UC Merced

Proceedings of the Annual Meeting of the Cognitive Science Society

Title

A Connectionist Model Of Instruction Following

Permalink

https://escholarship.org/uc/item/47w9x9gz

Journal

Proceedings of the Annual Meeting of the Cognitive Science Society, 17(0)

Authors

Noelle, David C.

Cottrell, Garrison W.

Publication Date

1995

Peer reviewed

eScholarship.org Powered by the California Digital Library University of California

A Connectionist Model Of Instruction Following

David C. Noelle & Garrison W. Cottrell
Department of Computer Science & Engineering
University of California, San Diego
9500 Gilman Drive
La Jolla, CA 92093-0114
{dnoelle,gary}@cs.ucsd.edu

Abstract

In this paper we describe a general connectionist model of "learning by being told". Unlike common network models of *inductive* learning which rely on the slow modification of connection weights, our model of *instructed* learning focuses on rapid changes in the activation state of a recurrent network. We view stable distributed patterns of activation in such a network as internal representations of provided advice – representations which can modulate the behavior of other networks. We suggest that the stability of these configurations of activation can arise over the course of learning an instructional language and that these stable patterns should appear as *articulated attractors* in the activation space of the recurrent network. In addition to proposing this general model, we also report on the results of two computational experiments. In the first, networks are taught to respond appropriately to direct instruction concerning a simple mapping task. In the second, networks receive instructions describing procedures for binary arithmetic, and they are trained to *immediately* implement the specified algorithms on pairs of binary numbers. While the networks in these preliminary experiments were not designed to embody the attractor dynamics inherent in our general model, they provide support for this approach by demonstrating the ability of recurrent backpropagation networks to learn an instructional language in the service of some task and thereafter exhibit prompt instruction following behavior.

Introduction

Connectionist models of human learning have focused primarily on problems of *inductive generalization*. They view learning as a process of extracting new knowledge from the statistical properties of a long series of exemplars. While this covers many cases, humans exhibit other learning behaviors which defy description as induction over examples. In particular, we are capable of acquiring new knowledge from *single* highly informative events, such as tasting sushi for the first time, seeing someone operate a new coffee machine, or hearing a lecture. A single sentence can have profound and lasting effects on our behavior. Furthermore, we are capable of integrating such rapidly acquired knowledge with knowledge gained inductively. If connectionism is to provide a sound computational framework for the entire range of human learning behaviors, it must be extended beyond induction.

Towards this end, we propose here a connectionist model of "learning by being told", and we demonstrate a somewhat more modest model of connectionist instruction following. We view "learning from instruction" as involving the demonstration of "appropriate" behavior immediately following the receipt of a linguistic directive. Our goal is to integrate standard connectionist learning methods with such rapid instructed learning to form a single cognitively plausible model. This multistrategy model should help explain both the *operationalization* of instruction into appropriate behavior (Hayes-Roth et al., 1980; Mostow, 1983) and the interaction effects between instructed learning and exemplar-based learning which have been observed in humans.

Our model is based on the observation that typical connectionist weight modification techniques are inherently too slow to account for instructed learning. Of course, large weight changes could be made in response to instruction, but in networks using distributed representations, such rapid weight modification tends to destroy previously acquired knowledge in dramatic and unrealistic ways. We focus instead on modeling a network's response to instruction as changes in its internal *activation state*. We suggest that our prompt response to instruction is best seen as motion in activation space – as the settling of a network's activation state into a (typically novel) basin of attraction corresponding to the received instruction.[1]

This idea may be illustrated by the Necker cube network (McClelland et al., 1986), shown in Figure 1. This small constraint satisfaction network models our perception of the line drawing of a cube, focusing on our tendency to interpret the drawing in one of two distinct ways. The processing elements in this network represent particular depth assignments, "front" or "back", for each vertex in the cube. Connection weights are selected so as to embody constraints on the interpretation of vertices. The result is a recurrent attractor network with two major basins of attraction in activation space. In other words, there are two main configurations of unit activations which are stable over time – one configuration for each interpretation of the Necker cube drawing. Given some starting levels of activation for the processing elements, the network will tend to settle into one of these attractor basins, forming a coherent internal representation of the cube in activation space. We may bias the network toward one interpretation over the other by providing input activation to the appropriate units. By manipulating the input to the units on one side, we can cause the network to rapidly move from one attractor basin to the next. Similarly, humans can be told to see the cube in a different way, and this input can change their perception of it.

