Chapter 2

Neurons, Matrices, and Models

Let us assume that we exist as bodies, that our bodies have heads, and our heads have brains. Brains in turn have neurons, lots of them, ten billion or more. These ten billion neurons have trillions of connections that allow the brain to carry out calculations that turn photochemical reactions in the retina into images, locate memories associated with those images, reckon their importance, and plan responses. The neurons, moreover, rely on their vast, intricate pathways to perform these calculations in several tenths of a second. The connections between neurons change as we develop and embody our memories, our learned aptitudes, our hopes, our selves.

This chapter and the next explore how neurons and their interconnections perform these feats, how they represent, process, remember and recall our engagement with the world. The initial inquiry in this chapter, however, begins not with neural physiology or anatomy but with more abstract models of connection and calculation. Underlying our understanding of how the brain works is the idea that information is stored in the strength of a neuron's connections to other neurons. Structuring and adjusting those connections shape what we know and how we process experience. Mathematicians, computer scientists and neuroscientists have developed increasingly sophisticated models to explain how

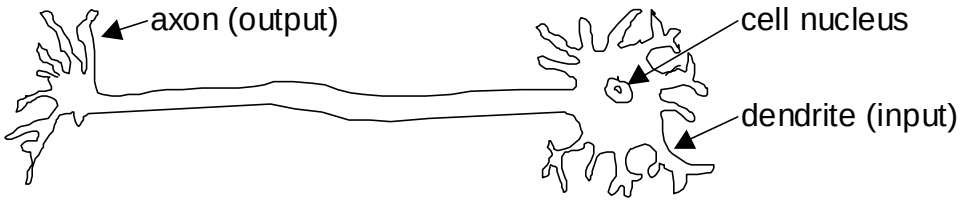
vast ensembles of neurons can in fact rely on changing connection strength to perform the sorts of calculations that make up our engagement with the world.

This chapter begins with the simplest model—a single layer of input neurons connected to a single output neuron—and then introduces progressive layers of complexity that modelers have added to more closely approximate the structures found in the brain. These models, however, are not just neutral paradigms: they commit us to a particular way of understanding the nature of perception, cognition, and memory. We therefore need to look at the details of the basic neural network models, including aspects of their mathematical formulation, to see both what these commitments are and why the models give rise to them.

Neurons

Neurons are the basis of the story in this chapter. They provide the inspiration for the model for the mind presented below, and they provide an empirically testable biological system that challenges and complicates the model.

Neurons are the class of cell types that make the nervous system work. They originate nervous system activation, transmit, and process it. Since the cells that embody the senses—eyes, ears, nose, touch, etc.—have different types of inputs, their structures are correspondingly varied. Similarly, the long neurons of the central nervous system, the neurons connected to muscles, and the various types of neurons inside the brain all have different roles and are structurally differentiated. Nonetheless, a generalized neuron looks like this:



The generic neuron's job is to receive chemical messages at the dendritic end and, when the messages cause the neuron to reach its activation threshold (the firing level), to transmit a signal to the axon end, where it is in contact with the dendrites of yet more neurons. These neurons in turn evaluate all the signals they receive, and so on. The "message" is a neurotransmitter secreted by the axons at the synaptic junction. (Here too, however, the situation is not neat. There are axon-to-axon connections, as well as dendrite-to-dendrite.) The neuron in its "quiet" state keeps an ion pump running on the cell membrane that moves positively charged sodium and calcium ions out of the cell and moves (positively charged) potassium in. There are also large organic molecules that carry a negative charge that cannot pass through the membrane and remain within the cell. The net result is a complex balancing of charge and ion concentration that can be strongly affected by changing the working of the ion pump. Neurotransmitters do precisely that. They cause the pump to either move yet more sodium out (hyperpolarization) or let it in (depolarization). If the net effect of all the neurotransmitters received by the dendrites causes a depolarization of a high enough value, the neuron "fires," that is, sends a pulse of local depolarization (called a spike) down the surface of the axon. At the synaptic junction at the end of the axon, the spike causes the cell to release more neurotransmitter. The released neuropeptide can be either

hyperpolarizing (which *inhibits* the firing of the receiving neuron) or depolarizing, which is excitatory. Finally, what one usually measures in looking at neural activity is the *spiking rate*—how frequently spikes of depolarization are being sent down the axon—rather than simply if the neuron fires at all.

This simple version of the neuron is all we need for most of the discussion that follows. Nonetheless, it is important to at least note that this model is a gross simplification. There are at least forty different neuropeptides at work in the brain, with mostly uncharted functions. Moreover, biological signals also influence the number of receptors for neurotransmitters, the efficiency of the pump, and on and on, while the structure of the synaptic junction also has an important role. As will be noted many times, given the complexity of biological systems, the phrase "biological reductionism" is a contradiction in terms (unless one is using "reductionism" technically.)[1]

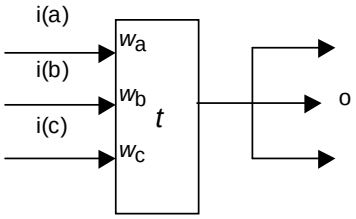
Artificial Neurons

The artificial neuron, taken by itself, is not very exciting. It is just a chip with a few wires going in and a few going out. For each input line there is an associated value

---

[1] The technical use of "reduction"—and particularly its contrast with "eliminativism"—is sufficiently important as to be worth a footnote. When we can understand a theory **A** strictly in terms of the workings of a theory **B**, then we have reduced **A** to **B**. If we now consider **A** to be simply false or so inadequate that it no longer serves any purpose, this is an *eliminative* account of **A**. The phlogiston theory of heat is a standard example. In contrast, if **A** remains a useful level of analysis, then the account is *reductive*. Chemistry is a standard example: we believe that in the end all chemical properties can be explained through quantum mechanics, but the study of chemistry usually deals with effects at a different level from the quantum mechanical and still provides a powerful level of analysis.

that indicates the weight of the connection, rather like the number of receptors for neurotransmitters at the synaptic junction. Finally, the input can be either excitory (a positive weighting) or inhibitory (a negative value). The job of the neuron, at least in the first approach, is to add up the values of the active input lines, see if they reach a preset threshold *t* value and, if so, activate the output lines. That's it.

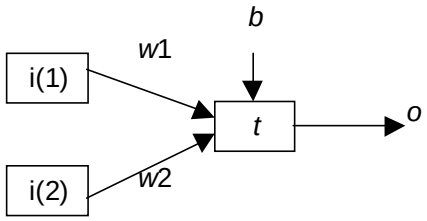


$$\text{output} = 1, \text{ if } \Sigma \text{ input}(i) \times w_i > t$$

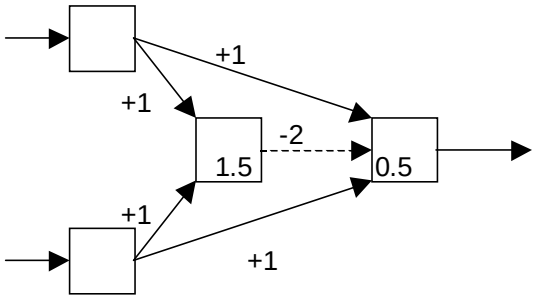
$$0, \text{ otherwise}$$

What does one *do* with an artificial neuron? The fact that a device so simple can do anything at all attracted some enthusiastic interest in the early days of the study of artificial intelligence (AI). In the 1950s interest focused on the "perceptron," a single layer of neurons plugged into a single output unit. Research had shown that perceptrons could be trained to recognize patterns and classify them as either "yes" or "no." Interest largely evaporated, however, when a serious flaw was revealed. A classic study (Minsky and Papert) demonstrated that the perceptron was almost useless for general computation. Here is one of the problems. Suppose you want the neuron system to respond like an XOR (exclusive OR) machine.

This book is part of an effort to reduce questions of "mind" to biology, but it by no means sees any reduction in complexity, subtlety, or beauty as part of the process.



That is, two input neurons would send messages to a third, output neuron as follows: if both inputs are either active ([1,1]) or inactive ([0,0]), then the output neuron's line is inactive, [0]. If the inputs are different (either [1,0] or [0,1]), then the output is active, [1]. Unfortunately, there are no values of w_1, w_2, b (bias), and t (threshold) that can produce this result, a result fundamental to any computing system.[2] Minsky and Papert did observe that if one added an extra neuron neither receiving direct input nor producing direct output and hidden to both (called, therefore, a hidden unit), then there is a solution:



[2] That is, one would have the following equations:

$(0 \times w_1) + (0 \times w_2) + b < t \quad \Rightarrow b < t \quad \Rightarrow t - b > 0$

$(1 \times w_1) + (0 \times w_2) + b \geq t \quad \Rightarrow w_1 + b \geq t \quad \Rightarrow w_1 \geq t - b \quad \Rightarrow w_1 > 0$

$(0 \times w_1) + (1 \times w_2) + b \geq t \quad \Rightarrow w_2 + b \geq t \quad \Rightarrow w_2 \geq t - b \quad \Rightarrow w_2 > 0$

Therefore, if $w_1 + b \geq t$, then $w_1 + w_2 + b \geq t$, since $w_2 > 0$. However, we also know

$(1 \times w_1) + (1 \times w_2) + b < t \quad \Rightarrow w_1 + w_2 + b < t$

which is a contradiction.

In this notation, the values next to the lines are the connection weights, and the values inside the boxes are the threshold values. Minsky and Papert went on to note, however, that if one allowed hidden units, then the system no longer could be trained in the manner of a perceptron and was thus unmanageable in its behavior. They could see no way a system with hidden units could be made to work as a computing device. Thus, for many years, interest in artificial neurons all but died.

Almost, but not quite. One element of the intellectual allure of artificial neurons is the strong suspicion that *something* like the model most surely must be at work in the brain. The standard symbolic processing approach to AI that flourished from the 1950s produced enormously complicated programs to mimic human judgments. Yet the human brain has the so-called "100 step program constraint." That is, the process of reacting to messages from the dendrites, producing a spike, transmitting it along the axon membrane, then releasing neurotransmitters at the synaptic junction takes about a millisecond ($1/1000^{th}$ of a second), while responses to events in the environment take on the order of a tenth of a second. So, whatever happens must take no more than about 100 steps. It is correspondingly clear, given the number of neurons (10 billion) and the number of synaptic junctions for each neuron ($\sim$1000), that the brain does its work through massively parallel computation. The question of how to get a huge system of (relatively) simple computing devices to work together to do what a brain does remained very compelling, even if no answers offered themselves.

Work with artificial neurons continued. The goal was largely the same as with the perceptron, but in a more generalized form. Researchers sought to teach

neurons to recognize patterns, i.e., to systematically modify the connection weights in arrays of neurons so that input patterns could be mapped to output values. This pattern recognition was seen as the particular strength of neural networks. Pattern recognition covers a lot of ground: numbers can be represented in binary formats as they are in a normal digital computer, and arithmetic, for instance, can be performed by mapping two binary strings as input to a single binary string (their sum) as output. In general, however, researchers have quite reasonably been much more interested in more complex types of pattern recognition such as visual scanners and the like. In any case, the difficulty, as discussed above, was in training an arbitrarily arrayed set of neurons to respond to produce the desired result by the time the activation had propagated to the output layer.

