ALL INFORMATION ON THIS DATA SHEET MUST BE UNCLASSIFIED

INDEPENDENT RESEARCH AND DEVELOPMENT DATA SHEET

Form Approved, Office Management and Budget
Approval No. 0704-011
Exp. Dte: 12/31/88

Information in Fields 4, 5, 9, 10, 13, 15-20, and 21-24 is furnished by the company designated in Field 6 in confidence, with the claims that: (a) it falls within the exception under subsection (b)(4) of 5 U.S.C. 552, and (b) it is subject to 18 U.S.C. 1905. The information contained herein is furnished for the sole purpose of identifying the subject program, and the DoD shall except as required by the resolution of litigation or the direction of preemptive authority (e.g., The President, Congress, Justice Department) preclude disclosure to other than duly authorized Government Personnel. Any authorized reproduction or disclosure of the information contained herein, in whole or in part, shall include this notice.

A. ACCESSION NUMBER	B. TRANSACTION TYPE	1. TECH PLAN FY	2. REPORT DATE	3. REPORT TYPE	4. PROJECT NUMBER
	A	89	890315	T	WS

5. PROJECT TITLE	NEURAL NETWORKS AND ARTIFICIAL INTELLIGENCE

6. ORGANIZATION NAME AND ADDRESS	6B. ORGANIZATION SOURCE CODE
SRI International 333 Ravenswood Avenue, Menlo Park, California 94025	410281
7. TECHNICAL PLAN FOCAL POINT NAME	**TELEPHONE**
RIORDAN, ROBERT W	415-859-3536

8. PLAN VOL/PG NO.	9. CATEGORY	10. SUBJECT FIELDS AND GROUPS		
	B	1209		

11. PROJECT START DATE	12. COMPLETION DATE	13. PROFESSIONAL MAN YEARS		14. PROJECT SENSITIVITY
		EST THIS YEAR	CUM TO DATE	
8706	8807	--	0.4	UNCL

15A. MISSION AREAS			15B. FUNCTIONS		
5.3			8.4		

15C. TECHNOLOGIES		
1.3.3	1.2	3.0

16. TECHNICAL CONTACT NAME	TELEPHONE
DAVIES, TODD R	415-859-2128

RELATED PROJECTS	17. CURRENT YEAR			18. PREVIOUS YEARS		

19. KEYWORDS	Neural networks; Computational semantics; Design principles; Parallel processing; Knowledge representation; Probabilistic reasoning; Knowledge compilation

20. RELATED DoD DOCUMENTATION	20B. INTERESTED DoD ORGANIZATIONS
	DARPA; ONR; AFOSR; AFWAL; ARO; SDIO; RADC; HEL; DTRC

21. PROBLEMS/22. OBJECTIVES/23. APPROACH/24. PROGRESS *(Use reverse for continuation of any item)*

DTIC FORM 271 AUG 88 (SRI on-line version)

Problem

Ever since the early attempts to build "intelligent" machines in the 1950s, research has been divided into two broad types. One, following the work of Newell, Simon, and McCarthy, has emphasized the use of recursive programming languages like Lisp and Prolog, symbolic data structures, and the declarative representation of knowledge. The other, following the work of McCulloch, Pitts, Rosenblatt, and Hebb, has focused on bounded state machines, mathematical models of neural circuits, and statistical learning algorithms. These two schools of thought are sometimes called "symbolism" and "connectionism," respectively, or "knowledge-based systems" and "neural networks." They were both pursued on something of an equal basis until the mid- to late 1960s, when Minsky (who is the only one of the major founders of AI to have focused on both approaches) converted from the neural nets (NN) school to the knowledge base (KB) school, and knowledge-based systems became dominant for a time. In this decade, neural networks have made a comeback, and the debate between the two schools has resumed full steam.

Some problems appear on their face to be more appropriate applications of one approach or the other. For instance, very high level reasoning of the kind often performed by experts with lots of book knowledge (e.g., mathematicians, medical doctors) seems at present more suited to a KB approach, while low-level pattern-matching and signal-processing tasks appear better for the NN approach. But there is a large array of problems in between, such as the more knowledge-intensive perceptual tasks, natural language tasks, and what is known technically as "common sense" reasoning. These three areas are collectively being treated as a battle ground of sorts, with both KB and NN researchers claiming that their approach is the best for these tasks.

