Knowledge in Logic Programming

An important concern in representing and storing knowledge is that it be done in a way that is compatible with the system that must access the knowledge. It is in this context that logic programming (see Section 6.7) often proves beneficial. In such systems knowledge is represented by "logic" statements such as

Dumbo is an elephant.

and

X is an elephant implies X is gray.

Such statements can be represented using notational systems that are readily accessible to the application of inference rules. In turn, sequences of deductive reasoning, such as we saw in Figure 11.5, can be implemented in a straightforward manner. Thus, in logic programming the representation and storage of knowledge are well integrated with the knowledge extraction and application process. One might say that logic programming systems provide a "seamless" boundary between stored knowledge and its application.

The functions in this starting collection form the "gene pool" from which future generations of programs will be constructed. One then allows the evolutionary process to run for many generations, hoping that by producing each generation from the best performers in the previous generation, a solution to the target problem will evolve.

Finally, we should recognize a phenomenon that is closely related to learning: discovery. The distinction is that learning is "target based" whereas discovery is not. The term *discovery* has a connotation of the unexpected that is not present in learning. We might set out to learn a foreign language or how to drive a car, but we might discover that those tasks are more difficult than we expected. An explorer might discover a large lake, whereas the goal was merely to learn what was there.

Developing agents with the ability to discover efficiently requires that the agent be able to identify potentially fruitful "trains of thought." Here, discovery relies heavily on the ability to reason and the use of heuristics. Moreover, many potential applications of discovery require that an agent be able to distinguish meaningful results from insignificant ones. A data mining agent, for example, should not report every trivial relationship it finds.

Examples of success in computer discovery systems include Bacon, named after the philosopher Sir Francis Bacon, that has discovered (or maybe we should say "rediscovered") Ohm's law of electricity, Kepler's third law of planetary motion, and the conservation of momentum. Perhaps more persuasive is the system AUTOCLASS that, using infrared spectral data, has discovered new classes of stars that were previously unknown in astronomy—a true scientific discovery by a computer.

Questions & Exercises

1. What is meant by the term *real-world knowledge,* and what is its significance in artificial intelligence?
2. A database about magazine subscribers typically contains a list of subscribers to each magazine but does not contain a list of those who do not subscribe. How, then, does such a database determine that a person does not subscribe to a particular magazine?
3. Summarize the frame problem.
4. Identify three ways of training a computer. Which one does not involve direct human intervention?
5. How do evolutionary techniques differ from more traditional computer learning techniques?

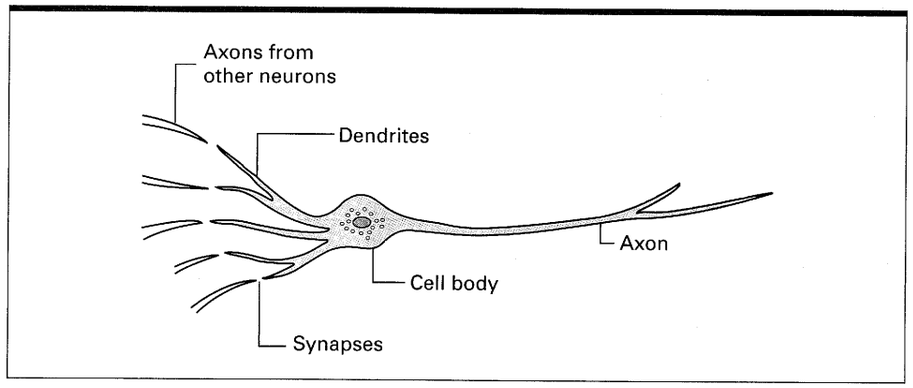
11.5 Artificial Neural Networks

With all the progress that has been made in artificial intelligence, many problems in the field continue to tax the abilities of computers based on the von Neumann architecture. Central processing units that execute sequences of instructions do not seem capable of perceiving and reasoning at levels comparable to those of the human mind. For this reason, many researchers are turning to machines with other architectures. One of these is the artificial neural network.