During the rather quiet years after the perceptron fizzled, researchers did what they tend to do: they chipped away at small portions of the larger general problems of neural networks. Some worked on the mathematical formalisms. Some sought solutions to particular problems that addressed narrower aspects of the question of training hidden units. Practical experience and theoretical models gradually accumulated until it reached the point where it became undeniable that neural nets really could work. McClelland and Rumelhart's two-volume anthology of studies, *Parallel Distributed Processing*, effectively announced the rebirth of the field to the larger scientific community.

From Artificial Neurons to Parallel Distributed Processing

Neurons have both many input synapses on which they perform simple arithmetic operations and many output connections to which they transmit the results. Matrix mathematics lets us model how systems of layers of neurons work. Modeling neural nets through matrix approximations allows us to study in particular how the emergent properties of complex systems evolve. The fundamental vector and matrix operations used in these models are not hard, even though some aspects of the models are growing ever more sophisticated and complex. The most basic formula in modeling neural networks, for example, is Hebb's algorithm for changing synaptic weights:

$$\Delta w_{ij} = \eta a_i b_j$$

(That is, the change in weight between the i^{th} input neuron and the j^{th} output neuron is a constant (η) times the activation states of the two neurons.) The synaptic strengths between neurons here takes the form of a "weighting matrix," while the activation states of both the input layer of neurons and the output layer are described as vectors. Because the world of distributed processing and distributed representation is difficult to approach without basic knowledge of these formalisms, a brief overview of the mathematics follows.

Scalars, Vectors, and Matrices

In everyday experience, quantities come in two types: *scalars* and *vectors*. I weigh about 170 pounds, a single number referred to as a **scalar**. In contrast, the Earth's

pull on me not only has an *amount*, but also a *direction* (toward the center of the Earth). This pull on me is a three-dimensional **vector** we can represent through a set of three scalars.

The numbers can be, for example, the usual type we think of: the components of the pull along the three axes (g_x,g_y,g_z). Here we use the earth's axis of rotation as the z-axis, and the x and y-axes are at a right angle on a plane that cuts through the center of the Earth at right angles to the z-axis. The choice of axes, however, in fact can be arbitrary: almost any three lines will do, even if these are the most convenient. Or one can use a completely different representational scheme that combines two angles and a length (r,θ,φ). The coordinate system is just a strategy for dividing up three-dimensional space. Because all the coordinate systems are dividing up the *same* space, however, the relationship *between* representational systems is *not* arbitrary and can be precisely defined by a **matrix** that maps the three values of the "pull" vector from one coordinate system into another. So, while a 3-d vector like the pull of gravity, or any force, velocity, or momentum can be described as a set of three numbers, one is better off thinking of a vector as a singular, internally structured entity represented by a more abstract notation.

$$\begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} = \begin{bmatrix} g_r \\ g_\theta \\ g_\varphi \end{bmatrix} = \vec{g}$$

The arrow on top designates a vector.

Many of the mathematical operations we perform on scalars can also be performed on vectors, so long as they are elements in a *linear space*.[3] Vectors can be multiplied by scalars:

$$\vec{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \rightarrow 5\vec{a} = \begin{bmatrix} 5a_1 \\ 5a_2 \\ \vdots \\ 5a_n \end{bmatrix}$$

Vectors of the same dimensionality can be added:

$$\vec{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \quad , \quad \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad , \quad \vec{a} + \vec{b} = \begin{bmatrix} a_1 + b_1 \\ a_2 + b_2 \\ \vdots \\ a_n + b_n \end{bmatrix}$$

Note that since addition just involves the addition of elements in each separate dimension, the usual rules for scalar addition also apply to vectors. That is,

$$\vec{a} + \vec{b} = \vec{b} + \vec{a}$$

$$(\vec{a} + \vec{b}) + \vec{c} = \vec{a} + (\vec{b} + \vec{c})$$

However, because vectors are multidimensional, they have some important properties not found in scalars that are related to their dimensionality. The most important is a pair of properties called *linear combination* and *linear independence*. That is, a vector ***w*** is said to be a linear combination of ***u*** and ***v*** if there are scalars a and b such that

$$\boldsymbol{w} = a\boldsymbol{u} + b\boldsymbol{v}$$

---

[3] This is essentially a tautology: linear spaces are in fact defined by the applicability of such

For example, if $u = \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix}$ and $v = \begin{bmatrix} 5 \\ 7 \\ 0 \end{bmatrix}$, then we can create linear combinations:

$$w_1 = 3u + (-1)v = \begin{bmatrix} -2 \\ 2 \\ 0 \end{bmatrix}$$

$$w_2 = (-2)u + 2v = \begin{bmatrix} 8 \\ 8 \\ 0 \end{bmatrix}$$

Both $\begin{bmatrix} -2 \\ 2 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 8 \\ 8 \\ 0 \end{bmatrix}$ can be described as linear combinations of $\begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 5 \\ 7 \\ 0 \end{bmatrix}$.

Correspondingly, a vector $\boldsymbol{w}_3$—for example, $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$—is linearly independent of $\boldsymbol{u}$ and $\boldsymbol{v}$ if such scalars do not exist. In the example, since the third value in both vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is 0, there are no scalars that, multiplied by zero and summed, will equal 1. It turns out that for an $\boldsymbol{n}$-dimensional space, any set of $\boldsymbol{n}$ linearly independent vectors can be used in linear combination to produce all the other vectors in the space. In the case above, since $\begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 5 \\ 7 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ are independent of one another, they can be multiplied by appropriate scalars and added together to get *any* other three-dimensional vector. A set of $\boldsymbol{n}$ linearly independent vectors forms a *basis* for the vector space and is said to *span* it.

---

rules as commutation, association, and so on, modeled on the properties of real numbers.

There are several different versions of what we might call multiplication, and they get a bit more conceptually complex. The first version is called the *dot-product*, or *inner product*, which produces a scalar, not another vector:

$$\vec{\mathbf{a}} \bullet \vec{\mathbf{b}} = a_1 b_1 + a_2 b_2 + \cdots a_{\mathbf{n}} b_{\mathbf{n}} \quad \text{that is} \quad \sum_{i=1}^{\mathbf{n}} a_i b_i$$

What the inner product measures is the amount of *overlap* of two vectors, called the projection. Consider some of the arithmetic properties of the inner product. Since

$$\vec{\mathbf{a}} \bullet \vec{\mathbf{b}} = \sum_{i=1}^{\mathbf{n}} a_i b_i$$

therefore

$$\vec{\mathbf{a}} \bullet (\vec{\mathbf{b}} + \vec{\mathbf{c}}) = \sum_{i=1}^{\mathbf{n}} a_i (b_i + c_i) = \sum_{i=1}^{\mathbf{n}} a_i b_i + \sum_{i=1}^{\mathbf{n}} a_i c_i = \vec{\mathbf{a}} \bullet \vec{\mathbf{b}} + \vec{\mathbf{a}} \bullet \vec{\mathbf{c}}$$

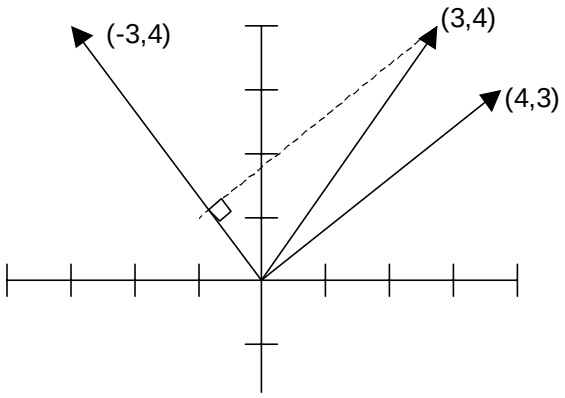
and

$$\vec{\mathbf{a}} \bullet c\vec{\mathbf{b}} = \sum_{i=1}^{\mathbf{n}} a_i (c b_i) = c \sum_{i=1}^{\mathbf{n}} a_i b_i = c(\vec{\mathbf{a}} \bullet \vec{\mathbf{b}})$$

Finally,

$$\vec{\mathbf{a}} \bullet \vec{\mathbf{b}} = \sum_{i=1}^{\mathbf{n}} a_i b_i = \sum_{i=1}^{\mathbf{n}} b_i a_i = \vec{\mathbf{b}} \bullet \vec{\mathbf{a}}$$

Consider a two-dimensional case:



The vector $\begin{bmatrix}3\\4\end{bmatrix}$ of course completely overlaps with itself and $\begin{bmatrix}3\\4\end{bmatrix} \bullet \begin{bmatrix}3\\4\end{bmatrix} = 3\times3 +$

$44 = 25$. Now compare the overlap of $\begin{bmatrix}3\\4\end{bmatrix}$ with $\begin{bmatrix}4\\3\end{bmatrix}$ (i.e. $3\times4 + 4\times3 = 24$) to the

overlap of $\begin{bmatrix}3\\4\end{bmatrix}$ and $\begin{bmatrix}-3\\4\end{bmatrix}$: $3\times (-3) + 4\times4 = 7$. Looking at the graph shows that the

overlap of $\begin{bmatrix}3\\4\end{bmatrix}$ with $\begin{bmatrix}4\\3\end{bmatrix}$ is much greater. Now, the overlap of $\begin{bmatrix}4\\3\end{bmatrix}$ and $\begin{bmatrix}-3\\4\end{bmatrix}$

($4\times(-3) + 3\times4 = 0$) is a special case. When the inner product is 0, two vectors are *orthogonal*, that is, at right angles so that there is no overlap at all.

