Chapter 9: Building Associations

The first part of this book developed an argument that synthetic psychology was one approach that could be fruitfully explored in the study of mind. In very general terms, the aim of synthetic psychology is to build mental phenomena from the bottom up. A synthetic psychologist could proceed by proposing some basic "building blocks" to be used, and then by seeing what kinds of interesting and surprising phenomena could be created when these basic components are combined. Some researchers have argued that the synthetic approach promises to provide theories of mental phenomena that are simpler than those that can be produced by applying more traditional analytic methodologies.

The purpose of the current chapter is to begin our exploration of synthetic psychology by examining a proposal for a set of "building blocks" that originated over 2,000 years ago, but which still plays an important role in modern research in psychology, cognitive science, and the philosophy of mind. This architecture is an instantiation of the set of principles that define what has come to be known as *association psychology* (Warren, 1921), or simply *associationism*.

The main objective of this chapter is to propose an associationist architecture, to create computer simulations from these components, and to use these simulations to explore some of the advantages and disadvantages of associationism. This will provide us with some "hands on" experience with synthetic psychology. Later chapters will use more advanced connectionist networks to explore more sophisticated issues. We will see that these more advanced networks are easily thought of as refinements of some of the connectionist ideas that are introduced here, so this chapter will also provide a foundation upon which later simulations are going to be constructed.

This chapter proceeds as follows. First, it presents a brief historical overview of associationism. This is culminates in an account of William James' theory of association, which is used to motivate a more modern account of associative mechanisms. Second, it introduces this modern account by describing the properties of a particular connectionist network, called a distributed associative memory. This account defines the properties of processing units, modifiable connections, and the general operations used to train the network and to retrieve associations that have been stored in it. Third, the chapter describes a particular learning rule for this type of connectionist network, the Hebb rule. Mathematical analyses and the results of computer simulations are used to show the advantages and disadvantages of this learning rule. Fourth, a second training procedure, the delta rule, is defined in an attempt to overcome some of the problems that were uncovered with Hebb-style learning. The chapter ends with some brief reflections about how one might use a computer simulation of a distributed associative memory to explore some issues that have arisen in the modern study of association and learning.

9.1 From Associationism To Connectionism

In 1921, Howard Warren published *A history of the association psychology*, which traced associationism from Aristotle's reflections on memory (B.C. 382 – 322) to the psychological theories proposed by Herbert Spencer and George Henry Lewes in the 1870s. As far as Warren was concerned, association psychology in its most focused form ended at this time: "The association psychology culminated with Bain, Spencer, and Lewes. The evolution doctrine of the two last writers affords a wider scope to the play of association; but at the same time it opens the door to other factors, which have tended to lessen the importance of association in the eyes of the empirical investigator" (Warren, 1921, p. 16). Accordingly, Warren organized his history by considering four different periods of thought that ended with the work of a select group of 19th century thinkers.

In this section of the chapter, I will provide a highly selective history of associationism, and I will organize this history by adopting Warren's (1921) method of considering different periods of thought. However, we will be parting company with Warren in two important ways. First, we will consider some aspects of associationism that persisted beyond the era of Bain. In particular, we will examine the associationism of William James, a psychologist whose contributions are only briefly considered by Warren. Second, we will use James' thoughts about associationism as a springboard to very modern associationist models in cognitive science. In particular, we will see that James laid the foundation for a particular type of connectionist model, called a *distributed associative memory*. Contrary to what Warren's history implies, associationism has survived, and even flourished, into the new millennium.

9.1.1 Philosophical considerations

A very long line of philosophers and psychologists are responsible for the development of associationism. Because of this, this theory of mental phenomena has been developed by an array of individuals who have espoused a bewildering array of conflicting positions (Warren, 1921). While there is enormous diversity amongst their positions, these thinkers were in considerable agreement about the methods to be used to support their claims about mentality. "The bond of union in the association movement […] is found chiefly in the unwavering devotion of its adherents to the empirical method in psychology" (p. 13). Until late in the 19th century, this empirical method consisted of the introspective examination of mental activities. One of the main observations that introspection revealed was that there existed sequences of thought which were experienced during memory and thinking. Associationism grew out of the attempt to provide lawful accounts of these sequences of thought.

9.1.1.1 Aristotelian Contributions

The earliest detailed introspective account of such sequences of thought can be found in the writings of Aristotle. In his short essay *De memoria et reminiscentia*, Aristotle provided an account of memory that "is fuller than that to be found in the best-known British empiricists" (Sorabji, 1972, p. 1). In the early part of this essay, Aristotle argued that the contents of memory are essentially visual images that resemble the things being memorized. "For it is clear that one must think of the affection, which is produced by means of perception in the soul and in that part of the body which contains the soul, as being like a sort of picture, the having of which we say is memory. For the change that occurs marks in a sort of imprint, as it were, of the sense-image, as people do who seal things with signet rings" (p. 50).

Later in the essay, Aristotle turned to the process of recollecting thoughts that have been remembered. His account of recollection has all of the elements of the association psychology from the 19th century. He focused upon the sequence of thought: "Acts of recollection happen because one change is of a nature to occur after another" (Sorabji, 1972, p.54). A particular sequence of images occurs because either this sequence is a natural consequence of the images, or because (through repetition) the sequence has been learned by habit. Recall of a particular memory, then, is achieved by cuing that memory with the appropriate prior images. "Whenever we recollect, then, we undergo one of the earlier changes, until we undergo the one after which the change in question habitually occurs."

For Aristotle, recollection by initiating a sequence of mental images was not a haphazard process. The first image in the sequence could be selected in such a way that the desired image would be recollected fairly easily, by taking advantage of possible relationships between the starting image and the image to be recalled. Aristotle considered three different kinds of relationships between the starting image and its successor: similarity, opposition, and (temporal) contiguity: "And this is exactly why we hunt for the successor, starting in our thoughts from the present or from something else, and from something similar, or opposite, or neighboring. By this means recollection occurs" (Sorabji, 1972, p. 54).

In Aristotle's account of recollection, we see three characteristics that recur in all the later theories that defined the association psychology. First, there is the (introspective) observation that thought occurs in sequences. Second, there is a claim about the nature of the mental entities that make up this sequence (e.g., mental images). Third, there is a claim about lawful relationships between these entities, such that when one comes to mind, this relationship will lead to the recollection of the next component of the sequence. These relationships are generally considered to be laws of association, and Aristotle's proposal of three such laws (which in later theories would be called the law of similarity, the law of contrast, and the law of contiguity or the law of habit) is completely consistent with proposals made centuries later.

Later researchers accepted the main points of Aristotle's associationism with only minor qualifications. For instance James (1890, p. 594) wrote, "Aristotle seems to have caught both the facts and the principle of explanation; but he did not expand his views." However, Aristotle's observations on memory were essentially ignored (perhaps because they were not understood – see (Warren, 1921, p.28) -- for many centuries. Advances in associationism did not occur until the 17th century.

9.1.1.2 17th Century Associationism In Philosophy

One prominent feature of Aristotle's treatment of associationism was that he only applied the laws of association to one domain of experience, that of memory. This feature was preserved through the middle ages. "The many commentators on Aristotle during the middle ages took up the passage on recollection which has been quoted. They discussed an amplified it, as they did every saying of the master, but without throwing any new light on association" (Warren, 1921, p. 30). One reason for this very long period of dormancy was the fact that departures from Aristotle were akin to heresy: "Any freshness or originality was frowned upon; the only advances came from new interpretations – and these too often were misinterpretations" (p. 30).

This situation began to change in the 17th century with the philosophical writings of Thomas Hobbes (b. 1588 – d. 1679). Hobbes was particularly important for setting the stage to broaden the import of association, by applying it to thought processes in general, and not just to memory in particular. He presented three separate themes that permeated the writings of those that followed him. First, he distinguished sense (or sensations) from memory; memory was viewed as mental images of what was sensed. Second, he noted that images are experienced in succession, and argued for the need to explain this succession. Third, he attempted to use principles of association to explain sequences of thought.

Hobbes' work on this third issue was not particularly successful, but his work inspired later philosophers who had greater success than did he. "The British thinkers who followed him developed their systems of psychology along the lines that he marked out; the notion of association, which he did little more than outline, became more and more prominent as the analysis was perfected" (Warren, 1921, p. 33).

The most important philosopher who followed Hobbes in this era was John Locke (b. 1632 – d. 1704). Locke coined the phrase "association of ideas", which first appeared as a chapter title in the fourth edition (1700) of *An essay concerning human understanding*. Locke's fame as a philosopher came late in his life; the first edition of this book was published in 1690 when he was 57 years old. However, this fame and influence was long lasting, and his chapter on association launched British empiricism.

Locke's work was a reaction against the nativism espoused in the philosophy of Descartes, and was primarily concerned with establishing experience as the foundation of all thought. Following Hobbes, Locke distinguished between ideas of sensation and ideas of reflection. He was particularly interested in the composition of simple ideas into more complex ideas, as well as the sequence of appearance of ideas. One reason for this interest was because these connections (from simple to complex, or from one idea to the next in a sequence) did not seem to necessarily reflect a natural order. Instead, Locke realized that these connections were due to experience. "There is another connexion of ideas wholly owing to chance or custom: ideas that in themselves are not at all of kin, come to be so united in some men's minds that it is very hard to separate them, they always keep in company, and the one no sooner at any time comes into the understanding but its associate appears with it; and if they are more than two that are thus united, the whole gang, always inseparable, show themselves together" (Locke, 1977, p. 219).

Interestingly, while Locke anticipated the law of frequency that was later endorsed by J. S. Mill, and alluded to association by contiguity and by similarity, he did not explore specific associative laws. One reason for this may be that his primary goal was to argue for the existence of ideas formed by association; this was more important to him than an analysis of associative mechanisms. A second reason may be because Locke was not in a position to offer any strong arguments in favor of any particular causal process underlying association. After describing association as being responsible for a keyboard player retrieving a long sequence of finger movements during a performance, Locke noted "whether the natural cause of these ideas, as well as that regular dancing of his fingers, be the motion of his animal spirits, I will not determine, how probable soever, by this instance, it appears to be so" (Locke, 1977, p. 220).

It is clear that the primary result of 17th century philosophy's analysis of association was to renew scholarly interest in this topic, and to set the stage for more technical advances that would come later. Issues that were pioneered by Aristotle once again became central concerns to philosophers, and did so in a context that permitted Aristotle's views to be criticized and modified.

9.1.1.3 18th Century Philosophy And Associationism

Locke's immediate philosophical successor was the Bishop of Cloyne, George Berkeley (b. 1685 – d. 1753). Unlike Locke, Berkeley published his most influential works at the relatively young age of 25. Berkeley was primarily important for transforming the problem of knowledge from one that was essentially philosophical to one that was more consistent with the strong psychological overtones that marked theories of association that developed later. Like Hobbes and Locke, Berkeley divided mental content into ideas of sensation and into ideas of imagination, and was primarily interested in accounting for the natural succession of ideas. He reiterated Aristotle's law of contiguity, and extended it to account for associations involving different modes of sensation. "From a frequently perceived connection, the immediate perception of ideas by one sense suggests to the mind others, perhaps belonging to another sense, which are wont to be connected with them" (Warren, 1921, p. 41). In other words, Berkeley – unlike Locke -- was one of the first philosophers after Aristotle to develop an account of "modes of association", which described the laws that determined how associations came to be.

A more detailed and elaborate theory of modes of association was to be found in the work of philosopher David Hume (b. 1711 – d. 1776). Hume, like his predecessors, began by dividing experience into impressions and ideas, and viewed the latter as being weaker or less vivid copies of the former. He then turned to consider principles that explained the connection between successive ideas.

In his original treatment, Hume, who was likely unaware of similar ideas put forth by Aristotle, proposed three different laws of association: resemblance, contiguity in time or place, and cause or effect. "That these principles serve to connect ideas will not, I believe, be much doubted. A picture naturally leads our thoughts to the original; the mention of one apartment in a building naturally introduces an enquiry or discourse concerning the others; and if we think of a wound, we can scarcely forbear reflecting on the pain which follows it" (Hume, 1952, p. 23).

Later, Hume argued that association by cause or effect could not be distinguished from association by contiguity, and thus settled on two associative laws: contiguity and resemblance (or similarity).

Hume's work on association was monumentally influential, but did have one shortcoming, in that Hume did not attempt to use his laws of association to account for all mental phenomena. This was not attempted until the treatment of association offered by David Hartley (b. 1705 – d. 1757). Hartley was not only able to show the broader implications of Hume's theory, but also provided one of the earliest examples of an attempt to root association in terms of brain function. Hartley constructed a theory of vibrations that attempted to draw a close correspondence between mental associations and neural activity. Hartley saw contiguity as the primary source of associations, and ignored Hume's law of resemblance. He also anticipated the associationism of J.S. Mill by recognizing repetition as a source of association, or at least as a factor that could affect the strength of an association.

Hartley had nowhere near the fame or impact of Hume, but he is usually regarded as the founder of associationism. This status is often granted to him not because of his theoretical innovations, but instead because of his unusual success in promoting associationism's cause. "He took Locke's little-used title for a chapter, 'the association of ideas', made it the name of a fundamental law, reiterated it, wrote a psychology around it, and thus created a formal doctrine with a definite name, so that a school could repeat the phrase after him for a century, and thus implicitly constitute him its founder" (Boring, 1950, pp. 193-194).

9.1.1.4 19th Century Philosophy And Associationism

The 19th century marked a period in which associationism evolved from a topic that was primarily philosophical into one that was predominately psychological. In 1829, James Mill (b. 1773 – d. 1836) published his *Analysis of the human mind*. The third chapter of this psychological text was on associationism. Many of the ideas put forth in this chapter were familiar: Mill divided mentality into sensations and ideas, where ideas were once again proposed as being copies or traces of sensations. Mill observed that sensations occur either simultaneously or in successive order, and that ideas presented themselves in the same sequence as did the sensations that they copied. His associationism, like those of the philosophers that we have already seen, attempted to account for the succession of ideas.

The 19th century was also an era in which writers assumed the fundamental notions of associationism as a given, and turned to fleshing out the details. Often this theme revealed itself in one writer critiquing the modes of association proposed by a predecessor. For Mill, the only law of association was contiguity. He explicitly denied Hume's laws of cause or effect and resemblance. Mill also emphasized the importance of individual associations varying in strength. For Mill, association was essentially a mechanical process by which complex ideas were created by associating simpler ideas together. Because of his mechanical metaphor, emergence played no role in Mill's associationism. For Mill, a complex idea was no more than the sum of its components, and if one understood these, then one should be able to completely understand the larger idea that they comprised.