Basic Properties

As introduced in Chapter 2, artificial neural networks are constructed from many individual processors, which we will call **processing units** (or just units for short), in a manner that models networks of neurons in living biological systems. A biological neuron is a single cell with input tentacles called dendrites and an output tentacle called the axon (Figure 11.15). The signals transmitted via a cell's axon

Figure 11.15 A neuron in a living biological system



reflect whether the cell is in an inhibited or excited state. This state is determined by the combination of signals received by the cell's dendrites. These dendrites pick up signals from the axons of other cells across small gaps known as synapses. Research suggests that the conductivity across a single synapse is controlled by the chemical composition of the synapse. That is, whether the particular input signal will have an exciting or inhibiting effect on the neuron is determined by the chemical composition of the synapse. Thus it is believed that a biological neural network learns by adjusting these chemical connections between neurons.

A processing unit in an artificial neural network is a simple device that mimics this basic understanding of a biological neuron. It produces an output of 1 or 0, depending on whether its effective input exceeds a given value, which is called the processing unit's **threshold** value. This effective input is a weighted sum of the actual inputs, as represented in Figure 11.16. In this figure, the outputs of three processing units (denoted by v_1, v_2, and v_3) are used as inputs to another unit. The inputs to this fourth unit are associated with values called **weights** (denoted by w_1, w_2, and w_3). The receiving unit multiplies each of its input values by the weight associated with that particular input position and then adds these products to form the effective input ($v_1w_1 + v_2w_2 + v_3w_3$). If this sum exceeds the processing unit's threshold value, the unit produces an output of 1 (simulating a neuron's excited state); otherwise the unit produces a 0 as its output (simulating an inhibited state).

Following the lead of Figure 11.16, we adopt the convention of representing processing units as rectangles. At the input end of the unit, we place a smaller rectangle for each input, and in this rectangle we write the weight associated with that input. Finally, we write the unit's threshold value in the middle of the large rectangle. As an example, Figure 11.17 represents a processing unit with three inputs and a threshold value of 1.5. The first input is weighted by the value -2, the second is weighted by 3, and the third is weighted by -1. Therefore if the unit receives the inputs 1, 1, and 0, its effective input is $(1)(-2) + (1)(3) + (0)(-1) = 1$, and thus its

Figure 11.16 The activities within a processing unit

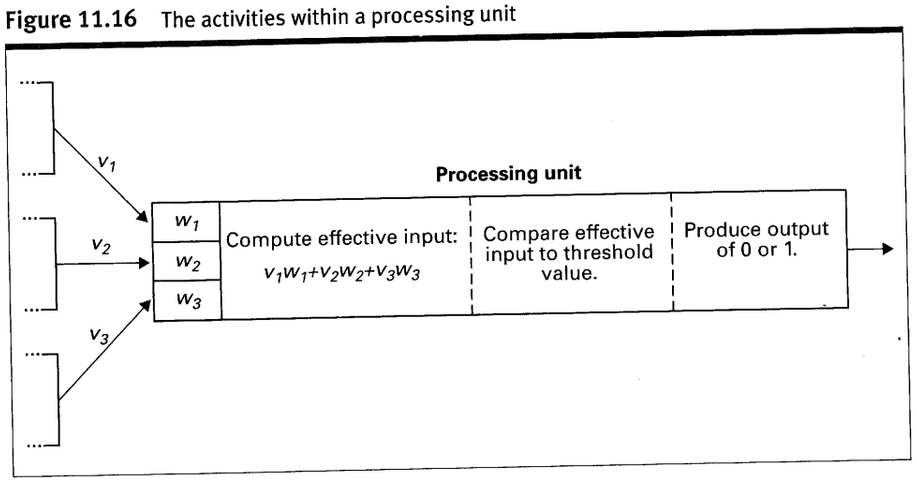
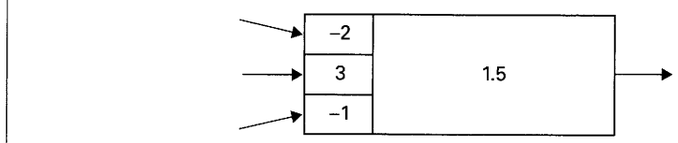