The reception of direct instruction may be viewed as a process similar to that embodied in this Necker cube example. In the case of instruction, natural language advice is to be

[1]Thanks are due to Paul Churchland for this notion (Churchland, 1990).

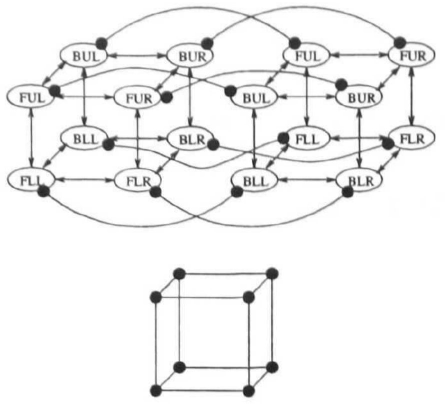

Figure 1: Necker Cube And Attractor Network

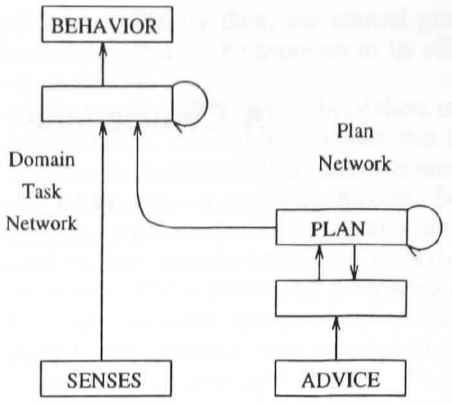

Figure 2: Generic Instructable Network Architecture

rapidly transformed into a coherent internal representation in activation space – a representation which may then be used to modulate the behavior of other networks in the performance of their tasks. As in the Necker cube system, we may encode these internal representations in the attractors of a recurrent network. Learning an instructional language may be seen as training the weights of such an attractor network so as to form a distinct basin of attraction for every meaningful sequence of advice. Most of these *articulated attractors*[2] need not be explicitly trained over the course of language learning, but they may come into existence, nonetheless, via interactions between trained attractors. In this way, some "spurious" attractor basins may actually be seen as *serendipitous basins* in that, while not explicitly trained, they have interpretations that "make sense" in the behavioral domain of the network. Thus, once the language of instruction is learned, such an attractor network may rapidly encode novel advice simply by moving to the corresponding basin of attraction. As in the Necker cube network, this motion in activation space may be driven by appropriate input activation. Following the lead of many connectionist models of natural language processing (Cottrell, 1989; Elman, 1990; St. John and McClelland, 1990), we may encode linguistic instructions as temporal sequences of such input activity, allowing advice to "push" the network into the appropriate basin of attraction. For this strategy to work, such an instructable network must support a distinct stable configuration of activation levels for every instruction sequence which might be presented. If such a combinatorial set of attractors is not present, the network will be limited in the number of different instruction sequences that it can understand and operationalize. With articulated attractors in place, however, novel instructions may be quickly molded into a coherent activation-based modulating force on some task behavior.

Note that this strategy of modeling instructed learning in activation space leaves weight modification in the capable hands of standard connectionist inductive learning algorithms. This allows instruction and induction to proceed in tandem to solve complex learning problems. Also, in addition to weight modifications based on behavioral feedback, Hebbian learning may be used to strengthen and deepen attractors which are regularly visited. Such weight changes would increase the likelihood of visiting those attractors in the future, making common instructional memories easy to instantiate.

Our approach may be illustrated by a generic network architecture, shown in Figure 2. In that diagram, boxes represent layers of processing elements, and arrows between boxes represent complete interconnections between layers. Temporal streams of tokens in an instructional language are presented at the *advice* layer, and this input activity is used to direct the settling of the attractor network at the *plan* layer. The stable configuration of activity levels at the *plan* is then used to modulate the behavior of a task oriented network, much like the "plan" layer of a Jordan network (Jordan, 1986). The connection weights may be trained using a standard inductive learning technique, such as backpropagation, with an error signal provided only on actual task performance. This allows the language of instruction to be learned in the service of a task (St. John, 1992). Such inductive learning may also be used to shape task performance through experience. Once the instructional language is learned, however, new behaviors may be elicited at the speed of activation propagation, without further weight modification, simply by presenting a novel stream of advice.