Mill's ideas were challenged and modified by his own son, John Stuart Mill (b. 1806 – d. 1873). John Stuart Mill provided many modifications to his father's theory of associationism. First, he argued that ideas were indistinguishable from sensations, and were not just less vivid copies. He then posited a completely different set of associative laws, which included a reintroduction of Hume's law of similarity: "The first is that similar ideas tend to excite one another. The second is that when two impressions have been frequently experienced (or even though of) either simultaneously or in immediate succession, then whenever one of these impressions or the idea of it recurs, it tends to excite the idea of the other. The third law is that greater intensity in either or both of the impressions is equivalent, in rendering them excitable by one another, to a greater frequency of conjunction" (Warren, 1921, p. 96).

One of John Stuart Mill's most interesting departures from his father's associationism was replacing a mechanistic account of complex ideas with an account that was described as a "mental chemistry". In this mental chemistry, when complex ideas were created via association, the resulting whole was more than just the sum of its parts. As a result, the laws governing the whole (e.g., successions to other ideas) could not be predicted by knowing the laws governing the simpler ideas that served as parts. In other words, John Stuart Mill proposed an associationism that endorsed an early form of emergence.

One of John Stuart Mill's friends was Alexander Bain (b. 1818 – d. 1903), who eventually became chair of logic at Aberdeen. "His book on logic is overshadowed by J.S. Mill's great classic, just as the latter's fragmentary psychology is dominated by Bain's exhaustive two-volume treatise" (Warren, 1921, p. 104). Bain's two major books, which constitute two volumes of one complete project, were *The senses and the intellect*, which was published in 1855, and *The emotions and the will*, which was published in 1859. Bain spent many of his later years revising these books, which served as the standard British psychology text for half a century.

Bain's associationism is, in many respects, a refinement of John Stuart Mill's. Bain invoked four different laws of association, and attempted to reduce all intellectual processes to these laws. One of these laws was the law of contiguity, which has been present in every theory of association that we have reviewed. A second was the law of similarity, which was revived from Hume by both Bain and J.S. Mill after being banished by James Mill. The third was the law of compound association: "Past actions, sensations, thoughts, or emotions are recalled more easily, either through contiguity or similarity, with *more than one* present object or impression" (Warren, 1921, pp. 107-108). This law was an important precursor to William James' treatment of associations between patterns, which we will consider in more detail shortly. The fourth was the law of constructive imagination: "By means of association the mind has the power to form new combinations or aggregates, different from any that have been presented to it in the course of experience" (p. 109). This law represents an important psychological contribution of Bain, in that he was attempting to explain creative thought in terms of associative principles.

9.1.2 Psychology, Associationism, and Connectionism

Bain represents a bridge between philosophical and psychological treatments of association. Bain stood "exactly at a corner in the development of psychology, with philosophical psychology stretching out behind, and experimental physiological psychology lying ahead in a new direction. The psychologists of the twentieth century can read much of Bain with hearty approval; perhaps John Locke could have done the same" (Boring, 1950, p. 240). In this section, we will consider some of the key developments of the psychological associationism that was inspired by Bain's work. However, this review will be extremely selective, because we will use it to motivate a discussion of a very particular kind of connectionist network.

9.1.2.1 19th Century Contributions Of William James

The pioneer of the "New Psychology" in North America was William James (b. 1842 – d. 1910). He received his medical degree from Harvard in 1869, and returned to Harvard as a professor later in his career. In 1872 he was a professor of physiology there, and offered instruction in physiological psychology as early as 1875. In 1885 he became a professor of philosophy, and in 1889 his title changed to professor of psychology. James created the first demonstrational psychology laboratory in North America, and in 1890 published a profoundly influential psychology text in two volumes, *The principles of psychology*. "The key to his influence lies…in his personality, his clarity of vision, and his remarkable felicity in literary style" (Boring, 1950, p. 509).

James' treatment of association is found in Chapter 14 of *The principles of psychology*. His thoughts on this topic were inspired by the work of Bain and others, but were also an innovative reaction against philosophical associationism. James was in particular concerned about the fact that philosophical associationism had not made any serious proposals concerning the causal mechanisms that instantiated modes of association. For example, he offers the following assessment of Bain: "His pages are painstaking and instructive from a descriptive point of view; though, after my own attempt to deal with the subject causally, I can hardly award to them any profound *explanatory* value" (James, 1890, p. 601).

James was of the opinion that explanatory accounts had eluded previous associationists because of a fatal flaw in their approach. This flaw was the assumption that associations were made between mental contents (e.g., the images, reflections, or ideas that had been proposed by most of James' predecessors as being copies or traces of sensations). James argued that if association was a mechanical process, then it must apply to objects and not ideas; he then proposed a particularly psychological theory by arguing that the objects being associated were brain states: "*Association*, so far as the word stands for an *effect*, *is between* THINGS THOUGHT OF – *it is* THNGS, *not ideas, which are associated in the mind*. We ought to talk of the association of *objects*, not of the association of *ideas*. And so far as association stands for a *cause*, it is between *processes in the brain* – it is these which, by being associated in certain ways, determine what successive objects shall be thought" (James, 1890, p. 554).

In terms of viewing association as an effect, James' theory was not a radical departure from others that we have considered in this chapter. First, he was primarily concerned with providing an account of the succession of thoughts. Second, his theory attempted to explain this succession via associative law. For James, the only explanatory mode of association was contiguity, which he called the law of habit. While he admitted that other factors could be described as affecting association (similarity, vividness, recency, emotional congruity), he attempted to show how all of these could be explained in terms of contiguity.

James was able to reduce other laws of association to the law of contiguity when he departed from the traditional view of association as an effect, and replaced it with the view of association as a cause. There are several central elements to his physiological account of association. First, James recognized that one idea or event could be represented in the brain as a pattern of activity across a set of more than one neuron. Second, he expressed his law of habit in terms of a process that affected the ease of transit of a nerve-current through a tract: "The psychological law of objects thought of through their previous contiguity in thought or experience would thus be an effect, within the mind, of the physical fact that nerve-currents propagate themselves easiest through those tracts of conduction which have been already most in use" (James, 1890, p. 563). Third, he viewed the succession of thoughts that one experiences as due to the fact that activity in one brain state (i.e., some set of neurons) leads to activity in some different brain state that had previously been associated with the first. "When two elementary brain-processes have been active together or in immediate succession, one of them, on reoccurring, tends to propagate its excitement into the other" (p. 566). Finally, James was predominately concerned with predicting which subsequent brain state would be activated by a prior brain state, given that one idea might be associated with a number of different ideas, other at different times or in different ways. James attempted to explain this kind of variation by realizing that any given neuron would be receiving signals from a number of other neurons, and that its degree of activation would depend on an entire pattern of input, and not upon an association with a single incoming signal. "The amount of activity at any given point in the brain-cortex is the sum of the tendencies of all other points to discharge into it, such tendencies being proportionate (1) to the number of times the excitement of each other point may have accompanied that of the point in question; (2) to the intensity of such excitements; and (3) to the absence of any rival point functionally disconnected with the first point, into which the discharges might be diverted" (p. 567).

The main physiological points of James' theory of association are summarized in Figure 9-1, which is analogous to his own Figure 40 in his chapter on association (James, 1890) (p. 570). The figure represents two ideas, one (*A*) being the last act of a dinner party, the other (*B*) being walking home through the frosty night. Each of these ideas is represented in the brain as a pattern of activity in a set of neurons. *A* is represented by activity in neurons *a, b, c, d,* and *e*; *B* is represented by neurons *l, m, n, o,* and *p*. The association between A and B occurs because A preceded B in the course of an evening. As a result, the neurons representing A were active immediately prior to the activity of the neurons representing B, and the tracts connecting the neurons (represented as the lines in Figure 9-1) were modified according to the law of habit. The ability of A's later activity to lead to the thought of B, is due to these modified connections between the two sets of neurons. "The thought of A must awaken that of B, because *a, b, c, d, e*, will each and all discharge into *l* through the paths by which their original discharge took place. Similarly they will discharge into *m, n, o*, and *p*; and these latter tracts will also each reinforce the other's action because, in the experience B, they have already vibrated in unison" (p. 569).

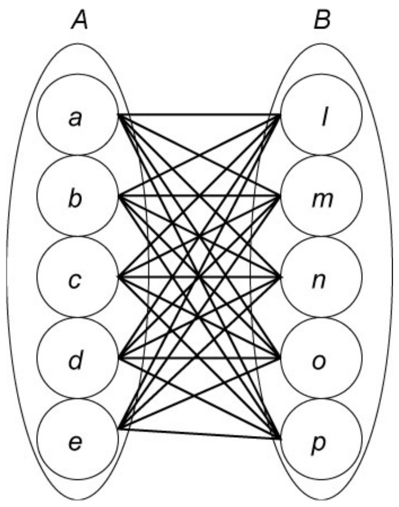

Figure 9-1. James' associative memory. See text for details on how this systems was theorized to

9.1.2.2 The Paired Associate Task

The type of association envisioned by James, and illustrated in Figure 9-1, leads to one methodological topic that will be central to the simulations that will be introduced later in the chapter. In James' example, associative memory is viewed as having two different functional stages. The first is learning, in which an association between two ideas is stored. As we saw in the previous section, this occurs when two ideas (A and B) occur either simultaneously or in close succession to one another. As a result of this co-occurrence, the connections between the neurons representing both A and B are modified to permit easier transmission of "nerve-currents". The second stage is recall. During this stage, only one of the two previous ideas is present (A). When its underlying neural processes become active, they serve to activate those associated with the other idea (B), bringing it to mind.

This two-stage account of association was used to develop a particular paradigm used to study human memory called the paired associate task. This method of examining memory presents stimuli in a fashion similar to what would be the case if someone were learning the vocabulary of a foreign language (Kintsch, 1970). Subjects learn a list of stimulus-response pairs. Sometimes this learned via the "study-test method". With this method, subjects are presented both members of the pair at the same time, and attempt to remember the association between the two. In the test phase of this method, subjects are only presented the stimulus, and must attempt to recall the associated response on their own. Sometimes the list is learned via the "anticipation method". In this method, subjects are only presented the stimulus term, and must generate the response on their own. On the first presentation, they will of course be forced to guess a response. Once this guess is made, the subject is provided with the stimulus and the correct response together. Later, the same stimulus will be presented, and the subject will be given the opportunity to recall the associated response. The paired-associate learning task was used with great success to study the issue of whether learning was all-or-none or was instead due to an increment in continuously varying response strength.

Mary Whiton Calkins (b. 1863 – d. 1930), who was among the first generation of women to enter psychology, invented the paired associate task (Furumoto, 1980). Calkins graduated in

1885 with degrees in philosophy and in the classics from Smith College. In 1887 she began to teach Greek, philosophy, and psychology at Wellesley college. She has created a psychology laboratory at Wellesley by 1891. While teaching, she furthered her education by attending Harvard from 1892 to 1895, during which time she was enrolled in James' psychology course. Because she was a woman, she was not awarded a PhD from Harvard, in spite of the fact that she had completed all of the requirements and had the support of her professors. In 1896, she published a paper in *Psychological Review* that provided the first description of the paired associate task. In this paper, she used the study-test method, employing colors as stimuli and numbers as cues. There is no doubt that she was inspired to invent this technique by considering ways in which James' theory of association could be put to the test in an experimental laboratory.

9.1.2.3 20th Century Models Of Distributed Memory

After the cognitive revolution in the second half of the 20th century, many researchers turned to using computer simulations to study human memory processes. In this section of the chapter, we will be interested in simulations that share two general characteristics. First, they are designed to perform the paired associate task, and are generally trained using some variation of the study-test method. Second, they are closely related to the kind of associative memory envisaged by James, and which was illustrated in Figure 9-1.

Some of the earliest research on parallel systems was concerned with the development of distributed memories capable of learning associations between pairs of input patterns (e.g. Steinbuch, 1961; Taylor, 1956), or of learning to associate an input pattern with a categorizing response (e.g., Rosenblatt, 1962; Selfridge, 1956; Widrow & Hoff, 1960). Figure 9-2 depicts a connectionist network that is often used to introduce some of the basic properties of PDP connectionism, and has come to be called the *standard pattern associator* (McClelland, 1986). This network is designed to implement what is known as a distributed associative memory that functions in a fashion that is very reminiscent of James' theory of association. For this reason, Figure 9-2 has been labeled in such a way as to draw out its similarities with Figure 9-1.

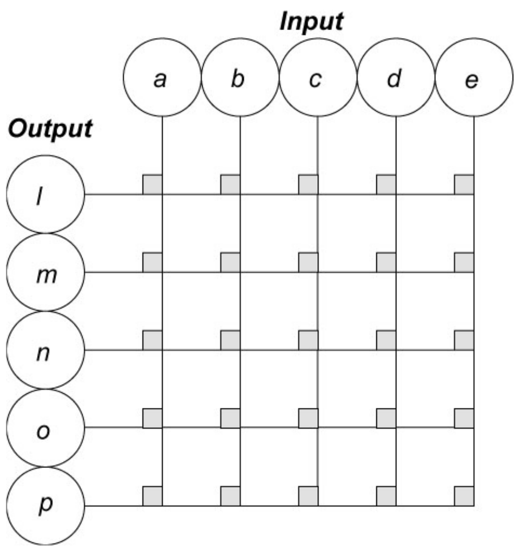

Figure 9-2. James' associative memory redrawn as the standard pattern associator.

When I first began work on learning in connectionist networks, I had the pleasure of working on a number of different projects with Don Schopflocher. Don was of the opinion that while there were many differences between connectionist researchers, they were all united by their tacit understanding of what Figure 9-2 represents. In support of Don's position, versions of Figure 9-2 have a long history (e.g., Kohonen, 1977, Fig. 1.9; McClelland & Rumelhart, 1988, Chap. 4 Fig. 3; Rumelhart, McClelland, & Group, 1986, Chap. 1 Fig. 12, Chap. 9 Fig. 18, Chap. 12 Fig. 1, Chap. 18 Fig. 3; Schneider, 1987, Fig. 1; Steinbuch, 1961, Fig.2; Taylor, 1956, Figs. 9 & 10). Of course, this history can be traced back to James' own figure in his chapter on association. If connectionists ever decide to wear an identifying tattoo, then perhaps Figure 9-2 would be an excellent candidate.

As will be detailed below, the standard pattern associator is constructed from the processing units and modifiable connections defined in the PDP architecture. It consists of two sets of processing units; one is typically called the input set, the other the output set. During a learning stage, the activation states of the input processing units are used to represent a cue pattern and the activation states of the output processing units are used to represent a to-be-recalled pattern. The connection weights are then modified to store the association between the two patterns. The standard pattern associator is called a distributed memory because this association is stored throughout all the connections in the network, and because one set of connections can store several different associations. During the recall stage, a cue pattern is presented to the network by activating the input units. This causes signals to be sent through the connections in the network. These signals, in accord with James' theory, activate the output processors. If the memory is functioning properly, then the pattern of activation in the output units will be the pattern that was originally associated with the cue pattern.

