Mathematical & Computational Applications, Vol. 4, No. 3, pp. 305-314, 1999.
© Association for Scientific Research

BUILDING EVOLUTION FRIENDLINESS INTO CELLULAR AUTOMATON DYNAMICS: THE CYTOMATRIX NEURON MODEL

Ahmet Ugur and Michael Conrad
Department of Computer Science
Wayne State University
Detroit, MI 48202 USA

Abstract- The cytomatrix neuron is a softened cellular automaton, roughly motivated by interactions that could occur in a molecular or cellular complex. Subcells exert graded influences on each other and provide a medium for the integration of input signals in space and tim . A readout element located in a suitably activated subcell triggers an output. The neurons are trained through variation-selection learning that acts on multiple dynamical parameters. Extensive experimentation with the model shows that the dynamics can be molded to yield different structures of generalization. Dimensionality can be increased by increasing the number of dynamical parameters open to variation and selection. Learning algorithms that vary the greatest number of parameters were found to have a greater variability in the structures of generalization and to yield higher performance values and learning rates. The focus here is on n-bit exclusive-OR tasks that are known to be hard due to their linear inseparability. The system successfully learned 2-bit and 4-bit exclusive-OR functions. The higher dimensional algorithms exhibited a relatively good performance on the 8-bit exclusive-OR function.

1. INTRODUCTION

Biological information processing is a complex phenomenon. Signals are combined in space and time at different levels of organization. Macro inputs from the environment are first combined by *inter*neuronal signaling mechanisms. Individual neurons then process the com lex spatiotemporal patterns of signals impinging on them to yield appropriate firing behavior. Their capacity to do so draws on *intra*neuronal molecular and biochemical processes. Evidence is accumulating that the cytoskeleton, the fibrous fine structure of cells, serves as a fast signal integrating network [1-5]. Biochemical and cytoskeletal activities are ultimately controlled by the shape-based pattern recognition capabilities of biological macromolecules.

Our group has concentrated on models in which the neurons are fairly complex entities. We call such models "neuromolecular" since the input-output transform performed by the neurons is controlled by internal biochemical and molecular processes [6-8]. The general picture is that the signaling mechanisms within the neuron, for example in the cytoskeleton, serve to combine presynaptic input signals in space and time. If the combination produces a suitable degree of activation at the location of a readout enzyme events are triggered that culminate in membrane electric activity. The models are trained to perform pattern recognition/effector control tasks through evolutionary (variation-selection) learning algorithms [9-11].

The model to be described here will referred to as the cytomatrix neuron, because of the above general motivation. The question to be addressed is this: what types of interactions among the components of such a network make it amenable to molding through variation and selection? The working hypothesis is that softening the network with multiple graded interactions and increasing the number of dimensions open to evolution increases evolution friendliness by increasing structure-function malleability. The representation is in terms of what we call a softened cellular automaton, essentially a cellular automaton dressed with graded interactions

Mathematical & Computational Applications, Vol. 4, No. 3, pp. 305-314, 1999.
© Association for Scientific Research

BUILDING EVOLUTION FRIENDLINESS INTO CELLULAR AUTOMATON DYNAMICS: THE CYTOMATRIX NEURON MODEL

Ahmet Ugur and Michael Conrad
Department of Computer Science
Wayne State University
Detroit, MI 48202 USA

Abstract- The cytomatrix neuron is a softened cellular automaton, roughly motivated by interactions that could occur in a molecular or cellular complex. Subcells exert graded influences on each other and provide a medium for the integration of input signals in space and time. A readout element located in a suitably activated subcell triggers an output. The neurons are trained through variation-selection learning that acts on multiple dynamical parameters. Extensive experimentation with the model shows that the dynamics can be molded to yield different structures of generalization. Dimensionality can be increased by increasing the number of dynamical parameters open to variation and selection. Learning algorithms that vary the greatest number of parameters were found to have a greater variability in the structures of generalization and to yield higher performance values and learning rates. The focus here is on n-bit exclusive-OR tasks that are known to be hard due to their linear inseparability. The system successfully learned 2-bit and 4-bit exclusive-OR functions. The higher dimensional algorithms exhibited a relatively good performance on the 8-bit exclusive-OR function.