One can think of an ***n***-dimensional vector as the displacement of a point in ***n***-space (move x amount in this direction, y amount in that, and so on). **Matrices** go one step further. They are entities with *two* dimensional values (say, ***m*** and ***n***) that *map vectors from one space into another* (here, ***m***-space into ***n***-space). That is, one starts with an ***n***-dimensional vector a_n, runs it through an m x n matrix $\mathbf{M}_{m,n}$,

and produces an m-dimensional vector b_m. As with vectors, there are some basic arithmetic functions. Addition:

$$\mathbf{A} = \begin{bmatrix} a^{1,1} & a^{1,2} & \cdots & a^{1,n} \\ a^{2,1} & a^{2,2} & \cdots & a^{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a^{m,1} & a^{m,2} & \cdots & a^{m,n} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} b^{1,1} & b^{1,2} & \cdots & b^{1,n} \\ b^{2,1} & b^{2,2} & \cdots & b^{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ b^{m,1} & b^{m,2} & \cdots & b^{m,n} \end{bmatrix}$$

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} a^{1,1} + b^{1,1} & a^{1,2} + b^{1,2} & \cdots & a^{1,n} + b^{1,n} \\ a^{2,1} + b^{2,1} & a^{2,2} + b^{2,2} & \cdots & a^{2,n} + b^{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a^{m,1} + b^{m,1} & a^{m,2} + b^{m,2} & \cdots & a^{m,n} + b^{m,n} \end{bmatrix}$$

Multiplication by a scalar:

$$3\mathbf{A} = \begin{bmatrix} 3a^{1,1} & 3a^{1,2} & \cdots & 3a^{1,n} \\ 3a^{2,1} & 3a^{2,2} & \cdots & 3a^{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 3a^{m,1} & 3a^{m,2} & \cdots & 3a^{m,n} \end{bmatrix}$$

The basic function of a matrix is to map vectors from one linear space onto another. This mapping is defined in matrix notation as multiplying the matrix by a vector $\boldsymbol{v}_\mathbf{n}$ in the initial n-space to get a vector $\boldsymbol{u}_\mathbf{m}$ in the second m-space:

$$\vec{v}^n = \begin{bmatrix} v^1 \\ v^2 \\ \vdots \\ v^n \end{bmatrix} \qquad (\mathrm{A}_{m,n})\vec{v}_n = \begin{bmatrix} v_1 a_{1,1} + v_2 a_{1,2} + \cdots + v_n a_{1,n} \\ v_1 a_{2,1} + v_2 a_{2,2} + \cdots + v_n a_{2,n} \\ \vdots \\ v_1 a_{m,1} + v_2 a_{m,2} + \cdots + v_n a_{m,n} \end{bmatrix} = \vec{u}_m$$

The notation makes matrix multiplication look horrendous, but perhaps the simplest way to approach it is to see that the ith element of the new vector $\boldsymbol{u}_m$ is

equal to the dot-product of $\boldsymbol{v}_n$ with $\boldsymbol{r}_i(\mathbf{A}_{m,n})$, the ith **row vector** of $\mathbf{A}_{m,n}$ (i.e. $[a_{i,1}\ a_{i,2}\ \ldots\ a_{i,n}]$).

The final type of operation we need to know to get a sense of the properties of vectors and matrices is multiplication of matrices by matrices. If we think of a vector as an $\boldsymbol{n} \times 1$ dimensional matrix (i.e., $\boldsymbol{n}$ rows, 1 column), then matrix multiplication is just an extension of the multiplication of matrices by vectors. That is, just as the ith element of a new vector $\boldsymbol{u}_m$ is equal to the dot-product of $\boldsymbol{v}_n$ with $\boldsymbol{r}_i(\mathbf{A}_{m,n})$, the ith **row vector** of $\mathbf{A}_{m,n}$, so in multiplying $\mathbf{A}_{m,n}$ by $\mathbf{B}_{n,k}$ to get $\mathbf{C}_{m,k}$,the ith element of the jth column ($\boldsymbol{c}_{i,j}$) is the dot-product of $\boldsymbol{c}_j(\mathbf{B}_{n,k})$, the jth **column vector** of $\mathbf{B}_{n,k}$ with $\boldsymbol{r}_i(\mathbf{A}_{m,n})$:

$$\begin{bmatrix} a^{1,1} & a^{1,2} & \cdots & a^{1,n} \\ a^{2,1} & a^{2,2} & \cdots & a^{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a^{m,1} & a^{m,2} & \cdots & a^{m,n} \end{bmatrix} \times \begin{bmatrix} b^{1,1} & b^{1,2} & \cdots & b^{1,k} \\ b^{2,1} & b^{2,2} & \cdots & b^{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ b^{n,1} & b^{n,2} & \cdots & b^{n,k} \end{bmatrix} = \begin{bmatrix} c^{1,1} & c^{1,2} & \cdots & c^{1,k} \\ c^{2,1} & c^{2,2} & \cdots & c^{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ c^{m,1} & c^{m,2} & \cdots & c^{m,k} \end{bmatrix}$$

$$\text{where } c_{i,j} = \boldsymbol{row}_i(\mathbf{A}_{m,n}) \bullet \boldsymbol{column}_j(\mathbf{B}_{n,k})$$

That is

$$c_{ij} = \sum_{x=1}^{n} a_{ix} b_{xj}$$

What does matrix multiplication represent? If a matrix is a transformation that maps vectors from one space into another, then matrix multiplication transforms that transformation function. That is,

$(\mathbf{B}_{n,k})\boldsymbol{u}_k = \boldsymbol{v}_n$ [$\mathbf{B}_{n,k}$ maps k-space vector to n-space vector]

$(\mathbf{A}_{m,n})\boldsymbol{v}_n = \boldsymbol{w}_m$ [$\mathbf{A}_{m,n}$ maps n-space vector to m-space vector]

$(\mathbf{A}_{m,n})((\mathbf{B}_{n,k})\boldsymbol{u}_k) = \boldsymbol{w}_m$ [$\mathbf{A}_{m,n}$ maps ($k \rightarrow n$)-space vector to m-space vector]

$((\mathbf{A}_{m,n})(\mathbf{B}_{n,k}))\boldsymbol{u}_k = \boldsymbol{w}_m$ [$\mathbf{A}_{m,n} \times \mathbf{B}_{n,k}$ maps k-space vector to n-space vector]

To consider this mapping function, suppose we have a 2-by-3 matrix that maps 3-dimensional vectors into a 2-dimensional space:

$$\mathbf{M} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \text{ and two vectors, } \boldsymbol{a} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \text{ and } \boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

Therefore

$$(\mathbf{M})\boldsymbol{a} = \begin{bmatrix} 1\times1 + 2\times-1 + 3\times1 \\ 4\times1 + 5\times-1 + 6\times1 \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

$$(\mathbf{M})\boldsymbol{b} = \begin{bmatrix} 1\times1 + 2\times1 + 3\times-1 \\ 4\times1 + 5\times1 + 6\times-1 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

However, because the matrix maps a three-dimensional vector into a two-dimensional space, there will be many vectors that map to the same resulting value. Here, for example, an infinite number of vectors can serve as $\boldsymbol{b}$ (i.e. $(\mathbf{M})\boldsymbol{b} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$) so long as $b_2 = 3 - 2b_1$, and $b_3 = b_1 - 2$. This fact that matrices that map from higher to lower order dimensional spaces essentially aggregate large sets of the input into any given output creates the logic of feature-detecting in neural systems, where a pattern (like "$b_2 = 3 - 2b_1$, and $b_3 = b_1 - 2$") determines the common "features" to be found in the input data set. This is the simple version of what happens when very large input vectors like the data from the retina goes into a layer of processing like the lateral geniculate nucleus of the thalamus, which then extracts pattern

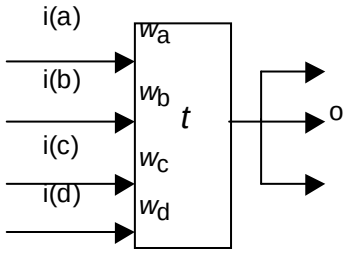
information that it transmits as smaller input arrays to the visual cortex. From layer to layer, vectors of larger dimension are mapped to vectors of smaller dimension as the most important features are extracted via the weighting matrices.

The point of all this explanation of the arithmetic notation is to show that there is nothing magical, mysterious, or even especially difficult here. Matrices and vectors allow us to describe objects in higher dimensional spaces and to perform simple mathematical operations on those objects: there is nothing here but extensions of addition and multiplication. It is important to have a clear rather than a vague understanding of what these crucial terms mean and how they are used because input and output vectors, weighting matrices, and rules for changing the values of the weighting matrix are the basic tools for neural net modeling. The ideas of dimensions and spanning sets will be important, but in the abstract they are quite simple: if one understands three dimensions, any number of dimensions is still just the same. The input and output vectors and weighting matrices can be as large as we want—that is, have as many elements as needed—without making the logic and systemic character of the operations any more complicated. A vector is a way to talk about a "thing" with an internal structure whose elements are represented by numbers: we can think of the output from the retina as such a vector, where the elements represent the individual sensory neurons and the scalars are their spiking rates. We need not worry about exactly how many neurons are sending data: what matters is the logic of processing, the operations performed on each element. Manipulation of vectors and matrices allows one to model these

operations with a clear, uncluttered notation that in fact represents an enormously complicated system of interacting parts.

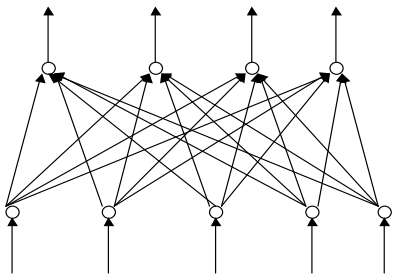
Artificial Neurons, Vectors, Matrices and Neural Nets

Let us return to our artificial neuron:



We can think of the input to the neuron as a vector $\boldsymbol{i}$, in this case with a dimensionality of 4. Similarly, the *weightings* for each of the inputs used by the neuron in calculating whether to fire is another vector $\boldsymbol{w}$, also with 4 elements. The firing function is $\mathsf{F}(\boldsymbol{i} \bullet \boldsymbol{w})$, where if $\boldsymbol{i} \bullet \boldsymbol{w} > t$ (the threshold), $\mathsf{F} = 1$, else $\mathsf{F} = 0$.

Now consider a more complicated situation. Suppose there are five input neurons in one layer connected to four output neurons in another:



To simplify the model, each input unit has only one input synaptic junction. If it fires, the unit fires, so the input the unit transmits to each of the output layer

neurons will be the same as its original input. Here, $\boldsymbol{i}$ is a vector describing the status of the five input connections of the bottom. Similarly, there is a vector $\boldsymbol{o}$ that is the status of the four output connections on the top layer. To calculate the output value of the first output neuron, we first calculate the net_1, the weighted sum of all the first unit's input values:

$$net_1 = \boldsymbol{w}_{11}\boldsymbol{i}_1 + \boldsymbol{w}_{12}\boldsymbol{i}_2 + \boldsymbol{w}_{13}\boldsymbol{i}_3 + \boldsymbol{w}_{14}\boldsymbol{i}_4 + \boldsymbol{w}_{15}\boldsymbol{i}_5$$

where , for instance, $\boldsymbol{w}_{12}$ is the weight the **first** output unit assigns to the input from the **second** input unit. Next, the firing function uses net_1 to calculate the output:

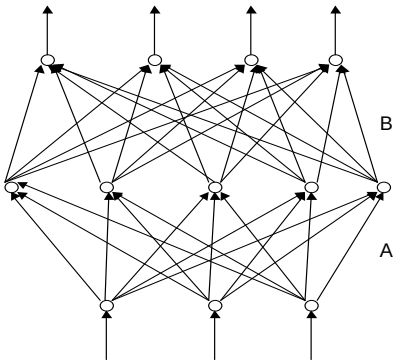
$$\boldsymbol{o}_1 = \mathsf{F}(net_1)$$

The weighting values for the connection can be seen as a matrix such that

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \\ w_{41} & w_{42} & w_{43} & w_{44} & w_{45} \end{bmatrix} \times \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \end{bmatrix} = \begin{bmatrix} net_1 \\ net_2 \\ net_3 \\ net_4 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \mathrm{F}(net_1) \\ \mathrm{F}(net_2) \\ \mathrm{F}(net_3) \\ \mathrm{F}(net_4) \end{bmatrix} = \begin{bmatrix} o_1 \\ o_2 \\ o_3 \\ o_4 \end{bmatrix}$$

Or, more simply, $o_j = \mathsf{F}((\mathbf{W}\boldsymbol{i})_j)$. (For even greater simplicity of notation, I shall drop the subscript on the output value of the j^{th} unit, o_j: o refers to the output value for any particular unit.)

