THE ELECTRIC CHICKEN - A LEARNING ROBOT

W.W. Armstrong, G. Phung,
C. Schneegans, P. Vaillancourt
Département d'informatique
Université de Montréal

1. Introduction.

In this article we report on a type of adaptive digital system which acts somewhat like a brain, and on the rôle that conversational computer graphics is destined to play in our research. First, let us make it clear that we are not simulating cerebral activity on either the neuronal or global level. Rather, we take networks composed of elements far simpler than neurons from the point of view of stimulus-response behavior, and we try to find a simple adaptive structure for them which will lead to learning when reinforcement and weakening are applied as in a Skinner box.

There have been attempts to do something like this before, using elements such as the Perceptron [1], the Adaline [2], the Neurotron [3], and the SLAM [4]. The first two employ the linear-threshold function commonly thought to model neuronal behavior, and the latter two use a universal boolean function approach similar to ours. Unfortunately, none of the elements named has been truly satisfactory for training in large multilayer networks.

Recent progress has been made in training elements called ABLEs (adaptive boolean logic elements) [5,6] in multilayer tree networks. These networks have also shown some ability to "generalize", in the sense that the same response learned to a training pattern is given for test patterns "similar" to it. This has led us to try ABLEs, each combined with a unit delay element, in networks with feedback loops. The tasks to be learned by the networks are all of the form : the output at a certain instant of time is to be a prescribed function of the L immediately preceding inputs. The upper limit for L in our early tests has been 3, although the principles used should be valid for larger L when better parameters for network and element structure are known. In determining these parameters, computer graphics will play a central rôle.

Those researchers who study the animal brain seem to have made very little progress so far in finding out how or where learning takes place in it [7], and it is hoped that this study of adaptive networks, even though not based on any biological findings, when completed, may provide some fresh insights into the behavior of adaptive networks in general.

2. Logic-delay networks as automata.

The objects simulated are finite networks of Mealy automata [8], each of which acts (if we neglect the state-transition structure for the moment) as a two-input logic gate whose output is subjected to one clock-cycle delay. The network elements have fixed connections with each other and with network input and output terminals.

Let there be K network elements denoted by superscripts 1 to K, and suppose that the superscripts K+1 to K+M represent the logical input terminals

of the network. If $y^j(t)$ denotes the logical signal (0 or 1) at element output or network input number j at sampling time t, then the action of element k can be described as follows. Its input at t is the pair $y^{k'}(t)$ $y^{k''}(t)$, where k' and k'' are the terminals to which the two logical input leads of element k are connected. Internally, it has four time-varying logical values $b^k_{00}(t)$, $b^k_{01}(t)$, $b^k_{10}(t)$, $b^k_{11}(t)$ which characterize its behavior at any instant. Its output $y^k(t+1)$ at t+1 is just $b^k_{ij}(t)$, where ij is the input pair at t, i.e. $i = y^{k'}(t)$, $j = y^{k''}(t)$. Determination of $y^k(t+1)$ would be done simultaneously for $k = 1,\ldots,K$ in a hardware realization of a network, but must be done serially in simulation. This constitutes the information processing of the network during one cycle.

Randomly chosen logical signals are presented to network inputs $y^{K+1}(t),\ldots, y^{K+M}(t)$ at each multiple of T and are held constant for a total of T cycles of the network : $y^{K+m}(nT+s) = y^{K+m}(nT)$ for $m = 1,\ldots, M$; $s = 1,\ldots, T-1$. At $t = (n+1)T$ the output of the network $y^K((n+1)T)$ (a single 0 or 1 for the sake of simplicity) is compared to the desired value, which is a prescribed function of $y^{K+m}(t)$ for $m = 1,\ldots,M$; $t = nT, (n-1)T,\ldots,(n-L+1)T$. (We shall see that the network must cycle several (T) times faster than inputs are presented to compensate for the delay in each element's output). If the network response is the desired one, a "reinforcement" signal is sent to all elements, and otherwise a "weakening" signal is sent.

The network uses these signals to carry out a search for a satisfactory configuration of the $b^k_{ij}(t)$ according to a heuristic embodied in the state-transition structure of the elements. When no further weakenings occur, the $b^k_{ij}(t)$ cease to vary, and we say that the desired behavior has been "learned". We shall study a particular adaptation procedure in more detail in section 3.