1. INTRODUCTION

Biological information processing is a complex phenomenon. Signals are combined in space and time at different levels of organization. Macro inputs from the environment are first combined by *inter*neuronal signaling mechanisms. Individual neurons then process the complex spatiotemporal patterns of signals impinging on them to yield appropriate firing behavior. Their capacity to do so draws on *intra*neuronal molecular and biochemical processes. Evidence is accumulating that the cytoskeleton, the fibrous fine structure of cells, serves as a fast signal integrating network [1-5]. Biochemical and cytoskeletal activities are ultimately controlled by the shape-based pattern recognition capabilities of biological macromolecules.

Our group has concentrated on models in which the neurons are fairly complex entities. We call such models "neuromolecular" since the input-output transform performed by the neurons is controlled by internal biochemical and molecular processes [6-8]. The general picture is that the signaling mechanisms within the neuron, for example in the cytoskeleton, serve to combine presynaptic input signals in space and time. If the combination produces a suitable degree of activation at the location of a readout enzyme events are triggered that culminate in membrane electric activity. The models are trained to perform pattern recognition/effector control tasks through evolutionary (variation-selection) learning algorithms [9-11].

The model to be described here will referred to as the cytomatrix neuron, because of the above general motivation. The question to be addressed is this: what types of interactions among the components of such a network make it amenable to molding through variation and selection? The working hypothesis is that softening the network with multiple graded interactions and increasing the number of dimensions open to evolution increases evolution friendliness by increasing structure-function malleability. The representation is in terms of what we call a softened cellular automaton, essentially a cellular automaton dressed with graded interactions and readout elements that trigger output. The model, because of its abstractness, could as well be as a network of interacting cells as a network of macromolecular components within a single cell.

We have elsewhere reported on the pattern generalization capabilities of the cytomatrix neuron, and on the evolutionary moldability of these capabilities. Here we will concentrate on a narrow task domain, namely the multi-bit exclusive-OR, that is generally considered difficult for adaptive systems and which therefore can serve to measure the utility of the softening strategies used.

2. THE MODEL

2.1 Overview

The model comprises a population of processing networks, called cytomatrix neurons, and an evolutionary learning algorithm. As noted above the neurons can be thought of as softened cellular automata whose components (or subcells) interact to combine input signals in space and time. When a suitable degree of excitation is produced in a subcell that is occupied by a readout element the neuron fires (schematically represented in Figure 1). The cellular automaton representation is intended to capture the fact of signal integration in the cytoskeletal-membrane system, but the model does not attempt to empirically model the complex (and only partly known) structure and function of this or any other actual system. For the present purposes it is sufficient to think of it as representing integrative processes in a patch of subcellular cytomatrix.

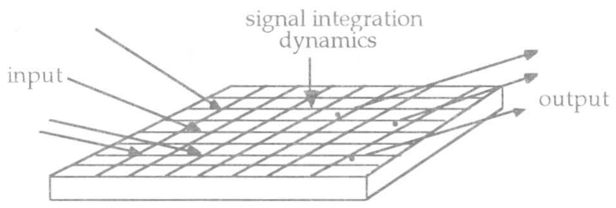

Figure 1: Schematic illustration of the cytomatrix neuron model. Black dots represent readout enzymes. The grid provides a medium for signal integration (not all sub-structures residing on the grid are shown).

2.2 The Neuronal Dynamics

The cytomatrix neuron is modeled with an 8×8 cellular automaton (with wraparound). The subcells (representing molecular components) can be in one of five states of activation (q0 to q4). The influence exerted on neighboring cells increases with the level of activation. In the present study a subcell is allowed to interact only with its nearest or next-nearest neighbors (including diagonal neighbors). But the model is so organized that interactions with more distant neighbors are possible. The interaction strengths fall off inversely with distance. Bridges hat amplify the interactions in a unidirectional fashion may also connect a subcell to any o. its nearest or next-nearest neighbors. Whether a subcell is in fact activated by the influences received from other subcells depends on an activation window. The window is defined by a lower threshold and an upper threshold (or ceiling) that are subject to variation and therefore may differ from subcell to subcell. The totality of influences is partitioned into three classes by the activation window (to be denoted by $\mathbf{W}_A$), the barrier below the activation window (to be denoted by $\mathbf{W}_L$), and the barrier above the activation window (to be denoted by $\mathbf{W}_H$). These are defined by the intervals $[\theta_L, \theta_H]$, $(0, \theta_L)$ and $(\theta_H, \infty) \cup \{0\}$, respectively, where θ_L *is* a lower threshold and θ_H is an upper threshold.

