}. \tag{3A}$$

In this matrix equation, $\mathbf{u}$ and $\mathbf{p}$ are the $N \times 1$ column matrices of coordinates of $\mathbf{s}$ in the unit and pattern systems, respectively.

To compute the pattern coordinates from the unit ones, we need the *inverse* matrix of $\mathbf{P}$:

$$\mathbf{p} = \mathbf{P}^{-1}\mathbf{u}. \tag{3B}$$

The existence of an inverse of $\mathbf{P}$ *is guaranteed by the linear independence of the* $\{\mathbf{p}_i\}$.

Let's consider a simple case with two units supporting the two patterns <1,2> and <3,1>. Here the pattern matrix is

$$\mathbf{P} = \begin{bmatrix} 1 & 3 \\ 2 & 1 \end{bmatrix}.$$

Thus the state $\mathbf{s}_1$ *representing .5 of pattern one and .3 of pattern two has pattern coordinates <.5,.3> and unit coordinates*

$$\mathbf{u}_1 = \begin{bmatrix} 1 & 3 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} .5 \\ .3 \end{bmatrix} = \begin{bmatrix} 1.4 \\ 1.3 \end{bmatrix}.$$

The two coordinate systems and $\mathbf{s}_1$ *are shown in Figure 6.*

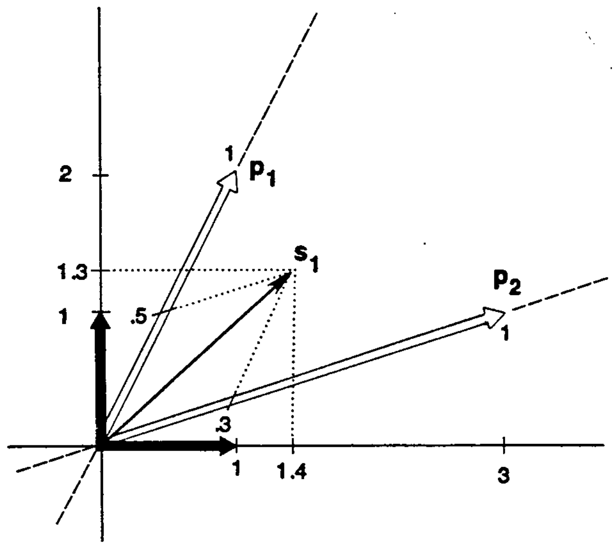

FIGURE 6. The state $\mathbf{s}_1$ in the pattern and unit coordinates.

To go from unit coordinates to pattern coordinates, we use the inverse of the pattern matrix:

$$\mathbf{P}^{-1} = \begin{bmatrix} -.2 & .6 \\ .4 & -.2 \end{bmatrix}.$$

Thus the state $\mathbf{s}_2$ *with unit coordinates* $<4,3>$ *has pattern coordinates*

$$\mathbf{p}_2 = \begin{bmatrix} -.2 & .6 \\ .4 & -.2 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

(It is easily verified that $<4,3>$ *is the sum of* $<1,2>$ *and* $<3,1>$*!) The state* $\mathbf{s}_2$ *is shown in Figure 7.*

The evolution Equation 2B for the linear model can now be transformed, by multiplication by $\mathbf{P}^{-1}$, from unit to pattern coordinates:

$$\mathbf{p}(t+1) = \mathbf{I}\,\mathbf{p}(t). \tag{4A}$$

Here the matrix $\mathbf{W}$ of interunit connection weights has become the matrix

$$\mathbf{I} = \mathbf{P}^{-1}\mathbf{W}\,\mathbf{P}. \tag{5}$$

The meaning of this matrix is clear upon writing the evolution equation out in component form:

$$p_i(t+1) = \sum_j I_{ij}\,p_j(t). \tag{4B}$$

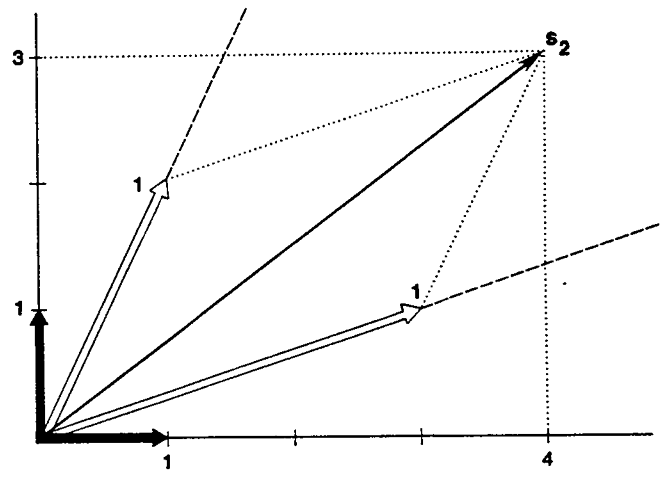

FIGURE 7. The state $\mathbf{s}_2$ in the unit and pattern coordinates.

Thus, I_{ij} is the interconnection strength from *pattern j* to *pattern i*. In a distributed conceptual interpretation, this number expresses the sign and strength of the evidential relation from the jth conceptual hypothesis to the ith conceptual hypothesis. Thus $\mathbf{I}$ governs the propagation of *inferences.*

The important result is that *the evolution equation for the linear model has exactly the same form in the pattern coordinate system as it had in the unit coordinates.*

THE ISOMORPHISM HYPOTHESIS HOLDS IN LINEAR SYSTEMS

The analysis at the end of the preceding section shows that for a *linear model* the evolution equation has exactly the same form in pattern coordinates as in unit coordinates. In other words, *there is an exact isomorphism between the lower and higher levels of description in linear models.*

This isomorphism has been viewed so far as a mapping between two descriptions of a given model. It can also be viewed as a mapping between the behavior of two different PDP models. One is the original model with which we began, a model supporting a distributed interpretation, having N units with interconnection weight matrix $\mathbf{W}$. Let's call this the *lower-level model*, M_l. The *higher-level model* M_h has a unit for every *pattern* of the distributed interpretation and has interconnection matrix $\mathbf{I}$. The law governing its processing (Equation 4) is exactly the same as that of the lower-level model (Equation 2).

The isomorphism maps states of the lower-level model into states of the higher-level model. Take any state $\mathbf{s}_l$ of the lower-level model. To find the corresponding state $\mathbf{s}_h$ of the higher-level model, express $\mathbf{s}_l$ in pattern coordinates. The coordinate for the ith pattern—its strength in state $\mathbf{s}_l$—gives the activation in state $\mathbf{s}_h$ of the ith unit of M_h. (In other words, the ith *pattern* coordinate of $\mathbf{s}_l$ equals the ith *unit* coordinate of $\mathbf{s}_h$.)

