FILE FOLDER LABELLED MAY 1972, AI PAPER. OFFSET PRINTED, WIDELY SENT.

Contents of Thesis Outline

NOW CALLED CRITIC OR VALUE APPROXIMATOR

Outline of Thesis Plans, 9/23/72

Paul J. Werbos

I. Proposed View of Mammalian Intelligence:

(i) **Mammalian intelligence has three basic tasks it must perform:**

(a) provide a description of its environment, from empirical data (and from the assumption that induction is valid);

(b) calculate its "reinforcement level", U, which may be regarded as a cardinal utility function;

(c) find the best possible approximate solution to the dynamic programming problem of maximizing the long-term expected value of U, given the description (a) of its environment.

(ii) I have argued for this approach elsewhere, and will not clutter up the present outline with a repitition of those arguments.

II. Thesis Objectives:

(i) To describe adaptive systems capable of performing tasks (a) and (c) above, in the same general way as the human brain. In other words, to formulate a theory of these two phases of human intelligence. (My long-term objective is to describe all three; my oral exam focused more on phase (b).) This objective has been completed enough for me to move on to my second objective; insofar as the theory is difficult to understand, these experiments will clarify, at least, the power of the systems described.

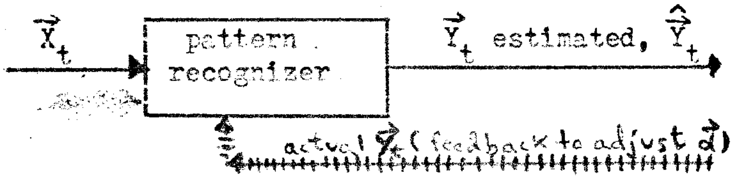
(ii) To program the major part of the proposed systems in FORTRAN, and to test their capabilities in problem-solving and pattern-recognition. Primary experiment: to use these systems to provide a "position evaluation system", for use in existing FORTRAN chess-playing systems, to improve their performance. Secondary experiments possible : almost any adaptive pattern-recognition problem, except for those requiring excessive computer memory; Uhr's handwriting samples should provide a possible empirical test, giving a comparison with previous adaptive pattern-recognition systems.

III. Sketch of the Mathematical Theory of Intelligence:

(0) In this section, I plan to sketch out a general mathematical class of intelligent systems, of which I claim the human brain is a member; in the third subsection, I will relate the theory to human brain anatomy.

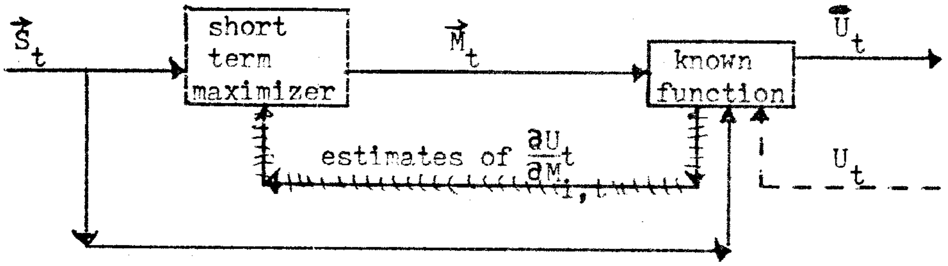
(i) Two types of subsystem seem unavoidable in any intelligent system:

(a) Pattern-recognizers, systems that estimate some dependent variable, $\vec{Y}_t$,


$\vec{X}_t$
pattern recognizer
$\vec{Y}_t$ estimated, $\hat{\vec{Y}}_t$
actual $\vec{Y}_t$ (feedback to adjust $\vec{\alpha}$)

from the value of an independent variable, $\vec{X}_t$; if a pattern-recognizer is adaptive, it must include internal coefficients, $\vec{\alpha}$, which get adjusted in time, to fit the estimates of $\vec{Y}_t$ to the actual observations. Standard statistical estimaters - like multiple regressions - are a form of adaptive pattern recognizer, but too weak for our purposes here.

(b) Short-term maximizers, systems that try to maximize



a known function of $\vec{M}_t$ and $\vec{S}_t$. $\vec{M}_t$ would correspond to "action by the system" (i.e. muscular activation), while $\vec{S}_t$ would correspond to "data from the environment" (sensory input). To calculate the derivatives above - which tell us how we should adjust our action vector - we clearly need to know U_t as a function of $\vec{M}$, and to work back through the known function via the chain rule for differentiation. If the relation of U_t to $\vec{S}_t$ and $\vec{M}_t$ is not known, at first, then it can be calculated by a pattern-recognizer.

Short-term maximizers and pattern-recognizers can be built on similar principles, which I will discuss below. Both may be "recursive"; i.e., they may generate a kind of internal memory, $\vec{Z}_t$, to be saved, and brought in as if it were part of $\vec{S}_{t+1}$ in the next cycle.

In the charts above and below, normal data flows are indicated by straight lines; feedback used to adjust the coefficients of the system receiving it is indicated by cross-lined lines; feedback to be passed on through a more or less fixed conduit is indicated by broken lines.

(ii) Let us assume that our intelligent system inputs $\vec{S}_t$ (including U_t as a component) and then outputs $\vec{M}_t$, in every time cycle. The first major component of our system is the "reality-constructor": a psttern-recognizer, with an internal memory, which predicts $\vec{S}_{t+1}$ as a function of $\vec{S}_t$, $\vec{Z}_t$ and $\vec{M}_t$. $\vec{S}_t$ alone describes only a part of the external environment, the part currently visible; $\vec{S}_t$ and $\vec{Z}_t$ combined form a reconstruction of the total external environment; let us call the combined vector, "$\vec{X}_t$". (This system is responsible for what psychologists call, "object permanence.")

In order to solve the problem of maximizing the expected value of U_t, in the long-term, we can use the Hamilton-Jacobi-Bellman formula:

$$J(\vec{X}_t) = (U_t - \bar{U}) + \max_{\vec{M}_t} E(J(\vec{X}_{t+1}) | \vec{X}_t, \vec{M}_t)$$

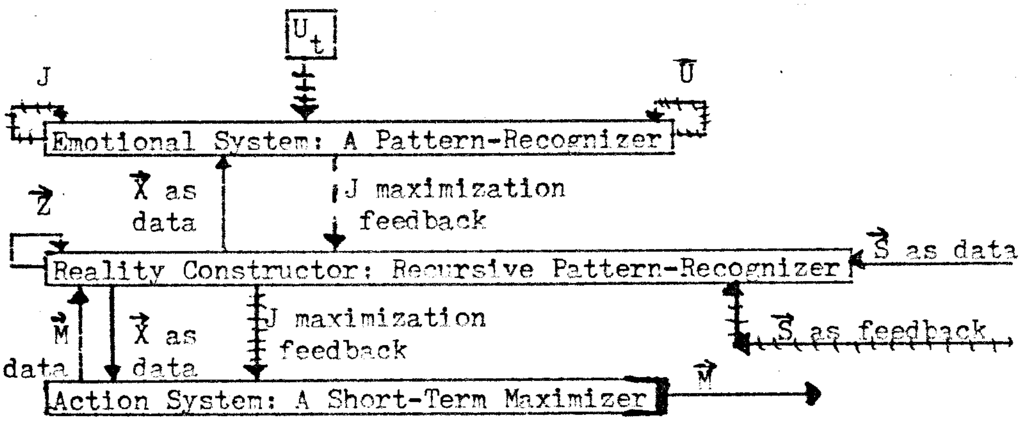
"$\bar{U}$" equals the mean U_t with the optimal strategy. Intuitively, $J(\vec{X}_t)$ refers to the total value of being in situation $\vec{X}_t$; it equals the intrinsic value of being in $\vec{X}_t$ (i.e. $U(\vec{X}_t)$) plus

~~the~~ the strategic value of being in $\vec{X}_t$ (i.e., $E(J(\vec{X}_{t+1}))$, the expected total value of future $\vec{X}$, given the right choice of action). Howard has shown that we can solve for the optimal strategy, $\vec{M}_t(\vec{X}_t)$, by iterating $\vec{M}^{(n)}(\vec{X}_t)$ and $J^{(n)}(\vec{X}_t)$ together:

$$J^{(n+1)}(\vec{X}_t) = U_t - \overline{U}^{(n)} + E(J^{(n)}(\vec{X}_{t+1}) \big| \vec{X}_t, \vec{M}^{(n)}(\vec{X}_t) \text{ at } t))$$

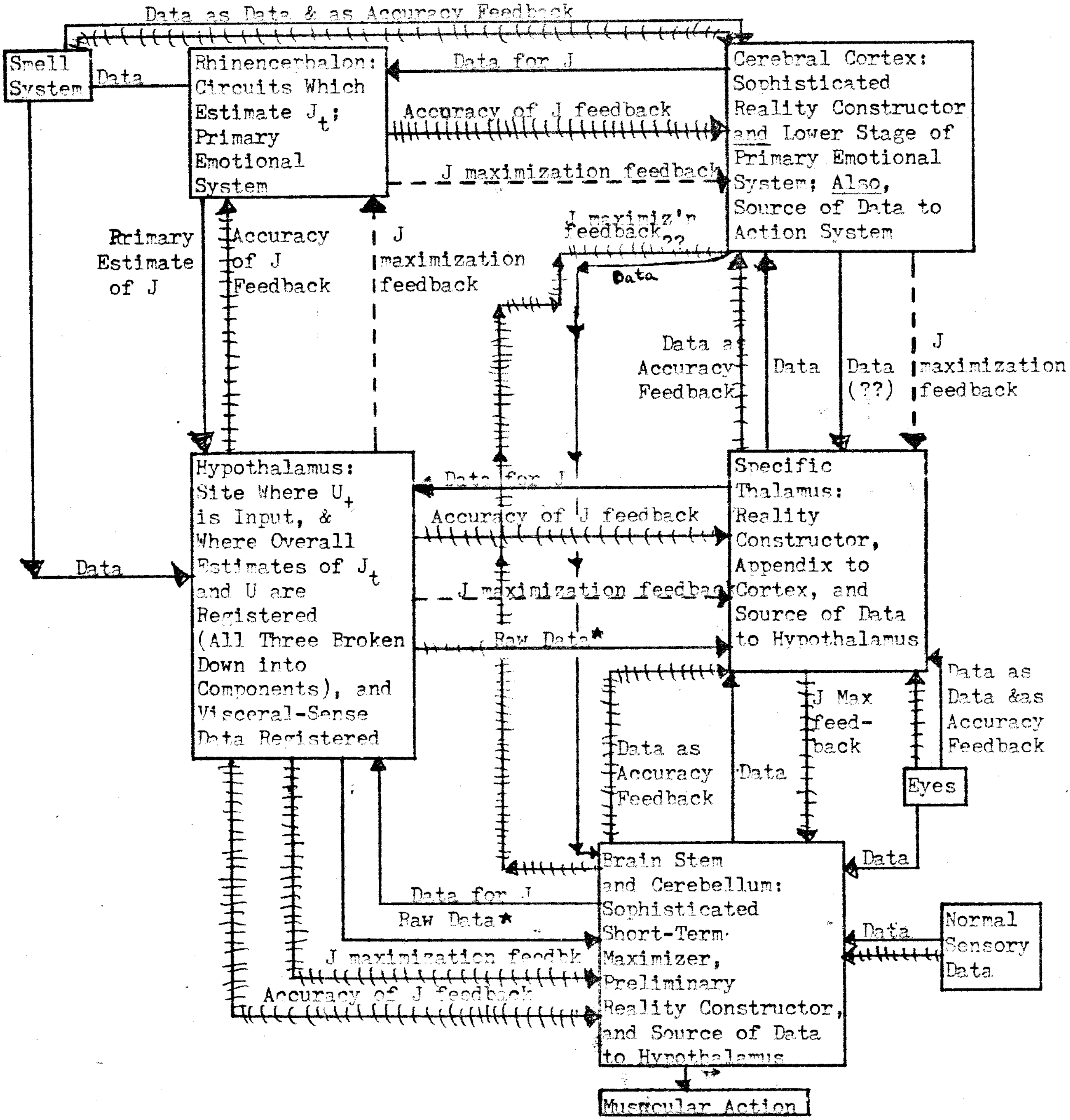
$$\vec{M}^{(n+1)}(\vec{X}_t) = \vec{M} \text{ which maximizes } E(J^{(n)}(\vec{X}_{t+1} \big| \vec{X}_t, \vec{M} \text{ at } t))$$

