May 26, 1972

An Outline of Thesis Work, Spring 1971 - Spring 1972, Paul J. Werbos

Given that this outline is being prepared in one draft, before the research work is complete, I must apologize in advance for the inevitable loose ends and the imperfect writing style.

I. Objective

Basic objective of the thesis: to contribute to the "theory of intelligence," as defined in my old Cybernetica paper. ("Elements of Intelligence", Cybernetica, #3, 1968.) In that paper, I pointed out a continuum that exists from "primitive" brains to more "advanced" brains, in evolution. Primitive brains tend to work by "innate releasing mechanisms," by inborn stereotyped responses to definite environmental triggers. Advanced brains, however, decide what actions they prefer to take less and less on the basis of any ~~their~~ instinctive preference for one kind of action or another; they tend to decide what actions to take by considering the likely results of these actions, and deciding which results they prefer. As they get more advanced, they also make conscious plans further and further into the future. In the limit, one can imagine a brain broken down into two distinct parts: (i) a "motivation system", which specifies which "results" the system prefers, on a fairly consistent basis; in other words, this system provides a cardinal utility function to the brain, ~~probably~~ ~~providing a measure of the current level of utility~~; (ii) an "intelligence", which has control over all motor systems, and which is built to maximize the long-term expected value of the inborn utility function.

Mankind is not perfectly an "advanced" animal; he has inborn reflexes, and inborn prejudices. The prejudices - the irrational fears and associations that can be unlearned with experience - can be treated analytically as a kind of "ancestral memory", an initial setting of the human memory banks in a configuration different from what most people would consider "blank";

these prejudices form an ~~adjunct~~ adjunct to the theory of motivation, but they do not give us any direct information about the <u>processing rules</u> which the brain uses on its memory banks.

In order to understand human thought, I would argue that we have to try to understand the "intelligence" of the human brain. (Also, we have to look at the motivation system, but not in this thesis.) The concept of "intelligence", as above, is an intellectually manageable idea, whose application to the human brain would really amount to some kind of "understanding" of thought processes. The accumulation of empirical data about the brain and behavior, <u>by itself</u>, cannot be expected to lead to a manageable system of understanding. Lettvin, in criticizing the ideas above, has said that, "The biochemistry of the human brain is really too complex to allow this or any other simple theoretical scheme to work. In fact, it's too complex for us ever to understand it; we can only hope to build up pieces of data here and there. You can't take an anthropomorphic view of the human brain." I would claim that the human brain, if not perfectly intelligent, has a fair amount of generalized intelligence; the function of the brain cannot be understood until that intelligence is analyzed, explained, and related to its biological basis.

Before I go on, I should allay some of the anxieties one might have about the philosophical implications of all this. First of all, an "inborn biological utility function" need not be limited to personal chemical inputs and outputs; such drives as "motherhood" and the "imprinting" of affection have a well-studied biological (i.e. hereditary) basis. One can argue that a pure "intelligence", with an inborn concern for the welfare of its descendants, plus an ability to consider the far future, would look fairly altruistic by normal human standards. Second, there is nothing in this model which says that an <u>actual</u> intelligence will always be able to tell the truth about its own dynamics, or even to find the best possible way to achieve its goals; therefore, this model is not refuted by the wild diversity of value schemes, and the lesser diversity of operational goals, that human beings have displayed in their history. The model does suggest that "intelligent" systems will switch to better ways of achieving their goals, as they develop a better understanding of themselves, their environment and their options; the ulterior motive of this thesis is to contribute a little bit to that historical process.

So: the first goal of this thesis is to explore the theory of "intelligence", the theory of systems which are capable of maximizing the long-term expected value of their utility functions, in new and complex environments. (The first subgoal of this research was to define a bit more precisely what we mean by "new and complex environments," in a generalized sense. ~~In~~ However, in practice, I often tend to reason from the general capabilities which mammals have had to develop over time.) The second goal is to pick out the particular type (or types) of intelligent system which correspond best to present empirical data about the human brain; the resulting model of human intelligence will likely be wrong, in the end, but it will provide a basis for experiments to probe the relevant issues, and for future models to account for the results of such experiments.

All schools of psychology agree that the human brain is capable of pursuing long-term goals, whether of reinforcement or of some other kind, fairly efficiently. From the theory of natural selection, we can deduce that the human brain was evolved to achieve some kind of biological success with maximum possible efficiency through time. Either way, the human brain is capable of finding approximate solutions to the dynamic programming problems implied in all these forms of behavior. From there, to our description of an "emotional system", is a short step.

of the environment it lives in. For simplicity, let us assume that its environment looks like a Markhov chain, a Markhov chain which will change if the organism's action-strategy changes, but a Markhov chain which will always have a unique equilibrium probability vector. In practice, our system ought to be able to handle tougher conditions than these, but it should have *at least* the capabilities that these conditions demand. Our system will face the "open-loop" control problem of finding an optimal strategy covering *every possible* contingency it will face; therefore, *it will have to try to find an approximation to the Bellman/Jacobi function, $J^0(x)$, given its utility function, $u(x)$.* Crudely, this function, $J^0(x)$, represents the *total* worth of the situation, "x", to our organism, the long-term worth, while the inborn utility function, $u(x)$, gives only the short-term or intrinsic value of being in x.

(In practice, the system only needs to choose between a limited number of near-term possibilities at a time; thus it needs to know J^0 only in a given region. However, under the conditions above, the value of J^0 and its derivatives in any region will depend, in general, upon one's strategy and values in all contingencies. *In practice*, a human's choice between present alternatives will often depend upon his preferences between contingencies not likely to arise very soon; for example, he may pick up a pen to write certain words, words he chooses in order to foster the success of projectss which will not bear fruit for many years. Thus, the system needs to look for a general approximation to J^0, not merely to the local derivatives of J^0, in order to carry out the necessary calculations.)

Conversely, a good approximation to J^0 would solve the problem of "planning" for our system; once the system knows J^0, it need only take actions to maximize J^0 in the short-term, a problem which should be relatively straightforward.

So: the system needs some kind of *central organizing principle*, a principle for putting together all of its data from subsystems, and coming up with an approximation to J^0.

If we treat our environment as a Markhov chain, as above, we may write:

$$\vec{X}_{t+1} = M(S)\vec{X}_t$$

"S" is a strategy; "$\vec{X}$" is a probability vector; M is our Markhov chain. The intrinsic utility of being in $\vec{X}$ depends on the probability, X_i, of being in any state "i", times the intrinsic utility, u(i) or simply U_i, of being in that state. Thus:

$$u(\vec{X}) = E(u(i), \text{ given } \vec{X}) = \sum_i pr(i)U_i = \sum_i X_i U_i = \vec{U}\cdot\vec{X}$$

Similarly, we may define $J^0(X) = \vec{J}\cdot\vec{X}$.

Howard has shown that we can write:

$$\vec{J} = \vec{J}M(S^*) + \vec{U} - (\vec{U}\cdot\vec{P})\vec{1},$$

where S* is an optimal strategy, $\vec{P}$ is the equilibrium probability vector for this strategy, and $\vec{1}$ has the component "1" in every state "i". (In words: the long-term value of each state is equal to its intrinsic value, above and beyond the intrinsic value we expect on the average, plus the long-term value to be expected of the situation which will follow it.) He has shown that we can evolve towards an optimal strategy by plugging in a suboptimal S*, solving for the resulting $\vec{J}$, then using that $\vec{J}$ to define a new S* - a bit better - and so on ad infinitum. So : at any one moment in time, our system, with a tentative strategy (S*) in mind, has to be able to calculate the $\vec{J}$ which follows from this ~~strateg~~ strategy, in order to produce constructive modifications to its strategy.