The evolutionary algorithm acts on the neuron by varying the locations of readouts and the parameters that control signal integration (the locations of neighbors, bridges and receptors, the subunit types, and thresholds defining activation windows). Varying the signal integration dynamics alters the manner in which potentially widely separated signal inputs are combined in space and time to yield internal patterns of activation. The readout enzymes interpret these patterns by responding to the activation level of the subcell in which they are located. In the present study 16 time steps are allowed for the neuron to accept or reject the input (i.e., to fire or not to fire). The signal integration is thus primarily mediated by the run-in phase of the dynamics.

The number of receptor enzymes is fixed and equal to the number of bit positions in the input vector. Receptor locations are distributed randomly, but are the same for each neuron if receptors are not subject to variation. Receptors and readouts are most conveniently viewed as superimposed on the cellular automaton, since they do not contribute to the interactions among the subcells apart from transmitting influences from the outside in the case of receptors. Output (1 or 0) may be initiated by any readout enzyme that is activated.

The term "softening" refers to the fact that the different activation states allow for graded interactions. The effective influence depends on the activation state of the sending cell and on the activation state of the receiving cell. Let **s** denote a sending subcell, **r** a receiving subcell, and let **I**[**s**, **r**, τ] denote the effective influence that **s** exerts on **r** at time τ. The nonlinear function used to determine this influence is

$$\mathbf{I}[\mathbf{s}, \mathbf{r}, \tau] = \mathbf{A}_{\mathbf{q}[\mathbf{s}, \tau]}\, \mathbf{A}_{\mathbf{t}[\mathbf{s}]}\, \mathbf{A}_{\mathrm{L}[\mathbf{s}, \mathbf{r}]}\, (1/\mathbf{R}^2)$$

where $\mathbf{A}_{\mathbf{q}[\mathbf{s}, \tau]}$ is an activation level that depends on the state of the sending subcell **s** at time τ, $\mathbf{A}_{\mathbf{t}[\mathbf{s}]}$ is a signal amplification factor that depends on the subcell type of **s**, $\mathbf{A}_{\mathrm{L}[\mathbf{s}, \mathbf{r}]}$ is a signal amplification factor that depends on the unidirectional connection (or link) type from **s** to **r**, **R** is the Euclidean distance between **s** and **r**. $\mathbf{A}_{\mathbf{q}[\mathbf{s}, \tau]}$ is set to 0.5 for the active state q1, 1.0 for the active states q2, q3 and q4, and 0 for the quiescent state q0. $\mathbf{A}_{\mathbf{t}[\mathbf{s}]}$ is set to 1.0, 0.75, 0.5 for subcell types 0, 1, and 2 respectively. Subcell types are selected randomly among these three types and kept the same for each neuron if types are not subject to variation. When a bridge is present the amplification factor $\mathbf{A}_{\mathrm{L}[\mathbf{s}, \mathbf{r}]}$ is set to 2.0, and set to 1.0 if absent. The total effective influence exerted on any given subcell **r** at time τ is obtained from

$$\mathbf{I}_{\mathbf{tot}}[\mathbf{r}, \tau] = \Sigma\, \mathbf{I}[\mathbf{s}_i, \mathbf{r}, \tau]$$

where the sum is taken over all subcells to which **r** is connected by the neighborhood relation and bridges.

In order for the neuron to fire at least one subcell must enter the active state q3. A readout enzyme must also be located in this subcell. The subcell then enters the firing state q4. If the neuron fires, all subcells are reset to the quiescent state q0 at the next time step. The state transitions are given in Table 1.

2.3 Evolutionary Learning Algorithm

The cytomatrix neuron processes information at two main levels of organization: at the level of the integrative dynamics and at the level of readouts. The integrative dynamics is subject to variation through windowing parameters, neighborhood and bridge connections, receptors, and subcell types. The presence or absence of readouts in particular locations is also subject to variation and selection. The evolutionary learning algorithm can proceed in a number of ways, e.g., through separate and alternate variation-selection on each parameter, or simultaneously. The alternating mode is used here, with alternation occurring every 16 generations (but after 32 if learning stagnates). See Figure 2 for a pseudocode description of the algorithm.