In summary, our general strategy involves:

- encoding instructions as temporal input sequences;
- training a recurrent network (the *plan network*) to form combinatorial representations of these sequences in its basins of attraction – representations shaped by error feedback from another network (the *domain task network*), which uses activity in the plan network to modulate the performance of some specific task;
- providing further inductive training as appropriate, allowing for the interaction of exemplar-based inductive learning with learning from instruction.

This paper presents an initial examination of this approach to instructed learning. Of particular interest is the plausibility of the claim that a connectionist network may be trained to promptly transform a temporal sequence of instructions into appropriate behavior in some domain. To test this assertion, some networks were made to follow instructions concerning a combinatoric discrete mapping task and others were trained to immediately implement algorithmic instructions for binary arithmetic. These networks were constructed without the ben-

[2]These have also been called *componential attractors* (Plaut and McClelland, 1993).

efit of stable attractors at the plan layer, so activity at that layer was artificially "frozen" once advice was received (St. John and McClelland, 1990). The utility of articulated attractors at the plan layer will be the primary focus of future work. The details of these two experiments are presented below, following a brief overview of related work.

Alternative Approaches

We are not the first to propose a technique for the direct instruction of connectionist models. Indeed, early connectionist networks using "localist" approaches were frequently "instructed" through the direct assignment of network parameters by knowledgeable researchers. This was possible since the networks involved parameters with well understood semantics. In such a framework, "instruction" was essentially "programming", involving the specification of processing elements, connections between elements, and specific connection weights. Some systems automated a large portion of this process, allowing symbolic rules to be directly *compiled* into network parameters (Cottrell, 1985; Davis, 1989). The main drawback of this "weight compilation" approach is the constraint that it places on the representational schemes available to the network. Specifically, since the semantics of network components are fixed, networks of this kind are not free to form arbitrary distributed representations appropriate for the demands of the task.

Some researchers have generated network models which are instructable in this "weight compilation" manner but are still free to develop arbitrary internal distributed representations through inductive training. In general, networks of this kind may only be instructed *before* inductive training begins, because standard connectionist learning methods often change the representational nature of weight values in hard to predict ways, making the direct manipulation of those weights in response to instruction quite problematic. One solution to this problem involves occasionally normalizing weight values back to configurations which are "meaningful" to the weight compilation process. This may be done by identifying and extracting the "rules" embodied in the trained network and then resetting weight values to encode exactly those extracted rules. Once reset in this way, new instructions may be incorporated into the network and the process of inductive learning may begin again. Weight compilation approaches of this kind have been successfully used to encode propositional logic rules (Towell and Shavlik, 1994), "fuzzy" classification rules (Tresp et al., 1993), simple mapping rules (McMillan et al., 1991), the transitions of finite state automata (Giles and Omlin, 1993), and advice for an agent in an artificial environment (Maclin and Shavlik, 1994).

Unfortunately, none of these models provide a connectionist explanation for how instructions are compiled into the network. Also missing is a connectionist mechanism for the rule extraction process which is needed to "reset" the semantics of weight values. Both of these processes, compilation and extraction, require the direct manipulation of the processing elements and the global coordination of weight values. While a connectionist explanation for these dynamic global restructuring processes may be possible, it is not clear what form such an explanation would take.

Perhaps the most important criticism of these models, however, is that the "language of thought" is preordained by the researcher. In order to continue to receive instruction after inductive learning has begun, the models must continuously reformulate their knowledge in the terminology of explicit linguistic instruction. Inductively learned nuances are discarded during rule extraction, leaving these models with a behavior that is consistently analogous to symbolic rule following. In essence, these models are trapped in the first stage of skill acquisition and cannot escape (Rumelhart and Norman, 1978).

By placing instructions in activation space, all of these problems may be avoided. Inductive weight modifications and instruction following may proceed simultaneously, and they may complement or interfere with each other in complex ways.

Instructed Associations

A number of initial experiments have been conducted, focusing solely on the ability of recurrent backpropagation networks (Rumelhart et al., 1986) to learn to operationalize instruction. In particular, issues concerning the benefits of attractor network dynamics for generalization to novel advice have been left for later inquiries. For these initial experiments, unit activations at the plan layer are artificially "frozen" in order to provide a stable internal representation of input instruction sequences. The goal of these early experiments is to demonstrate that a language of instruction may be learned inductively solely from error feedback on actual task performance.