The second major component of our intelligent system is an "emotional system": a pattern-recognizer (with or without memory) which adjusts the circuit calculating "$J(\vec{X}_t)$" to be a predictor of $U_t + J(\vec{X}_{t+1})$+constant, the last expression to be treated as if it were a constant function, as in normal iteration. (In the language of psychiatry, this might be called a "cathexis-generating system.") The calculation of the constant will require a slight modification to the way the pattern-recognizer is used. The third major component of our system is an "action system" : a short-term maximizer which maximizes $J(\vec{X}_{t+1}(\vec{X}_t, \vec{M}_t))$, given the known functions $\vec{X}_{t+1}(\vec{X}_t, \vec{M}_t)$ and $J(\vec{X}_t)$ developed by the two other major components. Thus the reality constructor will act as a conduit in transferring feedback from the emotional system to the action system, because it is part of the compound "known function."



In essence, that's all one has to put together to get a complete intelligent system. We have to design two types of subsystem - pattern-recognizer and short-term maximizer - and plug them in together as above. The three components can be changed around a little, to encourage the growth of internal memories of all kinds; also, the brain may use complex tricks to economize on time and circuitry, even when the benefits are only on the order of a 20% cost reduction. But the basic idea is as above.

(iii) In this theory, the anatomy of the human brain may be sketched out:


Data as Data & as Accuracy Feedback
Smell System
Data
Rhinencephalon: Circuits Which Estimate J_t; Primary Emotional System
Data for J
Accuracy of J feedback
J maximization feedback
Cerebral Cortex: Sophisticated Reality Constructor and Lower Stage of Primary Emotional System; Also, Source of Data to Action System
J maximiz'n feedback ??
Data
Primary Estimate of J
Accuracy of J Feedback
J maximization feedback
Data as Accuracy Feedback
Data
Data (??)
J maximization feedback
Hypothalamus: Site Where U_+ is Input, & Where Overall Estimates of J_t and U are Registered (All Three Broken Down into Components), and Visceral-Sense Data Registered
Data for J
Accuracy of J feedback
J maximization feedback
Raw Data*
Specific Thalamus: Reality Constructor, Appendix to Cortex, and Source of Data to Hypothalamus
Data
Data as Data &as Accuracy Feedback
Data as Accuracy Feedback
Data
J Max feed-back
Eyes
Data
Brain Stem and Cerebellum: Sophisticated Short-Term Maximizer, Preliminary Reality Constructor, and Source of Data to Hypothalamus
Data for J
Raw Data*
J maximization feedbk
Accuracy of J feedback
Normal Sensory Data
Data
Muscular Action

Many of these connections add a kind of fine tuning which will not be intelligible until we get to the problem of designing pattern-recognizers and short-term maximizers; they all have a sound foundation in neurophysiology, except for the two with question marks on them. For example, let me quote Morgan & Stellar: "Accounting in part for the apparently needless complexity of the olfactory system is the fact that smell was very important and highly developed in many of the lower animals. Man seems to carry over this system in its complexity, even though he has little use for it."(Physiological Psychology, p.113, 2nd ed.) The olfactory system does not appear to be unique, as we will see.

probably comes in like optical data, in practice

Essentially: the rhinencephalon plus the major part of the hypothalamus form the "emotional system" of our discussions above. The thalamus provides a "reality constructor." The cerebral cortex, however, is a hybrid system, which provides a larger "$\vec{X}_t$" input to the emotional system than the thalamus alone could provide. It, too, is a pettern-recognizer, designed to help predict $U_t+J(\vec{X}_{t+1})$ - as if the cortex were a lower part of the rhinencephalon - <u>and</u> the $\vec{X}_{t+1}$ generated by the thalamus, <u>both</u>, as a single combined vector. The brain stem and cerbellum are basically the action system, receiving feedback from the known model of $J(\vec{M}$,etc.) provided by the circuits above them. The calculation of J gets done upwards, from thalamus to cortex to rhinencephalon to hypothalamus, in Papez' classic loop; therefore, the feedback to our short-term maximizer flows in the reverse direction.

That would be the whole story, except that the brain has been built to take inexpensive shortcuts whenever possible, especially when the short-cuts were once present as major systems in earlier stages of biological evolution. If all the senses had a direct pathway to the diencephalon, as do vision and the chemical senses, then the lower brain could act completely as an action system. However, most raw sensory data and all direct muscular activations are registered only in the lower brain, which then relays a summary of both up to the thalamus; this summary can be of value to the thalamus only if the lower brain also acts as a partner with the thalamus in reality construction. (Even the cerebellum sends data to the reticular formation for use, not in action, but as data for the cerebral cortex.) Furthermore, the hypothalamus, in computing J, gets its data as directly as possible from the original source; therefore, it can draw data directly from the thalamus and brain stem, on simple variables such as posture, and encourage the brain stem to measure and maintain such variables effectively. Thus the lower brain is a <u>hybrid</u> reality constructor, action system and miniature emotional system; the construction of such hybrids will be discussed below.

The neurophysiologist may be somewhat impressed by this mathematical formulation of Papez' classic theory. However, he may wonder where I have put the corpus striatum. The striatum, I would claim, is a fossil cerebral cortex, kept intact (but reduced) in the mammalian brain, only because its simpler design allows it to carry out simple functions more cheaply than the cerebral cortex does them. The rhinencephalon is known (Pribram) to have three major divisions - the first, oriented towards smell (a thalamus for smells alone?), the second, which uses striatal data as its major input, and the third, which inputs from the cerebral cortex. Once it was thought that the striatum was a kind of motor system, because injuries there could lead to muscular tremors. However, when the connections <u>from</u> the thalamus <u>to</u> the striatum are cut off, such tremors disappear, implying that the main effect of the striatum on action comes by (chemical) feedback on the <u>reverse circuit</u> [illegible] thalamus to the brain stem (Gardner, <u>Fundamentals of Neurology</u>, p.286). As with the cerebral cortex, the outgoing electrical connections may be thought of as pure data, made available to the brain stem, but carrying no forceful feedbacks forwards along with it.

The epithalamus, in this framework, may be thought of as an annex to the hypothalamus; the pineal gland and the pituitary gland measure two distinct components of U_t.

The cerebellum and the brain stem, like the cerebral cortex and the striatum, may be treated as two systems designed to do the same thing, one in a sophisticated, expensive way, the other in a simpler way.

The nonspecific thalamus defines the unit "cycle time" used in the forebrain; it defines the difference between what I have called "t+1" and "t", for the subsystems most sensitive to this definition.

In REM sleep, the system reconstructs imaginary situations, and updates them by use of the (cortical) reality construction system; the reality construction system, after all, is adjusted to provide a model of the laws of nature, to predict future $\tilde{S}$. In REM sleep, the system adjusts the circuits which measure J and which maximize J to make them consistent with the known laws of nature.(i.e. It carries out a kind of simulation study.) In deep sleep, the subsystems attend to their own internal adjustments.

IV. A Discussion of Alternative Possible Theories of Intelligence.

From experience, we know that the human brain does not have to imagine the long-term implications , first, before every minor decision about every possible minor action. Somehow, it has circuits that allow one to evaluate different actions, directly, without having to form new images of the distant future, over and over again. From dynamic programming, we know that there is really only one way for such a system to choose between different actions, via a direct functional evaluation - by approximating the Bellman J function, cited above. This function is defined by the Hamilton-Jacobi-Bellman equation, as above; in an open-ended situation, we are forced to use this equation, as above, to form the core of our system.

However, there is one way we could apply the equation a bit differently - we could let the time interval between "t" and "t+1" vary substantially. We could let the time interval be very large on some levels, to allow our emotional system to jump ahead easily in time. We could imagine that the brain uses "plans" or "action-schemas" (of long duration) as its units of time and action, instead of restricting itself to small cycle times. George A. Miller, former head of the Harvard Psychology Department, has proposed such a theory; by now, the theory has received strong support in many places. The theory also makes mathematical sense; much of my work this past year has been in exploration of the details of the theory.

However: the mathematics of the theory turn out to be cumbersome, irrevocably so. They require a stratification of the cerebral cortex and of the thalamus into ~~different~~ disconnected regions, for different levels of time aggregation. They require that the long-term planning parts of the cortex (frontal lobes) be fairly free of normal (high frequency) brain waves, and they require that most "dreaming" involve no visual imagery; none of this makes empirical sense. Also, some of Piaget's work suggests that explicit "planning" is a difficult learned ability, similar to language. Most important of all: the simpler theory above can do the same work, anyway, if its pattern-recognizers are sophisticated enough.

For these reasons, I do not intend to program this alternative theory. For intellectual honesty, I will put a chapter in my thesis somewhere describing the problems of the theory in more detail. But the bulk of my thesis will simply formulate a newer, and simpler, theory of mammalian intelligence.

V. Adaptive Pattern Recognizers.

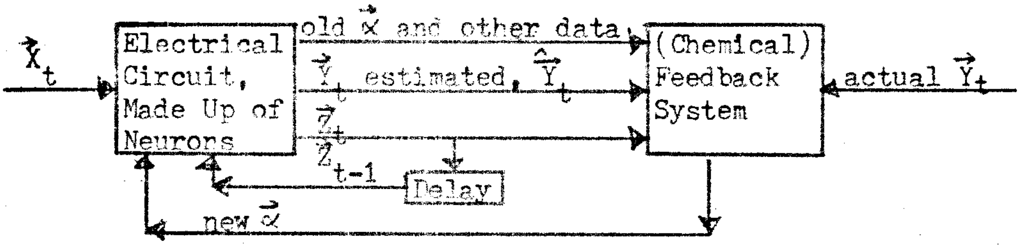
(i) General Description. An adaptive pattern recognizer, similar to those of the human brain, will have four critical properties:

(a) It will input an "independent" vector $\vec{X}_t$ and a "dependent" $\vec{Y}_t$ in every time cycle ($\vec{X}_t$ at the beginning of the cycle, $\vec{Y}_t$ the end, usually

(b) It will contain an electrical network, made up of large numbers of equivalent subsystems ("neurons", possibly a handful of different types), a network which inputs $\vec{X}_t$ and outputs a "predicted" $\vec{Y}_t$.

