4 Representation and Generalization in Neural Networks

When we look out at the world, reflected light enters each eye, falls on the retina, and activates some of the neurons there. If we were to record the complete pattern of activated and inactivated retinal neurons, the resulting pattern would be a recoding, or **representation,** of the visual scene. The activated retinal neurons send output that leaves the eye and eventually reaches the visual cortical areas, at the rear of the brain. Again, a pattern is created there, consisting of activated and inactive neurons. The visual scene has been rerepresented. In fact, the visual scene (or any sensory stimulation) goes through an arbitrary number of rerepresentations as it is processed by the brain.

Representations are not unique to the brain. We use representations throughout our daily life. The letters in written words are representations of sounds, and the words themselves represent ideas and concepts. The federal government may represent a person by her nine-digit Social Security number; her friends may represent her in conversation by a name; and the Department of Motor Vehicles may represent her by a sequence of letters and numbers on her driver's license. Each of these representations was chosen to suit a particular need. When we refer to our friends, for example, it is easier to use a short name than to memorize an arbitrary numerical code. However, the government, which must organize records on millions of people, needs a unique way to represent each one—hence, the Social Security number.

The neural network models that we presented in the preceding chapter all assumed a simple representational scheme for relating events in the world to components of the model. The scheme was as follows: Each distinct stimulus event, such as a tone, a light, or an airpuff, was represented by a single input node in the network; the occurrence of that stimulus event was modeled by activating the corresponding node. This activation propagated through the network, and the output node's activity represented the probability (or strength) of a response.

A representation that relates one stimulus (or stimulus feature) to one node (or component) of the model is referred to as a ***component representation*** *(or a* ***local***

representation). Although simple in conception, component representation has been sufficient for a broad range of applications in psychology, neuroscience, and engineering. However, this component representation has serious limitations, and neural network modelers—as well as animal learning theorists—have been grappling with these limitations for years. In this chapter, we review the key limitations of component representation in learning theories and discuss how psychologists and neural network modelers have addressed these limitations.

Section 4.1 introduces the general concepts of representation and generalization. A fundamental challenge for learning is as follows: On the one hand, people and animals often infer that similar stimuli should be associated with similar outcomes; on the other hand, when necessary, people and animals can learn that similar stimuli should be associated with different outcomes. There is always a trade-off between the former process (generalization) and the latter process (specificity) in learning. Section 4.2 discusses some modeling approaches to implementing generalization, and section 4.3 discusses ways to preserve specificity. One way to balance the generalization-specificity trade-off is to use multilayer networks, which incorporate one or more layers of nodes between the input and output nodes. These multilayer networks can develop new stimulus representations during learning. In later chapters, we will argue that this kind of adaptive representation is a primary function of the hippocampal region.

4.1 REPRESENTATION AND GENERALIZATION

Psychologists have long understood that *a fundamental property of animal and human learning is* ***generalization****, the degree to which learning about one stimulus transfers to other stimuli*. One classic generalization study involved training pigeons to peck at a key when a yellow light was present.[1] In this study, yellow light was the stimulus cue, and responding was measured in terms of how many times the pigeons pecked at the key while the light was on. Because the perceived color of light depends on its wavelength, usually given in nanometers (nm), or billionths of a meter, the "yellowness" of the light can be quantified by noting that the wavelength was 580 nm. After training, the pigeons were tested to see how they would respond to other wavelengths (colors) of light, including 560 nm (yellow-green), 520 nm (green), 600 nm (yellow-orange), and so on.

Not surprisingly, the pigeons responded most strongly to the yellow light on which they were trained, as shown in figure 4.1. However, the pigeons also responded to other colors of light, and the strength of their response

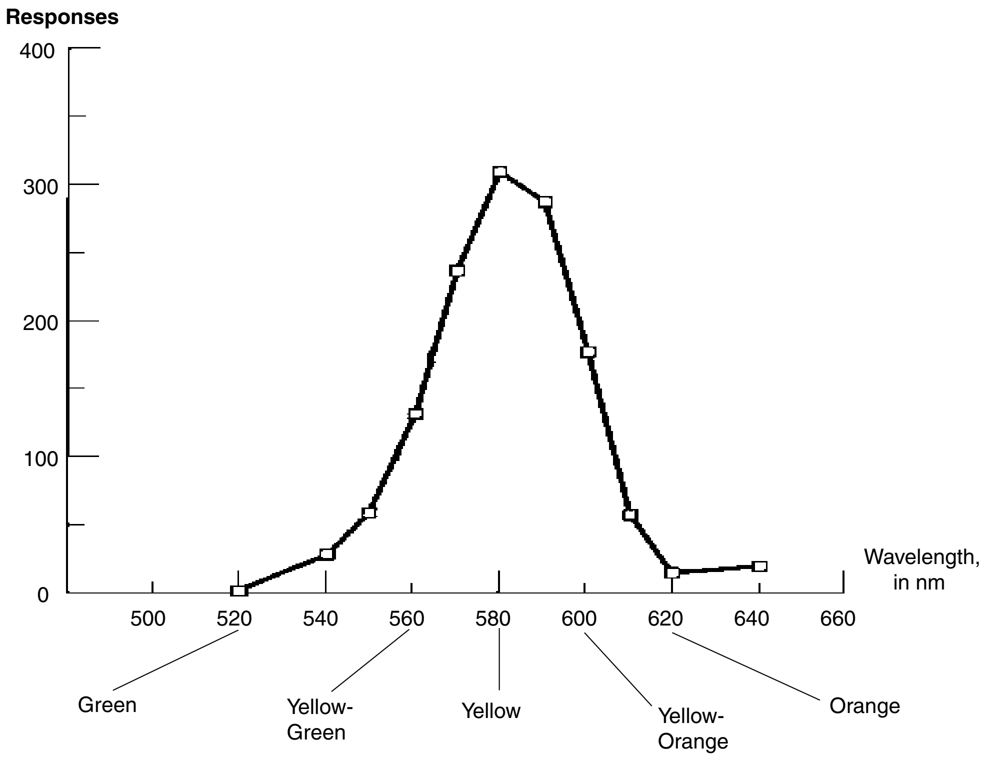


Figure 4.1 Stimulus generalization gradients in pigeons that were trained to peck at a key when illuminated with yellow light but not when dark. Yellow light has a wavelength of about 580 nanometers (nm). When the pigeons were tested with other colors of light, their response rates decreased as the test light colors got farther away from the trained color. Thus, a 560-nm light (yellow-green) still produced a strong response, but a 520-nm light (green) produced little or no responding. Similarly, 600-nm light (yellow-orange) produced more responding than 620-nm light (orange). This general shape, in which response falls off with increasing distance from the trained stimulus, is characteristic of generalization in a wide range of tasks and species. (Adapted from Guttman & Kalish, 1956, pp. 79–88.)

depended on how close (in terms of wavelength) the new color was to the trained yellow light. The less similar a test color was to the trained color, the less the pigeons pecked. The bell-shaped curve in figure 4.1 is known as a **generalization gradient,** and it is a common feature of generalization studies in a variety of modalities (e.g., using tones, shapes, brightnesses, or textures) in species ranging from honeybee to octopus to rabbit to human.[2]

Generalization based on physical similarity is fundamental to learning and allows us to apply prior learning to new instances. For example, an animal that eats a mushroom and becomes sick may generalize from this experience and avoid other mushrooms in the future. Similarly, a person who has learned to drive a Chevrolet is likely to be able to generalize all or most of this skill to driving a Toyota, even though the two cars have slightly different layouts for their steering wheels and gearshifts. *Generalization is thus critical for applying prior experience to new situations, without requiring fresh learning about every new instance*.

Generalization in One-Layer Networks

The one-layer networks that were introduced in chapter 3, with their component representation of features in the world, have serious limitations for capturing the natural patterns of animal and human generalization. Figure 4.2 shows a network with input nodes representing different colored lights. Like the pigeons in the experiment described above, this network can be trained to respond when the yellow light is present (the middle of the five input nodes). This training results in strengthening the weight from the yellow

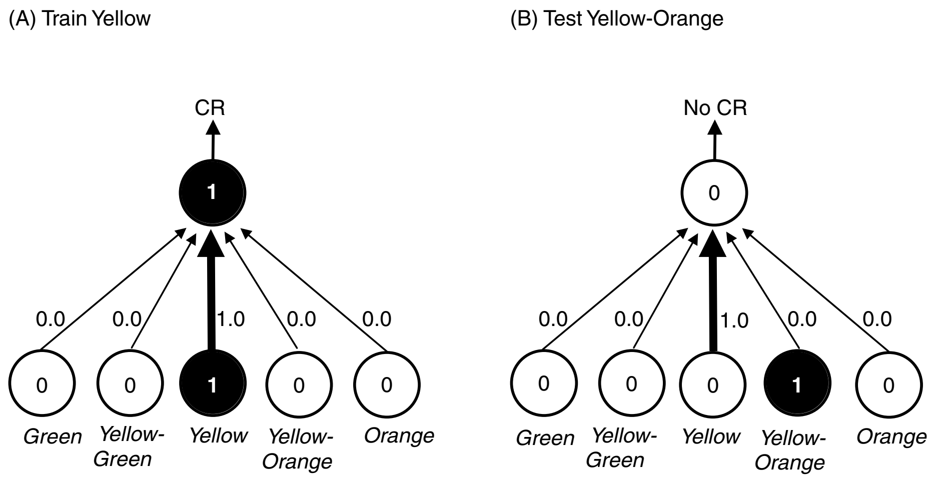


Figure 4.2 A network to implement the stimulus generalization experiment schematized in figure 4.1 could have one input node for each possible color of light. In effect, each color of light represents a different conditioned stimulus (CS). (A) The network is trained to respond to a yellow light. At the end of training, the weight from the "yellow" input node to the output node is strong. (B) When a yellow-orange light is presented to the network, it activates a different input node. This input node has never had its weight strengthened, so it does not activate the output node. The network gives no response to the yellow-orange light, despite the similarity in color to the previously trained yellow light.

input node to the output node, so that a response is generated in response to the yellow light, as shown in figure 4.2A.

