Solving Puzzles with a Connectionist Network Using a Hill-Climbing Heuristic

Alan H. Kawamoto

Dept. of Psychology
Carnegie-Mellon University
Pittsburgh, PA 15217
(412) 268-3157

ABSTRACT

A connectionist network has been used to simulate the solution, using a hill-climbing heuristic, of the DOG- > CAT puzzle (changing 1 letter at a time, generate a sequence of 3-letter words beginning with *DOG* and ending with *CAT)* and a simpler variant of the 8-tile puzzle, the dog-cat-mouse (DCM) puzzle devised by Klahr (1985). Distributed representations have been used to represent the different possible states of the puzzles. These states are learned by the network and become local energy minima of the system. To simulate the sequence of states corresponding to a solution of the puzzle, the initial state of the network is set to the start state, and the goal state is presented to the network as a continuous input. A sequence of states is generated by habituation, a short-term modification of the connection strengths whenever all the elements in the network are maximally or minimally activated, and by exploiting the property that successive states comprising the solution are similar.

Puzzles and games have been studied extensively because they illustrate the importance of search in issues of problem solving. In the classic 8-puzzle, for example, 8 uniquely numbered tiles fit into a 3-by-3 matrix with one open space. To solve the puzzle, a tile adjacent to the open space is slid into that position, which, in turn, creates a new configuration with the open space now in an adjacent position. This continues until the goal state (a particular configuration of the tiles) is attained.

Although quite a large number of distinct configurations exist (9! = 362,880, excluding rotations and reflections), a few moves are sufficient to get to the goal state from any starting state. From any given configuration, there are 2, 3, or 4 possible successive states depending on whether the open space is at a corner, an edge, or the center, respectively. If moves back to the previous state are not allowed, even fewer possibilities arise. The set of configurations that are possible comprises the *problem space* or *state space*. The different ways of transforming one state to another are known as *operators*.

One approach to solve this puzzle involves the use of *search trees* (or *graphs)*. Each node in the tree represents a particular configuration of the puzzle, and arcs connect possible successive configurations. Beginning at the root of the tree, the start state, the tree can be systematically searched in a breadth-first (exploring all the nodes at one level before going to the next lower level) or depth-first (following a path for a fixed number of levels, then retracing that path and exploring alternative paths if the search fails) manner.

In contrast to the algorithmic solution described above, a simple heuristic can be used instead. One such heuristic, *hill-climbing,* consists of selecting that child of the current node that is closest to the goal state, then selecting the best child of that node while ignoring its siblings and offspring of those siblings. No memory of previous states considered, nor the path that led to the current state need be stored. Such methods are appealing because they are usually

faster (when successful) and seem to mirror at least some of the processes of human problem solvers.

One important characteristic of possible successive states that will be exploited in this study is that changes from one state to another are incremental, with successive states being as similar as they could possibly be without being identical. Note, however, that not all such minimally distinct states are possible successors of each other. In the 8-puzzle, for example, only those involving the movement of a tile from its current position to an adjacent empty position are possible successors, whereas configurations with two tiles switched are not.

In implementations to date, each state has been been represented as a distinct node, with arcs representing legal moves between two states. A distributed representation (where each state is represented as a pattern of activity over a set of features) can, however, more naturally exploit the hill-climbing heuristic. * There has been a lot of discussion of distributed representations (Anderson, Silverstein, Ritz, and Jones, 1977; Hinton, McClelland, and Rumelhart, 1986; Knapp and Anderson, 1984; McClelland and Rumelhart, 1985) by investigators using an approach known in some circles as parallel distributed processing, or more generally, connectionism, that explores parallel computing architectures. Such distributed representations have been shown to be important in issues of learning and generalization. This study shows that this representation scheme can be exploited in issues of problem solving as well.

To date, one drawback with a connectionist approach arises from the fact that once the network of elements settle into a state, the network remains in that state. Thus, criticisms exist that while a solution bound by a set of constraints can be found in parallel, serial behavior requiring transitions to successive states cannot be simulated by these networks.

Rumelhart, Smolensky, McClelland, and Hinton (1986) pointed out that the dilemna regarding sequences of states can be resolved by noting that the system can change with a change in the external environment, a change that can be effected in some cases by the experiencer (e.g., game playing). They also noted that these ideas can be extended in a way that allows these networks to simulate mental simulation.