9.2 Building An Associative Memory

Up to this point in the chapter, we have reviewed a history of associationism that has culminated in the standard pattern associator. The remainder of this chapter is intended to provide a technical account of this kind of system. This account will accomplish several different goals. First, it will serve as an introduction to some of the building blocks of a connectionist network. These same building blocks will be used to construct more sophisticated networks in the chapters that follow. Second, it will provide some examples of how one might study paired-associate learning synthetically. Third, it will introduce the reader to a software package that can be used to empirically explore associative memory. The user interface for this software package will be maintained in programs that are introduced in later chapters.

9.2.1 Defining the problem

The purpose of the computer simulation is to build a memory system that is capable of storing associations between pairs of items. During a learning phase, the system will be presented pairs of stimuli. For each pair, it will determine how they are to be associated together, and store this association in memory. During a recall phase, the system will be presented with only one member of a pair. Using this member as a cue, it will use its memory to attempt to recall the other member of the pair to the best of its ability. In order to create a system that will behave in this fashion, we will construct a very simple connectionist network. The network will consist of an input "bank" of processing units, an output bank of processing units, and a set of modifiable connections between these two banks. The basic design of the network was illustrated in Figure 9-2. As we will see, several independent associations can be stored in the same set of connection weights.

9.2.2 The Network Architecture

9.2.2.1 Processing Units

Ultimately, both the input units and the output units can be considered as sets of numbers, with each number representing a property of an individual unit (e.g., its internal level of activity), and with the entire set of numbers representing a pattern across a whole bank of units (e.g., the pattern of activity of the bank of input units). It will be useful to represent these sets of numbers as vectors, because linear algebra provides an extremely compact and useful notation for exploring the properties of distributed associative memories and of other connectionist networks.

For example, we might represent the activity of input unit 1 with the numerical value a_1, the activity of input unit 2 with the numerical value a_2, and so on. The set of activities for all of the input units could be represented as the vector $\boldsymbol{a}$, whose first entry would be the value a_1, whose second entry would be the value a_2, and so on. By convention, when we talk about the vector $\boldsymbol{a}$ we will assume that it is a column vector. This means that when all of the values of the vector are listed out, they are strung out vertically in a column, as is shown in Figure 9-3. In some cases, the operations of linear algebra assume that a vector is a row vector, which means that when its values are listed out, they are strung out horizontally in a row, as is also shown in Figure 9-3.

The operation that converts a column vector into a row vector is called transposition. Because of this, if we were to indicate that a vector was a row vector, then we would do so with a notation that included a superscript "T", to explicitly indicate that the vector had been transposed. For instance, if vector ***a*** is a column vector of unit activities, then vector $\boldsymbol{a}^T$ would be a row vector of the same numerical activities.

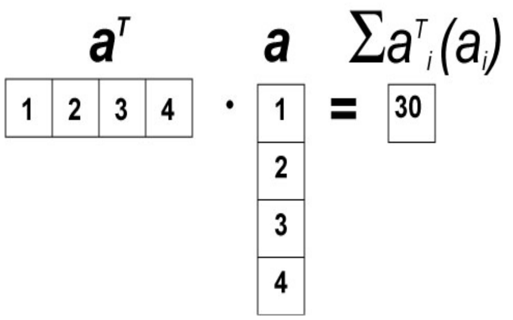

Figure 9-3. The inner product (see Equation 9-5) of the row vector a^T and the column vector *a* is a single number, which is the sum of the products of the entries of the two vectors.

In order to represent the properties of banks of units as vectors, we need to define some equations that dictate what numerical values should be inserted into the vectors. Any processing unit in a connectionist network can be described using three different mathematical equations. The first equation is the net input function, which describes how a processing unit computes the total signal coming into it from other processors in the network. The second equation is the activation function, which determines how a processing unit converts this input signal into a number that represents its internal level of activity. The third equation is the output function, which defines how a processor's internal level of activity is converted into a numerical signal that can be sent through connections to other processors in the network. When these three equations are used to describe the processors used in our memory network, it will become apparent that they are particularly simple.

In the distributed associative memory network that we are constructing, the activity values of the input units are always set by the programmer, who simply turns each input unit on to the desired level of activity (i.e., the level of activity that represents information about one member of the to-be-associated pairs of patterns). For the sake of consistency with later chapters, we will describe this in terms of a net input function. Specifically, the net input for input unit *i* (net_i) is equal to the environmental stimulation for that input unit (e_i):

$$net_i = e_i \qquad \text{(Equation 9-1)}$$

The input processors in the distributed associative memory are particularly simple because after their net input is computed, its value is used as the value of the processor's internal activity and as the value that the processor outputs to the output units. Mathematically speaking, the activation function and the output function for the input units are both identity functions. That is, the internal activity of input unit *i* (a_i) is defined as:

$$a_i = net_i \qquad \text{(Equation 9-2)}$$

Similarly, the output activity of input unit i (o_i) is defined as:

$$o_i = a_i \qquad \text{(Equation 9-3)}$$

During the learning phase, the output units are treated exactly as are the input units. That is, the programmer sets their activity values to represent the other member of the to-be-associated pair. Because of this, during learning, the output units can be described using exactly the same equations that were used to describe the input units (i.e., Equations 9-1, 9-2, and 9-3). During the recall phase, the output units have their net input determined by signals that are sent from the input units, and therefore require a slightly more elaborate net input equation.

Imagine a very simple network in which there are 8 different input units, and only 1 output unit. Each of the input units is linked to the output unit by a connection. Each connection is weighted, where a connection weight is simply some numerical value. When a numerical signal is sent through a connection, the connection scales the signal by multiplying it by the value of its connection weight. Let us represent the weight of the connection between input unit 1 and the output unit as w_1, between input unit 2 and the output unit as w_2, and so on. During recall in this simple network, each of the input units will be sending a signal to the output unit. Input unit 1 will be sending the signal o_1, input unit 2 will be sending the signal o_2, and so on. The signal o_1 will be multiplied by the weight value w_1 before it reaches the output unit. So part of the signal that reaches the output unit will be the value o_1w_1. Following the same logic for the other input units, the output unit will also be receiving the signal o_2w_2, o_3w_3, and so on. In other words, the total signal for the output unit – its net input – will be:

$$\begin{aligned} net_1 &= o_1w_1 + o_2w_2 + o_3w_3 + \ldots + o_8w_8 \\ &= \Sigma o_iw_i \end{aligned} \qquad \text{(Equation 9-4)}$$

Linear algebra can be used to make this equation more compact. (For an excellent introduction to linear algebra that is framed in the context of connectionist networks, the reader is referred to Jordan, 1986). Let us take the signals being output by the input units and represent them as the row vector $\boldsymbol{o}$, and let us take the set of connection weights between the input units and the output units and represent them as the column vector $\boldsymbol{w}$. These two vectors can be combined using an operation called the inner product or the dot product (see Figure 9-3). The result of this operation is a single number (net_1, representing the net input for output unit 1) whose value is defined in Equation 9-4 – in fact, Equation 9-4 shows how an inner product is to be computed. In the notation of linear algebra, the inner product that defines the net input for the output unit is:

$$net_1 = \boldsymbol{o}^{\boldsymbol{T}} \bullet \boldsymbol{w} \qquad \text{(Equation 9-5)}$$

One way to remember that the result of an inner product like Equation 9-5 is a single number is to note the number of rows in the first component ($\boldsymbol{o}^{\boldsymbol{T}}$ is a row vector, and therefore has only one row) and to note the number of columns in the second component ($\boldsymbol{w}$ is a column vector, and therefore has only one column). The result of the operation will have the same number of rows as the first component, and the same number of columns as the second component. In other words, the result of an inner product will be a single number – a vector with only one row and only one column.

The inner product described in Equation 9-5 defines the net input for a single output unit. We will see in later chapters that the inner product is a standard net input function for all of the processors in more sophisticated connectionist networks.

9.2.2.2 Modifiable Connections

In the previous section, when we defined the net input function for a single output unit during recall, we represented the set of connection weights from a bank of input units to the output unit as a vector. Our goal in designing the distributive associative memory is to have a system that uses more than one output unit, so that it can recall a complete pattern of activity. It stands to reason that we would need to represent the connection weights for this more complicated memory with a set of weight vectors, with each vector in the set holding the connection weights associated with one of the output units.

In linear algebra, this set of vectors would be represented as a single entity called a matrix. If our memory had n input units, and m output units, then all of the connection weights between the input and output processors would be represented by one weight matrix, $\boldsymbol{W}$, which would have n rows and m columns. Each entry in this matrix, w_{ij}, would contain a number representing the weight of the connection from input unit i to output unit j.

By representing all of the connection weights with the matrix ***W***, we can take advantage of linear algebra to create a very compact mathematical description of how weights are modified, and we can also define very simple equations that describe how information stored in this matrix can be retrieved when the memory system is presented with a cue. When the distributed associative memory stores associations between patterns, it does so by modifying the strengths of its connection weights. This is done in two steps.

First, the memory computes changes in weights that are required to represent the association between the pair of patterns presented to it during a learning trial. Later in this chapter we will discuss two different equations that could be used to compute the desired weight changes. In this first step, all of the desired weight changes are stored in the matrix Δ_{t+1}, where the subscript *t+1* indicates the learning trial during which the changes have been computed. This matrix has the same number of rows and columns as does matrix ***W***, and each entry δ_{ij} in this matrix represents the value by which the connection weight between input unit *i* and output unit *j* should be changed.

Second, the memory uses the matrix Δ_{t+1} to change the existing connection weights. Let us use the subscript *t* to represent the network's connection weights at a particular trial of learning. Using the current weights, represented in the matrix $\mathbf{W}_t$, and the desired weight changes, stored in the matrix $\Delta_{t+1,}$ the goal is to compute the new values of weights, which will be stored in the matrix $\mathbf{W}_{t+1.}$ This is done by computing the sum of the matrices that represent the current weights and the desired weight changes: $\mathbf{W}_{t+1} = \mathbf{W}_t + \Delta_{t+1}$. Every value w_{ij} at row *i* and column *j* of the new weight matrix is simply equal to the sum of the value w_{ij} in matrix $\mathbf{W}_t$ and of the value δ_{ij} in matrix Δ_{t+1}.

With this notation, learning can be described as a series of matrix additions. Imagine that prior to learning, our memory system is truly a "blank slate", because all of its connection weights are equal to zero. The null matrix, ***0***, is the special matrix that has every value in it equal to zero. So at time 0, before learning as started, we could declare that $\mathbf{W}_0 = \mathbf{0}$. At learning trial 1, the new weights ($\mathbf{W}_1$) are equal to the old weights (the null matrix) plus the desired weight changes (Δ_1). At learning trial 2, the new weights ($\mathbf{W}_2$) are equal to the old weights ($\mathbf{W}_1$) plus the desired weight changes (Δ_2). As can be seen from Table 9-1, this kind of learning can continue for as many trials as is desired. Furthermore, Table 9-1 demonstrates that at any point in time after learning has begun, the memory's connection weights are essentially the sum of a series of matrices each of which contains the weight changes that are desired to store an association between a pair of stimuli.

Trial (*t+1*)	**Equation Describing Weight Values**
0	$\mathbf{W}_0 = \mathbf{0}$
1	$\mathbf{W}_1 = \mathbf{W}_0 + \Delta_1 = \mathbf{0} + \Delta_1 = \Delta_1$
2	$\mathbf{W}_2 = \mathbf{W}_1 + \Delta_2 = (\Delta_1) + \Delta_2$
3	$\mathbf{W}_3 = \mathbf{W}_2 + \Delta_3 = (\Delta_1 + \Delta_2) + \Delta_3$
4	$\mathbf{W}_4 = \mathbf{W}_3 + \Delta_4 = (\Delta_1 + \Delta_2 + \Delta_3) + \Delta_4$

Table 9-1. Associative learning described a series of matrix additions.

9.2.2.3 The Retrieval Operation

Before introducing a specific equation for calculating the association between pairs of stimuli, let us assume that we have a distributed memory that has already undergone some training, and therefore has a pre-existing set of connection weights that are represented in the matrix ***W***. What we would like to do is to present a vector of activity to the input units of this memory that will be used as a cue to retrieve some information, which will also be represented as a vector of activity in the memory's output units. To define this kind of retrieval mathematically, let the column vector ***c*** represent the cue pattern, and let the column vector ***r*** represent the recalled pattern. In linear algebra, the equation for recall from the distributed associative memory is:

$$\boldsymbol{r} = \boldsymbol{Wc} \qquad \text{(Equation 9-6)}$$

In other words, if one takes the matrix of weights that have been produced by learning, and uses this matrix to premultiply the cue pattern's vector, the result will be a column vector that holds the recalled pattern.

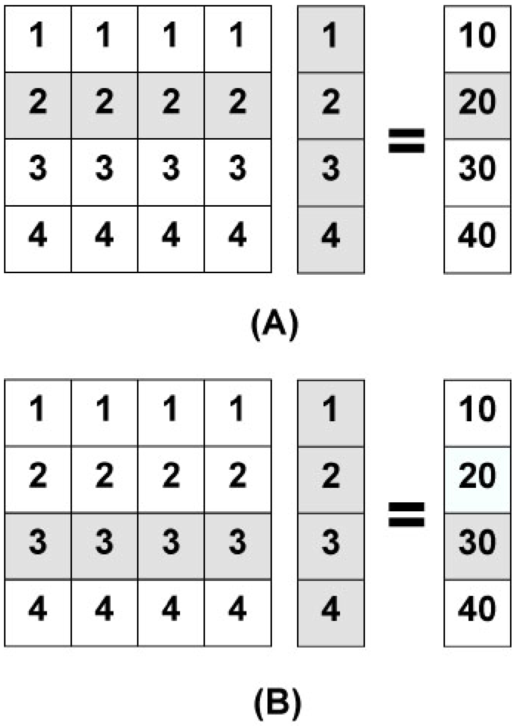

Figure 9-4. Recall as the premultiplication of a column vector by a matrix. (A) The second entry of the recall vector is the inner product of the second row of the matrix with the vector (see grey). (B) Similar logic defines the third entry of the recall vector.

For those unfamiliar with linear algebra, let us briefly examine the logic of Equation 9-6. When retrieving information from the distributed associative memory, the input units are activated, and send signals through weighted connections to the output units. The output units use these signals to compute their net input, which is also equal to their activation and to their output, as indicated in Equations 9-2 and 9-3. We saw earlier that the net input for a single output unit was the inner product between a vector of weights and a vector of activities. It stands to reason, then, that in order to compute the net input for several different output units, we will have to compute a series of different inner products.