Problems in this vast middle ground share several characteristics, aside from the fact that real-time solutions to all of them are crucial for any machine that gets around in the world with something like human intelligence (e.g., an autonomous mobile robot). First, they are all tasks that our brains perform very well from an early age—so well, in fact, that we would be quite content with any machine that could perform them only as well as an average person. Second, they are all very complicated tasks requiring lots of real world knowledge—most of the knowledge we have, perhaps, and at least much of what we learn as children. Third, the problems in their full form have all been resistant to AI solutions, and solutions of subproblems tend to be unmanageable as we scale up to the full problem because of the time, space, and labor requirements of current methods.

These characteristics of perception, natural language, and common sense have led some to conclude that the KB approach to AI should be replaced with the NN approach. Indeed, neural networks do seem to possess a number of the properties AI efforts could be helped by. Specifically,

(1) They are fast classifiers. Fixed-weight networks compute an answer for a given set of inputs in somewhere between constant time (for multilayer networks without feedback) and $O(n^2)$ where n is the number of units (for Hopfield networks). So once a network is programmed, it runs quickly relative to serial algorithms. It achieves this speed through parallelism (trading space for time), and, perhaps more importantly, by a closed-world assumption that maps every input to an output within the temporal bounds of the network.

(2) They learn. Neural networks can change their weights and/or connections, which makes them self-modifying programs. This can save labor when it is easier to present a set of training examples than to program the network or write a set of rules for solving a problem.

This is especially attractive since writing down all of common sense is beginning to look like an impossible task.

(3) They are probabilistic. Neural networks learn and classify using statistical optimization criteria, so they can deal well with uncertainty. Specifically they cope well with the conflicting and incomplete information given to an AI program that make knowledge bases without uncertainty subject to disabling inconsistency and nonmonotonicity.

(4) They are global. Neural networks' high degree of connectivity and their numerical weights give them a relatively high storage capacity, e.g., $O(n^3)$ bits for Hopfield networks with real-valued weights, as opposed to uniform Boolean circuits, which have capacity O(log n). Thus they are well suited to problems high in Kolmogorov complexity, or so-called "random problems," of which natural perception problems and reasoning or constraint satisfaction problems involving lots of impinging evidence are examples. These contrast with more structured problems, in which a short algorithm (relative to the size of the inputs) or set of axioms can be used to achieve the right answer for any set of inputs.[1] It often seems that toy reasoning problems are low in program complexity and high in space-time complexity, whereas real problems are just the opposite.

(5) They are fault-tolerant. Part of the extra space and connectivity used by neural networks is informationally redundant in a way that permits graceful degradation when parts of the network malfunction or when there is noise in the inputs. This contrasts sharply with traditional computing architectures, which require the integrity of each local bit.

All of these advantages lead us to think that AI could be helped substantially by neural networks. Nonetheless, many AI researchers have remained skeptical about NNs for reasons similar to those that led to the rejection of perceptrons in the 1960s. Specifically,

(1) Slow learning. Neural network learning algorithms such as backpropagation can take quite long to converge to the minimum of an error function, often requiring a few thousand complete presentations of the training set for very simple (e.g., the two-bit XOR) patterns. Efforts to develop faster algorithms are under way, but there appear to be important limits to these speedups. The convergence rate depends on specifics of the problem being worked on.[11]

(2) Spatial combinatorics. Neural networks differ in their space requirements. Multilayer networks, for instance, minimize the time required for classification and hence do just about all the computation in space, whereas dynamic networks use time to some extent. But just about all neural networks grow quite rapidly with the size of the problem, since when we minimize the use of time, the space complexity for classifying general patterns in n inputs is exponential in n. Connectivity also grows faster than the size of the problem (e.g., as $O(n^4)$ for Hopfield networks), which creates more and more difficulty as we scale up the number of units.

(3) Local minima. Just as insufficiently expressive languages can be barriers to learning, neural nets of the kind currently being investigated can get trapped in suboptimal states in the space of activation or of weights. Much of the problem can be eliminated if we choose a judicious input-output representation, but as the problem gets larger and more complicated this can be increasingly hard to do, and it becomes apparent that neural networks fail to eliminate the need for careful thinking about the problem. Whether one falls into local minima in a particular network depends on the initial state from which optimization

proceeds, but the severity of the landscape and the possibilities for serious suboptimality are determined by the network and problem themselves. As one might imagine, some problems contain lots of potential for getting trapped, while others are not so tricky.