Now consider what happens when, for the first time, the network is presented with a slightly different stimulus: a yellow-orange light (figure 4.2B). Although the yellow-orange input node is just to the right of the yellow input node, its weight has never been strengthened. Accordingly, the network does not respond at all to the yellow-orange light—or to any other novel color. Thus, a generalization gradient for this network, as shown in figure 4.3, would reflect strong responding to the trained stimulus but no responding to any other stimulus. This is in striking contrast to the gently sloping generalization gradient shown by animals and humans illustrated in figure 4.1.

The problem with the network in figure 4.2 is that it treats two physically similar stimuli—such as yellow and yellow-orange lights—exactly the same as it treats two dissimilar stimuli, such as the tone and light described in chapter 3. Clearly, however, animals and people do not treat the tone/light distinction equivalently to the yellow/yellow-orange distinction; we are far more likely to generalize what we know about a yellow light to a very similar yellow-orange light and far less likely to generalize what we know about a tone cue to a light cue. *A major challenge for learning theorists has been to understand how differences in stimulus similarity affect learning and generalization behaviors.*

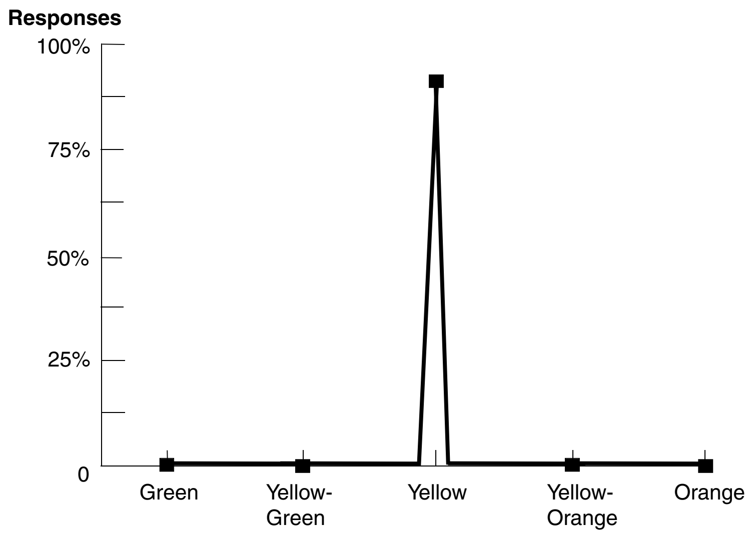


Figure 4.3 The stimulus generalization gradient for the network shown in figure 4.2A,B. The network responds strongly to the yellow input but does not respond to any other input. By contrast, animals and humans typically have smooth generalization gradients in which responses decrease as a function of increasing distance from the trained stimulus (refer to figure 4.1).

Two Challenges for Generalization in Networks

The problem with the simple network models from chapter 3, illustrated in figure 4.2, is that they miss an essential feature of how animals and people respond to stimuli that are similar to those they have experienced before. *In general, animals and people tend to assume that two stimuli that are physically similar will have similar consequences and generalize accordingly*.3

Of course, generalization does have its costs. Consider again the animal mentioned earlier that has encountered one mildly poisonous mushroom. If this animal generalizes from this one bad experience with a particular mushroom to every other similarly colored food, it will deprive itself of many perfectly satisfactory food sources. Likewise, a child who has one frightening experience with an aggressive dog may overgeneralize and develop a fear of all dogs. Thus, *there is also a need for specificity in learning, to allow superficially similar stimuli to be associated with different responses*.

How much generalization is appropriate depends on the particular situation, and it may be difficult to determine this in advance. In general, *there is always a trade-off between generalization and specificity in learning*. Understanding how people and animals balance these two competing forces has been a major concern of psychological learning theorists for many years. The sections to follow discuss some of the main psychological issues concerning generalization and similarity and review how these have informed or anticipated subsequent theories of brain function.

4.2 WHEN SIMILAR STIMULI MAP TO SIMILAR OUTCOMES

Given the importance of stimulus generalization in learning and the poor generalization shown by simple network models such as the one in figure 4.2, many researchers have sought methods to incorporate appropriate stimulus generalization. Central to this work was the realization that the component representation shown in figure 4.2 might be a convenience but probably bears little relation to how stimuli are encoded in the brain.

In addition to its problems with inappropriate generalization, a component representation also has serious limitations for representing larger-scale problems: It requires that one node be dedicated to each possible stimulus. Given the vast number of unique stimuli in the universe and the relatively low number any one individual could experience in a single lifetime, component representation is wasteful, if not impossible.

We now know that the brain generally represents information using **distributed representations,** in which each stimulus is encoded by many different neurons and each neuron may respond to conjunctions of features that may be present in many different stimuli.[4] This does not mean that neurons are not specific: An individual neuron in visual cortex may respond maximally to the sight of a particular bar rotated to a particular angle moving at a particular speed in a particular direction, while another neuron in sensory cortex might respond to a touch stimulus on a particular part of the body surface. But each of these neurons will be activated by any stimulus that shares the relevant features. Distributed representation (also called **coarse coding**) is efficient because a large number of stimuli can be processed by a smaller number of neurons, each of which encodes some features of the stimuli. Distributed representations also have the advantage that if some of the units are lost or degraded, the remaining units may be able to provide enough information to represent the stimuli adequately. The cost of a distributed representation is that recognizing a particular stimulus may require integrating information from the many different neurons that encode the stimulus features.

In the late 1950s W. K. Estes, a mathematical psychologist, showed that just this sort of encoding scheme could account for a wide range of stimulus generalization behaviors in animals and humans.[5] At around the same time, neuroscientists such as Karl Lashley and Donald Hebb were coming to similar conclusions, based on experimental and theoretical analyses of the brain.[6] Like Estes, they concluded that the brain most likely uses distributed representations to encode information. But they had only indirect evidence for this assumption. More recently, however, neuroscientists have found explicit empirical evidence for distributed representations in the visual system, sensory and motor cortex, and other brain regions.[7] As with the learning theories of chapter 3, behavior-driven and physiology-driven research converged on similar principles.

Application of Distributed Representations to Learning

To illustrate how a distributed representation uses a set of overlapping elements to encode input stimuli, turn back to the problem of training pigeons to peck at differently colored lights. Figure 4.4A shows a distributed representation for encoding the discrimination among colored lights.

Each color activates a subset of the nodes: Yellow activates the three centermost nodes, orange activates the three rightmost nodes, and so on. *The pattern of activated and nonactivated nodes is the representation of each color input.*

In figure 4.4A, the nodes are laid out **topographically,** meaning that nodes responding to physically similar stimuli—such as yellow and yellow-orange lights—are physically next to each other in the network. Actually, the physical layout itself is not really what is important. Rather, what is important is that *the degree of overlap between the representations of two stimuli reflects their*