Our first experiment focuses on the ability of these networks to learn to follow instructions concerning a simple associational mapping. Our domain task involves mapping inputs from a finite set into outputs from the same set. Correct mapping behavior is not specified by a collection of labeled examples, however, but by the direct communication of mapping rules. These rules may be viewed as statements such as, "When you see **rock**, say **paper**." Upon presentation of such rules, the network is to *immediately* change its behavior accordingly. Inductive training is used during an initial phase in which the network learns the instructional language, but once this initial training is complete, instruction following may proceed without weight modification. Also, this initial training phase exposes the model to only a fraction of the possible mappings, and the network is expected to generalize its instruction following behavior to novel instruction sequences.

The model, inspired by the architecture of the *Sentence Gestalt* network (St. John and McClelland, 1990), is shown in Figure 3. The boxes represent layers of sigmoidal processing elements and arrows between boxes represent complete interconnections between layers. Layer sizes are shown in parentheses. Symbolic instruction tokens, each encoded as localist "1-out-of-N" activation vectors, were presented sequentially at the advice input layer, and activation was propagated through the recurrent "Plan Network" to produce a pattern of activation at the plan layer. Each mapping rule was encoded as a sequence of three of these instruction tokens (a delimiter followed by the input/output pair) and each complete mapping consisted of three such rules. For example, the nine token advice sequence:

```
⇒ ROCK PAPER
⇒ SCISSORS ROCK
```

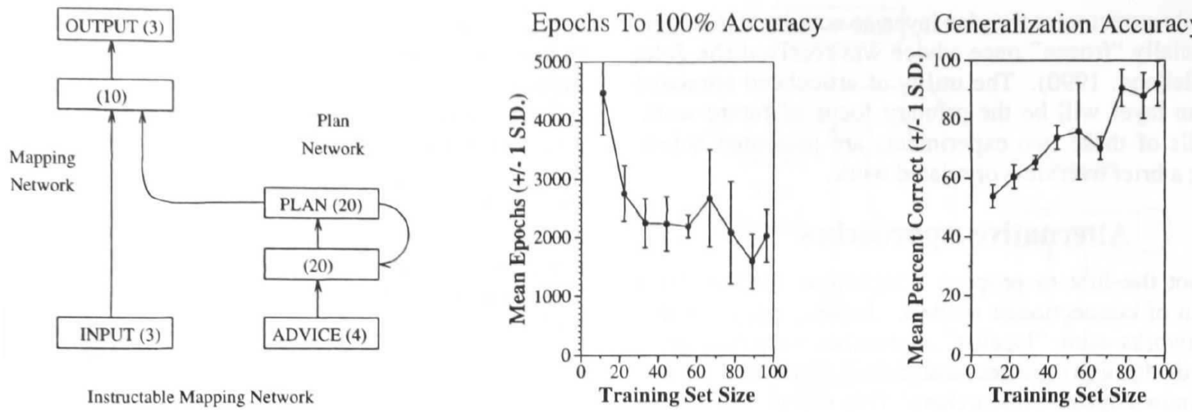

Figure 3: Instructable Mapping Network: Architecture, Training Time, & Generalization

⇒ PAPER SCISSORS

was used to communicate the three rules, "when you see ROCK, say PAPER", "when you see SCISSORS, say ROCK", and "when you see PAPER, say SCISSORS". When the presentation of a sequence of instruction tokens was complete, the activation at the plan layer was "frozen" and used to modulate the behavior of the "Mapping Network" as it performed the desired mapping. Input tokens, also encoded in a localist fashion, were presented at the input layer, and the network's response was read from the output layer. During the initial training phase, mean squared error was then computed at the output layer, based on the most recently presented instructions, and this error was backpropagated to allow for weight modifications throughout the network. The details of this training procedure were much like those of the *Sentence Gestalt* model. In particular, error was backpropagated through recurrent connections for only a single time step, and error was computed after the presentation of each instruction token. A learning rate of 0.05 was used, with no momentum. This initial inductive training period was ended when perfect performance was achieved on a training set of instruction sequences, or when this training set had been presented 5000 times.