We can further complicate the model by giving our input layer not 1 input but 3: one from each of three new units:



Given the earlier analysis, the output from layer A, o_A, will be the result of the weighting matrix $\mathbf{W_A}$ times a 3-dimensional input vector. That is, $o_A = F_A((\mathbf{W_A})i)$. In turn, that *output* from the 5-element layer, o_A is identically i_B, the *input* to the 4 top units of layer B. The final output (from the 4-element layer), o_B, is $F_B((\mathbf{W_B})i_B) = F_B((\mathbf{W_B})o_A)$. Therefore,

$$o_B = F_B((\mathbf{W_B})o_A) = F_B((\mathbf{W_B})(F_A((\mathbf{W_A}) i)))$$

Notice that if the firing function that determines the response of the neuron were simply the identity function (i.e., $F_A(u) = F_B(u) = u$), one could replace the two matrices with one, and

$$o_B = (\mathbf{W_B})((\mathbf{W_A}) i) = ((\mathbf{W_B})(\mathbf{W_A}))i$$

As a result, the three-layer network could be replaced by a two-layer version that uses the new matrix, $((\mathbf{W_B})(\mathbf{W_A}))$, as its weighting matrix. The same is true whenever the firing function is *linear*, that is, the output from the function is simply a scalar multiple of the input plus some constant ($f(x) = ax + b$). Thus the fact that

the firing function used in neural networks is *non-linear* is extremely important.[4] This type of firing function keeps the weighting matrices for discrete layers from collapsing in on one another and significantly changes the possibilities for the behavior of the network. Indeed, the XOR problem discussed above cannot be solved without a hidden layer using a non-linear firing function.

What can neural nets do?

None of this seems very promising so far: we have the ability to mathematically describe what layers of artificial neurons do. They map input vectors to output vectors through a system of connection weightings describable as a matrix. What is the pay-off? The pay-off is that neural nets can be trained to recognize patterns. There are three main types of neural nets that can recognize patterns of increasing complexity: a simple "unsupervised" feedforward network, a "supervised" feedforward network trained by back propagation of error correction, and a recurrent network where synaptic connections run in both directions. That is, "higher" units also feed their output back to lower levels. Beyond these three fixed typologies, however, there are networks where the number of nodes and the plasticity ("teachability") of the nodes change over time. These latter, time-dependent designs prove extremely important in current models for cortical network behavior.

---

[4] Although various algorithms have been used in modeling, the simplest non-linear function is a *step function*: when the inner product of the weighted sum of the inputs for a unit is less than 1, the output is 0; if it is greater than or equal to 1, the output is 1.

Unsupervised Learning

Even the behavior of the simple version of a neural network is quite remarkable. If one connects input neurons to output neurons and allows the output neurons to react to a set of input patterns following a simple rule, namely

$$\Delta w_{ij} = \eta a_i b_j$$

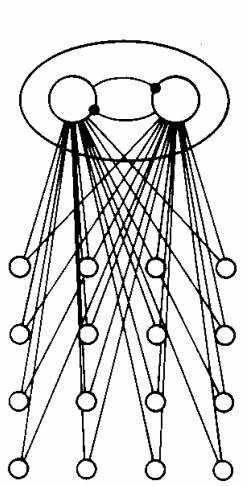
w_{ij} = the weight of the connection between the ith input and the jth output.

a_i = the activation value of the ith input

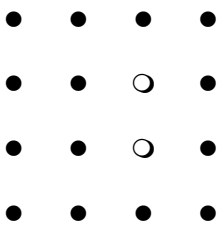
b_j = the activation value of the jth output

they will gradually discover a set of significant features that allow them to discriminate between the input patterns. The famous Hebbian learning rule says that if the output neuron reaches the firing threshold, increase the weight of all the connections that contributed to that firing. This type of "feature extraction" is called *unsupervised learning*. That is, no one—not the programmer, not the higher order units—is telling the units what the right answer (fire/don't fire) should be. In most cases, the system is actually a bit more complicated. First, total activation in the system is controlled either by making the increase in the weights relative (i.e. the sum of all the weight values remains fixed) or by adding a decay factor to the weighting values. Secondly, the output units usually are connected to one another to inhibit each other. That is, if one unit fires, it prevents all the other units from firing. This sort of winner-take-all rule is called *competitive learning*.

A simple example of this type of system is an analysis of "dipoles," input patterns where only two units are active and all others are silent.[5] For instance, a grid of sixteen units sends data (0 or 1) to two mutually inhibiting units:



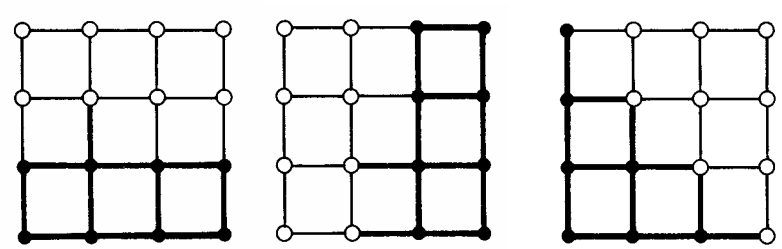
The input patterns have one restriction: at one time, only two units can be active, *and* they must be either vertically or horizontally adjacent. For example:



When the two final units stabilize their weight matrices, it turns out that they divide the 16-dot space in half. The way they divide the space (horizontal, vertical, or diagonal) depends on the initial values of the weights as well as the order of presentation of the sample patterns and varies from experiment to experiment.

---

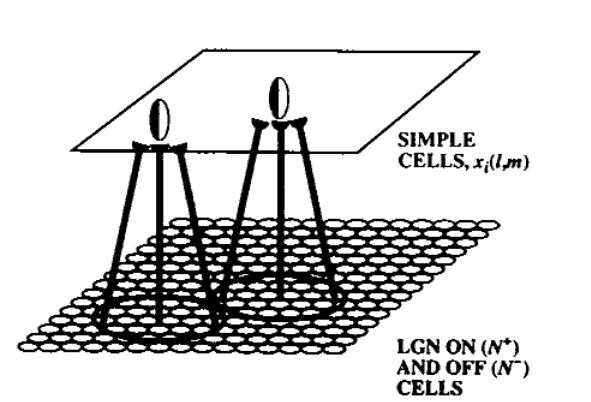
[5] This example comes from D. E. Rumelhart and D. Zipser, "Feature Discovery by Competitive Learning," in David E. Rumelhart and James L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations* (Cambridge, MA: MIT Press, 1986), pp. 170-177.



TRIAL 1 TRIAL 2 TRIAL 3

In the above diagrams, a filled circle means that Output Unit 1 gave the input from that unit a higher weight, and an empty circle means that Output Unit 2 gave the input from that unit a higher weight. A heavy line between units means that when the two input units were active, Output Unit 1 won the competition for activation (and inhibited Unit 2); a thin line means that Output unit 2 won the competition. Since the output units organize their responses through mutual inhibition, they must find *some* feature to divide the input domain, and so they discover a topographic map. If one uses *four* units in the final layer, they still divide the space but in a more complex manner. This behavior may not seem very remarkable, but it probably underlies many patterns of neural response in the brain. For example, Layer IV in the primary visual cortex (V1) has many neurons that respond to one eye but not the other, and they send their output to other layers for higher order processing that requires this selective response. Ocularity, as this feature is called, probably grows out of a simple, mutually inhibitory competitive learning rather than out of a distinctive specialization in the design of the neurons. A model for the connectivity between the lateral geniculate nucleus (LGN) in the thalamus and V1

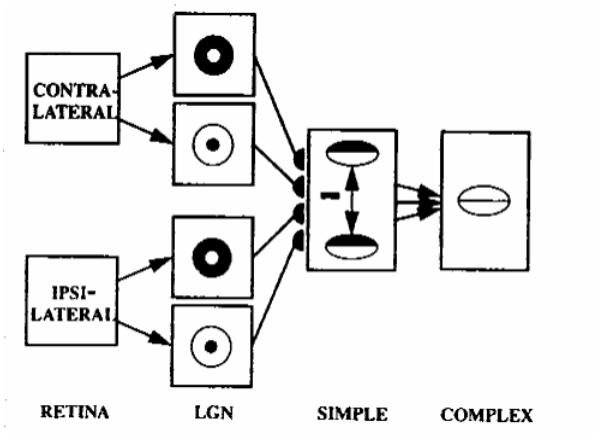
that can produce this result is more sophisticated in its details but does not look fundamentally different from the simple dipole configuration:[6]



Activation from the ON-center and OFF-center cells in the retina produce patterns of corresponding ON and OFF cells in the lateral geniculate nucleus. Neurons within a particular region of the LGN then contribute activation to neurons within a region of the primary visual cortex. The neurons in V1 are mutually inhibitory. What evolves are patterns of neurons (simple cells) that respond more strongly both to one eye than to the other and to patterns at particular angles. These neurons then pass their activation on to neurons with more complex activation patterns (complex cells):

---

[6] This model comes from Steven J. Olson and Stephen Grossberg, "A neural network model for the development of simple and complex receptive fields within cortical maps of orientation and ocular dominance," *Neural Networks* 11 (1998):189-208. For my discussion here, I rely on their analysis, which perhaps can represent the current state of the art of mathematical modeling (an art that is progressing extremely rapidly).



The next chapter will return to a discussion of the visual system: what matters for the moment is that relatively simple models of unsupervised learning relying on mutual inhibition can produce important properties of biological neural systems.

Unsupervised learning serves to divide patterns of input into mutually differentiating groups, but there are many types of problems that it is poor at solving, as Papert and Minsky pointed out. Overcoming those limits requires more complicated networks.

SUPERVISED FEED FORWARD SYSTEMS

The discovery of ways to train the hidden units in feedforward networks was the breakthrough that revived interest in neural nets. A simple example is letter recognition: suppose we had an 8×8 grid to read letters. The 64 input units fire if more than 40% of their area is black. They are connected to 8 hidden units that are in turn connected to eight output units.