There is, however, a class of stimuli that are *multistable* (e.g., the Necker cube). In these cases, a fixed stimulus can be perceived in two different configurations that alternate with each other. These stimuli suggest an alternative based on the notion of habituation, whereby continued activity of a pair of elements results in a short-term attenuation of that connection (Kawamoto & Anderson, 1985). This property will be used to simulate the solution, using "weak" methods, of two simple puzzles, the *DOG -> CAT* puzzle (i.e., change *DOG* to *CAT* by replacing one letter such that each letter triplet is a legal word) as well as a simpler variant of the 8-puzzle devised by Klahr (1985) known as the dog-cat-mouse (DCM) problem.

THE MODEL

The approach used here is based on the work of Anderson and his colleagues (Anderson, Silverstein, Ritz, & Jones, 1977; Anderson, 1983). Since this model has been described in detail elsewhere (Anderson, et al., 1977; Anderson, 1983; Kawamoto & Anderson, 1985), only the major points and more recent developments will be noted here. The use of a network of simple processing elements operating in parallel to simulate cognitive phenomenon is very similar to many others (see Feldman, 1985; McClelland & Rumelhart, 1986; Rumelhart & McClelland, 1986).

Network architecture

Auto-associative network. The principle network consisted of 216 elements, with each element forming connections to every other element in the network (hence the name, auto-

* The reader is cautioned to be aware that two closely related notions of hill-climbing are used in this paper. One is from the AI literature on search heursitics, and the other one, quite similar in spirit, arises from the connectionist literature in terms of searching for a particular state (energy minimum). In the framework of this paper, the former influences the selection of subsequent states, and the latter allows the network to find local energy minima. To minimize possible confusion, the terms *relaxing* or *settling* will be used in discussions of finding local energy minima.

associative). The activity of an element is represented as a real value ranging between -1.0 and + 1.0. These limits constrain the activity of the network within a high-dimensionality hypercube and is the basis for referring to this model as the "brain-state-in-a-box." The activity of an element changes with time, where time changes in discrete steps, such that subsequent activity of an element, $\mathbf{x}_i$, is simply the sum of the input, $\mathbf{s}_i$, some fraction, δ, of its activity at the previous iteration, and the activity of all the other elements, $\mathbf{x}_j$, weighted by the connection strength, ω_{ij}. That is,

$$\mathbf{x}_i(t+1) = LIMIT[\mathbf{s}_i + \delta \mathbf{x}_i(t) + \sum_j \omega_{ij}\, \mathbf{x}_j(t)], \quad (1)$$

where *LIMIT* constrains the activity to the range from -1.0 to + 1.0. The strengths of these connections are determined adaptively during the learning phase in the manner described in a later section. Since each element is connected with every other element, a feedback loop is formed and the state of the system continues to change until all the elements in the network are saturated (i.e., minimally or maximally activated).

Associative network. For the DCM puzzle only, the network will be supplemented with a second set of 216 elements whose activity represents the most recent puzzle configuration. Connections from *i* th element of this set of elements forms connections v_{ij} to the *j* th element of the first set of elements. The connections v_{ij} capture the constraint on legal moves.

Representation

In this approach, each state is represented as a pattern of features that are either on (a value of + 1.0) or off (a value of -1.0). For both the DOG -> CAT puzzle and the DCM puzzle, there are always a fixed number of components that comprise the configuration. In the first case, there are always only 3 letters, and in the second case, there are always a fixed number of positions and 3 tokens placed in those positions. A word or configuration is formed simply by concatenating patterns corresponding to letters, or tokens and their positions. A more detailed description of the representations will be postponed until the puzzles themselves are discussed.