Figure 11.17 Representation of a processing unit

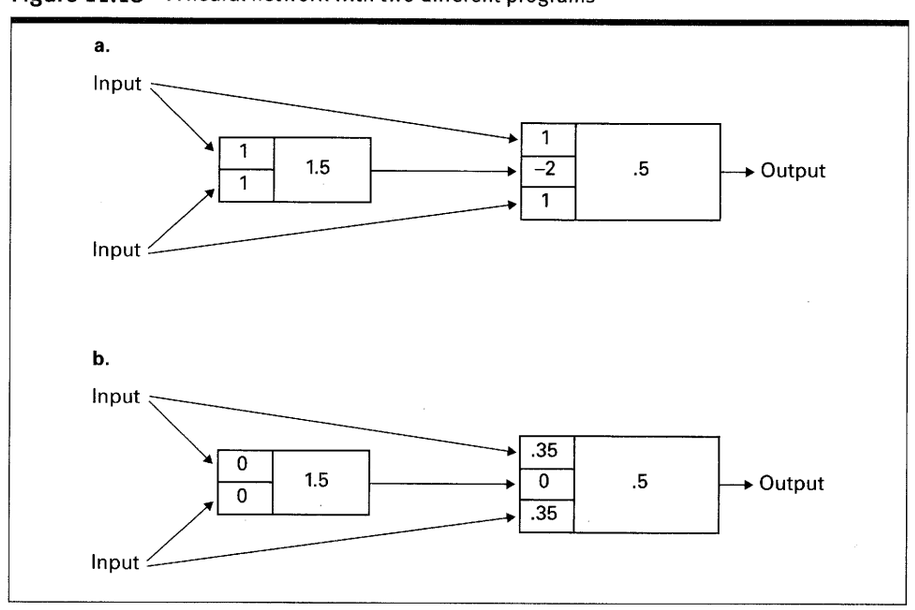


output is 0. But, if the unit receives 0, 1, and 1, its effective input is $(0)(-2) + (1)(3) + (1)(-1) = 2$, which exceeds the threshold value. The unit's output will thus be 1.

The fact that a weight can be positive or negative means that the corresponding input can have either an inhibiting or exciting effect on the receiving unit. (If the weight is negative, then a 1 at that input position reduces the weighted sum and thus tends to hold the effective input below the threshold value. In contrast, a positive weight causes the associated input to have an increasing effect on the weighted sum and thus increase the chances of that sum exceeding the threshold value.) Moreover, the actual size of the weight controls the degree to which the corresponding input is allowed to inhibit or excite the receiving unit. Consequently, by adjusting the values of the weights throughout an artificial neural network, we can program the network to respond to different inputs in a predetermined manner.

As an example, the simple network presented in Figure 11.18a is programmed to produce an output of 1 if its two inputs differ and an output of 0 otherwise. If,

Figure 11.18 A neural network with two different programs



however, we change the weights to those shown in Figure 11.18b, we obtain a network that responds with a 1 if both of its inputs are 1s and with a 0 otherwise.

We should note that the network in Figure 11.18 is far more simplistic than an actual biological network. A human brain contains approximately 10^{11} neurons with about 10^4 synapses per neuron. Indeed, the dendrites of a biological neuron are so numerous that they appear more like a fibrous mesh than the individual tentacles represented in Figure 11.15.

Training Artificial Neural Networks

An important feature of artificial neural networks is that they are not programmed in the traditional sense but instead are trained. That is, a programmer does not determine the values of the weights needed to solve a particular problem and then "plug" those values into the network. Instead, an artificial neural network learns the proper weight values via supervised training (Section 11.4) involving a repetitive process in which inputs from the training set are applied to the network and then the weights are adjusted by small increments so that the network's performance approaches the desired behavior. How the weights should be adjusted is the subject of research. What is needed is a strategy for modifying the weights so that each new adjustment leads toward the overall goal rather than destroying the progress made in the previous steps.

To demonstrate the problem, consider the task of training the network in Figure 11.19 (in which all the weights are set to the value 0) to produce an output of 1 exactly when its inputs are different. That is, we want the input patterns 1, 0 and 0, 1 to produce the output 1 while the input patterns 0, 0 and 1, 1 produce the output value 0. (We have already seen a solution to this problem in Figure 11.18a.) Let us begin the training process by assigning both inputs the value 1. We observe that the output is 0 (Figure 11.20a), which is the desired behavior, so we leave the network as it is and continue the training process by trying the input pattern 1, 0 (Figure 11.20b). This produces the output 0, whereas we want the output to be 1. Let us fix this by changing the upper weight of the second processing unit to 1 (Figure 11.20c). Now the network performs correctly for the input pattern 1, 0. At this point, we go back and retry the input pattern 1, 1. To our dismay, the network