Training sets of nine different sizes were examined, and five different random initial weight sets were used. Almost all of these trials resulted in 100% accuracy on the training set within the limit of 5000 epochs. In other words, these networks consistently learned to operationalize the instructions on which they were trained. As shown in Figure 3, generalization performance was also good, with accuracy values on non-training set instruction sequences appearing well above the chance level of 33% correct. Note that training set size is expressed in this figure as a percentage of the total number of possible instruction sequences. Three discrete inputs gave 27 possible mappings. For each mapping, there were 6 possible permutations of the mapping rules, for a total of 162 possible instruction sequences.

While very simple, this discrete mapping task poses interesting problems for inductive connectionist learning. Appropriate system behavior depends entirely on the given instructions. There are no other environmental regularities for the network to discover during training. Once completely trained to "understand" the instructional language, the network is required to modify its behavior *immediately* upon receipt of new instructions, without further weight modification. The way in which this discrete mapping problem forces the network to generalize in a systematic manner over the space of instruction sequences makes this a difficult learning problem, and it also makes it an ideal domain in which to test the power of the proposed network architecture. The results demonstrate that associational mapping instructions can indeed be operationalized by networks of this kind. However, these results also suggest a need for a mechanism to improve generalization performance (Noelle and Cottrell, 1994) – a need which might be met by the incorporation of an attractor network at the plan layer.

Instructed Algorithms

Our second experiment extends our task domain into the realm of sequential procedures. The goal is to demonstrate the ability of these recurrent networks to handle instructions concerning complex sequences of action. To this end, we focus on the domain of arithmetic on arbitrarily large binary integers. In previous work it was shown that recurrent neural networks may be inductively trained to perform a systematic procedure for multi-column addition (Cottrell and Tsung, 1993). Here we wish to examine the possibility of modulating such algorithmic behavior through direct instruction. Instructional tokens, each representing some atomic action, are to be used to communicate a sequential method for binary addition or subtraction to a network, and that network is to be trained to *immediately* implement the specified procedure.

The general structure of the Cottrell & Tsung addition model was used as the basis of our "Arithmetic Network", shown in Figure 4. Under this approach, arithmetic was seen as the iterative transformation of a written representation of two argument integers. The two numbers are assumed to be written so that columns align, and an attentional mechanism is assumed to focus processing on one digit column at a time. Solving an arithmetic problem involves iteratively performing a sequence of actions for each column and then attending to the next. In terms of the network architecture, the digits input layer contained a representation of the two digits in the current column (plus an extra input unit to signal when no columns remained). The actions output layer specified the action to be taken on the current time step, which was one of: "write a given digit as the result for the current column", "announce a carry or borrow", "move to the next column", or "announce

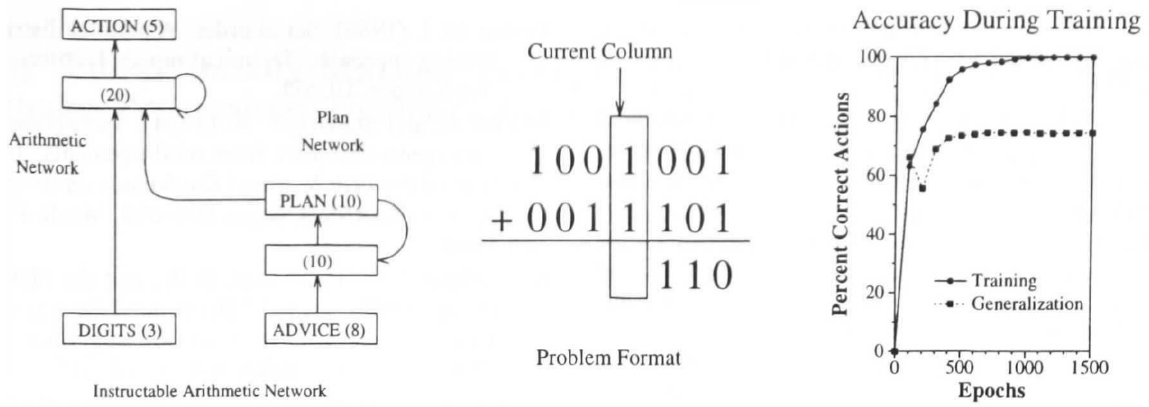

Figure 4: Instructable Arithmetic Network: Architecture, Task Format, & Learning Curve

completion of the current problem". The "Arithmetic Network" included a recurrent hidden layer, which was needed both to produce a sequential output and to "remember" the potential presence of a carry or borrow from the processing of the previous column.