Table 1: The state transition table. A dash indicates an irrelevant situation or condition. Y denotes the output of the neuron.

$q(\tau)$	**Next State** $q(\tau+1)$							
Present State	**Input**			**Active Window**			**Readout Present**	**Y**
	0	1	None	W_L	W_A	W_H		
q0	q1	q2	q0	-	-	-	-	0
q0	-	-	-	q0	q1	q0	-	0
q1	-	-	-	q1	q2	q0	-	0
q2	-	-	-	q1	q3	q0	-	0
				Readout Absent				
q3	-	-	-	q1	q2	q0	q4	0
q4	-	-	-	q0				1

The training set includes both positive and negative examples. The neuron is rewarded for accepting positive examples and for rejecting negative examples, and is punished for errors in this respect. Fitness (denoted by **F**) is defined as

$$\mathbf{F} = \alpha_r P_{acc} + \alpha_p P_{rej} + \beta_r N_{rej} + \beta_p N_{acc}$$

where P_{acc} is the number of accepted positive inputs, P_{rej} is the number of rejected positive inputs, N_{acc} is the number of accepted negative inputs, N_{rej} is the number of rejected negative inputs, α_r and β_r are reward coefficients, α_p and β_p are punishment coefficients, and $\alpha_r > 0$, $\beta_r > 0$, $\alpha_p \leq 0$, $\beta_p \leq 0$, $\alpha_r \in \Re$, $\beta_r \in \Re$, $\alpha_p \in \Re$ and $\beta_p \in \Re$. (The reward coefficients α_r and β_r are always positive, the punishment coefficients α_p and β_p are always negative or zero.) Reward and punishment coefficients used in experiments are described in Section 3. The population comprised 16 neurons randomly supplied with a small number of initial readouts and bridges (2 or 3 on average) and also with random activation windows. The 3 neurons with the highest fitness are selected for reproduction. If 3 or more tie for the highest fitness 2 of them will be selected randomly for reproduction and a neuron of next lower fitness will be selected as the third neuron. Each selected neuron is retained and its distribution of readouts, bridges, and window parameters are copied to an approximately equal fraction of the less fit neurons (with the most fit neurons copied to any remaining neurons in an arbitrary order). The copy process is accompanied by randomly adding or deleting readouts or receptors, randomly adding or deleting bridges or neighbors, randomly mutating types, or randomly and independently incrementing or decrementing lower and upper window parameters. For types and receptors the mutation probability was set at 0.5. For windows the mutation probability was set at 0.5, with an equal chance of increment or decrement. The actual magnitudes of the increments and decrements were also set to 0.5, but held within the interval [0.25, 9.75]. Other mutation rates used in experiments are described in Section 3. Mutating neighbors means assigning a random but potentially different neighborhood topology to a mutating unit. The strategy for doing this could be specified as gradual (mutate one neighboring connection at a time) or sudden (mutate several connections in a group).

```
Initialize parameters, evolving levels, and the population (the generation time T = 0);
terminated = FALSE;   T = 1;
While ((T <= T_maximum_allowed) and (NOT terminated)) do
begin
    Evaluate every neuron in the present population for all inputs in the training set;
             if the task is learned, terminated = TRUE;
    Select most fit neurons from the present population;
    Reproduce the present population to get the next population by copying structures of most fit neurons
              with variations to less fit neurons (variations are applied to current evolving levels only),
              but keeping the most fit neurons;
    Alternate evolving levels when stagnated or after a certain number of generations;
    T = T + 1;
end;
```

Figure 2: Pseudocode for a generalized multiple level evolutionary learning algorithm.