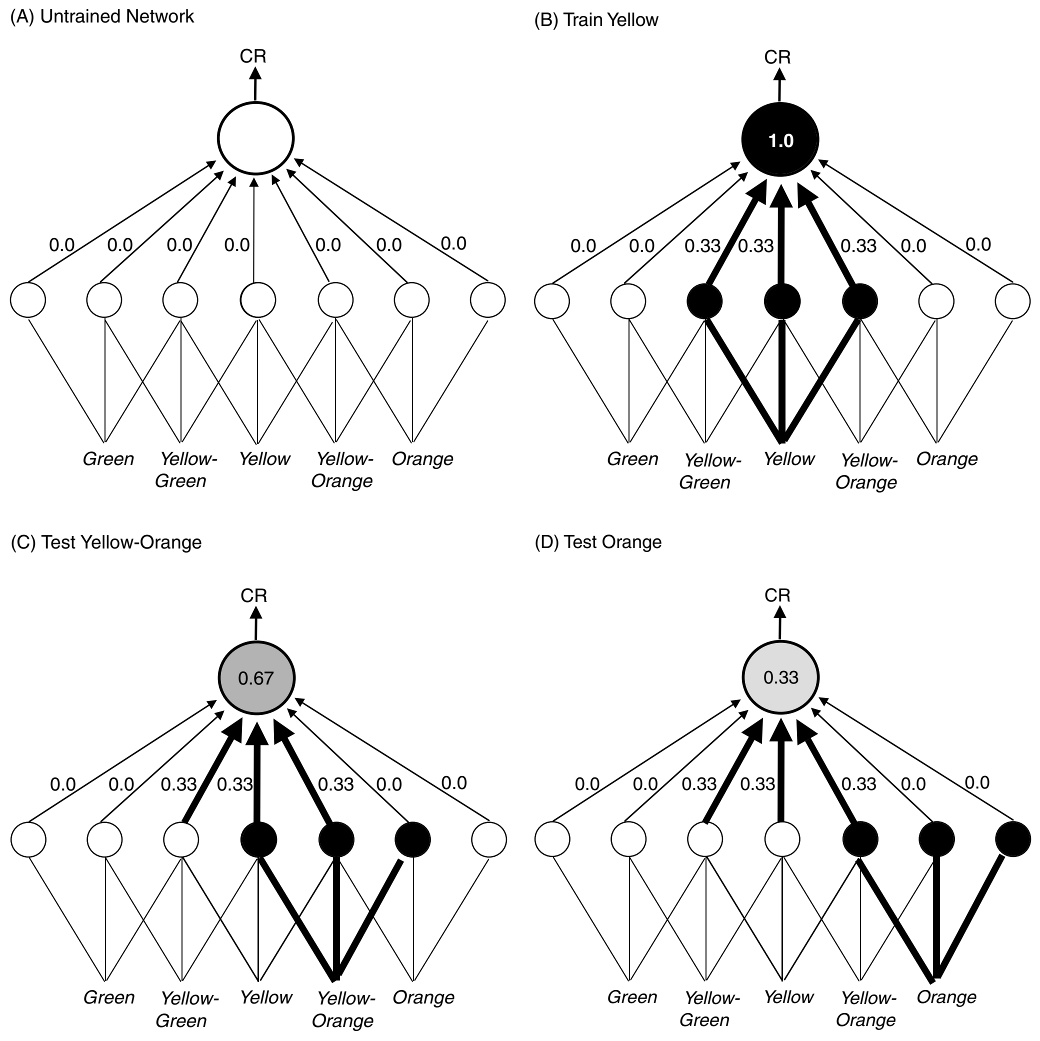


Figure 4.4 (A) A distributed representation of the color stimuli can be schematized as a network with a large number of input nodes connected to a single output node. One stimulus, such as yellow light, will activate a subset of these inputs nodes. A similar stimulus, such as yellow-orange light, will activate a subset of nodes that overlaps highly with the nodes activated by yellow. Orange light will activate a subset of nodes that overlaps minimally with the nodes activated by yellow. (B) When the network is trained to respond to yellow light, this corresponds to increasing the weights to the output node from all the input nodes that were activated by yellow light. (C) If this network is then tested with a new stimulus, such as yellow-orange light, the response will generalize depending on the overlap between the input nodes activated by yellow and by yellow-orange. (D) Increasingly different colors, such as orange, overlap still less with the representation of yellow and evoke correspondingly less response.

physical similarity. Thus, in figure 4.4A, there is more overlap between the representations for yellow and yellow-orange than between those for yellow and orange. There is no overlap between the representations of very different colors, such as green and orange.

Now suppose that this network is trained on the problem of responding to a yellow light. This is done within this distributed network by increasing the weights from the nodes that are active in response to yellow light (figure 4.4B). One way to do this would be with the Rescorla-Wagner learning rule from chapter 3, whereby each of the weights from the three active nodes would eventually equal one-third (approximately 0.33, as indicated in the figure). The weights from the two inactive nodes would remain at 0.

It is instructive to compare the distributed network in figure 4.4B, which was trained to respond to a yellow light, with the local-representation network from figure 4.2A, which was trained in the same way. The distributed network in figure 4.4B has the learning distributed across three weights of 0.33 each; in contrast, the component network in figure 4.2A has this same yellow→response rule localized as one single weight of 1.0. When tested on the original yellow light, these two networks yield identical behaviors: Both give a maximal response when this yellow light is presented.

However, the difference between the distributed and local versions of this associative rule becomes apparent when the networks are presented with other, similar stimuli. The distributed network's generalization behavior can be tested by presenting it with a yellow-orange light, as illustrated by figure 4.4C, where yellow-orange activates a representation that has considerable overlap with the trained yellow light. As a result, there is a reasonably strong response of 0.67, proportional to the two-thirds degree of overlap between the representations for yellow and yellow-orange lights. If the same network were tested with orange light, there would be less overlap and a weaker response of 0.33, as shown in figure 4.4D.

Figure 4.5 shows that the pattern of responding to a whole series of novel lights produces a stimulus generalization gradient that decreases with distance to the trained stimulus—just like the gradient seen in studies of stimulus generalization by animals and humans (figure 4.1). This is in marked contrast to the generalization gradient in figure 4.3, which was produced by the network with only a local component representation.

It is worth noting that the gradients in figure 4.1 and 4.5 are not identical—the network responds more strongly to orange light than the pigeons do—but this is less important. The exact width and shape of the generalization gradient can be manipulated by varying the number and overlap of nodes in the network. What is important is the overall shape: A network that uses

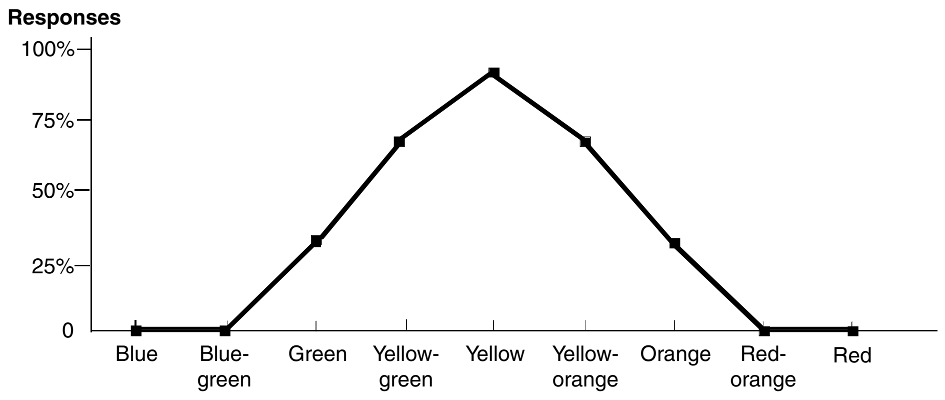


Figure 4.5 The trained network in figure 4.4 generates this stimulus generalization gradient, which shows decreasing responses as a function of distance from the trained stimulus. Responding is highest for the trained stimulus (yellow), decreases for slightly different colors (yellow-orange and yellow-green), and is still less for more distant colors (green and orange). Colors that do not overlap at all with the representation of yellow (e.g., blue, red) would evoke no response at all. This generalization curve is similar in shape to that seen in animals and humans (e.g., figure 4.1). The exact width and shape of the generalization gradient depend on how input nodes are allocated between and among different input patterns.

distributed representation tends to generalize much more realistically than one that uses a local representation.

Stimulus Generalization and Distributed Representations in Multilayer Networks

The distributed representation of the network in figure 4.4 can be viewed as an intermediate layer of nodes, which lie in between the original inputs (yellow, yellow-orange, and orange light) and the output. In effect, the original inputs have simply been left out of the drawing. Figure 4.6 shows what the network would look like if these original inputs are included; the distributed representation now falls in an **internal layer** of nodes (sometimes called a **hidden layer** in neural network jargon).

As before, there is an upper layer of associative weights connecting these nodes to the output, and changing these weights alters the network's output response. But now there is also a lower layer of representational weights that determines what representation is evoked by the inputs. In figure 4.6, yellow inputs again activate the leftmost internal-layer nodes, orange inputs again activate the rightmost internal-layer nodes, and yellow-orange inputs activate the center nodes. Again, there is more overlap between similar stimuli—such as yellow and yellow-orange—than between less similar stimuli, such

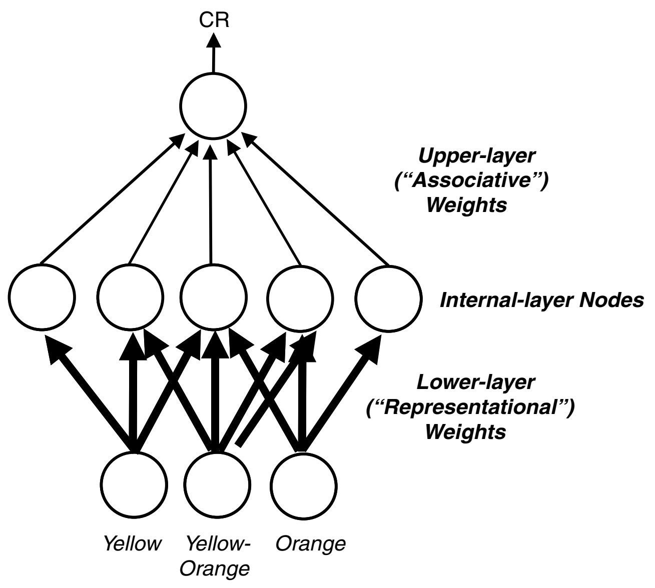

CR
Upper-layer ("Associative") Weights
Internal-layer Nodes
Lower-layer ("Representational") Weights
Yellow
Yellow-Orange
Orange

Figure 4.6 Another way to schematize the distributed representation is to consider a multilayered network. For simplicity, only the input nodes corresponding to yellow, yellow-orange, and orange are shown. Between the input nodes, which become active in response to particular stimulus inputs, and the output node is a third internal layer of nodes. Weighted connections from input nodes to internal-layer nodes activate a distributed representation across the internal-layer nodes. Weighted connections from internal-layer nodes to the output node determine the network's response to a given input. The lower layer of weights are *representational*, since stimulus representations in the internal layer depend on how those weights are set. The upper layer of weights are *associational*, since stimuli are associated with responses on the basis of how the internal-layer representations activate the output node.

as yellow and orange. For now, we assume that the weighted connections between input units and internal units are fixed, so that the representation of a particular stimulus is "hard-wired" and unalterable. Later, in section 4.3, we will discuss how these weighted connections might be modified to change representations.

