INDUCING (NEW) RULES IS DIFFERENT FROM ADJUSTING (OLD) PARAMETERS.

K.Prazdny

Artificial Intelligence Center, FMC Corporation
P.O. Box 580, Santa Clara, CA 95052
prazdny@ai.cel.fmc.com

INTRODUCTION.

Linear (auto)associators capture the structure inherent in a set of patterns as long as the ensemble of example patterns adheres to the linear predictability constraint. Conjunctions and exclusive-disjunctions, however, require that the compound features have a different associative strength than the individual input in isolation. In general, it is hard to specify in advance what order of conjunctions is required to capture the data dependencies. The major motivation behind the development of learning protocols for networks with hidden units (LeCun, 1985, 1986; Hinton, Sejnowski & Ackley, 1984; Rumelhart, Hinton & Williams, 1985) is to discover the relationships in the input automatically.

There is a close relationship between this work and modelling the phenomena of classical conditioning. There, the animal model tries to predict single event (US) while the (auto)associator must, in effect, develop such prediction for each individual vector element. In both cases, the goal is a method of determining which features are predictive of others and distinguishing useful cues from context and background noise. Several contemporary animal learning theories handle conjunctions and disjunctions by assuming that the co-occurence of two stimuli results in some new (external) "resonant" property being signalled by the perceptual system. Other models use "internal resonant" features: new compound features (boolean combinations of existing ones) are introduced into the representation as the result of system's prediction failure (Quinqueton & Sallantin, 1983; Schlimmer & Granger, 1986). Similarly, networks with hidden units can implement an arbitrary input/output mapping if they have the right connections and large enough set of hidden units (Minsky & Papert, 1969). For example, to solve the XOR problem, one can add a unit that detects the conjunction of the two inputs. This amounts essentially to enlarging the dimensionality of the input: from the point of view of the ouput unit the hidden unit is treated as another input unit. In this sense, networks with hidden units can be said to be discovering the "resonant" properties of stimulation (feature combinations predictive of the desired outcome) in a way similar to the classical conditioning models.[1] The number of such potential resonant properties in the general case of a non-linear autoassociator, where the value of a feature is predictable, in general, only from a non-linear combination of values of other set of features, grows exponentially with the vector length, n. Potentially, one needs 2^n-(n+1) "resonant" features for the prediction of a single element (i.e. the domain is the set of all subsets). Hidden units and recurrent connections are, in themselves, of little help, however. One has to find a useful way to use them. To illustrate, McClelland & Rumelhart (1986) attempted to implement a non-linear autoassociator for the "one-same-one" problem using hidden units and recurrent connections. To achieve the required efect they had to train the network with all possible completion patterns. That is, for each pattern to be learned (e.g. 111) they trained the network to associate all of the possible incomplete patterns (11?, 1?1, ?11) with the complete (111) pattern (McClelland & Rumelhart, 1986, p.211). This is, of course, "cheating": they have replaced an autoassociation task with an association task. In general, the enumeration of all possible completions is impractical; a daunting task for even moderately long inputs.

WHAT CAN THE NETWORKS LEARN?

One rather serious disadvantage of the PDP networks with fixed length input[2] is that the information they acquire in the course of their "education" is not cumulative. That is, having been taught in one situation is of absolutely no help in learning in a different but similar situation. There is no possibility of transfering the accumulated knowledge to a new but similar situation, as opposed to a new but similar input. Even the within-situation generalization is apparently difficult to achieve. Confronted with new patterns, the steepest descent weight updating procedures preferentially changes existing, already useful representational features because the error gradient (and thus the weight change) is directly proportional to the current weight magnitude (Sutton, 1986). This protocol thus destroys what has been learned previously. I will illustrate some of these points on the parity problem.

Parity can be defined in two ways for inputs in {0, 1}: (i) "the sum of inputs is odd", or recursively as (ii) parity(0)=value(0) and parity(n+1)=XOR[parity(n),value(n+1)] where value(n) is the value of the n^{th} input element, and parity(n) is in {0, 1}. Suppose we have a network with with m input and hidden units, and with a single output unit and teach the parity to the first n inputs (n<m). Does this "education" help in acquiring the parity problem of size n+1 (or n+k), e.g. does the network converge sooner? The answer is that previous experiences with the parity problem on n inputs is of absolutely no help in learning the parity problem of larger (or smaller) size. In fact, in none of our experiments did the network with weights obtained from the previous teaching run even converge (all unused inputs were kept constant at the resting, 0.5, level). This phenomenon is understandable in the view of learning as creating and traversing the "energy landscape" in the weight space: the energy minima are simply at different places! The networks with hidden units do not learn new *concepts*, they are "merely" discovering data dependencies (input/output contingencies) in a particular situation. Similarly, the classical conditioning models and algorithms for learning logical formulas learn "boolean combinations" of antecendent conditions, not the underlying concept.[3]