2.4 Algorithm Types

The features that may be subject to mutation were subcell types (i.e., strength of influ nce exerted by subcells), windows (i.e., receptiveness of subcells to influence), location of neighbors, location of bridge connections that mediate strong influences, and location of readouts and receptors. An instantiation of the generalized multiple level evolutionary learning algorithm given in Figure 2 may utilize any combination of these six features. Opening different features to evolution corresponds to opening up different dimensions in the phenotypic space and can be viewed as a way of increasing the dimensionality. Four different algorithms were developed. Algorithm 1 varies readouts, bridges and windows, as do algorithms 2, 3 and 4. Algorithm 1 is, accordingly, called the standard (multilevel) algorithm. Algorithm 2 adds neighbors, algorithm 3 adds neighbors and subunit types, and algorithm 4 adds neighbors, subunit types and receptors. The algorithms and which levels they evolve are given in Table 2.

Table 2: Algorithm types based on the structures that are subject to variation.

algorithm	mutating levels
algorithm 1	readouts, bridges, windows
algorithm 2	readouts, bridges, windows, neighbors
algorithm 3	readouts, bridges, windows, neighbors, types
algorithm 4	readouts, bridges, windows, neighbors, types, receptors

2.5 Task Description

Binary bit pattern recognition tasks were selected as the problem domain. Experiments require that the training set be generated first. The training set could be generated either systematically or randomly. The input vectors generated are related (i.e., are structured) when the systematic procedure is used. (A structured training set is generated by flipping a fixed number of bit positions selected randomly and holding other positions constant.) The positive training set is first generated. The noise level of an input vector is the minimum Hamming distance between the vector and the inputs constituting the positive training set, i.e., is measured by the number of bit changes required for the arbitrary selected patterns to be converted into one of the positive training patterns. The noise level defined above partitions the set of all inputs [12]. The full training set consists of the positive set plus negative examples chosen from the different noise levels. Noise level 0 corresponds to the positive training set itself.

The noise level is really an index of distance from the positive training set, not necessarily a measure of average distance. Noise tolerance increases as the number of input vectors accepted increases. Thus if the system completely learns the positive training set it will be completely tolerant to the bit vectors in this set. If there were no negative examples its tolerance to noise level 1 inputs would increase directly as the fraction of these increases, and so on for higher noise levels. If negative examples from noise level 1 are included in the training set tolerance is still defined the same way. The term generalization is strictly speaking more appropriate than tolerance, since we do not require that the acceptance be greater when the test inputs are closer to the training inputs. The structure of tolerance thus represents the structure of generalization.

Experiments were terminated as soon as one neuron completely learned the task, or terminated after a fixed number of generations if the task was not completely learned.

3. EXPERIMENTAL RESULTS

3.1 Preliminary Remarks

Numerous experiments have been performed with the model. For convenience we can divide these into three categories. The first includes results that demonstrate the system's ability to learn and to scale up as the difficulty of the task increases. The second includes the dimensionality experiments using variant algorithms. For extensive experiments on 5-bit and 8-bit tasks, see [7] and [13]. Scaled up-results and dimensionality experiments on 16-bit, 32-bit and 64 bit tasks was described in [12]. The third group of experiments, to be described here, was directed to exclusive OR (i.e., parity) tasks that put particularly difficult demands on a learning system.

With respect to the first two categories we can summarize the overall results thus. As with any dynamical learning system the cytomatrix neurons possesses certain inherent generalization (or grouping) tendencies. As the dynamics are made more plastic these grouping tendencies become increasingly responsive to the structure of the training set. Essentially the inherent tendencies themselves become more open to evolutionary molding. The characteristics of the positive training set, the size of the negative training set and the selection pressure put into the negative training set with respect to the noise levels have all been shown to affect the grouping tendencies. In most cases the moldability of these tendencies is also increased by increasing the number of parameters open to evolution. Increasing dimensionality by increasing the number of parameters open to evolution has been shown to affect the moldability of generalization in most cases. It also increases the performance and learning rate in most cases, but whether it does so depends on the complexity of the problem. As the problem becomes more complex opening up more parameters to evolution is generally helpful.

3.2 Exclusive OR (X-OR) Problem

The exclusive OR, denoted by $\oplus$, is a binary operation on the set of switching elements (i.e., elements from the set $\{0,1\}$) that maps its operands to 1 if their values are different and to 0 otherwise (i.e., if their values are the same). The operation is commutative, associative and distributive (see [14]). The value of X-OR is the modulo-2 sum of its operands. The n-bit X-OR function assigns the modulo-2 sum of its n-bit operands (i.e., assigns 1 if the number of 1s in the operand is odd and 0 if the number of 1s is even) and is also known as the "parity" function.