As in the discrete mapping experiment, the behavior of this domain task network was to be specified by a stream of input instruction tokens. Different instruction sequences could specify different orderings for sets of standard actions (e.g., announcing the carry before or after recording the digit sum) or could specify completely different arithmetic operations (e.g., subtraction rather than addition). Each sequence described three actions which were to be applied in the given order to each column of digits. For example, the usual form of addition was specified as, "`WRITE-SUM ANNOUNCE-CARRY NEXT-COLUMN`". Only six such instruction sequences constituted meaningful procedures:

```
  ANNOUNCE-CARRY WRITE-SUM NEXT-COLUMN
WRITE-SUM NEXT-COLUMN ANNOUNCE-PREV-CARRY
  WRITE-SUM ANNOUNCE-CARRY NEXT-COLUMN
 ANNOUNCE-BORROW WRITE-DIFF NEXT-COLUMN
WRITE-DIFF NEXT-COLUMN ANNOUNCE-PREV-BORROW
 WRITE-DIFF ANNOUNCE-BORROW NEXT-COLUMN
```

Despite the extremely small size of this set of possible algorithms, making generalization unlikely, the last of the subtraction sequences was avoided during the initial training phase, to be used, instead, as a test of generalization. Still, the primary goal was to have the network exhibit appropriate behavior when presented with any one of the training set instruction sequences.

This network was operated and trained in much the same manner as the discrete mapping network. Instruction tokens were presented sequentially at the advice input layer, using a localist code, and activity was propagated through the "Plan Network" to produce a plan layer activation vector representing the entire instruction sequence. Activation at the plan layer was then "frozen" and used as an input to the "Arithmetic Network", which then performed the specified procedure on a collection of binary number pairs.[3] Training was conducted as in the discrete mapping network, with error computed after the presentation of each instruction token and backpropagated through recurrent connections for only a single time step.

A large number of experiments were conducted using this architecture and task, varying hidden layer sizes and details of the training regimen. As Figure 4 demonstrates, it was possible to achieve perfect performance on the training set of instruction sequences, but generalization was essentially not achieved.[4] Still, the primary goal of this experiment was attained. The resulting model was capable of performing a number of different versions of addition and subtraction, and it could *immediately* modify its behavior, without weight adaptation, to match one of these algorithms upon presentation of the appropriate instructions. This experiment has shown that the proposed connectionist framework is sufficient to allow complex temporal behaviors to be modulated by input advice.

Conclusion

As an initial small step towards a comprehensive model of human learning, we have proposed a connectionist framework for "learning by being told". Our approach views linguistic advice as temporal input streams to a recurrent network, and the operationalization of that advice is seen as motion in the activation space of that network. "Meaningful" instruction sequences are internally represented by stable articulated attractors in that space, and these attractors arise either in the process of learning the instructional language or in later Hebbian modifications resulting from the repeated instantiation of neighboring attractors. By locating instructed learning in activation space, our framework avoids the problems inherent in "weight compilation" approaches, and provides a means for integrating induction and instruction.

In this paper, we have put off an examination of attractor dynamics and have focused, instead, on establishing the ability of connectionist networks to inductively learn an instructional language. We have demonstrated successful instruction following behavior in both a combinatoric mapping task and in a domain involving the performance of systematic procedures. In both domains, networks inductively acquired

[3]Typically, all number pairs of up to three digits in length were used to provide training problems.

[4]The accuracy measurement shown in Figure 4 is a measure of correct *actions* over time. The displayed 74% generalization accuracy shows that many actions were performed correctly, but it masks the fact that systematic mistakes on the test set instruction sequence kept the network from attaining the complete correct answer for even a single subtraction problem when given this instruction sequence.

the ability to operationalize instructions, with error feedback provided only for actual behavior on the task. The requirements of the domain drove the representational structure at the plan layer. The ability of these networks to understand novel instruction sequences was also quite reasonable, especially considering their limited exposure to the instructional language, but there is still much room for improvement here. In particular, the utility of attractor network dynamics for generalization to novel advice sequences, a key feature of our model, has not yet been tested. We conjecture that the introduction of an attractor network at the plan layer will give rise to articulated attractors in activation space and that these will facilitate systematic generalization to novel sequences of instruction. Examining this claim will be the primary aim of future work.