So for example if the state $\mathbf{s}$ *of the lower-level model is a superposition of the first two patterns, say,*

$$\mathbf{s} = .6\mathbf{p}_1 + .3\mathbf{p}_2$$

then the corresponding state in the higher-level model would have activation .6 in Unit 1, .3 in Unit 2, and 0 elsewhere.

The mapping between the lower- and higher-level models is an isomorphism because the evolution equation for one model is mapped into the evolution equation of the other, so that the *behavior* of one model is exactly mapped onto that of the other. That is, if the two models start off in corresponding states, and are subject to corresponding inputs, then for all time they will be in corresponding states.[6]

The significance of the behavioral equivalence of the two models can be brought out in an example. Consider a linear version of the letter perception model (that is, remove the nonlinearities from the equations of the model, maintaining the original nodes and interconnections). As originally proposed, this model, M_h, has a local conceptual interpretation. Suppose we wanted to rebuild the model with a distributed conceptual interpretation, with hypotheses about words and letters represented as activation patterns over the units of a model M_l.

First, each conceptual hypothesis (i.e., unit in the original model) would be associated with some specific activation pattern over units in M_l. The inference matrix of M_h, which gives the positive and negative strengths of connections between conceptual hypotheses in the original model, would be algebraically transformed (following Equation 5), using the activation patterns defining the distributed interpretation. This new matrix defines the correct weights between units in M_l. This sets up the lower-level model.

To run the model, inputs are chosen and also an initial state. Both are originally specified in conceptual terms, and must be algebraically transformed to M_l (following Equation 3A). This defines the inputs and initial state of the lower-level model. Then the lower-level model is run for a length of time. The model's response to the input is determined by taking the final state, representing it in pattern coordinates, and reading off the activations of the corresponding conceptual hypotheses.

What the isomorphism tells us is that after all this effort, the result will be *exactly* the same as if we had simply run the higher-level model.

The higher-level model can be viewed as a conceptual description of the lower-level model in which details of the "implementation patterns" have been ignored. The behavioral isomorphism implies that these details have no effect at all on the behavior of the model. However, the two models do implement the same processing differently, and they will differ in how they respond to modifications of the processing mechanism itself. Thus the behavioral effect of destroying a unit will

[6] If the lower-level model has external inputs to the units, exactly the same transformation that maps *states* of M_l to states of M_h also maps *inputs* of M_l to inputs of M_h. This can be verified by taking the matrix form of the new linear evolution equation,

$$\mathbf{u}(t+1) = \mathbf{W}\,\mathbf{u}(t) + \mathbf{i},$$

where $\mathbf{i}$ is the $N \times 1$ column matrix of external inputs, and performing the same operations as in the preceding section to transform it from the unit to pattern basis.

be different for the two models, and the demands on connection adaptation, i.e., learning will differ. Understanding these phenomena in terms of the lower- and higher-level models provides an opportunity to exploit more of the power of the techniques of linear algebra. They will be discussed in the following two sections, which can be skipped without diminishing the comprehensibility of the subsequent discussion.

LOCALIZED DAMAGE AND THE ISOMORPHISM OF LEVELS

Suppose we remove one unit from the lower-level model. What will be the corresponding modification in the isomorphic higher-level model? That is, what will be the effect on the model's ability to process the patterns that are meaningful in the distributed interpretation?

Removing unit ν can be viewed as insisting that it have activation zero and no incoming connections. This amounts to allowing only states s that have

$$u_\nu(\mathrm{s}) = 0.$$

This change in the kinematics of the system brings with it a change in the dynamics. We need to follow this change through, transforming it to the higher-level model.

The evolution equation must be changed so that only allowed states are reached; this amounts to saying that the activation of unit ν will never change from zero. This can be done as follows. The column matrix $\mathbf{W}\,\mathbf{u}(t)$ *has for its* ν*th element the inputs coming in to unit* ν*; rather than this for the* ν*th element in* $\mathbf{u}(t+1)$ *we want zero. So we apply a "damage matrix" that "projects out" the component along* $\mathbf{u}_\nu$*:*

$$\mathbf{D}_{\mathbf{u}_\nu} = \mathbf{1} - \mathbf{u}_\nu\,\mathbf{u}_\nu^{\mathrm{T}}.$$

(Here $\mathbf{1}$ *is the unit or identity matrix, and* $\mathbf{T}$ *is the matrix transpose operation.) The new evolution equation becomes*

$$\mathbf{u}(t+1) = \mathbf{D}_{\mathbf{u}_\nu}\,\mathbf{W}\,\mathbf{u}(t).$$

Introducing $\mathbf{D}$ *is equivalent to the more obvious step of replacing the* ν*th row of* $\mathbf{W}$ *by zeroes, or of simply removing it altogether, along with the* ν*th element of* $\mathbf{u}$*. However, it is difficult to map these surgeries onto the higher-level model to see what they correspond to. By instead performing the "damage" by introducing a new linear operation (multiplication by* $\mathbf{D}$*), we can again use simple linear algebra and transparently transform the "damage" to the pattern basis.*

We can view $\mathbf{D}$ *as a projection onto the states* orthogonal *to* $\mathbf{u}_\nu$ *if we introduce the inner product in which the unit basis vectors are orthonormal. Then the inner product of two states* $\mathbf{s}$ *and* $\mathbf{s}'$, $(\mathbf{s},\mathbf{s}')$, *can be easily computed using the column matrices* $\mathbf{u}$ *and* $\mathbf{u}'$ *of the unit cooordinates of these states:*

$$(\mathbf{s},\mathbf{s}') = \mathbf{u}^T \mathbf{u}'.$$

While the unit basis is orthonormal, the pattern basis may not be. The relevant matrix is $\mathbf{M}$ *with elements*

$$\mathrm{M}_{ij} = (\mathbf{p}_i, \mathbf{p}_j) = \sum_\nu u_\nu(\mathbf{p}_i)\, u_\nu(\mathbf{p}_j) = \sum_\nu \mathrm{P}_{\nu i}\mathrm{P}_{\nu j},$$

i.e.,

$$\mathbf{M} = \mathbf{P}^T \mathbf{P}.$$

If $\mathbf{M}$ *is the unit matrix* $\mathbf{1}$, *then the pattern basis is orthonormal, also, and the inner product of two states* $\mathbf{s}$ *and* $\mathbf{s}'$ *can be computed using the column matrices* $\mathbf{p}$ *and* $\mathbf{p}'$ *of their pattern coordinates using the same simple formula as in the unit coordinates:*

$$(\mathbf{s},\mathbf{s}') = \mathbf{p}^T \mathbf{p}' \qquad \text{[orthonormal patterns only]}.$$