(4) Poor generalization. A problem that is in practical situations related to slow learning rates is that the network can fail to correctly classify new input patterns even when it has learned all of the correct classifications for the training set. This is an important shortcoming because for problems of reasonable size, it is impossible to present all of the input patterns, even for one training epoch, since the number of such patterns is exponential in the number of inputs. Thus presenting many complete runs through the training set, as the learning algorithms often require to reach an optimum (which may not even be global), is completely out of the question. The problem exists for any type of network, including multilayer and Hopfield nets, that are used as recognizers or associative memories. One approach to solving this problem is to maximize, by judicious selection of network size, the probability of correct generalizations for novel inputs given that the training set was chosen to be a good sample of all possibilities. But some error is still expected in new cases, and in addition this approach can be difficult or impossible to implement because we often do not know much about the distribution of the possible input patterns, and the criteria for correct generalizations cannot be inferred from the training set and so may be idiosyncratic.

(5) Output underdetermination. One source of optimism about neural networks for solving complex AI problems is that the inputs for such problems (e.g., vision problems) generally have a great deal of distributed informational redundancy in them, so that individual failures or errors do not matter very much. This is sometimes mistakenly taken to imply that correct learning and classification is overdetermined by the inputs in such problems. Indeed, in most of the toy examples that have been worked on thus far, overdetermination does appear to hold. But in real situations, it is usually not possible to solve a problem using input data from just one sensor or other input source, despite the high redundancy in the inputs. This underdetermination can arise for a number of reasons. For instance, the context can be important (as in speech recognition), or the input source may be a limited perceptual window on the problem (as when we have a single, monocular frame in vision), or background information whose acquisition is separated in time from the current input can be crucial to an interpretation (as in natural language anaphora). Thus there is often no single right answer for a given input pattern that will hold across different situations, and folding in all of the inputs that could bear on a problem (like all of the previous inputs, or all of the other sensors, or a huge data base of facts) as part of a single nonmodular network can lead to complete intractability and an impossibility of training for even one pattern. This is perhaps the most serious technical problem with the neural network approach currently being taken by most researchers, to the extent that we want to apply NNs to real tasks.

The technical efforts to improve neural networks have so far focused mostly on the first few problems noted above, like learning speed and network size. Yet these do not appear to be the most serious barriers to the widespread use of neural networks. Moreover, the approaches that have been taken in trying to solve them (as well as the few efforts undertaken to address local minima and generalization) have almost all been based on general properties of networks, e.g., learning algorithms that are faster in general, or size considerations that apply to all problems. It looks as though a great deal of effort aimed at solving problems in this way could provide us with algorithms and architectures as generally well-tuned as possible but still prone to serious errors to which people will be very sensitive. These

errors would come from those parts of the problems stated above (including part of the speed problem, a good deal of the size, local minima, and generalization problems, and possibly all of the underdetermination problem) which are intrinsically unsolvable using general approaches.

If we think about it, the real solutions to the general problems with NNs must vary with the specific application, involving principles for taking advantage of what we know about particular tasks to which the networks will be applied. This leaves open the question of what those principles are, but we can get a feeling for what is required by considering how a designer's knowledge about a particular task could help solve problems 1-5. In that spirit, learning speed can be improved if we start with an initial set of weights or connections that is closer to the final set, and if we select modules for learning to avoid the kind of competition that slows down learning in the XOR case, as well as to speed it up in cases where subproblems are cooperative; space can be conserved if we eliminate many unnecessary connections and nodes using our knowledge about independence and what is sufficient to determine what; local minima can be avoided if we start in a state that lies in a globally optimal basin of attraction, i.e. that is somewhat close to correct before we run the network; generalization can be improved if we select cases to illustrate rules that we know to apply in the specific classification task, and if we structure the network so that the rules will not interfere with each other and so that similar cases (as both we and the network would define "similar") will have similar classifications; and underdetermination can be avoided if we integrate sensor data with that from other knowledge sources, as in a distributed problem solving system with different modules and limited communication between them.