We can solve Howard's equation, formally:

$$\vec{J} = \vec{U}(I-M_0)^{-1} \qquad \text{(Where } M_0 = M-M^{\infty}.\text{)}$$

There are many ways one could try to compute J from these formulas. One could try to compute $(I-M_0)^{-1}$ directly, by row and column operations, but this would be totally impractical; the true M_0, in this case, involves every possible configuration of the system's universe, and we must consider simpler such approximations ~~as~~ only if they come out as good approximations to the true formula. One might expect, in practice, that "M" would be represented by a circuit

giving out sample X_{t+1}, from sample X_t; thus, one might expect that matrix "multiplication" - the application of such circuits in series - would be realistic, while row and column operation~~s~~ would not.
So: we could try a simple iterative technique to get the inverse, the technique on which Howard's formula is ~~effected~~ effectively based:

$$N_{n+1} = M_0 N_n + I \qquad (1)$$

(Roughly speaking, this is the basis of standard dynamic programming, and equivalent to my Cybernetica proposal. Howard's original formula is reminiscent of price theory.) This procedure is not adequate for our purposes. M_0 is supposed to be a model taking us up one minimal unit in time, from t to t+1. Each use of these iterative formulas would take at least one minimum time interval for our system to perform, probably a lot more; yet, with each use, we push up our effective time-horizon only 1 unit of time ahead into the future. If we changed our strategy at any decent rate, we would never be able to look ahead into the future by more than a tiny fraction of our past lifetime.
So: we have to find a much faster technique, that does not involve row and column operations, to invert the overall matrix, $I-M_0$.

Using multiplication, we have one way to build up quickly to what amounts to "distant time intervals." We can square M_0 to get M_0^2, square that to get M_0^4, square that to get M_0^8, and so on; the amount of work required for a given future time-horizon is only proportional to the log of the horizon distance, allowing substantial foresight.
This method can be used to compute an inverse, by storing the high powers of M_0, and then computing:

$$\sum_{n=0}^{2^m} M_0^n = \sum_{n=0}^{2^{m-1}} M_0^n + M_0^{2^{m-1}} \sum_{n=0}^{2^{m-1}} M_0^n \qquad (2)$$

This kind of method, jumping across long time-intervals, is necessary, in one form or another, to any system capable of computing its "J" with enough foresight. Intuitively, ~~that~~ this method requires us to develop models that allow us to jump ahead, to predict the distant

(distant by many seconds) consequences of our present strategy; that's what $M^{2^n}(S)$ represents; given that mammals do appear to have that capability, this idea is quite reasonable. Even in my *Cybernetica* article, I recognized that such capabilities must exist *somewhere* in the brain. The problem, here, is that the capabilities must exist in the *central organizing system* of the brain. Given that one could use many *different* time-intervals of prediction, for *different* purposes, it is easy to imagine subsystems based on long time-intervals; but it is hard to imagine how the central organizing system could rely on *one* specialized system of time-units to weigh the recommendations of these *different* subsystems. The best way to solve this paradox was not, as Mao would put it, by seizing the contradiction, but by ignoring it for the moment.

To solve precisely for $\vec{J}$, in our formula, we would have to consider more than just time economies. $I-M_0$, in practice, would be a very low-density matrix, since one does not expect drastic changes in the state of the world instantaneously. Our problem is to invert a low-density matrix with a minimum number of operations, involving a minimum number of terms. We will have the further problem of finding a natural way to ~~relate~~ represent the inverse, because we could not hope to store such a large, full matrix explicitly. If we can construct the inverse *implicitly*, by imagining several linear circuit layers connected together, instead of one big layer corresponding to the matrix, we can avoid taking the time and trouble to multiply out the submatrices; we will still be able to multiply this system by $\vec{U}$ on the left, as before, and take advantage of the same economies which make Howard's formula superior to equation (1), and thereby cut the costs of an exact calculation down to a minimum. ~~Axximixix~~ The same technique could also be used in input-output economics.

Suppose we wish to invert a matrix which can be decomposed as follows:

$$\left\|\begin{matrix} B & W_1 & W_2 & W_3 \\ A_1 & M_1 & 0 & 0 \\ A_2 & 0 & M_2 & 0 \\ A_3 & 0 & 0 & M_3 \end{matrix}\right\|$$

(A_i and W_i represent the alpha and omega of the system's presence in their region.)

(In other words, let us assume that there are three - really n - groups of states such that a state in one group is never followed directly by a state in another group, but rather by a "transition state" in the central "transition group" of states. Low density matrices which do not have such subgroups cannot, except in special cases, be dealt with economically in this context, for reasons which will become clearer later on.)

By row and column techniques we get the inverse:

$$\left\|\begin{matrix} G & -GW_1M_1^{-1} & -GW_2M_2^{-1} & -GW_3M_3^{-1} \\ -M_1^{-1}A_1G & M_1^{-1}+M_1^{-1}A_1GW_1M_1^{-1} & M_1^{-1}A_1GW_2M_2^{-1} & M_1^{-1}A_1GW_3M_3^{-1} \\ -M_2^{-1}A_2G & M_2^{-1}A_2GW_1M_1^{-1} & M_2^{-1}+M_2^{-1}A_2GW_2M_2^{-1} & M_2^{-1}A_2GW_3M_3^{-1} \\ -M_3^{-1}A_3G & M_3^{-1}A_3GW_1M_1^{-1} & M_3^{-1}A_3GW_2M_2^{-1} & M_3^{-1}+M_3^{-1}A_3GW_3M_3^{-1} \end{matrix}\right\| \qquad (3)$$

$(G = (B - W_1M_1^{-1}A_1 - W_2M_2^{-1}A_2 - W_3M_3^{-1}A_3)^{-1})$

Even in the case of economics, where one usually prefers an explicit inverse, this techniques of computation can save a lot of effort, especially since the "M_i" matrices may be ordered in a simple way, and since there may be a small number of transition industries under some decompositions.

In the case of intelligent systems, this kind of decomposition leads to a set of iterative techniques which can work more effectively through time than that of equation (1), without the troublesome artificiality of (2).

We can calculate, first, the $W_iM_i^{-1}$ by iterative means:

[redacted]

$$(\text{"}W_iM_i^{-1}\text{"})_{n+1} = (\text{"}W_iM_i^{-1}\text{"})_n(I-M_i) + W_i \qquad (4)$$

Intuitively, this represents the matrix of transitions from the i-th region to the transition region; it represents the probability that ~~the first transition from state j in this region~~ a system starting out in state j, in region i, will first escape to the transition region by a transition to transition state k. (The careful reader, comparing this formula to Howard's, will note that our "M_i" are terms in the decomposition of $I-M_0$, not of M_0.) W_i gives us the probability that such a transition will be made in one step, according to our original decomposition, while the other term on the right side of (4) gives us the probability ~~that a transition to k will be made eventually from that state~~ of a transition to another state in region i from which ~~a transition~~ the first escape will be to state k; the logic of this formula is the same as the logic of Howard's.

If we think of the actions to be taken in region "i" as forming a "substrategy", or a "subplan" (Miller, Galanter) or a "schema" (Piaget), then this matrix ($W_iM_i^{-1}$) describes compactly the final result of applying that schema, as a function of the initial state in region "i". Building up from these descriptions, we can calculate G; each step in the iterative calculation of G represents a step of several time units, since each term "$W_iM_i^{-1}A_i$" represents a transition over several time units. (Other low-density matrices, which cannot be decomposed in this way, would allow short-length connections between most pairs of states, making it impossible to achieve a description of longish time-intervals involving few states on one side of a transition matrix.) This decomposition of a strategy (matrix) into a hierarchy of substrategies can clearly be continued on down, with substrategies of substrategies, if the M_i can be decomposed in a way similar to that of the overall matrix. What we wind up with is a central organizing principle based upon a hierarchy of "plans" and "subplans"; this idea fits well with traditional psychological theory, yet, as formulated above, it is free from the ambiguity and determinism of traditional theories.

If we look carefully at this formulation, we can see three ways to broaden it, right away.

First of all, we could try to decompose our state vectors by another simple operation, besides tensor addition as above; there is one other simple binary operation - tensor multiplication. Normally, it would be too much work to decompose a matrix this way, but our individual "states", in this case, each represent a configuration of a large set of input variables; therefore, our state vectors can be treated as tensor products of the smaller state vectors associated with the configurations of subsets of these input variables.
In effect, we can sometimes break the world up into a number of independent, parallel sets of variables; we can simplify our decision problems by treating these sets as separate little worlds, at least in the contingency regions where these little worlds do not interact. So we can decompose our overall schema into <u>sequential</u> subschemas, as before, or into <u>parallel</u> subschemas. (I have also tried a few other plausible decomposition approaches, but, thank goodness, they didn't work.)