Otherwise one must use the general formula of which this is a special case:

$$(\mathbf{s},\mathbf{s}') = \mathbf{p}^T \mathbf{M}\, \mathbf{p}'.$$

(Since $\mathbf{u} = \mathbf{P}\mathbf{p}$, *this is just* $\mathbf{u}^T \mathbf{u}'$.*)*

Now we apply the standard procedure for changing the dynamical equation to the pattern basis. This gives

$$\mathbf{p}(t+1) = \mathbf{D}_\mathbf{d}\, \mathbf{I}\, \mathbf{p}(t).$$

Here, the "damage vector" $\mathbf{d}$ *is the column matrix of pattern coordinates of* $\mathbf{u}_\nu$*:*

$$\mathbf{d} = \mathbf{P}^{-1}\mathbf{u}_\nu$$

and $\mathbf{D}_\mathbf{d}$ *is again the matrix that orthogonally projects out* $\mathbf{d}$*:*

$$\mathbf{D}_\mathbf{d} = \mathbf{1} - \mathbf{d}\,\mathbf{d}^T \mathbf{M}.$$

(If the pattern basis is not orthonormal, the inner product matrix $\mathbf{M}$ *must appear in the orthogonal projection matrix* $\mathbf{D}$*; if the pattern basis is orthogonal, then* $\mathbf{M} = \mathbf{1}$ *so it has no effect.)*

So the corresponding damage in the higher-level model is removal of the pattern *represented by* $\mathbf{d}$*: All allowed states will be orthogonal to this pattern. Introducing* $\mathbf{D}$ *here is equivalent to making a change in the inference matrix* $\mathbf{I}$ *that is more complicated than just converting a row to zeroes. All connections coming into the patterns that employ the deleted unit will be altered in a rather complicated way;* $\mathbf{I}$ *is replaced by* $\mathbf{D}_\mathbf{d}\,\mathbf{I}$.

Thus corresponding to the damage produced in the lower-level model by removing unit ν is the removal of all states in the higher-level model that are not orthogonal to a particular state of the higher-level units. This state corresponds under the isomorphism to $\mathbf{u}_\nu$; it represents exactly the superposition of lower-level patterns in which the activations on all lower-level units exactly cancel out to 0, except for the removed unit, which ends up with an activation of 1.

Let's return to the two-unit example from the section "Pattern Coordinates." Suppose the first unit is removed. This eliminates the horizontal axis from the state space; that is, now only states that have zero x-coordinate are allowed. This is shown in Figure 8. How does this look in pattern coordinates? The damage vector $\mathbf{d}$ *has unit coordinates* $<1,0>$. *Its conceptual coordinates are therefore*

$$\mathbf{p} = \begin{bmatrix} -.2 & .6 \\ .4 & -.2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -.2 \\ .4 \end{bmatrix}.$$

(The weighted sum of the two patterns, with weights $<-.2,.4>$, *is seen to be 0 on the second unit, 1 on the first.) Thus in the pattern coordinates, the damage has been to remove the state* $<-.2,.4>$, *leaving only states orthogonal to it. This of course is just a different description of exactly the same change in the state space: see Figure 9.*

Thus removing a lower-level unit corresponds to removing a pattern of activation over the higher-level units; under a distributed conceptual interpretation, this amounts to removing "pieces of concepts"—the pieces relying on that unit.

The same analysis can be run in reverse, to show that "removing a higher-level unit" (e.g., a concept) amounts to performing a rather complicated change in the weight matrix that has the effect of eliminating from the state space those states not orthogonal to the pattern

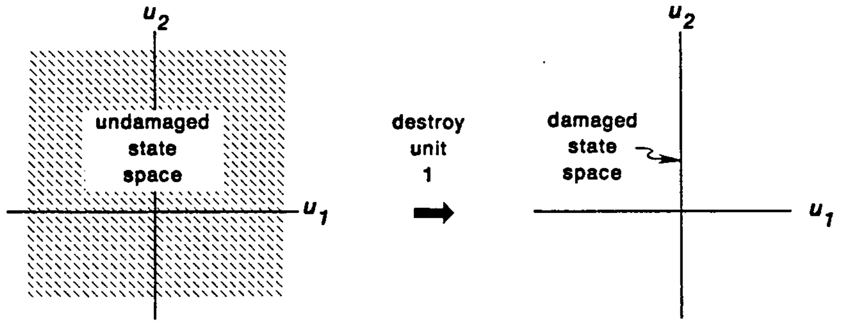

FIGURE 8. When Unit 1 is destroyed, the two-dimensional state space shrinks to the one-dimensional space with zero activation for the destroyed unit.

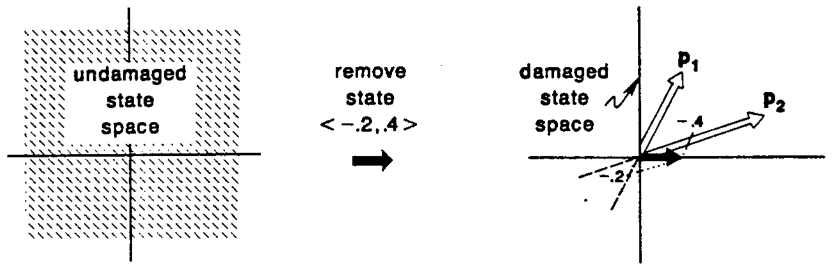

FIGURE 9. In pattern coordinates, the destruction of Unit 1 is described as the removal of the state <−.2,.4>, leaving only the states orthogonal to it.

corresponding to the deleted higher-level unit. In short, localized damage in one model is distributed damage in the other. Geometrically, the picture is similar in the two cases; localized damage removes the states orthogonal to a coordinate axis, while distributed damage removes the states orthogonal to a vector that is not a coordinate axis. Since the higher-level coordinates simply employ different axes than the lower-level coordinates, the picture makes good sense.

LEARNING OF CONNECTIONS AND THE ISOMORPHISM OF LEVELS

One major conceptual distinction between local and distributed interpretations is that in the former case the individual elements of the interconnection matrix can be readily interpreted while in the latter case they cannot. In a version of the reading model with a local conceptual interpretation, the positive connection from the unit representing "the first letter is *A*" and that representing "the word is *ABLE*" has a clear intuitive meaning. Thus the connection matrix can be set up intuitively, up to a few parameters that can be adjusted but which have clear individual interpretations. By contrast, the interconnection matrix of the modified reading model defined in the section "The Isomorphism Hypothesis Holds in Linear Systems," with a distributed conceptual interpretation, had to be obtained by algebraically transforming the original interconnection matrix: This matrix cannot be obtained by intuition, and its individual elements (the connection strengths between individual units in the distributed model) have no conceptual interpretation.

This way of generating the interconnection matrix for the distributed model seems to give a primal status to the local model. There is, however, another way to produce the interconnections in the distributed model: through a learning procedure. The two procedures I will discuss are the original Hebb learning procedure and its error-correcting modification, the delta rule.

