Discussion

The combined backpropagation/Cauchy network trains significantly faster than either algorithm alone, and is relatively insensitive to the values of the coefficients. Convergence to a global minimum is guaranteed by the Cauchy algorithm; hundreds of training experiments have produced no case in which the network became trapped in a local minimum. Network paralysis has been solved by the use of a selective weight-compression algorithm that has produced convergence in all tests to date, without materially increasing training time.

Despite these encouraging results, the method is not fully evaluated, especially on large problems. Much more work will be required to determine its advantages and disadvantages.

References

Geman, S., and Geman, D. 1984. Stochastic relaxation, Gibbs distributions and Baysian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6:721–41.

Hinton, G.E., and Sejnowski, T. J. 1986. Learning and relearning in Boltzmann machines. In *Parallel distributed processing*, vol.1, pp. 282–317. Cambridge, MA: MIT Press.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. 1953. Equations of state calculations by fast computing machines. *Journal of Chemistry and Physics* 21:1087–91.

Parker, D. B. 1987. Optimal algorithms for adaptive networks: Second order backpropagation, second order direct propagation, and second order Hebbian learning. In *Proceedings of the IEEE First International Conference on Neural Networks*, eds. M. Caudill and C. Butler, vol. 2, pp. 593–600. San Diego, CA: SOS Printing.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. 1986. Learning internal representations by error propagation. In *Parallel distributed processing*, vol.1, pp. 318–62. Cambridge, MA: MIT Press.

Szu, H., and Hartley, R. 1987. Fast simulated annealing. *Physics Letters* 1222(3,4):157–62.

Wasserman, P. D. 1988. Combined backpropagation/Cauchy machine. *Neural Networks: Abstracts of the First INNS Meeting, Boston 1988*, vol. 1, p. 556. Elmsford, NY: Pergamon Press.

Philip D. Wasserman 1989
Neural Computing Theory and Practice
Van Nostrand Reinhold, New York
Chap. 6

Hopfield Nets

The networks presented in previous chapters are nonrecurrent; that is, there is no feedback from the outputs of the networks to their inputs. The lack of feedback ensures that the networks are unconditionally stable. They cannot enter a mode in which the output wanders interminably from state to state, never producing a usable output. This highly desirable characteristic comes with a price; nonrecurrent networks have a repertoire of behavior that is limited compared to their recurrent kin.

Because recurrent networks have feedback paths from their outputs back to their inputs, the response of such networks is dynamic; that is, after applying a new input, the output is calculated and fed back to modify the input. The output is then recalculated, and the process is repeated again and again. For a stable network, successive iterations produce smaller and smaller output changes until eventually the outputs become constant. For many networks, the process never ends, and such networks are said to be unstable. Unstable networks have interesting properties and have been studied as examples of chaotic systems. The large subject of chaos is outside of the scope of this volume, however. Instead, we concentrate on stable networks, that is, those that eventually produce a constant output.

Stability problems stymied early researchers. No one was able to predict which networks would be stable and which would change continuously. Furthermore, the problem appeared so difficult that

many researchers were pessimistic about finding a solution. Fortunately, a powerful network theorem that defines a subset of the recurrent networks whose outputs eventually reach a stable state has been devised (Cohen and Grossberg 1983). This brilliant accomplishment opened the door to further research, and today many scientists are exploring the complicated behavior and capabilities of these systems.

John Hopfield has made important contributions to both the theory and application of recurrent systems. As a result, some configurations have become known as Hopfield nets. A review of the literature shows that many others have conducted research on these and similar devices; for example, Grossberg (1987) has studied the general properties of networks similar to many of those presented here. The works cited at the end of this chapter are not intended to constitute an exhaustive list of titles on the subject of recurrent systems; rather they are accessible writings that can serve to explain, amplify, and extend the contents of this volume.

RECURRENT NETWORK CONFIGURATIONS

Figure 6-1 shows a recurrent network consisting of two layers. The format is somewhat different from that found in the work of Hopfield and others, however, it is functionally equivalent and ties in well with the networks presented in earlier chapters. Layer 0, as in previous illustrations, serves no computational function; it simply distributes the network outputs back to the inputs. Each layer 1 neuron computes the weighted sum of its inputs, producing a NET signal that is then operated on by the nonlinear function F to yield the OUT signal. These operations are similar to the neurons of other networks (see Chapter 2).