Learning

An important aspect of modeling efforts of this type is the training involved in producing a network that successfully generates the desired output for a given input. This is achieved by modifying the connection strengths between pairs of elements to capture the correlations in their activities. One approach introduced by the early investigators of learning in networks (Rosenblatt, 1962; Widrow & Hoff, 1960) is to use "error-correction" schemes. Modification of the synaptic weights proceeds by minimizing (i.e., correcting) the error between the desired output and the actual output. For example, if the pattern **g** (representing a 3-letter word or a particular puzzle configuration) is to be learned, the difference between the correct value of the i *th* element of **g**, $\mathbf{g}_i$, and its actual value after a single iteration through the network, $\mathbf{g}_i'$, is used to determine the extent of the modification. Here, $\mathbf{g}_i$ is simply

$$\mathbf{g}_i' = LIMIT[\sum_j \omega_{ij}\, \mathbf{g}_j]. \quad (2)$$

After a learning trial, each connection strength ω_{ij} is modified by

$$\Delta\omega_{ij} = \eta(\mathbf{g}_i - \mathbf{g}_i')\mathbf{g}_j, \quad (3)$$

where η is a scalar learning constant.

For the DCM puzzle, the connections v_{ij} capture the constraint of allowable moves. In this case, the location of the open position of the puzzle always changes with each move. This has been implemented by associating the pattern representing the open position in each of the 4 different slots with possible successive open positions. These connections were modified according to a Hebbian learning scheme (Anderson, et al., 1977).

Energy

Although learning is generally an important consideration for this modeling approach, this particular aspect is not of primary importance in

this study. Rather, this simulation exploits the property that learning algorithms of the type used here lead to the input patterns becoming local energy minima (i.e., stable states) of the system (Golden, 1986; Hopfield, 1982). In these networks, each successive state, **x**, is more energetically favorable than the previous state. For the DOG -> CAT puzzle, an energy measure analogous to one used by Rumelhart, et al. (1986) is used,

$$E(t) = -\sum_i \sum_j \omega_{ij}\, \mathbf{x}_i(t)\mathbf{x}_j(t) - \sum_i \mathbf{x}_i\,(t)\mathbf{s}_i. \quad (4)$$

The hill-climbing search heuristic arises from the contribution of the stimulus, **s**, and the current activity, **x**. For the DCM puzzle. the additional constraint imposed by allowable transitions yields

$$E(t) = -\sum_i \sum_j \omega_{ij}\, \mathbf{x}_i(t)\mathbf{x}_j(t) - \sum_i \mathbf{x}_i\,(t)\mathbf{s}_i \quad (5)$$
$$-\sum_i \sum_j \nu_{ij}\, \mathbf{c}_j \mathbf{x}_i(t),$$

where **c** represents the most recent stable state.

Habituation

Since the elements at one layer are interconnected, a positive feedback loop is formed. Thus, once all the elements in the network are saturated, they tend to remain saturated. One solution to get the system out of this stable state is to *habituate* the system by a short-term modification of the connection strengths (Kawamoto & Anderson, 1985). Given the stable state **c**, a "corner" of the hypercube, the connections are modified by

$$\Delta\omega_{ij} = \gamma \mathbf{c}_i \mathbf{c}_j, \quad (6)$$

where γ is negative valued scalar constant. This modification takes place on each iteration the system is in the stable state. However, this modification is only temporary as the effect decreases exponentially with time.

Essentially, habituation results in a change in the energy landscape; i.e., the current stable state becomes less energetically favorable. At some point, that state no longer corresponds to a local energy minimum and the state of the network moves away from this point toward a new local energy minimum.

DOG- > CAT PUZZLE

Puzzle Description

The objective of this puzzle is to transform a given 3-letter word, *DOG,* to a different 3-letter word, *CAT,* by replacing a single letter of the current word on each successive move such that each new letter triplet forms a valid word.

Representation

In this simulation, the 30 3-letter words listed in Table 1 were learned by the network. Each letter is represented by a 72-dimensional random vector. These patterns are simply concatenated in the appropriate order to produce the pattern for each word.

Simulation

One aspect of this puzzle that makes it somewhat difficult is the large number of possible operators (25 possible letter replacements at each of 3 letter positions) for any given state, most of which lead to non-legal states (i.e., non-words). Rather than trying all possibilities, one strategy is to substitute a letter from the current word with the letter in the corresponding position from the goal, e.g., C in position 1. (These possiblities would all be "uphill" in terms of the hill-climbing heuristic.) This will limit initial consideration to just 3 very likely possibilities. Once a possibility has been generated, it can be tested to see whether or not the resulting triplet is a word or not. In contrast to this sequential method of considering different possibilities, the approach used here essentially considers all choices in parallel. Moreover, the model, in general, "considers" only legal words.