The notation in Equation 9-6 represents performing a series of inner products. Each entry in the recall vector ***r*** is the inner product between the cue vector and one of the rows of the weight matrix. For example, the second entry in r is equal to the inner product between the second row of ***W*** and the column vector ***c*** (see Figure 9-4A). Similarly, the third entry in r is the inner product between the third row of ***W*** and the vector ***c*** (see Figure 9-4B).

This operation is consistent with the rule of thumb that we introduced earlier when discussing the inner product. The matrix ***W*** will have *m* rows, and the vector ***c*** has 1 column. So, we expect the result of Equation 9-6 to be a vector with *m* rows and 1 column – in other words, a column vector of the same size as ***c***.

9.2.2.4 Hebb-Style Learning

Up to this point, we have described how vectors are used to represent properties of processing units, how matrices are used to represent connection weights, how linear algebra provides a mathematical operation that uses a cue vector to retrieve a recall vector from a matrix of existing weights, and how associative learning can be described in generic terms as a series of sums of matrices. The only remaining piece of information required for a complete description of a distributed associative memory is a specific equation that defines how the desired weight changes are to be computed and stored in the matrix Δ_{t+1}. In this section, we will introduce one simple and historically important learning rule, called the Hebb rule. Later in this chapter, we will explore the Hebb rule's advantages and disadvantages, and use its disadvantages to motivate a second learning rule.

Donald Hebb (1904-1985) was one of the most influential figures in psychology (Klein, 1999). Born in Nova Scotia, Hebb graduated from Dalhousie University in 1925, and completed his Ph.D. at Harvard in 1936 after working with Lashley. Hebb's seminal contribution to psychology was his book *The Organization of Behavior: A Neuropsychological Theory* (Hebb, 1949). At the time that this book was published, physiological psychology was in decline because of the popularity of behaviorism. Hebb's book reversed this trend by attempting to explain behavior by appealing to properties of the nervous system. The book "wielded a kind of magic in the years after its appearance. It attracted many brilliant scientists into psychology, made McGill University a North American mecca for scientists interested in brain mechanisms of behavior, led to many important discoveries, and steered contemporary psychology onto a more fruitful path" (Klein, 1999, p. 2).

One of the central ideas that made Hebb's (1949) work so influential was the notion of a cell assembly. "The general idea is an old one, that any two cells or systems of cells that are repeatedly active at the same time will tend to become 'associated', so that activity in one facilitates activity in the other" (p. 70). The result of this kind of process is the creation of coordinated systems, or assemblies, of cells that act in sympathy with one another. Activity in one of the cells would lead to activity in the other cells that were part of the assembly. Hebb emphasized the utility of this kind of biological construct for explaining a variety of perceptual and motivational phenomena.

A crucial component of cell assembly theory was an account of how assemblies came into existence. Hebb (1949) is perhaps most famous for his statement of a principle of synaptic change for the creation of cell assemblies: "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased" (p. 62). Hebb believed that the mechanism underlying the change in the strength of the synapse between the two neurons was an increase in the area of contact between the two, but such hypotheses could not be tested at the time his work was published. Advances in neuroscience have led to a discovery of a phenomenon, called long-term potentiation, that is often cited as a biologically plausible instantiation of Hebb's theory (e.g., Brown, 1990; Martinez & Derrick, 1996).

In the late 1950s, the advent of digital computers enabled researchers to use simulations to explore the advantages and disadvantages of Hebb's (1949) theory of synaptic change. In one famous study, Rochester, Holland, Haibt, and Duda (1956) simulated a network of 69 simple neurons, with each neuron connected to 10 others. Rochester et al. updated connection weights using a modified version of Hebb's proposal. While the general spirit of the proposal was maintained, when weights were updated, they were normalized to prevent them from growing out of bounds. What this meant was that if the strength of one connection were increased, then the strength of other connections would be decreased at the same time. As well, Rochester et al. introduced the notion of "neural fatigue", which meant that one of their simulated neurons was less likely to fire if it had recently been active. After running this simulation, Rochester et al. examined the connection weights that emerged in an attempt to identify whatever cell assemblies had emerged. They found no evidence for the existence of cell assemblies in their simulation, and concluded that Hebb's theory as stated was not sufficient for their production.

Rochester et al. (1956) developed a second simulation using an unpublished modification of Hebb's theory that was proposed by Milner, and which later appeared in *Psychological Review* (Milner, 1957). In Hebb's original theory, and in Rochester et al.'s first simulation, there were no inhibitory connections. All of the connection weights (and all of the neural signals) in the simulation were positive and excitatory. Milner's proposal was to include inhibitory connections in the theory, under the assumption that there would be excitatory connections within a cell assembly, but activity in one cell assembly would tend to decrease activity in other cell assemblies via inhibitory signals. This proposal – endorsed by Hebb in a revision of his original theory (Hebb, 1959) – led to a simulation that did produce evidence of the emergence of cell assemblies.

In modern connectionist simulations, the goal of Hebb-style learning is not specifically to create cell assemblies, but is instead to create associations between patterns of activity, so that later when one pattern of activity is presented, the other pattern will be recalled. In other words, modern Hebb-style learning is one approach to defining “association by contiguity” or “the law of habit”. Nevertheless, inhibition is an important component of this type of learning, and is included in a distributed associative memory in two different ways.

First, and consistent with proposals described above, connections between processing units can either be excitatory (i.e., have a positive connection weight) or be inhibitory (i.e., have a negative connection weight). Second, and deviating from research in the 1950s, processing units can themselves be sending a signal that is excitatory (i.e., positive processing unit activity) or inhibitory (i.e., negative processing unit activity). In many respects, these assumptions violate Hebb’s attempt to develop a biologically plausible account of behavior. For instance, in the version of the distributed associative memory that we will develop below, at one moment a processing unit (or connection) can be excitatory, but at another moment the same unit (or connection) can be inhibitory. This kind of proposal is biologically implausible (Crick & Asanuma, 1986). However, it leads to a very simple mathematical description of Hebb-style learning, as we will see shortly.

As was noted above, Hebb's (1949) basic idea about learning was that if an input neuron and an output neuron were both active at the same time, then the synapse between them should be strengthened. “The assumption, in brief, is that a growth process accompanying synaptic activity makes the synapse more readily traversed” (p. 60). The logic of this proposal was that with the strengthening of the synapse, in situations in which the input neuron became active, there would be an increased likelihood of the output neuron becoming active as well. This is because the output neuron would receive increased stimulation (via the reinforced synapse) from the input neuron.

In modern variations of Hebb-style learning, particularly those based upon the assumption that processor activity can be either inhibitory or excitatory, the goal of connection weight changes is not to increase the likelihood of activity in an output unit. Rather, the goal is to change the weight in such a way that the relationship between input and output unit activities is enhanced. In other words, if at some learning trial an input unit is in one state *x*, and the output unit is in some other state *y*, then the connection weight should be changed so that later if the input unit returns to state x, then its signal through the connection should increase the likelihood of recreating state *y* in the output unit.

Hebb's (1949) view of learning is an example of enhancing one aspect of this relationship. To place his original proposal in the more modern context of a connectionist network, it was assumed that if an input unit and an output unit were both excited (positive activity), then the weight of the connection between them should be made more excitatory (i.e., more positive). Later, if the input unit exhibits positive activity, this would lead to a more positive signal (the positive activity multiplied by the more excitatory connection weight) being sent to the output unit, which would increase the net input to the output unit, and which would in turn increase the likelihood that the output unit would also exhibit positive activity.

Importantly, connection weights can be changed to enhance other relationships between input and output unit activities. For example, consider the situation where both an input unit and an output unit were inhibited (negative activity). To increase the probability that this pattern would occur later, one would again make the weight of the connection between them more excitatory. Later, if the input unit exhibits negative activity, this would lead to a more negative signal (the negative activity multiplied by the more excitatory connection weight) being sent to the output unit, which would decrease the net input to the output unit. As a result, the output unit would be more likely to assume negative activity. Similarly, imagine the situation in which the input unit was inhibited, but the output unit was excited. To increase the probability that this pattern would occur later, one would make the weight of the connection between the two units more *inhibitory*.

Later, if the input unit exhibits negative activity, this would lead to a more *positive* signal (the negative activity multiplied by the more inhibitory connection weight) being sent to the output unit, which would increase the net input to the output unit. As a result, the output unit would be more likely to assume positive activity. Similar logic would dictate that if the input unit was excited and the output unit was inhibited at the same time, then the connection between them should again be made more inhibitory. Table 9-2 summarizes the desired direction of weight changes given the possible states of connected input and output units.

Activity Of Input Unit	Activity Of Output Unit	Direction Of Desired Weight Change
Positive	Positive	Positive
Negative	Negative	Positive
Negative	Positive	Negative
Positive	Negative	Negative

Table 9-2. The direction of weight changes that will enhance the relationship between patterns of input and output unit activities.

What remains is to convert the qualitative account of desired weight changes that is given in Table 9-2 into a quantitative equation that will generate numbers that can be used to fill in the values of the matrix Δ_{t+1} during learning. An examination of the table provides a clear indication of the kind of mathematical operation to use. Note that if one were to take the value (i.e., the mathematical sign) of each of the first two columns and multiply them together, then the result would be the value in the third column of the table. In modern Hebb-style learning, the basic assumption is that the desired weight change for the connection between input unit *i* and output unit *j* is equal to the product of the activities of the two units:

$$\delta_{ij} = a_i \bullet a_i \qquad \text{(Equation 9-7)}$$

Equation 9-7 has two main advantages. First, under the assumption that unit activities can have negative or positive values, this equation creates weight changes of the desired sign according to Table 9-2. Second, this equation generates weight changes that reflect the relative amount of activity in both units. Imagine that the two processing units were both exhibiting positive activities, but that the two activities were very weak (e.g., values of, say, 0.05). It would seem plausible in this situation to not make a very large change to the connection weight. Equation 9-7 accomplishes this. For example, when two fractional positive values are multiplied together, as would be the case in our imagined situation, the resulting connection weight change is positive, but is also very small. Conversely, if both processing units were exhibiting very large activities, then it stands to reason that the connection between them should be changed a great deal. Again, Equation 9-7 automatically accomplishes this.

One minor modification to Equation 9-7 permits the exploration or manipulation of a richer notion of learning. One can imagine some situations in which a system is capable of learning a great deal, and other situations in which a system is less capable of learning. For instance, my kids are more likely to learn things in school when they are rested than when they are tired. In Hebb-style learning, such general effects can be modeled by using a learning rate, which is a constant used to scale the result of Equation 9-7 up. Traditionally, the Greek letter η represents the learning rate. When η is small or fractional, the desired weight changes will be small, which is analogous to the situation in which a tired child is trying to learn. When η is large, the desired weight changes will be amplified, which is analogous to the situation in which a rested child is trying to learn. This is all accomplished by multiplying the desired weight changes by the learning rate, as is indicated in Equation 9-8:

$$\delta_{ij} = \eta\,(a_i \bullet a_i\,) \qquad \text{(Equation 9-8)}$$

To bring this discussion to a close, Equation 9-8 defines how Hebb-style learning can be used to compute the desired change for a single weight in the distributed associative memory. Linear algebra provides a very compact notation for defining every entry in the matrix Δ_{t+1}. Remember the rule of thumb that claimed that the result of combining two vectors together had as many rows as the first vector in the combination, and had as many columns as the second vector. We used this rule of thumb to predict that when the inner product was computed (e.g., Equation 9-5) the result would be a number (i.e., a vector with one row and one column. Imagine we had two vectors, ***c*** and ***d***, and combined them in the reverse order than that used in Equation 9-5. In other words, what if they were multiplied together in an expression in which the transposed vector was the second component, instead of being the first: $\boldsymbol{d} \bullet \boldsymbol{c}^T$? Using our rule of thumb, we would not predict that we would get a single number. Instead, we would predict that the result of this equation would be a full matrix with as many rows as were in vector ***d***, and as many columns as were in $\boldsymbol{c}^T$. This matrix-producing operation is called the outer product.

The outer product is used to define how all of the desired weight changes for the distributed associative memory are to be calculated. Imagine that vector ***d*** represents some pattern of activity that has been presented to the output units of the memory, and that vector ***c*** represents some pattern of activity that has been presented to the input units of the memory. The desired weight changes are defined as:

$$\Delta_{t+1} = \eta\, (\boldsymbol{d} \bullet \boldsymbol{c}^T) \qquad \text{(Equation 9-9)}$$

The calculation of the outer product is illustrated in Figure 9-5. Every entry δ_{ij} in the matrix Δ_{t+1} is equal to the value c_i multiplied by the value d_j. This is the outer product. The result of this operation is then scaled by the learning rate, by multiplying it by the learning rate constant η.

9.2.3 Behavior Of The Distributed Associative Memory

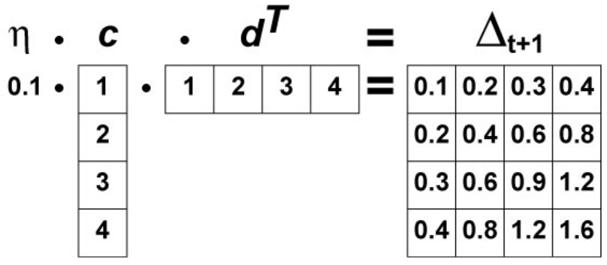

Figure 9-5. Using the outer product to define the desired weight changes in accordance with Hebb-style learning.

Now that Hebb-style learning has been defined using the outer product of two vectors, we are in a position to examine the behavior of this kind of memory system. This section does this in two ways. First, it uses what we have learned about linear algebra to explore the properties of this memory system. Second, it uses these equations to develop a computer simulation of a distributed associative memory that can be empirically explored.

9.2.3.1 Computational Account Of The Model

Dawson (1998) has argued that one of the key approaches taken by cognitive scientists to explain an information processing system is computational. In adopting the computational approach, one formally defines some characteristics of interest in a system (i.e., in some mathematical or logical notation). Then one uses formal operations to explore the properties of the system, typically by constructing mathematical or logical proofs.

One of the reasons that linear algebra was used to define the properties of the distributed associative memory in the previous sections was because it permits us to examine the system computationally. In particular, we can quickly manipulate the memory system's equations to generate proofs about its ability to function. We can also use the equations to determine whether there are some general situations in which it will fail to operate as intended.

As the first step in the computational analysis of a distributed associative memory governed by Hebb-style learning, let us make some simplifying assumptions. First, let us assume during learning that η has a value of 1. Because of this, it will be omitted from the learning equations. This is only being done to simplify the equations. The effect of different values of η will be explored in more detail in the simulation described in Section 9.2.3.2.