All of these approaches require us to use knowledge that we have about a problem in selecting network structures and initial values for parameters. In other words, they require that we do some initial programming of the network. This leads us to three additional problems with neural networks that affect our ability to program and understand (or verify) them:

(6) Lack of locality. As was mentioned earlier, it is an advantage that neural networks take a global approach to problem solving, but the representation of knowledge in NNs is also global, and this creates problems for building knowledge into them. In general, we cannot simply build links between nodes incrementally without worrying about how such a link fits into the entire problem representation. There are exceptions in cases like the traveling salesman problem, which have a great deal of regularity to their structure and hence are easy to think about as global problems. In other problems, however, links that seem to make sense to form separately lead either to violations of the required network structure (and hence often to oscillations or chaos) or to unintended flows of evidence, feedbacks, and so forth. Judea Pearl has attempted to solve this problem in his work on causal polytrees[12] but we have recently uncovered cases in which our natural knowledge about problems does not map easily onto one of these networks (see Progress). The lack of locality makes interpreting a network's parameters difficult as well.

(7) Restrictive syntax. To ensure the nice computational properties mentioned earlier and to ensure convergence in dynamic networks, those who have invented particular network architectures have placed restrictions on the types of connections allowed. For example, Hopfield networks and Boltzmann machines require symmetric weights and no connections from a unit to itself, multilayer networks must be feed-forward only, and causal polytrees must be singly-connected. Sometimes our knowledge of evidential relationships simply does not obey such restrictions, although once expressed it can usually be recast in a form that does obey them (see Progress).

(8) Lack of semantics. Neural networks contain parameters like weights, activation values, bias terms, learning rate constants, temperatures and the like, and functions like energy, which from the standpoint of representing information and knowledge are, at least initially, rather obscure. What does it mean to say that the weight between two nodes is some number, or that the node's activation is another number? These are crucial questions if we are to set the parameters by hand according to our understanding of evidential relationships, or if we are to interpret the knowledge or assumptions contained in a network whose parameter values have already been set by some process. Some work on the semantics was done before this project,[9,8] but most of the important questions remain unanswered. The primary difficulty is that, while we now have a handle on what parameters mean—in terms of other meanings ascribed to nodes to which they are attached (see Progress)—we have yet to understand nodes compositionally as responding to particular conditions in the input set, to define how meaning is distributed across parameters in a network, and to characterize the conditions under which the entire network converges to a particular minimum of its summarizing function. Hopes for verification procedures must rest partially on semantic understanding, since empirical tests on a limited set of examples can be risky in real situations.

Consideration of the general problems 1-5 led us to conclude that we need to make substantial use of our knowledge about a particular task and domain, and build it into the network's structure and initial state. Problems 6-8 suggest that it may be too difficult to do this directly. Instead it appears that we should make use of the KB approach in some way, since it is geared toward solving the programming and verification problems of neural networks. Specifically, building knowledge bases in a sentential language gives us the following advantages:

(1) It gives us a convenient way to enter what we know about the task and domain at whatever level of detail we seem to have in mind. For instance, we can say simply that proposition A supports proposition C as we would in a production system of heuristics or a truth maintenance system, without specifying some numerical probability, or we can specify exact probabilities if we want.

(2) It eases the nonlocality problem (number 6) by giving us a way of stating axioms or constraints somewhat independently, with the usual concerns about consistency of the KB.

(3) It gives us a much less restrictive, more natural syntax than the ones required for NNs, with which we can set forth the facts of the problem.

(4) It removes us from the semantic obscurity of NNs by giving us a language (chosen from the repertoire of AI knowledge representation formalisms) with a well-understood semantics.

So we appear to have established by rather broad arguments (whose technical details are much too complicated to present here) that neural networks have a lot to offer to AI, in the form of the five advantages mentioned earlier, and that, in turn, the KB approach to AI has a lot to offer as a solution to the problems that plague NNs. A natural idea would be to try to combine these two historically competing approaches to solve problems in perception, natural language, and common sense reasoning. How would we do this? Well, the advantages of the KB approach are primarily ones affecting the design phase for building a neural network that overcomes problems 1-5, whereas the advantages of the NN approach are ones that affect how much design we have to do (because of learning) and how well the problem is solved at run-time. So the obvious way to combine the approaches would be to define our knowledge in a KB first, possibly using the AI tools that have been built over the years such as theorem provers and other inference engines, and tools for entering knowledge. Then, when we felt

that we had a good theory of the task and its domain, we could convert the KB into an appropriate neural network which would embody the knowledge contained in the KB. After network learning, we could try to verify the network for correctness by looking at what knowledge it has learned, and the easiest way to do this would be to construct a KB from the NN itself. The difficulty, then, and the main problem to which this project has been addressed, is

How do we go from a KB to an NN, and back again?