If connectionism is to provide a unified modeling framework for human learning, it must incorporate a variety of learning strategies. Inductive generalization forms a natural foundation for such a framework, since so many learning problems may be expressed in terms of statistical induction. Instructional learning is a natural partner for induction, since it is strong where induction is weak. Learning from instruction is fast compared to inductive learning, and its resistance to negative contingencies makes it especially useful for learning difficult behaviors with delayed rewards. Perhaps most importantly, linguistic instruction is a primary means of efficiently transferring the vast collection of accumulated cultural knowledge to each individual. Learning from instruction has become a critical component of human development, so it is natural to make it a central focus of efforts in cognitive modeling.

References

Churchland, P. M. (1990). Talk presented at the Konnektionismus in Artificial Intelligence und Kognitionsforschung conference. Salzburg, Austria.

Cottrell, G. W. (1985). Parallelism in inheritance hierarchies with exceptions. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 194–202, Los Angeles.

Cottrell, G. W. (1989). *A Connectionist Approach to Word Sense Disambiguation*. Research Notes in Artificial Intelligence. Morgan Kaufmann, San Mateo.

Cottrell, G. W. and Tsung, F.-S. (1993). Learning simple arithmetic procedures. *Connection Science*, 5(1):37–58.

Davis, L. (1989). Mapping classifier systems into neural networks. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 1*, pages 49–56, San Mateo. Morgan Kaufmann.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Giles, C. L. and Omlin, C. W. (1993). Rule refinement with recurrent neural networks. In *1993 IEEE International Conference on Neural Networks*, pages 801–806, San Francisco. IEEE Neural Networks Council.

Hayes-Roth, F., Klahr, P., and Mostow, D. J. (1980). Advice-taking and knowledge refinement: An iterative view of skill acquisition. Technical report, Rand Corporation.

Jordan, M. I. (1986). Serial order: A parallel distributed processing approach. Technical report, Institute for Cognitive Science, UCSD.

Maclin, R. and Shavlik, J. W. (1994). Incorporating advice into agents that learn from reinforcements. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, pages 694–699, Menlo Park. AAAI Press.

McClelland, J. L., Rumelhart, D. E., and the PDP Research Group (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 2. The MIT Press, Cambridge.

McMillan, C., Mozer, M. C., and Smolensky, P. (1991). The connectionist scientist game: Rule extraction and refinement in a neural network. In *Proceedings of the 13th Annual Conference of the Cognitive Science Society*.

Mostow, D. J. (1983). Machine transformation of advice into a heuristic search procedure. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*, volume 1, pages 367–403. Tioga Publishing Company, Palo Alto.

Noelle, D. C. and Cottrell, G. W. (1994). Towards instructable connectionist systems. In Sun, R. and Bookman, L., editors, *Computational Architectures Integrating Neural And Symbolic Processes*. Kluwer Academic Publishers, Boston.

Plaut, D. C. and McClelland, J. L. (1993). Generalization with componential attractors: Word and nonword reading in an attractor network. In *Proceedings of the 15th Annual Conference of the Cognitive Science Society*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E., McClelland, J. L., and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. The MIT Press, Cambridge.

Rumelhart, D. E. and Norman, D. A. (1978). Accretion, tuning, and restructuring: Three modes of learning. In Cotton, J. W. and Klatzky, R., editors, *Semantic Factors in Cognition*. Erlbaum, Hillsdale.

St. John, M. F. (1992). Learning language in the service of a task. In *Proceedings of the 14th Annual Conference of the Cognitive Science Society*.

St. John, M. F. and McClelland, J. L. (1990). Learning and applying contextual constraints in sentence comprehension. *Artificial Intelligence*, 46(1–2):217–257.

Towell, G. G. and Shavlik, J. W. (1994). Refining symbolic knowledge using neural networks. In Michalski, R. S. and Tecuci, G., editors, *Machine Learning: A Multistrategy Approach*, volume 4, pages 405–429. Morgan Kaufmann, San Mateo.

Tresp, V., Hollatz, J., and Ahmad, S. (1993). Network structuring and training using rule-based knowledge. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, *Advances in Neural Information Processing Systems 5*, San Mateo. Morgan Kaufmann.