The second assumption involves the properties of the to-be-learned vectors that will be used in the equations below. Let us imagine that there are four of these vectors: ***a, b, c***, and ***d***. We will assume that this set of vectors is orthonormal. At a general level, what this assumption means is that each of these vectors has a length of 1.00, and is completely uncorrelated with the other three vectors in the set. Mathematically, this assumption involves assuming certain properties are true of the inner products of the vectors in this set. In particular, it is assumed that if one takes the inner product of a vector with itself, the result will be equal to 1. However, if the inner product is taken between a vector and a different member of the set, the result will be equal to 0. For example, this assumption means that $\boldsymbol{a^T \bullet a = 1}$, but that $\boldsymbol{a^T \bullet b = 0}$, $\boldsymbol{a^T \bullet c = 0}$, and $\boldsymbol{a^T \bullet d = 0}$. The importance of this second assumption will be apparent shortly.

Now let us define a simple learning sequence in which the distributed associative memory first learns the association between ***a*** and ***b*** by computing the outer product $\boldsymbol{b \bullet a^T}$ and then learns the association between ***c*** and ***d*** by computing the outer product $\boldsymbol{d \bullet c^T}$. This process of learning is detailed in Table 9-3, which is essentially the same as Table 9-1 with a few more specific details added because of our knowledge of which vectors are being learned at each trial:

Trial (t+1)	Operation	Equation Describing Weight Values
0	Start with the ***0*** matrix	$\boldsymbol{W_0 = 0}$
1	Associate ***a*** with ***b***	$\boldsymbol{W_1 = W_0 + \Delta_1}$ $\boldsymbol{= 0 + (b \bullet a^T)}$ $\boldsymbol{= (b \bullet a^T)}$
2	Associate ***c*** with ***d***	$\boldsymbol{W_2 = W_1 + \Delta_2}$ $\boldsymbol{= (b \bullet a^T) + \Delta_2}$ $\boldsymbol{= (b \bullet a^T) + (d \bullet c^T)}$

Table 9-2. Learning two pairs of vectors

Now that the distributed memory has learned two different associations, we can use linear algebra to predict its ability to recall remembered information. In this example, information is retrieved from the memory system is achieved by presenting either vector ***a*** or vector ***c*** as a cue and using the retrieval operation that was defined in Equation 9-6. If recall is correct, then when ***a*** is presented as a cue, the vector ***b*** should be retrieved; when ***c*** is the cue, ***d*** should be retrieved. Table 9-3 provides the mathematical details about recall from the memory. It takes Equation 9-6, and replaces the weight matrix with the more detailed expression for the weights that was provided in Table 9-2. It then works the cue vector into the parentheses. When this is done, two inner products are revealed. Because of our assumption that the set of vectors is orthonormal, one of the inner products works out to 0, canceling out a vector. The other inner product works out to 1. As a result, correct recall is achieved.

Cue	Recall Equation	Comments
a	$r = W_2a$ $= ((b \bullet a^T) + (d \bullet c^T))a$ $= b \bullet a^T \bullet a + d \bullet c^T \bullet a$ $= b \bullet (a^T \bullet a) + d \bullet (c^T \bullet a)$ $= b(1) + d(0)$ $= b$	Equation 9-6 Expand ***W₂*** from Table 9-2 Move vector ***a*** into the parentheses Identify the inner products with parentheses Compute inner products (orthonormal assumption) ***b*** is correctly recalled
c	$r = W_2c$ $= ((b \bullet a^T) + (d \bullet c^T))c$ $= b \bullet a^T \bullet c + d \bullet c^T \bullet c$ $= b \bullet (a^T \bullet c) + d \bullet (c^T \bullet c)$ $= b(0) + d(1)$ $= d$	Equation 9-6 Expand ***W₂*** from Table 9-2 Move vector ***c*** into the parentheses Identify the inner products with parentheses Compute inner products (orthonormal assumption) ***d*** is correctly recalled

Table 9-3. Correct recall of different associations from the same memory.

The equations that we have just been manipulating in Tables 9-2 and 9-3 make two important points. First, they have shown that when we make a particular assumption about the relationships between the patterns being associated, Hebb-style learning works. Furthermore, they show that this is accomplished with a single set of connections between processing units. The weight matrix ***W₂*** is a single entity, but from Table 9-3 it is clear that it holds information about the association between ***a*** and ***b*** and between ***c*** and ***d***. Second, these equations demonstrate a computational analysis (Dawson, 1998) of this kind of memory system. We have been able to use mathematics to demonstrate correct learning and recall; we did not need to program a simulation of this system to investigate these properties.

Computational analyses can also be used to demonstrate some of the problems with Hebb-style learning. The assumption that the set of to-be-associated vectors is orthonormal is extremely strong. What it amounts to is the claim that there can be absolutely no correlation between different patterns at all. If we were to be learning associations between entities in the world, then this assumption would be very limiting. For instance, in many cases we would expect there to be similarities or correlations between these objects. Indeed, one would expect – as did many of the associationists – that such correlations would be an important aid to memory.

Cue	Recall Equation	Comments
a	$r = W_2a$ $= ((b \bullet a^T) + (d \bullet c^T))a$ $= b \bullet a^T \bullet a + d \bullet c^T \bullet a$ $= b \bullet (a^T \bullet a) + d \bullet (c^T \bullet a)$ $= b(1) + d(½)$ $\neq b$	Equation 9-6 Expand ***W₂*** from Table 9-2 Move vector ***a*** into the parentheses Identify the inner products with parentheses Compute inner products ***b*** is not correctly recalled!
c	$r = W_2c$ $= ((b \bullet a^T) + (d \bullet c^T))c$ $= b \bullet a^T \bullet c + d \bullet c^T \bullet c$ $= b \bullet (a^T \bullet c) + d \bullet (c^T \bullet c)$ $= b(½) + d(1)$ $\neq d$	Equation 9-6 Expand ***W₂*** from Table 9-2 Move vector ***c*** into the parentheses Identify the inner products with parentheses Compute inner products ***d*** is not correctly recalled!

Table 9-4. Incorrect recall due to correlation between *c* and *a*.

To examine the effect of correlation on Hebb-style learning, let us make a slight modification to our orthonormality assumption. We will again be interested in learning associations between four different vectors, ***a***, ***b***, ***c***, and ***d***. We will assume once again that the inner product of any of these vectors with itself will result in a value of 1. We will also assume that ***a***, ***b***, and ***d*** are uncorrelated, so that the inner product of one of these vectors with one of the other two in this group of three will result in a value of 0. All of these assumptions were used in our previous analyses. Our change in assumptions will involve vector ***c***. We will assume that this vector is still not correlated with vectors ***b*** or ***d***, but that it does have a strong correlation with vector ***a***. In particular, we will assume that the inner product of ***c*** with ***a*** is equal to ½.

Table 9-4 provides the equations for recall with our change in assumption about the relationship between ***c*** and ***a***. In this case, because these two vectors are correlated, their inner product does not equal 0, and as a result does not completely cancel out part of the recall equation. As a result, there is noise or error added to the recall. Instead of recalling b when presented ***a*** as a cue, the memory recalls ***b*** plus some added noise: ***b + ½d***. Instead of recalling d when presented ***c*** as a cue, the memory recalls ***d*** plus some added noise: ***d + ½b***. The amount of noise that is evident in the recall is exactly equal to the correlation between ***c*** and ***a***. If this correlation were to increase, then the amount of noise in the recalled vectors would also increase. If this correlation were to decrease, then the amount of noise in the recalled vectors would also decrease. It is only when this correlation is equal to 0 that there is no noise and recall is perfect.

The linear algebra that we have just reviewed has shown that Hebb-style learning of associations has problems when the to-be-associated patterns are correlated with one another. This provides one strong suggestion that a different approach to learning associations should be considered if one is interested in training a distributed associative memory. In the next section, we will explore Hebb-style learning with a computer simulation in an attempt to identify some further problems. Later, these problems will lead to a reformulation of the rule that we use to modify connections in the memory system.

9.2.3.2 Observing The Behavior

The preceding sections have provided a mathematical description of a distributed associative memory, a formal definition of one method for storing associations in this memory, and mathematical proofs that show situations in which this memory works perfectly, as well as circumstances in which this memory does not function as well as desired. In this section, we will examine this same memory and learning rule, but instead of working with the system computationally, we will work with it algorithmically by observing the performance of a computer simulation.

Given the mathematical understanding of the memory that we have already achieved, one might wonder about the need for creating a computer simulation. However, a working computer simulation can quickly shed some light on practical issues that are not explicitly addressed in mathematical proofs. How fast is this type of learning when the memory is simulated on a digital computer? How is performance affected when the size of the memory grows? How does the learning rate affect performance? "Behavior is sometimes explainable in retrospect, but it is necessary to do the numerical experiments to see if ideas are actually workable, or if unforeseen problems appear. They often do. As only one example, there are a number of learning rules that can be proved to work mathematically. Unfortunately, when simulations are done, learning times are found to be enormous, totally outside the boundaries of practicality. Or the results are immensely sensitive to noise, or error, or to values of particular parameters" (Anderson & Rosenfeld, 1988, p. 65).

The simulation that was used to generate the results that are described below was programmed in Visual Basic 6.0 as an instructional tool to be used to explore a distributed associative memory. At the end of this chapter are directions for obtaining a free copy of this software, instructions for its use, and example training sets. This software can be used to perform all of the experiments that are described below, and will save the results in a variety of formats (text files, Microsoft Excel spreadsheets) for later exploration and analysis. The reader can also use the instructions to create their own training sets in a format that can be read by the network.

The first demonstration using this software involves testing the memory's performance when it learns the associations between a set of orthonormal vectors. I used Maple 5 to create a set of eight different vectors, with each vector having eight different entries. Each of these vectors was scaled to have a length of 1, and the set of vectors was constructed in such a way that the inner product of a vector with itself was equal to 1, and the inner product of a vector with any other vector in the set was equal to 0. This set of vectors is given in Table 9-5, and was used to create eight different stimulus-response pairs to be used to train the network. The pairs were vectors ***a*** and ***h***, ***b*** and ***g***, ***c*** and ***f***, ***d*** and **e**, ***e*** and ***d***, ***f*** and ***c***, ***g*** and ***b***, and **h** and ***a***.

Name	Values Of Each Vector							
a	0.27	0.39	-0.31	-0.16	0.64	-0.48	0.10	0.07
b	0.35	-0.22	0.06	-0.28	0.29	0.28	-0.47	-0.61
c	0.32	0.62	0.19	-0.08	-0.23	0.24	-0.47	0.38
d	-0.33	0.25	0.20	0.01	0.53	0.65	0.30	0.05
e	0.25	0.45	-0.01	-0.05	-0.37	0.06	0.54	-0.55
f	-0.04	-0.05	-0.83	-0.34	-0.16	0.38	0.01	0.15
g	-0.52	0.37	-0.29	0.43	-0.01	-0.10	-0.41	-0.38
h	0.51	-0.14	-0.23	0.77	0.11	0.24	0.07	0.08

Table 9-5. A set of eight orthonormal vectors used for training.

This set of eight stimulus-response pairs that were created by using this table were presented to the network under the following conditions. First, the learning rate η was set to the value of 0.10. Second, the network was trained for 10 epochs, at which time training stopped. Each epoch represents a "sweep" through each of the eight stimulus-response pairs. During one epoch, each pair was taken once, presented to the network (i.e., by presenting the cue pattern to the input units and the response pattern to the output units), and the weights were modified according to the Hebb rule. Thus at the end of training in this experiment, each pair had been presented to the network ten different times.

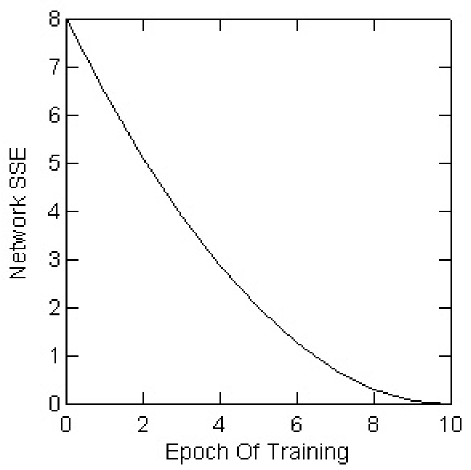

Figure 9-7. Network error as a function of training in Simulation 1

In order to examine the performance of the network, a measure of total network error was computed at the completion of every training epoch. This was done as follows: First, each cue member of a stimulus-response pair was presented to the input units of the network, and the network's response to this cue was computed using the recall equation. Second, the resulting network response was compared to the correct response. This was accomplished by subtracting the actual activity of each output unit from the correct or desired activity. This difference was then squared to remove any resulting negative signs. The squared values obtained for each of the eight output units were summed together for each of the patterns, and then these eight sums were themselves summed together to create a single measure. This measure is the sum of squared error (SSE), and provides an index of network error summing across all output units and all training patterns. If the network were correctly recalling all of the associates, then the SSE would have a value of 0.

Figure 9-7 shows the SSE that was computed for the network after each epoch of training in Simulation 1. This figure demonstrates a systematic decline in network error with each sweep through the training patterns, and shows that SSE reached a value of 0 at the end of the 10th epoch. It would be expected that 10 presentations of each pair would be required to generate perfect recall in this experiment, because the learning rate η was set at 0.10. In this case, the behavior of the network is completely consistent with our expectations from the mathematical treatment that was presented earlier.

	Input 1	**Input 2**	**Input 3**	**Input 4**	**Input 5**	**Input 6**	**Input 7**	**Input 8**
Output 1	-0.28	0.28	-0.57	0.34	0.42	-0.11	0.09	0.46
Output 2	0.28	-0.10	-0.40	-0.09	0.12	0.81	0.23	-0.16
Output 3	-0.57	-0.40	-0.22	-0.10	-0.18	-0.16	0.56	-0.28
Output 4	0.34	-0.09	-0.10	-0.42	0.66	-0.41	0.13	-0.27
Output 5	0.42	0.12	-0.18	0.66	-0.19	-0.26	0.18	-0.46
Output 6	-0.11	0.81	-0.16	-0.41	-0.26	-0.03	0.11	-0.24
Output 7	0.09	0.23	0.56	0.13	0.18	0.11	0.71	0.24
Output 8	0.46	-0.16	-0.28	-0.27	-0.46	-0.24	0.24	0.53

Table 9-6. The final weights from the network trained in the first study.

Table 9-7 provides the connectionist weights that are found in the network at the end of Simulation 1. An examination of this set of weights indicates that the associative memory is indeed distributed. We know from the recall performance of the network that these weights are storing information about eight different associations. However, in looking at these weights, we do not see any evidence that these associations are stored locally. For instance, it does not appear that one row of the weight matrix stores information about one association, and that another row stores information about a different association. All of the weights have been affected by training, and information about all eight associations is distributed throughout the entire weight matrix.