In other words, how can a network be made to embody, or be interpreted as, declarative knowledge?

Objective

The primary objective of this project was to develop methods for solving the problem stated above, namely to compile or translate knowledge of the kind we might find in a KB written in some AI formalism into one or more of the standard types of NN, and to recover such knowledge from NNs by looking at their parameter values and structure. Since a good deal of skepticism has been expressed from colleagues working on the KB approach about whether the NN approach has anything to offer them, and vice versa, and since the perspective from which this project starts is one that treats both approaches as essential to AI, a secondary goal was to develop deep justifications for combining the approaches in this way in hopes of overcoming some resistance. It is hoped that the project will be a springboard for further efforts on this problem, funded by external sources and supporting the development of software and applications.

Approach

The approach taken was to divide the project into three areas.

(1) Knowledge compilation. Algorithms were developed for translating knowledge bases written in declarative, sentential AI formalisms to neural networks obeying different syntactic restrictions. Three such algorithms were developed: one for going from first order predicate logic to multilayer Boolean (or perceptron) networks, one for going from propositional or modal logic with possible inconsistencies to a nonstandard network with guaranteed stability, and one (developed more recently as part of another project) for going from defeasible, nonprobabilistic theories to Pearl's causal polytrees.

(2) Network semantics. Mathematical investigations were made of different interpretations for the activation value, weight, and bias terms in logistic function units, in terms of local node meanings. The assumptions embodied in the use of a logistic function for combining evidence locally were also investigated.

(3) Deep justifications. An effort was made to write a long, detailed analysis of the relative advantages of the NN and KB approaches for achieving efficiency in design and computation. The goal was to explicate, with more specific references to the published literature, the tradeoffs summarized in the Problem section of this report.

Progress

Knowledge Compilation

The three algorithms described below await a more complete explication in a forthcoming document.[7] All three of them have been developed fully enough to be implemented, but currently they exist only on paper and hence have not been tested. A white paper is in preparation that proposes further development, implementation, and application of these types of algorithms.[6]

The first algorithm developed was one for constructing three-layer circuits from theories expressed in predicate logic. In this algorithm, the universe of discourse is assumed to have a finite number r of referents (object constants). The language is assumed to have p_i predicates of arity i, where $i \geq 0$. The knowledge base or theory T is a finite set of sentences in this language. The algorithm has two phases: a translation phase, in which the knowledge embodied in the sentences is put into a network form, and a running phase, in which forward inference is performed. The network constructed has three layers of units, with feed-forward connections between layers. The first layer units are called p-units, the second layer a-units, and the third layer c-units.

The translation phase proceeds as follows. First we put T in conjunctive normal form (CNF). Next we create r^i p-units for each i-ary predicate appearing in T and its negation. These are all the possible ground literals. For each clause in T, we create the 2^{k-2} possible material implication statements (rules), where k is the number of literals in the clause, and then convert each rule's antecedent to CNF. We next create r^m a-units for each clause in each antecedent, where m is the number of literals in the clause, and we create r^i c-units for each consequent clause (in both the a-unit and c-unit cases avoiding duplication when such a clause already exists at that layer). The connections formed from p-units to a-units (layer 1 to layer 2) include one for each ground literal appearing in the ground clause represented by a given a-unit, and the function computed is disjunction. The a-units are connected to all the c-units for which they are antecedents, and the function computed is conjunction, i.e., the c-unit turns on if and only if all its inputs are on. Finally, the clauses naming c-units are used to generate 2^{k-2} rules each, for k literals in a clause, and new a-units and c-units are added to reflect these new rules, with this step repeated until there are no new a-units or c-units to add.

