COGNITIVE SCIENCE 9, 147–169 (1985)

A Learning Algorithm for Boltzmann Machines*

DAVID H. ACKLEY
GEOFFREY E. HINTON
Computer Science Department
Carnegie-Mellon University
TERRENCE J. SEJNOWSKI
Biophysics Department
The Johns Hopkins University

The computational power of massively parallel networks of simple processing elements resides in the communication bandwidth provided by the hardware connections between elements. These connections can allow a significant fraction of the knowledge of the system to be applied to an instance of a problem in a very short time. One kind of computation for which massively parallel networks appear to be well suited is large constraint satisfaction searches, but to use the connections efficiently two conditions must be met: First, a search technique that is suitable for parallel networks must be found. Second, there must be some way of choosing internal representations which allow the preexisting hardware connections to be used efficiently for encoding the constraints in the domain being searched. We describe a general parallel search method, based on statistical mechanics, and we show how it leads to a general learning rule for modifying the connection strengths so as to incorporate knowledge about a task domain in an efficient way. We describe some simple examples in which the learning algorithm creates internal representations that are demonstrably the most efficient way of using the preexisting connectivity structure.

1. INTRODUCTION

Evidence about the architecture of the brain and the potential of the new VLSI technology have led to a resurgence of interest in "connectionist" sys-

* The research reported here was supported by grants from the System Development Foundation. We thank Peter Brown, Francis Crick, Mark Derthick, Scott Fahlman, Jerry Feldman, Stuart Geman, Gail Gong, John Hopfield, Jay McClelland, Barak Pearlmutter, Harry Printz, Dave Rumelhart, Tim Shallice, Paul Smolensky, Rick Szeliski, and Venkataraman Venkatasubramanian for helpful discussions.

Reprint requests should be addressed to David Ackley, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA 15213.

tems (Feldman & Ballard, 1982; Hinton & Anderson, 1981) that store their long-term knowledge as the strengths of the connections between simple neuron-like processing elements. These networks are clearly suited to tasks like vision that can be performed efficiently in parallel networks which have physical connections in just the places where processes need to communicate. For problems like surface interpolation from sparse depth data (Grimson, 1981; Terzopoulos, 1984) where the necessary decision units and communication paths can be determined in advance, it is relatively easy to see how to make good use of massive parallelism. The more difficult problem is to discover parallel organizations that do not require so much problem-dependent information to be built into the architecture of the network. Ideally, such a system would adapt a given structure of processors and communication paths to whatever problem it was faced with.

This paper presents a type of parallel constraint satisfaction network which we call a "Boltzmann Machine" that is capable of learning the underlying constraints that characterize a domain simply by being shown examples from the domain. The network modifies the strengths of its connections so as to construct an internal *generative* model that produces examples with the same probability distribution as the examples it is shown. Then, when shown any particular example, the network can "interpret" it by finding values of the variables in the internal model that would generate the example. When shown a partial example, the network can complete it by finding internal variable values that generate the partial example and using them to generate the remainder. At present, we have an interesting mathematical result that guarantees that a certain learning procedure will build internal representations which allow the connection strengths to capture the underlying constraints that are implicit in a large ensemble of examples taken from a domain. We also have simulations which show that the theory works for some simple cases, but the current version of the learning algorithm is very slow.

The search for general principles that allow parallel networks to learn the structure of their environment has often begun with the assumption that networks are randomly wired. This seems to us to be just as wrong as the view that *all* knowledge is innate. If there are connectivity structures that are good for particular tasks that the network will have to perform, it is much more efficient to build these in at the start. However, not all tasks can be foreseen, and even for ones that can, fine-tuning may still be helpful.

Another common belief is that a general connectionist learning rule would make sequential "rule-based" models unnecessary. We believe that this view stems from a misunderstanding of the need for multiple levels of description of large systems, which can be usefully viewed as either parallel or serial depending on the grain of the analysis. Most of the key issues and questions that have been studied in the context of sequential models do not magically disappear in connectionist models. It is still necessary to perform

searches for good solutions to problems or good interpretations of perceptual input, and to create complex internal representations. Ultimately it will be necessary to bridge the gap between hardware-oriented connectionist descriptions and the more abstract symbol manipulation models that have proved to be an extremely powerful and pervasive way of describing human information processing (Newell & Simon, 1972).

2. THE BOLTZMANN MACHINE

The Boltzmann Machine is a parallel computational organization that is well suited to constraint satisfaction tasks involving large numbers of "weak" constraints. Constraint-satisfaction searches (e.g., Waltz, 1975; Winston, 1984) normally use "strong" constraints that *must* be satisfied by any solution. In problem domains such as games and puzzles, for example, the goal criteria often have this character, so strong constraints are the rule.[1] In some problem domains, such as finding the most plausible interpretation of an image, many of the criteria are not all-or-none, and frequently even the best possible solution violates some constraints (Hinton, 1977). A variation that is more appropriate for such domains uses weak constraints that incur a cost when violated. The quality of a solution is then determined by the total cost of all the constraints that it violates. In a perceptual interpretation task, for example, this total cost should reflect the implausibility of the interpretation.

The machine is composed of primitive computing elements called *units* that are connected to each other by bidirectional *links.* A unit is always in one of two states, *on* or *off,* and it adopts these states as a probabilistic function of the states of its neighboring units and the *weights* on its links to them. The weights can take on real values of either sign. A unit being on or off is taken to mean that the system currently accepts or rejects some elemental hypothesis about the domain. The weight on a link represents a weak pairwise constraint between two hypotheses. A positive weight indicates that the two hypotheses tend to support one another; if one is currently accepted, accepting the other should be more likely. Conversely, a negative weight suggests, other things being equal, that the two hypotheses should not both be accepted. Link weights are *symmetric,* having the same strength in both directions (Hinton & Sejnowski, 1983).[2]

[1] But, see (Berliner & Ackley, 1982) for argument that, even in such domains, strong constraints must be used only where absolutely necessary for legal play, and in particular must not propagate into the determination of *good* play.

[2] Requiring the weights to be symmetric may seem to restrict the constraints that can be represented. Although a constraint on boolean variables A and B such as "$A \equiv B$ with a penalty of 2 points for violation" is obviously symmetric in A and B, "$A \Rightarrow B$ with a penalty of 2 points for violation" appears to be fundamentally asymmetric. Nevertheless, this constraint can be represented by the combination of a constraint on A alone and a symmetric pairwise constraint as follows: "Lose 2 points if A is true" and "Win 2 points if both A and B are true."