Suppose that we teach a network the parity problem with 2, 3, 4, ... inputs. What kind of computational mechanism would be required to abstract the concept of parity from such a teaching sequence?[4] An obvious answer may be that one needs another network that sees all the weights and biases, and how they are being modified from one *instance* of the concept to another, and that changes its internal structure so that when a given situation arises it programs the weight distribution of the network appropriately. The problem with this approach is that there is no guarantee of any lawfull relationship between the weight distributions accross the various instances of a given concept. That is, the set of weights for the (n+1) problem may not be predictable from the set of weights for the (n-k) problem. With large enough number of units and connections (weights) relative to the minimum necessary for the given task, i.e. with large degree of freedom, there is a large number of ways in which a given set of input/output specification can be mapped into the weight space, i.e. the mapping is one-to-many. In other words, there is no guarantee that the regularities obvious in one domain will appear in the weight space. In addition, one cannot, in general, know in advance the limit on the number of inputs.[5] A concept can describe an infinite number of situations: it has a generative quality of a rule not captured by the stimulus-response associations of a PDP network.[6]

CONCLUSION.

There are two distinct (not neccessarily related) problems: (I) how do you create/develop a new concept (e.g. parity) as opposed to (II) how do you solve a particular problem (e.g. parity with n inputs). An intelligent agent should (probably) be required to posses the ability to solve (II) using (I). Genuine concept learning, if it can be implemented using connectionist architectures at all, has to be done by a mechanism that monitors the adapting network not limited or constrained by the number of inputs. This evaluative mechanism must be capable of gathering information accross different situation or instances of the same concept. It is this mechanism that would learn new concepts as opposed to adapting to the immediacy of the incoming stimulation. It is doubtfull that this can be done in a connectionist system (of either the weight adjustment or signature table variety) where all knowledge is constrained by a finite structure and there is no distinction between information and control. The ability to construct and manipulate symbolic structures and procedures seems necessary.

REFERENCES.

Chaitin G.J., Randomness and mathematical proof, *Scientific American*, 232, 5, 1975 (May)

LeCun Y., A learning scheme for assymetric threshold networks, *Proceedings Cognitiva-85*, 1985

McClelland J. Rumelhart D., A distributed model of human learning and memory, *Parallel distributed processing*, D. Rumelhart & J. McClelland (eds), 170-215, 1986

Quinqueton J. Sallantin J., Algorithms for learning logical formulas, *Proceedings IJCAI*, 476-478, 1983

Rumelhart D.E. Hinton G. Williams R.J., *Learning internal representations by error propagation*, Tech.Rep. ICS-8506, Institute for Cognitive Science, UCSD, La Jolla

Schlimmer J.C. Granger R.H., Simultaneous configural classical conditioning, *Proceedings Conference Cognitive Society*, 141-153, 1986

Sutton R., Two problems with backpropagation, *Proceedings Conference Cognitive Society*, 823-831, 1986

Notes

[1]Context-sensitive encoding where a code for an element depends on other elements corresponds to creation of compound features or "resonant" properties that are imposed on a system, usually a linear (auto)associator, from the outside. Context-sensitive encoding is performed in an attempt to achieve linear separability that enables the use of a linear system (which has nice and predictable "generalization" properties).

[2]This characteristic shows the close similarity of the PDP networks to the classical pattern recognition work. Input vectors are of fixed length, i.e. the information is coded by position, each position is a different feature, a different (orthogonal) dimension of a (finite dimensional) vector space. Thus, there can be no shift invariancy: 0100 is orthogonal to 0010. Suppose that the input is a histogram. Then <0 16 0 0 0> is more similar to <0 0 16 0 0> than to <0 0 0 0 16> because nearby vector elements encode similar values. In the vector space formalism where the distance is equivalent to the inner product all three are orthogonal (i.e. dissimilar).

[3]How would models relying on explicit rule generation handle the parity concept? They would end up with the exhaustive enumeration of the permissible combinations. E.g., for the 2 input parity (XOR) problem, the system will end up with a compound feature "((1 and 0) or (0 and 1))", and for the 3 input situation it would produce "((1 and 0 and 0) or (0 and 1 and 0) or .....)" or perhaps, given previous experience with the XOR problem, "(((1 and 0) and 0) or ((0 and 1) and 0) .....)" It is relatively easy to envisage a concept formation mechanism operating on such rules that would develop the parity concept defined by (ii).

[4]It may be interesting to know if and how well humans and e.g. the dogs can do this, i.e., if they can develop a genuine parity concept as opposed to a set of responses to a set of situations. Does teaching parity on e.g. 2 and 3 inputs produce generalization, i.e. correct response on e.g. 4 inputs?

[5]There are other, related problems that do not require variable length inputs: parametrized concepts. Consider, for example, the concept of negation (Rumelhart & McClelland, 1986) on a vector with fixed length, N. The concept has, in its simplest form, only one parameter: the position of the negation bit. Is there a connectionist mechanism that can, after it learns concept examples parametrized by the first n inputs, generalize to the remaining (untaught) N-n parameters?

[6]The notion of a concept is probably intimately related to the notion of a program: two strings of different length can be described by the same program. Unfortunately, the theory of program-size complexity (e.g. Chaitin, 1975) while relevant here is not constructive and cannot offer any guidance in the construction of a program from examples.