Binary Systems

In Hopfield's early work (1982), the function F was a simple threshold. The output of such a neuron is one if the weighted sum of the outputs of the other neurons is greater than a threshold T_j; otherwise it is 0. It is calculated as follows:

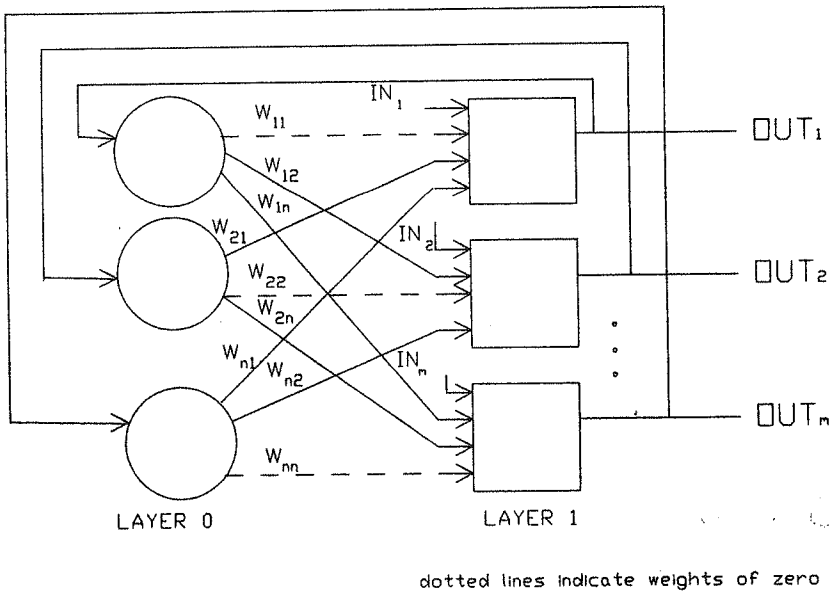

Figure 6-1. Single-Layer Recurrent Network

$$\text{NET}_j = \sum_{i \neq j} w_{ij}\text{OUT}_i + \text{IN}_j \qquad (6\text{-}1)$$

$$\text{OUT}_j = 1 \text{ if } \text{NET}_j > T_j$$

$$\text{OUT}_j = 0 \text{ if } \text{NET}_j < T_j$$

$$\text{OUT}_j \text{ unchanged if } \text{NET}_j = T_j$$

The *state of a network* is simply the set of the current values of the OUT signals from all neurons. In the original Hopfield network, the state of each neuron changed at discrete random times; in later work, the neuron states could change simultaneously. Because the output of a binary neuron can be only one or zero (intermediate levels are not allowed), the current state of the network forms a binary number, each bit of which represents the OUT signal from a neuron.

The network's operation is easily visualized geometrically. Figure 6-2a shows the case for two neurons in the output layer in which each of the four system states (00, 01, 10, 11) labels a vertex

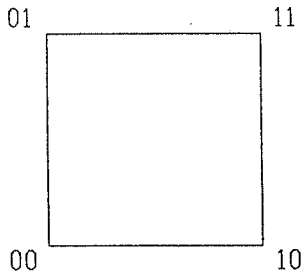

Figure 6-2a. Two Neurons Produce Four System States

of a square. Figure 6-2b shows a three-neuron system represented by a cube (in three-dimensional space) having eight vertexes, each labeled with a three-bit binary number. In general, a system with n neurons has 2^n distinct states and is associated with an n-dimensional hypercube.

When a new input vector is applied, the network moves from vertex to vertex until it stabilizes. The stable vertex is determined by the network weights, the current inputs, and the threshold value. If the input vector is partially incorrect or incomplete, the network stabilizes to the vertex closest to the one desired.