Second, we can consider the practical fact that the same "subschema" (e.g. walking) will occur, over and over again, with certain variations, even though the same exact set of world-contingencies will not recur; thus, for practical purposes, the brain can store a large set of subschemas which are "different," formally, as a single parametric family of subschemas, by means of a single ~~parametri~~ circuit allowing variation in its input parameters. (This is similar to parallel decomposition.)If we represent our subschemas in this way, then a given subschema could be used within several <u>different</u> supraschemas; therefore, we will get a lattice structure of organization, instead of a simple hierarchy.

Third, we can go beyond the question of determining $\vec{J}$ for a given "S". Howard's system involved the creation of a new S, S_2, by choosing a new action, at each time t, to maximize J at t+1. In effect, his system of updating strategies has the same defect as his formula for updating J; it advances only one time-unit for each cycle of computation. If we use schema-completions as a new unit of time in calculating J, we can <u>also</u> use schema-choices as a new unit of decision. This also corresponds to our intuitive ideas about planning. This may lead us to an anxiety: the matrix of possible transitions is much more dense than the matrix of specific transitions that we <u>decide</u> to allow under any given efficient strategy; however, Howard's work still allows us to use the specific matrix to get J, <u>even when</u> we are contemplating the use of a new subschema which will change the pattern of transitions. <u>In principle</u>, we could optimize our overall strategy simply by optimizing at the lowest level, as in Howard's original formula; however, if we allow positive changes in our strategy on <u>all</u> levels, we can speed up our approach to the global optimum, without losing precision in our final choice.

Having broadened our formulation, we have to smooth it off into something more practical. Our goal was to find $\vec{U}(I-M_0)^{-1}$, not $(I-M_0)^{-1}$. By looking for $\vec{U}(I-M_0)^{-1}$, we can simplify our calculations further. Looking back at (3), we can see the possibility of calculating $\vec{U}M_i^{-1}$, $W_iM_i^{-1}$, and G in order to eliminate calculations based on a large number of possible states. Intuitively, this means that we can <u>describe</u> each subschema - for use higher-up - in terms of $\vec{U}M_i^{-1}A_i$ (<u>expected</u> total intrinsic to be earned in a subschema, from the moment of our entry) and $W_iM_i^{-1}A_i$ (transition matrix from entries into the subschema to exits from the subschema); these terms, plus B, are enough to let us compute G, and to compute the result of multiplying $\vec{U}$ times the first column of ~~(3)~~ formula (3). This tells us the value of $\vec{J}$ in the critical, transition states, and allows us to choose between possible subschemas. Let us call this portion of $\vec{J}$, $\vec{J}_G$. Inside of each subschema, to allow suboptimization, the $\vec{J}$ consists of two terms - a $\vec{U}M_i^{-1}$ term, the $\vec{J}$ of the region "i" ~~all by itself~~ considered in isolation, plus $\vec{J}_G W_i M_i^{-1}$; in other words, back in (3), the later columns of $(I-M_0)^{-1}$ are the sum of an M_i^{-1} term, in the row corresponding to region "i" itself, plus a lot of other terms which equal the first column multiplied by $W_i M_i^{-1}$. Intuitively, this means that we determine the $J^0$of a state inside a subschema by finding the J^o of each of the states the schema may eventually lead to ($\vec{J}_G$), by weighing these J^0's according to the probability that our given state will lead to any one of the final states ($W_i M_i^{-1}$), and by adding a term to account for the total u we can expect to get before we leave the ~~schema~~ subschema.

Now that we have smoothed out our framework a little, we can broaden it some more. We can come back and account for the old problem of $\vec{U}\cdot\vec{P}$, a term we have formally incorporated into M_0 but effectively ignored. This term represents the average intrinsic utility expected per unit time, in the long-term. In effect, it represents the utility of time. $\vec{U}\cdot\vec{P}$ will have to be calculated somehow by our system; also, our system will have to be able to deal with changing $\vec{U}\cdot\vec{P}$, as it changes its strategy. We can deal with this by treating time as a kind of separate, parallel variable to the other variables in our system. The time implications of each subschema can be summarized adequately by a vector, $\vec{T}$, analogous to $\vec{U}M_i^{-1}$, representing the expected time from a given state in the schema to exit from the schema; given that the cost of taking a certain amount of ~~tm~~ time is independent of the state finally reached, it does not matter how that cost is distributed between different states eventually reached, nor is the probability distribution relevant. Within each schema, we will get a new term in our formula for J:

$$\vec{J} = \vec{U}M_i^{-1} + \vec{J}_G W_i M_i^{-1} + \bar{u}\vec{T},$$

where u is the current estimate of U P.

And that's about where I was back in late October. I figured that the problem of describing a schema (getting $W_i M_i^{-1} A_i$) by means of a definite circuit could be treated as a classical problem in statistical prediction or pattern recognition, based on data partly from experience and partly from simulation. I figured that the same techniques could be used to calculate all the J's in the formulas above, including $\vec{U}M_i^{-1}$, by using Howard's formula (using *different* effective time-units, as indicated above) as the basis for another statistical prediction system. The $W_i M_i^{-1}$ term bothered me a bit, however; the greatest generality comes from assuming that a schema will decompose these final states into *distinct* result categories, but will also allow supraschemas to specify parameters to further describe the variations of $\vec{J}_G$ within each category. (For example, one might "succeed" or "fail", yet still try to achieve the maximum level of success.) In February, I worked on this again, and concluded that this formulation of $W_i M_i^{-1}$ is really the most general, for the purposes of intelligent systems; however, I have not yet been able to find convincing arguments that mammals have a natural ability to deal with distinct result categories. The tricky part in caring for schemas comes in the aggregation of new schemas and in the splitting up of old ones; these issues, I dealt with in more detail in ~~March~~ February and March. Also, I dealt with the question of special expoential learning systems, to find the equilibrium of a Howard-type equation relatively quickly; these simple techniques turn out to be very useful *within* the context of a plan hierarchy, but do not have the ability to replace it. In late October, after I put together the ideas above, my next step was to take a harder look at the "classic problems of pattern recognition," which are clearly essential to any model of intelligence.

III. Pattern Recognition: The Paradigm of Time-Series Prediction

The problem of general pattern recognition turned out to be trickier than I'd expected. My work for Deutsch last year was a paper of advanced techniques in predicting statistical time-series; also, last year, I had looked carefully at the work of Minsky and Uhr, and had proposed a general model of pattern recognition as a supplement to my old _Cybernetica_ paper. In practice, the problem turned out to be quite complex.

Logically, it seemed reasonable to work with the following paradigm: to predict $\vec{X}_{t+1}$ (an input vector) from $\vec{X}_t$ by means of a circuit which "duplicates the actual process"; i.e. a circuit which conjures up likely $\vec{X}_{t+1}$ in proportion to their actual probability of occurring after the given data, $\vec{X}_t$. Such a circuit could be called a "basic model of the environment of the organism." The same circuit could be used, however, to "classify" patterns, i.e. to predict a description vector, $\vec{d}$, from the given input, $\vec{X}$; it could be used to adjust the function $J^0(\vec{X}_t)$ to be a good predictor of $u(\vec{X}_t)+J^0(\vec{X}_{t+1})$, as in Howard's formula, by treating $J^0(\vec{X}_{t+1})$ as a trial function, temporarily fixed in value. Also, if we give our system a "memory blackboard" (or a "symbolic blackboard"), $\vec{r}_t$, we can try to develop _two_ circuits, one to predict $\vec{X}_{t+1}$ given $\vec{r}_t$ and $\vec{X}_t$, the other to calculate $\vec{r}_{t+1}$ from $\vec{r}_t$ and $\vec{X}_t$; we can expand our prediction paradigm to include the joint adaptation of both systems. In this case, $\vec{r}$ will tend to be an extrapolated image, based on past memory, of the state of the _entire universe_, including those parts of the universe not now visible to the system. ($\vec{r}$ will also serve as storage for the intermediate results of calculations which could not be performed quickly enough.) Such an expanded pattern-recognition system I call an RCM, or Reality Construction Module.

So: how do we build an RCM?