Integrating Distributed Representations with Error-Correction Learning

A multilayered network that incorporates distributed representations (figure 4.6) can be used to illustrate and explain several very counterintuitive empirical results regarding generalization. For example, it might seem logical to expect that the optimal way to train an animal (or a network) to respond to a yellow light would be simply to present the yellow light over many training trials and provide reinforcement on each trial. However, Rescorla noted that the Rescorla-Wagner model, when applied to a network with distributed representations, leads to a different prediction.[8]

Rescorla's analysis of his model as applied to this kind of training paradigm can be understood as follows: First, the network is trained to respond to yellow light (figure 4.7A). As usual, this means that those internal-layer units that are activated by the yellow input have their connections to the output node strengthened. Using the Rescorla-Wagner learning rule, each of the connections from the leftmost internal nodes will have a weight of 0.33 at the end of training. When yellow light is presented, the response will be the sum of the activated internal nodes times their weights:

$$\begin{aligned}\text{Response} &= (1.0 \times 0.33) + (1.0 \times 0.33) + (1.0 \times 0.33) + (0.0 \times 0.0) + (0.0 \times 0.0)\\ &= 1.0\end{aligned}$$

This is the maximal response; further training with the yellow light will not result in any further weight changes because there is no more error to correct, and therefore the Widrow-Hoff (Rescorla-Wagner) rule indicates no further weight changes.

Suppose that the network is next trained to respond to an orange light. On the first trial (figure 4.7B), the network will give a small response to orange, proportional to the small overlap between the trained yellow and novel orange inputs. The Rescorla-Wagner rule states that the connections from each active weight should then be increased so as to reduce the error in response to the orange stimulus. After several training trials, the weights will be adjusted so that the network gives a strong response to orange (figure 4.7C):

$$\begin{aligned}\text{Response} &= (0.0 \times 0.33) + (0.0 \times 0.33) + (1.0 \times 0.56) + (1.0 \times 0.22) + (1.0 \times 0.22)\\ &= 1.0\end{aligned}$$

Notice that the weights from the two rightmost nodes have increased from zero—and the weight from the central node has increased too. Now, if the yellow light is presented again (figure 4.7D), the response is the sum of the weighted connections from all active internal units:

$$\begin{aligned}\text{Response} &= (1.0 \times 0.33) + (1.0 \times 0.33) + (1.0 \times 0.56) + (0.0 \times 0.22) + (0.0 \times 0.22)\\ &= 1.22\end{aligned}$$

Note that this is stronger than the originally trained response to a yellow light! This seems paradoxical. However, when Rescorla conducted a similar experiment with rats, this is exactly what he found,[9] just as the Rescorla-Wagner model predicted.

gluc_c04.qxd 8/29/00 1:43 PM Page 93

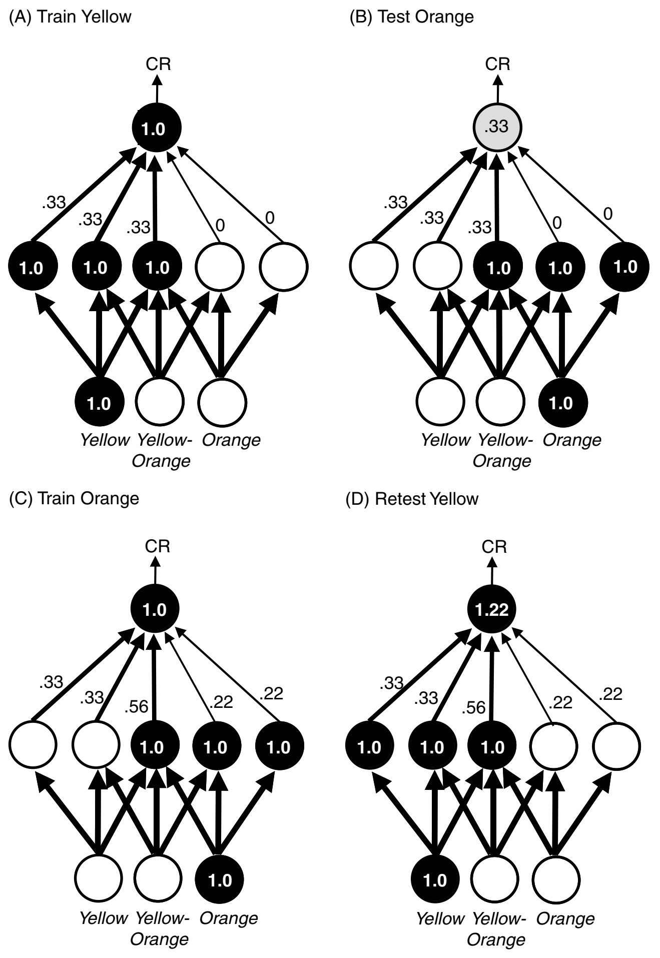


Figure 4.7 The use of a distributed representation can explain some counterintuitive generalization results, including one predicted by Rescorla (1976) and later confirmed in animals. (A) The network is trained to respond to yellow light. This is done by applying the Widrow-Hoff learning rule to the upper layer of weights. The internal nodes that are activated by yellow have their connections to the output strengthened. With sufficient training, the network gives a strong response (1.0) to yellow. Further training with yellow light will not result in any further increases in connection weights or response strength. (B) If an orange stimulus is presented, the network gives a weak response, proportional to the relatively low overlap between the representation of orange and that of yellow. (C) The network is now trained on orange. The weights from internal nodes that are activated by orange are strengthened. Note that the weight from the center internal node, activated by both yellow and orange, increases beyond its original level. (D) When yellow is presented once again, the increase in the center internal node weight causes a stronger response (1.22) than was originally trained. Thus, additional generalization training on a similar (orange) light increased the response to yellow light beyond what could have been achieved by training to the yellow light alone.

These and other findings demonstrated that distributed representations provide a framework for capturing empirical data on stimulus similarity and generalization, especially when combined with the error-correcting learning procedure from the Rescorla-Wagner model (also known as the Widrow-Hoff rule).

The Limits of Similarity-Based Generalization

So far, this chapter has focused on how generalization to novel stimuli is affected by the manner in which a network (or brain) represents information. Generalization allows efficiency in learning: It is not necessary to explicitly learn a response to each different stimulus if learning about one stimulus can generalize to other, similar stimuli. However, generalization can have its costs: It hinders the ability to make fine discriminations among similar stimuli that are to be mapped to different responses.

For example, a child learning to read must discriminate between the letters "G" and "C," which have very different meanings, even though their appearance may differ by only a single serif. Conversely, "G" and "g" may look dissimilar, yet they encode the same letter and, in many situations, should be treated the same way. Ideally, the letter stimuli should be represented somewhere in the brain so that there is considerable overlap between the representations for the letters "G" and "g," which have the same meaning, but little representational overlap between "G" and "C," despite their superficial physical similarity. Thus, *the optimal degree of generalization depends not only on the physical similarity between stimuli but also on the similarity (or difference) between their meanings, or consequences*.

4.3 WHEN SIMILAR STIMULI MAP TO DIFFERENT OUTCOMES

Suppose that an animal is trained that either a tone or a light predicts an unconditioned stimulus (US) (such as a shock or airpuff to the eye) but that there will be *no* US when the tone and light appear together. Thus, the animal must learn to respond to the individual tone or light cues but to withhold responding to the tone and light compound. Naturally, there should also be no response when no stimulus is present, that is, when neither the tone nor the light is present. This task is known by two different names: Psychologists call it **negative patterning,** while mathematicians and neural network researchers call it the **exclusive-OR** (or XOR) task. By either name, it is an extremely difficult task.

Because both the tone and the light cues are part of the tone-and-light compound, there is a natural tendency to generalize from the component cues to the compound and vice versa. However, such generalization will interfere with learning because the components and the compound are associated with very different outcomes. Thus, negative patterning is difficult because it requires suppressing the natural tendency to generalize among similar stimuli.

With extensive training, the negative patterning tasks can be mastered by rats, rabbits, monkeys and humans. Figure 4.8A shows one example of negative patterning in rabbit eyeblink conditioning.[10] After a few days of training, animals learn to give strong responses to either the tone alone (open

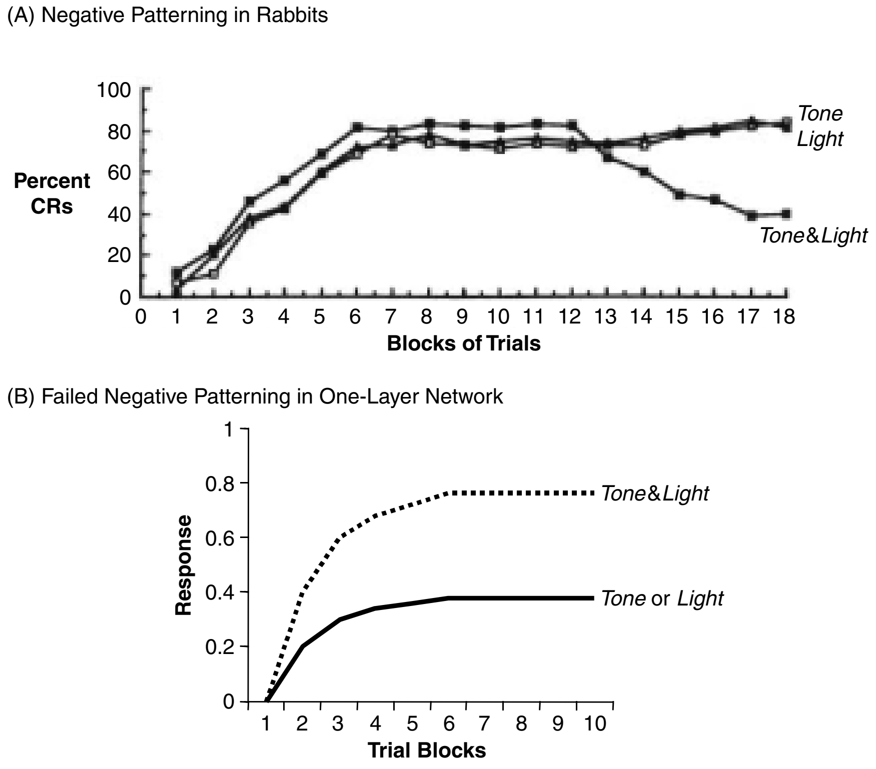


Figure 4.8 Negative patterning involves learning to respond to two CSs (e.g., tone and light) when presented separately but withholding response when the two CSs are presented together. (A) Rabbit eyeblink conditioning data (as reported in Kehoe, 1988): With extensive training, rabbits produce blink CRs to the tone or light separately but not to the compound. (Adapted from Kehoe, 1988, Figure 9.) (B) A simple network, like that shown in figure 4.9A, cannot solve the problem. Trained with the Widrow-Hoff learning rule (and learning rate 0.2), the network comes up with a "compromise" solution: responding partially to the tone or light alone but always responding more to the compound than to the components. Further training will not improve performance.

triangles) or the light alone (open squares). However, the rabbits generalize incorrectly and also give strong responses to the compound tone and light stimulus (solid circles). Only with extensive training do the rabbits begin to suppress responding to the compound while maintaining strong responses to either component alone.

