Genetic Algorithms for Genetic Neural Nets

David H. Sharp, John Reinitz, and Eric Mjolsness

Research Report YALEU/DCS/TR-845
January 1991

Genetic Algorithms for Genetic Neural Nets

David H. Sharp*[1], John Reinitz†[2] and Eric Mjolsness‡[3],

*Theoretical Division, Los Alamos National Laboratory
Los Alamos, NM 87545

†Department of Biological Sciences, Columbia University
New York, NY 10027
and
Medical Informatics Program, Yale University
Box 3333 New Haven, CT 06510

‡Department of Computer Science, Yale University
P.O. Box 2158 Yale Station, New Haven CT 06520-2158

February 1, 1991

Abstract

In contrast to most synthetic neural nets, biological neural networks have a strong component of genetic determination which acts before and during experiential learning. Three broad levels of phenomena are present: long-term evolution, involving crossover as well as point mutation; a developmental process mapping genetic information to a set of cells and their internal states of gene expression (genotype to phenotype); and the subsequent synaptogenesis. We describe a very simple mathematical idealization of these three levels which combines the crossover search method of genetic algorithms with the developmental models used in our previous work on "genetic" or "recursively generated" artificial neural nets [18] (and elaborated into a connectionist model of biological development [19]). Despite incorporating all three levels (evolution on genes; development of cells; synapse formation) the model may actually be far cheaper to compute with than a comparable search directly in synaptic weight space.

[1]Work supported by the United States Department of Energy.
[2]Supported in part by grant 5-T15-LM07056 from the National Library of Medicine.
[3]Supported in part by the Air Force Office of Scientific Research under grant AFOSR 88-0240.

1 GENES, CELLS, AND NETWORKS

Biology has motivated research on genetic algorithms as well as on synthetic neural nets. What biological phenomena could motivate a synthesis of these two approaches? We propose that such a synthesis can be based on the interplay of objects at three levels of organization: genes, cells, and networks of cells. Dynamics at the level of genes (point mutations, crossover, inversion and so forth) has been schematized for use in genetic algorithms. Likewise the dynamics of networks of neurons, including learning, are abstracted in synthetic neural nets. The missing dynamical system is development: the elaboration of genetic information to produce neurons and their initial connections. A phenomenological modelling framework for development was presented in [19]. A simplified version of this model, together with simple genetic algorithms and neural nets, results in a synthesis of these ideas in a unified connectionist model.

1.1 The Model

The main parameters in a neural net are its real-valued connection strengths W_{ij} between neuron i and neuron j. In our model, initial values of $\mathbf{W}$ result from a development process which creates neurons (indexed by i) and their internal state vectors $\mathbf{v_i}$. The state vector for one neuron could represent the concentrations of important proteins within that neuron. The "degree of match" between two state vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ determines the connection strength W_{ij}:

$$W_{ij} = f(\mathbf{v}_i \cdot \mathbf{Q} \cdot \mathbf{v}_j) = f(\sum_{ab} Q_{ab} v_i^a v_j^b). \tag{1}$$

This is a sigmoidal function of a quadratic form of state vectors $\mathbf{v}_i$ and $\mathbf{v}_j$. It could be simplified by taking $\mathbf{Q}$ to be the identity matrix and by taking f to be the identity function. Connection matrices (such as $\mathbf{W}$) specified as the degree of match between code vectors (such as $\mathbf{v}_i$) also occur in models of the immune system [5, 20].

Next we require an equation for the state vectors. The state vector of a cell is determined by that of its parent in a lineage tree of cell divisions. If a designated component of cell i's state vector is above threshold, that cell will divide to form two new cells indexed by (i, k) (where k is 0 or 1) whose state vectors are [19]

$$v_{(i,k)}^a = v_i^a + r_a g\left(\sum_b T_k^{ab} v_i^b + h_k^a\right). \tag{2}$$

This equation determines the course of development in our simplified system. Equation (2) is excerpted from a connectionist model of a gene regulation *circuit* whose connection matrix is $\mathbf{T}$. This circuit expresses the fact that proteins are gene products which can activate or repress the synthesis of proteins by other genes. Genes are indexed by a, and the protein product of gene a in cell i is v_i^a. For a fuller discussion of this type of model, including its elaboration as a phenomenological model for biological development, see [19].