The first time that I worked on this, back in 1964, I relied on the "Threshold Logic Unit" model of the neuron, a model which originated with McCulloch and Pitts and which is popular to this day. A lot of early "perceptrons" were based on ad hoc systems for adapting the coefficients of these units, systems totally unrelated to established knowledge in statistics and optimization theory. Nowadays, people seem to be more interested in proving theorems about the theoretical possibilities of TLU networks, when the coefficients may be plugged in by fiat, than in studying adaptive networks; this may be due to the basic intractability of the TLU concept. But the human brain does not have all its coefficients plugged in by fiat; we cannot ignore the question of adaptive routines for our perceptual networks. Back in 1965, I noted that the RCM proposed above has to have some ability to "throw dice", in order to simulate stochastic processes. I also noted that timing uncertainties, etc., do introduce random factors in human neurons. Therefore, I proposed a "PLU" concept. The TLU was designed to calculate a linear combination, W, of its inputs, weighing them by a set of arbitrary coefficients; then, if W is greater than zero, the TLU fires a one, or otherwise it fires a zero. (Minus one, in some formulations.) The PLU also calculates a W, but if W is between one and zero, the PLU behaves differently from the TLU; it will "throw dice" to decide whether to fire a one, with a probability W of firing one. (A neurophysiologist at RAND has recently proposed an equivalent model of the neuron, on empirical grounds.) In the Cybernetica article, I proposed an RCM system based on PLU's; this November, however, I proved that that system, while correct in principle, is incredibly inefficient; I was able to give a good indication that neither TLU's nor PLU's can give us an efficient basis for pattern recognition.

So: I returned to a simpler, more direct approach. Suppose that we had a direct network (no r for now!) going from the data $\vec{X}_t$ to the estimate $\hat{\vec{X}}_{t+1}$, made up of layers of similar subsystems:

$$\vec{Z}_i = f(\vec{a}_i, \vec{Y}_i, \vec{R}),$$

where $\vec{a}_i$ is a set of coefficients (a_{ij} being a component of $\vec{a}_i$), where $\vec{Y}_i$ is a set of Z_j's from the next lower layer of the system, and where R is a random vector. Suppose that we try to measure $\langle\frac{\partial e}{\partial a_{ij}}\rangle$, the average change in the error of our prediction with a change in each coefficient; if there were any direct index of how to adjust coefficients, bit by bit, surely this derivative would have to be measured, either by itself or along with second derivatives. (i.e. we minimize error.) We could not measure the average derivative efficiently unless we could also measure the derivative in a particular case history, with only two time-points of data. ($\vec{X}_t$ and $\vec{X}_{t+1}$.) But the effect of a_{ij} on $\vec{X}_{t+1}$ and on error is mediated by its effect on Z_i; thus we need to be able to compute $\frac{\partial e}{\partial Z_i}\cdot\frac{\partial Z_i}{\partial a_{ij}}$. (The Z_i derivative is controlled for other Z_j on the same level, and for subsequent $\vec{R}$'s and coefficients; the a_{ij} derivative is controlled for other inputs to Z_i. Formally, we are invoking the chain rule.) Given that $\frac{\partial e}{\partial Z_i}$ is computed back from $\frac{\partial X_j}{\partial Z_i}$, it should be clear that we want f to be differentiable in $\vec{Y}_i$, for all $\vec{a}$'s and $\vec{R}$'s; given that we also want $\frac{\partial Z_i}{\partial a_{ij}}$ to be defined, we also want f to be differentiable in $\vec{a}$, for all $\vec{Y}$'s and $\vec{R}$'s. These two conditions will be met by almost any reasonable f that we might think of, except for the PLU and the TLU. The conditions do require that Z_i varies continuously between its minimum and its maximum; the PLU and the TLU restrict Z_i to be 1 or 0. In practice, current neurophysiology does show that many neurons tend to fire in bursts of spikes, bursts which vary continuously in strength. Therefore, I propose a different model of the elementary neuron - the "CLU", or continuous logic unit. A CLU would calculate W, a linear combination of its inputs, just like its predecessors. If W is less than zero or greater than one, ~~respectively~~ the CLU will fire a zero or a one, respectively, just like its predecessors; however, if W is in-between, the CLU will simply fire a W. ("One", in this case, does not mean one spike, but one maximum burst. In principle, if we leave the zero-floor in place, we could forget about this ceiling of "one" on the neuron's output; we would still have an adequate nonlinear, continuous unit.)

Once we have a network of CLU's or the like, we can take one of two approaches. Either we can adapt a_{ij} more or less in proportion to $\langle\frac{de}{da_{ij}}\rangle$, or we can account for the second derivatives. The second approach turns out to be equivalent to nonlinear multiple regression; it also turns out to be too expensive for an efficient brain.

So: supposing that we have an historical record of $\vec{X}_t$ and $\vec{r}_t$ from time 0 up to time T, what is the fastest way to calculate "$\langle\frac{de}{da_{ij}}\rangle$" ? Let me assume that:

$$L(t) = (1-\theta)L(t-1) + \theta\sum_i e(X_{t,i}, \hat{X}_{t,i}), \qquad (5)$$

where L is the overall loss function due to errors up to time t, and where θ gives us the freedom to allow an expoential decay in the value of past information. Let me also assume that:

$$\vec{X}_t = \vec{g}(\vec{X}_{t-1}, \vec{r}_{t-1}, \vec{a})$$
$$\vec{r}_t = \vec{h}(\vec{X}_{t-1}, \vec{r}_{t-1}, \vec{a}),$$

where $\vec{a}$ is a set of coefficients such that each coefficient connects only one component of $\vec{r}_{t-1}$ or $\vec{X}_{t-1}$ to one component of $\vec{r}_t$ or $\vec{X}_t$. (This is for simplicity, and leads to no loss of generality. A time-gap of "1", here, refers to the cycle time of our individual neurons. If predictions were made across longer time-intervals, allowing us to compute more interaction terms, we could treat $\vec{r}$ as an intermediate storage, or we could simply declare that the function "e" is automatically zero except at times of a new data cycle; we could declare that "$\vec{h}$" gets an input equal to either zero, or to the last data input, or to the last data input plus a timing signal. I am assuming here that our neurons are CLU's or some variation on CLU's.)

We want to know $\frac{\partial L(T)}{\partial a_{ij}}$, controlled for $\vec{r}_0$, for $\vec{a}$, and for the actual values of $\vec{X}_t$ in the entire interval.

Let me define:

$$\lambda_{ij}(T,t) = \frac{\partial L(T)}{\partial a_{ij}}, \ \underline{\text{controlled}} \text{ for } L(t), \vec{r}_t \text{ and for } \vec{X} \text{ in the interval } t \text{ to } T.$$

Using the chain rule, we will have, for a_{ij} which provide inputs to r:

$$\lambda_{ij}(T,t-1) = \lambda_{ij}(T,t) + \left.\frac{\partial r_{i,t}}{\partial a_{ij}}\right|_{\substack{\text{other}\\ \text{inputs}\\ \text{to } r_{i,t}}} \cdot \left.\frac{\partial L(T)}{\partial r_{i,t}}\right|_{\vec{r}_t, L(t), \vec{X} \text{ from } t \text{ to } T}$$

The first of the two derivatives will be trivial to compute, with CLU's; to compute the second, we have to work backwards from time T using the chain rule again:

$$\frac{\partial L(T)}{\partial r_{t-1,i}} = \sum_k \frac{\partial L(T)}{\partial r_{t,k}} \cdot \frac{\partial r_{t,k}}{\partial r_{t-1,i}} - \theta^{T-t} \sum_k \frac{\partial e}{\partial \hat{X}_{k,t}}(X_{t,k}, \hat{X}_{t,k}) \frac{\partial X_{t,k}}{\partial r_{t-1,i}}$$

Given that $\lambda_{ij}(T,T)=0$ and $\frac{\partial L(T)}{\partial r_{t,k}}=0$, we can use these two formulas to work back from $\lambda_{ij}(T,T)$ to $\lambda_{ij}(T,0)$, which is what we want to calculate. For each step backwards in time, we have to update one number (λ_{ij}) for each coefficient, and one number $(\frac{\partial L}{\partial r_i})$ for each component of r; we only need to use enough storage to store the current values of each set of numbers. The sums over k should be fairly easy to compute in each case, since they only involve the r_k connected directly to r_i by CLU units. (i.e. the sum is nonzero only for terms with k such that a_{ki} is nonzero.) The total amount of processing to calculate these numbers is on the order of mT, where m is the total number of coefficients. One could not hope for much better, even for the coefficients which affect $X_{t,k}$ directly, and whose adaptation is trivial, by the method above. (Furthermore, if one is worried about hill-jumping, one should remember that any trial feature, set into this system with zero coefficients attached to its output, initially, will gradually be "listened to" via significant nonzero coefficients in other CLU's, if it helps predict anything; therefore, there is always an uphill path from an initial predictor system to a better predictor system, or at least an uphill path to a third system making use of both to get better results than either one of them. When we set up trial features, we will admittedly be more likely to find a simple feature of value than a complex one, since we have to rely on chance; however, pattern recognition itself is limited by the basic epistemological assumption that we can rely on Occam's Razor. Without Occam's Razor, induction itself would be impossible; I would add a discussion of this point, if this paper were not too long already.)