The resulting structure is related to a system described by Hopfield (1982), and as in his system, each global state of the network can be assigned a single number called the "energy" of that state. With the right assumptions, the individual units can be made to act so as to *minimize the global energy*. If *some* of the units are externally forced or "clamped" into particular states to represent a particular input, the system will then find the minimum energy configuration that is compatible with that input. The energy of a configuration can be interpreted as the extent to which that combination of hypotheses violates the constraints implicit in the problem domain, so in minimizing energy the system evolves towards "interpretations" of that input that increasingly satisfy the constraints of the problem domain.

The energy of a global configuration is defined as

$$E = -\sum_{i<j} w_{ij} s_i s_j + \sum_i \theta_i s_i \tag{1}$$

where w_{ij} is the strength of connection between units i and j, s_i is 1 if unit i is on and 0 otherwise, and θ_i is a threshold.

2.1 Minimizing Energy

A simple algorithm for finding a combination of truth values that is a *local* minimum is to switch each hypothesis into whichever of its two states yields the lower total energy given the current states of the other hypotheses. If hardware units make their decisions asynchronously, and if transmission times are negligible, then the system always settles into a local energy minimum (Hopfield, 1982). Because the connections are symmetric, the difference between the energy of the whole system with the k^{th} hypothesis rejected and its energy with the k^{th} hypothesis accepted can be determined locally by the k^{th} unit, and this "energy gap" is just

$$\Delta E_k = \sum_i w_{ki} s_i - \theta_k \tag{2}$$

Therefore, the rule for minimizing the energy contributed by a unit is to adopt the *on* state if its total input from the other units and from outside the system exceeds its threshold. This is the familiar rule for binary threshold units.

The threshold terms can be eliminated from Eqs. (1) and (2) by making the following observation: the effect of θ_i on the global energy or on the energy gap of an individual unit is identical to the effect of a link with strength $-\theta_i$ between unit i and a special unit that is by definition always held in the *on* state. This "true unit" need have no physical reality, but it simplifies the computations by allowing the threshold of a unit to be treated in the same manner as the links. The value $-\theta_i$ is called the *bias* of unit i. If a perma-

nently active "true unit" is assumed to be part of every network, then Eqs. (1) and (2) can be written as:

$$E = -\sum_{i<j} w_{ij} s_i s_j \tag{3}$$

$$\Delta E_k = \sum_i w_{ki} s_i \tag{4}$$

2.2 Using Noise to Escape from Local Minima

The simple, deterministic algorithm suffers from the standard weakness of gradient descent methods: It gets stuck in *local* minima that are not globally optimal. This is not a problem in Hopfield's system because the local energy minima of his network are used to store "items": If the system is started near some local minimum, the desired behavior is to fall into that minimum, not to find the global minimum. For constraint satisfaction tasks, however, the system must try to escape from local minima in order to find the configuration that is the global minimum given the current input.

A simple way to get out of local minima is to occasionally allow jumps to configurations of higher energy. An algorithm with this property was introduced by Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller (1953) to study average properties of thermodynamic systems (Binder, 1978) and has recently been applied to problems of constraint satisfaction (Kirkpatrick, Gelatt, & Vecchi, 1983). We adopt a form of the Metroplis algorithm that is suitable for parallel computation: If the energy gap between the *on* and *off* states of the k^{th} unit is ΔE_k then regardless of the previous state set $s_k = 1$ with probability

$$p_k = \frac{1}{(1 + e^{-\Delta E_k / T})} \tag{5}$$

where T is a parameter that acts like temperature (see Figure 1).

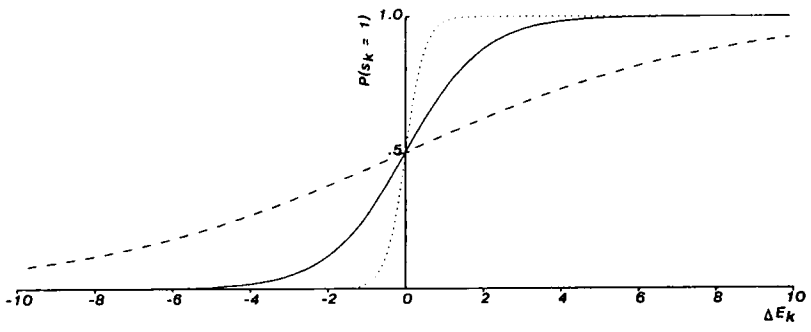

Figure 1. Eq. (5) at $T=1.0$ (*solid*), $T=4.0$ (*dashed*), and $T=0.25$ (*dotted*).

The decision rule in Eq. (5) is the same as that for a particle which has two energy states. A system of such particles in contact with a heat bath at a given temperature will eventually reach thermal equilibrium and the probability of finding the system in any global state will then obey a Boltzmann distribution. Similarly, a network of units obeying this decision rule will eventually reach "thermal equilibrium" and the relative probability of two global states will follow the Boltzman distribution:

$$\frac{P_\alpha}{P_\beta} = e^{-(E_\alpha - E_\beta)/T} \tag{6}$$

where P_α is the probability of being in the α^{th} global state, and E_α is the energy of that state.

The Boltzmann distribution has some beautiful mathematical properties and it is intimately related to information theory. In particular, the difference in the log probabilities of two global states is just their energy difference (at a temperature of 1). The simplicity of this relationship and the fact that the equilibrium distribution is independent of the path followed in reaching equilibrium are what make Boltzmann machines interesting.

At low temperatures there is a strong bias in favor of states with low energy, but the time required to reach equilibrium may be long. At higher temperatures the bias is not so favorable but equilibrium is reached faster. A good way to beat this trade-off is to start at a high temperature and gradually reduce it. This corresponds to annealing a physical system (Kirkpatrick, Gelatt, & Vecchi, 1983). At high temperatures, the network will ignore small energy differences and will rapidly approach equilibrium. In doing so, it will perform a search of the coarse overall structure of the space of global states, and will find a good minimum at that coarse level. As the temperature is lowered, it will begin to respond to smaller energy differences and will find one of the better minima within the coarse-scale minimum it discovered at high temperature. Kirkpatrick et al. have shown that this way of searching the coarse structure before the fine is very effective for combinatorial problems like graph partitioning, and we believe it will also prove useful when trying to satisfy multiple weak constraints, even though it will clearly fail in cases where the best solution corresponds to a minimum that is deep, narrow, and isolated.