In each cycle of training, the network compares the activation of the output units with a *target*, the eight-bit number that is the ASCII value of the input letter. The comparison generates an error analysis that is fed back to the hidden units, which then alter the connection strength they assign to the activation of the input units. This modifying of weights based on comparison with an expected answer is why the process is called *supervised learning*. By the end of training, the output value should match the target, the ASCII value of the scanned input letter. The strategy is to attempt to minimize the error that each unit contributes to the total number of mistakes in the system. Since changes in the weights are small, the system uses what is called a gradient descent: if a small change in one weight, holding all the others still, decreases the error, the change is made. Slowly but surely the system heads to the lowest-error solution.

After running through the entire alphabet several thousand times, the system stabilizes to a set of weightings that gives the correct results. The final letter reader is quite robust: if an "a" is poorly written or damaged, or an input unit fails, the output units still can come up with the right answer. If we look inside the output units as we degrade the quality of the input, however, we can see that the weighted sum of the outputs from the previous layer (the *net* value) begins to suggest other possibilities ever more strongly. Since the firing function is differentiable, we can also see fractional output values where the final units are giving their "best guess."[7]

---

[7] The three books that provide an excellent overview of the emerging paradigms of connectionist neuroscience discuss supervised learning models in detail. My account is brief because the more recent modeling in neuroscience seems to stress recurrent

There are two basic problems with supervised learning that make it useful in computer science and in business but not very promising as a model for cortical learning strategies. The first is procedural. Although feedback connections exist at many levels of cortical and subcortical structure, the sorts of calculations needed to make the back-propagation of error work are exceedingly unrealistic for a biological system. The second problem is the very concept of supervision: the cortical systems in general do not have access to "correct" output patterns. The "meaning" built into the system essentially rests on an external "deus ex machina." For this strategy to work in a biological context, DNA would need to determine internal structures in an astonishing detail that is not very neurologically probable.

Time-Dependent Networks

Experience with neural network modeling is increasing rapidly. It is perhaps important to note here that research in network design has at least three distinct goals. The first is simply to see what properties emerge from network architectures as a form of basic applied mathematical research in complex dynamic systems, with no concern for immediate applicability. The second goal is to explore architectures to provide solutions for specialized problem domains: neural networks based on the back propagation of error have proved to be very powerful pattern detectors in

---

architectures that do not rely on pre-given targets and on the back-propagation of error. For detailed discussion of supervised learning, see "Computational Overview" (Chapter 3) in Patricia Churchland and Terrance J. Sejnowski, *The Computational Brain* (Cambridge, MA: MIT Press, 1992), Paul Churchland, *The Engine of Reason, the Seat of the Soul: A Philosophical Journey into the Brain* (Cambridge, MA: MIT Press, 1995), and Mark H.

financial markets, signal processing, voice recognition, and the like. They also hold promise for robotics and artificial intelligence. Whether these algorithms have any *biological* plausibility does not matter: they get the job done. In contrast, the third goal—important for this book—is the effort to use mathematics to model the synaptic organization of the brain.

More sophisticated architectures, learning rules, and mathematical tools have allowed researchers to test the behavior of networks that increasingly incorporate features of biological systems and have created a powerful synergy between neuroscientists, mathematicians, and computer scientists. The emergent behavior of the increasingly complex systems can be tested against the actual anatomical and physiological data and helps clarify patterns within neuronal systems.[8]

It turns out that in this world of biological modeling, sophisticated variations on basic Hebbian (unsupervised) learning rules can go a long way. For example, even a fairly simple array that incorporates a random distribution of sparsely connected excitory neurons, inhibitory interneurons, and a "trophic factor" (to capture aspects of biological constraints on plasticity) will produce neurons that respond to "A XOR B" when given a random set of input patterns from an A unit (0 or 1) and a B unit (0 or 1). The system as a whole settles into feature detectors

---

Johnson, *Developmental Cognitive Neuroscience: an Introduction* (Oxford and Cambridge, MA: Blackwell, 1997).

[8] For a good example of the introduction of biological parameters, see Joshua Brown, Daniel Bullock, and Stephen Grossberg, "How the Basal Ganglia Use Parallel Excitatory

that respond to various combinations of A and B input. The largest group of units settle into simply FALSE (A=0 and B=0). The next largest groups settle into unitary functions: (A=0), or (A=1), or (B=0), or (B=1). Next, however, are the combinations: (A=1 and B=1), (A=1 or B=1), (A=0 and B=1), etc. , and the combinations of combinations like the XOR function (i.e. (A=1 and B=0) or (A=0 and B=1)).

If the model is changed to introduce a wave of plasticity by slowly shifting the peak concentration of trophic factor from left to right (i.e., at first the left hand side has the largest scaling factor $\eta(j,t)$ in the Hebbian weighting update rule $\Delta w_{ij} = \eta(j,t)a_i b_j$, while those to the far right do not change at all, but then the values drift rightward.), the system shifts the simple feature detectors to the left (i.e. they change and stabilize early), and the complex feature detectors shift to the right (i.e. they change and stabilize late).[9] Since patterns of neuronal plasticity created by sequential arborization and pruning of synapses appear throughout the cortex—with perhaps the longest maturation rate in the frontal cortex—the results from the modeling experiment are important. Most multi-layer neural network models simply assume the existence of the layers based on adult architecture. The modeling of a wave of plasticity demonstrates—at least as a possibility to be explored—that the developmental dynamics of synaptic growth and death can

---

and Inhibitory Learning Pathways to Selectively Respond to Unexpected Reward Cues," *Journal of Neuroscience* 19.23 (December 1, 1999):10502-10511.

[9] Jeff Shrager and Mark H. Johnson, "Dynamic Plasticity Influences the Emergence of Function in a Simple Cortical Array," *Neural Networks* 9.7 (1996):1119-1129.

indeed produce the required layering that moves from simpler to more complex, aggregated feature detectors.

Catastrophic Interference

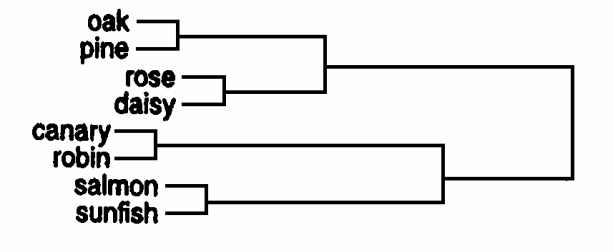
There is a very different issue of temporal sequencing in neural networks that presents a fundamental problem with great biological significance. A neural network, once trained, has devised a strategy of responses embodied in the connection strengths that defines what it sees as the implicit dimensionality of the input domain. In the dipole problem, the final output has only two possibilities: when one unit is on, the other must be off. The system accordingly finds a way to divide the domain of all possible input combinations into two halves. The two output units could send their activation to one higher-order unit that would activate only if the left output unit were active: this single binary unit (with value 0 or 1) could adequately represent the final state of a system with two possible states. The letter reading system is more complicated: it needs to devise a strategy to make 52 discriminations. This requires six output units to represent the approximately 2^6 (64) possible states. Suppose, however, that the letter reading system initially is trained on only one half of the letters. Only five output units would be needed, and one would discover that when the system stabilized, the activation of one of the six final units reflected some linear combination of the responses of the other five. If the system then is trained on the complete alphabet, the recognition process would suffer from what is called *catastrophic interference*. As the system tried to learn the new letters, the new patterns would interfere with

the previous training. The system's ability to recognize the first set of letters would catastrophically decline. Its old approach to dividing up the input space would not work anymore, and it would need to reorganize itself.

Catastrophic interference appears even under less extreme circumstances. James McClelland trained a supervised network on a set of patterns describing living things using sentences like:

> Robin can grow, move, fly.
> Oak can grow.
> Salmon has scales, gills, skin.
> Robin has wings, feathers, skin.
> Oak has bark, branches, leaves, roots.

On looking at how the network divided the object space, McClelland and his associates discovered that the network developed the following taxonomy:[10]



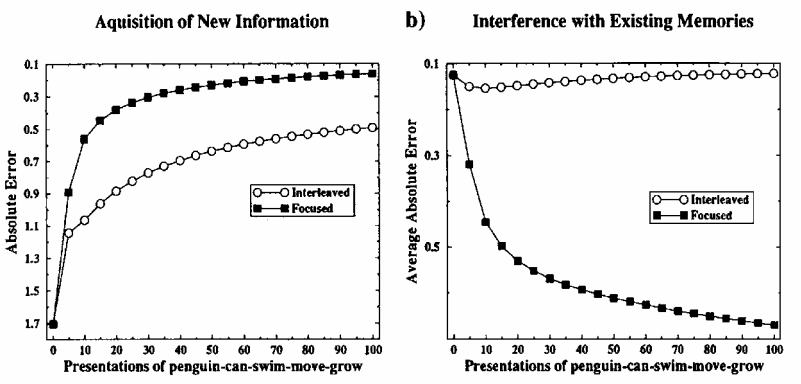
Then they taught the system about penguins:

> Penguin can swim, move, grow.
> Penguin has wings, feathers, skin.

---

[10] The length of the lines indicate the relative closeness of the categories. Thus the rose/daisy inclusive group is closer to the pine/oak than the canary/robin is to the salmon/sunfish. James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly, "Why There Are Complementary Learning Systems in the Hippocampus and Neocortex: Insights from the Successes and Failures of Connectionist Models of Learning and Memory," *Psychological Review* 102.3 (1995): 419-57.

Penguins, needless to say, violated the rules of the world as the network knew them. When the stabilized system was trained over and over with just the penguin information, catastrophic interference arose. However, McClelland's team also discovered that such interference could be greatly restricted if the new training set contained both the penguin information and the old data:[11]



In certain domains, like the primary sensory cortices, the neurons receive input that spans the complete set of possible inputs from the very beginning. That is, a baby encounters a world of shapes, colors, and movements that reasonably represent the combinations in the natural world. Strategies developed by the thalamus and the early visual cortex to extract features in the environment (line segments, surfaces, edges, colors, etc.) will work throughout the rest of life because the new data will be part of the same domain of fundamental structures even if the particular combinations constantly change. Other higher-order features are not so universal. The common example is the phonological features of a particular natural language. Babies learn to respond to the phonological distinctions of their parents'

[11] McClelland, et al., "Complementary Learning Systems," p. 435

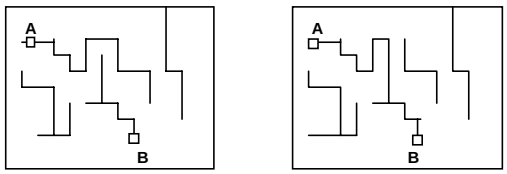
language—and cease to respond to distinctions *not* in that language—at a very young age. Even here, however, learning a new set of features would not produce too many errors because the overlap between phonological systems would still be much greater than the differences. Higher order categorical systems provide a far greater experiential and theoretical challenge. Humans go through many different stages of learning about the world, and we learn about only little pieces of it at a time. One major challenge for neural network modelers has been to propose ways in which we can learn French without forgetting English or learn geometry without forgetting algebra. Simple Hebbian networks will need to become much more elaborate before they keep from crashing. In fact, however, in recent years the network designs *have* become much more elaborate, and the dominant models rely on recurrent connection patterns.[12]

Recurrent Connections and Feed-back Networks

Feed-forward systems, whether biologically plausible or more general, are poor at handling many important classes of patterns. First, feed-forward systems are essentially instantaneous: all the neurons in a layer send their data at one time. Patterns that unfold over time are hard to capture. So are patterns that are best explored in a serial rather than parallel manner, like determining connectedness. Consider, for example, the two sets of lines and end-points A and B below.