(c) Each subsystem will have internal coefficients, $\vec{\alpha}$, which determine its electrical response-function, and which will be adjusted by a "chemical feedback" system designed to encourage P($\vec{Y}_t$ will be the circuit's prediction $|\vec{X}_t$) to equal P($\vec{Y}_t$ will equal the actual $\vec{Y}$ input late in the cycle$|\vec{X}_t$), approximately.

(d) The electrical circuit may also generate a "$\vec{Z}_t$", to be input in the next time cycle as an annex to X_{t+1}.



$\vec{X}_t$
Electrical Circuit, Made Up of Neurons
old $\vec{\alpha}$ and other data
$\vec{Y}_t$ estimated, $\hat{\vec{Y}}_t$
$\vec{Z}_t$
$\vec{Z}_{t-1}$
Delay
new $\vec{\alpha}$
(Chemical) Feedback System
actual $\vec{Y}_t$

Concept of Pattern-Recognizer: NOT Physical Description

(e) In practice, the electrical circuit and the feedback system need not be turned on in every time cycle, if there are reasons to turn them off from time to time; our requirement, mathematically, is to set up a system which produces an electrical network capable of predicting $\vec{Y}_t$.

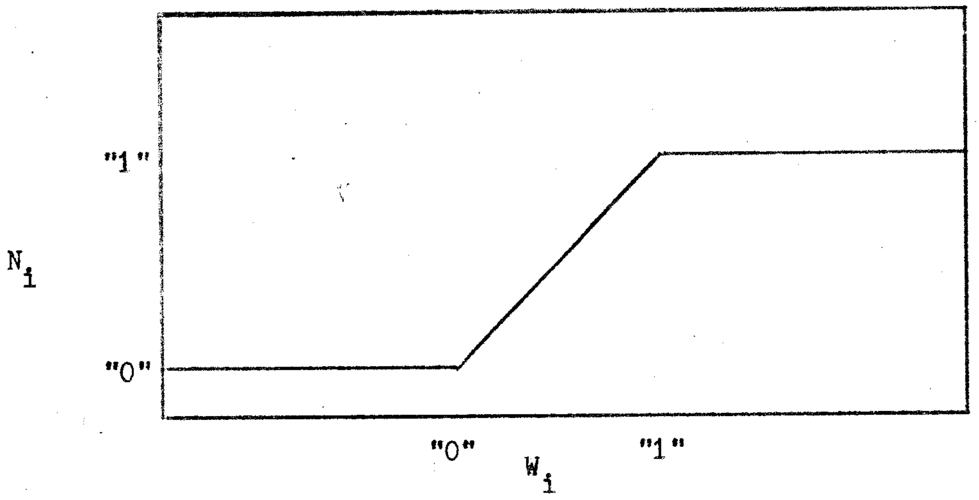
(ii) A simple pattern-recognizer, without a recursive component $\vec{Z}_t$, can be designed as follows:

(a) The subsystems, "neurons", may be ordered in such a way that each neuron, #j, can receive the output, N_i, of another neuron, #i, only if i precedes j; this is simply a more convenient way of saying that the circuit is not recursive. The prediction of $\vec{Y}_t$ would come from the last group of neurons; each component of $\hat{\vec{Y}}_t$, $\hat{Y}_{i,t}$, would be generated as the output of a neuron, as N_{k_o+i}. k_o plus the number of components in $\vec{Y}_t$ would, of course, equal the total number of neurons.

(b) Let us assume that the output of every neuron, N_i, is bounded by a minimum, which we may call "0", and a maximum, "1". We can assume that the dendrites of neuron #i calculate a number:

$$W_i = \sum \alpha_{ij} N_j,$$

the sum to be taken over the few neurons, #j, which connect to neuron #i; N_i may equal W_i, unless $W_i < 0$ or $W_i > 1$, in which cases N_i is brought back to 0 or 1.



The linear model to calculate W_i is standard in the literature, and will also simplify the chemical feedback system; the graph above for N_i as a function of W_i will simplify the feedback system, while the usual TLU model would not allow a rational feedback system at all. (Given that neurons often tend to output bursts of spikes, of varying intensity, the old all-or-nothing formulation is unnecessary.)

Given an error function, $e(\vec{Y}_t, \hat{\vec{Y}}_t)$, we can try to minimize E(e) by steepest descent; i.e. we can change each α_{ij}, in every time period, by an amount proportional to:

$$\frac{-\partial e}{\partial \alpha_{ij}} .$$

From knowing $\vec{Y}_t$ and $\hat{\vec{Y}}_t$, we should already know:

$$\frac{\partial e}{d\hat{Y}_i}(t) = \frac{\partial e}{dN_{k_0+i}}(t) \; ; \qquad \text{(assuming no connections amongst last neuron}$$

to calculate the derivative with respect to α_{ij}, we need only use the chain rule:

$$\frac{de}{d\alpha_{ij}} = (N_j \frac{de}{dN_i} ; \quad \text{(i.e. impact of } \alpha_{ij} \text{ mediated by } N_i \text{, its only direct site of impact)} \quad \text{or zero, if } W_i>1,<0$$

the derivative with respect to N_i, for any neuron #i, can be calculated backwards, from the last neurons to the earliest, by the chain rule, which gives:

$$\frac{de}{dN_i} = \sum_j \frac{de}{dN_j} \times (\alpha_{ji} \text{ if } 0 \leq W_j \leq 1 \text{; otherwise zero)}$$
$$+ \text{ direct effect, if any, of } N_i \text{ as part of } \vec{Y}.$$

This formula can be calculated out, in a physical network of neurons, by having de/dN_i sent backwards, along the same pathways as the electrical connections follow, as a kind of chemical feedback. (A more precise formulation of how the chain rule is to be used here may be found in Section III of my report of May 1972.)

(c) For each α_{ij}, we have the problem of figuring out θ_{ij}:

$$\Delta\alpha_{ij} = -\theta_{ij} \frac{de}{d\alpha_{ij}} .$$

As α_{ij} converges to a stable value, we would want θ to decrease. We can set:

$$\Delta\theta_{ij} = k\theta_{ij} \, s((\Delta\alpha_{ij}, t)(\Delta\alpha_{ij}, t-1)),$$

where s is the sign function; i.e. s=+1 for positive arguments, -1 for negative. This formula will adjust θ_{ij} until the increments of α_{ij} have no correlation with each other through time, as statistical theory recommends. We have three ways of deciding "k":(i) a priori; (ii) by a formula similar to that for θ_{ij} itself, using an a priori k_2; (iii) by a global k, adjusted by a formula similar to that for θ_{ij} itself, but <u>summed</u> over all active (i,j) pairs, with an a priori k_2. One can imagine more complicated possibilities, but the justifications for them do not hold up under careful scrutiny; the details are not worty going into here. With <u>any</u> of these three adaptation systems, θ will grow larger whenever de/dα tends to be consistent from one iteration to the next; therefore, these systems, used as emotional systems, may converge on the true J more ~~qui~~ quickly than a system obeying Howard's original formula.

(iii) There are four basic types of adaptive pattern-recognizer which appear relevant to the design of the human brain:

(a) The simple pattern-recognizer, as above.

A critical property of this pattern-recognizer: in one "time cycle", the "$\vec{Y}_t$" must be calculated up from the $\vec{X}_t$, and the chemical feedbacks must ~~be~~ then be calculated back down to neurons near the $\vec{X}_t$ level. This would suggest that one "time cycle" must correspond to <u>many</u> time-cycles of processing by individual neurons. (This is not obvious; one could still hope to process a number of different "$\vec{X}_t$" in the same system, in parallel, each a neuron cycle behind the other. However, this would make the chemical feedback process much too unmanageable; the feedback formula for $de/dN_i(t)$ requires knowledge about $W_j(t)$, not about $W_j(T)$, where "T" is the time cycle neuron #j would be processing electronically at the time it receives feedback about time t.) One would need a <u>pulsing system</u>, to signal the arrival of a new time-cycle to the pattern recognizer.

In practice, the nonspecific thalamus does generate such a pulsing signal, about 0.1-.05 sec. per pulse (alpha rhythm and beta rhythm, respectively). These time cycles are much larger than the 3 milliseconds or so per synapse in the cerebral cortex. Movie makers have known for a long time that human visual perception does, in practice, pulse itself in time cycles larger than the time of movie frames (about 1/16 sec.). Also, one might expect that vision - which requires a more complex hierarchy of processing than does touch to predict even short-term movements - would operate on a larger time-cycle than does touch; this does turn out to be true. (Visual alpha rhythm versus somatic beta/gamma rhythms.)

<u>However</u>: all of these timing requirements also apply to recursive pattern-recognizers, the ones ~~which~~ with a $\vec{Z}_t$ added. Also, if there are pattern-recognizers in the brain stem, they are not controlled by thalamic pulsing or by any other known pulsing system. Therefore, we have to consider both extended and abbreviated versions of the simple system above.

(b) The recognizer-arc system - an abbreviated pattern-recognizer.

From the reasoning above, it should be clear that even the simplest multi-stage pattern-recognizer requires some kind of pulsing system, to prevent neurons from operating at their natural maximum frequency. Therefore, if we wish to construct a pattern-recognizer *without* such pulsing, it must be a one-stage system. In other words, the neurons predicting $\vec{Y}_t$ must take their input *directly* from $\vec{X}_t$, not from other neurons in the same system. We adapt these neurons as we do the neurons in the multi-stage system; this is only a special case of the mathematics above.

How can we construct a complex hierarchy of pattern-recognition out of one-stage pattern-recognizers?

Suppose that we are trying to predict $\vec{S}_{t+1}$ from $\vec{M}_t$ and $\vec{S}_t$. With a one-stage pattern-recognizer, we can generate predictions, $\hat{\vec{S}}_{t+1}$, essentially available *at* time t+1. That's it. To achieve the *effect* of a two-stage pattern recognizer, we can then treat $(\vec{S}_{t+1}-\hat{\vec{S}}_{t+1})$ as a *new* vector of variables, to be input and predicted by another one-stage pattern-recognizer built on top of the first. And so on. In fact, we could even pass the new vector on up to a higher, multi-stage pattern recognizer. (We would also perhaps pass on all other data at time t+1 as data for making the prediction.) Such a stepladder system of pattern-recognizers I would call a "pattern-arc system" - named for the stepladder system of sensory/motor arcs ("reflex arcs") familiar in the spinal cord and brain stem.

The pattern-arc system has one advantage over a simple multi-stage system, but many disadvantages. The advantage: it cuts the unit of time down to the neuron cycle time; this can be very helpful to animals whose survival depends on split-second reactions, and whose reaction networks take a significant fraction of a second to do their work. However, this is a matter of low-level motor coordination, not a matter of thought. The brain does have such "arc" systems in the brain stem, involving both pattern recognition and motor control. But on a higher level, the brain relies on a pulsed system. The arc system may work well when predictions can be made one step at a time, but when one needs several levels of processing to make any decent predictions, then one needs an optimal, pulsed system. Thus, kinesthetic and auditory senses go through unpulsed, brain stem arcs before being sent up to the thalamus, but vision, which requires complex pattern-recognition for even the simplest predictions, goes directly to the thalamus.

