Understanding biological computation: Reliable learning and recognition

(self-repair/learning and selectivity/computational dynamics)

T. HOGG AND B. A. HUBERMAN

Xerox Palo Alto Research Center, Palo Alto, CA 94304

Communicated by Murray Gell-Mann, June 29, 1984

ABSTRACT We experimentally examine the consequences of the hypothesis that the brain operates reliably, even though individual components may intermittently fail, by computing with dynamical attractors. Specifically, such a mechanism exploits dynamic collective behavior of a system with attractive fixed points in its phase space. In contrast to the usual methods of reliable computation involving a large number of redundant elements, this technique of self-repair only requires collective computation with a few units, and it is amenable to quantitative investigation. Experiments on parallel computing arrays show that this mechanism leads naturally to rapid self-repair, adaptation to the environment, recognition and discrimination of fuzzy inputs, and conditional learning, properties that are commonly associated with biological computation.

Information processing in biological systems possesses a number of intriguing fundamental characteristics that are difficult to understand in terms of sequential computing mechanisms. One of these is fault-tolerant behavior with respect to both internal failures and input data errors, allowing the system to operate with distorted, or fuzzy, inputs. Further examples are associative memory, in which one input can elicit a whole repertoire of responses, and conditional learning, in which the ability to learn something is facilitated by previous knowledge of something similar to it. Finally, there is the ability to adapt to changes in the environment, or plasticity, with the associated mechanism of selection out of degenerate initial configurations. While higher-level cognitive processes, such as reasoning and language understanding, also pose interesting questions (1, 2), we will focus our attention on what we perceive are the basic building blocks of biological computation.

In this paper, we present the hypothesis that the brain operates reliably, even though individual components may intermittently fail, by computing with dynamic attractors. Specifically, such a mechanism exploits collective behavior of a dynamical system with attractive fixed points in its phase space. In contrast to the usual methods of reliable computation involving a large number of redundant elements, introduced by Brueckner, Gell-Mann, and von Neumann (3–5) many years ago, this technique of self-repair only requires collective computation with a few units, and it is amenable to quantitative investigation.

To examine the consequences of this hypothesis, we first present an array of parallel processors, which computes in such a fashion and exhibits the features mentioned above (6). We also outline quantitative techniques that provide an explicit method of measuring the behavior of such processes. As will be shown, such computation mechanisms are capable of displaying most of the fundamental features commonly associated with biological computation.* They also provide a fertile ground for studying the behavior of systems of many interacting components that operate concurrently.

Although its applicability to biological systems cannot be proven, the study of such computing structures suggests that our hypothesis may indeed be relevant to biological systems regardless of the detailed operation of individual neurons. (For a discussion of the underlying biology, see ref. 7.) This dynamical mechanism differs from models of the brain based on analogies between Hebb's hypothesis and magnetic systems (8–11), where memory is attributed to static configurations of spin-coupling constants incapable of healing themselves. Furthermore, in these models fluctuations in the coupling strengths can produce errors in the outputs because of the degenerate nature of the energy minima. Also, the recognition processes involved in the nonlinear cases necessitate Monte Carlo routines in which the spin system may not converge to a fixed point but instead to a strange attractor (12), and when it does there is no a priori knowledge of when to halt. In contrast, our machine requires no random sampling and completes a recognition process in a fixed number of steps (equal to the number of rows in the array).

On a somewhat different level, the existence of self-repairing machines operating with our hypothesis leads to simple quantitative experimental measurements of systems exhibiting the features listed above. We believe that a good deal of quantitative information will be needed to understand the spontaneous emergence of collective computational properties in both brains and machines.

Computing with Attractors

The "Prism" machine that we designed consists of a rectangular array of simple processors, each of which operates on integer data received locally from its neighbors. Overall input and output to the machine takes place only along the edges. Such an architecture has the advantage of being readily implemented in VLSI chips, because long wires connecting distant processors are not needed. Although we found that a number of different rules and connections produced the same general behavior, for the sake of simplicity we focus on one specific implementation.