---

[12] Chapter Three returns to this topic in discussing a proposed role of the hippocampus in the forming of memories.



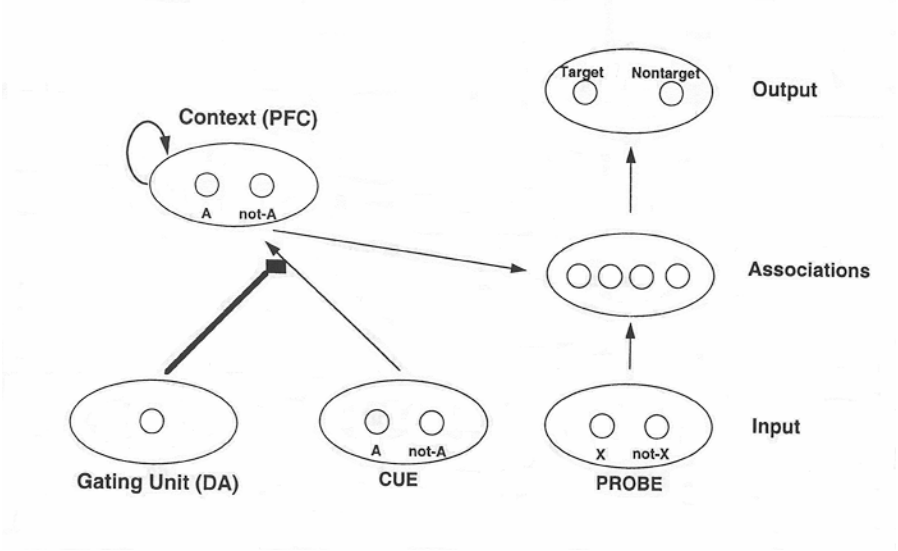
Seeing whether A and B are connected cannot be done efficiently by analyzing all elements in parallel. The better approach is where higher and lower units are allowed to pass activation information back and forth. This solution is a *recurrent* network. [13] In a recurrent network, higher-level units have feed-back connections that provide input to lower-level processing layers. The term *recurrent connection*, however, refers specifically to a unit whose output activation loops back to serve as part of its own input: it turns out that many neural systems may rely on the properties of units with recurrent connections within recurrent networks.

One simple version of a network with recurrent connections looks like the diagram below.[14] The network is an effort to model the role of prefrontal "working memory" in a delayed judgment exercise. In the experiment, a person is shown a series of cue cards ("A" or "B") and target cards ("X" or "Y") and is supposed to indicate when an "X" follows after an "A."

---

[13] The patterns are in Peter R. Roelfsema and Wolf Singer, "Detecting Connectedness" *Cerebral Cortex* July/August 1998, V 8 N5:385-396.

[14] Todd S. Braver, Deanna M. Barch, and Jonathan D. Cohen, "Cognition and Control in Schizophrenia: A Computational Model of Dopamine and Prefrontal Function," *Biological Psychiatry* 46 (1999):320.



The neural network model uses a set of "context" units with recurrent connections (i.e. output from these units loops back to serve as part of the units' input) to store information about the previous card. In this case, what matters (and must be stored) is simply whether the previous card was an "A." Because the connection weight for the recurrent connection is very high, when one of these context units is activated, its own output will continue to keep itself active until some new cue data comes to change its state. This persisting "context" information then becomes available for use by the "Associations" units that combine prior cue and current target information to assess a response. In this particular model, since the experimenters wanted only one combination to activate the "target" output unit, they used back-propagation of error (supervised learning) to train the connection weights in the system.

Neural networks with layers of units with recurrent connections have important properties that extend well beyond their role in capturing time-dependent patterns. Recurrently connected competitive networks, for example,

evolve the sorts of "attractor basins" that systems using back-propagation of error create. That is, networks define a differentiated input-space in which the recurrent connections cause the pattern of activation produced by the input values in any part of a particular region (an attractor basin) of that space to shift to a steady state after several iterations. The steady states of activation into which the layer can fall are the attractors that define the network's analysis of the implicit dimensionality of the input-space. Attractor networks are important because they are very tolerant of noisy input and—as a related effect—they are very good at pattern completion. Given data from only a small set of input units, that is, activation spreads across the layer to arrive at the best-guess attractor state.

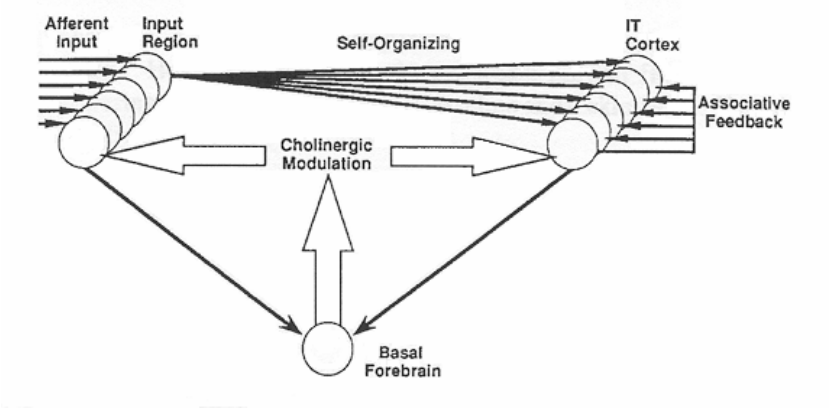
One problem with recurrently connected units is that at times their activity should depend on the internal state of the layer, and at times their state needs to be updated.[15] Moreover, it should be possible for new data to force the redefinition of the system of attractor basins. Here researchers have followed the lead of biological systems and introduced gating connections, ways of changing the way the units process their input data. The "Gating Unit" in the delayed judgment system above provides a way to model the role of dopamine (DA) on working memory in the prefrontal cortex. Under ordinary circumstances, *tonic* DA activity is believed to aid the maintenance of patterns in working memory while *phasic* activation helps shift patterns. The particular system above was designed to test problems with instabilities in *phasic* activity. Too little dopamine, for example, prevents the

---

[15] For a discussion of this issue, see Braver, et al., p. 316.

context unit from being reset, so it is locked into whatever state it is in when the gating activation disappears.

Another system, developed by Vikaas Sohal and Michael Hasselmo, looks at a similar problem of controlling the updating of a layer of recurrent units. They model the behavior of neurons in a higher region of visual processing—the inferotemporal (IT) cortex—when connected to gating feedback from the cholinergic activation of the basal forebrain.



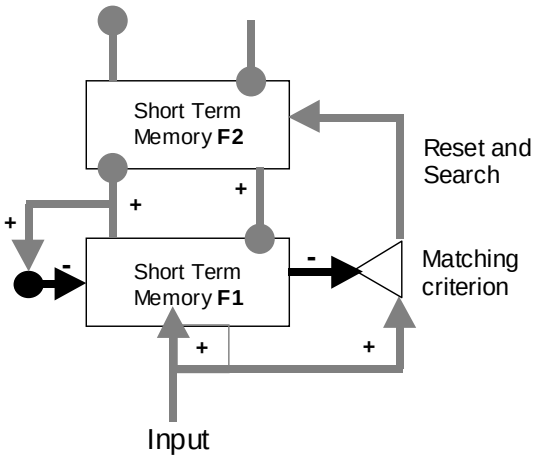
The question they explore is how feedforward connections in the brain can continue to organize themselves in defining an input space without undue influence from the attractors that already exist. How can a system know when to reorganize because the input is truly new as opposed to merely settling down into an attractor state on the assumption that the data is just noisy? Sohal and Hasselmo elegantly propose that the mediation of gating by acetylcholine from the basal forebrain offers at least one solution. The neurotransmitter acetylcholine shuts down recurrent connections but allows feed-forward activation. In their model, input patterns that already are part of the response repertoire of the system activate only a few neurons in the total population, and the strongly active neurons in turn

inhibit the others. Unknown patterns, in contrast, present a free-for-all as the neurons compete among themselves in forming a response. This broad activation is then sufficient to activate the cholinergic neurons of the basal forebrain that prevent the recurrent connections in the IT cortex from pushing the activation toward a pre-set attractor state. The IT feedforward system then learns the new pattern through competitive Hebbian procedures. As the competitive inhibition has more clearly defined winners and losers, total activation within the layer drops, and the cholinergic activation drops correspondingly, while the recurrent connections become increasingly strong and help stabilize the new attractor configuration.

Feedback systems of many sorts—from tightly coupled arrays of recurrent units to more diffuse long-distance feedback connections—have attracted intense inquiry because they are ubiquitous throughout the brain. For example, the lateral geniculate nucleus (LGN) in the thalamus is the first processing stage for information coming from the retinas en route to the visual cortex. However, there are about ten times as many neurons coming *from the cortex to the LGN* as there are neurons leading *to* the visual cortex. Reciprocal connections of this sort between higher and lower processing stages appear throughout the cortex. The question, of course, is why.

One of the earliest proposals for the function of recurrency in the cortex was Stephen Grossberg's Adaptive Resonance Theory (ART), which has proven very influential as a general way of thinking about the role of feedback in cortical processes and as a technique for mathematical modeling in such systems as the IT-basal-forebrain model discussed above. The design of an adaptive resonance model

is as follows: the first layer of feature detectors extract a set of features **F1** out of the input data that it stores as a pattern of excitation that it sends as a hypothesis to the second layer. This input from the first layer creates a pattern of excitation **F2** that is both sent to the next layer beyond and also creates two types of data to be looped back to the first layer. When the second layer fires, it creates both a *general* low level of *inhibition* as well as a specific pattern of excitation. Through this reciprocal connection, the second layer increases the activity of first layer features associated with its own best guess based on the initial **F1** and inhibits the activation of all other units. At this point, either the layers head toward agreement (an attractor), or, if the mismatch between the two layers is too great, the second layer is reset and the weights of its feature extracting matrix are adjusted to accommodate the new data.[16]



[16] This account and diagram come from Stephen Grossberg, "The Link between Brain Learning Attention, and Consciousness," *Consciousness and Cognition* 8 (1999):1-44. The arrows ending in triangles have fixed connection values, but those with circles can be adjusted.

Two points are especially worth noting. First, unlike supervised learning systems that rely on back-propagation of error, the ART model uses local Hebbian learning algorithms. Secondly, systems based on ART-like designs actually do evolve into functioning, stable networks: they prove adequate to train hidden units, the daunting task where simple feedforward perceptrons failed.