3. A LEARNING ALGORITHM

Perhaps the most interesting aspect of the Boltzmann Machine formulation is that it leads to a domain-independent learning algorithm that modifies the

connection strengths between units in such a way that the whole network develops an internal model which captures the underlying structure of its environment. There has been a long history of failure in the search for such algorithms (Newell, 1982), and many people (particularly in Artificial Intelligence) now believe that no such algorithms exist. The major technical stumbling block which prevented the generalization of simple learning algorithms to more complex networks was this: To be capable of interesting computations, a network must contain nonlinear elements that are not directly constrained by the input, and when such a network does the wrong thing it appears to be impossible to decide which of the many connection strengths is at fault. This "credit-assignment" problem was what led to the demise of perceptrons (Minsky & Papert, 1968; Rosenblatt, 1961). The perceptron convergence theorem guarantees that the weights of a single layer of decision units can be trained, but it could not be generalized to networks of such units when the task did not directly specify how to use all the units in the network.

This version of the credit-assignment problem can be solved within the Boltzmann Machine formulation. By using the right stochastic decision rule, and by running the network until it reaches "thermal equilibrium" at some finite temperature, we achieve a mathematically simple relationship between the probability of a global state and its energy. For a network that is running freely without any input from the environment, this relationship is given by Eq. (6). Because the energy is a *linear* function of the weights (Eq. 1) this leads to a remarkably simple relationship between the log probabilities of global states and the individual connection strengths:

$$\frac{\partial \ln P_\alpha}{\partial w_{ij}} = \frac{1}{T}[s_i^\alpha s_j^\alpha - p_{ij}'] \tag{7}$$

where s_i^α is the state of the i^{th} unit in the α^{th} global state (so $s_i^\alpha s_j^\alpha$ is 1 only if units i and j are both on in state α), and p_{ij}' is just the probability of finding the two units i and j on at the same time when the system is at equilibrium.

Given Eq. (7), it is possible to manipulate the log probabilities of global states. If the environment directly specifies the required probabilities P_α for each global state α, there is a straightforward way of converging on a set of weights that achieve those probabilities, provided any such set exists (for details, see Hinton & Sejnowski, 1983a). However, this is not a particularly interesting kind of learning because the system has to be given the required probabilities of *complete* global states. This means that the central question of what internal representation should be used has already been decided by the environment. The interesting problem arises when the environment implicitly contains high-order constraints and the network must choose internal representations that allow these constraints to be expressed efficiently.

3.1 Modeling the Underlying Structure of an Environment

The units of a Boltzmann Machine partition into two functional groups, a nonempty set of *visible* units and a possibly empty set of *hidden* units. The visible units are the interface between the network and the environment; during training all the visible units are clamped into specific states by the environment; when testing for completion ability, any subset of the visible units may be clamped. The hidden units, if any, are never clamped by the environment and can be used to "explain" underlying constraints in the ensemble of input vectors that cannot be represented by pairwise constraints among the visible units. A hidden unit would be needed, for example, if the environment demanded that the states of three visible units should have even parity—a regularity that cannot be enforced by pairwise interactions alone. Using hidden units to represent more complex hypotheses about the states of the visible units, such higher-order constraints among the visible units can be reduced to first and second-order constraints among the whole set of units.

We assume that each of the environmental input vectors persists for long enough to allow the network to approach thermal equilibrium, and we ignore any structure that may exist in the *sequence* of environmental vectors. The structure of an environment can then be specified by giving the probability distribution over all 2^v states of the v visible units. The network will be said to have a perfect model of the environment if it achieves exactly the same probability distribution over these 2^v states when it is running freely at thermal equilibrium with all units unclamped so there is no environmental input.

Unless the number of hidden units is exponentially large compared to the number of visible units, it will be impossible to achieve a *perfect* model because even if the network is totally connected the $(v+h-1)(v+h)/2$ weights and $(v+h)$ biases among the v visible and h hidden units will be insufficient to model the 2^v probabilities of the states of the visible units specified by the environment. However, if there are regularities in the environment, and if the network uses its hidden units to capture these regularities, it may achieve a good match to the environmental probabilities.

An information-theoretic measure of the discrepancy between the network's internal model and the environment is

$$G = \sum_{\alpha} P(V_\alpha) \ln \frac{P(V_\alpha)}{P'(V_\alpha)} \tag{8}$$

where $P(V_\alpha)$ is the probability of the α^{th} state of the visible units when their states are determined by the environment, and $P'(V_\alpha)$ is the corresponding probability when the network is running freely with no environmental input. The G metric, sometimes called the asymmetric divergence or informa-

tion gain (Kullback, 1959; Renyi, 1962), is a measure of the distance from the distribution given by the $P'(V_\alpha)$ to the distribution given by the $P(V_\alpha)$. G is zero if and only if the distributions are identical; otherwise it is positive.

The term $P'(V_\alpha)$ depends on the weights, and so G can be altered by changing them. To perform gradient descent in G, it is necessary to know the partial derivative of G with respect to each individual weight. In most cross-coupled nonlinear networks it is very hard to derive this quantity, but because of the simple relationships that hold at thermal equilibrium, the partial derivative of G is straightforward to derive for our networks. The probabilities of global states are determined by their energies (Eq. 6) and the energies are determined by the weights (Eq. 1). Using these equations the partial derivative of G (see the appendix) is:

$$\frac{\partial G}{\partial w_{ij}} = -\frac{1}{T}(p_{ij} - p'_{ij}) \tag{9}$$

where p_{ij} is the average probability of two units both being in the *on* state when the environment is clamping the states of the visible units, and p'_{ij}, as in Eq. (7), is the corresponding probability when the environmental input is not present and the network is running freely. (Both these probabilities must be measured at equilibrium.) Note the similarity between this equation and Eq. (7), which shows how changing a weight affects the log probability of a single state.

To minimize G, it is therefore sufficient to observe p_{ij} and p'_{ij} when the network is at thermal equilibrium, and to change each weight by an amount proportional to the difference between these two probabilities:

$$\Delta w_{ij} = \epsilon(p_{ij} - p'_{ij}) \tag{10}$$

where ϵ scales the size of each weight change.

A surprising feature of this rule is that it uses only *locally available* information. The change in a weight depends only on the behavior of the two units it connects, even though the change optimizes a global measure, and the best value for each weight depends on the values of all the other weights. If there are no hidden units, it can be shown that G-space is concave (when viewed from above) so that simple gradient descent will not get trapped at poor local minima. With hidden units, however, there can be local minima that correspond to different ways of using the hidden units to represent the higher-order constraints that are implicit in the probability distribution of environmental vectors. Some techniques for handling these more complex G-spaces are discussed in the next section.