We also wish to describe changes in the genes and the effect this produces in the regulatory circuit, specified by $\mathbf{T}$ and $\mathbf{h}$. (Since the threshold $\mathbf{h}$ can be regarded as an

extra column of **T** connecting to an extra constant component of **v**, we will omit explicit discussion of **h** in what follows.) Dynamics at the level of genes is described schematically by point mutation and crossover operations, modeled as in genetic algorithms. It would be a confusion of levels to apply these operations directly to the gene-gene interaction matrix **T**, since it is genes and not their interactions that are mutated. Thus point mutation and crossover should not be modeled as acting on elements of **T**, but as acting simultaneously on its rows and columns. One consistent way to do this is to represent "gene" a as a pair of vectors $\mathbf{p}^a$ and $\mathbf{c}^a$, subject to crossover and point mutation, which interact with gene b to determine T^{ab}. This formulation is motivated by the observation (see for example [21], [8], and [11]) that eucaryotic genes contain very large control regions called *promoters*. In fact, for many genes the promoter is significantly larger than the region which codes for protein. The promoters contain a large number of binding sites for protein products of other genes that control transcription from that promoter. A functional module of such binding sites (defined by certain experimental manipulations enabled by molecular genetics) is known as an *enhancer*. We regard $\mathbf{p}^a$ as having a set of scalar components p^a_α, each of which characterizes an enhancer. The protein product of gene b acts by binding to these sites. The action of the protein product of gene b when *bound* to these sites is given by the vector $\mathbf{c}^b$, with components c^b_β, where the index β, like α, ranges over the set of enhancers. In terms of $\mathbf{p}^a$ and $\mathbf{c}^b$, T^{ab} can be written as

$$T^{ab} = h(\mathbf{p}^a \cdot \mathbf{R} \cdot \mathbf{c}^b) = h(\sum_{\alpha\beta} R_{\alpha\beta} p^a_\alpha c^b_\beta). \tag{3}$$

This equation, also, could be simplified by taking **R** to be the identity matrix and h to be the identity function. With this parameterization of **T**, point mutations are single-element changes in p^a_α or c^a_α. Crossover is defined by a linear ordering on the a index and a corresponding linear ordering of the genes $(\mathbf{p}^a, \mathbf{c}^a)$. The usual crossover operation can be applied to two such strings of genes by swapping corresponding contiguous blocks of adjacent genes.

In previous work we have used a related "genetic neural net", optimized using simulated annealing of parameters appearing in a recursive growth rule (described in section 2.2.4), to solve a coding problem. Because it was the growth rule rather than the neural connection matrix **W** that was optimized, the net could be thoroughly trained on small coding problems and then scaled up to coding problems too large to handle otherwise.

In the rest of this paper, we will discuss the relationship between this simple three-level model and three areas of research that correspond to its genetic, developmental, and neural levels. The model's genetic level has analogs in the field of artificial genetic algorithms. The developmental level is a simplification of a phenomenological modelling framework for biological development. The neural level is a new way to bring the learning capabilities afforded by genetic algorithms to the field of neural nets.

1.2 The Genetic Level and Genetic Algorithms

Equations (1-3) define the moves in a genetic algorithm search procedure for a population of neural networks which are to be scored by their performance on neural net tasks. This scor-

ing function may include simulated experiential learning as well as simulated computation of network outputs.

Equations (1) and (3) each define a matrix in terms of a pair of matrices and a constant quadratic form on a new vector space, which introduces a new type of index. Thus N^2 parameters are determined by $O(NM)$ parameters, where M is the dimension of the new vector space. Highly structured matrices result from taking $M = O(1)$, and general matrices result from taking $M = N$. This reparameterization is a very cheap way to describe substructure in a multilevel model. It is usually difficult to include in one model coupling between effects at different length scales without a catastrophic increase in computational expense. This catastrophe is avoided here since the translation between levels is just matrix multiplication. The reparameterization does not introduce an additional level of loop nesting or an additional time scale, which are often the cause of high computational expense. In fact, the computational expense may be greatly reduced by reducing the number of free parameters.

A different way to use genetic algorithms in the search for successful neural nets is provided by the combination of classifier systems governed by genetic algorithms [13, 14] and Farmer's mapping of classifier systems onto neural nets [6]. In this mapping, a classifier rule corresponds to a set of connections which share a common strength parameter. Neurons are labeled by possible messages. The connection strengths can learn from experience by means of the bucket brigade algorithm. This use of genetic algorithms with neural nets has the advantage that it involves a *grammar*, the set of rules in the classifier system, which imposes network structure and which can be used to incorporate prior knowledge of the sort commonly expressed in AI knowledge representation schemes [7, 1].