Figure 11.19 An artificial neural network

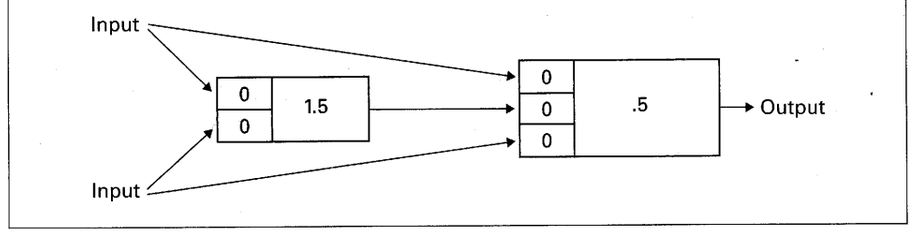
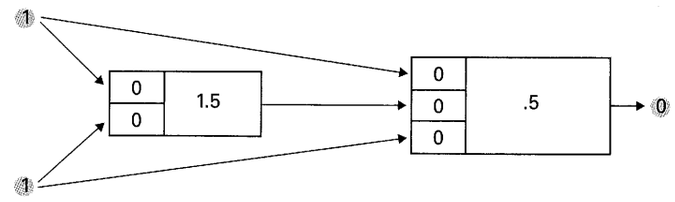
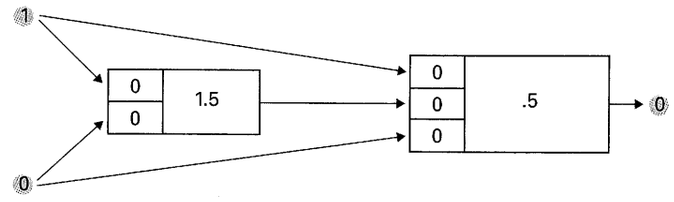


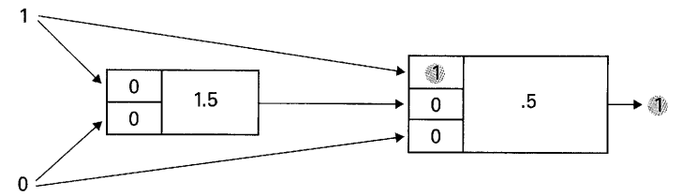
Figure 11.20 Training an artificial neural network



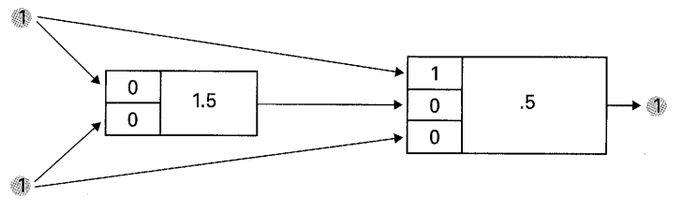
a. The network performs correctly for the input pattern 1, 1.



b. The network performs incorrectly for the input pattern 1, 0.



c. The upper weight in the second processing unit is adjusted.



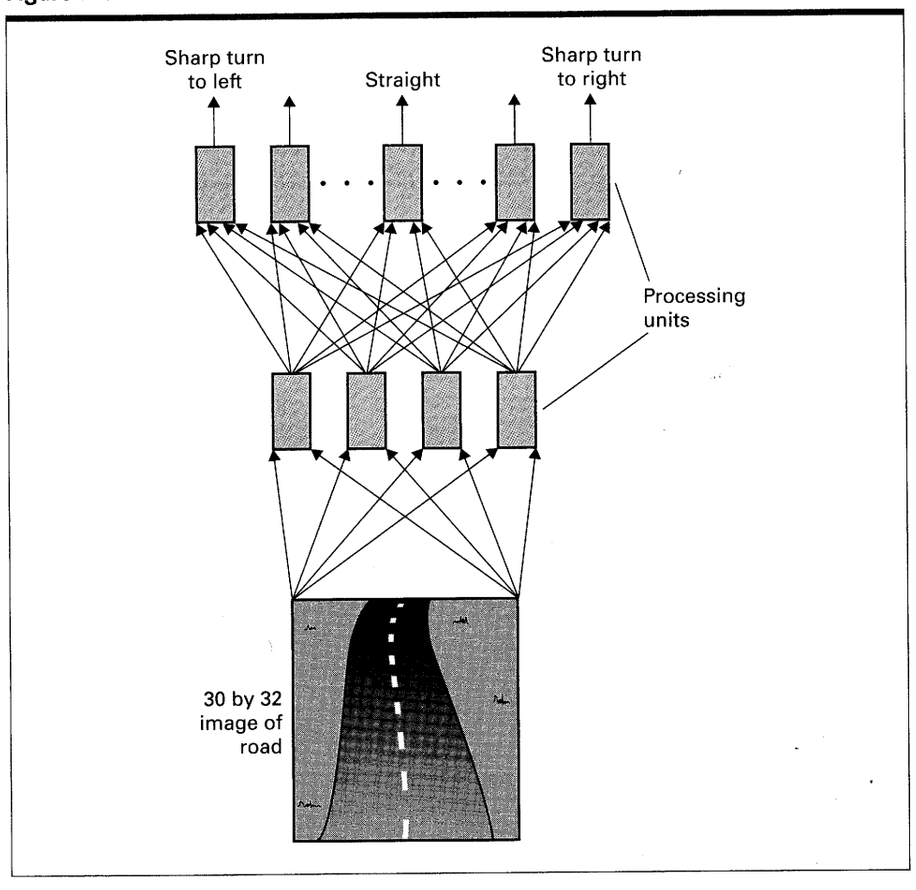
d. However, the network no longer performs correctly for the input pattern 1, 1.