However: this model of pattern recognition does not fit with anything we know about human intelligence. It requires a long, synchronized history of the system's perceptions, in detail. Even Penfield, the apostle of total recall, admits that perceptual data usually are not stored by the brain in any detail; it would be difficult to imagine the mechanism of such storage. Furthermore, this model requires a <u>backwards</u> kind of dreaming. When humans do give up their perceptual centers (e.g. visual cortex) to dreaming, they usually dream forwards in time. In deep sleep, the visual cortex does not show the coordination that the model above would require. (MIT Neurosciences Symposium, Volume 2.)

On the other hand, we can't adapt such a network of neurons efficiently by a forwards-moving routine. I spent quite awhile trying to find some kind of forwards-moving adaptation routine. The basic problem is that one has to update $(\frac{\partial r_{i,t}}{\partial a_{jk}})$, where i need have no special relation to j and k; this is a huge set of numbers. Multiple regression, by the way, imposes a similar requirement, since one has to correlate $(\frac{\partial X_{t,i}}{\partial a_{jk}})$ with $(\frac{\partial X_{t,i}}{\partial a_{lm}})$ across all time-periods and all choices of i, to calculate the cross-correlation matrix between the changes produced by da_{jk} and da_{lm}.

So: we face another paradox.

Suppose we tried to eliminate the "$\vec{r}$" memory between different time-periods. Would that help simplify our problems? If our network for predicting $\vec{X}_{t+1}$

had several stages, operating either in parallel or one-after-another, we would still have to work backwards from the top stage to the bottom stage after every feedback, by the formulas above. We would either have to store everything in the network in a coherent, synchronized record - again, a doubtful idea - or we would have to update $\frac{\partial L}{\partial a_{ij}}$ on the spot, as we get our input; the second choice seems unavoidable. Suppose that we had ten layers operating in parallel, with "1" the cycle time of each layer and "10" the time-gap we are predicting across. Then we would really be predicting $\vec{X}_{t+10}$ from $\vec{X}_t$, $\vec{X}_{t+11}$ from $\vec{X}_{t+1}$, and so on. While we were processing data at each stage, feedback would be coming back down to the same cells from the other direction. The bottom layer would have to wait precisely twenty cycles before its ~~prediction~~ normal output had effect all the way forwards, and produced feedback ten cycles back; if its timing were off by only one cycle - a 3% error, rounded off - it would be using data from the wrong cycle. Therefore, cells in this layer would have to have a memory of their actions twenty cycles back, ready for instant retrieval, ~~perfectly synchronized with~~ based on perfect knowledge of the timing connections of other layers (which make it 20 cycles, not22), in a strange kind of push-down storage. This would require a very rigid inborn layering system, to keep the time relations all rigidly correct.

There is a simple way to overcome these difficulties, at minimal real cost to our intelligent system: by deliberately bringing in data in cycles. Thus our system could bring in $\vec{X}_t$, and wait until $\vec{X}_{t+10}$ is in hand before doing anything new. Then each cell would only have to remember its action in the last cycle of prediction, and it wouldn't have to keep track of what layer it is in. In fact, the system could even bring in $\vec{X}_{t+10}$ to the feedback system only, then bring in $\vec{X}_{t+20}$ for prediction of $\vec{X}_{t+30}$, bring $\vec{X}_{t+30}$ in for the feedback system, and so on. In general, data-cycling is essential for simplifying the feedback problems I have discussed. In practice, this is exactly what the human system of pattern-recognition does; visual information, which requires several ~~layers of~~ levels of processing in the cortex (17,18 and 19 at least), is cycled in from the thalamus according to an alpha rhythm of about 1 cycle per 1/12 second. This compares to about 3 milliseconds per synapse, and about 1/100 second maximum per processing cycle. Other kinds of perceptual information, requiring less processing, can be cycled in at a higher frequency. (e.g. somatosensory information, which comes in with a high-frequency rhythm.) The considerations above provide an explanation for such brain rhythms.

Unfortunately, the explanation above is not the only possible explanation. We have partly resolved the paradox of forwards versus backwards: we have decided that the feedback system does work backwards, and may go backwards, say, ten cycles to process a set of data, but that the system as a whole moves forwards - over longer cycles - in the list of data pairs it considers. (Like the old "one step backwards, two steps forwards" idea.) But we haven't brought back the $\vec{r}$ vector yet. There are three different ways we could bring it back; I'm not sure which is best, and I admit that this is one of the most serious ambiguities in my present model.(The only other, so far, has been my uncertainty about distinct result categories, in describing $W_iM_i^{-1}$.) First, we could simply stick in $\vec{r}_t$ along with $\vec{X}_t$ as a kind of data input; when we feedback to $\vec{X}_t$, we could feed back one more stage with the $\vec{r}_t$, since these cells would have no other feedback to process in that feedback cycle. (i.e. There would be no feedback from $\vec{r}_{t+1}$ available in the current cycle. Still, we might need a little trick with memory to record the inputs to $\vec{r}_{t+1}$, for use in the next cycle.) This would keep the procedures of the paragraph above intact; given that some kind of short-term memory, or reality construction, is absolutely necessary, it would be hard to imagine that we could accept less provision than this for $\vec{r}_t$. Also,people have found clearcut recursive, reverberating circuits among the stellate cells, in the fourth layer of visual cortex, the cells which receive the raw input.

(cf. Beritashvili in *A Handbook of Contemporary Soviet Psychology*, Coleman ed.) On the other hand, we might consider a larger role for $\vec{r}_t$, by allowing recursive processing *within* the normal layers. (The neurology, as above, suggests as much.) It seems wasteful to think that we should use each level of cells, one at a time, while leaving most of the levels idle most of the time; if recursive connections were allowed, we might be able to use a given cell several times in each cycle. So: two new possibilities. In possibility number two, we could have each cell remember only its *last* active configuration in the past processing cycle. This would allow us to continue on, as in the paragraph above. However, there are many variations of this idea. Instead of retaining the last active configuration, the system could calculate an average configuration, based on an expoential learning system to weight the average towards more recent configurations.The system could refuse to send back more feedback after its first input of feedback in any feedback cycle, or it could reduce its attention to later feedback according to an expoential decay function. These expoential decays are related to the autocorrelations of the variables affected. Unfortunately, possibility number two looks like the most likely, from the anatomical data. Possibility number three is to allow these recursive links, but to adapt them honestly, by the formulas above. Then each cell would have to remember its input and output for *each* neuron processing cycle; we would be back to the need for a synchronous memory. (Notice that this would not obviate the need for an alpha rhythm: if there were ten different inputs cycling at any time through visual cortex, then each cell in this recursive net would *also* have to process feedback on ten different levels *at once*, one level for each level of depth in its memory; the human brain may be subtly engineered, but it would have little reason to engage in *that* kind of wild complexity, given the few extra benefits to be had.) ~~This~~ But the third possibility, *with* an alpha rhythm, is not quite as bad as the early cycle-free system I talked about.

At the end of each major processing cycle (X_{t+10} received), each cell could immediately begin to play the cycle backwards at the same rate it moved forwards before. Synchronization would be achieved automatically, due to the common starting time; there would be no need for an elaborate system to communicate time between cells. If the feedback part of the cycle were *extended*, to allow the system to play back memories from *before* X_t, we could adapt r_t as accurately as we like, by the original formulas. The timing of this approach would be no more awkward than that of playing back to the beginning of the cycle.