I will only briefly and imprecisely discuss these two learning procedures; a fuller discussion is presented in Chapter 2, Chapter 8, and Chapter 11. In the Hebb procedure, the interconnection strength between two units is increased whenever those units are simultaneously active. This can be implemented by adding to the strength the product of the units' activation. In the delta rule, this is modified so that what is added to the strength is the product of the input unit's activation and the *difference* between the output unit's activation and the value it should have according to some outside "teacher."

The relevant results about these procedures, for present purposes, are these: The delta rule will eventually produce the interconnection matrix that minimizes the error, measured relative to the teacher. The Hebb procedure will give essentially the same result *in the special case* that the activation patterns the model must respond to are mutually *orthogonal*. In this case, the error-correction feature of the delta rule is superfluous.

There is an intuitive explanation of this result. Suppose a new input activation pattern must be associated with some output pattern. In general, that input pattern will, by virtue of previously learned associations, produce some output pattern. The delta rule adds into the interconnections just what is needed to *modify* that output pattern, to make it the correct one. The Hebb procedure adds connections that will themselves produce the output pattern *completely* from the input pattern, ignoring connections that have already been stored. If the new input pattern is orthogonal to all the already learned patterns, then a *zero* output pattern will be produced by the previously stored associations.[7] That is why, with orthogonal inputs, the simple Hebb procedure works.

The point is that in general, the delta rule will produce the correct interconnection matrix for a distributed model, as it will for a local model. This represents a degree of parity in the status of the two models. However, this parity has its limits. The Hebb rule will in general *not* produce the correct matrix for a distributed model, unless the patterns have been carefully chosen to be orthogonal. (A simple example of such orthogonality is a strictly local model in which each input is uniquely represented by a single unit.)

Thus the lower- and higher-level models may differ in whether Hebb learning works; this is because what is local to the connection between

[7] Orthogonality means that the inputs to each output unit cancel each other out completely.

two units in one model will in general not be local in the other, and some special arrangement—orthogonality of patterns—is needed to ensure that nonlocalities do not cause any problems.

This issue of the orthogonality of the patterns relates back to kinematics. To analyze what the individual components of the interconnection matrix need to be, it is necessary to add to the vector space of states the additional geometric structure of an inner product; *it is this that tells whether states are orthogonal. The inner product is defined so that the unit basis is orthonormal. When the pattern basis is orthonormal as well, the transformation from unit to pattern coordinates is a* rotation, *and it preserves the constraints on the interconnection matrix elements. In particular, the adequacy of the Hebb procedure is invariant under the transformation from the lower- to higher-level model. In the general case, when the pattern basis is not orthonormal, this invariance is broken.*

NONLINEARITY AND RESTRICTED STATE SPACE

Having established the validity of the isomorphism of levels hypothesis for linear PDP models, it is time to consider quasi-linear systems with nonlinearities. To understand the effects of these nonlinearities, it is helpful to go back to kinematics.

The state space S_L of linear PDP models is all of N-space, where N is the number of units in the model. By contrast, the standard state space S of general PDP models is the solid hypercube in N-space described in the section "Kinematics." This represents states in which each unit has activation within the limited range $[-m, M]$. Such a restriction is motivated within the neural interpretation by the limited range of activity of individual neurons; it is motivated within the conceptual interpretation by the desirability of limiting the possible influence of a single conceptual hypothesis. (PDP models are feedback systems that tend to be difficult to control unless activation values have a ceiling and floor. Nonlinearities also allow multilayer PDP networks to possess greater computational power than single-layer networks: see Chapter 2.)

Whatever the motivation, the restriction of states to the cube S, instead of allowing all states in N-space S_L, means that the general linear evolution equation (with unrestricted weight matrix **W**) is unacceptable. Introducing the nonlinear sigmoidal function F of Equation 1 with range $[-m, M]$ solves this problem by brute force.

Unlike S_L, the hypercube S looks quite different in different coordinate systems. In unit coordinates, all coordinates are limited to $[-m, M]$; with a different set of axes, like those of pattern coordinates, the allowed values of a given coordinate vary, depending on the values of the other coordinates. (See Figure 10.) No longer is it possible for the description of the system to be the same in the two coordinate systems. As we shall see explicitly, the exact ismorphism of levels that holds for linear models breaks down for nonlinear models.

Another characterization of the kinematic effect of restricting S to a hypercube is that now states can be distinguished according to their position relative to edges, faces, corners, and so on. Choosing to represent a concept, for example, by a point in the corner of S will produce different behavior than choosing a point in the middle of a face. These distinctions simply cannot be made in the linear state space S_L.

A crucial effect of limiting the state space to S is that now *some* superpositions of states will be in S while others will not; it will depend on the position of the states relative to the boundary of the hypercube. Because some patterns cannot superimpose, i.e., coexist, they must instead *compete* for the opportunity to exist. Other patterns can coexist, so the choice of patterns in an interpretation will matter. In particular,

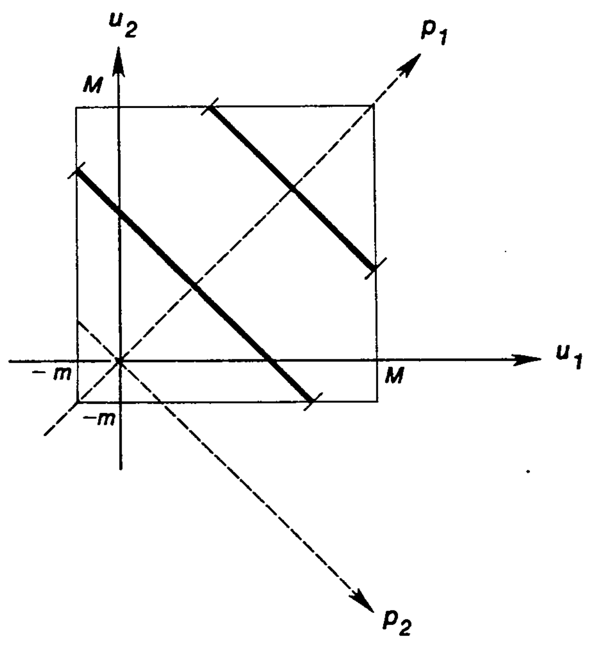

FIGURE 10. In unit coordinates, the allowed values of u_2 are $[-m,M]$ regardless of the value of u_1. In the pattern coordinates shown, the allowed values of p_2 depend on the value of p_1, as shown by the two diagonal bars. These bars correspond to two different values of p_1, and their length indicates the allowed values of p_2.

the choice of nonoverlapping patterns (e.g., over single units)—a local or quasi-local interpretation—will produce different behavior than the choice of overlapping patterns—a genuinely distributed interpretation.