Not only do ART-like models utilize the feedback connections observed in neuroanatomy, but they also specifically help explain the role of the recurrent circuits from higher to lower levels in the training of lower level response patterns.[17] Also, adaptive resonance models offer an account of learning that largely avoids the problem of catastrophic interference that plagues artificial neural networks. Adaptive resonance models are less vulnerable to this problem because the neurons are operating at "two different time scales: a fast dynamic state estimation process that [for example] allows the visual system to anticipate incoming stimuli and a slower Hebbian synaptic weight change mechanism."[18] From the high-level interplay between perception, memory, and evaluation to the attentional modulation of low-level sensory processing, the top-down/bottom-up interactions modeled in ART-like systems are central to cortical functioning. As

---

[17] See, for example, David J. Krup, Asif A. Ghazanfar, and Miguel A. L. Nicolelis, "Immediate thalamic sensory plasticity depends on corticothalamic feedback," *PNAS 96* (July 1999):8200-8205. The report interestingly proposes a tonic inhibition generated by the corticofugal connections that is countered by more specific excitory feedback circuits that correspond to the combination of inhibition and excitation of the ART model (p. 8204).

[18] Rajesh P. N. Rao and Dana H. Ballard, "Dynamic Model of Visual Recognition Predicts Neural Response Properties in Visual Cortex," *Neural Computation 9* (1997):753. Their

experience with the mathematics of recurrent systems grows, researchers have grown ever better at modeling the interactions at all the different levels. Edmund Rolls and colleagues, for example, have used the slow-decaying, self-sustaining activation of recurrent connections to provide a means for developing view-invariant representations of objects in high-level visual processing structures. That is, a horse seen from the front or rear, for example, looks very different from a horse seen from the side. How the brain manages to bind all these views to one object is no small feat. Rolls proposes that the IT cortex learns to associate naturally occurring incrementally shifted views with one another and can do so because the former view persists in recurrent circuits long enough for the association to be learned.[19]

The modeling of systems like object recognition makes very clear a key characteristic of neural networks in general: the distinction between treating object representations as components of memory or as part of sensory processing within the mathematical model essentially collapses. As part of top-down feedback, they are memory, while as a stage in bottom-up analysis, they are part of the visual system.[20] Yet both roles involving the same neurons are usually required for the

---

model differs from Grossberg's by introducing *intralayer* recurrent "state" neurons in addition to the interlayer recurrent connections.

[19] See the discussion in Peter McLeod, Kim Plunkett and Edmund T. Rolls, *Introduction to Connectionist Modelling of Cognitive Processes* (Oxford: Oxford University Press, 1998), 292-302.

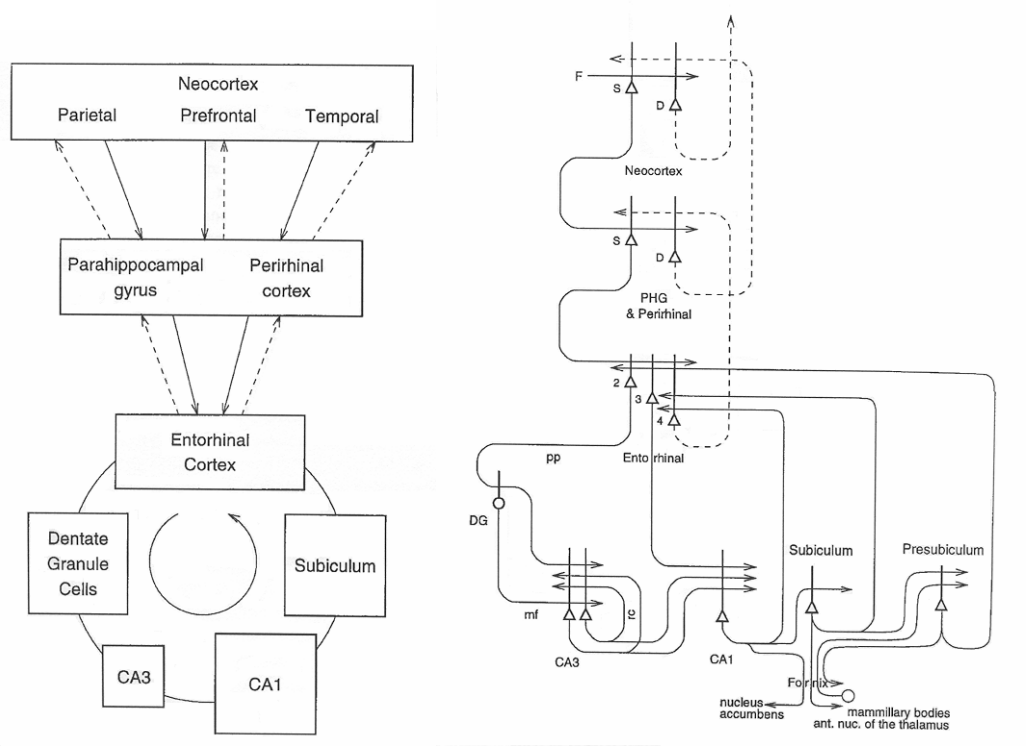
[20] Sohie Lee Moody, Steven P. Wise, Giuseppe di Pellegrino, and David Zipser in their very good article, "A Model That Accounts for Acitivity in Primate Frontal Cortex during a Delayed Matching-to-Sample Task" (*Journal of Neuroscience*, January 1, 1998, 18(1):399-410), note that although their initial design assigned two units to be storage units and six

system to function. Indeed Rolls also has been developing approaches to the hippocampal system that structures these complex interactions between long-term memory and perception. The hippocampus, a brain structure important in memory formation (and perhaps in recall), will be discussed in greater detail in the next chapter. But recent models that incorporate the patterns of recurrent and feedback connections found in the hippocampal system are an important advance. The mathematical models contribute significantly to understanding the implications of these types of connections. In exploring the role of the hippocampus in the fast encoding of episodic memory and in its recall, Rolls focuses in particular on modeling the behavior of sparse connections from the dentate gyrus to the region CA3 in combination with the dense recurrent connections (**rc** in the diagram) within CA3 itself both.[21] Extending the model, he ties this internal organization of the hippocampus into the larger recurrent memory systems of which it is a part. He develops a model at two levels of detail.

---

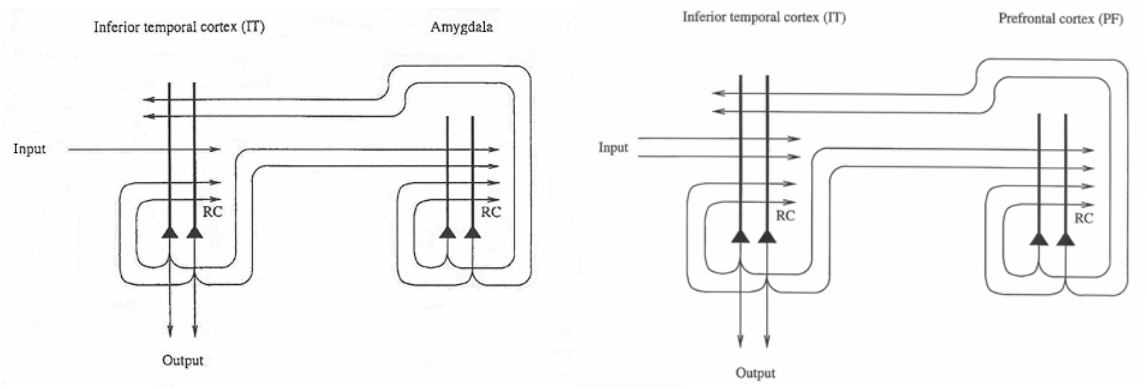
for processing the comparison between present and past input, the actual functioning of the units in the resulting network defied such a simple division of labor. The distribution of memory and processing throughout the system grew complex and many units served multiple roles.

[21] Edmund T. Rolls, "Hippocampo-Cortical and Cortico-Cortical Backprojections," *Hippocampus* 10 (2000):380-388.



In the first diagram, the parietal cortex participates in the so-called "where" stream of visual processing while the temporal cortex is part of the "what" stream. The prefrontal cortex is the site of working memory that contributes to high-order evaluation of sensory information and its integration with memory and planning. These cortical systems feed into the perirhinal cortex that may be crucial for the long-term encoding of episodic memory as coherent associations of objects and sequences of actions. The hippocampus, through the mediation of the entorhinal cortex, mediates the formation, preservation, and recall of these multimodal high-level associations. Note that all the cortical connections are bidirectional. The second diagram is a simplified model of the connections in the hippocampal system suggested by the block structure of the first diagram. The small triangles are pyramidal neurons: the lines coming out the top represent dendrites and lines

crossing those dendrites provide input. The line coming out of the base of the triangle is the axon carrying output activation. The dotted lines in both diagrams represent feedback connections. The diagram of connections provides a schematic for mathematical modeling of the over-all system. However, the behavior of many layers of reciprocally connected recurrent systems is extremely hard to analyze, so Rolls and his colleagues have been looking at simpler models of *pairs* of reciprocally connected systems to begin to explore the properties of their interactions. For example, they have worked on connections between the IT cortex and prefrontal cortex to model the dynamics of working memory on the biasing of perception, and they have modeled connections between the IT cortex and the amygdala to examine the impact of emotional state (mood) on perception.[22]



It is clear from the diagrams that Rolls treats both systems as relying on the same principles despite the quite different cortical functions they serve. First, there are recurrent connections that create attractor basins, facilitate pattern completion, and

[22] For the model of the amygdala-IT interaction, see E.T. Rolls and S. M. Stringer, "A model of the interaction between mood and memory" in *Network: Computation in Neural Systems* 12(2001):89-109. The IT-PFC model comes from E. T. Rolls, "Hippocampo-Cortical and Cortico-Cortical Backprojections," p. 386.

allow self-sustaining activation. Secondly, mutual feedback between the modules make Grossberg's ART-like resonant states possible. The IT cortex will respond more strongly to partial views of "objects" that correspond, on the one hand, to current mood (the amygdala model) and, on the other, to recently seen objects (the PFC model.) Rolls stresses, however, that (without adding the complication of gating mechanisms), the strength of the internal recurrent connections must be much higher than the strength of the feedback connections between modules for the two modules to not swamp one another.