This negative patterning task can be applied to a simple network containing one input node for the tone, one input node for the light, and one output node, as shown in figure 4.9. When this network is trained on the negative patterning task, it can learn to respond appropriately to the individual cues; but it can never learn to do this while also withholding responding to the compound cue, as is illustrated in figure 4.8B.

Figure 4.9 shows the reason for this failure. To produce correct responding to the tone alone, the weight from the input unit encoding tone must be strengthened to 1.0 (figure 4.9A). To produce correct responding to the light alone, the weight from the input unit encoding light must also be strengthened to 1.0 (figure 4.9B). But this means that if the tone and light are presented together, activation will flow through those weighted connections and produce strong responding to the compound—stronger responding, in fact, than to either component alone (figure 4.9C).

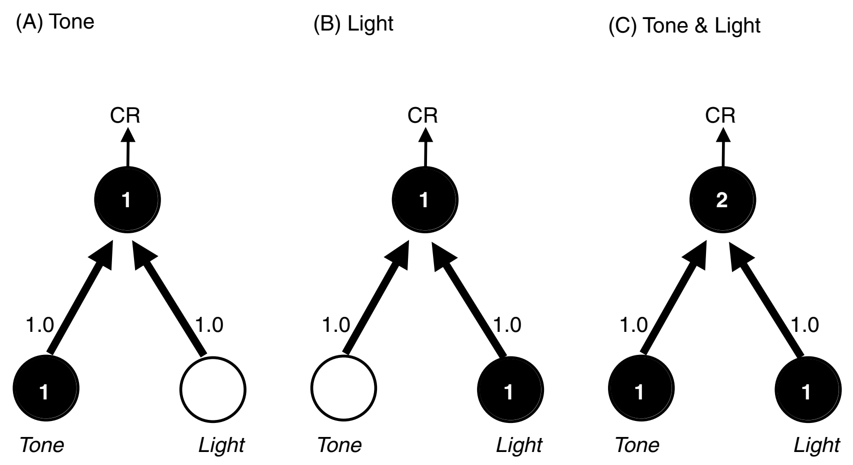


Figure 4.9 A simple, three-node network cannot solve the negative patterning problem. (A) To correctly generate a strong response to the tone CS, there must be a strong weight on the connection from that input to the output. (B) To correctly generate a strong response to the light CS, there must be a strong weight on the connection from that input to the output. (C) But then, when both tone and light CSs are present, the network will incorrectly give a strong response; in fact, the response to the compound (tone and light) will always be stronger than the response to either component cue alone.

All the weights could be decreased, of course, to reduce responding to the compound in figure 4.9C, but this would also incorrectly reduce responding to the components. In fact, there is no way to set connection weights in the network of figure 4.9 so that the network responds correctly to all three different types of training trials.

Multilayered Networks and Configuration

Widrow and Hoff and Rescorla and Wagner were all well aware that their learning procedure could not provide a solution to this type of task. Both sets of researchers were also aware of one possible solution to this limitation: adopting additional nodes to represent cue configurations. This can be done by using internal-layer nodes, as is shown in figure 4.10. One internal-layer node (tone-only) becomes active whenever the tone is present, one (light-only) becomes active whenever the light is present. A third node (tone&light) responds to the configuration, becoming active only when *both* tone and light are present. This is accomplished by specifying that there is a minimum total input (known as a **threshold**) that must be reached before the node can become active.

Figure 4.10A shows that when the tone (alone) is present, the tone-only node becomes active, and this in turn activates the output node. Similarly, in figure 4.10B, the light activates the light-only node, which activates the output node. However, when both tone and light are present (figure 4.10C), all three internal-layer nodes are activated. The tone-only and light-only nodes each send excitatory activation to the output node. However, the configural tone&light node sends an inhibitory signal to the output node, which counteracts the excitatory activation. Activation in the output node is 0, and the network does not respond to the configuration of tone and light. Thus, this multilayer network with configural internal-layer nodes can solve the negative patterning problem.

Negative patterning is just one example of a larger class of problems, involving configurations of stimuli, that single-layer networks cannot solve. *Configural tasks require sensitivity to the configurations (or combinations) of stimulus cues, above and beyond what is known about the individual stimulus components*.

Configural learning also occurs in human learning. It is especially apparent in categorization learning, the process by which humans learn to classify stimuli into categories. In several papers over the late 1980s, one of us (Mark Gluck) and his graduate advisor (Gordon Bower) used simple configural networks to model some aspects of human categorization learning.[11] These

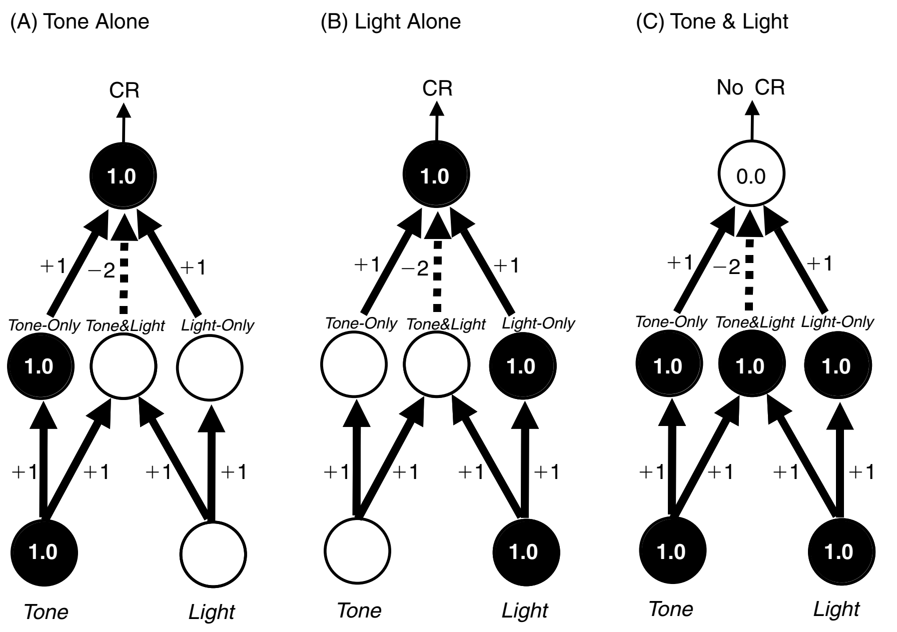


Figure 4.10 A multilayer network with configural internal-layer nodes can solve the negative patterning problem. One internal-layer node (tone-only) becomes active whenever the tone CS is present, one (light-only) becomes active whenever the light CS is present, and one (tone&light) becomes active when both tone and light are present but not when only one is present. (A) When the tone alone is present, the tone-only internal-layer node sends excitatory activation to the output node, and the network generates a response. (B) When the light alone is present, the light-only internal-layer node sends excitatory activation to the output node, and the network generates a response. (C) When both tone and light are present, the tone&light configural node is also activated. This sends strong inhibitory signals to the output node, counteracting the excitatory signals from the tone-only and light-only nodes, so the net output is 0. Thus, the network responds to the components but not to the configuration of tone and light.

models were not meant to capture all of the richness of category learning by people, but they did show that some underlying principles of category learning could be understood as reflecting the same error-correcting principles that are evident in the Rescorla-Wagner model of conditioning.

One of these studies started with an experiment testing human subjects' ability to learn a fictitious medical diagnosis. On each trial, subjects were given the description of a hypothetical patient who had particular symptoms. Subjects were required to determine on the basis of the pattern of symptoms whether each patient had a rare disease. Certain symptom configurations indicated the rare disease; other configurations indicated that the patient did not have the rare disease. The key design principle was that no single symptom was perfectly predictive of the disease or its absence; instead, subjects had to learn about configurations of symptoms. The use of a network with configural nodes can provide a reasonable approximation to human performance in many cases.[12]

For example, figure 4.11A shows a network that can learn to diagnose hypothetical patients on the basis of whether the patients complain of symptoms including fever, ache, and soreness.* By setting appropriate weights from each internal-layer node to the output, the network can learn which symptoms—and which configurations of symptoms—should lead to diagnosis of the rare disease. For example, if a patient reports fever and soreness, this would activate the internal-layer nodes corresponding to these individual symptoms as well as their configuration (figure 4.11B). By setting upper-layer weights appropriately, the network could learn to diagnose this symptom pattern as being indicative of the disease; other symptom configurations could indicate the absence of the disease, much as in the negative patterning problem.