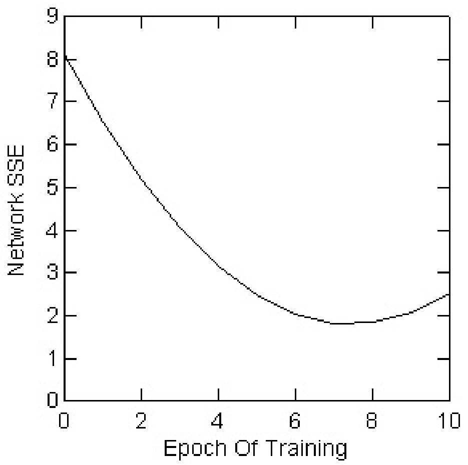

Figure 9-8. Network error as a function of training in Simulaton 2. SSE never reaches 0.

Earlier, we saw that our computational analyses demonstrated that one problem with Hebb learning was that it failed to produce perfect results when the patterns being associated did not conform to the orthonormality assumption. When this assumption is violated, this means that some of the vectors in the training set are similar to, or correlated with, others. To examine this kind of problem with the computer simulation, a new training set was constructed. It used the first seven vectors that were listed above in Table 9-5. However, vector ***h*** was replaced with a new vector. This vector was created by taking the first 4 entries from vector ***a***, and using them as the first four entries in the new vector ***h***. Then the last 4 entries from vector ***b*** were taken, and used as the last 4 entries in vector ***h***. As a result of this manipulation, vector ***h*** had substantially high correlations with vectors ***a*** and ***b*** (0.36 and 0.78 respectively). These correlations represent a violation of the orthonormality assumption.

In Simulation 2, this new table of vectors was used to create a set of eight stimulus-response pairs in accordance with the procedure that was used in the first simulation. The network was then trained on these eight associates for 10 epochs, using the exactly the same method that was described earlier. Figure 9-8 illustrates the performance of the network, and confirms our expectations that correlations between vectors disrupt Hebb learning. The figure provides a graph of network SSE as a function of training. As can be seen from this graph, SSE has not reached a value of 0 during this simulation. This shows that recall was not perfect.

Another important way in which correlation can be built into a training set is to make the training set linearly dependent. This means that one vector in the set is equal to a weighted sum of other vectors in the set. To illustrate the effect of this on Hebb learning, a linearly dependent training set was created. This was done by defining a new vector ***h*** as being equal to ½ (***a*** + ***b***). That is, each entry of vectors ***a*** and ***b*** were added together and then divided by 2 to produce a new vector that was the average of vectors ***a*** and ***b***. This new vector ***h*** is obviously correlated with both vectors ***a*** and ***b*** because it was constructed from them. Simulation 3 was conducted with this linearly dependent training set, using the same methodology that was employed in the first two studies. Figure 9-9 illustrates the performance of the network, and shows again that the network was not able to learn this training set properly.

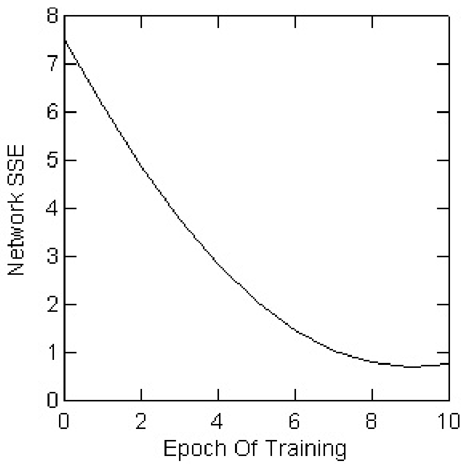

Figure 9-9. Network error as a function of training in Simulation 3.

A more detailed analysis of the kinds of errors made by the network in Simulation 3 clearly show that its performance is being affected by the linear dependence that was built into the training set. Table 9-7 provides the errors (desired activity – observed activity) made by each of the output units to each of the 8 cue patterns. As can be seen from the table, the network's errors in recall are quite selective. Indeed, for five of the eight cues, the network's recall is perfect, because there is no error in any of the output units. The only time that error is observed is when ***a***, ***b***, or ***h*** is used as a cue. For two of these situations, the error is fairly small. For the third situation, when the cue is a blend between two other cues that the network has already seen, there is substantial error in all of the output units.

Stimulus		Error Calculated For Each Output Unit							
Cue	**Response**	**OUT 1**	**OUT 2**	**OUT 3**	**OUT 4**	**OUT 5**	**OUT 6**	**OUT 7**	**OUT 8**
a	***h***	-0.14	-0.20	0.15	0.08	-0.32	0.24	-0.05	-0.03
b	***g***	-0.14	-0.20	0.15	0.08	-0.32	0.24	-0.05	-0.03
c	***f***	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
d	***e***	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
e	***d***	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
f	***c***	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
g	***b***	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
h	***a***	0.24	-0.03	0.05	-0.19	0.09	-0.14	0.35	0.36

Table 9-7. Network errors from the third simulation study.

A closer examination of Figures 9-8 and 9-9 reveals a second counterintuitive property of Hebb learning. In both figures, it appears that SSE has reached a minimum value before the 10th sweep of training, and is actually rising as more training is conducted. Our general sense about learning is that this shouldn't be happening. We would normally expect that more learning should result in better performance. So, if learning is operating the way that we would expect, then SSE should not increase. Is the apparent increase in the two figures real, or is it a minor aberration? If it is real, is it due to Hebb learning in general, or is it a more specific result of violating the orthonormality assumption?

To explore these questions in more detail, Simulation 4 was carried out. This simulation was identical to Simulation 1, with the exception that instead of training the network on the orthonormal patterns for 10 epochs, it was trained for 30 epochs. The results of Simulation 4 are provided in Figure 9-10. As can be seen in this figure, after network SSE drops to 0 following the 10th epoch of training, SSE begins to rise, and is actually quite large at the end of the 30th epoch. At the end of this simulation, total network error is over three times the size that it was before any learning had started at all. This unfortunate finding is due to the fact that Hebb learning modifies weights after each stimulus presentation, even when the weights should not be changed. In other words, Hebb learning does not use any feedback about the errors that the network is making. If it did, then this would prevent it from making unnecessary changes to its weights, and from undoing the correct learning that it has already accomplished.

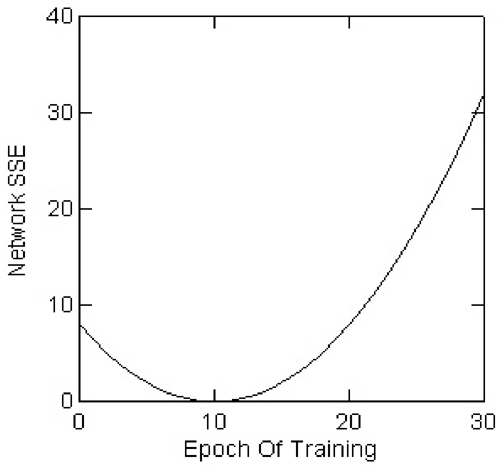

Figure 9-10. Network SSE in Simulation 4.

9.3 Beyond The Limitations Of Hebb Learning

The previous section used computational analyses and computer simulations to examine some of the properties of a distributed associative memory that uses the Hebb rule to learn associations between presented pairs of stimuli. In this section, we will quickly review some of the drawbacks of this type of system. We will then use these drawbacks to motivate discussion of a second learning rule. The goal of this new rule is to train the memory in such a way that some of the problems that we have identified are solved.

9.3.1 The Limitations Of Hebb Learning

There are three general reasons that the Hebb learning rule has enjoyed a great deal of popularity amongst researchers who are interested in developing theories of associative memory. First, we saw in the historical review of associationism that one of the constants from one theory to the next was the inclusion of the law of contiguity. The Hebb rule is an elegant statement of this fundamental mode of association. Second, in modern cognitive science there is an increasing desire to relate properties of functional theories to neural mechanisms (Dawson, 1998). The Hebb rule is one of the few connectionist learning rules that seems to be biologically plausible. Many researchers have taken pains to point out the similarities between Hebb's account of learning and the biological mechanisms that govern long-term potentiation in the brain (Brown, 1990; Cotman, Monaghan, & Ganong, 1988; Martinez & Derrick, 1996). Third, even when memory systems trained by Hebb-style learning rules make mistakes, these mistakes are interesting, because in many cases they are analogous to the kinds of errors that one finds in human experiments on associative learning (Eich, 1982; Murdock, 1982, 1997).

In spite of these attractions, the theoretical and empirical evidence that we have collected earlier in this chapter points to some severe limitations of a distributed associative memory that is trained by the Hebb rule. First, the memory only works well when the stimuli being associated are completely uncorrelated. As soon as the orthonormality assumption is violated, one cannot guarantee that the memory will recall the correct response when given a cue. Second, the Hebb rule is not sensitive to the performance of the memory system. This means that the Hebb rule will modify network connections even in situations where these modifications are not required because perfect recall has been achieved.

9.3.2 Overcoming The Limitations

The combination of these problems with Hebb learning and the general attractiveness of this learning rule suggests that we should attempt to explore some ways in which the rule can be improved without throwing away many of its attractive properties. The purpose of this section is to describe such a refinement, and to define a new rule called the delta rule. We will see that the delta rule ultimately relies on association by contiguity, and therefore maintains many of the essential properties of the Hebb rule. However, the delta rule is explicitly designed to teach a network by providing it feedback about the kinds of errors that it makes. As a result, the delta rule provides one approach to overcoming some of the limitations of Hebb learning that we have already encountered.

9.3.2.1 Supervised Learning

In connectionist research, a common distinction is made between unsupervised learning and supervised learning. In unsupervised learning, a network modifies its connection weights in an attempt to remember regularities that it has discovered in its environment. However, it never receives any information about what some programmer might think are desirable regularities. It therefore also never receives any feedback about whether its responses are correct or incorrect. In this regard, Hebb learning is an example of unsupervised learning. The fact that Hebb learning does not take into account errors that are being made by a network accounts for problems like the increase in network SSE that was illustrated in Figure 9-9.

In supervised learning, the goal of learning is for a network to generate a set of responses that are desired by a programmer (or a teacher). When the network generates a response to a stimulus, this observed response is compared to a desired response, which is often called the target response. Typically, one compares these two responses by subtracting the observed response (O) from the target response (T) for each output unit in the network. That is, the error for output unit i (ε_i) is:

$\varepsilon_l = T_i - O_i$ (Equation 9-10)

One of the advantages of supervised learning is that learning is only driven by mistakes. This implies two different things. First, if no mistake is made, then no learning will occur, because no learning is required. Second, the degree of learning should be proportional to the degree of error. If a system makes a very large error, then there should be very large changes to its connection weights. However, if a system makes a very small error, then there should be a correspondingly small change to its connection weights. If we could replace the Hebb rule with a supervised learning rule that operated in this fashion, then we would definitely be in a position to solve one of the problems with Hebb learning that we have already identified. To be more specific, if our distributed associative memory was supervised when it learned, then we would not observe the problem that was illustrated in Figure 9-9, because once total SSE had dropped to 0, no more connection weight changes would occur.

The question is how to reformulate the Hebb rule in such a way that it can be converted from an unsupervised learning rule to a supervised learning rule. As a first pass at the logic of this reformulation, consider Table 9-8, which is a variation of Table 9-2. The purpose of Table 9-8 is to consider the activity in a single input unit, treating it for the sake of simplicity as being merely positive or negative. This input unit is connected to a single output unit, and the error for this unit has been calculated according to equation 9-10 after some pattern has been presented to the network. Again, for simplicity's sake, we consider the result of this calculation to be a value that is positive, negative, or equal to zero. The table lays out the possible combinations of input values and error values in order to make clear what would need to happen to the weight of the connection between the two units in order to reduce the error that was produced the next time that the pattern was presented to the network.

Activity Of Input Unit	T - O	Implication	Operation To Reduce Error	Direction Of Desired Weight Change
Positive	Positive	T > O	↑ O	Positive
Positive	Negative	T < O	↓ O	Negative
Positive	Zero	T = O	None	Zero
Negative	Positive	T > O	↑ O	Negative
Negative	Negative	T < O	↓ O	Positive
Negative	Zero	T = O	None	Zero

Table 9-8. The logic of weight changes during supervised learning. T represents the target value for an output unit, and O represents the observed value for the output unit. See text for further details.

For example, consider the first three rows of the table, for which the input unit has been activated with some positive value. In the first case, the error value is positive. This means that the target activity is greater than the observed activity. In order to reduce error, this means that the observed activity must be increased. For this pattern, this could be accomplished by making the connection weight more positive, because this would amplify the positive signal being sent by the input unit. In the second case, the target activity is smaller than the observed activity, which means that the observed activity has to be made smaller to reduce error. This would be accomplished by making the connection weight more negative, because this would attenuate the positive signal being sent by the input unit. In the third case, the target activity is equal to the observed activity, which indicates that no change should be made at all to the connection weight.

Similar logic can be followed for the remaining three rows in the table. However, because in these instances the input unit activity is negative, the change to the connection weight will be opposite in direction to the changes that were just described. In the first case, the connection weight must be made more negative in order to amplify (i.e., make more positive) the negative signal being sent by the input unit. In the second case, the connection weight must be made more positive in order to attenuate (i.e., make more negative) the negative signal coming from the input unit. Of course, in the third case there again would be no change made to the connection weight because there is zero error being generated by the output unit.

Earlier in this chapter, we motivated the rule for Hebb-style learning by observing that if we multiplied the first two columns of Table 9-2 together, the result would be the third column. A similar situation now arises in our discussion of supervised learning. If one were to take the first two columns of Table 9-8 and multiply them together, the result would be the last column of the table, which indicates the direction of weight change to make in order to reduce error. This inspires the following learning rule for a single connection between input unit i and output unit j:

$$\delta_{ij} = a_i\ (T_j - a_i) \qquad \text{(Equation 9-11)}$$

where δ_{ij} is the desired weight change, is the a_i activity of the input unit, T_j is the target activity for the output unit, and a_i is the observed activity in the output unit.

Equation 9-11 has two very nice properties that suggest that it is an excellent choice for a supervised learning rule for connections in a distributed associative memory. First, the equation changes the weight in the direction that is required to reduce error, because the equation is consistent with the logic that we worked through when discussing Table 9-8. Second, it is sensitive to amount of error. If the value of T_j - a_i is large, then the change in the weight will be large. If the value is small, then the change in the weight will be small. If the value is zero, then – crucially – there will be no change in weight. This equation places a natural brake on the learning process, solving one of the problems that we identified with the Hebb rule.