All nodes are assigned an initial value of 0 except for those whose clauses appear in T, which are assigned value 1. Equivalences between c-units and a-units, between c-units and p-units, and between a-units and p-units are searched for and listed. In the running phase, the units each compute their functions and update their values, and after each pass, the list of equivalences is checked and all clauses proven at a later layer result in values of 1 for the equivalent nodes at previous layers. The process repeats until there are no changes in values for one complete pass.

This algorithm allows parallel inference of a broad class of the consequences of T, but not all of them. For instance, it does not compute tautological consequences, in general. It does forward chaining, so to query it one would have to create or identify a unit representing the particular ground clause of interest, or inspect larger numbers of units for quantified queries. The space requirements for the translation are linear in the number of predicates and axioms of T, polynomial (with order depending on the maximum predicate arity and number of literals per clause) in r (the number of referents), and exponential in the predicate arities and number of literals per clause. So this is an algorithm that rewards low arity and short axioms, prefers a relatively small number of referents, and can tolerate fairly high numbers of axioms and predicates. Since these numerical relationships often hold, this

algorithm seems promising for many inference problems involving reasoning about a relatively small number of objects, but other algorithms could be developed with different cost priorities. The temporal requirements of this algorithm have not been worked out yet, but they don't appear to be exponential in any of the quantities previously mentioned.

The second algorithm developed translates statements in a propositional logic, which may be inconsistent, as constraints in a nonstandard network. Each statement in the KB is viewed as expressing one or more binary logical relations such as AND or IFF. There are sixteen possible binary logical relations, and each translates into a piece of circuitry with logistic activation or perceptron units ranging in activation value from 0 to 1. Some difficulty in the translation is caused by the fact that a 0 activation value has no effect on a perceptron or logistic unit, but according to the logical theory the 0 is supposed to contain information about the other unit. Hence, we use a scheme for inverting activation values through other adjacent units, so that some relations require four units to translate them. The network is built incrementally from these constraints, and since it obeys the even-numbered cycle criterion of Hirsch (1987) it is guaranteed to stabilize to a local minimum of its summarizing function, which maximizes the extent to which the constraints are obeyed in the inference.

The third algorithm translates from default rules with exceptions to causal polytrees. Defeasible rules are rules of the form "The presence of A supports the presence of B, but the presence of C defeats this support relation." The canonical example in AI is "Typically birds fly," but "Ostriches don't fly." Causal polytrees are feed-forward networks in which directed links represent conditional probabilities, and which are constrained to be singly connected, so that no more than one path exists between any two nodes.[12] The translation proceeds in five steps:

(1) Default rules (as well as rigid rules and facts) are translated individually into constraints on a probability model. For instance, the rule that A supports B would give rise to a constraint that $P(B \mid A) > P(B)$.

(2) The propositional variables appearing in the constraints are assigned an ordering according to what we could infer about their causal structure from the orderings of support relations. This is a necessary step in defining a causal polytree and also an important one since the right order can mean the difference between a manageable and an unwieldy network.

(3) A probability model is found satisfying the constraints, heuristically geared toward the maximum entropy solution.

(4) The probability model, together with the ordering, is used to construct what Pearl calls a "Bayesian network" and according to his algorithm.

(5) One or more of Pearl's techniques for coping with loops in the underlying graph, and with hyperconnections in the Bayesian network, are employed to make a causal polytree. The resulting network is guaranteed to compute in time proportional to the longest path in the network.[12]

The speedup relative to default inference engines comes from a combination of trading space for time and exploiting assumed probabilistic independence relationships introduced in the construction of the probability model. Nonetheless all the information that strictly follows from the default rules is retained in any feasible probability model. This translation scheme allows evidential relationships to be specified in the defeasible rules that are not directly translatable into a polytree. For instance, if a block is being pushed by forces that oppose each other, we naturally say that the left moving force supports left movement of the block and similarly for the right moving force, and additionally that right movement blocks

the support we would ordinarily get for the block moving left from knowledge of a left moving force, and vice versa. A local translation of these types of statements into links in a Bayesian network results in violation of the syntax for Bayesian networks, so a translation can proceed only on the basis of a global consideration of all the evidence bearing on a proposition. This is much better left to a translation program than dealt with by the person constructing a network, since it is possible to come up with a translation with the right properties. This algorithm has been developed and is currently being refined as part of a project sponsored by the ONR.