After the 30 words from Table 1 were learned, the network was tested to determine whether a sequence of stable states corresponding to a solution could be generated. The initial state of the network was set to *DOG,* and on each iteration, the pattern corresponding to the goal state *CAT,* was provided as input to the

Table 1. List of the 30 3-letter words learned by the network.

AIM	DOT	LIP
APE	DUB	MAD
ARC	EEL	PIN
BUN	EGO	ORE
CAR	GNU	ROB
CAT	GUM	SUB
COG	ILL	SUN
COT	IMP	TIP
DIG	IRE	URN
DOG	LED	USE

Table 2. A simulation of the solution DOG -> COG -> COT -> CAT.

0.	DOG
1.	DOG
20.	_OG
23.	_O_
36.	_OG
47.	COG
48.	COG
59.	CO_
78.	_O_
87.	COT
88.	COT
95.	C_T
98.	__T
111.	C_T
114.	CAT

Note: The underscores indicate that the subset of elements are not all saturated.

network. A selected sample of states during the iteration are displayed in Table 2. Whenever the network reaches a stable state, the connections are habituated in the manner described earlier.

The sequence of stable states is *DOG -> COG -> COT -> CAT*. While there are no characters common to both the starting state, *DOG*, and goal state, *CAT*, each successive state is always closer to the solution: i.e., the solution, in terms of heuristic search, is strictly uphill. Note that the model did not generate non-words such as *DAG*. This is a general tendency that arises because the lowest energy states of the network correspond to the words learned during the initial training period.*

There are actually two solutions to this puzzle. In addition to the solution generated here, an alternative is the sequence *DOG -> DOT -> COT -> CAT*. Both *DOT* and *COG* represent equally good states with respect to the goal state. In this particular solution, the nondeterminism has been resolved successfully and a legal stable state was generated.

When an impasse in problem solving is reached, headway can often be made by working backwards from the goal state toward the start state. In the AI literature, such an approach is used in searching simultaneously from the start state to the goal state and from the goal state to the start state. Here too, such an approach can be used, and the network successfully simulates the solution in the reverse direction, *CAT -> COT -> DOT -> DOG*. Note that here, the solution takes *DOT* as an intermediate state rather than *COG*.

Although there are false peaks in this problem, a problem does not arise because the initial state is at the base of the tallest peak. The next puzzle illustrates a case where the network begins at a false peak and gets down from it.

DCM PUZZLE

Puzzle Description

In this variant of the 8-puzzle, there are 3 different pieces, a dog, a cat, and a mouse, that must be placed in the configuration shown in Figure 1. A piece can be moved only to the open position, and only if there is a connection, indicated by the solid lines, from its current posi-

* There is, however, no guarantee that the network will not generate non-words because there are essentially only pair-wise connections between elements comprising a pair of letters. The probability of settling into states corresponding to words increases if there are additional units unique to a given word as proposed by Hinton (1981).

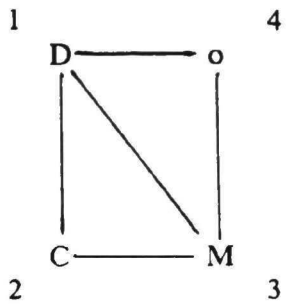

Figure 1. Configuration of the DCM puzzle. There are 4 positions and three tokens, with a token allowed to move only to the open position (if there is a connnection between the two locations).

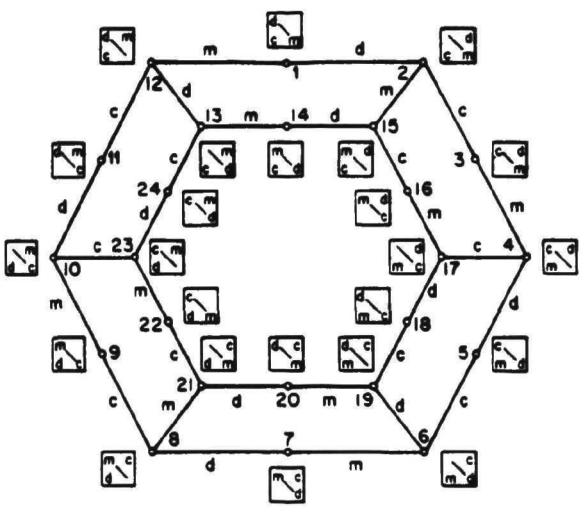

Figure 2. State space representation of the DCM puzzle. [From Klahr, 1985. Copyright 1985 by The Society for Research in Child Development. Reprinted by permission.]

tion to the open position.