no longer processes that pattern correctly. Indeed, it now produces an output of 1 (Figure 11.20d). Let us fix this by changing the upper weight in the second processing unit back to 0. Alas, we are back where we started, and continuing this process will merely lead us through an endless training cycle as each correction we make counteracts the previous correction.

Fortunately, significant progress has been made in the development of successful training strategies, as testified by the ALVINN project cited in the previous section. Indeed, ALVINN was an artificial neural network whose composition was surprisingly simple (Figure 11.21). Its input was obtained from a 30 by 32 array of sensors, each of which observed a unique portion of the video image of the road ahead and reported its findings to each of four processing units. (Thus, each of these four units had 960 inputs.) The output of each of these four units was connected to each of 30 output units, whose outputs indicated the direction to steer. Excited processing units at one end of the 30 unit row indicated a sharp turn to the left, while excited units at the other end indicated a sharp turn to the right.

ALVINN was trained by "watching" a human drive while it made its own steering decisions, comparing its decisions to those of the human, and making

Figure 11.21 The structure of ALVINN (Autonomous Land Vehicle in a Neural Net)



slight modifications to its weights to bring its decisions closer to those of the human. There was, however, an interesting side issue. Although ALVINN learned to steer following this simple technique, ALVINN did not learn how to recover from mistakes. Thus, the data collected from the human was artificially enriched to include recovery situations as well. (One approach to this recovery training that was initially considered was to have the human swerve the vehicle so that ALVINN could watch the human recover and thus learn how to recover on its own. But unless ALVINN was disabled while the human performed the initial swerve procedure, ALVINN learned to swerve as well as to recover—an obviously undesirable trait.)

Associative Memory

The human mind has the amazing ability to retrieve information that is associated with a current topic of consideration. When we experience certain smells, we might readily recall memories of our childhood. The sound of a friend's voice might conjure an image of the person or perhaps memories of good times. Certain music might generate thoughts of particular holiday seasons. These are examples of **associative memory**—the retrieval of information that is associated with, or related to, the information at hand.

To construct machines with associative memory has been a goal of research for many years. One approach is to apply techniques of artificial neural networks. For instance, consider a network consisting of many processing units that are interconnected to form a web with no inputs or outputs. (In some designs, called Hopfield networks, the output of each processing unit is connected as inputs to each of the other units; in other cases the output of a unit may be connected only to its immediate neighbors.) In such a system, the excited units will tend to excite other units, whereas the inhibited units will tend to inhibit others. In turn, the entire system may be in a constant state of change, or it may be that the system will find its way to a stable configuration where the excited units remain excited and the inhibited units remain inhibited. If we start the network in a nonstable configuration that is close to a stable one, we would expect it to wander to that stable configuration. In a sense, when given a part of a stable configuration, the network might be able to complete the configuration.

Now suppose that we represent an excited state by 1 and an inhibited state by 0 so that the condition of the entire network at any time can be envisioned as a configuration of 0s and 1s. Then, if we set the network to a bit pattern that is close to a stable pattern, we could expect the network to shift to the stable pattern. In other words, the network might find the stable bit pattern that is close to the pattern it was given. Thus if some of the bits are used to encode smells and others are used to encode childhood memories, then initializing the smell bits according to a certain stable configuration could cause the remaining bits to find their way to the associated childhood memory.