Once G has been minimized the network will have captured as well as possible the regularities in the environment, and these regularities will be enforced when performing completion. An alternative view is that the net-

work, in minimizing G, is finding the set of weights that is most likely to have generated the set of environmental vectors. It can be shown that maximizing this likelihood is mathematically equivalent to minimizing G (Peter Brown, personal communication, 1983).

3.2 Controlling the Learning

There are a number of free parameters and possible variations in the learning algorithm presented above. As well as the size of ϵ, which determines the size of each step taken for gradient descent, the lengths of time over which p_{ij} and p'_{ij} are estimated have a significant impact on the learning process. The values employed for the simulations presented here were selected primarily on the basis of empirical observations.

A practical system which estimates p_{ij} and p'_{ij} will necessarily have some noise in the estimates, leading to occasional "uphill steps" in the value of G. Since hidden units in a network can create local minima in G, this is not necessarily a liability. The effect of the noise in the estimates can be reduced, if desired, by using a small value for ϵ or by collecting statistics for a longer time, and so it is relatively easy to implement an annealing search for the minimum of G.

The objective function G is a metric that specifies how well two probability distributions match. Problems arise if an environment specifies that only a small subset of the possible patterns over the visible units ever occur. By default, the unmentioned patterns must occur with probability zero, and the only way a Boltzmann Machine running at a non-zero temperature can guarantee that certain configurations *never* occur is to give those configurations infinitely high energy, which requires infinitely large weights.

One way to avoid this implicit demand for infinite weights is to occasionally provide "noisy" input vectors. This can be done by filtering the "correct" input vectors through a process that has a small probability of reversing each of the bits. These noisy vectors are then clamped on the visible units. If the noise is small, the correct vectors will dominate the statistics, but every vector will have some chance of occurring and so infinite energies will not be needed. This "noisy clamping" technique was used for all the examples presented here. It works quite well, but we are not entirely satisfied with it and have been investigating other methods of preventing the weights from growing too large when only a few of the possible input vectors ever occur.

The simulations presented in the next section employed a modification of the obvious steepest descent method implied by Eq. (10). Instead of changing w_{ij} by an amount proportional to $p_{ij}-p'_{ij}$, it is simply incremented by a fixed "weight-step" if $p_{ij}>p'_{ij}$ and decremented by the same amount if $p_{ij}<p'_{ij}$. The advantage of this method over steepest descent is that it can cope

with wide variations in the first and second derivatives of G. It can make significant progress on dimensions where G changes gently without taking very large divergent steps on dimensions where G falls rapidly and then rises rapidly again. There is no suitable value for the ϵ in Eq. (10) in such cases. Any value large enough to allow progress along the gently sloping floor of a ravine will cause divergent oscillations up and down the steep sides of the ravine.[3]

4. THE ENCODER PROBLEM

The "encoder problem" (suggested to us by Sanjaya Addanki) is a simple abstraction of the recurring task of communicating information among various components of a parallel network. We have used this problem to test out the learning algorithm because it is clear what the optimal solution is like and it is nontrivial to discover it. Two groups of visible units, designated V_1 and V_2, represent two systems that wish to communicate their states. Each group has v units. In the simple formulation we consider here, each group has only one unit on at a time, so there are only v different states of each group. V_1 and V_2 are not connected directly but both are connected to a group of h hidden units H, with $h < v$ so H may act as a limited capacity bottleneck through which information about the states of V_1 and V_2 must be squeezed. Since all simulations began with all weights set to zero, finding a solution to such a problem requires that the two visible groups come to agree upon the meanings of a set of codes without any *a priori* conventions for communication through H.

To permit perfect communication between the visible groups, it must be the case that $h \geq log_2 v$. We investigated minimal cases in which $h = log_2 v$, and cases when h was somewhat larger than $log_2 v$. In all cases, the environment for the network consisted of v equiprobable vectors of length $2v$ which specified that one unit in V_1 and the corresponding unit in V_2 should be on together with all other units off. Each visible group is completely connected internally and each is completely connected to H, but the units in H are not connected to each other.

Because of the severe speed limitation of simulation on a sequential machine, and because the learning requires many annealings, we have primarily experimented with small versions of the encoder problem. For example, Figure 2 shows a good solution to a "*4-2-4*" encoder problem in

[3] The problem of finding a suitable value for ϵ disappears if one performs a line search for the lowest value of G along the current direction of steepest descent, but line searches are inapplicable in this case. *Only* the local gradient is available. There are bounds on the second derivative that can be used to pick conservative values of ϵ (Mark Derthick, personal communication, 1984), and methods of this kind are currently under investigation.

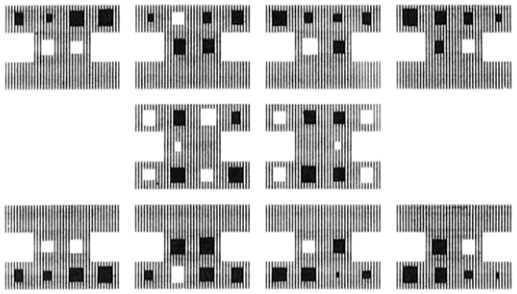

Figure 2. A solution to an encoder problem. The link weights are displayed using a recursive notation. Each unit is represented by a shaded T-shaped box; from top to bottom the rows of boxes represent groups V_1, H, and V_2. Each shaded box is a map of the entire network, showing the strengths of that unit's connections to other units. At each position in a box, the size of the white (positive) or black (negative) rectangle indicates the magnitude of the weight. In the position that would correspond to a unit connecting to itself (the second position in the top row of the second unit in the top row, for example), the bias is displayed. All connections between units appear twice in the diagram, once in the box for each of the two units being connected. For example, the black square in the top right corner of the leftmost unit of V_1 represents the same connection as the black square in the top left corner of the rightmost unit of V_1. This connection has a weight of -30.

which $v = 4$ and $h = 2$. The interconnections between the visible groups and H have developed a binary coding—each visible unit causes a different pattern of *on* and *off* states in the units of H, and corresponding units in V_1 and V_2 support identical patterns in H. Note how the bias of the second unit of V_1 and V_2 is positive to compensate for the fact that the code which represents that unit has all the H units turned off.