Representation

Disregarding rotations and reflections, there are 24 possible configurations. A graph showing all possible states and legal transitions is shown in Figure 2.

Table 3 shows the representation of the 24 possible states of the puzzle. Each configuration is represented in the following way: The piece at each of the 4 positions *(D, C, M,* and *O)* representing Dog, Cat, Mouse, and the Open position, respectively) as well as each piece's position *(d, c,* and *m)* with a numerical subscript from 1 to 4 corresponding to its position), are indicated by a distinct slot.

Table 3. Representation of all possible configurations.

1	DCMO	$d_1c_2m_3$
2	OCMD	$d_4c_2m_3$
3	COMD	$d_4c_1m_3$
4	CMOD	$d_4c_1m_2$
5	CMDO	$d_3c_1m_2$
6	OMDC	$d_3c_4m_2$
7	MODC	$d_3c_4m_1$
8	MDOC	$d_2c_4m_1$
9	MDCO	$d_2c_3m_1$
10	ODCM	$d_2c_3m_4$
11	DOCM	$d_1c_3m_4$
12	DCOM	$d_1c_2m_4$
13	OCDM	$d_3c_2m_4$
14	MCDO	$d_3c_2m_1$
15	MCOD	$d_4c_2m_1$
16	MOCD	$d_4c_3m_1$
17	OMCD	$d_4c_3m_2$
18	DMCO	$d_1c_3m_2$
19	DMOC	$d_1c_4m_2$
20	DOMC	$d_1c_4m_3$
21	ODMC	$d_2c_4m_3$
22	CDMO	$d_2c_1m_3$
23	CDOM	$d_2c_1m_4$
24	CODM	$d_3c_1m_4$

Note: The number for each configuration corresponds to that depicted in Figure 2.

Simulation

Once the legal positions and constraints on the operators have been learned, the ability of the network to simulate the solution of the puzzle was observed. One example run is shown in Table 4. The start state is configuration 14, and the network successively reaches configurations 13, 12, and 1, the goal state.

During the course of running a number of simulations, it was observed that not all states the network settled into corresponded to a configuration. For example, two pieces were sometimes "moved" simultaneously to the open position, creating *2* new open positions and

Table 4. Simulation of solution of the DCM puzzle.

0	MCDO	$d_3c_2m_1$
1	MCDO	$d_3c_2m_1$
19	_CDO	$d_3c_2m_1$
42	_CD_	$d_3c_2m_1$
51	_CD_	d_3c_2__
56	OCD_	d_3c_2__
57	OCD_	$d_3c_2m_4$
58	OCDM	$d_3c_2m_4$
59	OCDM	$d_3c_2m_4$
66	_CDM	$d_3c_2m_4$
89	_C_M	$d_3c_2m_4$
95	_C_M	__c_2m_4
100	_C_M	__c_2__
105	_C_M	d_1c_2__
107	_COM	d_1c_2__
108	_COM	$d_1c_2m_4$
109	DCOM	$d_1c_2m_4$
110	DCOM	$d_1c_2m_4$
117	DC_M	$d_1c_2m_4$
149	DC_M	d_1c_2__
150	D___M	d_1c_2__
154	DC___	d_1c_2__
158	DC___	d_1c_2**
161	DC___	d_1c_2__
163	DC___	__c_2__
167	DC___	__c_2m_3
168	DC_O	__c_2m_3
169	DCMO	__c_2m_3
171	DCMO	$d_1c_2m_3$

Note. The underscores indicate that the corresponding set of units are not saturated. The asterisk indicates that although the set of units are saturated, they do not correspond to a defined pattern.

combination of the two pieces at the position that was formerly empty. This was a manifestation of the non-determinism involved in choosing a successive state and has been observed in young children (Klahr, 1985). In other cases, the equivalent of 2 moves were sometimes taken during one settling period.