Take for concreteness the allowed unit coordinate range $[-m,M]$ *to be* $[-1,+1]$.

Consider for example a local conceptual interpretation in which Concepts 1 and 2 are represented by Units 1 and 2, respectively. In unit coordinates, then, the representation of the hypotheses are $<1,0,0,0, \ldots ,0>$ *and* $<0,1,0,0, \ldots ,0>$. *Then the superposition of these two states is simply* $<1,1,0,0, \ldots ,0>$. *This falls within* S, *and is therefore kinematically allowed.*

By contrast, consider the simple distributed representation in which the two hypotheses are respectively represented by $<1,1,0,0, \ldots ,0>$ *and* $<1,-1,0,0, \ldots ,0>$. *Now the superposition of the two states is* $<2,0,0,0, \ldots ,0>$ *which is kinematically* forbidden. *The situation is graphically depicted in Figure 11.*

The basic observation is that superimposing states representing maximal confidence in two conceptual hypotheses is kinematically allowed when the corresponding patterns do not overlap—always true in a local interpretation—but kinematically forbidden when they do overlap—sometimes true in a genuinely distributed interpretation.

The kinematic *restriction that states stay inside the hypercube has a* dynamical *consequence in the sigmoidal function F. It will now be verified that F does indeed lead to a greater difficulty in superposing* $<1,1>$ *and* $<1,-1>$ *than in superposing* $<1,0>$ *and* $<0,1>$. *(All coordinates in this note will be unit coordinates.)*

Let $\mathbf{F}$ *be the function that takes a vector, passes all its unit coordinates through F, and uses the resulting numbers as the unit coordinates of the output. I will show that F retards the growth in length of a vector along the edge (the* $<\alpha,0>$ *direction or* $\mathbf{e}$*) more than along the diagonal (the* $<\beta,\beta>$ *direction or* $\mathbf{d}$*). These retardation factors are*

$$R_{\mathbf{e}} = \frac{|\mathbf{e}|}{|\mathbf{F}(\mathbf{e})|} = \frac{\alpha}{F(\alpha)}$$

and

$$R_{\mathbf{d}} = \frac{|\mathbf{d}|}{|\mathbf{F}(\mathbf{d})|} = \frac{\sqrt{2}\beta}{\sqrt{2}F(\beta)} = \frac{\beta}{F(\beta)}.$$

As shown in Figure 12, these retardation factors are the reciprocals of the average slope of the F curve between the origin and the x values α *and* β, *respectively. Since the F curve is concave downward, as x increases, this average slope diminishes so its reciprocal increases. That is, the retardation will be greater as* α *and* β *grow; F squashes vectors more and more strongly the closer they get to the edge of the state space. A fair comparison between the retardation along* $\mathbf{e}$ *and* $\mathbf{d}$ *requires that these vectors be of equal length. In that case,* α *is greater than* β *(by a factor of* $\sqrt{2}$*); this means the retardation is greater for* $\mathbf{e}$, *i.e., along the edge.*

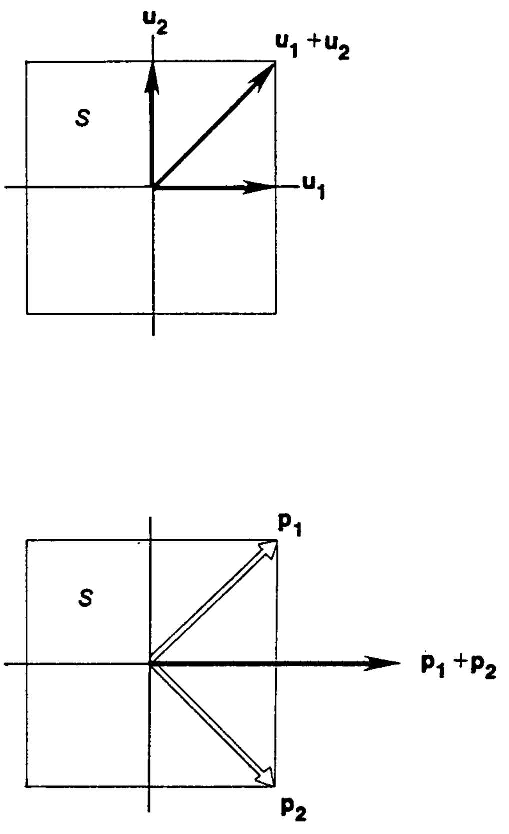

FIGURE 11. *A*: In a local interpretation in which Concepts 1 and 2 are represented by vectors $\mathbf{u}_1$ and $\mathbf{u}_2$, the superposition $\mathbf{u}_1 + \mathbf{u}_2$ lies in S (along the diagonal). *B*: In a distributed interpretation in which Concepts 1 and 2 are represented by vectors $\mathbf{p}_1$ and $\mathbf{p}_2$, the superposition $\mathbf{p}_1 + \mathbf{p}_2$ lies outside of S (in the direction of an edge).

Competition of patterns that are forbidden to superimpose—what I will call *natural competition*—does not exist in linear models. The subsequent discussion of nonlinear models will focus on natural competition and how it distinguishes between local and distributed models.

Natural competition resembles inhibition, which of course can be present in linear models. There is a crucial difference, however. Inhibition is *explicitly inserted into the interconnection matrix.* Competition arises independently of the interconnection matrix; it depends on the overlap of patterns and on the nonlinearity. Loosely, we can say that whether two states naturally compete depends only on *kinematics* (their position in state space relative to the edges of the space), and not on

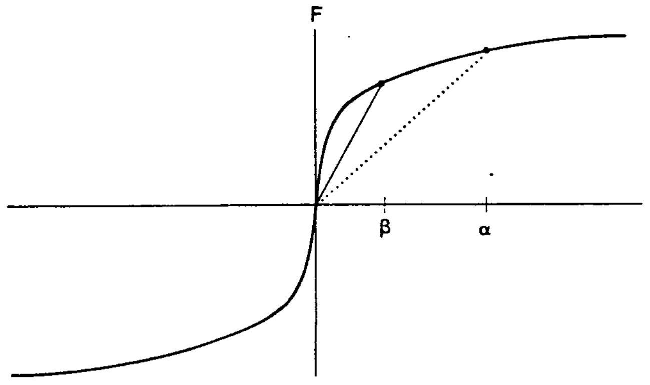

FIGURE 12. The slope of the solid line, $F(\beta)/\beta$, exceeds the slope of the dotted line, $F(\alpha)/\alpha$, because $\beta<\alpha$ and F is concave downward (for positive inputs).

the *dynamics* (interconnection matrix); whether two states mutually inhibit depends on the dynamics but not on the kinematics. If natural competition is to be exploited computationally, distributed interpretations are needed.