The cerebellum, on the other hand, *is* pulsed, at a frequency of 200 cycles per second; according to recent work by Eccles, this pulsing seems to be generated by a "recurrent inhibition" system in the core of the cerebellum, similar to the nonspecific system of the thalamus. (This high frequency might involve the use of an expensive, electronic feedback system within the cerebellar cortex, in place of the usual chemical system.) Apparently, the brain stem is a kind of primitive equivalent to the cerebellum, preserved because it can carry out the simpler functions of the lower brain at a lower cost in terms of circuit complexity,.

Given that my primary goal in this thesis is to describe the higher intelligence of mammals, given that my experiments will not address themselves to the question of motor coordination, and given that we are facing serious pressures to speed up this work, the primitive arc systems of the brain stem will receive no further attention here.

(c) Recursive Pattern-Recognizers.

From the two subsections above, we should already begin to realize that timing efficiency is the chief source of trouble in designing an intelligent system. We were able to design an optimal multi-stage pattern-recognizer, but only by installing a pulsing system to give it time to feed back from each prediction, $\vec{Y}_t$, down to the level of the raw data, $\vec{X}_t$, used to generate the prediction. With recursive systems, this kind of optimality becomes, not merely expensive, but impossible. (For biological brains, at least.) If we were satisfied to optimize the circuit generating $\vec{Y}_t$ from $\vec{X}_t$ and $\vec{Z}_{t-1}$, we could treat the last two vectors as raw data and proceed as before. But how do we optimize the circuit which generates Z_t? In other words, how do we optimize the recursive pattern-recognizer as a total system?

In principle, we should use the formulas of page 8 to feed back from each prediction, $\vec{Y}_t$, down to the level of the raw data used to generate the prediction: $\vec{X}_t, \vec{X}_{t-1}, \ldots \vec{X}_{-m}$. For a computer brain, this approach might make a lot of sense (exact formulas in my outline of May 1972); it would give us a way to translate the charts on pages 3 and 4 directly into mathematics. The computer brain could simply stop everything, at some times t, and feed backwards through its past history. (Hybrid systems and short-term maximizers would involve no extra difficulty at all, since any derivative would be fed back by the same formulas, as on page 8.) But biological brains have shown neither willingness nor ability to do this sort of thing. Admittedly, humans do stop things to go to sleep. But REM sleep takes imaginary situations forwards in time, not real ones backwards. Deep sleep, according to recent research (e.g. Evarts, MIT Neurosciences Series), involves a slackening of cells' interconnections even within the cerebral cortex; it also involves forwards-moving "abstract dreaming" within the hippocampus; in short, it does not involve a synchronized, realistic movement backwards in time. In fact, the neurons of biological brains probably couldn't come close to the total recall of previous states and the total synchronization required by such a feedback system. In short: the human brain muddles through with a feedback timing system somewhat less than optimal.

There are two probable alternatives for such an ad hoc system - a limited memory system, and a decay system. With a limited memory system, the system remembers $\vec{Z}_{t-1}$ and the old inputs to Z_{t-1} even after $\vec{Z}_t$ has been generated, in the electronic phase of each time-cycle; in the chemical phase, it then feeds back to the input coefficients of $\vec{Z}_{t-1}$, using the formulas from page 8. A decay system is a more primitive, ad hoc system; it would use the current X_t, or even a moving average of past and present X_t, to [illegible] ~~feedback formula~~ the feedback formula of any feedback which reaches it; the feedback passed on would be decayed to a fraction, 1-θ, of what it would have been, to account for this degrading of information. The moving average and the feedback decay would both be based on θ, the same expoential decay factor, probably equal to one minus the autocorrelation of $N_{i,t}$. Whatever its other disadvantages, a decay system would allow small neurons to operate at their natural frequencies, without direct control, and to establish recursive connections which *are not* delayed by a large time-cycle (i.e. from "t-1" to "t", which equals many *neuron* time-cycles).

The small stellate neurons of the cerebral cortex (Layer IV) seem to be a clear case of cells operating on a decay system; they have multiple recurrent interconnections, under no external pulsing control, operating at high frequency. The large pyramidal cells of all three cortices - cerebral, cerebellar and limbic - could be operating, in theory, by a limited memory system, since all of them are under the direct control of some pulsing system.(e.g. nonspecific thalamic fibers to apical dendrites of pyramids in visual cortex, observed by Scheibel and Scheibel, Rockefeller *Neurological Sciences Study Program*, vol.ii.) *Both* feedback systems will be programmed in FORTRAN - plus perhaps a more powerful backwards-memory system - in the next phase of my thesis work.

The limited memory system, when used in a *hybrid* emotional and reality-constructor system, may be what the statisticians call "inefficient but exact"; in other words, it may force our circuits to converge to the right configuration, but at less than the maximum possible speed. It may force the cortex to construct an active memory, $\vec{Z}_t$, which records everything about the external world of relevance to the system's decision-making. If such a system has a set of decay-based circuits attached to it - like the stellate cells - these cells will not corrupt the higher system, but will merely provide it with a larger set of data to work with.

(d) Negentropic Pattern-Recognizers

My original goal was to conjure up $\hat{Y}_t$ according to their actual probabilities. The systems above do not do this; they are deterministic. Also, we have little reason to believe that these systems will help *extract out* the variables of greatest interest to higher-order systems.

At first, these points may seem academic. Suppose, for simplicity, that we presume our system gets raw input in the form of ones and zeroes. (This is still consistent with our assumption that intermediate signals may be continuous.) Then for each $Y_{t,i}$, our pattern-recognizers would still produce a $\hat{Y}_{t,i}$ which varies from zero to one; we could interpret $\hat{Y}_{t,i}$ as a probability, and plug into our pattern-recognizer any error function which encourages $\hat{Y}_{t,i}$ to equal the probability of $Y_{t,i}$ being "one". (To minimize $E((Y_{t,i}-\hat{Y}_{t,i})^2)$, for example, we set $\hat{Y}_{t,i}=p(Y_{t,i})$. A question of minimizing $p(1-\hat{Y})^2+(1-p)\hat{Y}^2$.) Then, to conjure up *possible* $Y_{t,i}$, we could simply "throw the dice" for each i, based upon $\hat{Y}_{t,i}$ as the probability of firing a "one".

However, life is not so simple. A simple example will show why this approach will not work, and why its failure is of enormous importance.

probably equal to one minus the autocorrelation of $N_{i,t}$. Whatever its other disadvantages, a decay system would allow small neurons to operate at their natural frequencies, without direct control, and to establish recursive connections which are not delayed by a large time-cycle (i.e. from "t-1" to "t", which equals many neuron time-cycles).

The small stellate neurons of the cerebral cortex (Layer IV) seem to be a clear case of cells operating on a decay system; they have multiple recurrent interconnections, under no external pulsing control, operating at high frequency. The large pyramidal cells of all three cortices - cerebral, cerebellar and limbic - could be operating, in theory, by a limited memory system, since all of them are under the direct control of some pulsing system. (e.g. nonspecific thalamic fibers to apical dendrites of pyramids in visual cortex, observed by Scheibel and Scheibel, Rockefeller Neurological Sciences Study Program, vol. ii.) Both feedback systems will be programmed in FORTRAN - plus perhaps a more powerful backwards-memory system - in the next phase of my thesis work.

The limited memory system, when used in a hybrid emotional and reality-constructor system, may be what the statisticians call "inefficient but exact"; in other words, it may force our circuits thxxxxxxxx to converge to the right configuration, but at less than the maximum possible speed. It may force the cortex to construct an active memory, $\vec{Z}_t$, which records everything about the external world of relevance to the system's decision-making. If such a system has a set of decay-based circuits attached to it - like the stellate cells - these cells will not corrupt the higher system, but will merely provide it with a larger set of data to work with.

(d) Negentropic Pattern-Recognizers

My original goal was to conjure up $\hat{\vec{Y}}_t$ according to their actual probabilities. The systems above do not do this; they are deterministic. Also, we have little reason to believe that these systems will help extract out the variables of xx greatest interest to higher-order systems.

At first, these points may seem academic. Suppose, for simplicity, that we presume our system gets raw input in the form of ones and zeroes. (This is still consistent with our assumption that intermediate signals may be continuous.) Then for each $Y_{t,i}$, our pattern-recognizers would still produce a $\hat{Y}_{t,i}$ which varies from zero to one; we could interpret $\hat{Y}_{t,i}$ as a probability, and plug into our pattern-recognizer any error function which encourages $\hat{Y}_{t,i}$ to equal the probability of $Y_{t,i}$ being "one". (To minimize $E((Y_{t,i}-\hat{Y}_{t,i})^2)$, for example, we set $\hat{Y}_{t,i}=p(Y_{t,i})$. A question of minimizing $p(1-\hat{Y})^2+(1-p)\hat{Y}^2$.) Then, to conjure a possible $Y_{t,i}$, we could simply "throw the dice" for each i, based upon $\hat{Y}_{t,i}$ as the probability of firing a "one".

However, life is not so simple. A simple example will show why this approach will not work, and why its failure is of enormous importance.

probably equal to one minus the autocorrelation of $N_{i,t}$. Whatever its other disadvantages, a decay system would allow small neurons to operate at their natural frequencies, without direct control, and to establish recursive connections which <u>are not</u> delayed by a large time-cycle (i.e. from "t-1" to "t", which equals many <u>neuron</u> time-cycles).

The small stellate neurons of the cerebral cortex (Layer IV) seem to be a clear case of cells operating on a decay system; they have multiple recurrent interconnections, under no external pulsing control, operating at high frequency. The large pyramidal cells of all three cortices - cerebral, cerebellar and limbic - could be operating, in theory, by a limited memory system, since all of them are under the direct control of some pulsing system. (e.g. nonspecific thalamic fibers to apical dendrites of pyramids in visual cortex, observed by Scheibel and Scheibel, Rockefeller <u>Neurological Sciences Study Program</u>, vol.ii.) <u>Both</u> feedback systems will be programmed in FORTRAN - plus perhaps a more powerful backwards-memory system - in the next phase of my thesis work.

The limited memory system, when used in a <u>hybrid</u> emotional and reality-constructor system, may be what the statisticians call "inefficient but exact"; in other words, it may force our circuits ~~thxxxxxxx~~ to converge to the right configuration, but at less than the maximum possible speed. It may force the cortex to construct an active memory, $\vec{Z}_t$, which records everything about the external world of relevance to the system's decision-making. If such a system has a set of decay-based circuits attached to it - like the stellate cells - these cells will not corrupt the higher system, but will merely provide it with a larger set of data to work with.

(d) Negentropic Pattern-Recognizers

My original goal was to conjure up $\hat{Y}_t$ according to their actual probabilities. The systems above do not do this; they are deterministic. Also, we have little reason to believe that these systems will help <u>extract out</u> the variables of ~~xx~~ greatest interest to higher-order systems.

At first, these points may seem academic. Suppose, for simplicity, that we presume our system gets raw input in the form of ones and zeroes. (This is still consistent with our assumption that intermediate signals may be continuous.) Then for each $Y_{t,i}$, our pattern-recognizers would still produce a $\hat{Y}_{t,i}$ which varies from zero to one; we could interpret $\hat{Y}_{t,i}$ as a probability, and plug into our pattern-recognizer any error function which encourages $\hat{Y}_{t,i}$ to equal the probability of $Y_{t,i}$ being "one". (To minimize $E((Y_{t,i}-\hat{Y}_{t,i})^2)$, for example, we set $\hat{Y}_{t,i}=p(Y_{t,i})$. A question of minimizing $p(1-\hat{Y})^2+(1-p)\hat{Y}^2$.) Then, to conjure up <u>possible</u> $Y_{t,i}$, we could simply "throw the dice" for each i, based upon $\hat{Y}_{t,i}$ as the probability of firing a "one".