In particular, each processor has an integer memory value, constrained to lie in a fixed interval, and it obtains inputs from its neighbors in the previous row. The connections in a unit cell are shown in Fig. 1. Each processor multiplies the difference in its inputs by its internal memory value with a saturation to prevent unbounded growth in the data values. This procedure amounts to image processing through local enhancement of edges. Further details of the specific rules used here are given in ref. 4. In addition, the internal memory values can be adaptively adjusted to further emphasize differences in the data. Measurements show that this array is capable of adapting to a sequence of inputs and then recog-

Fig. 1. Diagram of the local connections between elements in the array. Diagonal lines carry data down through the array, and horizontal connections allow each element to adjust its internal state based on its neighbors' outputs.

nizing them, even when they are slightly distorted. This adaptive process can be quantified by using the stroboscopic sampling technique of d'Humieres and Huberman (13). If the external input signal to the array at step $k$ is denoted by $S(k)$, and if $R(k)$ is the resulting output vector from an array with $m$ rows, then the distance between the outputs corresponding to successive input periods is defined by

$$d(t) = \max_{\|P\|} \|R(m + k + P) - R(m + k)\|,$$

where $P$ is the period of the input cycle (4 in this case), and $k$ ranges over the set $\{tP + 1, tP + 2, \ldots, (t + 1)P\}$—i.e., the times at which the $t$th period of the input signal enters the array. As seen in Fig. 2, for an array of 64 elements a periodic presentation of 4 randomly chosen inputs produces a rapid convergence to a fixed point, as indicated by the fact that the distance $d$ goes to zero.

The fast approach to an attractor in phase space signals the emergence of a self-organized state. This higher level of organization is characterized by the fact that the period of the fixed point in the adaptation process is much smaller than the combinatorial capacity of the array.

These attractive fixed points produce recognition processes that are nonlinear because of saturation in the local computation rules. By fixing the values of the memory states and then sending inputs through the array in any order, we were able to determine which learned inputs produced different outputs. These processes map many different input patterns into each output and also are such that, in general, the output of the array is insensitive to small variations in the inputs. Thus, small distortions (as measured by their distance) of the learned inputs still produce the same outputs.

An interesting application of this architecture for pattern recognition of broad categories and distinctions therein is illustrated in Fig. 3. By setting the parameters in the two arrays to different values, one can control the size of the attractors. For instance, the array on the left could be set to have large attractors with many input patterns producing the same output. If the other array is set to produce narrow attractors for the same inputs, it will distinguish among inputs that the first one would classify as the same. Thus, by reading the output first on the left and then on the right, one can produce a broad category that can then be finely described by reading the second half.

**The Encoding Problem**

Since brains seem to have their own way of encoding patterns or concepts that are perceived to be similar, it is of interest to see if a mapping can be constructed between close
patterns as learned by our parallel machine and those learned by animal brains. That this is indeed possible is shown by the following experiment. Recently, Blough (14) has measured the ability of pigeons to learn and recognize the letters of the alphabet by determining the statistical correlation matrix produced by a recognition protocol of capital letters of the English alphabet. We have encoded his results by constructing a two-dimensional confusion matrix in which letters that were often confused are placed close together. By assigning a numerical scale to the axes, we have mapped each letter onto a pair of integers. This is illustrated in Fig. 4a, where we have parametrized his data by using integers between 0 and 9. Thus, for instance, the letters O, Q, and C were often confused and appear close together.

With this identification, a string of letters can be converted into a sequence of numbers, which can then be processed by the machine. Using this encoding, the machine was able to recognize distorted inputs by mapping them into the same outputs as the original undistorted inputs. An example is shown in Fig. 4b, where the machine first learned four 5-letter words and was later presented with a particular word, HELLO. As can be seen, the outputs produced by this word and by the word HEILQ are exactly the same.

We should also mention that by making the output vector an address in a database, the Prism can be made into a flexible content-addressable memory. This property, which is sometimes also called associative memory, is the ability to elicit a whole class of responses given any member as an input. The grouping of specific inputs into the same class requires the same procedure for proper encoding as was used above.

Adaptation and Learning

It has been pointed out that one of the central problems of neurobiology consists of understanding the mechanism whereby initial degenerate neuronal configurations develop the necessary specificity to deal with a changing environment (15). Moreover, one would like to understand how adaptation and learning can be placed within a computational framework and whether this mechanism can be quantified given a changing environment. We will now show that this selection process follows naturally from our hypothesis that computing with attractors leads to fault-tolerant behavior.