4.1. The 4-2-4 Encoder

The experiments on networks with $v = 4$ and $h = 2$ were performed using the following learning cycle:

1. *Estimation of* p_{ij}: Each environmental vector in turn was clamped over the visible units. For each environmental vector, the network was allowed to reach equilibrium twice. Statistics about how often pairs of units were both on together were gathered at equilibrium. To prevent the weights from growing too large we used the "noisy" clamping technique described in Section 3.2. Each *on* bit of a clamped vector was set to *off* with a probability of 0.15 and each *off* bit was set to *on* with a probability of 0.05.
2. *Estimation of* p'_{ij}: The network was completely unclamped and allowed to reach equilibrium at a temperature of 10. Statistics about

co-occurrences were then gathered for as many annealings as were used to estimate p_{ij}.

3. *Updating the weights:* All weights in the network were incremented or decremented by a fixed weight-step of 2, with the sign of the increment being determined by the sign of $p_{ij} - p'_{ij}$.

When a settling to equilibrium was required, all the unclamped units were randomized with equal probability on or off (corresponding to raising the temperature to infinity), and then the network was allowed to run for the following times at the following temperatures: [2@20, 2@15, 2@12, 4@10].[4] After this annealing schedule it was assumed that the network had reached equilibrium, and statistics were collected at a temperature of 10 for 10 units of time.

We observed three main phases in the search for the global minimum of G, and found that the occurrence of these phases was relatively insensitive to the precise parameters used. The first phase begins with all the weights set to zero, and is characterized by the development of negative weights throughout most of the network, implementing two winner-take-all networks that model the simplest aspect of the environmental structure—only one unit in each visible group is normally active at a time. In a *4-2-4* encoder, for example, the number of possible patterns over the visible units is 2^8. By implementing a winner-take-all network among each group of four this can be reduced to 4×4 low energy patterns. Only the final reduction from 2^4 to 2^2 low energy patterns requires the hidden units to be used for communicating between the two visible groups. Figure 3a shows a *4-2-4* encoder network after four learning cycles.

Although the hidden units are exploited for inhibition in the first phase, the lateral inhibition task can be handled by the connections within the visible groups alone. In the second phase, the hidden units begin to develop positive weights to some of the units in the visible groups, and they tend to maintain symmetry between the sign and approximate magnitude of a connection to a unit in V_1 and the corresponding unit in V_2. The second phase finishes when every hidden unit has significant connection weights to each unit in V_1 and analogous weights to each unit in V_2, and most of the different codes are being used, but there are some codes that are used more than once and some not at all. Figure 3b shows the same network after 60 learning cycles.

Occasionally, all the codes are being used at the end of the second phase in which case the problem is solved. Usually, however, there is a third and longest phase during which the learning algorithm sorts out the remaining conflicts and finds a global minimum. There are two basic mechanisms

[4] One unit of time is defined as the time required for each unit to be given, on average, one chance to change its state. This means that if there are n unclamped units, a time period of 1 involves n random probes in which some unit is given a chance to change its state.

involved in the sorting out process. Consider the conflict between the first and fourth units in Figure 3b, which are both employing the code $< -, + >$. When the system is running without environmental input, the two units will be on together quite frequently. Consequently, $p'_{1,4}$ will be higher than $p_{1,4}$ because the environmental input tends to prevent the two units from being on together. Hence, the learning algorithm keeps decreasing the weight of the connection between the first and fourth units in each group, and they come to inhibit each other strongly. (This effect explains the variations in inhibitory weights in Figure 2. Visible units with similar codes are the ones that inhibit each other strongly.) Visible units thus compete for "territory" in the space of possible codes, and this repulsion effect causes codes to migrate away from similar neighbors. In addition to the repulsion effect, we observed another process that tends to eventually bring the unused codes adjacent (in terms of hamming distance) to codes that are involved in a conflict. The mechanics of this process are somewhat subtle and we do not take the time to expand on them here.

The third phase finishes when all the codes are being used, and the weights then tend to increase so that the solution locks in and remains stable against the fluctuations caused by random variations in the co-occurrence statistics. (Figure 2 is the same network shown in Figure 3, after 120 learning cycles.)

In 250 different tests of the *4-2-4* encoder, it always found one of the global minima, and once there it remained there. The median time required to discover four different codes was 110 learning cycles. The longest time was 1810 learning cycles.

4.2. The 4-3-4 Encoder

A variation on the binary encoder problem is to give H more units than are absolutely necessary for encoding the patterns in V_1 and V_2. A simple example is the *4-3-4* encoder which was run with the same parameters as the *4-2-4* encoder. In this case the learning algorithm quickly finds four different codes. Then it always goes on to modify the codes so that they are optimally spaced out and no pair differ by only a single bit, as shown in Figure 4. The median time to find four well-spaced codes was 270 learning cycles and the maximum time in 200 trials was 1090.

4.3. The 8-3-8 Encoder

With $v = 8$ and $h = 3$ it took many more learning cycles to find all 8 three-bit codes. We did 20 simulations, running each for 4000 learning cycles using the same parameters as for the *4-2-4* case (but with a probability of 0.02 of reversing each *off* unit during noisy clamping). The algorithm found all 8

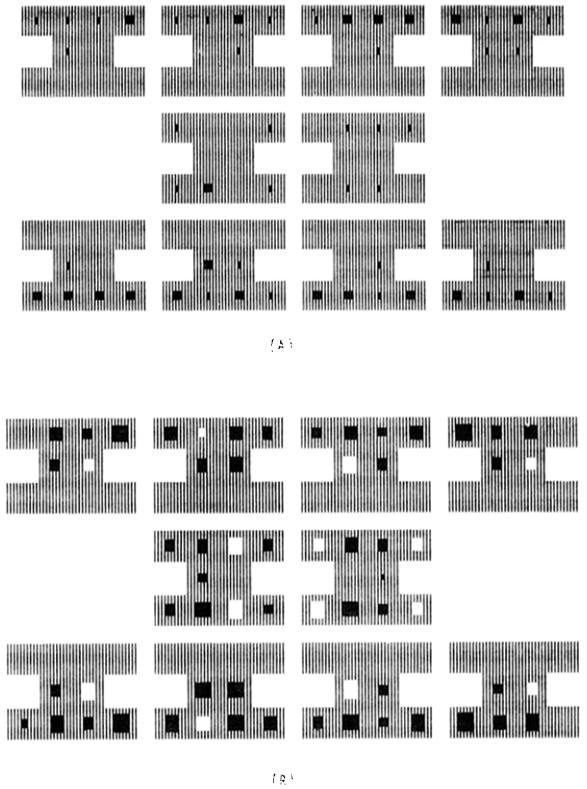

Figure 3. Two phases in the development of the perfect binary encoding shown in Figure 2. The weights are shown (A) after 4 learning trials and (B) after 60 learning trials.