The automata theorist may regard learned behavior as a solution to a synthesis problem, and indeed, a very general one, for any Mealy automaton with boolean inputs and outputs can be realized by an appropriately connected network of logic gates and unit delays [8]. Combining two-input logic and delay into a single building-block leads to some restriction of the class of automata realizable, however to restore complete generality except for a delay in the output, it is merely necessary to allow the network elements to cycle several times for each cycle of the automaton to be synthesized and to sample and hold inputs and outputs of the network accordingly. We leave the details of the proof to the interested reader.

It is partly for the sake of uniformity that we combine logic and delay into one element, for then, not only does one type of element suffice for a very general class of tasks, but in forming a network, any connection of an element logical input to an element logical output or to a network input terminal is legitimate even if it creates a feedback loop. Another reason, based entirely on conjecture, is that sequences of elements might serve as a form of temporary delay-line memory. In any case, most of what we say could be extended to networks of logic elements where at least one delay element is present in each feedback loop.

Since we suppose that network connections are fixed in advance, and that all we can do to obtain a solution is to vary the boolean functions of

the gates, the problem is not solvable by classical synthesis procedures for Mealy automata. Furthermore, if the given network is too small or unfortunately connected, there may be no solution ; and at the opposite extreme, the solution learned may be realizable with far fewer elements than are provided in the network.

It is easy to see that one could construct a network to perform the given tasks by simply entering the M input signals into M shift-registers of length L at each cycle of the "Skinner box" and adjusting the output corresponding to each minterm of the resulting M x L array using the reinforcement and weakening signals. This method would only work if the product ML is small (<25) because of the number of minterms (2^{ML}) required, and would not make any use of similarity between sequences of inputs. Fortunately, in cases where learning is required for many input terminals and a long past history, it will usually be the case that "similar" stimuli will be required to evoke identical responses. Because of the results in [6] concerning generalization, our choice of logic-delay elements as components seems quite natural and promising.

3. Heuristics for learning.

What must be devised is some way of telling when the value of each $b^k_{ij}(t)$ should be inverted in the hope of improving network performance. When the value of $b^k_{ij}(t)$ is used to determine the output $y^k(t+1)$, this may contribute strongly to an immediately following reinforcement or weakening, and probably less strongly to future reinforcements and weakenings. At least, the assumption that such is the case can be used to derive a measure of satisfaction with the value of b^k_{ij} at any instant t.

To implement this measure in a way that attempts to be economical in terms of hardware, we include four "satisfaction" counters in each element with values $c^k_{ij}(t)$ in the set $\{0,1,\ldots, MAX\}$, and four "input" counters with values $d^k_{ij}(t)$ in the set $\{0,\ldots,2T\}$. The d's are initialized to 0 at the start of training, and $d^k_{ij}(t)$ is incremented by one unit each time element k receives input ij. The $b^k_{ij}(0)$ are initialized randomly to 0 or 1 and the $c^k_{ij}(0)$ are initialized to MAX. Every T cycles of the network a reinforcement or weakening occurs, and this is used to update the values of the b's and c's as follows :

If $y^K(nT)$ is the desired response, then there is a reinforcement signal which sets

$$c^k_{ij}(nT) = \min \{c^k_{ij}(nT-1) + d^k_{ij}(nT-1),\ MAX\}$$

for $k = 1,\ldots, K$; $i = 0,1$; $j = 0,1$. The b's are left unchanged.

Similarly, a weakening signal causes

$$c^k_{ij}(nT) = \begin{cases} c^k_{ij}(nT-1) - d^k_{ij}(nT-1) & \text{if this difference is non-negative,} \\ MAX & \text{otherwise (i.e. reset),} \end{cases}$$

$$b^k_{ij}(nT) = \begin{cases} b^k_{ij}(nT-1) & \text{if the above difference is non-negative,} \\ \overline{b^k_{ij}(nT-1)} & \text{otherwise (i.e. invert).} \end{cases}$$

Before the next T cycles are begun, the input counters are reset:

$$d^k_{ij}(nT) = \begin{cases} \lfloor d^k_{ij}(nT-1)/2 \rfloor & \text{if the above difference is non-negative,} \\ 0 & \text{otherwise.} \end{cases}$$

Doing this instead of resetting all d's to zero allows the actions of element k under input ij since the last time b^k_{ij} was changed to be held responsible in part for future reinforcements and weakenings.

Another adjustment which may be performed with a certain probability in any network cycle is a universal unit down-count of all "satisfaction" counters. This is a trick which worked well for ABLE networks helping to push learning to completion. We shall omit the details here.