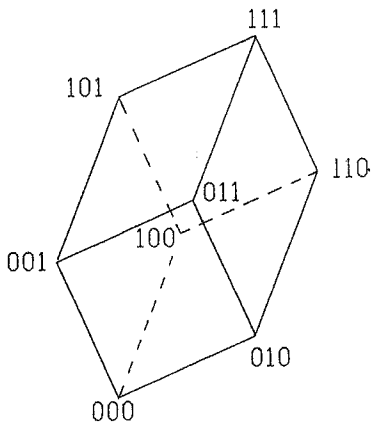

Figure 6-2b. Three Neurons Produce Eight System States

Stability

As with other networks, the weights between layers in this network may be considered to form a matrix $\mathbf{W}$. Cohen and Grossberg (1983) have shown that recurrent networks are stable if the matrix is symmetrical with zeros on its main diagonal; that is, if $w_{ij} = w_{ji}$ for i not equal to j, and $w_{ii} = 0$ for all i.

The stability of such a network may be proven through an elegant mathematical technique. Suppose a function can be found that always decreases each time the network changes state. Eventually this function must reach a minimum and stop, thereby ensuring that the network is stable. The function that follows is called a Liapunov function and works in just such a manner on the recurrent networks presented above:

$$E = (-1/2)\sum_i \sum_j w_{ij}\,\mathrm{OUT}_i\,\mathrm{OUT}_j - \sum_j I_j\,\mathrm{OUT}_j + \sum_j T_j\,\mathrm{OUT}_j \quad (6\text{-}2)$$

where

E = an artificial network energy
w_{ij} = weight from the output of neuron i to the input of neuron j
OUT_j = output of neuron j
I_j = external input to neuron j
T_j = threshold of neuron j

The change in energy E, due to a change in the state of neuron j, is

$$\delta E = -\left[\sum_{i \neq j}(w_{ij}\,\mathrm{OUT}_i) + I_j - T_j\right]\delta\,\mathrm{OUT}_j$$

$$= -[\mathrm{NET}_j - T_j]\,\delta\,\mathrm{OUT}_j \quad (6\text{-}3)$$

where $\delta\,\mathrm{OUT}_j$ is the change in the output of neuron j

Suppose that the NET value of neuron j is greater than the threshold. This will cause the term in brackets to be positive and, from Equation 6-1, the output of neuron j must change in the positive direction (or remain constant). This means that $\delta\,\mathrm{OUT}_j$ can be only positive or zero, and δE must be negative; hence, the network energy must either decrease or stay constant.

Next, assume that NET is less than the threshold. Then $\delta\,\mathrm{OUT}_j$

can be only negative or zero; hence, again the energy must decrease or stay constant.

Finally, if NET equals the threshold, δ_j is zero and the energy remains unchanged.

This shows that any change in the state of a neuron will either reduce the energy or maintain its current value. Because the energy shows this continuous downward trend, eventually it must find a minimum and stop. By definition, such a network is stable.

The network symmetry criterion is sufficient, but not necessary, to define a stable system. There are many stable systems (e.g., all feedforward networks!) that do not satisfy it. Also, examples can be shown in which minute deviations from symmetry can produce continuous oscillations; however, approximate symmetry is usually adequate to produce stable systems.

Associative Memory

Human memory operates in an associative manner; that is, a portion of a recollection can produce a larger related memory. For example, hearing only a few bars of music may recall a complete sensory experience, including scenes, sounds, and odors. By contrast, ordinary computer memory is location addressable; an address is applied and the data occupying that address is returned.

A recurrent network forms an associative memory. Like human memory, a portion of the desired data is supplied and the full data "memory" is returned. To make an associative memory using a recurrent network, the weights must be selected to produce energy minima at the desired vertexes of the unit hypercube.

Hopfield (1984) has developed an associative memory in which the outputs are continuous, ranging from +1 to −1, corresponding to the binary values 0 and 1, respectively. The memories are encoded as binary vectors and stored in the weights according to the formula that follows:

$$w_{ij} = \sum_{d=1 \text{ to } m} (\text{OUT}_{i,d}\,\text{OUT}_{j,d}) \tag{6-4}$$

where

m = the number of desired memories (output vectors)

d = the number of a desired memory (output vector)

$\text{OUT}_{i,d}$ = the ith component of the desired output vector

This expression may be clarified by noting that the weight array $\mathbf{W}$ can be found by calculating the outer product of each desired vector with itself (if the desired vector has n components, this operation forms an n-by-n matrix) and summing all of the matrixes thus formed. This may be expressed symbolically as follows:

$$\mathbf{W} = \sum_i \mathbf{D}_i' \mathbf{D}_i \tag{6-5}$$

where $\mathbf{D}_i$ is the ith desired row vector.