However, life is not so simple. A simple example will show why this approach will not work, and why its failure is of enormous importance.

Suppose that our visual field, for $\vec{Y}$, consists of a simple 2 X 4 visual grid. Suppose that the visual field takes on only two configurations, in practice. In configuration A, there is a black square (1's) in the top 2 X 2 part of the field, and a white square (0's) in the bottom. In configuration B, the white square is on top and the black square on the bottom. At each time, t, the field takes on a new configurayion, A or B, with equal probability and with no regard for either $\vec{X}$ data or for its own previous configuration. Using the pattern-recognizers described above, every cell in the grid will be given a $\hat{Y}_{t+1}$ of 50%, at all times. If we "throw the dice" for each of these cells separately, the odds will be more than a hundred to one against our predicting either configuration A or configuration B. (Probability equals $(\frac{1}{2})^8+(\frac{1}{2})^8$~~=[illegible]~~ 1/128.) Clearly, this point is more than academic; the main task of human vision is to select out objects which are stable in themselves, but which change somewhat erratically in their positions in our visual fields. Even if we could make a deterministic prediction of $\vec{Y}_t$ based on all the data implicit in $\vec{X}_t$, it would be extremely difficult for a system to learn how to make such a prediction unless it could make full use of an intermediate set of predictor features, which are only enough to offer stochastic predictions; in other words, the deterministic pattern-recognizers above may find it difficult to find a pathway from their initial ignorance to their final perfection, because they may be unable to operate effectively (and develop) in the intermediate zone of probabilistic knowledge. This effect may explain the weakness of all adaptive perceptron devices to date.

~~Sax~~

So: how do we account for this effect?

Let us go back for a moment to the idea of "minimizing an error function", which we took for granted above. We didn't specify which error function to use, or why we should use such an approach. In classical statistics, people have found good reason to justify the choice of mean square error as an error function: minimizing the sum of squares through time turns out to be the same, in the limit, as maximizing the likelihood of one's model being true, for linear Gaussian processes. However, we are not dealing with Gaussian processes here. The nonlinearity of our processes is so pervasive that we are better off thinking of Y as pure information, codeable into zeroes and ones.

So: how would we apply the maximum likelihood method to a simple (Markhovian) series of zeroes and ones?

Suppose that we have a model, $p(y_t)=f(t,a)$. ("t", in this case, could stand for the effect of all kinds of extra data. However, we are interested only in how to adapt the coefficient "a"; in other words, we wish to find the value for "a" such that the model has a maximum probability of being true, given that the model is true for some "a".)

Suppose that we have a series of data, y_t, from time zero to time T. Then:

$$p(a\left|\{y_t,\ t=0,T\}\right.) = p(\{y_t,\ t=0,T\}\backslash a)\frac{p(a)}{p(\{y_t,\ t=0,T\})}$$

$$= \frac{p(a)}{p(\{y_t,\ t=0,T\})} \prod_{t=0}^{T} (y_t f(t,a)+(1-y_t)(1-f(t,a)))$$

The term in the denominator of the left-hand term is the same for all "a"; therefore, in choosing "a" to maximize the likelihood of our model, we can ignore this term. P(a) is really a distribution, representing the apriori probability of any model. In general, in statistics, one ignores this term, since its relative effect becomes negligible in time, and since there is no philosophical reason to give a higher apriori probability to one "a" or another, except to a=0. (Occam's Razor.) In choosing the best nonzero value for "a", we can ignore p(a) (formally, set it to da), and then later send a garbage collection routine back to see if we have found reason to set "a" significantly different from zero; if not, we can delete the term in f which has "a" as its coefficient, and try another term to replace it, more or less at random. So, for now, we can concentrate on finding the best nonzero value for "a", which means maximizing:

$$\prod_{t=0}^{T} (y_t f(t,a) + (1-y_t)(1-f(t,a))$$

We want to find an error function, $e(y_t,t)$, such that minimizing the sum of "e" through time is equivalent to maximizing the product above. Clearly, we have to take (minus) the logarithm of the term above to achieve the equivalence:

$$e(x_t,t) = -\log(y_t f(t,a) + (1-y_t)(1-f(t,a)))$$

$$= -y_t \log f(t,a) \ - (1-y_t)\log(1-f(t,a)) \text{ (as } y_t = 0 \text{ or } 1)$$

So: the error function is simply a joint entropy function! To optimize our predictions, we simply minimize the entropy of our errors. More precisely, we are trying to minimize the extra information content required, above and beyond our predictions, to give the true value of y_t through time.

Now let's go back to our difficulties with simulation, above. Our problem, essentially, was that the probabilities of the different $Y_{t,i}$, even given $\vec{X}_t$, were interdependent; the determinations of different components of $\vec{Y}_t$ were not statistically independent events. In other words, the information content (entropy) of $\vec{Y}_t$ as a whole, given $\vec{X}_t$, may be much less than the sum of the entropies of its parts (the $Y_{t,i}$). (This is the essence of the "gestalt" idea.)

In order to simulate $\vec{Y}_t$, given $\vec{X}_t$, economic considerations dictate that the brain simulate large numbers of variables all at once, not one after the other. The only way to do this is by recoding our uncertainties about $\vec{Y}_t$, given $\vec{X}_t$, into a set of independent variables, which we can deal with separately, all at once, in simulating $\vec{Y}_t$. In short, we wish to find a technique to encode $\vec{Y}_t$ (given $\vec{X}_t$) into a new vector, $\vec{R}_t$, of zeroes and ones, whose entropy does equal the sum of the entropies of its parts. We wish to make sure we can recode $\vec{R}_t$ and $\vec{X}_t$ back into $\vec{Y}_t$, so that we can simulate $\vec{Y}_t$ by simulating all the components of $\vec{R}_t$ in parallel and recoding. The components of $\vec{R}_t$, unlike the elements of our simpler pattern-recognizers, would tend to represent unusual joint events in $\vec{Y}_t$ and $\vec{X}_t$; they would tend to represent the unusual patterns and events which disrupt the more predictable monotony of human vision. In practice, this is exactly what Lettvin and Hubel have found in the higher visual centers of mammals and frogs - edge detectors, detectors of vertices as special angles, and even "newness" detectors which respond to moving objects which change their direction or velocity. A pattern-recognizer built on these principles will automatically select out the most interesting features of its environment, for study by higher-up systems in the brain. They will also select out interesting patterns of action initiated by the intelligent system itself; this would form the foundation on which "planning" is learned.

Now: the technical details.

If we give ourselves a fairly free hand in selecting $\vec{R}_t$ as a function of $\vec{Y}_t$ and $\vec{X}_t$, we cannot be sure of finding a straightforward way of recoding it back into $\vec{Y}_t$. In general, we would have to use pattern-recognition to try to predict $\vec{Y}_t$ back from $\vec{R}_t$ and $\vec{X}_t$. Suppose that we define the "gross $\vec{R}_t$" to be the combination of our original $\vec{R}_t$, plus the information required to give us $\vec{Y}_t$ back, item by item, from the proposed $\vec{Y}_t$ we got back from recoding. Given all the information in "gross $\vec{R}_t$", we can get back $\vec{Y}_t$ exactly, simply by recoding $\vec{R}_t$ into an estimated $\vec{Y}_t$ and then correcting the result. Therefore, gross $\vec{R}_t$ cannot have less information content than does $\vec{Y}_t$ itself, given $\vec{X}_t$. Therefore, the minimum of the sum of the entropy of the parts of gross $\vec{R}_t$ cannot be less than the entropy of $\vec{Y}_t$ given $\vec{X}_t$. We can bring this sum of entropies down to this lowest possible minimum by converging on a simple $\vec{R}_t$ with the properties called for earlier: i.e. the sum of the entropy of the components of $\vec{R}_t$ proper equals the entropy of $\vec{Y}_t$ given $\vec{X}_t$, while the recoding process involves no error or entropy at all. Therefore, by setting up a circuit which minimizes the sum of the entropies of the components of gross $\vec{R}_t$, we can get what we wanted in the first place. Admittedly, we could reach the same minimum in another way: if the errors of recoding are all independent of each other, and independent of the components of $\vec{R}_t$ proper, and if the components of $\vec{R}_t$ proper are all independent. (Otherwise, the sum of the entropy of the parts would be greater than the entropy of the whole, which in turn is greater than or equal to the minimal entropy-level we are trying to reach for the parts.) However, in this case, gross $\vec{R}_t$ fits all our original criteria for $\vec{R}_t$ proper.

So: we can set up a network with two levels:

$$\vec{R}_t = \vec{f}(\vec{Y}_t, \vec{X}_t)$$
$$\vec{Y}_t = \vec{g}(\vec{R}_t, \vec{X}_t)$$

to be adapted like our simple pattern-recognizers, but with feedback coming in from each component of $\vec{R}$, and from the error of each component of $\vec{Y}_t$. Notive that we are worried about the entropy of $\vec{R}$ controlled for $\vec{X}$; given that we are worried about keeping the components of $\vec{R}$ independent of each other, and not about keeping them independent of $\vec{X}$, we can get away with measuring the entropy of $\vec{R}_{t,i}$ controlled for some function of $\vec{X}_t$. In short, we can try to "predict" $\vec{R}_{t,i}$ from information abut $\vec{X}_t$, and count in just the entropy of the error. (This is unnecessary, formally, but it makes it much easier for us to bring in recursive data, if such data is generated by giant pyramid cells in the cerebral cortex.) We can even set up several layers of $R_{t,i}$; each layer could use the actual value of $R_{t,i}$'s from previous layers, for use in predicting its own $R_{t,j}$'s.

As with our simple pattern-recognizers, we can avoid a formal layering structure by using a pulsing system and time to define the layers. We can define special units, called Pyramids, to care for one component of $R_{t,i}$ each. In the bottom part of each Pyramid, there would be a network to define the actual value of $R_{t,i}$; the bottom part would have access to all information in the system from $\vec{Y}_t$ and $\vec{X}_t$, both. In the top part of the Pyramid would be a circuit to predict the value of $R_{t,i}$; the top part would have access only to information from $\vec{X}_t$ (including $\vec{Z}_{t-1}$), and from Pyramids which have already released the actual value of their $R_{t,i}$. (i.e. effectively, lower-level Pyramids.) At a certain time, dependent on the state of its inputs (or on the time since the beginning of the alpha cycle, or, more likely, on a signal from the nonspecific thalamus for those pyramids which receive such a signal), the Pyramid would carry out two actions at once. It would compare its prediction of $R_{t,i}$ against the actual value, and store the entropy of the error, plus other details, for distribution in the feedback part of the cycle; it would, at the same time, reduce its $R_{t,i}$, for use by any other Pyramid in the system. Pyramids and the simpler kinds of neurons discussed above would be mixed together in a common network, with the same kinds of timing possibikies discussed above.