For expository convenience, for the remainder of this chapter, except where otherwise indicated, we take $M = 1$ and $m = 1$, so the allowed activation range is $[-1,1]$.

The plan for the remainder of this chapter is first to explicitly show the breakdown of the isomorphism hypothesis for nonlinear models, next to analyze natural competition and its effects, and finally to summarize the results and discuss the value of a mathematical approach to the mind/brain problem.

THE ISOMORPHISM HYPOTHESIS FAILS IN NONLINEAR MODELS

To investigate the isomorphism hypothesis in nonlinear PDP models, as before we take a model with a distributed interpretation and transform the description from unit to pattern coordinates. We then compare the new form of the evolution equation with the original form; if the form has changed, the hypothesis fails. To start with, we restore the nonlinear sigmoidal function F to the evolution equation; we will discuss the thresholding function G later.

Our evolution equation in unit coordinates is Equation 1A with G removed:[8]

$$u_\nu(t+1) = F[\sum_\mu W_{\nu\mu} u_\mu(t)]. \tag{6A}$$

Using Equations 3 and 5, this can be transformed to the pattern coordinates:

$$p_i(t+1) = \sum_\nu P^{-1}{}_{i\nu} F\left[\sum_k P_{\nu k} \sum_j I_{kj} p_j(t)\right]. \tag{6B}$$

Now in the linear case, when F is absent, the pattern matrices $\mathbf{P}^{-1}$ and $\mathbf{P}$ cancel each other out: the patterns don't matter. By definition of nonlinearity, here F prevents the cancellation, and the equation in pattern coordinates is more complicated than it is in unit coordinates. *There is no isomorphism of levels.* The choice of patterns can matter behaviorally.

As in the linear case, we can construct a higher-level model, with one unit for each pattern. The evolution equation for the higher-level model is Equation 6B; it is not the equation for a PDP model, so the higher-level model, while behaviorally equivalent to the lower-level model, *is not a PDP model.* Here is how the higher-level model works. The units add up the weighted sum of their inputs from other units, using the inference matrix **I**, just as in the linear case. However, what happens next is complicated. To determine its new value, a unit must find out what the weighted sum of inputs is *for all the other units*, form a weighted sum of these (with weights determined by the patterns), and then pass the resulting value through the nonlinear function F. But this is not the end. Now the unit must find out what number all the other units have gotten out of F, then form a weighted sum of these (again, with weights determined by the patterns). This finally is the new value for the unit.

Thus, what distinguishes the higher- and lower-level models is that the nonlinearity in the lower-level model is applied *locally in each unit* to its weight sum of inputs, while in the higher-level model the nonlinearity is applied *globally* through consultation among all the units. It is this nonlocal nonlinearity that makes the higher-level model not a PDP model.

[8] Using the vector-valued function **F** defined in the previous section, the evolution equation can be written in a concise, coordinate-free form:

$$\mathbf{s}(t+1) = \mathbf{F}[\mathbf{U}\,\mathbf{s}(t)]$$

where **U** is the linear transformation on *S* which is represented by matrix **W** in unit coordinates and by matrix **I** in pattern coordinates. This equation is the nonlinear generalization of Equation 2B.

The higher-level model just considered has the same behavior as and different processing rules from the lower-level model. We can also consider another higher-level model that has the same processing rules as and different behavior from the lower-level model. This model also has one unit for each pattern of the lower-level model, and also uses the interconnection matrix **I**. However, each unit computes its new value simply by passing its weighted sum of inputs through F, without the added complications of consulting other units as in the other higher-level model. Because these complications are removed, the behavior of this model will be different from the lower-level model; that is, if the two models were started in corresponding states and given corresponding inputs, they would *not* continue to stay in corresponding states. However, the linear core of this higher-level model uses the correct matrix **I**, so its behavior should be related to that of the lower-level model in meaningful ways. A more precise comparison has not been carried out.[9]

NATURAL COMPETITION IN DISTRIBUTED NONLINEAR MODELS

One aspect of the level isomorphism for linear models is that there is no competition between patterns that share units; they simply superimpose without conflict. As discussed in the section "Nonlinearity and Restricted State Space," however, in nonlinear models, patterns that fall on the boundary of the state space space—involving saturated units—often cannot superimpose and remain in the state space. This gives rise to natural competition between strong patterns that require common units.

In this section I will analyze a simple example of natural competition, contrasting the cases of distributed and local models. I will work through the effect for a *particular* nonlinear function F, and then show it obtains for all suitable functions F. I will then show explicitly how

[9] A possibility that deserves investigation is that the pattern coordinates determined by specific activation patterns should be defined some way other than through superposition, i.e., other than as a change of basis in a vector space. If this approach succeeded in saving the isomorphism hypothesis for the nonlinear case, it could destroy it for the linear case, for which change of basis is guaranteed to leave the evolution equation invariant. To save the isomorphism hypothesis in the nonlinear case, however, the pattern coordinates would probably have to be determined *jointly* by the distributed patterns *and* the nonlinearities; a successful procedure might well reduce to change of basis in the limit as the nonlinearity disappears.

the effect can simulate inhibitory connections between units representing incompatible hypotheses in the higher-level model.

Consider a PDP model with a distributed interpretation involving the patterns $<1,1>$ *and* $<1,-1>$. *A superposition of these two states with weights .3 and .2, respectively, would produce, in a linear model, the state with unit coordinates* $<.5,.1>$ *(this state has pattern coordinates* $<.3,.2>$ *of course). In the nonlinear model, the numbers .5 and .1 are each passed through the function F to get the unit coordinates of the new state. Consulting the F function of Figure 13, we see that* $F(.5) = .9$ *and* $F(.1) = .6$, *so the new state has unit coordinates* $<.9,.6>$. *The pattern coordinates of this state, as shown in the section called "Pattern Coordinates," are* $<.75,.15>$. *Thus the strength of pattern 1 relative to pattern 2 has been* amplified *by the factor*

$$\frac{.75/.15}{.3/.2} = 3.33.$$

The dominance of the stronger pattern has been enhanced by the nonlinearity, just as the dominance of stronger nodes is increased by mutual inhibition. This is natural competition.

This competitive effect is not present if the patterns do not overlap. In fact the nonlinearity diminishes dominance in this case. If the "patterns" are $<1,0>$ *and* $<0,1>$*—i.e., if the model is local—and we again consider combining .3 of the first with .2 of the second, then the weighted sum is* $<.3,.2>$ *and the new state, after passing the unit coordinates through F, is* $<.78, .7>$ *The "amplification" factor is therefore*

$$\frac{.78/.7}{.3/.2} = .74.$$

In other words, the nonlinearity is actually countercompetitive for local models.