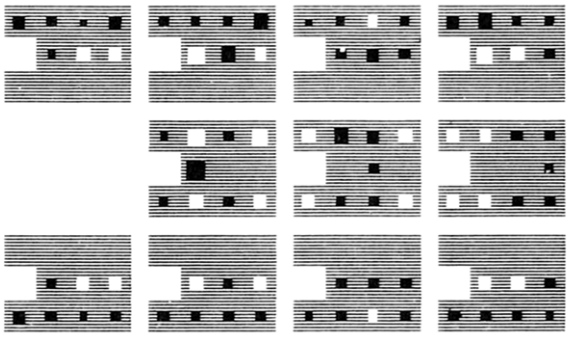

Figure 4. A 4-3-4 encoder that has developed optimally spaced codes.

codes in 16 out of 20 simulations and found 7 codes in the rest. The median time to find 7 codes was 210 learning cycles and the median time to find all 8 was 1570 cycles.

The difficulty of finding all 8 codes is not surprising since the fraction of the weight space that counts as a solution is much smaller than in the *4-2-4* case. Sets of weights that use 7 of the 8 different codes are found fairly

rapidly and they constitute local minima which are far more numerous than the global minima and have almost as good a value of G. In this type of G-space, the learning algorithm must be carefully tuned to achieve a global minimum, and even then it is very slow. We believe that the G-spaces for which the algorithm is well-suited are ones where there are a great many possible solutions and it is not essential to get the very best one. For large networks to learn in a reasonable time, it may be necessary to have enough units and weights and a liberal enough specification of the task so that no single unit or weight is essential. The next example illustrates the advantages of having some spare capacity.

4.4. The 40-10-40 Encoder

A somewhat larger example is the *40-10-40* encoder. The 10 units in H are almost twice the theoretical minimum, but H still acts as a limited bandwidth bottleneck. The learning algorithm works well on this problem. Figure 5 shows its performance when given a pattern in V_1 and required to settle to the corresponding pattern in V_2. Each learning cycle involved annealing once with each of the 40 environmental vectors clamped, and the same number of times without clamping. The final performance asymptotes at 98.6% correct.

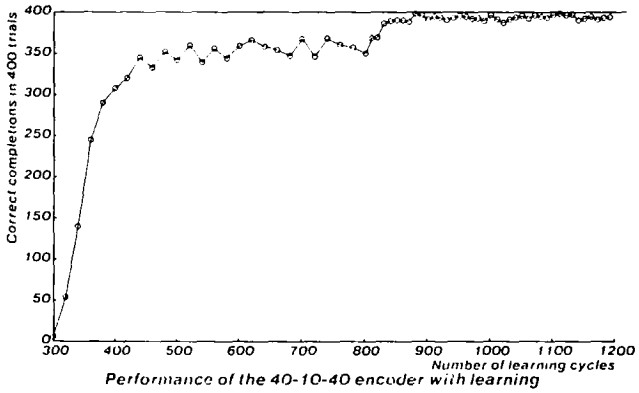

Figure 5. Completion accuracy of a 40-10-40 encoder during learning. The network was tested by clamping the states of the units in V_1 and letting the remainder of the network reach equilibrium. If just the correct unit was on in V_2, the test was successful. This was repeated 10 times for each of the 40 units in V_1. For the first 300 learning cycles the network was run without connecting up the hidden units. This ensured that each group of 40 visible units developed enough lateral inhibition to implement an effective winner-take-all network. The hidden units were then connected up and for the next 500 learning cycles we used "noisy" clamping, switching *on* bits to *off* with a probability of 0.1 and *off* bits to *on* with a probability of 0.0025. After this we removed the noise and this explains the sharp rise in performance after 800 cycles. The final performance asymptotes at 98.6% correct.

The codes that the network selected to represent the patterns in V_1 and V_2 were all separated by a hamming distance of at least 2, which is very unlikely to happen by chance. As a test, we compared the weights of the connections between visible and hidden units. Each visible unit has 10 weights connecting it to the hidden units, and to avoid errors, the 10 dimensional weight vectors for two different visible units should not be too similar. The cosine of the angle between two vectors was used as a measure of similarity, and no two codes had a similarity greater than 0.73, whereas many pairs had similarities of 0.8 or higher when the same weights were randomly rearranged to provide a control group for comparison.

To achieve good performance on the completion tests, it was necessary to use a very gentle annealing schedule during testing. The schedule spent twice as long at each temperature and went down to half the final temperature of the schedule used during learning. As the annealing was made faster, the error rate increased, thus giving a very natural speed/accuracy trade-off. We have not pursued this issue any further, but it may prove fruitful because some of the better current models of the speed/accuracy trade-off in human reaction time experiments involve the idea of a biased random walk (Ratcliff, 1978), and the annealing search gives rise to similar underlying mathematics.

5. REPRESENTATION IN PARALLEL NETWORKS

So far, we have avoided the issue of how complex concepts would be represented in a Boltzmann machine. The individual units stand for "hypotheses," but what is the relationship between these hypotheses and the kinds of concepts for which we have words? Some workers suggest that a concept should be represented in an essentially "local" fashion: The activation of one or a few computing units is the representation for a concept (Feldman & Ballard, 1982); while others view concepts as "distributed" entities: A particular pattern of activity over a large group of units represents a concept, and different concepts corresponds to *alternative* patterns of activity over the same group of units (Hinton, 1981).

One of the better arguments in favor of local representations is their inherent modularity. Knowledge about relationships between concepts is localized in specific connections and is therefore easy to add, remove, and modify, if some reasonable scheme for forming hardware connections can be found (Fahlman, 1980; Feldman, 1982). With distributed representations, however, the knowledge is diffuse. This is good for tolerance to local hardware damage, but it appears to make the design of modules to perform specific functions much harder. It is particularly difficult to see how new distributed representations of concepts could originate spontaneously.

In a Boltzmann machine, a distributed representation corresponds to an energy minimum, and so the problem of creating a good collection of distributed representations is equivalent to the problem of creating a good "energy landscape." The learning algorithm we have presented is capable of solving this problem, and it therefore makes distributed representations considerably more plausible. The diffuseness of any one piece of knowledge is no longer a serious objection, because the mathematical simplicity of the Boltzmann distribution makes it possible to manipulate all the diffuse local weights in a coherent way on the basis of purely local information. The formation of a simple set of distributed representations is illustrated by the encoder problems.