The first two possibilities make a lot more sense, economically, than the third. But they seem to "cheat" in forming memory. They would allow us to save, from time t-1, the information of greatest use to predicting time t. (This *includes* the memories available at time t-1.) They do *not* allow us to consider the entire future in deciding what to save; in other words, they don't let us save the information of greatest use to tomes t+1 and later, except by accident. *However*: even our "right" formulas are designed to let us remember the right details for prediction over short time-intervals. To remember information important over long time-intervals, we may have to resort to direct use of long-time schemas anyway.
If there is a real universe, with a few prominent variables of great importance that we can see only indirectly, we would probably save them even in the crude memory ~~schemas~~ systems of possibilities one and two, If the variables we save depend all that much on time horizon, then *none* of the variables of short-term importance can expect to have unusual long-term importance; at any rate, there would be little reason to expect one short-term set to be that much more important than another. The extra cost of possibility three seems to be more than its value justifies.
Possibility two clearly looks best.

Before we go on to more interesting issues, there are a couple of other possibilities here I should mention in passing, for completeness.

One might imagine that possibility two could be true in the cortex, while possibility three could be true in the hippocampus, a specialized long-cycle-time organ with a known relation to short-term memory capacities. Or the hippocampus might save a memory of its recent states, one configuration per major processing cycle, to allow something like possibility three at a lower cost. But if we look, in the hippocampus, for something like the stellate cells of the cortex, we find only "basket cells", which do not seem to form the same kind of recursive network. (cf. The Eccles symposium, Brain Mechanisms and Consciousness, 1965. I may have the title backwards.) The extra costs of possibility three make me suspect that the hippocampus is more likely to be using possibility one or no recursion at all then possibility three; if the hippocampus were used to make emotional evaluations of hypothetical possibilities, then "recursive memory" between cycles might simply get in the way of keeping the different possibilities distinct. I am not yet ready to make a definite comment on this possibility.

In principle, one might also design a system with "functional feedback." One could put in a special adaptive network, parallel to the major one, to try to predict the feedback which will come back to $\vec{r}_t$, ~~on~~ as a function of the known values of $\vec{r}_t$ and $\vec{X}_t$; at the end of the feedback cycle of the major network, the minor network could then be adjusted. The feedback of the major network could use this extra network to give it feedback for $\vec{r}_{t+1}$, given the known values of $\vec{r}_{t+1}$ and $\vec{X}_{t+1}$. This three-cycle or four-cycle recognition system could eliminate certain kinds of short-sighted bias, but it seems very unrealistic as a model of anything in the mammalian brain.

The last ten pages add up to a fairly conventional, fairly explicit model of a pattern recognition system; systems on this model I call CRCM's, or Continuous Reality Construction Modules. However, this model neglects two critical points:

(i) My initial goal was to conjure up $\hat{X}_{t+1}$ according to their actual probabilities. The system above does not do this. Also, there is no reason to believe that this system will help *extract out* variables of interest to higher-order systems.

(ii) In the beginning of this paper, I stated that the first subgoal of this thesis would be to define what I meant by "new and interesting environments." The planning system I discussed in the previous section is not seriously affected by the lack of such a definition; we presume that our pattern recognition system will take care of these assumptions about the environment when it produces a model, "M(S)", for the motivation system to use. *However*, when we talk about pattern recognition, we should make these assumptions more explicit, and we should make sure that our pattern recognition system fits with the assumptions as well as possible.

The first point I have been able to deal with fairly effectively, especially this past month. The second point has caused me more headaches than any other aspect of the thesis; I think I have it mostly licked, by now, but there is still some work left to do. (As I finish writing up my work on the first of these points, I find that I'll have to limit my discussion of the second to a dogmatic summary. Hadn't expected to write so much.)

So: the first point.

To begin with, the point seems a bit 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 $X_{t,i}$, our ~~pattern rec~~ CRCM will still produce an $\hat{X}_{t,i}$ which varies from zero to one; we could interpret $\hat{X}_{t,i}$ as a probability, and plug into our CRCM an error function which encourages $\hat{X}_{t,i}$ to equal the probability of $X_{t,i}$ being "one". (To minimize $E((X_{t,i}-\hat{X}_{t,i})^2)$, for example, we set $\hat{X}_{t,i}=p(X_{t,i})$. A question of minimizing $p(1-\hat{X})^2+(1-p)\hat{X}^2$.) Then, to conjure up <u>possible</u> $X_{t,i}$, we could simply "throw the dice" for each i, based upon $\hat{X}_{t,i}$ 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 input field consists of a simple 2 X 4 visual grid. Suppose that the visual field only takes on 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 is on the bottom. At each time, t, the field takes on a new configuration, either A or B, with equal probability and with no regard for its previous configuration. <u>Using the pattern recognition systems above, every cell in the grid will be given an</u> $\hat{X}_{t,i}$ <u>of 50%, at all times</u>. If we "throw the dice" for each of these cells separately, the odds would be more than a hundred to one against our predicting either configuration A or configuration B. (Probability equals $(\frac{1}{2})^8+(\frac{1}{2})^8=1/128$.) 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. <u>Even if we could make a deterministic prediction of</u> $\vec{X}_{t+1}$ <u>based on all the data implicit in</u> $\vec{X}_t$<u>, 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</u>; in other words, the deterministic recognition systems ~~above may find it difficult to find a~~

above may find it difficult to find a pathway ~~for~~ 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. (Footnote, of special interest to Prof. Deutsch: this effect may also be relevant to the effectiveness of social perception in large organizations. The ability to accept uncertainties, and to deal with them coherently, may be essential to social learning and creativity.)

So: how do we account for this effect?

Formally, we have to account for the fact that $p(X_{t,i})$ is not independent of $p(X_{t,j})$. If we wanted to be simple-minded, we could set up our network, not to give $p(X_{t,i})$ separately, but to give

$p(X_{t,1})$, $p(X_{t,2} | X_{t,1})$, $p(X_{t,3} | X_{t,2}, X_{t,1}) \ldots$ $p(X_{t,n} | X_{t,n-1} \ldots X_{t,2}, X_{t,1})$.

To simulate $\vec{X}_t$, we could simulate $X_{t,1}$, then $X_{t,2}$ in light of $X_{t,1}$, and so on, until we had simulated all the components of $\vec{X}$. Given that $p(\vec{X})$ does equal the product of the conditional probabilities above, even when its components are mutually dependent, this simulation process will conjure up X's in accord with their actual probabilities.

To learn the conditional probability, $p(X_{t,i} | X_{t,i-1} \ldots X_{t,1})$, we could set up a pattern recognition routine like those above, using $\vec{X}_{t-1}$ and $X_{t,i-1}$ through $X_{t,1}$ as inputs to predicting $X_{t,i}$. In practice, it would be hopelessly extravagant to have a separate circuit for every input variable; it would also be hopelessly slow to deal with one input after another, through such a long list.

To achieve the same effect in a more economical way, we will have to formulate the issue in a more elegant way. However: it should be clear that there is no way to account for the mutual dependence of the $X_{t,i}$, without accounting for the conditional probabilities in one way or another; we will need to have circuits to predict $X_{t,i}$, given other information about $\vec{X}_t$.

Let us go back to the simple idea of "minimizing an error function," which we mentioned above. We didn't specify which error function to use, or why we should use such an approach. In classical statistics, people have found a 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. The nonlinearity of our processes is so pervasive, that we are better off thinking of our input as pure information, codable into zeroes and ones, in the paragraph above.

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

Suppose that we have a model, $p(x_t)=f(t,a)$. ("t", in this case, can cover all kinds of extra data; for now, however, we are interested only in how to adapt the coefficient "a".) Suppose that we have a series of data, x_t, from time zero to time T. Then:

$$p(a|\{X_t,t=0,T\}) = p(\{x_t,t=0,T\}|a)\frac{p(a)}{p(\{x_t,t=0,T\})}$$

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

The term in the denominator of the right-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 <u>a priori</u> 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 a priori probability to one "a" or another, <u>except</u> to a=0. (Occam's Razor again.) In choosing the best nonzero value for "a", we can delete 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:

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

We want to find an error function, $e(x_t,f)$, such that ~~maximizing~~ 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:

$$e(x_t,f) = -\log(x_t f(t,a) + (1-x_t)(1-f(t,a)))$$
$$= -x_t \log f(t,a) - (1-x_t)\log(1-f(t,a)) \quad (\text{as } x_t=0 \text{ or } 1)$$

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 ~~x_t~~ x_t through time.