Once the weights are determined, the network may be used to produce the desired output vector, even given an input vector that may be partially incorrect or incomplete. To do so, the outputs of the network are first forced to the values of this input vector. Next, the input vector is removed and the network is allowed to "relax" toward the closest deep minimum. Note that the network follows the local slope of the energy function, and it may become trapped in a local minimum and not find the best solution in a global sense.

Continuous Systems

Hopfield (1984) shows other cases in which the activation function F is continuous, thereby more accurately simulating the biological neuron. A common choice is the S-shaped sigmoid or logistic function

$$F(x) = 1/(1 + e^{-\lambda \text{NET}}) \tag{6-6}$$

where λ is a coefficient that determines the steepness of the sigmoidal function. If λ is large, F approaches the threshold function previously described; smaller values for λ produce a more gentle slope.

Like the binary system, stability is ensured if the weights are symmetrical; that is, $w_{ij} = w_{ji}$ and $w_{ii} = 0$ for all i. An energy function that proves such networks stable has been devised, but it is not treated here due to its conceptual similarity to the discrete case.

Interested readers should consult Cohen and Grossberg (1983) for a more complete treatment of this important topic.

If the value of λ is large, continuous systems perform much like discrete binary systems, ultimately stabilizing with all outputs near zero or one, a vertex of the unit hypercube. As λ is reduced, stable points move away from the vertexes, disappearing one by one as λ approaches zero. Figure 6-3 shows an energy contour map for a continuous system consisting of two neurons.

Hopfield Nets and the Boltzmann Machine

Hopfield nets suffer from a tendency to stabilize to a local rather than a global minimum of the energy function. This problem is largely solved by a class of networks known as Boltzmann machines, in which the neurons change state in a statistical rather than a deterministic fashion. There is a close analogy between these methods and the way in which a metal is annealed; hence, the methods are often called *simulated annealing*.

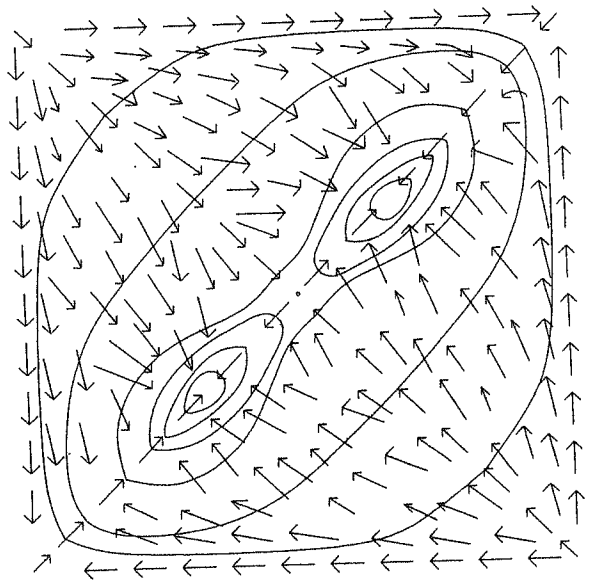

Figure 6-3. Energy Contour Map

Thermodynamic Systems (Simulated Annealing)

A metal is annealed by heating it to a temperature above its melting point, and then letting it cool gradually. At high temperatures, the atoms possess high energies and move about freely, randomly assuming every possible configuration. As temperature is gradually reduced, the atomic energies decrease and the system as a whole tends to settle into a minimum-energy configuration. Finally, when the cooling is complete, a state is reached where the system energy is at a global minimum.

At a given temperature, the probability distribution of system energies is determined by the Boltzmann probability factor

$$= \exp(-E/kT)$$

where

E = system energy
k = Boltzmann's constant
T = temperature

From this it may be seen that there is a finite probability of the system's possessing high energy even at low temperatures. Likewise, there is a small but calculable probability that a kettle of water on a fire will freeze before it boils.

The statistical distribution of energies allows the system to escape a local energy minimum. At the same time, the probability of high system energy decreases rapidly as temperature drops; hence, there is a strong bias toward low energy states at low temperatures.