Now consider the artificial neural network shown in Figure 11.22. Each circle in the figure represents a processing unit whose threshold value is recorded inside the circle. The lines connecting circles represent two-way connections between the corresponding units. That is, a line connecting two units indicates that the output of each unit is connected as an input to the other. Thus the output of the center unit is connected as an input to each of the units around the perimeter, and the output of each of the units around the perimeter is connected as an input to the center unit as well as an input to each of its immediate neighbors on the perimeter. Two connected units associate the same weight with each other's output. This common weight is recorded next to the line connecting the units. Thus the unit at the top of the diagram associates a weight of -1 with the input it receives from the center unit and a weight of 1 with the inputs it receives from its two neighbors on the perimeter. Likewise, the center unit associates a weight of -1 with each of the values it receives from the units around the perimeter.

The network operates in discrete steps in which all processing units respond to their inputs in a synchronized manner. To determine the next configuration of the network from its current configuration, we determine the effective inputs of each unit throughout the network and then allow all the units to respond to their inputs at the same time. The effect is that the entire network follows a coordinated sequence of compute effective inputs, respond to inputs, compute effective inputs, respond to inputs, etc.

Consider the sequence of events that would occur if we initialized the network with its two rightmost units inhibited and the other units excited (Figure 11.23a). The two leftmost units would have effective inputs of 1, so they would remain excited. But, their neighbors on the perimeter would have effective inputs of 0, so they would become inhibited. Likewise, the center unit would have an effective input of -4, so it would become inhibited. Thus the entire network would shift to

Figure 11.22 An artificial neural network implementing an associative memory

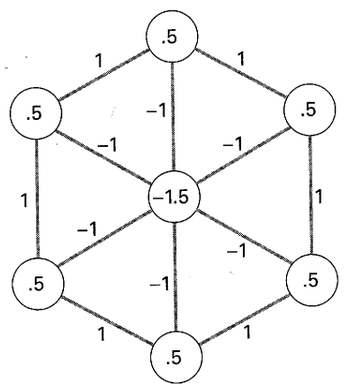
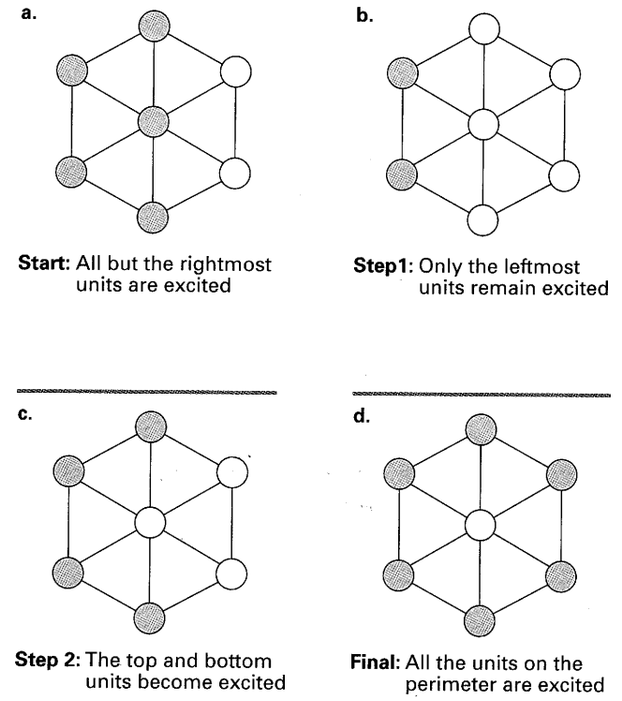


Figure 11.23 The steps leading to a stable configuration



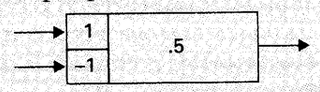
the configuration shown in Figure 11.23b in which only the two leftmost units are excited. Since the center unit would now be inhibited, the excited conditions of the leftmost units would cause the top and bottom units to become excited again. Meanwhile, the center unit would remain inhibited since it would have an effective input of -2. Thus the network would shift to the configuration in Figure 11.23c, which would then lead to the configuration in Figure 11.23d. (You might wish to confirm that a blinking phenomenon would occur if the network were initialized with only the upper four units excited. The top unit would remain excited while its two neighbors on the perimeter and the center unit would alternate between being excited and inhibited.)