To study adaptation and selection within this context, we have performed a number of experiments that measure the size of the attractors of the Prism computers. Starting with a uniformly initialized array, we determined that adaptation to a sequence of input patterns made the attractors for the learned patterns larger at the expense of others. This was determined by examining the fraction of 10,000 random inputs that were mapped into each output. Also, the size of the attractors was characterized by measuring the fraction of all points at various distances from each learned input that was mapped to the same output as the learned input. This fraction is shown as a function of distance in Fig. 5 for three inputs learned by an 8 × 8 array. As can be seen, patterns close to the learned ones are more likely to be classified into the same group as the learned patterns. Therefore, an aggregation of patterns is produced that increases the computer's tolerance to errors in the inputs.

Finally, the self-organizing properties of these computing structures are illustrated by the fixed-point configuration of the array. As shown in Fig. 2b, the process of adaptation leads to a removal of the initial degeneracy (i.e., all memory elements start at the same value) by selecting a final state out of the multiplicity of possible ones. We will show below how learning additional inputs affects this process.

Self-Repair

The existence of stable attractors for these computing structures provides a mechanism of self-repair during the adaptation process. The dynamics of the array causes small fluctuations in either data or memory values to relax toward the fixed points. This general method of self-repair is very different from the standard techniques used for reliable computing. The latter are based on multiplexing schemes with majority rules and require an enormous amount of hardware per operation (3). Our technique uses instead the stable collective modes of the computer. Hence, the dynamics of this parallel machine is analogous to the behavior of a dissipative

![Fig. 5. Fraction of samples that are mapped into the same output as each of the learned inputs, as a function of the distance between the sample and the learned input. Data are shown for an 8 × 8 array after it had adapted to three different inputs.](Image)
FIG. 6. Conditional learning example. (a) Adaptation to a new set of inputs close to the original ones. (b) Adaptation to a new set far from the original set. In both cases, the new array is also shown with the modified values circled.

dynamical system with many degrees of freedom.

To study this effect, we introduced errors during the adaptive process either while the array was changing or after it converged to a fixed point. Such errors were introduced by randomly modifying some of the memory values and then allowing the array to proceed as before (soft upsets), or by permanently freezing the state of some of the elements as well as their outputs (hard upsets). Fig. 2c shows the state of the array immediately after five soft errors were introduced, with the circles indicating which elements were changed. The resulting rapid recovery is seen in Fig. 2a to the right of the arrow. The state of the array after it recovered was the same as shown in Fig. 2b. Furthermore, small errors in both data and memory in the recognition (non-adaptive) mode
produced the same output. Typically, soft upsets could be introduced in 20% of the memory elements of the array without causing it to relax to another attractor. Hard errors also allow for relaxation to the attractor provided they are fewer in number.

Conditional Learning

Another feature exhibited by biological computation is conditional learning. By this is meant the ability to learn a set of input patterns faster if they are close to previously learned ones. That these computational structures can display this behavior was established by examining the relative rates of convergence produced by different sets of inputs. Fig. 6 shows the results of such an experiment. First, a set of well-separated inputs (i.e., large euclidean distance between them) were presented to the array, and convergence to a fixed point was verified. Second, the dynamics of the array as it adapted to a second set of slightly different patterns was observed, as shown in Fig. 6a. As can be seen, learning the new set is especially rapid. Third, a new set of patterns differing greatly from the original ones was presented to the array. As depicted in Fig. 6b, the convergence is slower and the final state of the machine differs greatly from that obtained in the first case. Thus, our computing structure can learn inputs close to those it already knows with only small adjustments in its internal state, whereas large changes are required when learning very different inputs. This implies that slight alterations in the environment can be dealt with by very minor changes in structures that display collective computation and self-repair.

Summary

In this paper, we have shown that it is indeed possible to compute with attractive fixed points and that this mechanism leads naturally to a set of properties commonly associated with biological computation. These include fast self-repair, adaptation to the external environment, recognition of fuzzy inputs and discrimination among them, associative memory, and conditional learning. In contrast with many other schemes that have been proposed, computing with dynamic attractors shows self-organizing behavior, which is fault tolerant in the sense that changes in the parameters of the system do not result in drastically different states of the machine after a short recovery time.

As to the applicability of our results to actual brains, we should mention that they are not restricted to discrete structures, because continuous dynamical systems can also possess attractive fixed points. There remains, however, the issue of the classical Pavlovian conditioning, which was not addressed in this paper. In spite of this, these experiments suggest that computing with attractors may be applicable to biological computation.

We thank E. Wollack for his help in digitizing the pigeon confusion matrix. This work was supported in part by Research Contract N00014-82-0699 from the U.S. Navy Office of Naval Research. T.H. would also like to thank the Xerox Palo Alto Research Center for a fellowship.