Statistical Hopfield Networks

If the state-change rules for the binary Hopfield net are determined statistically rather than deterministically as in Equation 6-1, a simulated-annealing system results. To accomplish this, the probability of a weight change is determined by the amount by which the NET output of a neuron exceeds its threshold. In symbols, let

$$E_k = \mathrm{NET}_k - \theta_k$$

where

NET_k = the NET output of neuron k

θ_k = the threshold of neuron k

and

$$p_k = 1/[1 + \exp(-\delta E_k/T)]$$

(note the Boltzmann probability function in the denominator)

where T is artificial temperature.

In operation, the artificial temperature T is set to a high value, neurons are clamped to an initial state determined by an input vector, and the network is allowed to seek an energy minimum according to the procedure that follows:

1. For each neuron, set the state to one, with a probability equal to p_k; otherwise, set its state to zero.
2. Gradually reduce the artificial temperature and repeat step 1 until equilibrium is reached.

Generalized Networks

The Boltzmann-machine technique can be extended to networks of virtually any configuration, although stability cannot be guaranteed. To do so, simply select one set of neurons to serve as inputs and another set to serve as outputs. Clamp the input set to the values of the input vector and allow the network to relax according to steps 1 and 2 above.

A training procedure for such a network has been described by Hinton and Sejnowski (1986), consisting of the steps that follow:

1. Calculate clamped probabilities.
   a. Clamp input and output neurons to the training vector values.
   b. Allow the network to find equilibrium.
   c. Record the output values for all units.
   d. Repeat steps a through c for all training vectors.
   e. Calculate P^+_{ij}, or the probability over all training vectors that unit i and unit j are both one.

2. Calculate unclamped probabilities.
   a. Starting from a random state, allow the network to "free run" with no inputs or outputs clamped.
   b. Repeat step 2a a large number of times, recording values of all neurons.
   c. Calculate P^-_{ij}, or the probability that units i and j are both one.
3. Adjust network weights as follows:

$$\delta w_{ij} = \eta [P^+_{ij} - P^-_{ij}]$$

where

δw_{ij} = the change in weight w_{ij}

η = the learning rate coefficient

APPLICATIONS

Analog-to-Digital Converter

In recent works (Hopfield and Tank 1985; Tank and Hopfield 1986), an electrical circuit has been presented that uses a recurrent network to produce a four-bit analog-to-digital converter. Figure 6-4 shows a block diagram of the circuit, with amplifiers serving as artificial neurons. Resistors, representing weights, connect each neuron's output to the inputs of all others. To satisfy the stability constraint, no resistor connects a neuron's output to its own input and the weights are symmetrical; that is, a resistor from the output of neuron i to the input of neuron j has the same value as the resistor from the output of neuron j to the input of neuron i.

Note that the amplifiers have both normal and inverting outputs. This takes into account the case in which a weight must be negative, while permitting the use of ordinary positive-valued resistors for all weights. All possible resistors are shown in Figure 6-4; however, in no case is it necessary to connect both the normal and inverted outputs of a neuron to another neuron's input.

In a realistic circuit, each amplifier will have a finite input resistance and input capacitance that must be included to characterize the dynamic response. Network stability does not require that these elements be the same for all amplifiers, nor need they be

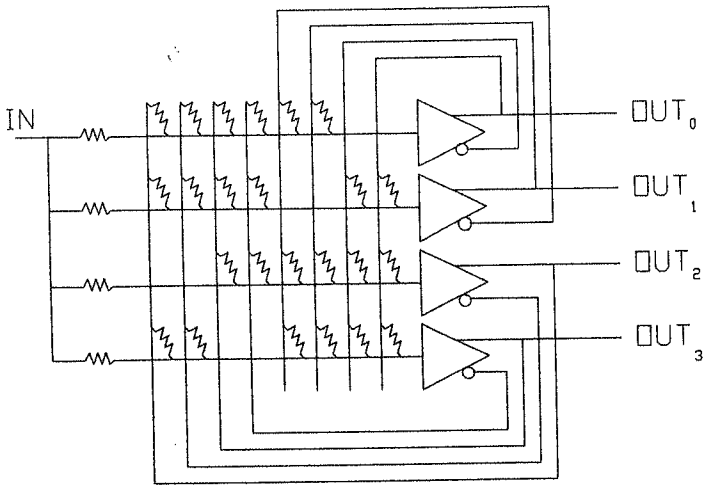

Figure 6-4. Four-Bit Analog-to-Digital Converter Using Hopfield Net

symmetrical. Because these elements affect only the time required to reach a solution and not the solution itself, they have been omitted to simplify the analysis.