5.1. Communicating Information between Modules

The encoder problem examples also suggest a method for communicating symbols between various components of a parallel computational network. Feldman and Ballard (1982) present sketches of two implementations for this task; using the example of the transmission of the concept "wormy apple" from where it is recognized in the perceptual system to where the phrase "wormy apple" can be generated by the speech system. They argue that there appears to be only two ways that this could be accomplished. In the first method, the perceptual information is encoded into a set of symbols that are then transmitted as messages to the speech system, where they are decoded into a form suitable for utterance. In this case, there would be a set of general-purpose communciation lines, analogous to a bus in a conventional computer, that would be used as the medium for all such messages from the visual system to the speech system. Feldman and Ballard describe the problems with such a system as:

- Complex messages would presumably have to be transmitted sequentially over the communication lines.
- Both sender and receiver would have to learn the common code for each new concept.
- The method seems biologically implausible as a mechanism for the brain.

The alternative implementation they suggest requires an individual, dedicated hardware pathway for each concept that is communicated from the perceptual system to the speech system. The idea is that the simultaneous activation of "apple" and "worm" in the perceptual system can be transmitted over private links to their counterparts in the speech system. The critical issues for such an implementation are having the necessary connections available between concepts, and being able to establish new con-

nection pathways as new concepts are learned in the two systems. The main point of this approach is that the links between the computing units carry simple, nonsymbolic information such as a single activation level.

The behavior of the Boltzmann machine when presented with an encoder problem demonstrates a way of communicating concepts that largely combines the best of the two implementations mentioned. Like the second approach, the computing units are small, the links carry a simple numeric value, and the computational and connection requirements are within the range of biological plausibility. Like the first approach, the architecture is such that many different concepts can be transmitted over the same communication lines, allowing for effective use of limited connections. The learning of new codes to represent new concepts emerges automatically as a cooperative process from the G-minimization learning algorithm.

6. CONCLUSION

The application of statistical mechanics to constraint satisfaction searches in parallel networks is a promising new area that has been discovered independently by several other groups (Geman & Geman, 1983; Smolensky, 1983). There are many interesting issues that we have only mentioned in passing. Some of these issues are discussed in greater detail elsewhere: Hinton and Sejnowski (1983b) and Geman and Geman (1983) describe the relation to Bayesian inference and to more conventional relaxation techniques; Fahlman, Hinton, and Sejnowski (1983) compare Boltzmann machines with some alternative parallel schemes, and discuss some knowledge representation issues. An expanded version of this paper (Hinton, Sejnowski, & Ackley, 1984) presents this material in greater depth and discusses a number of related issues such as the relationship to the brain and the problem of sequential behavior. It also shows how the probabilistic decision function could be realized using gaussian noise, how the assumptions of symmetry in the physical connections and of no time delay in transmission can be relaxed, and describes results of simulations on some other tasks.

Systems with symmetric weights form an interesting class of computational device because their dynamics is governed by an energy function.[5] This is what makes it possible to analyze their behavior and to use them for iterative constraint satisfaction. In their influential exploration of perceptrons, Minsky and Papert (1968, p. 231) concluded that: "Multilayer machines with loops clearly open up all the questions of the general theory of automata." Although this statement is very plausible, recent developments

[5] One can easily write down a similar energy function for asymmetric networks, but this energy function does not govern the behavior of the network when the links are given their normal causal interpretation.

suggest that it may be misleading because it ignores the symmetric case, and it seems to have led to the general belief that it would be impossible to find powerful learning algorithms for networks of perceptron-like elements.

We believe that the Boltzmann Machine is a simple example of a class of interesting stochastic models that exploit the close relationship between Boltzmann distributions and information theory.

> All of this will lead to theories [of computation] which are much less rigidly of an all-or-none nature than past and present formal logic. They will be of a much less combinatorial, and much more analytical, character. In fact, there are numerous indications to make us believe that this new system of formal logic will move closer to another discipline which has been little linked in the past with logic. This is thermodynamics, primarily in the form it was received from Boltzmann, and is that part of theoretical physics which comes nearest in some of its aspects to manipulating and measuring information.
>
> (John Von Neumann, *Collected Works* Vol. 5, p. 304)

APPENDIX: DERIVATION OF THE LEARNING ALGORITHM

When a network is free-running at equilibrium the probability distribution over the visible units is given by

$$P'(V_\alpha) = \sum_\beta P'(V_\alpha \wedge H_\beta) = \frac{\sum_\beta e^{-E_{\alpha\beta}/T}}{\sum_{\lambda\mu} e^{-E_{\lambda\mu}/T}} \tag{11}$$

where V_α is a vector of states of the visible units, H_β is a vector of states of the hidden units, and $E_{\alpha\beta}$ is the energy of the system in state $V_\alpha \wedge H_\beta$

$$E_{\alpha\beta} = -\sum_{i<j} w_{ij} s_i^{\alpha\beta} s_j^{\alpha\beta}.$$

Hence,

$$\frac{\partial e^{-E_{\alpha\beta}/T}}{\partial w_{ij}} = \frac{1}{T} \; s_i^{\alpha\beta} s_j^{\alpha\beta} e^{-E_{\alpha\beta}/T}.$$

Differentiating (11) then yields

$$\frac{\partial P'(V_\alpha)}{\partial w_{ij}} = \frac{\frac{1}{T}\sum_\beta e^{-E_{\alpha\beta}/T} s_i^{\alpha\beta} s_j^{\alpha\beta}}{\sum_{\alpha\beta} e^{-E_{\alpha\beta}/T}} - \frac{\sum_\beta e^{-E_{\alpha\beta}/T} \frac{1}{T}\sum_{\lambda\mu} e^{-E_{\lambda\mu}/T} s_i^{\lambda\mu} s_j^{\lambda\mu}}{\left(\sum_{\lambda\mu} e^{-E_{\lambda\mu}/T}\right)^2}$$

$$= \frac{1}{T} \left[\sum_{\beta} P'(V_\alpha \wedge H_\beta) s_i^{\alpha\beta} s_j^{\alpha\beta} - P'(V_\alpha) \sum_{\lambda\mu} P'(V_\lambda \wedge H_\mu) s_i^{\lambda\mu} s_j^{\lambda\mu} \right].$$