When the feedback cycle gets back to our Pyramid, the Pyramid (like any neuron) will receive one significant feedback - the total derivative of entropy elsewhere with respect to $R_{t,i}$; the Pyramid itself will have generated $R_{t,i}$ from an interñal probability, q, generated by a linear circuit in the bottom of the Pyramid, and will also have retained p, the estimated probability of $R_{t,i}$ according to the top of the Pyramid. It is unpleasant to evaluate $R_{t,i}$, a zero/one variable, by means of a derivative; however, there is no choice here, and we can console ourselves with the thought that the circuits which decide how to use $R_{t,i}$ are more formally exact. In any case, each input to the top, $\vec{X}_j$, will receive a feedback:

$$\frac{\partial p}{\partial X_j} \cdot \frac{\partial E(p,q)}{\partial p} ,$$

where E(p,q) is the entropy of observing a probability q when you had been expecting a probability p:

$$E(p,q) = - \frac{q}{p} \log \frac{q}{p} - \frac{(1-q)}{(1-p)} \log \frac{(1-q)}{(1-p)}$$

(A crude system could look directly at $E(p,R_{t,i})$, but that would be a waste of information.)
Each input to the bottom, Y_j, will receive:

$$\frac{\partial q}{\partial Y_j} \cdot \left(\frac{\partial E(p,q)}{\partial q} + \frac{\partial E^*}{\partial R_{t,i}} \right),$$

where the last term is the feedback input to the Pyramid. (The initial trial of a Pyramid is also a bit tricky; if necessary, one could set it up to duplicate exactly the output of a lower Pyramid, and replace the lower one, if the errors of that lower Pyramid correlate with those of other Pyramids on the same level.)

The neuroanatomical evidence in favor of this idea is very strong; in fact, I found it very difficult to to imagine that a biological brain could contain such circuits until I reviewed the microanatomy of the cortex. The large pyramid cells of the cerebral cortex reach from the top layer of the cortex (Layer I) down almost to the very bottom (Layer VI). In between, each cell has a long shaft, which can develop a graded spike of its own. (Recent work in this area indicates that the spikes coming down the shaft can vary continuously in intensity, while the final spike that comes out of the cell body, at the bottom of the pyramid, is 0/1.) The top of the pyramid, in Layer I, receives input only from fibers in this layer; Szengothai has shown that these fibers are almost entirely "recurrent collaterals", fibers branching up and out from the main axons coming out of the cell bodies of nearby pyramids. Some recent work suggests (Eccles) that there are a few strange cells outputting to the upper layers of cortex, <u>inputting only from pyramid collaterals</u>; these cells would merely be accessories to the pyramid tops in processing data which they are entitled to receive. Direct input from the specific thalamus, by contrast, goes straight to Layer IV, almost all a thick plexus of medium/small cells called Layer IV stellate cells. This input gets to the pyramids by synapses to the cell body of each pyramid, in Layer V. The cerebellum, incidentally, also maintains a distinction between the upper "plia" and the lower plexus; however, the details of that analysis must wait.

One should note that a negentropic pattern-recognizer could also profit from "deep sleep". Normal neuron coefficients, in a pattern-recognizer, have to be adapted together with the other coefficients which they influence. But the top subcircuits which predict the $R_{t,i}$ of a pyramid do not have any direct electronic effect on other cells (except in the simulation, or "dreaming", mode.). A single pyramid could record its most interesting inputs from the daytime, and adapt to them during the night.

(iv) Miscellaneous Aspects of Pattern-Recognizers.

(a) The philosophy of induction.

All of the pattern-recognizers above are set up to improve their performance by "steepest descent", sometimes called "hill-climbing". An adaptive pattern-recognizer clearly should have the ability to "climb" to the top of any "hill" it finds itself on. But most students of artificial intelligence would assert strongly that hill-climbing is not enough. A system needs the ability to <u>leap</u> to a new concept.

I would counter by saying that the systems above are necessary, to create a framework which <u>includes</u> hill-climbing. If one can design a system that "leaps" to new concepts, then presumably we can use that system like the "garbage collector" mentioned on page 14: i.e. to <u>suggest</u> new concepts for predicting $\overline{Y}_t$, circuits which could be made available (on a trial basis) as new inputs to the rest of a standard pattern-recognizer; as in multiple regression, a powerful new predictor variable should quickly accumulate a high coefficient <u>within</u> the adaptive system, and should then be absorbed by the normal system.

One might argue that I have not, in fact, provided such a system for "hill-leaping"; I have proposed, instead, the creation of trial neuron-connections "more or less at random." I would counter, first, that the most powerful "leaps of insight" which humans experience tend to be the result of complex strategies - <u>learned</u> strategies - involving the use of language; the leaps of insight shown by lower mammals are not nearly as impressive, despite the equivalence of their brain structure to ours. Furthermore, I would argue that one cannot tell which models will work until one has tried them.

One might argue that the "garbage collection" system above could exhaust the simpler new circuits available, but would not be able to give enough attention to the more complex explanations. I would counter that one can do no more than try out new ideas according to their probability - before the fact - of being true. In this case, at the most basic level, we have no manageable source of information about which ideas are likely to work <u>except</u> by looking at their simplicity. This system does, definitely, discriminate in favor of simpler models; it does attribute a higher apriori probability to simpler models. (The "truth of a model" could be represented, by the way, as a component of $R_{t,i}$; in effect, a negentropic pattern-recognizer can define its uncertainty well enough for decision-making.)

Students of language and philosophy have long recognized that one can never escape entirely from such apriori biases; Bayes' Law clarifies the matter:

$$p(\text{Theory}|\text{observations}) = \frac{p(\text{observations}|\text{theory})\ p(\text{theory})}{p(\text{observations})}$$

Kant used the term "apriori synthetic" to describe the system of biases which give us "p(theory)". Carnap described a probabilistic approach to the apriori synthetic, based upon Occam's Razor. Prof. Solomonoff, a friend of Minsky, has pointed out the generality of Carnap's procedure, when "simplicity" is defined in terms of an appropriate language. (FORTRAN plus a random number generator would seem to be quite good enough.) After a tedious analysis - which would clutter up this outline far too much - I have come to the conclusion that neuron networks, as above, are about as close to an "appropriate language" as one can hope to create in normal physical circuits; the resulting imperfections can be made up for only by the conscious use, by our system, of learned, sequential languages like English and FORTRAN.(Therefore, we reaffirm precisely one orthodoxy - the philosophical view, post Wittgenstein, that the assumptions and rules governing languages are in large part learned by a process related to trial and error, rather than dredged up ~~from the preverbal~~ directly from the preverbal basis of preverbal induction.)

(b) Chemical Memory and the Adjustment of Pyramid Cells

Hyden, a few years ago, set off a great wave of research by suggesting that certain neurons - especially large pyramid cells - do <u>not</u> obey the usual electronic rules for determining their input/output relations; instead, he suggested that they encode their past experience (interesting pairs of $\vec{X}_t$ and $R_{1,t}$ in our framework), and sift through their past experience to find pairs whose X matches the current electronic input vectors as closely as possible; they then fire according to the $R_{1,t}$ of the closest matching pairs, if any. (This process, extended to allow the simultaneous processing of many molecules inside of and between cells, would even give us a closer approximation to a Solomonoff-type general language.) However, in the domain of classical physics (room temperature), one could not expect that a cell could sift through such a large body of data in time to keep up with the usual electronic processing rates. In areas 17,18 and 19 of the visual cortex, electronic processing rules seem enough to predict the behavior of all cells ever observed. Perhaps such cells operate electronically sometimes, and chemically at others; but in this case how would they know which to do when? There is no anatomical sign of an appropriate signalling system to tell them which to do. A chemical signalling system of a general kind might do the job - but only by instituting gross changes in the behavior of large numbers of cells, as in <u>the onset of dreaming or of deep sleep</u>.If, in normal waking experience, these cells do not behave according to Hyden's hypothesis, then their behavior during sleep ahould also focus on optimizing normal, electronic behavior, as above. So: essentially we reject Hyden's hypothesis.

On the other hand, we did suggest a process for adjusting the tops of pyramid cells in deep sleep, based upon the chemical encoding of memories from the wakeful period; we are left with five possible ways of carrying this through:

(A) Forgetting it.

(B) Encoding those memories which are most "interesting", according to a simple rule for what is "interesting".

Presumably, a memory is "interesting" in proportion to the error-reduction achieved by taking the memory into account; if we assume that the optimal expected change in α_{ij} is roughly equal to $-\theta_{ij}(\partial e/\partial\alpha_{ij})$, as our adjustment formulas stated, then we get:

$$\text{Interest} = \Delta e \approx -\tfrac{1}{2}\sum_{i}\theta_{ij}\left(\frac{\partial e}{\partial\alpha_{ij}}\right)^2 \quad (j = j \text{ of pyramid})$$

A system can set its minimum interest level just high enough to keep its memory stocks full, but not overstocked.

(C) Fusing its memories, by a simple rule. The "distance" of two memories could be defined by:

$$\text{Distance} = \sum_{i} \theta_{ij}(X_{i1} - X_{i2})^2 \quad (\vec{X}_1 \text{ and } \vec{X}_2 \text{ the input vectors})$$

A certain distance range could be set up so that memories below that distance are always fused together. This range would be set to make the number of fused memories just enough to use up most of the memory capacity of our cell. An "intensity" counter would have to be added to each memory molecule, to indicate the number of actual memories contributing to a single fused memory.

(D) A combination of (B) and (C).

(E) An aposteriori coding system for memories. Even with apriori systems, the portion of "$\vec{X}$" encoded could not realistically include all the tens of thousands of inputs to the far top of a typical large pyramid cell; the partially processed inputs which make it to Layers II and III would make better candidates for storage.

However: an aposteriori system for encoding (and fusing) these memories might work much better - if only one could spare the cells to set up such a network. The objective would be to set up a coding circuit, $\vec{H}=\vec{h}(\vec{X},R_1)$, such that $\vec{H}$ is limited to a certain number of component variables, and $\partial e/\partial\alpha_{ij}$ can be reconstructed as accurately as possible from knowledge of $\vec{H}$ and $\vec{\alpha}_{(j)}$ alone. This would require two circuits to be adapted together, as a simple pattern-recognizer:

$$\frac{\partial\hat{e}}{\partial\alpha_{ij}} = L_1(\vec{\alpha}_{(j)},\vec{H})$$

$$\vec{H} = \vec{h}(\vec{X},R_1),\ \vec{H} \text{ of } \underline{\text{fixed}} \text{ dimensionality.}$$

The accuracy of the error estimates would provide the major source of feedback. This system seems unrealistically cumbersome, but the upside-down stellate cells (Layers II&III stellate cells) recently discovered to be very numerous (Eccles, Brain Mechanisms and Consciousness) in the cerebral cortex, have all the necessary properties of a recoding network; they receive data from pyramid bases, and climb up along the dendrites of pyramids in an interweaving pattern of some kind, not fully understood as yet.

Of these possibilities, (D) seems to be the most likely, by far. It helps account for the drive of the organism to get four hours of deep sleep every night, whatever may happen to dreaming (MIT Neurosciences Series); the interesting memories of the day may have to be fused onto existing templates, to make room available for a new day's experience.

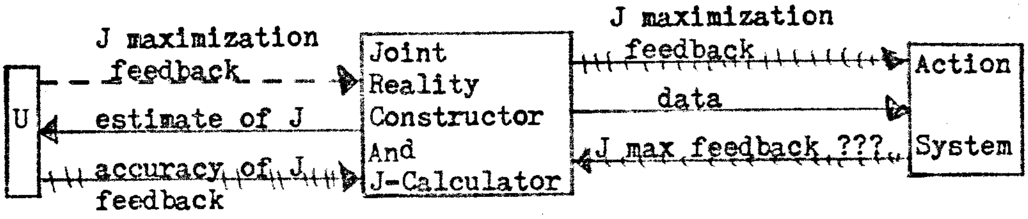
For the sake of completeness, however, I will program all five possibilities.