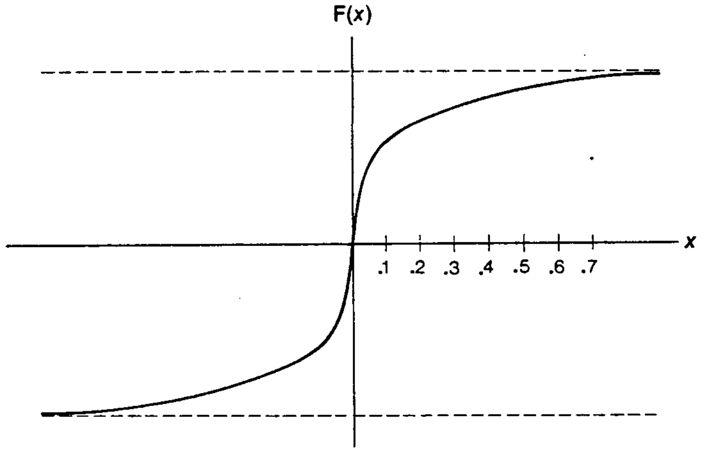

FIGURE 13. An example of a sigmoidal function F.

For these patterns, the conclusions that the amplification factor is greater than one for the distributed model and less than one for the local model is not dependent on the particular F values used above; they hold for any negatively accelerated F function. The distributed amplification is (letting .3 be replaced by x and .2 by y):

$$\frac{\tfrac{1}{2}[F(x+y)+F(x-y)]\,/\,\tfrac{1}{2}[F(x+y)-F(x-y)]}{x\,/\,y}.$$

Letting $x+y = w$ and $x-y = z$, this becomes

$$\frac{\tfrac{1}{2}[F(w)+F(z)]\,/\,\tfrac{1}{2}[F(w)-F(z)]}{\tfrac{1}{2}[w+z]\,/\,\tfrac{1}{2}[w-z]} = \frac{\tfrac{1}{2}[F(w)+F(z)]\,/\,\tfrac{1}{2}[w+z]}{[F(w)-F(z)]\,/\,[w-z]}.$$

As shown in Figure 14, this is the ratio of the slopes of two lines that can be drawn on the graph of F, and because F is negatively accelerated, this ratio must be greater than one.

The local amplification is simply

$$\frac{F(x)/F(y)}{x/y} = \frac{F(x)/x}{F(y)/y}.$$

This is the ratio of the average slope of F between the origin and x and between the origin and y; as already pointed out in section 10, the negative acceleration of F implies that this slope will be less for the greater interval (x), so the local amplification is always less than one.

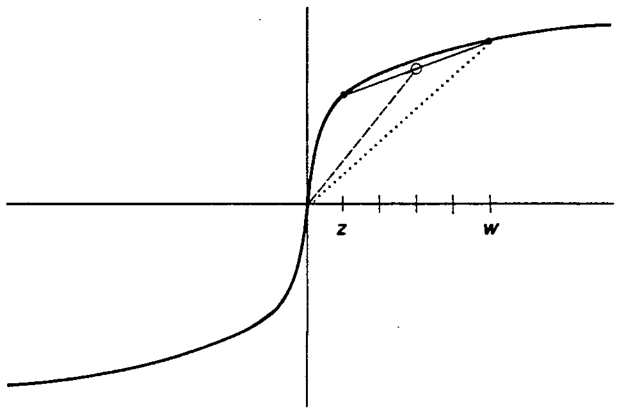

FIGURE 14. The slope of the solid line is $[F(w)-F(z)]/(w-z)$. The point marked x is the midpoint of this line—the vector average of the endpoints—with coordinates $<\tfrac{1}{2}[w+z],\tfrac{1}{2}[F(w)+F(z)]>$. Thus the slope of the dashed line is

$$\frac{\tfrac{1}{2}[F(w)+F(z)]}{\tfrac{1}{2}[w+z]}.$$

Since F is concave downward, the slope of the dashed line is greater (slope of dashed line > slope of dotted line > slope of solid line).

It is useful to explicitly relate natural competition to mutual inhibition; we shall do this for the above case. The pattern coordinates of the state created in the nonlinear model by superposing <1,1> with strength x and <1,−1> with strength y are

$$p_1 = \tfrac{1}{2}[\mathrm{F}(x+y) + \mathrm{F}(x-y)]$$
$$p_2 = \tfrac{1}{2}[\mathrm{F}(x+y) - \mathrm{F}(x-y)].$$

Now let us substitute

$$\mathrm{F}(x+y) = s_+ \, [x+y]$$
$$\mathrm{F}(x-y) = s_- \, [x-y].$$

s_+ is the average slope of F between the origin and $x+y$; this is less than s_-, the average slope of F between the origin and $x-y$ (see Figure 15). This substitution gives, on regrouping,

$$p_1 = \alpha x - \gamma y$$
$$p_2 = \alpha y - \gamma x$$

where

$$\alpha = \tfrac{1}{2}(s_+ + s_-)$$
$$\gamma = \tfrac{1}{2}(s_- - s_+).$$

These equations, displayed graphically in Figure 16, show that the effect of the nonlinearity on the $<x,y>$ combination is the same as having an inhibition of strength γ between units representing the initial and final strengths of the different patterns, along with an excitation of strength α between units for the initial and final values of the same patterns. The magnitude of the inhibition γ depends on the difference in average slope of F between the origin and $x-y$ on the one hand and $x+y$ on

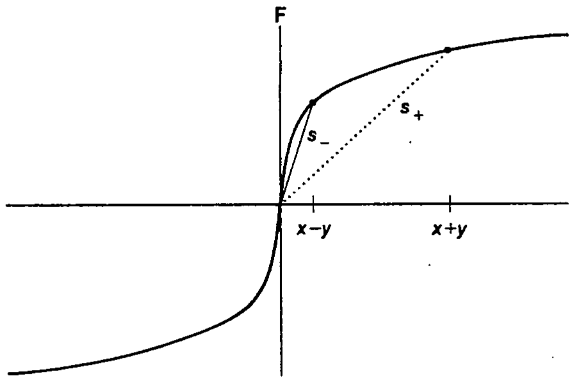

FIGURE 15. Because F is negatively accelerated, the slope of the solid line, s_-, is greater than the slope of the dotted line, s_+.

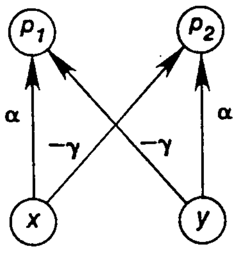

FIGURE 16. A network displaying the effective inhibition produced by natural competition between overlapping patterns $\mathbf{p}_1$ and $\mathbf{p}_2$.

the other. In the region sufficiently close to the origin that the graph of F is nearly linear, γ is zero. Thus for sufficiently weak patterns there is no competition; how weak is "sufficient" will depend on how large the linear range of F is.