These algorithms trace the development of thinking in this project about translations from KB to NN systems. Some previous work has been done in this area by Dana Ballard (1986), who constructed networks for logical query answering, but we are not aware of other work that has focused on translations of theories incorporating uncertainty or inconsistency.

Network Semantics

Translation to more complicated NN systems with semantics less straightforward than Pearl's or than Boolean networks requires us to develop a theory of the meanings of their parameters and states. This is also essential if we are to recover KBs from networks that learn. The results of this investigation appeared in the First Annual Meeting of the International Neural Network Society.[4]

A common choice for the activation function that is computed at each node in a model neural network is one that is given by the logistic equation

$$\alpha_j = \frac{1}{1 + \exp[-(\Sigma_{i \in I_j} w_j x_i + \theta_j)/T]}$$

in which α_j is the activation value (in [0,1] for the jth unit (node), I_j is the set $\{i_1, \ldots, i_n\}$ of inputs to j, w_{ij} is the weight (in $\mathbb{R}$) connecting an input i to unit j, x_i is the state (usually binary or in [0,1]) of i, θ_j is the threshold or bias (in $\mathbb{R}$) for j, and T is the temperature or gain parameter (in $\mathbb{R}^+$, assumed herein to be 1). The activation value α_j is usually thought of as representing either the computed probability of the proposition or event ascribed to j as its content, or the probability that j, as a binary unit, will turn on (i.e., that its state x_j will be 1 rather 0). The weight and threshold terms are often not given a clear probabilistic interpretation by those who use them in models, but intuitively a weight w_{ij} expresses the strength of evidence in favor of (or against) the proposition represented by x_j that is provided by $x_i = 1$, and the threshold θ_j expresses the (positive or negative) tendency for x_j to go to 1 in the absence of any of the inputs being on or "true." If α_j is identified with the probability of an event or proposition B_j (which could be identical to the event $x_j = 1$) given the input state vector (or vector of truth values) $\mathbf{x} = (x_{i_1}, \ldots, x_{i_n})$ then the logit transformation of the probability $p(B_j \mid \mathbf{x})$ defines an alternative form of the logistic equation (with $T = 1$), namely,

$$\text{logit}[p(B_j \mid \mathbf{x})] = \ln \frac{p(B_j \mid \mathbf{x})}{1 - p(B_j \mid \mathbf{x})} = \sum_{i \in I_j} w_{ij} x_i + \theta_j \quad .$$

If we are to interpret networks with logistic units in probabilistic terms, or if we wish to know how to translate probability statements into parameter values for such networks, then two questions that arise are:

(1) How can activation values, weights, and thresholds be expressed as functions of probabilities?

(2) What assumptions about probabilities are embodied in the logistic model?

Little if any attention appears to have been paid to these problems by statisticians, probably because logistic regression models are used primarily for prediction and the exact analysis of coefficients is not of great interest.[3] Hinton & Sejnowski (1983) discuss possible definitions for the weight and threshold parameters for the case of one input state x_i, under the assumptions that α_j represents $p(B_j)$, and that $x_i = 1$ represents the presence of input evidence A_i, and $x_i = 0$ represents its absence $\neg A_i$. The definitions on which they settle, for the restricted case of $|I_j| = 1$, amount to the following:

$$\theta_j = \text{logit}[p(B_j)] + \sum_{i \in I_j} \ln \frac{p(\neg A_i \mid B_j)}{p(\neg A_i \mid \neg B_j)}, \quad w_{ij} = \ln \frac{p(A_i \mid B_j)}{p(A_i \mid \neg B_j)} - \ln \frac{p(\neg A_i \mid B_j)}{p(\neg A_i \mid \neg B_j)} \quad .$$

A similar semantics has been proposed by Geffner & Pearl (1987). A very useful feature of these definitions, as Hinton and Sejnowski point out, is that the function for weights is symmetric in A_i and B_j, implying that $w_{ij} = w_{ji}$. The semantics therefore provides a justification for the symmetry constraints often imposed by network architectures.