Now, let's go back to our problem of interdependent probabilities in $\vec{X}_t$. Our problem, precisely, was that the probabilities of $X_{t,i}$, given $\vec{X}_{t-1}$, were interdependent. In other words, the information content (entropy) of $\vec{X}_t$ as a whole, given $\overleftrightarrow{X}_{t-1}$, may be much less than the sum of the entropies of its parts (the $X_{t,i}$). (Footnote, of special interest to Prof. Deutsch: Gestalt psychologists and political scientists have often said that ,"The whole is more than the sum of its parts." The organization content of the total visual field, as a system, may be defined as its theoretical information capacity, minus its entropy (disorder). The theoretical information capacity of the whole will equal the sum of the information capacities of its parts; however, given that its entropy will be less than (or=) to the sum of the entropy of its parts, we may say, ~~qi~~ quantitatively that the ~~information~~ organization content of the whole will be more than (or =) the sum of the organization content of its parts. It is much greater, at least, in the case of human perception.)

In order to simulate $\vec{X}_t$, given $\vec{X}_{t-1}$, economic considerations dictate that we 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 $\vec{X}_t$, given $\vec{X}_{t-1}$, into a set of independent variables, which we can deal with separately, all at once, in simulating $\vec{X}_t$. In short, we wish to find a technique to <u>encode</u> $\vec{X}_t$ (given $\vec{X}_{t-1}$) into a new vector, $\vec{R}_t$, of zeroes and ones, whose entropy <u>does</u> equal the sum of the entropies of its parts. We wish to make sure we can <u>recode</u> $\vec{R}_t$ and $\vec{X}_{t-1}$ back into $\vec{X}_t$, so that we can <u>simulate</u> $\vec{X}_t$ by simulating all the components of $\vec{R}_t$ in parallel and recoding. The components of $\vec{R}_t$, <u>unlike</u> the elements of a simple CRCM, would tend to represent unusual joint events in $\vec{X}_t$ and $\vec{X}_{t-1}$; they would tend to represent the unusual patterns and events which disrupt the more predictable monotony of human vision. <u>In practice</u>, this is exactly what Lettvin and Hubel have found in the higher visual systems of mammals and frogs - edge detectors, detectors of vertices at special angles, and even "newness" detectors which respond to moving objects which change their direction and velocity. A pattern recognition system built on these principles will automatically select out the most interesting features of its environment, for study by higher-up systems in the brain. Systems of this type I call BRCM's, for Bayesian Reality Construction Modules.

Now: the technical details.

If we give ourselves a fairly free hand in selecting $\vec{R}_t$ as a function of $\vec{X}_t$ and $\vec{X}_{t-1}$, we cannot be sure of finding a straightforward way to recode it back into $\vec{X}_t$. In general, we will have to use pattern recognition to try to predict $\vec{X}_t$ back from $\vec{R}_t$ and $\vec{X}_{t-1}$. Suppose that we define the "gross $\vec{R}_t$" to be the <u>combination</u> of our original $\vec{R}_t$, plus the information required to give us $\vec{X}_t$ back, item by item, from the proposed $\vec{X}_t$ we got back from recoding. Given all of the information in "gross $\vec{R}_t$", we can get back $\vec{X}_t$ exactly,

[redacted]

simply by recoding $\vec{R}_t$ into an estimated $\vec{X}_t$ and then correcting the result. Therefore, gross $\vec{R}_t$ cannot have less information content than does $\vec{X}_t$ itself, given $\vec{X}_{t-1}$. Therefore, the minimum of the sum of the entropy of the parts of gross $\vec{R}_t$ cannot be less than the entropy of $\vec{X}_t$ given $\vec{X}_{t-1}$. We can bring this sum of entropies down to this lowest possible minimum by converging on an $\vec{R}_t$ proper with the properties called for above: i.e. the sum of the entropy of the components of $\vec{R}_t$ proper equals the entropy of $\vec{X}_t$ given $\vec{X}_{t-1}$, while the recoding process involves no error or entropy at all.

There is one other way of minimizing the entropy of the parts of gross $\vec{R}_t$: 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 will be greater than the entropy of the whole, which in turn is greater than or equal to the entropy-level we are trying to reach.)

So: we can set up a network with two levels - $\vec{R}_t=\vec{f}(\vec{X}_t,\vec{X}_{t-1})$ and $\vec{X}_t=\vec{g}(\vec{R}_t,\vec{X}_{t-1})$ - to be adapted like a CRCM, but with feedback (an error function term) coming in from the entropy of each component of R and from the error of each component of $\vec{X}$. (Notice, when we estimate the probability of $X_{t,i}$, *controlled* for $\vec{R}_t$, we are, as promised, estimating a probability controlled for knowledge about $\vec{X}_t$.) Notice that we are worried about the entropy of $\vec{R}_t$ *controlled* for $\vec{X}_{t-1}$; given that we are worried about separating the components of $\vec{R}_t$ from each other, and not about separating them from $\vec{X}_{t-1}$, we can get away with measuring the entropy of $R_{t,i}$ *controlled* for some function of $\vec{X}_{t-1}$. In short, we can try to "predict" $R_{t,i}$ from information about $\vec{X}_{t-1}$, and count in just the entropy of the error. (This will work better, because $R_{t,i}$ would often represent part of our error in predicting $\vec{X}_t$ from $\vec{X}_{t-1}$; the requirement that $R_{t,i}$ be zero or one would force our system to develop contorted coding systems for these errors, were it not for our little trick here.) We can even set up several *layers* of $R_{t,i}$, so long as we do not go to our old extreme of having one layer for every bit of input; this may be convenient in a network which would have several layers of CLU processing anyway. Each layer could use the actual value of $R_{t,i}$'s from previous layers, for use in trying to predict its own $R_{t,j}$.

As with the CRCM's, we can avoid formal layering by using 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 $\vec{X}_t$ and $\vec{X}_{t-1}$, 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 $\vec{X}_t$, and from Pyramids which have already released the actual value of their $R_{t,i}$. (i.e. effectively, lower-level ~~Py~~ Pyramids.) At a certain time, dependent on the state of its inputs (or on the time since the beginning of the alpha cycle, or even on a signal from a central coordinating system), 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 CLU's would be mixed together in a common network, with the same timing and the same feedback problems as those of a pure CRCM. Under our old possibility three - a doubtful possibility - Pyramids could actually record predictions and release $R_{t,i}$ on each processing cycle of the system. Under ~~possibility~~ possibilities two and one, ~~and~~ a Pyramid would limit itself to remembering (and reacting to) the time when it gets a wave of inputs from its top part; it would be like the CLU's in this respect.

When the feedback cycle gets back to our Pyramid, the Pyramid (like a CLU cell) 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 internal probability, q, generated in the ~~a~~ bottom of the ~~Pyramid~~ Pyramid, and will ~~retain~~ have also retained p, the estimated probability of $R_{t,i}$ according to the top of the Pyramid. q and p will be generated by CLU-type units within the Pyramid. It is unpleasant to evaluate $R_{t,i}$, a zero/one variable, by means of a derivative; however, there is no other reasonable way for us to measure and limit the amount of entropy that passes through this process, and it is impractical to measure the exact difference in entropy caused by a discrete change in $R_{t,i}$. (If the network using $R_{t,i}$ were linear life would be easier on both counts. But the human environment is not linear.) It would be possible to make one small concession: the last cells before $R_{t,i}$ could pretend that the derivative of entropy with respect to their outputs is constant, and use that to estimate the discrete result of changing $R_{t,i}$ from zero to one or vice-versa. In any case, each input to the top, 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. (Also, the initial set-up of a Pyramid can be carried through, if necessary, by getting it to duplicate exactly a higher Pyramid, exactly, if the errors of that higher Pyramid correlate with those of other Pyramids on the same level.)