One should note that these pyramid memories are emphatically not the only possible form of long-term memory available to the cortex, in this model. The alpha coefficients themselves provide a form of memory. Highly interesting experiences could even be translated immediately into the basis of a new pyramid cell (Layer II or III ?), especially if ordinary recursive memories "rehearse" the memory enough; this is one of the principles in designing a "garbage collection" system.

Internuncial cells - small pyramids - in cerebral cortex could be interpreted as Pyramid cells which do not generate ordinary recursive memories, unlike large Pyramids, which generate both. (Small pyramids do not appear to be controlled directly by nonspecific thalamic fibers.) On the other hand, one might imagine that internuncial pyramids are given extra (negative) feedback, or even transferred memory molecules, to encourage them to represent special cases of the large pyramids they output to; this might be of some value in developing a specialized memory and planning system. For completeness, this possibility will also be programmed.

(c) The Irrelevance of Second-Order Feedback Conflicts.

In my chart of the brain, I drew a diagram of a sort which may have seemed quite problematic:



If the same circuit, in the middle, decides what the estimate of J will be, and also helps maximize the estimated J, would this not encourage a kind of dishonesty by this circuit? The naswer is no. The J maximization feedback to the middle circuit comes to it from the lower circuit, to encourage better [illegible] transfer from the middle to the lower (i.e. feedback is fed back along the "data" circuit); the J feedback to the lower system, in turn, comes from a feedback process which assumes that the J-measurement coefficients of the middle circuit will be held constant. In short, the middle circuit is unable to perceive the "value", in J maximization, of cheating by changing its J-measurement coefficients. If the middle circuit received J-maximization feedback directly from U, that might be a different matter. Feedback from a lower system to a higher system may seem artificial here, but it is the only way a lower system can extract out the data it wants from a distant higher system.

Our chart of the brain stem's role appears even worse; the lower system receives J-accuracy feedback and J-measurement feedback from the same source. However, the hypothalamus does have access to more objective sources of data, to replace the brain stem data in cases where the latter may grow biased.

VI. Computer Programs Planned

(0) General Plans.

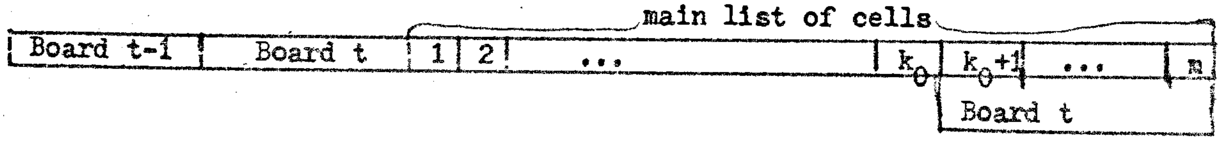
My goal in this thesis will not be to simulate all the minor details of the human brain. My goal will be to show that the general concepts of "reality constructor" and "emotional system" defined above are powerful enough, given the pattern recognition systems above, to allow us to duplicate the power of the human brain in solving certain problems that involve "higher intelligence." The game of chess - which requires both the ability to perceive complex patterns and to formulate complex strategies - would be an ideal test. Chess-playing programs have been written before, but few of them adaptive, and all of them somewhat weak in their play. All of these programs have routines to look through the tree of legal moves, and to min-max; the problem comes when the programs try to evaluate the configurations that come at the end of each tree of moves. The programs do not seem to have the "feel" that a good chess-player has. My "emotional system" is designed to provide such a "feel".

The "J" function which comes from my "emotional systems" should provide the powerful evaluation-system that the present programs lack. My first goal, then, will be to get hold of an existing FORTRAN chess-playing program, and to show that the "emotional system" which I will program can improve the performance of the existing programs. In theory, an "emotional system" with enough time and space available should be able to supplant all of the foresight functions of the present programs; I may try this experiment, too, though a negative result would not be so conclusive, given the normal limits on computer time and space.

When these experiments are complete, I also intend to carry out experiments oriented towards pattern-recognition proper. Uhr's handwriting samples offer one experimental check against previous programs, though I may try out something else, involving more complex sequences of patterns, if possible. The prediction and composition (simulation) of music would offer an ideal test, if only there were other programs available for an objective comparison. At any rate, once the basic "emotional system" and "reality constructor" are programmed, they can be applied immediately to a large number of problem areas.

(1) FORTRAN Subroutine #1: "PATT1": A Simple, Nonrecursive Negentropic Pattern-Recognizer for a Computer.

In the game of chess, the need for a negentropic "reality constructor" becomes very great; the positions of the pieces are highly interdependent, and we have very good reasons to try to extract out information about the most interesting configurations. Oddly enough, there isn't much need for this reality constructor to contain a memory of previous states of the chessboard; almost all the important information is there, sitting on the board. So let us begin by designing a simple negentropic pattern recognizer, designed to "predict" the state of the chessboard given the previous state of the board.



The memory set-up of this subroutine will be as in the chart above. There will be an ordered list of cells, from cell #1 to cell #m. Some of these cells will be "pyramids", others "stellates". Each stellate cell will have a list of input sources, which may include the final outputs of <u>any</u> cells earlier in the main list, and any data about the chessboard at times t or t-1. Associated with each input will be a coefficient, alpha. Each pyramid cell will have <u>two</u> lists of input sources, a "top" list and a "bottom" list. The bottom list may include any earlier data, as with the stellate cells; also, it will include a coefficient for each input. The top list may include <u>only</u> data from the chessboard at time t-1, and data from the output of previous pyramid cells; this list, too, includes a coefficient for each input. The cells from k_0+1 to m will all be pyramid cells, but their "bottoms" will be prescribed in advance; the bottom lists here will each include one item - a component of the state of the board at time t - to be input with a coefficient of one.

At the completion of each move by its opponent, our intelligent system will immediately call in PATT1 to analyze what has happened. It will update the record, above, of the state of the board at times "t-1" and "t". Then it will go through the "electronic cycle" of the system; more precisely, it will call on a subroutine, ELEC1. ELEC1 will proceed, from cell #1 to cell #m, to calculate the output of each cell. When it gets to a stellate cell, #i, it will first calculate W_i, the sum of each input source multiplied by the coefficient attached to it; then it will calculate its own output, N_i, from W_i, from the simple relation graphed on page 8. When ELEC1 gets to a pyramid cell, it will use the same procedure to determine the output of the top of the cell and the output of the bottom; then, however, it will call on a random number generator to generate a number in the interval zero to one, with a flat probability distribution. If the random number is less than the output of the bottom of the cell, then the output of the cell as a whole will be set to "one"; otherwise, it will be set to zero.

After ELEC1 has finished with cell #m, then PATT1 will call on FEED1, the "chemical feedback" subroutine. FEED1 will first initialize the "feedback level" at every stellate cell, at every pyramid top and at every pyramid bottom to zero. It will then work its way back from cell #m to cell #1 to determine the feedback levels. When it reaches a pyramid, it will set the feedback level at the top to:

$$\frac{\partial E(p,q)}{\partial p} ,$$

Where E(p,q) is the function on page 17, with p the output of the top of the pyramid and q the output of the bottom. (If this term gets too high, we can put in a cutoff point. This cutoff point will be one of the basic parameters of the system; in effect, it tells us to restrict p to an interval slightly narrower than zero to one.) If the pyramid is one of the cells between #k_0+1 and #m, FEED1 will not give any feedback to the bottom of the cell, because there is nothing there to adapt. Otherwise, FEED1 will increase the feedback level at the bottom by:

$$\frac{\partial E(p,q)}{\partial q} .$$

<u>Then</u> FEED1 will actually feed back, by the formulas on page 8. First, it will look at the top of the pyramid. If the W_i used to generate p does not itself equal p, then there should be no feedback from the top of the pyramid, in theory; FEED1 should move on to the bottom. In practice, it would probably be convenient for us to have FEED1 continue anyway. For every cell which provides an input source to the top of the pyramid, the feedback level of that cell (of its bottom, if a pyramid) will be increased by an amount equal to the feedback level of the top of our pyramid multiplied by the input coefficient, alpha, of that cell to our pyramid. Then FEED1 will look at the bottom of our pyramid. If the W_i used to generate q does not equal q, then FEED1 will be finished with this pyramid, and move on to the cell just before it. If W_i equals q, then, for every cell which provides an input source to the bottom of our pyramid, the feedback level of that cell will be increased by an amount equal to the feedback level of the bottom of our pyramid, multiplied by the input coefficient, alpha, of that cell to our pyramid.

Those are the procedures used by FEED1 when it encounters a pyramid cell on its path from cell #m to cell #i. When FEED1 hits a stellate cell, the procedures are simpler. FEED1 will assume that the feedback level of the cell is already known. If the W_i which generated the output, N_i, of the stellate cell, does not equal N_i, FEED1 will move on to the next cell, #i-1. Otherwise, every cell which provides an input source to the stellate cell will have its feedback level increased by an amount equal to the feedback level of our stellate cell, multiplied by the input coefficient, alpha, of that input cell to our stellate cell.

After FEED1 has gotten through with cell #1, PATT1 takes over again. PATT1 will now call in a subroutine to adjust all the coefficients, alpha. I will write three different subroutines, ADJ1, ADJ2 and ADJ3, which can adjust these coefficients. Our present memory setup requires, for each cell #i, two matching lists - a list of input sources, and an alpha for each source. (All of this twice for pyramids, of course.) To use ADJ1, we will require two more lists to be matched to the present two - a list called theta, and a list called sign1. ADJ2 requires three new lists - theta, sign1 and sign2. ADJ3 requires four - theta, sign1, k, sign2.

When any of the ADJ subroutines is called, it will scan systematically through all the coefficients, alpha in our system; the order of scan doesn't matter, but it will be easiest to start with cell #1, to scan through its list(s) in order, and so on up to cell #m. For each stellate cell, pyramid top or pyramid bottom, ADJ will first make sure that W_i equals the actual output produced; as before, if this condition is not met, ADJ will simply pass on to the next list. If it is met, the feedback to the coefficient will be set equal to the product of the feedback level of the cell (botton/top)whose input list it is on, multiplied by the output of the input source which the coefficient is matched to.(This may include cells <u>or</u> board data.) With ADJ1, if the sign of this feedback level equals sign1, then theta will be multiplied by k, a basic parameter supplied to ADJ1; if the sign is different, theta will be divided by k; if either is zero, then theta will be held constant. Then sign1 will be replaced by the sign of the current feedback level to alpha. With ADJ2 and ADJ3, theta will be adjusted in the same way, except that k would be taken from the list "k", in ADJ3, or from a global variable called "k", in ADJ2. With ADJ2 and ADJ3, the sign of this change in theta would then be compared with sign2. If the signs were the same, k would be multiplied by k_2, a basic parameter supplied to both ADJ2 and ADJ3; if they were different, k would be divided; otherwise, it would stay the same. (For ADJ2, one would expect a much smaller k_2 than for ADJ3.) Then sign2 would be replaced by the sign of the current change in theta. Finally, with any of the ADJ routines, alpha will be increased by theta times the current feedback level to alpha. And that will be that.