The application assumes that a threshold function is used (the limit of the sigmoid function as λ approaches ∞). Furthermore, all of the outputs are changed at the beginning of discrete time intervals called epochs. At the start of each epoch, the summation of the inputs to each neuron is examined. If it is greater than the threshold, the output becomes one; if it is less than the threshold, it becomes zero. Neuron outputs remain unchanged during an epoch.

The object is to select the resistors (weights) so that a continuously increasing voltage X applied to the single-input terminal produces a set of four outputs representing a binary number, the value of which is an approximation to the input voltage (see Figure 6-5). First, the energy function is defined as follows:

$$E = -1/2[X - \sum_j 2^j \text{OUT}_j]^2 + \sum_j (2^{2j-1})[\text{OUT}_j(1 - \text{OUT}_j)] \qquad (6\text{-}7)$$

where X is the input voltage.

When E is minimized, the desired outputs have been reached. The first expression in brackets is minimized when the binary number formed by the outputs is as close as possible (in the least-squares sense) to the analog value of the input X. The second bracketed expression goes to 0 when all of the outputs are either 1 or 0, thereby imposing the constraint that the outputs have only binary values.

If Equation 6-7 is rearranged and compared with Equation 6-2, the resulting expression for the weights is

$$w_{ij} = -2^{(i+j)}$$
$$y_i = 2^i \qquad (6\text{-}8)$$

where

w_{ij} = conductance (the reciprocal of resistance) from the output

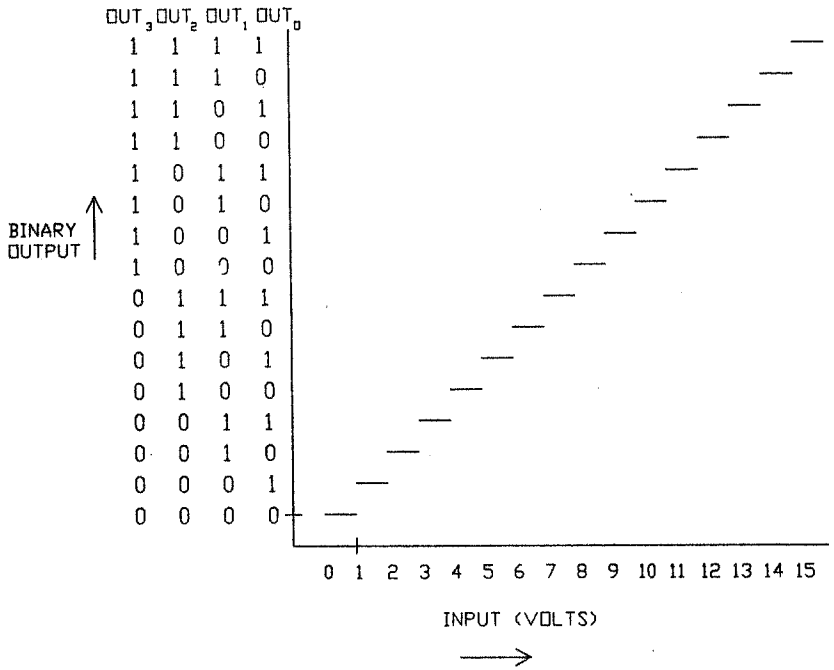

Figure 6-5. Four-Bit Analog-to-Digital Converter Ideal Input–Output Relationship

of neuron i to the input of neuron j (this must also equal the conductance from the output of neuron j to the input of neuron i)

y_i = conductance from the input X to the input of neuron i.

To produce a circuit with practical values of resistance and power dissipation, all weights must be scaled by a multiplicative constant.

The idealized input–output relationship of Figure 6-5 will be realized only if the inputs are set to zero prior to performing a conversion. If this is not done, the network may become trapped in a local minimum of the energy function and produce incorrect outputs.