Although this type of simple network model can be applied only to a restricted range of experimental circumstances, it has shown surprising accuracy in predicting human behavior within that range, including data on the relative difficulty of learning various classifications and their responses to generalization tests involving various combinations of cues.[13]

Configuration and Combinatorial Explosion

The previous section argued that networks that incorporate configural nodes can provide additional learning power that enables the solution of the negative patterning problem and other complex tasks. Moreover, this configural-cue representation has been shown to accurately capture some aspects of human categorization and animal learning behaviors.

However, it should be obvious that the approach is inherently limited. The disease-diagnosis network of figure 4.11 has one internal-layer node for each symptom, and one for each possible combination of symptoms. As long as there are only three symptoms, this is acceptable as a psychological model because it requires keeping track of weights on only seven nodes. But suppose the problem required learning about ten symptoms; in this case, more

*In their original papers, Gluck and Bower did not draw their networks in the same way as shown in figure 4.11; they showed only the configural (internal-layer) and output nodes. However, the formal properties of the networks were exactly the same as shown in figure 4.11.

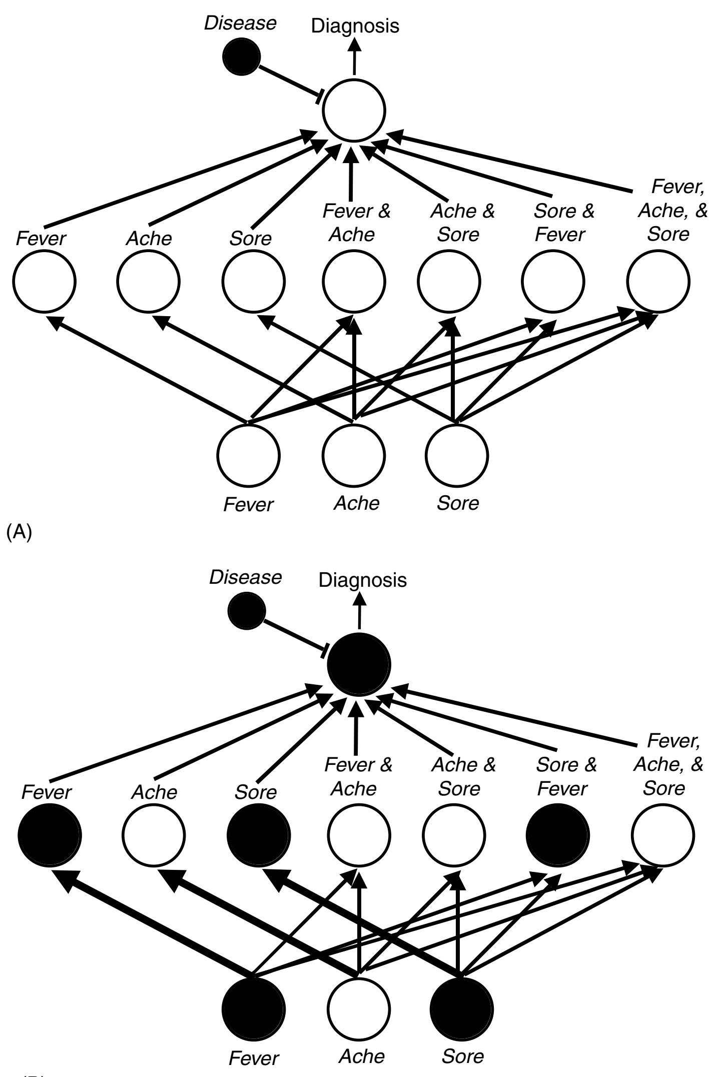


Figure 4.11 A network to encode a categorization problem, in which a hypothetical patient's configuration of symptoms is used to diagnose the presence or absence of a disease. The presence or absence of individual symptoms, as well as combinations of two or more symptoms, may be predictive. For simplicity, the numbers corresponding to weight strengths are not shown in this and subsequent figures; instead, line thickness approximates the strength of connections. (B) For example, the presence of both fever and soreness might predict the presence of the disease, but fever or soreness alone (or in combination with a third symptom, ache) might predict the opposite. This kind of network with configural nodes can mimic many aspects of how humans learn categorization tasks.

than a thousand internal-layer nodes would be required to encode every possible combination of one or more symptoms! And this is only for a "toy" problem; in the real world, there is a potentially unlimited number of stimuli that could occur singly or in combination. If it were necessary to devote one neuron to encode each possible configuration, the requirements would quickly surpass even the vast number of neurons in the brain. This dilemma is called a **combinatorial explosion,** reflecting the rapid expansion of resources required to encode configurations as the number of component features increases.

Even if combinatorial explosion were not a problem, devoting one node to each possible configuration is wasteful. Although there may be a vast number of possible stimulus configurations, any one individual will experience only a small fraction of these configurations in a single lifetime.

At the same time, the power of configural networks argues that this general approach can be very fruitful, both as a computational tool and as a psychological model. What is really needed is a way to harness the computational power of configural nodes without vast numbers of nodes to encode all possible configurations. The simplest way out of this dilemma is to assume that configural nodes are not prewired but, rather, are created as needed. Thus, suppose that the problem is to learn which combinations of ten symptoms predict a rare disease. As was mentioned above, more than a thousand nodes would be required to code every possible configuration of symptoms. If only a few of these thousand symptom configurations actually occur, then only a few configural nodes would be needed—just enough that there is one configural node for each configuration that might appear.

Of course, for most interesting problems, it is impossible to know in advance exactly how many configurations might occur. One possible solution is simply to construct a new internal-layer node each time a new configuration is encountered during learning. Such a network is often called an **exemplar model,** since each internal-layer node responds to one "example" of stimulus configurations that has been experienced so far. Such exemplar models do capture some aspects of human learning.[14] However, an exemplar model will eventually suffer from an accumulation of internal-layer nodes, as more and more examples are experienced. Further, an exemplar model employs a local representation—one internal node for each input pattern—rather than a distributed representation. Section 4.2 described how distributed representations are often better choices when it comes to modeling stimulus generalization, although some exemplar models include strategies to allow generalization.[15]

Fortunately, there is an alternative approach that involves creating configural nodes that are part of a distributed representation. This approach, called the **error backpropagation algorithm,** is one of the most powerful learning algorithms ever developed—and one of the most widely used in neural network applications in psychology and engineering.[16] The next section describes this algorithm.

Learning New Representations in Multilayer Networks

In the previous sections, we discussed how the addition of internal-layer nodes in a network can allow for stimulus generalization and configural learning. In both cases, we assumed that the weights from the input layer to internal layer nodes were preset to allow the internal-layer nodes to respond to useful configurations. This strategy is fine for a very simple problem in which the optimal internal-layer representation is known *a priori*. However, if a problem is even moderately complex, it is rarely obvious how the lower-layer weights should be set.

On the other hand, it turns out to be a very thorny mathematical problem to have the network discover its own lower-layer weights. The Widrow-Hoff rule (embodied in the Rescorla-Wagner model) provides a simple mechanism for changing weights based on the error between actual learned response and the desired output (an accurate prediction of the US). This works fine for the upper layer of weights in a multilayer network because the desired output for these nodes is provided by the presence or absence of the US. However, there is no clear teacher for determining the desired pattern of activations at the internal layer of nodes, and therefore, there is no simple way to train the lower layer of weights using the Widrow-Hoff rule. Thus, *although the Widrow-Hoff rule works very well for networks with only a single layer of weights, it does not apply directly to multilayer networks*.

Widrow and Hoff were well aware of the limitations of their learning rule, but at the time of their original work, no solution was available. In 1969, Marvin Minsky and Seymour Papert published an influential book called *Perceptrons* (an early name for neural networks) arguing that the inability to train multilayer networks was serious enough to make neural networks all but useless as computational tools.[17]

What was missing back then was a way to train multilayer networks. Specifically, how could error correction learning be applied to internal-layer nodes, for which no desired output signal was available? Subsequently, an approach known as the error backpropagation algorithm was developed. It says that the error at a given internal-layer node is a function of the error at all the output layer nodes to which that internal-layer node connects. In effect, the error at the output layer is propagated backward to the internal layer.

The mathematical specification of this algorithm is fairly complex, and an outline of the procedure is given in MathBox 4.1. However, a simple example will suffice to illustrate the general principles. Consider again the negative patterning problem (tone → US, light → US, tone&light → no US), and the network shown in figure 4.12A with four internal-layer nodes labeled A, B, C, and D. The backpropagation algorithm assumes that all weights are initialized to small, random values, represented in the figure by connections of varying thickness. On the first trial, a light is presented (figure 4.12B); it causes some random pattern of activation in the internal-layer nodes, depending on the random weights; in this example, node A is strongly activated, nodes B and D are weakly activated, and node C is only minimally activated. Activation from the internal-layer nodes feeds up to partially activate the output node, producing a response of 0.3.

Now, it is time to train the weights to reduce output error. Since the US was present on this trial, the desired output is 1.0, and—just as in the Widrow-Hoff rule—the error is therefore $1.0 - 0.3 = 0.7$. This error is distributed among the upper-layer weights according to a weight change rule similar to the Widrow-Hoff rule (figure 4.12C): the weight from the most active node A to the output node is strongly increased; the weights from B and D are moderately increased; and the weight from C is increased only a minimal amount.

Now the lower-layer of weights are updated (figure 4.12D). Internal-layer node A was highly active and also has a strong weight to the output node; therefore, it influenced activity in the output node a great deal, so it deserves a large share of the blame for the output node's error. But, A's activation reflects the weights feeding into it, so they should share the blame. As a result, the weights coming into A are changed substantially. In this case, the weight from the active light input node is strengthened. This means that next time the light is presented, node A will be activated more strongly, and it in turn will cause more output activation. This process is repeated for the other internal layer nodes. For example, node C was minimally active and also had a small weight to the output node; therefore, it could not have exerted much influence over the output; the lower-layer weights coming into C are therefore changed only minimally. Once all the lower-layer weights are updated, the trial finishes; a new stimulus is presented, and the cycle begins again.