Every once in a while, PATT1 will call on another subroutine, "GARBAG", to carry out the "garbage-collection" process described above. GARBAG will have to be rather ad hoc; there are no rigorous rules for it to follow. It would include two subroutines, AXONG and CELLG.

AXONG would be given a parameter, A, from the outside, indicating the desirable turnover rate of axons per call on AXONG. It would maintain a global variable, "T", for "interest threshold", which it would increase after every call in which it kept too many connections intact, and decrease after every call in which it broke too many connections. For each connection, it would look at the absolute value of the ratio of alpha over theta; if this ratio is less than T, AXONG will erase the connection, and generate a new one by an ad hoc sort of routine. (A more sophisticated form of AXONG might require an increase in the number of memory lists.) AXONG would be given a list of ad hoc parameters from the outside - probability that it will create a new input to the same cell, probability that it will add a connection to another cell chosen at random, probability that it will wander down the input network of the cell which has just lost one input, etc.

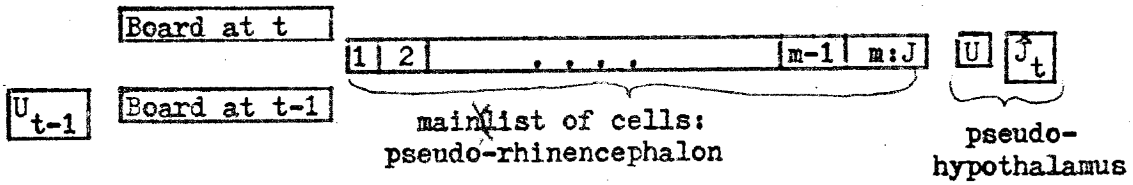
CELLG will be similar. To begin with, we could look at the sum of the interest ratios of the connections <u>from</u> our cell <u>to</u> other cells, and use a threshold "T" to decide when to erase a cell.

The garbage-collection routine, which sets up the pattern of connections, could be used to enforce a "layering" pattern closer to the realities of the human brain; each "layer" would constitute an interval of cells, each compelled to input data only <u>before</u> the interval it belongs to. This option, and the parameters required to set it up, will also be programmed into GARBAG.

The pattern-recognizer above, like the cerebral cortex, only learns to perform efficiently after it has enough experience. Therefore, the pattern-recognizer used for chess will have to be given much experience in playing chess against the standard FORTRAN player mentioned above; it would also profit by sifting through "book games", especially insofar as these games have been recorded somewhere in machine-readable form. (I might wind up encoding such games myself, if not.) The chessboard will be encoded in binary form, row by row, by using four data bits to encode each of the 13 configurations possible to each chess-board square (empty, white queen, etc.); two more bits may be added to indicate the occurrence of check. Other details will depend upon the chess-playing program which this system will interface with.

(ii) FORTRAN Subroutine #2: EMOT1: A Simple Emotional System for a Computer

The "value" of a chess-board configuration depends almost entirely on the present state of the board; therefore, a simple, nonrecursive pattern-recognizer should be good enough to give us a decent "emotional system", as on page 3, to evaluate the configuration. A "negentropic" pattern-recognizer only adds something when there are several variables to predict (and take advantage of thereby). Our emotional system predicts only one - J; therefore, a simple pattern-recognizer is good enough. Note that our system is designed to play a continuous series of games, one after another.



The memory set-up for EMOT1 will be as above. There will be a list of m cells, all stellate; each cell will have a list of input sources, including previous cells and data from the chess-board. In this case, however, the cells are only free to specify input data from the "current" configuration of the board, which may turn out to refer to either time, t or t-1. With each input source, there will be a coefficient alpha. There will also be a "hypothalamus cell", labelled U above, and a storage location called $\hat{J}_t$. U_{t-1} - the intrinsic utility of the state of the board at time t-1, must also be input.

The critical subroutine, EMOT1, is the subroutine which adapts the circuit to give a good estimate of J. EMOT1, first of all, "plugs in" the data from time t to the stellate network. In other words, it calls on ELEC1 - the same subroutine we used with PATT1 - to work out the output of our stellate cells, from #1 to #m, using data from time t whenever the stellate cells call for data from the chess-board. (After all, a network of stellate cells only is just a special case of the networks described above.) Then control returns to EMOT1, which interprets the output from cell #m as an estimate of J_t; it stores this estimate in the storage location marked $\hat{J}_t$. Then it "plugs in" the data from time t-1; it calls on ELEC1 again to recalculate the output of the stellate cells, this time using data from t-1 every time that the stellate cells call for chess-board data. The feedback level of all the stellate cells will then be set to zero, except for cell #m and the hypothalamus cell, whose feedback level will be set to $\hat{J}_t+U_{t-1}-\bar{U}-\hat{J}_{t-1}$. ($\hat{J}_{t-1}$, of course, equals the actual output of cell #m, after our second use of ELEC1; $\bar{U}$ comes from the hypothalamus cell. This feedback formula corresponds to the derivative of mean square error in our J estimate; the continuous variation of J makes this the appropriate error function here.) FEED1 - the same feedback routine as before - can be used to feed back from stellate cell #m on down to cell #1. ADJ can be used, exactly as before, to adjust all the coefficients alpha in the stellate network. If we treat $\bar{U}$ as if it were a coefficient alpha, with its overall feedback level used as the feedback to the coefficient, we can adjust $\bar{U}$ in exactly the same way. And we can call on GARBAG every once in a while here, too. And that's all there is to it.

It should be very interesting to psychologists that $\tilde{U}$, the expected "normal" level of satisfaction in life, and the θ of $\bar{U}$, the readiness to change this expectation, should seem so fundamental to the machinery of the brain. Also of interest: this system is more ~~system is~~ likely to learn to play chess if winning generates a U of +1, if a draw generates a $+\frac{1}{2}$, and if all other conditions produce a U of zero for the relevant move; negative reinforcement is likely to make the system try to avoid playing the game at all, if possible, or at least to avoid experimenting with interesting situations. (Different reinforcement patterns will be tried, but the positive reinforcement strategy gives much greater assurance of good results.)

(iii) Interface of the Emotional Sysstem and the Reality Constructor.

The emotional system above gives us the estimates of J that we need for our chess-playing program. But what about the reality constructor? Reality constructors, in general, have only two ways of making a contribution:(i) by making new variables available to other systems, especially to the emotional system;(ii) by allowing the simulation of what happens in hypothetical situations, as in "dreaming." In the case of chess-playing computers, we have little reason to institute a "dreaming" subroutine; we can have the entire system play ~~itself~~ its mirror image in a game of chess, just as easily, to adapt J. So: our goal is to make sure that the variables extracted by our reality constructor are actually used by the emotional system.

The simplest form of interface which will work is the one suggested on page 3. Whenevrr a new situation occurs on the board, at time T, we can call on PATT1 to go through the process defined above. Then we can define the "greater" chessboard to include the state of the chessboard proper at time T, and also the state of the normal electronic outputs of all the cells (or at least the pyramids) in the network of cells which PATT1 operates on - i.e. the outputs generated when PATT1 processes the data from times T-1 and T. The emotional system will process the state of the "greater" chessboard at times T and T-1, by the rules of the section above. (When J is evaluated over a hypothetical situation, by the normal chess-playing program, the ~~[illegible]~~ normal program will have also suggested the series of plays leading up to that situation; thus, the state of the "greater chessboard" in such situations could be calculated very easily.)

A subtler, and better, interface will be one analogous to the one used by the human brain. The J system will still operate over the "greater chessboard", but it will also feed back to the reality constructor, to change the definition of the greater chessboard. In effect, we will replace our pure reality constructor with a hybrid system. To adapt such a system, in a computer, we can: (i) Store the state of the chessboard proper at time t-2, in a special storage site, while registering t-1 and t in the normal storage sites belonging to the reality constructor, as above;(ii) Call ELEC1 to calculate the output of the reality-constructor network;(iii) From the outputs of the reality construction network, and the state of the chessboard proper at time t as stored in the reality constructor, plug in the "state of the greater chessboard at time t" into the stellate network of the emotional system; (iv) Call ELEC1 in the emotional system, to calculate $\hat{J}_t$; (v) Transfer the data about the chessboard ~~frm~~ proper at times t-2 and t-1 to the "t-1" and "t" storage areas of the reality constructor;

(vi) Call ELEC1 in the reality constructor; (vii) Plug in the state of the greater chessboard at t-1 to the stellate network of the emotional system; (viii) Call ELEC1 in the emotional system; (ix) The first critical step: call FEED1 as before in the emotional system, but with one critical difference - this time, calculate the feedback level all the way back to the variables describing the greater chessboard (the mathematics is precisely the same as before, except that we have to add more storage room to hold the new feedback information); (x) Adjust the emotional system as before; (xi) Clear the feedback levels of the reality constructor to zero as before; (xii) The second critical step: take the feedback levels of variables in the greater chessboard, as calculated in the emotional system, multiply them by a new global parameter ("epsilon") and then <u>add</u> them to the feedback levels of the output of the corresponding cells in the reality constructor; (xiii) <u>Now</u> apply FEED1 and ADJ to the reality constructor, as before. Variation in inborn parameters like epsilon from person to person may be one of the reasons for persistent differences in cognitive style. In the limit, as epsilon goea to zero, we come up with a system exactly like our original chart on page 3, with a distinct reality constructor.

(iv) A Note on More Sophisticated Circuits

In the above, I promised to program many different types of system eventually. For the most part, these new types can be built fairly easily, by extending the basic systems above.

For example, a negentropic pattern-recognizer can be made recursive, by the "limited memory" timing system. One could add a set of "Giant Pyramid" cells to the cell network of the pattern-recognizer above; they would number m+1 through n. Each giant pyramid cell would have two input lists, just like the standard pyramid cells. <u>However</u>: attached to each input source and alpha coefficient would be another storage location, for storing the most recent value of the output of the input source. All cells, prior or not in our ordering system, would be allowed to cite the giant pyramids as an input source. PATT1 would start out just as before, with ELEC1 calculating the output of cells #1 through #m. Then the feedback levels would all be set to zero, as before, and FEED1 would feed back as before, starting with cell #m; this time, however, it will feed back to the giant pyramids, as a natural result of these cells' role as an input to normal cells. Then ADJ will be applied exactly as before, to all cells, <u>including</u> giant pyramids; however, with the giant pyramids, ADJ will find the "output of the input source" term it needs, before calculating the feedback level to any coefficient, by looking at the "most recent output" list right in the giant pyramid. Then - <u>after</u> the chemical feedback cycle - the electronic cycle of the giant pyramids will be carried through by a new subroutine, ELEC2. ELEC2 will go through each pyramid, one by one, from #m+1 to #n. For each item on each input list, it will look back and write in the most recent output level of that input source; then it will compute p,q and Q for the pyramid, as in any pyramid cell, based on those output levels. And that would be that. Recursive memory may be unnecessary for a chess-playing system, but recursive processing may still be very helpful there because of its analytic power.

"Deep sleep routines", as on page 19, can be set up by adding long lists of "encoded experience" to each pyramid cell, and by tossing in a "storage" subroutine to look at the current contents of every pyramid, one by one, and put those into storage which are "interesting" enough; these routines would be called by PATT1 between its calls on FEED1 and ADJ.

And so on. But there is enough basis here already to begin programming and experimentation; further details will be in the Ph.D. thesis itself.