The Traveling Salesman Problem

The "traveling salesman problem," or TSP, is an optimization task that arises in many practical situations. It may be stated as follows: given a group of cities to be visited and the distance between each city, find the shortest tour that visits each city only once and returns to the starting point. The problem has been proven to be one of a large set of problems termed "NP complete" (nondeterministic polynomial) (Garey and Johnson 1979). NP complete problems have no known method of solution better than trying all the possibilities, nor, according to most mathematicians, is any superior method likely to be found. Because such an exhaustive search is generally impractical for more than a few cities, heuristic methods have been applied to find solutions that are acceptable, if not optimal.

The solution using recurrent networks described by Hopfield and Tank (1985) is typical in that regard; the results are not guaranteed to be optimal. Still, an answer is reached so rapidly that the technique may prove useful in certain cases.

Suppose that the cities to be visited are lettered A, B, C, and D, and that the distance between the pairs is d_{ab}, d_{bc}, and so on.

The solution is an ordered set of n cities. The problem is then to map this onto the computational network, using neurons in the high-gain mode (λ approaching ∞). Each city is represented as a row of n neurons. One and only one such neuron in a row may be

set to one (all others must be set to zero). This neuron set to one indicates the order in which a specific city is visited during the tour. Figure 6-6 shows such a result, where city C is visited first, city A is visited second, city D is visited third, and city B is visited fourth. This requires n^2 neurons, a number that grows rapidly with the number of cities. The length of such a tour would be $d_{ca} + d_{ad} + d_{db} + d_{bc}$. Because each city is visited only once, and only one city is visited at a time, there is only a single 1 in each row and column. For an n-city problem there are $n!/(2n)$ distinct tours. If $n = 60$, there are 69.34155×10^{78} possible tours. Considering that there are only 10^{11} stars in the Milky Way galaxy, it becomes clear why calculating all possibilities for a 1,000-city tour would take geological time on the world's fastest computer.

Let us demonstrate how to set up a network to solve such an NP-complete problem. Each neuron is identified by double subscripts, indicating the city and the order in which it is visited. For example, OUT_{xj} indicates that city x was the jth city in the tour.

The energy function must satisfy two requirements: first, it must be low for only those solutions that produce a single 1 in each column and row. Second, it must favor solutions having short paths.

The first requirement is satisfied by the three-summation energy function that follows:

$$E = \mathrm{A}/2 \sum_X \sum_i \sum_{j \neq i} \mathrm{OUT}_{Xi}\,\mathrm{OUT}_{Xj} + \mathrm{B}/2 \sum_i \sum_X \sum_{Y \neq X} \mathrm{OUT}_{Xi}\,\mathrm{OUT}_{Yi} + \mathrm{C}/2 \left[\left(\sum_X \sum_i \mathrm{OUT}_{Xi} \right) - n \right]^2 \qquad (6\text{-}9)$$

where A, B, and C are constants. The resulting set of rules is as follows:

1. The first triple summation is zero if, and only if, each row (city) contains no more than a single 1.
2. The second triple summation is zero if, and only if, each column (tour position) contains no more than a single 1.

CITY	ORDER VISITED 1	2	3	4
A	0	1	0	0
B	0	0	0	1
C	1	0	0	0
D	0	0	1	0

Figure 6-6. Traveling Salesman Tour

3. The third summation is zero if, and only if, there are exactly n 1s in the matrix.

The second requirement—favoring short tours—is satisfied by adding a term to the energy function as follows:

$$E = 1/2D\sum_X \sum_{Y \neq X} \sum_i d_{XY}\, OUT_{Xi}(OUT_{Y,i+1} + OUT_{Y,i-1}) \quad (6\text{-}10)$$

Note that this term represents the length of any valid tour. The subscripts are defined modulo n for convenience; that is, $OUT_{n+j} = OUT_j$, where D is a constant.

With sufficiently large values for A, B, and C, the low-energy states will represent valid tours, while a large value for D ensures that a short tour will be found.

Next, the weights must be found. This involves relating the terms in the energy function to those of the general form (see Equation 6-2). The result is as follows:

$w_{xi,yj} = -A\,\delta_{xy}(1-\delta_{ij})$ (prevents more than a single 1 within a row)

$-B\,\delta_{ij}(1-\delta_{xy})$ (prevents more than a single 1 in a column)

$-C$ (global inhibition)

$-D\,d_{xy}(\delta_{j,i+1} + \delta_{j,i-1})$ (distance term)

where $\delta_{ij} = 1$ if $i = j$ and otherwise is 0.