After a large number of training trials, the network settles into a state in which it gives the correct output to each input stimulus. Figure 4.12E shows one such final state. Presentation of the light alone activates node A strongly, which in turn produces output activation; presentation of the tone alone

MathBox 4.1 Error Backpropagation

The error backpropagation algorithm extends the Widrow-Hoff learning rule to deal with multilayer networks. A canonical version of the backpropagation algorithm was described by Rumelhart, Hinton, and Williams (1986).

Each trial is divided into two phases: a *forward* pass, during which all nodes process inputs and produce outputs, and a *backward* pass, during which nodes compute errors and adapt weights.

In the forward pass, each node computes activation and output, much as in the Widrow-Hoff rule (MathBox 3.1):

$$V_j = \sum_i o_i w_{ij} + \theta_j \tag{4.1}$$

$$o_j = f(V_j) \tag{4.2}$$

Equations 4.1 and 4.2 apply to both internal-layer and output-layer nodes. Equation 4.1 includes a *bias* term θ_j that represents the node's baseline tendency to become active, independent of external inputs. The Widrow-Hoff activation rule (Equation 3.1) is a special case of this rule, in which there is no bias term.

The *output function f* is usually defined as a sigmoid function, for example,

$$f(V_j) = \frac{1}{1 + e^{-V_j}} \tag{4.3}$$

The graphical representation of this sigmoid function is shown below. One implication of a sigmoidal function is that node output is clipped at a maximum 1.0 and minimum 0.0.

Once all nodes have computed their output, the forward phase is completed, and the backward, weight-adapting phase begins.

For output nodes, error ∂_j is defined as

$$\partial_j = (d_j - o_j)f'(V_j) \tag{4.4}$$

Here, d_j is the desired output of node *j* and o_j is the actual output; f' is the first derivative of the output function *f*. Using the sigmoidal output function of Equation 4.3, the first derivative is

$$f'(V_j) = V_j(1-V_j) \tag{4.5}$$

Note that if a different output function is used, that is, $f(x) = x$, then $f'(x) = 1.0$, and Equation 4.4 reduces to the Widrow-Hoff error computation rule (Equation 3.4).

For internal-layer nodes, desired output d_j is not defined. Instead, the error at internal-layer node *j* is estimated by backpropagating some of the error from output nodes *k* to which *j* connects:

$$\partial_j = \left(\sum_k \partial_k w_{jk} \right) f'(V_j) \tag{4.6}$$

Once error is calculated for all nodes, all nodes in both the internal and output layers update weights according to the rule

$$\Delta w_{ij} = \beta \partial_j o_i \tag{4.7}$$

where β is a learning rate (typically between 0.01 and 0.1) and o_i is the output of node *i.* Again, this is the same as the Widrow-Hoff learning rule (Equation 3.5).

Most implementations of error backpropagation also include a *momentum* term, α. In this case, the weight change on each trial is influenced not only by the current error, but also by previous weight changes:

$$\Delta w_{ij} = \beta \partial_j o_i + \alpha \Delta w_{ij}{}^* \tag{4.8}$$

The momentum α is generally set less than 1.0, and $\Delta w_{ij}{}^*$ is the amount by which the weight was changed on the previous trial. Use of momentum tends to speed overall learning in the network by ensuring that weights change smoothly, in one direction, across trials, rather than zigzagging up and down on alternate trials.

The interested reader should consult the following for a complete derivation of the backpropagation algorithm, its uses and shortcomings: Rumelhart, Hinton, & Williams (1986); Stone (1986); Hinton (1989, 1992); Stork (1989); Minsky & Papert (1998); and Reed & Marks (1999).

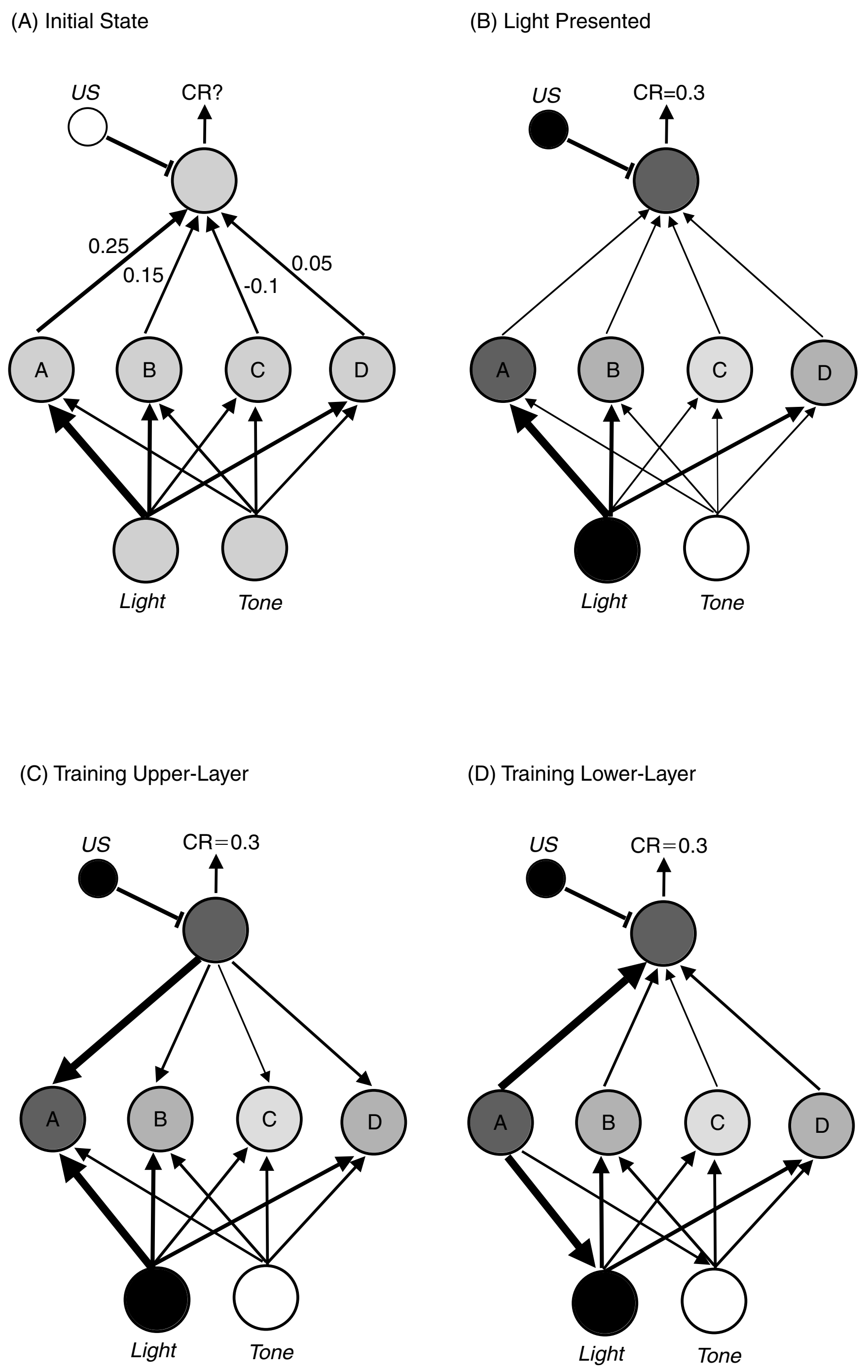
activates node D strongly, which in turn produces output activation; presentation of the tone and the light together also activates nodes B and C, which inhibit output activation. Thus, the network correctly responds to the individual cues but not to the compound stimulus, thereby correctly solving the negative patterning task.

There are actually many ways in which the network of figure 4.12 could have solved the negative patterning problem. In the example shown, internal-layer node A was set to respond to the tone and C was set to respond to the light. But the mapping could have just as easily been reversed, and the problem would still have been learned in the same basic way. Very different solutions are possible too. Which particular solution is found depends on the number of internal-layer nodes, the network learning rate, and even the order of stimulus presentation. Even if all these things are held constant, the way in which the random initial weights are set will influence the final outcome. If the network in figure 4.12A is reinitialized with a different set of starting values for the weights and then trained on the negative patterning problem again, it may find a slightly different—but possibly equally valid—solution, such as the one shown in figure 4.12F. As a result, when neural network researchers present a learning curve summarizing the network performance (e.g., figure 4.13), the results are usually an average of the performance of a number of different experiments (called **simulation runs**) with the same network but different initial weights.

Computational Limitations of Backpropagation

The backpropagation algorithm has several shortcomings. First, the algorithm is relatively complex, as a glance at MathBox 4.1 will confirm, and this complexity slows learning. The network of figure 4.12, which includes only two input nodes and three stimulus patterns, may still require about 500 trial blocks, each of which includes one example of each stimulus type, to learn this task.* A network with only two internal-layer nodes can require anywhere from a few hundred to a few thousand training trials to learn the negative patterning problem.[18] Learning can be faster if the mathematical parameters (defined in MathBox 4.1) are set appropriately, but it is sometimes hard to know in advance what is an appropriate setting. By contrast, the configural-cue network in figure 4.10, with its hand-coded lower-layer

*The exact learning time for a network that is trained with error backpropagation will vary according to several of the parameters outlined in MathBox 4.1, as well as with the network architecture.