The 2-bit X-OR function is known as a classical example of linearly inseparable functions from the threshold logic point of view used in most artificial neural network models. Linear seperability for n-bit inputs means the existence of an (n−1) dimensional hyperplane from the geometrical point of view. Thus it answers the following question: for inputs of the form $x_1x_2 \ldots x_n$, are there weights $w_1, w_2, \ldots, w_n$, and a threshold θ such that $w_1x_1 + w_2x_2 + \ldots + w_nx_n = \theta$ hold? The weights and the threshold here are real numbers. Such a hyperplane, if it exists, separates 1 inputs from 0 inputs (i.e., maps to 1 if $w_1x_1 + w_2x_2 + \ldots + w_nx_n > \theta$ and to 0 if $w_1x_1 + w_2x_2 + \ldots + w_nx_n \leq \theta$). The 2-bit X-OR is not linearly separable and therefore can not be realized by a single McCulloch-Pitts neuron. More generally, a single layer perceptron or artificial neural network that is threshold logic based can only realize linearly separable functions (see [15]). Any n-bit X-OR function for $n > 2$ is clearly linearly inseparable. The proof can be done by induction using the associativity property.

3.3 N-bit Exclusive-OR Experiments

Experiments were performed on the 2-bit, 4-bit and 8-bit X-OR tasks. For each task class one half of the set of all inputs are mapped to 1 and the other half to 0 using the appropriate n-bit X-OR mapping. The 1 inputs are used as the positive training set and 0 inputs are used as the negative training set, thereby yielding one level of noise only. Algorithms 1-4 were applied. The reward and punishment parameters were set as follows: $\alpha_r = 4.175$, $\alpha_p = -2.193$, $\beta_r = 2.177$, and $\beta_p = -8.543$. With this choice, the neurons are rewarded more for correct recognition of negatives than for correct recognition of positives. The mutation rates were set at 0.2 for readouts and bridges, with the probability of the mutation being a deletion set higher than the probability of its being an addition (0.8 as compared to 0.2) and at 0.2 for neighbors. Experiments started with the same initial conditions. The neighborhood topology was assigned randomly in the beginning for algorithm 1 (then stayed the same) and varied for algorithms 2, 3 and 4. The different types of variation operations were applied in the alternating manner noted above. A gradual (or fine) mutation strategy was used for mutating neighbor-neighbor connections, with an equal probability of the mutation being an addition or a deletion (i.e., 0.5 for addition and 0.5 for deletion). The subunit types were also assigned randomly in the beginning. These stayed the same for algorithms 1 and 2 and varied for algorithms 3 and 4. The receptor locations were assigned randomly in the beginning, and stayed the same for algorithms 1, 2 and 3 and varied for algorithm 4. The maximum number of generations allowed for learning was 5000 (over 32,000 for 8-bit X-OR task).

The 2-bit X-OR task performances and learning rates are given in Table 3. Algorithms 3 and 4 always learned the task completely with the gradual neighbor mutation strategy, with algorithm 4 being the fastest. Algorithms 1 and 2 did not complete learning under these conditions but did so (though not shown in the table) when the neighbor mutation strategy was made sudden (i.e., neighbors mutate in groups) and the mutation probability was increased. The difference is due to the fact that the low mutation rate used in the standard setup yielded overly weak signal propagation for the 2-bit case.

The 4-bit X-OR task performances and learning rates are given in Table 4. Again, algorithm 4 was the fastest.

The 8-bit X-OR task performances and learning rates are given in Table 5. (The total number of training inputs in this case is 128, with 64 positives and 64 negatives.). The normalizod fitness was 78.2% for algorithm 3 and 78.4% for algorithm 4. Algorithm 3 and 4 showed the best performance.