In addition, each neuron has a bias weight x_i connected to +1 with a value of Cn.

Hopfield and Tank (1985) report an experiment in which the TSP was solved for 10 cities. In this case, they chose the excitation function

$$OUT = 1/2[1 + \tanh(NET/u_0)]$$

As a result, 16 out of 20 trials converged to valid tours, and about 50% of the solutions were one of the shortest tours as found by exhaustive search. This result is more impressive if one realizes that there are 181,440 possible valid tours.

It has been reported that the convergence of Hopfield's solution to the traveling salesman problem is highly dependent upon the coefficients, and that there is no systematic way to determine their values (Van den Bout and Miller 1988). These authors propose another energy function, with only one coefficient the value of which is easily found. In addition, they present a new convergence algorithm. It may be expected that new and better methods will continue to be developed, as a fully satisfactory solution would have many important applications.

DISCUSSION

Local Minima

Starting from suitable initial conditions, the analog-to-digital converter network finds a single optimal solution. This is due to the simple nature of the energy surface for this problem. In the TSP, the energy surface is highly convoluted—full of dips, valleys, and local minima—and there is no guarantee that a global optimal solution will be found, or even that the solution will be valid. This raises serious questions about the reliability of the network and the credibility of its solutions. The limitations of the network are mitigated by the fact that finding global minima for NP-complete problems is an intractable problem that has not been solved in a reasonable amount of time in any other way; methods that are much slower and inherently serial produce results that are no better.

Speed

The rapid computational capability of the network is a major advantage. This arises from the highly parallel nature of the convergence process. If implemented in analog electronics form, solutions seldom take more than a few network time constants. Furthermore, the convergence time changes little with the size of the problem. Contrast this with the more than exponential increase in processing time with conventional approaches. Single-processor simulations cannot take advantage of this inherently parallel architecture, but modern multiprocessor systems, such as the Connection Machine (with 65,536 processors!), hold great promise in solving previously intractable problems.

Energy Function

It is not a trivial matter to find the function that maps a problem onto the general network-energy function. Existing solutions have been achieved through ingenuity and mathematical expertise, talents that are always in short supply. For certain problems, methods exist to determine the network weights in a systematic fashion; these techniques are studied in Chapter 7.

Network Capacity

The maximum number of memories that may be stored in a Hopfield network is a current research topic. Because a network of N binary neurons can have as many as $2N$ states, researchers were surprised to find that the maximum memory storage capacity was much less than this.

If too many memories are stored, the network will not stabilize on some of them. Furthermore, it can remember things it has not been taught; that is, it can stabilize to a solution that is not among the desired vectors. These characteristics perplexed early researchers, who had no mathematical way to determine how many memories could be stored without encountering the problems.

Recent research has cast much light on this matter. For example, it had been conjectured that the maximum number of memories K that can be stored in a network of N neurons and recalled without error is less than cN^2, where c is a positive constant greater than one. While this limit is approached in some cases, in general it proved to be excessively optimistic; Hopfield (1982) showed experimentally that the general capacity limit was actually more like $0.15N$. Abu-Mostafa and St. Jacques (1985) have shown that the number of such states cannot exceed N, a result that is compatible with observations of actual systems and is as good an estimate as is available today.

CONCLUSION

Recurrent networks are fertile subjects for continued research. Their dynamic behavior creates new and interesting possibilities and certain unique problems. As we point out in Chapter 9, the power and problems translate into the optical domain, where they create fascinating image-recognition capability in addition to perplexing limitations.

References

Abu-Mostafa, Y. S., and St. Jacques, J. 1985. Information capacity of the Hopfield model. *IEEE Transactions on Information Theory* 31(4): 461–64.

Cohen, M. A., and Grossberg, S. G. 1983. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man and Cybernetics* 13:815–26.

Garey, M. R., and Johnson, D. S. 1979. *Computers and intractability.* New York: W. H. Freeman.

Grossberg, S. 1987. *The adaptive brain*, vols. 1 and 2. Amsterdam: North-Holland.

Hinton, G. E., and Sejnowski, T. J. 1986. Learning and relearning in Boltzmann machines. In *Parallel distributed processing*, vol. 1, pp. 282–317. Cambridge, MA: MIT Press.

Hopfield, J. J. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Science* 79:2554–58.