A final, more speculative aspect of recurrent models that is currently the subject of intense debate is their role in conscious experience. When we are conscious, what are we conscious of? What exactly is the visual imagery we "see?" Where in the brain—at what level of abstraction—is the detailed sensory information registered for synthesis by consciousness, and how is it accessed? The same can be asked for other modalities. When we hear, what are we hearing? How aggregated is the data to which conscious processes have access? Why and how, for example, does the conscious auditory system backtrack in real time to reinterpret earlier sounds based on later ones?[23] Theorists have noted that there is no distinction between storage and processing units in connectionist models. The greenness of "green" is not stored in some address that can be passed to upstream systems. Instead, in resonance models, all layers of the perception system are still

---

[23] This is a key example that Stephen Grossberg gives to illustrate the synthesized quality of conscious experience in "The Link between Brain Learning, Attention, and Consciousness," *Consciousness and Cognition* 8 (1999):1-44.

active and talking to one another. The responses of the V4 region of the visual cortex that determine the "red" of the rose precisely innervate the *red of the rose* of conscious experience.[24]

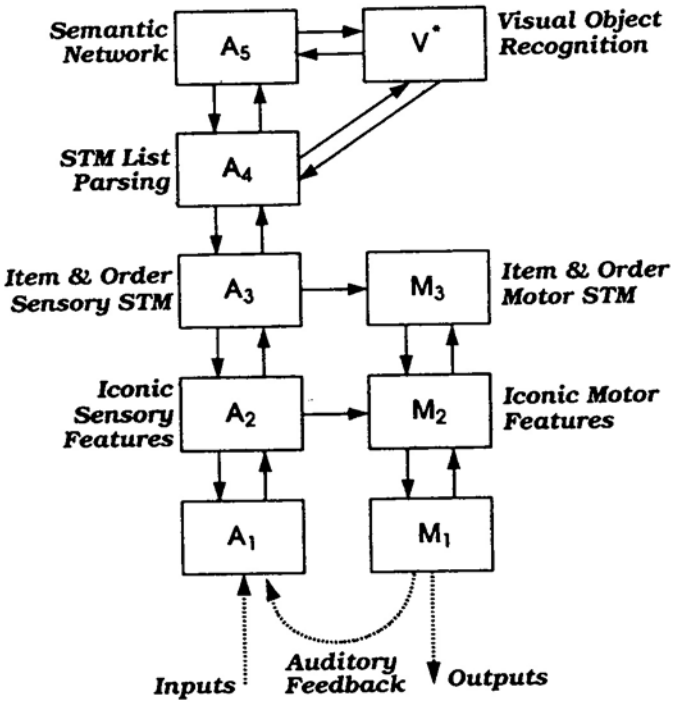
Grossberg suggests that only those systems subserved by adaptive resonant circuitry can become conscious.[25] He furthermore stresses that the sensory systems rely on a hierarchy of reciprocally connected (i.e. potentially resonant) levels of object processing to abstract ever higher order features from the raw sensory data. In modeling the auditory system, for example, Grossberg proposes such a hierarchical model to account for the requirements of sentence parsing and also includes multimodal and motor system connections to link parsing to speech production.[26] The model in particular attempts to account for ways in which conscious speech perception remarkably uses *later* input to fill in gaps introduced into the auditory stream. That is, we think we hear in the auditory present, but in fact there is enough of a time delay in the generation of "conscious" hearing for the auditory system to use later data to make sense of "current" speech. Such completion requires seamless top-down changes in basic sensory awareness.

---

[24] Zeki and Bartels in particular stress the decentralized character of visual consciousness and the collapse of the distinction between processing and perceptual systems. J. Zeki and A. Bartels, "Toward a Theory of Visual Consciousness," *Consciousness and Cognition* 8 (1999):225-259.

[25] Stephen Grossberg, "The Link between Brain Learning, Attention, and Consciousness."

[26] Athanassios Protopapas, "Connectionist Modeling of Speech Perception," *Psychological Bulletin* 125.4 (1999):427. The figure is adapted from S. Grossberg, "Competitive Learning From Interactive Activation to Adaptive Resonance," *Cognitive Science* 11 (1987):52. In the chart, STM stands for "short term memory."

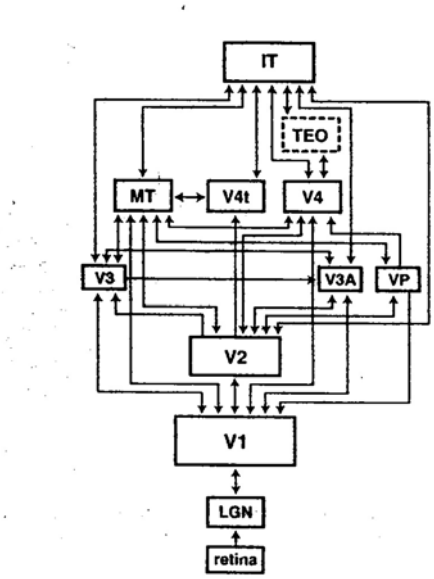


Daniel Pollen, in a proposal consistent with Grossberg's model, argues that even "early" visual sensory regions participate in conscious visual experience.[27] First, Pollen charts the bidirectional interactions *within* the visual system, with the inferotemporal (IT) cortical areas as the primary gateway to higher "object" processing.[28] (Here the abbreviations are: TEO = temporal occipital area, MT = medio-temporal, and LGN = lateral geniculate nucleus of the thalamus.) Not all of the feedback connections from IT and TEO are shown.

---

[27] Daniel A. Pollen, "On the Neural Correlates of Visual Perception," *Cerebral Cortex* 9 (Jan./Feb. 1999):4-19.

[28] Daniel A. Pollen, "On the Neural Correlates of Visual Perception," based on a diagram in Zilles and Clark, "Architecture, connectivity and transmitter receptors in human extrastriate visual cortex. Comparison with non-human primates," *Cerebral Cortex* 12 (1998).



The basic point to stress here is that contemporary modeling follows the structure of the cortex in proposing that our cognitive systems work through the interaction of layer after layer of recurrent neural networks, all of them seemingly trainable through Hebbian associative learning processes. The picture of course is in fact far more complex than these initial models suggest. Many more components need to be integrated: prefrontal cortex for planning, the anterior cingulate cortex for attention, the hippocampus for learning, not to mention the motor cortex and cerebellum. Moreover, the actual mechanisms for long-term learning—through synaptic arborization and pruning (i.e. the growth and removal of "branches" on the neuron's axon), changes in protein structures at synapses, changes in receptor density and efficiency, and the control of gene expression—will prove more complex than simple Hebbian models. However, the current approximations achieved through modeling suggest that we are, in general, on the right track.

Distributed Representation

Suppose a system modeled on the human visual cortex is trained to read the printed words "cat" and "mouse" correctly. In such a system, where exactly are these terms in the network? There was no single unit in the letter scanner above that identified c, a, or t. Suppose we add a neural network to process semantic structures activated by the words read by the scanner. In such a network, no single unit represents "cat" as "–human, +predator, +furry,..." In both cases, the output derived from multiplying a weighting matrix stored in the network by an input vector. Multiply the same matrices by different input vectors, m-o-u-s-e or "mouse", and you get different but still coherent results. For each input pattern, the representation (the information that generates the trained output vector) is *distributed* throughout the weighting matrix. During the training period, the network learns a system to mutually differentiate the input vectors within the set: the fact that the network deals with a *set* of data is important. Any new items would "mean" within the logic of differentiation the network acquired in learning the initial set of patterns.

This model of arrays of neurons trained to detect patterns through adjusting the connection weights between units in the array is the basic paradigm of *connectionist* approaches both to artificial intelligence and to the brain itself. The approach is both very powerful and very unnerving.

The approach is unnerving on many levels. In the humanities, for example, theorists from Saussure to Quine to various versions of post-modernism have reflected on how and what words mean and have argued that they derive their

meaning from their place in a larger system of meaning. Words are not so much atoms as positions in a structure. This metaphor of a point or intersection or small delimited space in some complex topology is comforting because it still presents words and their meanings as somehow localized. The world of distributed representation offers no such images. How the system handles any particular input (its way of defining the input's meaning) is scattered throughout all the processors in the web.[29] It presents the true dispersal of meaning—the radical form of a familiar argument.

Distributed processing—that is, the matrix functions that handle distributed representations—is also disturbing because it is not rule-based. There are no "if-then" clauses, no step by step algorithms. What the layers of units in the processing system do can perhaps be described through such "if-then" notation; that is, the system is perhaps a "Turing-equivalent machine." Yet how the neural net goes about its business of being trained and processing input patterns has little to do with algorithmic symbol manipulation. For most people, this distinction may not be very significant. For students of analytic philosophy of mind and of the traditional version of the study of artificial intelligence, however, algorithms speak of order, clarity, rigor, and precise method. Neural networks possess these attributes at the local level—in the behavior of individual units and in the algorithms for adjusting

---

[29] In fact there are various ways of representing information that run from the fully distributed to the strongly localized, where only one cluster of processors in a sparse network responds to one input.

their connection weights—but the behavior of a neural network is an emergent property of the entire system.[30]

Indeed, the emergent properties that arise from networks of artificial neurons provide a final level of discomfort. Much good work has been done on the training of neural networks as types of complex dynamic systems. Many training algorithms have been explored, and many fascinating issues have arisen. For example, as the error-correction formula for the generalized delta rule suggests, training a network is essentially a process of minimizing the total error of the system for the set of input vectors. Most training strategies are a version of a "gradient descent" where small changes in connection values are made and tested to see their impact on the total error. The problem is, however, that the weights can drift into a situation where any change increases the error yet the current solution is not the *best* solution. This is the problem of "local minima." The way to avoid these premature solutions is to increase the step size for changes in training the weights (the value η in the generalized delta rule). Yet if the step size is too big the weights may never settle down to any optimal solution. These issues of *process*, though complex, are susceptible to systematic exploration. In contrast, the *results* of the training process—the logic of the pattern recognition system—grow

---

[30] There are many debates within the philosophical community about the adequacy of neural network accounts to describe human experience. Perhaps two of the best-known participants in these debates are Jerry Fodor, whose most recent book *Concepts: Where Cognitive Science Went Wrong* (Oxford: Oxford University Press, 1998) continues to roil the waters, and John R. Searle, whose *The Rediscovery of the Mind* (Cambridge, MA: MIT Press, 1992) continues his argument about the centrality of conscious experience that cannot be caught by connectionist paradigms.

increasingly unmanageable as the size of the network grows. For relatively simple problems, for example, researchers have elegantly exploited the properties of matrices and matrix multiplication to analyze the neural network's view of the dimensionality of the input domain, that is, what features of the set of patterns are important for efficiently discriminating between patterns. As the input patterns grow more numerous and complex, the size of the network has to grow, and the mathematics for analyzing the results becomes intractably complex. Yet to have a neural network solve a problem of pattern ordering but not to have a clue how the solution works can be a bit unsettling.

Given all the anxieties neural networks can engender, what do they offer in return? First, at the empirical level, they are indeed powerful and robust pattern detectors. Once a network has been trained on a set of patterns (input vectors) that span the domain of the input, it can cope with any other variation. Moreover, the network can handle degraded information where part of the pattern is missing or hidden by noise. Networks usually can also handle the loss of some of their artificial neurons without a significant loss in their trained ability. At the theoretical level, neural networks provide a conceptually elegant model for learning and brain function. They provide models for solving two tough questions. First, since the DNA sequencing for brain structure surely cannot encode fine details to represent information about the body or the environment, how do brains quickly build the representations they need to function? Secondly (although the questions are related), how does the brain use its billions of neurons and trillions of synaptic junctions to process input from the sensory apparatus and respond all within one

hundred steps? The next chapter will discuss the fit between the connectionist model for neurons and what we actually know about the structure and functioning of the brain.