The task gets much harder as the number of bits increases. This is due to the fact that this particular task class shows the highest degree of symmetry from a syntactic point of view. All bits need to be examined order to classify an n-bit input correctly. Hamming distance, and the minimum and the maximum of the Hamming distances can be used to measure how close the positive (or negative) inputs are to each other. Recall (from Section 2.5) that a task is structured if the bit patterns are generated from a template in a way that puts some structure into the training set. The Hamming measures for a structured task with n-bit inputs that includes with 2^{n-1} positives, 2^{n-1} negatives are: minimum 1, maximum (n–1), and average (n–1)/2. The corresponding n-bit X-OR measures are: minimum 2, maximum n, and average n/2. Clearly the bit patterns that must be grouped together are more different in the case of the n-bit X-OR.

Table 3: The 2-bit X-OR task performances using variant algorithms.

algorithm	**% Performance (% positive, % negative)**	**generation**
algorithm-1	75 (50, 100)	5000
algorithm-2	75 (50, 100)	5000
algorithm-3	100 (100, 100)	768
algorithm-4	100 (100, 100)	337

Table 4: The 4-bit X-OR task performances using variant algorithms.

algorithm	**% Performance (% positive, % negative)**	**generation**
algorithm-1	100 (100, 100)	4591
algorithm-2	100 (100, 100)	4796
algorithm-3	81 (62, 100)	5000
algorithm-4	100 (100, 100)	916

Table 5: The 8-bit X-OR task performances using variant algorithms.

algorithm	**% Performance (% positive, % negative)**	**generation**
algorithm-1	50 (0, 100)	37,200
algorithm-2	56 (12, 100)	32,900
algorithm-3	73 (55, 92)	52,400
algorithm-4	73 (50, 95)	51,800

Experiments comparable to the 8-bit X-OR were also performed for an 8-bit structured task with flipping template xxxxxxx1, where x denotes a flipping position for generating the positive training set. All algorithms learned the task completely. Algorithm 1 learned in 1591 generations. Algorithms 2, 3 and 4 learned in 76 generations (too fast for these algorithms to show different mutational sequences). Another 8-bit structured task with flipping template xxxxxxx0 which yields a 00000000 in the positive training set was also investigated. Again, all algorithms learned the task completely. Algorithm 4 was the fastest. Clearly structured tasks are much easier to learn than a maximally symmetrical unstructured task.

The above experiments demonstrate that the cytomatrix neuron learned 2-bit and 4-bit X-OR tasks successfully and continued to learn the much harder 8-bit X-OR task. Also, the experiments further illustrated that increasing the dimensionality in the way described in Section 2.4 increases the training performances and the learning rate.

4. CONCLUSIONS

The essence of pattern recognition is grouping. An n-bit pattern allows for 2^n possible variants. The number of possible two category groupings of these patterns grows as 2 to the 2^n [16]. The manner of grouping depends in part on the structure inherent in the pattern set and in part on the grouping tendencies inherent in the dynamics of the pattern recognizing system. Systems exposed to the same training set of patterns will group, or generalize, differently if they have different dynamical properties.

Opening up the dynamics to evolution should open up the grouping tendencies to evolution. This has been the main motivation for our work with the cytomatrix neuron model. But simply opening up more and more parameters to evolution is not sufficient. If the dynamics are not evolutionarily malleable it will become locked into a local regime. The grouping tendencies will be rigidified. Structure-function malleability — what we have referred to as evolution-friendliness — must be built into the dynamics. The rather extensive experiments performed with the cytomatrix neuron model demonstrate this. Increasing the grading of the interactions increases the responsiveness of the grouping tendencies to the structure of training set. Increasing the number of parameters open to evolution also increases the responsiveness under these soft conditions. In general enhanced learning rates and performances are achieved, at least for more difficult problems. The grading of interactions and the opening of more parameters to variation and selection effectively increases the number of possible pathways for attaining satisfactory task performance.

The experiments reported here have focused on what is in a sense the complement to the generalization problem, namely learning a total training set that demands maximum malleability in order to prevent the system from grouping the inputs in an undesirable way. Exclusive-OR, because of its linear inseparability, fits this profile. Complete learning was possible for the 2-bit and 4-bit X-OR tasks. Higher learning rates were achieved for the 8-bit case as the number of parameters open to evolution (the dimensionality) increased. The results are consistent with our previous results on controllability of generalization in response to the structure of the training set. On the one side the possibilities for transforming the generalization properties increases as the malleability of the dynamics increases; on the other the possibilities for being trapped in a dynamics that enforces undesirable grouping decreases.