Despite the seeming complexity of this state-transition structure for our logic-delay elements, it is easily programmed in a variety of machine languages (for speed), and might be realizable in hardware by a single module per element.

4. Simulation using computer graphics.

Since the "fixed connections" synthesis problem involves boolean logic in an unfamiliar and complex way, it would seem almost impossible to solve it in practice except by heuristic search. Our hope is that improved heuristic procedures may be devised using insight gained by observing visually, and interacting with, a simulation of network operation. We are encouraged by results on networks without delays or feedback loops, where a heureistic has been devised improving on trial-and-error by a factor of over one million for networks with five input terminals [5].

A set of programs has been written for the CDC 1700 DIGIGRAPHICS system at the Université de Montréal whereby an adaptive network is made to serve as the "brain" of an "electric chicken". The effigy of a fowl appears on the CRT display, intently concentrating on an array of three lights within a "Skinner box". The lights are made to light up at random at times $0, T, 2T, \ldots$ and rules can be prescribed for when the chicken should touch a button below the lights. If he touches after any one of a list of prescribed sequences of length three of patterns of illumination he gets a reinforcement, which shows up as a pellet of food which he eats. If he fails to touch the button in these circumstances, he gets a weakening signal appearing as an electric shock under his feet. Weakening also occurs if he touches the button when he shouldn't.

Thus the adaptive network is encouraged to become a (definite) automaton whose output {touch, don't touch} depends on the last $L = 3$ input symbols from the alphabet of eight illumination patterns.

The percentage of successful responses appears as a graph as learning progresses, and certain parameters can be varied interactively. So far,

we have not written routines to permit observation of the operation of the network - the signals, the values of b's, c's, and d's, and the statistics of various changes. We must do considerable experimentation before we can answer such questions as :

1) What is the best way to represent visually the significance of signals in these networks ?

2) What are the different paths between input and output terminals, and how does each affect behavior ?

3) What fraction of the element output signals tend to become constant when the lights are held constant for several network cycles, and to what degree is this usable as a mechanism for short-term memory ?

4) Is there a systematic connection structure which improves performance over the essentially random connections we are now using ?

At the present time (April 1971) we have been able to obtain complete learning of simple tasks with L = 2. For example : "if the bottom light is illuminated and then the top one lights up, touch the button". Here the illumination of other lights does not matter and the sequences requiring touching may overlap. A task involving only the most recent input, e.g. "if the middle light is illuminated, touch the button", is solved very rapidly. Hence it is probably safe to assert that the use of logic-delay elements is sound in principle. The near future should shed much more light on the potentialities of networks of these elements.

The program for the electric chicken employs a monitor written by Claude Schneegans to control the "Skinner box". It is designed to be called by a user program written in FORTRAN or ASSEMBLER, or both, which describes the operation of the chicken's "brain". In this way it should be possible for researchers who are more interested in biological brain modelling to try out their theories of learning mechanisms and compare their results to ours.

5. Summary.

A program has been developed to test and display visually the progress of learning in certain adaptive systems under a regime analogous to operant conditioning in a Skinner box. A particular type of adaptive network element is proposed which has proved successful in early trials. It is hoped that conversational computer graphics can provide the insight necessary to determine optimal network parameters for this type of element. This type of simulation may also be useful in studies of biological learning networks.

References.

[1] M. Minsky and S. Papert, Perceptrons, an Introduction to Computational Geometry, MIT Press, Cambridge, Mass. 1969.

[2] B. Widrow and J.B. Angell, Reliable, Trainable Networks for Computing and Control, Aerospace Engineering 21, 1962, pp. 78-123.

[3] R.J. Lee et al, Theory of Probability State Variable Systems, Vol.1 Clearinghouse for Federal Scientific and Technical Information, U.S.A. AD 427 872, 1963.

[4] J. Aleksander, Some Psychological Properties of Digital Learning Nets, Int. J. Man-Machine Studies 2, 1970, pp. 189-212.

[5] W.W. Armstrong, Training Strategies for Networks of Adaptive Boolean Logic Elements, Publ. #20 (revised March 1971), Dépt. d'informatique, Université de Montréal.

[6] W.W. Armstrong, A Practical Solution to the Multilayer Learning Problem, Publ., Dépt. d'informatique, Université de Montréal

[7] H. Blum, A New Model of Global Brain Function, Perspectives in Biol. and Med. 10 no.3, pp. 381-408.

[8] T.L. Booth, Sequential Machines and Automata Theory, John Wiley and Sons 1967.