The conclusion, then, is that with overlapping patterns like those occurring in truly distributed models, F *amplifies* differences in the strengths of patterns; with nonoverlapping patterns like those of local or quasi-local models, F has the opposite effect. This is natural competition at work.

When viewed at the higher level of patterns, natural competition acts like inhibition between overlapping patterns. The nonlinearities *automatically* create significant inhibition between *strong* patterns but insignificant inhibition between *weak* patterns. So for a distributed conceptual interpretation, it is impossible to have a *high* degree of confidence in more than one conceptual hypothesis represented on a given set of units, but the system can simultaneously entertain without difficulty a low degree of confidence in several such hypotheses. If we think of the units on which conceptual hypotheses are represented as some kind of semantic features, then two hypotheses that call for different values of the same features—two hypotheses represented by overlapping patterns—are hypotheses that are semantically incompatible. Thus distributed representation offers an attractive form of automatic inhibition between mutually incompatible hypotheses.

These considerations suggest that we could try to make a higher-level PDP model approximately behaviorally equivalent to a distributed nonlinear lower-level PDP model by inserting additional inhibitory weights between units representing competing patterns. These extra weights might approximately serve the function of the complicated nonlocal nonlinearity in the truly isomorphic non-PDP higher-level model. Of course, assigning fixed magnitudes to those weights would only approximate the real situation in the lower-level model in which the degree of competition varies with the strength of the patterns. With nonlinear models, it does seem that higher-level models must incorporate some form of complexity that goes beyond the standard PDP framework in order to really capture the subtlety of the interaction of patterns in the lower-level model; the degree of failure of the isomorphism hypothesis seems to be significant.

The failure in the isomorphism hypothesis induced by F is compounded by the nonlinear thresholding function, G. In the previous section we saw that the nonlinearity introduced by F becomes nonlocal in the higher-level model. Exactly the same analysis applies to the nonlinearity introduced by G.

It is interesting to consider whether a *local* thresholding of the activation of patterns in the higher-level model, since it can't be created by putting a local G in the lower-level model, can be affected some other way. Natural competition functions like a threshold on inhibition. When a concept is weakly present, it creates essentially no inhibition; when it is strongly present, it generates considerable inhibition. Thus, it is as though there were a threshold of activation below which a concept is incapable of influencing other concepts inhibitively. The threshold is not sharp, however, as there is a gradual transition in the amount of inhibition as the concept strengthens.

CONCLUSION

If mind is to be viewed as a higher-level description of brain, some definite procedure is needed to build the higher-level description from the lower. One possibility is that the higher level describes *collective* activity of the lower level, for example, patterns of activation over multiple lower-level units. In the analysis of dynamical systems, it is customary to create such higher-level descriptions, using higher-level units that are collective aspects of many lower-level units. The higher-level description employs new mathematical variables, each of which is defined by combining many variables in the lower-level description. This requires that the lower-level description be expressed in terms of mathematical variables.

PDP models constitute an account of cognitive processing that does rest on mathematical variables: the activation of units. Thus these models can be described at a higher level by using the preceding method. The lower-level description of the system amounts to a set of variables and an evolution equation; the higher-level description is created by defining new variables out of many old ones and substituting the new variables for the old in the evolution equation.

Analyzing PDP models in this way leads to several observations. The core of the evolution equation, where the knowledge of the system is employed, is linear. The dynamical systems corresponding to PDP models are quasi-linear. A central kinematic feature of these systems is that the state spaces lie in vector spaces, and linear operations play a very special role. Superposition is fundamental to the way parallel distributed processing works. Linear algebra gives us both concepts for understanding PDP models and techniques for analyzing them. It tells us that there are many equally good coordinate systems for describing the states, and how to transform descriptions from one coordinate system to another. In particular, it suggests we consider a coordinate system based on the patterns, and use this to frame our higher-level description. It tells us how to transform the knowledge (interconnection matrix) from one level of description to the other. Linear algebra tells us that linear operations are invariant under this kind of transformation, and therefore if the evolution equation is linear, its form will be the same in the two descriptions. This is the isomorphism of levels for linear PDP models.

Linear algebra also alerts us to the fact that edges to the state space interfere with superposition. This leads to an evolution equation that modifies pure linear superposition by adding nonlinear operations. The breakdown of superposition at the boundaries of state space leads to competition between states that cannot superpose. The lack of invariance of the bounded state space under linear operations leads to a breakdown of the isomorphism of levels in the corresponding nonlinear PDP models.[10]

The concepts of linear algebra add considerably to our understanding of PDP models, supplementing the insight that comes from simulating PDP models on computers. We can analyze with precision, for example, the effects of localized damage or of synaptic modification upon the higher-level description of the processing. We can understand why the choice of patterns doesn't matter in linear models and why it does in nonlinear models. We can even get some handle on what kind of effects these choices have for nonlinear models, although the picture needs much more clarification.

[10] Note that the nonlinearity in quasi-linear systems is minor compared to that in highly nonlinear systems. In PDP models, the knowledge is contained in the linear part—it is this part that is learned, for example, in adaptive models—while the nonlinearity is uniform across the units, and fixed from the outset. In a highly nonlinear model, the activation value of a unit would be a nonlinear function of the other activations, with parameters encoding the unit's knowledge that are used in arbitrary ways, rather than merely as coefficients in a weighted sum. In such a model, superposition and linear algebra might have no relevance whatever.

The isomorphism of levels hypothesis constitutes a mathematically analyzable question about the mind/brain duality, within the framework of PDP models of cognition. These models serve well because of their independent interpretations as models of neural and conceptual processing.

While the isomorphism of levels hypothesis speaks to a fairly philosophical issue, the analyses of this chapter show that no strong argument about the hypothesis can have validity unless it makes sharp enough distinctions among models to differentiate between linear and nonlinear PDP models: The matter demands a certain degree of mathematical care. From an appropriate formal viewpoint, however, we have seen that conceptual accounts of mind and physiological accounts of brain *can* be two descriptions of a single cognitive system.

ACKNOWLEDGMENTS

While I carry full responsibility for the views expressed in this chapter, the critical role played by insights of Jay McClelland and Dave Rumelhart is fully and gratefully acknowledged. This chapter is actually in major part the result of a joint effort, and I thank Dave and Jay for the ideas and understanding they have shared with me. Many thanks, too, to Eileen Conway and Mark Wallen for excellent graphics and computer support. The work was supported by the Systems Development Foundation, the Alfred P. Sloan Foundation, National Institute of Mental Health Grant PHS MH 14268 to the Center for Human Information Processing, and Personnel and Training Research Programs of the Office of Naval Research Contract N00014-79-C-0323, NR 667-437.