This derivative is used to compute the gradient of the G-measure

$$G = \sum_{\alpha} P(V_\alpha) \ln \frac{P(V_\alpha)}{P'(V_\alpha)}$$

where $P(V_\alpha)$ is the clamped probability distribution over the visible units and is independent of w_{ij}. So

$$\frac{\partial G}{\partial w_{ij}} = -\sum_{\alpha} \frac{P(V_\alpha)}{P'(V_\alpha)} \frac{\partial P'(V_\alpha)}{\partial w_{ij}}$$

$$= -\frac{1}{T} \sum_{\alpha} \frac{P(V_\alpha)}{P'(V_\alpha)} \left[\sum_{\beta} P'(V_\alpha \wedge H_\beta) s_i^{\alpha\beta} s_j^{\alpha\beta} - P'(V_\alpha) \sum_{\lambda\mu} P'(V_\lambda \wedge H_\mu) s_i^{\lambda\mu} s_j^{\lambda\mu} \right].$$

Now,

$$P(V_\alpha \wedge H_\beta) = P(H_\beta | V_\alpha) P(V_\alpha),$$
$$P'(V_\alpha \wedge H_\beta) = P'(H_\beta | V_\alpha) P'(V_\alpha),$$

and

$$P'(H_\beta | V_\alpha) = P(H_\beta | V_\alpha). \tag{12}$$

Equation (12) holds because the probability of a hidden state given some visible state must be the same in equilibrium whether the visible units were clamped in that state or arrived there by free-running. Hence,

$$P'(V_\alpha \wedge H_\beta) \frac{P(V_\alpha)}{P'(V_\alpha)} = P(V_\alpha \wedge H_\beta).$$

Also,

$$\sum_{\alpha} P(V_\alpha) = 1.$$

Therefore,

$$\frac{\partial G}{\partial w_{ij}} = -\frac{1}{T} [p_{ij} - p'_{ij}]$$

where

$$p_{ij} \stackrel{def}{=} \sum_{\alpha\beta} P(V_\alpha \wedge H_\beta) s_i^{\alpha\beta} s_j^{\alpha\beta}$$

and

$$p'_{ij} \stackrel{def}{=} \sum_{\lambda\mu} P'(V_\lambda \wedge H_\mu) s_i^{\lambda\mu} s_j^{\lambda\mu}$$

as given in (9).

The Boltzmann Machine learning algorithm can also be formulated as an input-output model. The visible units are divided into an input set I and an output set O, and an environment specifies a set of conditional probabilities of the form $P(O_\beta|I_\alpha)$. During the "training" phase the environment clamps both the input and output units, and $p_{ij}s$ are estimated. During the "testing" phase the input units are clamped and the output units and hidden units free-run, and $p'_{ij}s$ are estimated. The appropriate G measure in this case is

$$G = \sum_{\alpha\beta} P(I_\alpha \wedge O_\beta) \ln \frac{P(O_\beta|I_\alpha)}{P'(O_\beta|I_\alpha)}$$

Similar mathematics apply in this formulation and $\partial G/\partial w_{ij}$ is the same as before.

REFERENCES

Berliner, H. J., & Ackley, D. H. (1982, August). The QBKG system: Generating explanations from a non-discrete knowledge representation. *Proceedings of the National Conference on Artificial Intelligence AAAI-82*, Pittsburgh, PA, 213-216.

Binder, K. (Ed.) (1978). *The Monte-Carlo method in statistical physics*. New York: Springer-Verlag.

Fahlman, S. E. (1980, June). The Hashnet Interconnection Scheme. (Tech. Rep. No. CMU-CS-80-125), Carnegie-Mellon University, Pittsburgh, PA.

Fahlman, S. E., Hinton, G. E., & Sejnowski, T. J. (1983, August). Massively parallel architectures for AI: NETL, Thistle, and Boltzmann Machines. *Proceedings of the National Conference on Artificial Intelligence AAAI-83*, Washington, DC, 109-113.

Feldman, J. A. (1982). Dynamic connections in neural networks. *Biological Cybernetics, 46*, 27-39.

Feldman, J. A., & Ballard, D. H. (1982). Connectionist models and their properties. *Cognitive Science, 6*, 205-254.

Geman, S., & Geman, D. (1983). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. Unpublished manuscript.

Grimson, W. E. L. (1981). *From images to surfaces*. Cambridge, MA: MIT Press.

Hinton, G. E. (1977). *Relaxation and its role in vision*. Unpublished doctoral dissertation, University of Edinburgh. Described in D. H. Ballard & C. M. Brown (Eds.), *Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, 408-430.

Hinton, G. E. (1981). Implementing semantic networks in parallel hardware. In G. E. Hinton & J. A. Anderson (Eds.), *Parallel Models of Associative Memory*. Hillsdale, NJ: Erlbaum.

Hinton, G. E., & Anderson, J. A. (1981). *Parallel models of associative memory*. Hillsdale, NJ: Erlbaum.

Hinton, G. E., & Sejnowski, T. J. (1983a, May). Analyzing cooperative computation. *Proceedings of the Fifth Annual Conference of the Cognitive Science Society*. Rochester, NY.

Hinton, G. E., & Sejnowski, T. J. (1983b, June). Optimal perceptual inference. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Washington, DC, pp. 448–453.

Hinton, G. E., Sejnowski, T. J., & Ackley, D. H. (1984, May). *Boltzmann Machines: Constraint satisfaction networks that learn*. (Tech. Rep. No. CMU-CS-84-119). Pittsburgh, PA: Carnegie-Mellon University.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences USA*, 79, 2554–2558.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*, 671–680.

Kullback, S. (1959). *Information theory and statistics*. New York: Wiley.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., & Teller, E. (1953). Equation of state calculations for fast computing machines. *Journal of Chemical Physics, 6*, 1087.

Minsky, M., & Papert, S. (1968). *Perceptrons*. Cambridge, MA: MIT Press.

Newell, A. (1982). *Intellectual issues in the history of artificial intelligence*. (Tech. Rep. No. CMU-CS-82-142). Pittsburgh, PA: Carnegie-Mellon University.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall, 1972.

Ratcliff, R. (1978). A theory of memory retrieval. *Psychological Review, 85*, 59–108.

Renyi, A. (1962). *Probability theory*. Amsterdam: North-Holland.

Rosenblatt, F. (1961). *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*. Washington, DC: Spartan.

Smolensky, P. (1983, August). Schema selection and stochastic inference in modular environments. *Proceedings of the National Conference on Artificial Intelligence AAAI-83*. Washington, DC. 109–113.

Terzopoulos, D. (1984). *Multiresolution computation of visible-surface representations*. Unpublished doctoral dissertation, MIT, Cambridge, MA.

Waltz, D. L. (1975). Understanding line drawings of scenes with shadows. In P. Winston (Ed.), *The Psychology of Computer Vision*. New York: McGraw-Hill.

Winston, P. H. (1984). *Artificial Intelligence*. (2nd ed.) Reading, MA: Addison-Wesley.