The definitions given above may be difficult to understand intuitively. However, if we extend the analysis of Hinton and Sejnowski to the case of multiple inputs, then an equivalent form can be given for these equations that defines θ_j as a function of one probability value, and requires just one additional probability value to be specified for each w_{ij}, viz,

$$\theta_j = \text{logit}[p(B_j \mid \neg A_{i_1}, \ldots, \neg A_{i_n})], \quad w_{ij} = \text{logit}[p(B_j \mid A_i \text{ and } \neg A_k \text{ for all } k \text{ s.t. } k \in I_j,\ k \neq i)] - \theta_j \quad .$$

The parameters can thus be defined as functions of the probability of a hypothesis given evidence, rather than the reverse, and this may be easier to think about. Because the logit mapping is one-to-one we can recover these probabilities from the parameter values. Extension to multiple inputs also allows us to prove the following: For θ_j and w_{ij} defined as above, $p(B_j \mid \mathbf{x})$ can be computed as a logistic function of $\Sigma_{i \in I_j} w_{ij} x_i + \theta_j$ for each input vector $\mathbf{x}$ iff

$$p(\mathbf{x} \mid B_j) = \prod_{i \in I_j} p(x_i \mid B_j) \text{ and } p(\mathbf{x} \mid \neg B_j) = \prod_{i \in I_j} p(x_i \mid \neg B_j) \quad .$$

In words, assumptions of conditional independence among the components of $\mathbf{x}$ for both B_j and $\neg B_j$ are necessary and sufficient conditions for exact representation of the probability model in a logistic function. From the definitions given above for w_{ij} and θ_j one can derive a system of linear equations that significantly overdetermine the parameters and probabilities. When the conditional independence

assumptions are violated, error-minimization (e.g., orthogonalization, least-squares) can be applied to the system to determine an optimal interpretation or assignment for the parameters.

Deep Justifications

Some effort was expended in writing a lengthy paper detailing how NN and KB approaches achieve or fail to achieve various types of efficiency in design and computation. It was intended as a resource guide for those interested in what has been discovered about the complex tradeoffs in AI, but it was not completed. The effort had to be abandoned after about 70 pages (or approximately 2/3 of the report) had been written[5] because of obligations the investigator had to another project, funded by the ONR. It is hoped that this paper will be completed at some point in the coming year.

References

1. Abu-Mostafa, Y.S., 1986: "Neural Networks for Computing?" In *Neural Networks for Computing*, J.S. Denker, ed. (American Institute of Physics, New York, New York), pp. 1-6, Snowbird, UT 1986.

2. Ballard, D., 1986: "Parallel Logical Inference and Energy Minimization." *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, Pennsylvania (August 11-15), pp. 203-208.

3. Cox, D.R., 1970: *The Analysis of Binary Data*. (Methuen & Co. Ltd., London).

4. Davies, T.R. 1988: "Some Notes on the Probabilistic Semantics of Logistic Function Parameters in Neural Networks," *Neural Networks*, Vol. 1, Supplement 1, Abstracts of the First Annual INNS Meeting, Boston, Massachusetts (September 6-10), p. 88.

5. Davies, T.R. 1988: "Operative Principles in Artificial Intelligence and Neural Networks: Toward a Framework for Evaluating Efficiency Tradeoffs," Unfinished manuscript (August 16).

6. Davies, T.R., 1989: "Compiling Large Knowledge Bases into Fast Networks." White Paper, Artificial Intelligence Center, SRI International, Menlo Park, California (in preparation).

7. Davies, T.R., 1989: "Translating from Knowledge Bases to Parallel Networks," Technical Note, Artificial Intelligence Center, SRI International, Menlo Park, California (forthcoming).

8. Geffner, H. and Pearl, J. 1987: "On the Probabilistic Semantics of Connectionist Networks." *IEEE First International Conference on Neural Networks*, San Diego, California (June 21-24), pp. II-187-195.

9. Hinton, G.E. and Sejnowski, T.J., 1983: "Optimal Perceptual Inference." *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* Washington, D.C. (June 19-23), pp. 448-453.

10. Hirsch, M., 1987: "Convergence in Neural Nets." *IEEE First International Conference on Neural Networks,* San Diego, California (June 21-24), pp. II-115-125.

11. Minsky, M.L. and Papert, S.A., 1988: *Perceptrons* (Expanded Edition), (The MIT Press, Cambridge, Massachusetts).

12. Pearl, J. 1988: *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference,* (Morgan Kaufmann, Los Altos, California).