This idea of Pyramid units is one of the best in this paper, because I was not the one who thought of it. I reached the point of seeing the need for an R vector, but I could not imagine a practical way to carry through the idea in a normal biological system. Then I looked at the Scheibel and Scheibel article in the Rockefeller Neurosciences Study Program Volume II. The large pyramid cells of the visual cortex fit the need above, exactly. Each such cell, usually called a Layer V pyramid, reaches from the very top of the cortex (of Layer I) down to almost the bottom (Layer VI). In betweem, 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 1/0.) The top of the pyramid, in layer I, receives new input only from the fibers in this layer; Svengothai (sp?) 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. In only one of the other sources I read could I find a definite indication of any other cells sending axons to this area; the Eccles volume mentioned above reported a group of strange little cells, unique in a number of ways, <u>inputting only from pyramid collaterals</u>, and outputting to probably layers I and II, These cells, as reported, would merely be accessories to the pyramid tops in processing data which they are entitled to receive. Direct inputs from the thalamus, by contrast, go straight to Layer IV, almost all a thick plexus of medium/small cells called Layer IV stellate cells. Most of this input gets to the pyramids via synapses to the cell body of each pyramid, in Layer V. (Scheibel disagrees, but Eccles and Beritashvili and others appear to have overwhelming evidence against him.) The medium-sized cells in Layer VI have direct access to Layer IV inputs, but they send their output fibers into the white matter underneath layer VI;

these association fibers presumably come back into the cortex on the same footing with fibers carrying new data.

(One may wonder how the pyramids could get $\vec{X}_{t-1}$ input in this situation. But such input was only added as a minor afterthought on page 31, anyway. If desired, such input could simply come from memory of the output of other pyramids from the end of the cycle before; this would work especially well if pyramids could integrate the input they receive over the whole interval from one release to the next.)

Layers II, II and IV of the cortex are trickier to deal with, because of ambiguous data; it is hard to say exactly where the "top" of a pyramid ends and the "bottom" begins. However, there is little doubt that the upper inputs into the pyramid are integrated, and the apical shaft begins to send its calculations down, above layer IV. The cells in layers II and III are all small pyramids, which also get their inputs from Layer I; thus inputs to the "top system" from these cells would not contradict our theory. (Well, there are also those strange little upside-down stellate cells, mentioned above, and I am assuming that we are defining the boundary between Layers III and IV according to the system commonest in the literature.) Scheibel claims that the shaft itself receives input directly from the sensory centers of the thalamus, from the opposite side of the cortex, and from the nonspecific nuclei of the thalamus. These claims, if true, would merely tell us that the boundary between the "top" and the "bottom" may be somewhere in the upper part of the shaft, But Scheibel's claims about sensory input are doubtful, because of sources cited above (the Eccles volume, for example, claims that the climbing fibers Scheibel talks about tend to be intracortical fibers, according to degeneration experiments); input from other pyramids on the opposite side of the cortex, passing down and up again through the white matter below, sounds even harder to validate, and would not be a fundamental problem anyway;

input from the nonspecific system of the thalamus is known to be central in cortical timing mechanisms, and could be available solely to tell the pyramid when to release $R_{t,i}$. Therefore, the simplest explanation still makes sense: that the boundary between "top" and "bottom" may occur at the place where a spike from the shaft would meet the cell body.(Implying no stellate-pyramid contact on Layer IV.) The cell bodies of these pyramids are very large, and capable of carrying through the more complex adaptation routines I have suggested above.

The BRCM system proposed above has one other major advantage: it is capable of "sleep." By "sleep," I mean a process for adapting the coefficients of a system at a special time, when the system is not dealing actively with its environment. <u>Normal</u> CRCM coefficients, and the coefficients of systems to define $R_{t,i}$, have to be adjusted as part of a larger system, since their feedback comes from a set of derivatives which would be changed quickly (i.e. sent to zero) by the adaptation of other parts of the system; it would be invalid to adjust all these parts, separately, by means of past feedbacks, given this kind of interconnection. However, the (top) subcircuits which predict the $R_{t,i}$ of one pyramid, based on the $R_{t,j}$'s of others, do not produce any direct output at all, except when our model is used for simulation. If a single pyramid could make molecular records of inputs to its top, and of its q, over a number of interesting cases, it could go back over the most interesting cases at night, and adapt its coefficients quickly. The cell would be able to adjust its input coefficients much more quickly than it could in the daytime, because it could carry out one cycle of adaptation for every cycle of its own processing, instead of following the slow frequency of the alpha rhythm; also, it could focus its attention on the cases of greatest interest to itself, by any number of chemical means which I am in no position to speculate about. Evarts, in the MIT Neurosciences Volume II, has shown that V pyramids do loosen their interrelations in deep sleep, that they fire on the order of ten times the normal rate, and that smaller cortical cells slow down at the same time.

IV. Other Hypotheses Studied

Before I close this up, I should make a few brief comments about the rest of my work this year. (Also, I should apologize for making this longer than expected; my handwriting seems to have grown denser.)

Back in 1965, when I worked on defining the philosophical basis of pattern recognition, I came up with the idea of using Occam's Razor to define the apriori probability of different models (the p(a)-type terms, which never disappear in Bayesian statistics), in a way which would lead to some kind of "open-minded system." A friend of Minsky's, Solomonoff, published a similar idea in the mid-sixties, which Minsky, at least, agreed was solid. This idea, applied directly to pattern recognition, places severe demands on our systems. There are only two ways to meet those demands completely: (i) by a neuron adaptation system in which glial cells bind several axons, dendrites, and even synapses, together, and force them to behave in a similar way; (ii) by a chemical processing system, in which large numbers of molecules pass from cell to cell, allowing local information transfers on a large scale, and exploiting the domain of quantum chemistry. None of these make much sense for mammalian brains; also, there is good psychological reason to suspect we do not have the abilities that would result. There are two ways to compromise between these demands and the simplicity of the BRCM idea: (i) Hyden-type associative memories in pyramidal neurons, perhaps in the smaller pyramids of layers II and III of the cortex, layers which exist in neocortex but not in primitive cortex; (ii) "families" of neurons, designed to deal with different objects of the same kind or with specialized applications of a given action-schema; these families may involve groups of pyramids from layers II and III, clustered about one or more layer V pyramid. This is what I may have to work on next.

In February and March, I spent a lot of time dealing with the care and cultivation of schemas. Different schemas may still use a standard thalamic coding system to describe themselves. The cortex may perform the $W_iM_i^{-1}$ function, of describing the results of a schema, and also of proposing standard subschemas. The limbic system may calculate $\vec{J}_G$ from ~~the cortical data~~ data in the cortex, or at least J^0 of the results described, and may even help in the basic reticular ~~for~~ formation task of proposing different subschemas from the standard ones. The reticular formation defines what is an actual, engaged plan, versus what is hypothetical; thus the lower tegmentum stifles the actions dreamed up by the cortex during sleep, according to the MIT report above, and the reticular formation censors directly the output of motor pyramids. The cerebellum is less interesting - a servomechanism designed to smooth out motor actions, on the Howard/BRCM principles which we ~~g~~ rejected as a central organizing ~~sche~~ principle on grounds of its short-sightedness. The corpus striatum would be a relic of the ~~primitive~~ precursors of the cerebral cortex, for schemas capable of engagement but not of description,schemas subject to learning by experience but not by simulation. The thalamus may be a kind of timing, switching and storing center for the cortex, telling the cortex what to predict, telling different ~~areas~~ regions of the cortex to adapt to different levels of abstraction, and initiating brain rhythms.

Most of the work in caring for schemas comes from the utilitarian programs which decide which schemas to use, and which provide gradual adaptation for their coefficients. Simulation and experience provide two different ways of adapting these coefficients; basically, I propose that simulation should be used in the beginning, and that experience should be used to adapt an <u>extra</u> circuit to measure the <u>bias</u> of the models coming from simulation. Similar bias systems are needed in dealing with schemas adapted to specialized functions. Coherent dreams and Penfield-type memories both puzzle me; even ~~if the~~ in the hippocampus or mammillary bodies, no one cell could be expected to construct such coherent memories, so another kind of memory system is needed, involving some kind of avalanche of associations.

Interesting questions: how the same pyramids described above can take on motor functions in other parts of the cortex, reacting to ~~degrees of pressure~~ the level of pressures towards action and inaction; what happens to recursion when we create hypothetical description and prediction routines; where we put the "globschema", the schema at the top of our lattice; whether Lettvin's "sameness" cells in the frog colliculus have a lesson to teach us about object-identity.