In solving puzzles, one way to minimize the path length for a solution is to avoid backup, i.e., not returning to the previous state. Here, this constraint is imposed as a result of habituation. Previous states are less likely to be returned to simply because these states are no longer energetically favorable.

DISCUSSION

In this study, the solution of two puzzles using a hill-climbing search heuristic has been simulated within a connectionist framework. This is implemented by using a distributed representation to capture similarity of the states in the problem space. When the goal state is provided as an input, the more similar a state is to the goal state, the more energetically favorable that state is. Since the network settles into local energy minima, the states the network settles into are similar to the goal state. Once the network settles into a stable state, the connections are habituated. This changes the energy landscape and allows the network to move from the current state to a new stable state that is closer to the goal state.

Networks of this type can also go from one stable state to another (with a fixed input) if the activity of units are stochastic. In such cases, the probability that the network is in a particular state is determined from the Boltzmann distribution (Ackley, Hinton, & Sejnowski, 1985; Selman, 1985).

Although it has been demonstrated here that connectionist networks can simulate solution of puzzles, the behavior simulated is that of a naive problem solver. It would be nice if the network could also learn the *optimal* solution given a particular start state and goal state. An even more ambitious goal is for the network to generate a representation scheme that would allow it to generalize to configurations that were isomorphic. This would almost surely require the network to be able to train hidden units using methods studied by Ackley, et al. (1985) and Rumelhart, Hinton, and Williams (1985).

Acknowledgements

This work was completed while the author was supported by a grant from the Sloan Foun-

dation. The DOG -> CAT puzzle was pointed out by Jeff Shrager, and the DCM puzzle was brought to my attention by Jay McClelland. I would like to thank Jay McClelland for useful discussions of the problem, and Richard Golden for commenting on a draft of the manuscript.

References

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science,* **9,** 147-169.

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

Anderson, J. A. (1983). Cognitive and psychological computation with neural models. *IEEE: Systems, Man, and Cybernetics.* **SMC-13,** 799-815.

Feldman, J. (Ed.). (1985). *Cognitive Science,* **9,** special issue on connectionism.

Golden, R. M. (1986). The "brain-state-in-a-box" neural model is a gradient descent algorithm." *Journal of Mathematical Psychology,* **30,** 73-80.

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., McClelland, J. L., & Rumelhart, D.E. (1986). Distributed representations. in D. R. Rumelhart & J. L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. v. 1: Foundations.* Cambridge, MA: Bransford Books MIT Press.

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.

Kawamoto, A. H., and Anderson, J. A. (1985). A Neural Network Model of Multistable Perception. *Acta Psychologica.* **59,** 35-65.

Klahr, D. (1985). Solving problems with ambiguous subgoal ordering: Preschooler's performance. *Child Developement,* **56.** 940-952.

Knapp, A., & Anderson, J. A. (1984). A signal averaging model for concept formation. *Journal of Experimental Psychology, Learning, Memory, and Cognition.* **10,** 616-637.

McClelland, J. L., and Rumelhart, D. E. (1985). Distributed memory and the representation of general and specific information. *Journal of Experimental Psychology: General.* **114,** 159-188.

McClelland, J. L., and Rumelhart, D. E. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, v. 2:* Cambridge, MA: Bradford Books MIT Press.

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms,* Washington: Spartan Press.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propogation. *ICS Report 8506.*

Rumelhart, D. E., and McClelland, J. L. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. v. 1: Foundations.* Cambridge, MA: Bradford Books/MIT Press.

Rumelhart, D. E., Smolensky, P., McClelland, J. L., & Hinton, G. E. (1986). Parallel distributed processing models of schemata and sequential thought processes. in D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. v. 1: Foundations.* Cambridge, MA: Bransford Books MIT Press.

Selman, B. (1985) Rule-Based Processing in a Connectionist System for Natural Language Understanding. *Technical*Report Computer Systems Research Institute, University of Toronto.

Widrow, G., & Hoff, M. E. (1960). Adaptive switching circuits. *IRE, Western Electronic Show and Convention, Convention Record, Part 4,* 96-104.