The dynamical mode of pattern generalization instantiated by the cytomatrix neuron stands somewhere between syntactic and statistical approaches. The syntactic approach allows for pattern grouping based on internal structure, since rule-like regularities in the patterns rather than overall morphological similarity is the dominant fact. But in general syntactic pattern recognizers are expressed in rule-like formalisms and hence are too brittle to learn through adaptive methods. Statistical pattern recognition (the underlying process in many connectionist neural net models) is more robust, but in general more reflective of overall similarity and therefore less flexible so far as grouping tendencies are concerned. The signal integration capacities of the cytomatrix neuron allow for a rule-like capability, but at the same time the graded aspects of the dynamics afford a high degree of robustness. The same principles can operate at multiple levels of biological organization, from the intercellular to the intramolecular. Pattern recognition in its larger sense cannot of course be reduced to grouping on the basis of structural attributes. The functionality of the structure and the context in which it is embedded must also be considered. Procedural modes plausibly work in concert with dynamic modes in natural biological systems. Our working hypothesis is that dynamic pattern processors like the cytomatrix neuron should be key components of multi-level architectures that combine to accommodate these complementary modes in a synergistic manner.

ACKNOWLEDGMENTS

This research was supported by the National Science Foundation under Grants ECS-9404190 and EIA-9729818.

BIBLIOGRAPHY

[1] E. A. Liberman, S. V. Minina, O. L. Mjakotina, N. E. Shklovsky-Kordy, and M. Conrad, Neuron generator potentials evoked by intracellular injection of cyclic nucleotides and mechanical distension, *Brain Research* 338, 33-44, 1985.

[2] S. R. Hameroff, *Ultimate Computing*, North-Holland, Amsterdam, 1987.

[3] G. Matsumoto, S. Tsukita, and T. Arai, Organization of the axonal cytoskeleton: differentiation of the microtubule and actin filament arrays. In Warner, F. D. and McIntosh, J. R., editors, *Cell Movement*, vol. 2: Kinesin, Dynein and Microtubule Dynamics, 335-356, Alan R. Liss Inc., New York, 1989.

[4] M. Conrad, Molecular computing. In Yovits, M., editor, *Advances in Computers*, 235-324, Academic Press, New York, 1990.

[5] P. Werbos, The cytoskeleton: why it may be crucial to human learning and neurocontrol, *Nanobiology* 1, 75-96, 1992.

[6] M. Conrad, R. R. Kampfner, K. G. Kirby, E. Rizki, G. Schleis, R. Smalz, and R. Trenary, Towards an artificial brain, *BioSystems* 23, 175-218, 1989.

[7] A. Ugur and M. Conrad, Multiple level evolutionary learning in neuronal pattern recognition. In McDonnell, J. R., Reynolds, R. G. and Fogel, D. B., editors, *Evolutionary Programming IV*, 271-288, MIT Press, Cambridge, 1995.

[8] J.-C. Chen and M. Conrad, Evolutionary learning with a neuromolecular architecture: a biologically motivated approach to computational adaptability, *Soft Computing* 1, 19-34, 1997.

[9] M. Conrad, Evolutionary learning circuits, *Journal of Theoretical Biology* 46, 167-188, 1974.

[10] R. R. Kampfner and M. Conrad, Computational modeling of evolutionary learning processes in the brain, *Bulletin of Mathematical Biology* 45, 931-968, 1983.

[11] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, Piscataway, 1995.

[12] A. Ugur, The Cytomatrix Neuron Model: An Evolutionary Dynamical Approach to Pattern Generalization, Ph.D. thesis, Wayne State University, 1998.

[13] A. Ugur and M. Conrad, Structuring pattern generalization through evolutionary techniques. In Angeline, P. J., Reynolds, R. G., McDonnell, J. R. and Eberhart, R., editors, *Lecture Notes in Computer Science 1213 (Evolutionary Programming VI)*, 311-321, Springer-Verlag, Berlin, 1997.

[14] Z. Kohavi, *Switching and Finite Automata Theory*, McGraw-Hill, New York, 1978.

[15] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*, MIT Press, Cambridge, 1995.

[16] M. L. Minsky and S. Papert, *Perceptrons,* MIT Press, Cambridge, 1969.