9.3.2.2 The Delta Rule

The final step in defining a supervised learning rule for the distributed associative memory is to take Equation 9-11 and modify it by including a learning rate, and by expressing it in terms of linear algebra so that we can use one equation to define the changes for all the weights in a network that consists of multiple input and output units. When we defined the Hebb rule, we used the outer product of two vectors – scaled by a learning rate – to define the matrix of weight changes Δ_{t+1}. We can also follow this procedure for defining our supervised learning rule, which is called the delta rule. Let us assume that vector $\boldsymbol{c}^T$ represents some pattern of activity that has been presented to the input units of the memory. Let us also assume that the vector ***t*** (for target) defines the vector that *should* be correctly recalled from the memory when ***c*** is used as the cue in Equation 9-6. Let vector ***o*** (for observed) be the actual activity that is generated in the output units when ***c*** is the cue. The desired weight changes, scaled by the learning rate η, are defined as:

$$\Delta_{t+1} = \eta\ ((\boldsymbol{t} - \boldsymbol{o}) \bullet \boldsymbol{c}^T) \qquad \text{(Equation 9-12)}$$

The expression ***t*** – ***o*** in Equation 9-12 is the difference between two vectors. The result of this operation will be another vector, with the same number of entries that would be found in either vector ***t*** or vector ***o***. Let us name this third vector ε, to represent the fact that it is a vector of error values. Consistent with our definition of error in Equation 9-10, each entry ε_i in this vector is equal to the value ($t_i - o_i$). With this definition of the error vector, we can rewrite Equation 9-12 as:

$$\Delta_{t+1} = \eta\ (\varepsilon \bullet \boldsymbol{c}^T) \qquad \text{(Equation 9-13)}$$

Equation 9-13 is important in that it makes very explicit the relationship between the delta rule and the Hebb rule. If you compare it to Equation 9-9, you will see that the two learning rules are very similar. The delta rule essentially involves Hebb learning, but this learning is not carried out until after a couple of preliminary steps have been taken. First, a cue vector is presented to the existing memory to see what vector would be recalled from the memory if no weight changes were made at all. Second, an error vector is computed by subtracting this observed recall vector from the target vector. Third, Hebb learning is performed, but the association that is learned is between the cue vector and the error vector. The point of doing this – as was explained in our discussion of Table 9-8 – is to only make changes in weights that are necessary to reduce error. If the error vector is full of zeroes, then no weight changes will be made when Equation 9-13 (or 9-12) is applied.

9.3.2.3 The Power Of The Delta Rule

In this section, we are going to examine the performance of the delta rule, and compare it to the performance of Hebb learning, by repeating the four simulation experiments that were described earlier. It is also possible to compare the two rules by doing a computational analysis of the delta rule, and comparing the conclusions drawn from that analysis to those that we drew after working through the proofs in Tables 9-3 and 9-4. While this isn't done in the current chapter, mathematical treatments of the delta rule are available in the literature. A particularly good one is provided by Stone (1986).

Because the delta rule is error correcting, instead of training the associative network for a set number of iterations, in each simulation we will train the network until SSE reaches a suitably low value. In these studies that follow, a value of 0.01 SSE was chosen as the criterion to use to stop training, because this level of SSE is small enough for us to say that the network has correctly learned the task. (If this low level of SSE was not reached after 5000 epochs, then training was also stopped, under the assumption that this was sufficient time to learn a small set of associates. If SSE had not dropped to below the criterion after this amount of training, then it was likely that the network could not learn the associations). In order to enable a comparison between the two learning rules, all other aspects of training (learning rate, stimuli) were identical to those used in the previous four simulations.

Simulation 5 is our first test of the delta rule. In it, the delta rule was used to train the distributed memory on the associations between the orthonormal set of patterns, using the same stimulus set that was used to collect the data for Figure 9-7. The network converged after 34 epochs, at which time network SSE was equal to 0.0094, indicating that the network's performance was nearly perfect. This shows that the delta rule is indeed capable of training associations, but with this particular learning rate, the training is slower than we saw with the Hebb rule. However, if training is continued, the delta rule continues to improve performance. Figure 9-11 illustrates network SSE during the course of 50 epochs of training. At the end of 50 epochs, the delta rule has reduced the network's error to 0.0002. In other words, this particular memory system is performing in a fashion that is in more accordance with our intuitions: when the memory has more repetitions on the paired associates, its performance improves.

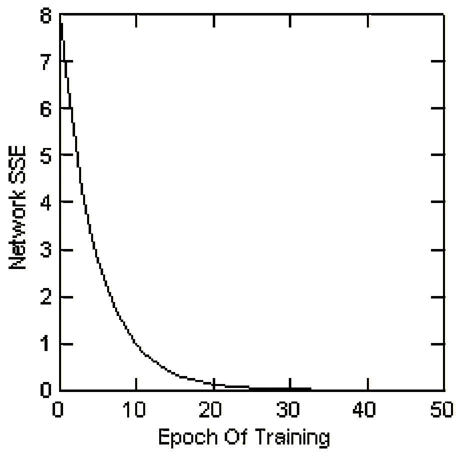

Figure 9-11. Network SSE when the delta rule is used in Simulation 5.

Figure 9-11 illustrates an important emergent property of this kind of learning. Notice how network SSE decreases exponentially, with a great deal of learning occurring early in training, but with learning slowing down as training proceeds. This is to be expected because the amount of learning depends upon the amount of error that the network is making (see Equation 9-12). As the network learns more, its error is reduced, and as a result learning slows down. We saw this pattern earlier in Chapter 4 when we discussed mathematical models of learning in general, and the Rescorla-Wagner learning rule (Rescorla & Wagner, 1972) in particular. Figures 4-1 and 4-2 illustrate the dynamics of Rescorla-Wagner learning, and show how it decelerates over time. This is because learning is driven in that model by the difference between the current associative strength and the maximum possible strength – as the two strengths become more similar, learning slows down. The only reason that Figures 4-1 and 4-2 move in the opposite direction of Figure 9-11 is because the former two figures plot learning, while the latter plots error. One of the important findings that demonstrated a strong relationship between connectionism and mathematical models in psychology was a proof that showed that learning rules like the delta rule are indeed equivalent to the Rescorla-Wagner rule (Sutton & Barto, 1981).

Figure 9-11 demonstrates that the dynamics of delta rule learning are quite different from those of Hebb rule learning. Because of this, does the network learn different regularities depending on what learning rule is used? Table 9-9 provides the connection weights in the associative memory from simulation five at the end of 50 epochs of training. A comparison of this table to Table 9-6 indicates that the connection weights from the two simulations are nearly identical. Any minor discrepancies between the two probably account for some of the tiny errors that are made by the network trained by the delta rule, whose SSE is not perfectly equal to zero. With further delta rule training, we would expect that the two networks would have learned exactly the same kind of information, at least for stimulus-response pairs that had been taken from a set of orthonormal vectors.

	Input 1	**Input 2**	**Input 3**	**Input 4**	**Input 5**	**Input 6**	**Input 7**	**Input 8**
Output 1	-0.27	0.27	-0.57	0.34	0.41	-0.11	0.09	0.46
Output 2	0.27	-0.10	-0.39	-0.09	0.12	0.81	0.23	-0.16
Output 3	-0.57	-0.39	-0.22	-0.10	-0.18	-0.16	0.56	-0.28
Output 4	0.34	-0.09	-0.10	-0.42	0.65	-0.41	0.13	-0.27
Output 5	0.41	0.12	-0.18	0.65	-0.19	-0.26	0.18	-0.46
Output 6	-0.11	0.81	-0.16	-0.41	-0.26	-0.03	0.11	-0.24
Output 7	0.09	0.23	0.56	0.13	0.18	0.11	0.70	0.24
Output 8	0.46	-0.16	-0.28	-0.27	-0.46	-0.24	0.24	0.53

Table 9-9. The final weights from the network trained in the fifth study.

This is not to say that the two learning rules will lead to exactly the same connection weights in all situations, however. One of the interesting properties of the delta rule is that it is more powerful than Hebb learning. Because the rule works explicitly to reduce output unit error, it turns out that there are some associations that can be stored in a network using the delta rule, but which cannot be stored if the network is trained using the Hebb rule. To illustrate this, Simulation 6 was conducted. In this case, the training set was the same as that used in simulation two, where a new vector ***h*** was created by taking the first half of vector ***a*** and the second half of vector ***b***. We saw in the second simulation that this prevented the Hebb rule from learning; it reached a minimum SSE of 1.82 after 7 epochs, and then SSE began to grow again, reaching a value of 2.54 after 10 sweeps of training. The question of interest is whether delta rule learning leads to any better performance than this.

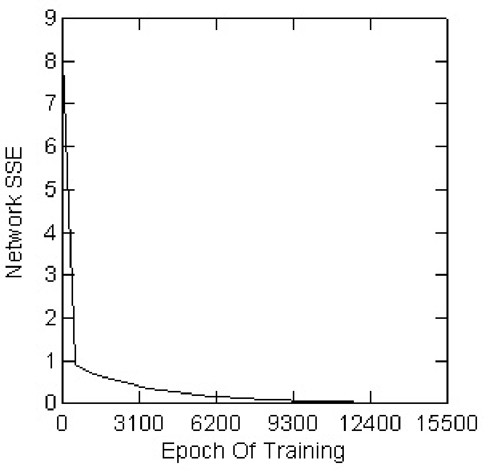

Figure 9-12. Network SSE in Simulation 6. Note that error drops to near zero, which was not the case in Figure 9-8.

When this set of associations is trained using the delta rule, the network is able to learn the problem.

After 15,503 iterations, network SSE has dropped below 0.01, and the training has stopped. With this small level of total error, network performance is near perfect for all eight stimulus-response pairs. The course of learning is illustrated in Figure 9-12. From this figure, it would appear that two stages of learning are evident. In the first stage, there is an extremely rapid drop in overall SSE. In the second stage – which starts at the sharp elbow on the left side of the graph – there is a much more gradual decrease in error. It is likely that this more complicated pattern of error change is the result of dealing with two different types of patterns. In the first instance, much of the sudden drop in error is likely due to the ability of the network to quickly learn the associations among the stimulus-response pairs that have not been affected by the change in vector h. In the second instance, the network is slowly adapting itself to deal with the more problematic instances.

It should be pointed out that while the learning depicted in Figure 9-12 was very slow, the delta rule is not necessarily limited to this kind of performance. For example, if one trains a network on exactly the same set of associations, but sets the learning rate at 0.75, then the network will converge on a solution (i.e., reach an SSE of less than 0.01) after only 775 epochs. Clearly the speed of learning is markedly affected by the choice of learning rate.

How is it possible for the delta rule to come up with a set of connection weights that can store these eight associates, while this was not possible when the Hebb rule was used to train the network? One empirical clue to this additional power comes from examining the connection weights in the network at the end of training. The weights are provided in Table 9-10. For simpler problems (i.e., learning associations involving orthonormal vectors), the matrices of connection weights that resulted were symmetric (see Tables 9-6 or 9-9). This means that the value in cell wij is the same as the value found in cell w_{ji}. If one examines Table 9-10, however, it is clear that this matrix is not symmetric at all. The ability of the delta rule to create a set of connection weights that are not symmetric means that it can store a wider range of associations than can be learned via the Hebb rule. This is because the Hebb rule will always produce a symmetric set of connection weights. Because the delta rule is not limited to producing symmetric weights, it is more powerful – a fact that we have already demonstrated by showing that the delta rule learned the set of associations in Simulation 6, while the Hebb rule was unable to learn the same set of associations in Simulation 2.

	Input 1	**Input 2**	**Input 3**	**Input 4**	**Input 5**	**Input 6**	**Input 7**	**Input 8**
Output 1	3.23	-0.75	-2.1	5.71	1.01	1.7	0.56	0.97
Output 2	-1.22	0.53	0.18	-2.62	0.11	-0.22	0.06	-0.37
Output 3	0.56	-0.73	-0.71	1.63	0.01	0.42	0.72	-0.12
Output 4	-1.68	0.01	0.97	-2.91	-0.3	-0.8	-0.21	-0.59
Output 5	3.82	-0.69	-1.74	5.63	0.63	1.24	0.67	0.04
Output 6	-7.8	2.84	3.28	-11.91	-1.85	-3.69	-0.96	-1.37
Output 7	6.19	-1.61	-2.07	9.53	1.15	3.33	1.52	1.12
Output 8	6.43	-2.04	-2.83	9.03	0.39	3.01	1.03	1.39

Table 9-10. Connection weights from the sixth simulation.

This is not to say that the delta rule is all-powerful, however. Computational analyses of this rule have demonstrated that it permits associations to be learned when some correlations exist between vectors, but it is unable to learn associations when other correlations exist. In particular, the delta rule is not capable of correctly recalling associations when the training set is linearly dependent. To demonstrate this point, a seventh simulation was run in which the delta rule was trained on the linearly dependent patterns that were created for simulation 3. This simulation was run for 20,000 epochs. However, after all of this training, the network had not converged upon a solution. By the end of the first 1000 sweeps of training, total error had dropped to about 0.544. Further training did not lead to any noticeable improvement. (However, further training also did not lead to the network performing any poorer, which demonstrates again one advantage of the delta rule over the Hebb rule.)

Stimulus		Error Calculated For Each Output Unit							
Cue	Response	OUT 1	OUT 2	OUT 3	OUT 4	OUT 5	OUT 6	OUT 7	OUT 8
a	***h***	-0.13	-0.06	0.04	0.09	-0.14	0.13	-0.14	-0.14
b	***g***	-0.13	-0.06	0.04	0.09	-0.14	0.13	-0.14	-0.14
c	***f***	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
d	***e***	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
e	***d***	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
f	***c***	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
g	***b***	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
h	***a***	0.25	0.11	-0.07	-0.17	0.27	-0.25	0.26	0.26

Table 9-11. Network errors from the seventh simulation.

However, even this inability to learn to deal with the linearly dependent patterns did not lead to a complete breakdown in learning. Table 9-11 provides the error calculated for each output unit and each stimulus at the end of training. As was the case for Hebb learning, perfect recall was observed for 5 of the associates (compare this table with Table 9-6). The network only had difficulty learning the associates that used ***a***, ***b***, or ***h*** as the cue. This was exactly the case for Hebb learning. The only apparent difference between the two error tables is that the delta rule led to smaller values of error whenever mistakes were being made.

9.4 Associative Memory And Synthetic Psychology

9.4.1 Summary Of Key Points

This chapter has explored the building of associations between patterns as a first step in introducing some of the essential features of connectionist modeling. The chapter began with an historical overview of associationism. This overview revealed two repeating themes. First, associationists were concerned with a particular mental regularity, sequences of thought. They were struck by the compelling fact that one idea often calls to mind another. Second, associationists were interested in using laws of association to explain mental sequences. One law that seems to appear in every theory of associationism is the law of contiguity, which essentially says that if two ideas occur close together in time, then the association between them will become stronger.

The historical overview of associationism led to a description of a particular connectionist network, called a distributed associative memory. This memory is associative in the sense that when two patterns are presented to it, it stores an association between them. Later, the presentation of one of the patterns is intended to lead to the retrieval of the other from the memory. This memory is distributed in the sense that it uses one set of connection weights to store information about many different associations, and this stored information is distributed throughout the entire set of weights.