Finally, observe that the network has two stable configurations: one in which the center unit is excited and the others are inhibited, and another configuration in which the center unit is inhibited and the others are excited. If we initialize the network with the center unit excited and no more than two of the other units excited, the network will wander to the former stable configuration. If we initialize the network with at least four adjacent units on the perimeter in their excited

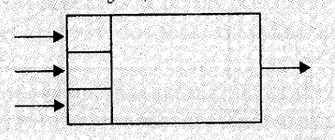
states, the network will wander to the latter configuration. Thus we could say that the network associates the former stable configuration with initial patterns in which its center unit and fewer than three of its perimeter units are excited, and associates the latter stable configuration with initial patterns in which four or more of its perimeter units are excited. In short, the network represents an elementary associative memory.

Questions & Exercises

1. What is the output of the following processing unit when both its inputs are 1s? What about the input patterns 0, 0; 0, 1; and 1, 0?



2. Adjust the weights and threshold value of the following processing unit so that its output is 1 if and only if at least two of its inputs are 1s.



3. Identify a problem that might occur in training an artificial neural network.
4. To which stable configuration will the network in Figure 11.22 wander if it is initialized with all its processing units inhibited?

11.6 Robotics

Robotics is the study of physical, autonomous agents that behave intelligently. As with all agents, robots must be able to perceive, reason, and act in their environment. Research in robotics thereby encompasses all areas of artificial intelligence as well as drawing heavily from mechanical and electrical engineering.

To interact with the world, robots need mechanisms to manipulate objects and to move about. In the early days of robotics, the field was closely allied with the development of manipulators, most often mechanical arms with elbows, wrists, and hands or tools. Research dealt not only with how such devices could be maneuvered but also with how knowledge of their location and orientation could be maintained and applied. (You are able to close your eyes and still touch your nose with your finger because your brain maintains a record of where your nose and finger are.) Over time robots arms have become more dexterous to where, with a sense of touch based on force feedback, they can handle eggs and paper cups successfully.

Recently, the development of faster, lighter weight computers has lead to greater research in mobile robots that can move about. Achieving this mobility has led to an abundance of creative designs. Researchers in robot locomotion have developed robots that swim like fish, fly like dragonflies, hop like grasshoppers, and crawl like snakes.

Wheeled robots are very popular since they are relatively easy to design and build, but they are limited in the type of terrain they can traverse. Overcoming this restriction, using combinations of wheels or tracks to climb stairs or roll over rocks, is the goal of current research. As an example, the NASA Mars rovers used specially designed wheels to move on rocky soil.

Legged robots offer greater mobility but are significantly more complex. For instance, two-legged robots, designed to walk as humans, must constantly monitor and adjust their stance or they will fall. However, such difficulties can be overcome, as exemplified by the two-legged humanoid robot named Asimo, developed by Honda, that can walk up stairs and even run.

Despite great advances in manipulators and locomotion, most robots are still not very autonomous. Industrial robot arms are typically rigidly programmed for each task and work without sensors, assuming parts will be given to them in exact positions. Other mobile robots such as the NASA Mars rovers and military Unmanned Aerial Vehicles (UAVs) rely on human operators for their intelligence.

Overcoming this dependency on humans is a major goal of current research. One question deals with what an autonomous robot needs to know about its environment and to what degree it needs to plan its actions in advance. One approach is to build robots that maintain detailed records of their environments, containing an inventory of objects and their locations with which they develop precise plans of action. Research in this direction depends heavily on progress in knowledge representation and storage as well as improved reasoning and plan-development techniques.

An alternative approach is to develop reactive robots that, rather than maintaining complex records and expending great efforts in constructing detailed plans of action, merely apply simple rules for interacting with the world to guide their behavior moment by moment. Proponents of reactive robotics argue that when planning a long trip by car, humans do not make all-encompassing, detailed plans in advance. Instead, they merely select the major roads, leaving such details as where to eat, what exits to take, and how to handle detours for later consideration. Likewise, a reactive robot that needs to navigate a crowded hallway or to go from one building to another does not develop a highly detailed plan in advance, but instead applies simple rules to avoid each obstacle as it is encountered. This is the approach taken by the best-selling robot in history, the iRobot Roomba vacuum cleaner, which moves about a floor in a reactive mode without bothering to remember the details of furniture and other obstacles. After all, the family pet will probably not be in the same place next time.