(A) Initial State
US
CR?
0.25
0.15
-0.1
0.05
A
B
C
D
Light
Tone
(B) Light Presented
US
CR=0.3
A
B
C
D
Light
Tone
(C) Training Upper-Layer
US
CR=0.3
A
B
C
D
Light
Tone
(D) Training Lower-Layer
US
CR=0.3
A
B
C
D
Light
Tone

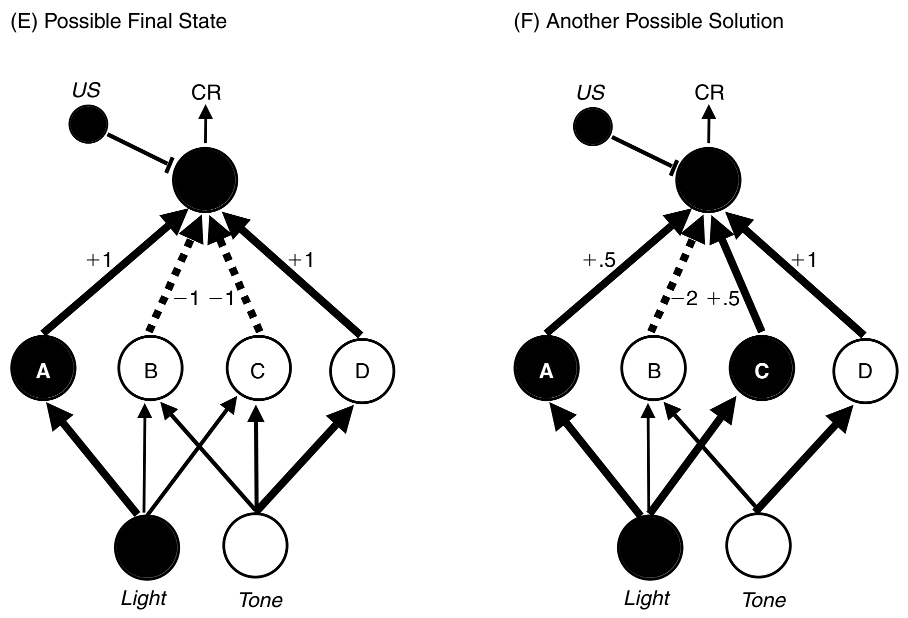


Figure 4.12 (A) A multilayer network with four internal-layer nodes A, B, C, and D. All weights are initialized to small random values. To avoid clutter, only the weights on the upper layer are shown; small random values are set on all lower-layer weights as well. (B) A light CS is presented. Because of the random weights, this stimulus causes strong activity in internal-layer node A, some activity in B and D, and little activity in C. This pattern of activation feeds into the output node, which produces a response (in this example, the response is 0.3). (C) Training the upper layer weights is similar to the Widrow-Hoff rule. The error is computed as the difference between desired output (US = 1) and actual output (CR = 0.3). A fraction of this error is distributed among all active internal-layer nodes; since A was the most active, it receives most of the weight change. In the future, when node A is again active, it will now produce more activation in the output node. (D) The lower-layer weights are changed by backpropagating error from the upper layer. Node A, which was highly active, receives a large fraction of the error. This is distributed as weight change along the connections feeding into A. In this case, the connection to the active light node is strengthened; the tone node was inactive, so there is no change to this connection. Next time the light stimulus is present, it will cause more activity in A, which will in turn cause a stronger output. A similar process is repeated for each of the remaining internal-layer nodes B, C, and D. (E) After numerous training trials on the negative patterning problem, the network might find a solution as shown: When the tone alone is present, node A is activated and in turn activates the output node. When the light alone is present, node D is activated and in turn activates the output node. When both tone and light are present, configural nodes B and C are also activated, and inhibitory output from these nodes counteracts the excitatory output from A and D; the output node does not become active. This is a similar solution to the one constructed by hand in figure 4.10 except for the presence of two configural nodes (B and C) instead of one. (F) If the same network from figure 4.12A had been initialized with different random weights, it might have developed a different but equally valid solution. For example, the light alone might activate nodes A and C, the tone alone might activate node D, and node B might respond to the configuration. This network might show a learning curve indistinguishable from the one generated by the network in (E).

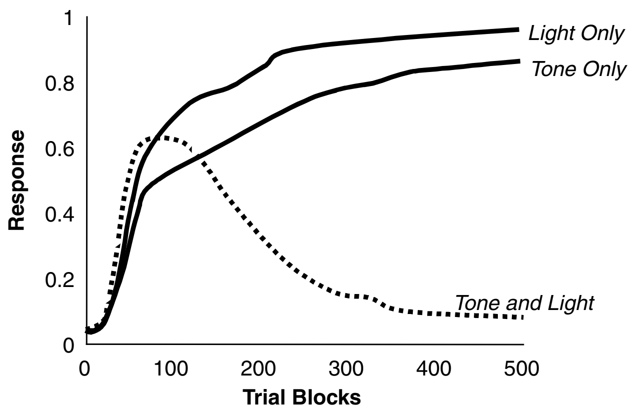


Figure 4.13 Learning curve, averaged over ten simulation runs for a network with the architecture shown in Figure 4.12A, trained on the negative patterning problem (tone → US, light → US, tone & light → no US). Each trial block includes one presentation of each stimulus type. Out of the ten simulation runs, only one failed to learn to respond to light and tone but not the compound; even after 1000 trial blocks, this network was still responding more strongly to the compound than to the components. Compare to the experimental results from rabbits shown in figure 4.8A.

weights, can learn the same problem in a few dozen trials. Thus, there is a cost to flexibility in representations: The more flexible the network, the longer it takes to construct new representations and hence the longer it takes to learn.

A second limitation of the error backpropagation algorithm is that it does not always work. Figure 4.13 shows performance on the negative patterning problem averaged over ten simulation runs. Out of these ten simulation runs, one failed to reach a solution: It persistently responded more to the compound stimulus than to the components, even after 5000 training trials. In such cases, the only solution is to stop the network, wipe its memory by returning all weights to initial random values, and begin training again, hoping for better results. This is not necessarily a fatal problem for the model as a psychological theory of animal and human learning; in most psychology learning experiments, a small number of the subjects (animals or people) often fail to solve complex training tasks. Like the networks, they seem to get stuck and never find a solution.

A further concern about the backpropagation algorithm is that it involves a large number of parameters, including the number of internal-layer nodes, the learning rate, and others (interested readers can refer to MathBox 4.1 for details). Each of these parameters may be adjusted to improve how well the network learns. The optimal values for these parameters are generally different for each task and therefore must often be chosen and fine-tuned by hand. A great deal of research has been devoted to optimizing performance on a particular problem by changing some parameter in the algorithm, but these solutions often apply well to one specific problem (or problem domain) without applying well to other problems (or problem domains).

Despite all these limitations on performance, the backpropagation algorithm has proven extremely useful as an engineering tool. In fact, given enough internal-layer nodes and enough training trials, a network that is trained by backpropagation can, in principle, learn any arbitrary relationship from a set of inputs to a set of outputs.[19]

The backpropagation algorithm has been used to train networks for all kinds of practical applications, including networks that detect forged signatures, pronounce typewritten text, discover faults in mechanical equipment, interpret sonar returns, and diagnose diseases.[20] Backpropagation is probably the single most widely used algorithm for training neural networks in business and engineering applications. Countless algorithmic variations have been developed to address various shortcomings or to speed learning.[21] However, these variations typically increase mathematical complexity while producing only moderate improvements in performance. Therefore, the canonical version of backpropagation described here remains the standard method for training most multilayer networks. In fact, when researchers announce that they are "using a neural network" for a particular application, it is usually a safe bet that they are employing the error backpropagation algorithm.

Psychological and Biological Validity of Backpropagation

Backpropagation's success as an engineering tool does not necessarily imply anything about its validity as a psychological model of learning or as a model of the biological mechanisms underlying that learning. Given that the backpropagation algorithm can produce complex learning behaviors, it is natural to wonder how well this learning procedure succeeds as a psychological or biological model of learning.

On the surface, there is an immediate difficulty with this comparison. The backpropagation algorithm requires that nodes be able to send error information backward to the nodes that originally provided their input (refer to figure 4.12D). This is problematic for models of biological learning: The connections in the brain are so complex and extensive that it seems unlikely that there could be precise reciprocity in neuronal connections; these and other biological features are often conveniently ignored by neural network modelers.[22] Several authors have suggested ways in which something like backpropagation could be implemented without requiring such precise backward connections.[23] More recent neurophysiological studies have suggested that neurons may indeed be able to send information backward to the neurons that activated them.[24] While backpropagation may not be plausible as a literal model of neuronal learning, it may still be useful as a systems-level description of learning in a brain network.

In the next chapter, we will discuss ways in which a slightly different kind of network architecture—the autoassociator and its derivative, the autoencoder—can capture some aspects of brain anatomy and physiology, particularly in the hippocampus.

SUMMARY

• Generalization refers to the tendency to transfer learning about one stimulus to other, superficially similar stimuli. Generalization is critical to apply prior experience in new situations without requiring fresh learning about every new instance. Conversely, specificity allows learning different responses to stimuli that may superficially look very similar. The optimal degree of generalization versus specificity therefore depends not only on the superficial similarity of stimuli, but also on their meaning or consequences for a user.

• Component representations relate one stimulus (or feature) to one node in the model. In distributed representations, information about each stimulus (or feature) is encoded by many nodes. Distributed representations provide a framework for encoding stimulus similarity and allow stimulus generalization.

• Distributed representations can be implemented by adding an internal layer of nodes to a neural network, between the input and output nodes.

• Although the Widrow-Hoff rule works very well for networks with only a single layer of weights, it does not apply directly to multilayer networks. The error backpropagation algorithm extends the Widrow-Hoff rule to allow changes to the lower layer of weights, thereby changing stimulus representations in the internal layer.