One of the main goals of describing the distributed associative memory was to introduce some of the basic concepts of the connectionist architecture. Three different categories of connectionist concepts were introduced. The first concept was the processing unit, defined by a net input function, an activation function, and an output function. The second concept was the modifiable connection, whose weight is used to scale a numerical signal that is sent through it. The third concept was the learning rule, which is used to define how connection weights are modified to store the associations that are experienced by the network. Most of these concepts were framed in the context of linear algebra. For instance, we showed how the inner product of two vectors can be used to define a processor's net input function, how the outer product of two vectors can be used to define a matrix of desired weight changes, and how the premultiplication of a vector by a matrix can be used to define how information is retrieved from a distributed associative memory.

Two different learning rules were defined for this particular memory system. The first was the Hebb rule, which defined associations in terms of contiguity of processor activities. Mathematical analyses of Hebb-style learning indicated that it worked quite well in a situation in which the entire set of patterns being associated were uncorrelated. However, correlations between training patterns were shown to produce systematic errors. These two conclusions were also demonstrated with a computer simulation. The simulation also showed that an additional problem with Hebb learning is that it is not error correcting, and as a result makes unwanted changes to connection weights when more training is conducted than is required.

The second learning rule that was defined was the delta rule, which was explicitly designed to solve this last problem with Hebb learning. The delta rule uses feedback about processing unit errors to set connection weights. As a result, learning is driven by the size of error, and when error is reduced to zero, learning stops. A computer simulation was used to demonstrate this property of the delta rule. It was also used to demonstrate that the delta rule is more powerful than the Hebb rule, in that it was able to learn associations between patterns with some correlations between them. The Hebb rule was unable to do this. However, some correlations between patterns still presented problems to the delta rule. In particular, sets of vectors that are linearly dependent still cause systematic problems for delta rule learning.

9.4.2 Implications For Synthetic Psychology

The primary goal of describing the distributed associative memory in this chapter was to introduce some basic notions about connectionist models. We will see in the chapters that follow that many advances in connectionist architectures can be described as "tweaks" or elaborations of some of the concepts that were introduced in this chapter. However, it is important to realize that distributed associative memories are interesting in their own light, and can be used to synthetically explore some issues in the psychology of learning and memory. Indeed, the software that was used to conduct the simulations that were described earlier, and which is freely available (see below), might be used by the reader to explore issues that have not been covered at all in this chapter.

Associative learning is still a fundamental topic in psychology and cognitive science, and there is a variety of research streams that are worthy of consideration, and of possible future exploration with computer simulations.

One key area of research is the study of associative learning in animals. Throughout the history of this topic, the underlying assumption has been that the discovery of elementary associative laws that govern animal learning can be used to aid in the understanding of more complex types of learning and cognition observed in humans. However, the current state of this field would suggest that these associative laws are complex, and a surprising variety of theories have been proposed in recent years. For example, it is well known that there are a number of regularities in learning that cannot be explained by the Rescorla-Wagner model (Miller, Barnet, & Grahame, 1995). Because of this, many different models have been proposed in an attempt to either broaden the scope of the Rescorla-Wagner model, or to replace it with a theory that has been derived from an alternative framework (for reviews see Pearce & Bouton, 2001; Wasserman & Miller, 1997). "Other cognitive processes such as attention, memory, and information processing are now being invoked to help explain the facts of associative learning. The next several years of research will be exciting ones, as neuroscientists and cognitive scientists join experimental psychologists in an interdisciplinary attack on the challenging problems of associative learning and behavior change" (Wasserman & Miller, 1997, p. 598).

With respect to this interdisciplinary research program, distributed associative memories may provide an interesting environment in which new ideas about associative learning can be explored. For instance, we noted earlier that the delta rule has been proven to be formally equivalent to the Rescorla-Wagner rule (Sutton & Barto, 1981). Presumably, this implies that it too suffers from the same limitations that have been motivating new theories about associative learning in animals. Can these new theories be implemented in the contiguity-based scheme that we have been developing in this framework? Can attentional modulations be added to a distributed associative memory by manipulating stimulus encodings, and then applying something like the delta rule?

There has also been a considerable amount of interest in using models like the one that has been introduced in this chapter to account for a number of different regularities in human memory (Anderson, Silverstein, Ritz, & Jones, 1977; Eich, 1982; Hinton & Anderson, 1981; Murdock, 1982, 1985; Pike, 1984). One reason for this interest has been the fact that when distributed memories make errors, these errors are systematic, and can be related back to the kinds of errors that are made by human subjects in associative memory experiments. For example, we saw earlier that under certain conditions a distributed associative memory will generate responses that represent “blends” of different memories. Memory models of this type also exhibit emergent behaviors that suggest that they provide an excellent environment in which human associative memory can be explored. “Current connectionist models have been successful in accounting for a range of basic phenomena such as the effect of contingency on associative learning, as well as more complex effects such as enhanced responding to an unseen prototype pattern and partial memory for the training items” (Shanks, 1995, p. 151).

Interestingly, one of the primary attractions of distributed associative memories has been the fact that they offer theories that appear to be more biologically plausible than their competitors (Hinton & Anderson, 1981; Shanks, 1997). Indeed, many researchers have recently been interested in taking networks like the ones that have been described in this chapter, or more sophisticated networks, and using these simulations to study neural mechanisms of learning and memory (Brown, 1990; Cotman et al., 1988; Foster, Ainsworth, Faratin, & Shapiro, 1997; Gluck & Myers, 1997; Lynch, 1986; Martinez & Derrick, 1996).

The reason for this interest has been the discovery of a particular neural phenomenon, called long-term potentiation. Long-term potentiation is the long-lasting increase of synaptic efficiency that occurs when two connected neurons are active (or nearing activity) at roughly the same time. This increase in efficiency appears to be related to the properties of a particular receptor mechanism, the NMDA receptor. It also appears to be related behaviorally to memory and spatial learning mediated by the hippocampus, because chemicals that block NMDA receptors disrupt these behaviors. In short, the biochemical study of long-term potentiation appears to be revealing the mechanisms that underlie the kind of neural changes that motivated theories of association by both James (1890) and Hebb (1949).

However, with this increased understanding of long-term potentiation, and with an emerging and detailed understanding of neural mechanisms, there has also been an increased need to propose more sophisticated models of synaptic change. Brown et al. (1990) note that there have been anywhere from 50 to 100 theories of this type, and proceed to review only a subset of these. They classify them as being Hebbian algorithms, generalized Hebbian algorithms, and global control algorithms. Again, one question to ask is how might this more sophisticated rules be incorporated into the models that have been described in the current chapter. Do these rules result in solving some problems that were not solved by the delta rule? If implemented, do these rules lead to behavioral results that are more or less consistent with the performance of human subjects in memory experiments?

One theme that seems to be emerging in even this cursory glance at the current state of research related to distributed associative memories is that, while interesting, the versions of the networks that were described in this chapter are not as powerful as would seem to be required to keep up with advances in the field. What general approach could be used to increase the power of these networks? In the next two chapters, we will consider two very basic – but critical – modifications. In Chapter 10, we will consider some of the implications of changing the activation function from being linear (as is the case in Equation 9-2) to being nonlinear. In Chapter 11, we will consider how the use of a nonlinear activation function permits even more power through the use of additional layers of processing units separating network input from network output.

9.5 Software Availability

The software that was used to run the simulations described in this chapter is available, free of charge, from my website: http://www.bcp.psych.ualberta/~mike/book2/. At the time of writing, the website includes a Visual Basic 6.0 program and some example training files that can be downloaded, as well as instructions on using the program and on creating new training files that might be of interest. In the near future, Java versions of this program will also be available on the website.

9.6 References

Anderson, J. A., & Rosenfeld, E. (1988). *Neurocomputing: Foundations of Research*. Cambridge, MA: MIT Press.

Anderson, J. A., Silverstein, J. W., Ritz, S. A., & Jones, R. S. (1977). Distinctive features, categorical perception and probability learning: Some applications of a neural model. *Psychological review, 84*, 413-451.

Boring, E. G. (1950). *A History Of Experimental Psychology*. New York, NY: Appleton-Century-Crofts.

Brown, T. H. (1990). Hebbian synapses: Biophysical mechanisms and algorithms. *Annual Review of Neuroscience, 13*, 475-511.

Cotman, C. W., Monaghan, D. T., & Ganong, A. H. (1988). Excitatory amino acid neurotransmission: NMDA receptors and Hebb-type synaptic plasticity. *Annual Review of Neuroscience, 11*, 61-80.

Crick, F., & Asanuma, C. (1986). Certain aspects of the anatomy and physiology of the cerebral cortex. In J. McClelland & D. E. Rumelhart (Eds.), *Parallel Distributed Processing* (Vol. 2). Cambridge, MA: MIT Press.

Dawson, M. R. W. (1998). *Understanding Cognitive Science*. Oxford, UK: Blackwell.

Eich, J. M. (1982). A composite holographic associative recall model. *Psychological Review, 89*, 627-661.

Foster, J., Ainsworth, J., Faratin, P., & Shapiro, J. (1997). Implementing a mathematical model of hippocampal memory function. In M. A. Conway (Ed.), *Cognitive Models Of Memory*. Cambridge, MA: MIT Press.

Furumoto, L. (1980). Mary Whiton Calkins (1863 - 1930). *Psychology of Women Quarterly, 5*, 55-68.

Gluck, M. A., & Myers, C. E. (1997). Psychobiological models of hippocampal function in learning and memory. *Annual Review of Psychology, 48*, 481-514.

Hebb, D. O. (1949). *The Organization Of Behavior*. New York: John Wiley and Sons.

Hebb, D. O. (1959). A neuropsychological theory. In S. Koch (Ed.), *Psychology: A Study Of A Science. Volume1: Sensory, Perceptual, And Physiological Foundations* (pp. 622-643). New York, NY: McGraw-Hill.

Hinton, G. E., & Anderson, J. A. (1981). *Parallel Models Of Associative Memory*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Hume, D. (1952). *An Enquiry Concerning Human Understanding*. La Salle, IL: The Open Court Publishing Company.

James, W. (1890). *The Principles Of Psychology, Volume One*. New York, NY: Dover Publications.

Jordan, M. I. (1986). An introduction to linear algebra in parallel distributed processing. In D. Rumelhart & J. McClelland (Eds.), *Parallel Distributed Processing, Volume 1*. Cambridge, MA: MIT Press.

Kintsch, W. (1970). *Learning, Memory, And Conceptual Processes*. New York, NY: John Wiley & Sons.

Klein, R. M. (1999). The Hebb legacy. *Canadian Journal of Experimental Psychology, 53*(1), 1-3.

Kohonen, T. (1977). *Associative Memory: A System-Theoretical Approach*. New York: Springer-Verlag.

Locke, J. (1977). *An Essay Concerning Human Understanding*. London: J.M. Dent & Sons.

Lynch, G. (1986). *Synapses, Circuits, And The Beginnings Of Memory*. Cambridge, MA: MIT Press.

Martinez, J. L., & Derrick, B. E. (1996). Long-term potentiation and learning. *Annual Review of Psychology, 47*, 173-203.

McClelland, J. L. (1986). Resource requirements of standard and programmable nets. In D. Rumelhart & J. McClelland (Eds.), *Parallel Distributed Processing* (Vol. 1). Cambridge, MA: MIT Press.

McClelland, J. L., & Rumelhart, D. E. (1988). *Explorations In Parallel Distributed Processing*. Cambridge, MA: MIT Press.

Miller, R. R., Barnet, R. C., & Grahame, N. J. (1995). Assessment of the Rescorla-Wagner model. *Psychological Bulletin, 117*(363-386).

Milner, P. M. (1957). The cell assembly: Mark II. *Psychological Review, 64*(4), 242-252.

Murdock, B. B. (1982). A theory for the storage and retrieval of item and associative information. *Psychological Review, 89*, 609-626.

Murdock, B. B. (1985). Convolution and matrix systems: A reply to Pike. *Psychological Review, 92*, 130-132.

Murdock, B. B. (1997). Context and mediators in a theory of distributed associative memory (TODAM2). *Psychological Review, 104*, 839-862.

Pearce, J. M., & Bouton, M. E. (2001). Theories of associative learning in animals. *Annual Review of Psychology, 52*, 111-139.

Pike, R. (1984). Comparison of convolution and matrix distributed memory systems for associative recall and recognition. *Psychological Review, 91*, 281-294.

Rescorla, R. A., & Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In A. H. Black & W. F. Prokasy (Eds.), *Classical Conditioning II: Current Research And Theory* (pp. 64-99). New York, NY: Appleton-Century-Crofts.

Rochester, N., Holland, J. H., Haibt, L. H., & Duda, W. L. (1956). Tests on a cell assembly theory of the action of the brain, using a large digital computer. *IRE Transactions On Information Theory, IT-2*, 80-93.

Rosenblatt, F. (1962). *Principles Of Neurodynamics*. Washington: Spartan Books.

Rumelhart, D. E., McClelland, J. L., & Group., t. P. (1986). *Parallel Distributed Processing, V.1*. Cambridge, MA: MIT Press.

Schneider, W. (1987). Connectionism: Is it a paradigm shift for psychology? *Behavior research methods, instruments, & computers, 19*, 73-83.

Selfridge, O. G. (1956). Pattern cognition and learning. In C. Cherry (Ed.), *Information Theory*. London: Butterworths Scientific Publications.

Shanks, D. R. (1995). *The Psychology Of Associative Learning*. Cambridge, UK: Cambridge University Press.

Shanks, D. R. (1997). Representation of categories and concepts in memory. In M. A. Conway (Ed.), *Cognitive Models Of Memory*. Cambridge, MA: MIT Press.

Sorabji, R. (1972). *Aristotle On Memory*. Worcester, GB: Ebenezer Baylis and Son.

Steinbuch, K. (1961). Die lernmatrix. *Kybernetik, 1*, 36-45.

Stone, G. O. (1986). An analysis of the delta rule and the learning of statistical associations. In D. E. Rumelhart & J. McClelland (Eds.), *Parallel Distributed Processing* (Vol. 1, pp. 444-459). Cambridge, MA: MIT Press.

Sutton, R. S., & Barto, A. G. (1981). Toward a modern theory of adaptive networks: Expectation and prediction. *Psychological Review, 88*(2), 135-170.

Taylor, W. K. (1956). Electrical simulation of some nervous system functional activities. In C. Cherry (Ed.), *Information Theory*. London: Butterworths Scientific Publications.

Warren, H. C. (1921). *A History Of The Association Psychology*. New York, NY: Charles Scribner's Sons.

Wasserman, E. A., & Miller, R. R. (1997). What's elementary about associative learning? *Annual Review of Psychology, 48*, 573-607.

Widrow, B., & Hoff, M. E. (1960). Adaptive switching circuits. *Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record, Part 4*, 96-104.