Yet another way to combine genetic algorithms and neural nets is to define crossover and point mutations operations directly on a connection matrix [16]. This method produces *unstructured* neural nets: every parameter in the connection matrix is independent. Consequently it is very hard to map a grammar into the neural net, so at least that particular way of introducing high-level abstractions and prior knowledge into the network is not available.

By comparison with these approaches, in our model the developmental process maps a gene regulation circuit to a neural net. The genetic circuit is the intervening step between a genetic algorithm and a neural net also capable of experiential learning. As we just saw, a classifier system could play this role also. However, the developmental process provides considerably more flexibility for specifying a structured network and its connection matrix than does a classifier system. In fact the connectionist model of development can be considered as a small computation, as is done in [19]. It remains to be shown that prior knowledge and network structure at the level of abstraction available to a grammar can be incorporated into such networks. We will address this question in section 2.

1.3 The Developmental Level and Biological Models

In the study of biological evolution, the interaction of evolution and development has historically been a prominent theme [9] and is again today [3]. For example, heterochrony (evolutionary changes in developmental timing) is an important mechanism in evolution.

Such phenomena would be expected to occur in our multilevel model.

Equation (2) must be amplified considerably to arrive at a model which could be applied to real biological problems. A step in this direction has been taken in [19, 22], in which we presented a phenomenological modeling framework for development whose purpose is to provide a systematic method for discovering and expressing correlations in experimental data on gene expression and other developmental processes.

The modeling framework is based on a connectionist or "neural net" dynamics for biochemical regulators, coupled to "grammatical rules" which describe certain features of the birth, growth, and death of cells, synapses and other biological entities. Spatial geometry can be included, although this part of the model is not complete. The framework was applied to derive a rigorously testable model of the network of segmentation genes operating in the blastoderm of *Drosophila.* The elaboration of the equations for interphase by writing T^{ab} as a product of $\mathbf{p}^a$ and $\mathbf{c}^b$ is under consideration as a model of eucaryotic promoters.

The principal differences from the simplified model of equation (2) are the addition of a grammar of possible biological objects and processes, and the presence of continuous-time connectionist dynamics to model processes such as interphase.

1.4 The Neural Level and Synthetic Neural Nets

It is possible to parameterize a neural net in different ways. For example, the connection matrix $\mathbf{W}$ may be regarded as a set of independent parameters that specify the network. Equation (1), and therefore our model, provides a different parameterization in which the independent parameters are, ultimately, p^a_α and c^a_α. As we have seen, classifier systems provide another parameterization and many others have been used in the neural network field.

The various parameterizations differ widely in their functionality and cost of computation. They operate at different degrees of abstraction, have different properties when tested for generalization, and are best optimized by different search algorithms. The cost of computing and learning depends strongly on the network parameterization chosen. These issues are the subject of the next section.

2 PARAMETERIZATION OF NEURAL NETS

2.1 Cost of Genetic Neural Nets

The computational cost of the neural network parameterization specified by equations (1-3) is low, as discussed above, for two reasons. Equations (1) and (3) introduce new levels of modelling by matrix multiplication but may substantially reduce the number of free parameters. Also the introduction of a genetic algorithm search, including crossover, provides a way to parallelize a naturally serial search procedure such as simulated annealing, which is analogous to point mutation alone.

For fast special-purpose hardware such as reconfigurable neural net chips or SIMD parallel computers, there may be an additional computational advantage to networks that

can be described by parallelizable developmental dynamics. A serial host computer can transmit the developmental dynamics, specified by **T** and **Q**, rather than the much larger neural network, specified by **W**, to the parallel hardware. Since equation (2) is a parallel dynamical system for development, it can be efficiently run on a special-purpose machine. The resulting neural net can then be run or trained on the same machine. Consequently it would be interesting and useful to have a "compiler" which translates from high-level descriptions of a network's architecture and function to a parallel growth rule that can build the corresponding neural net. One such compiler is used in [15].

2.2 Functionality of Neural Net Parameterizations

2.2.1 Generalization

Generalization is an important issue in neural net learning. Typically, fewer trainable parameters decrease the size of the training set which a network must learn in order to successfully generalize beyond that training set. To understand this point, we refer to previous results on the application of the Vapnik-Chervonenkis dimension to neural network learning [2]. The VC dimension is a technical way to bound the number of different functions (relating input and output states of the neural net) that can be expressed as one varies the parameters which determine the connections in a neural network. The theorems yield formulae giving the size of the training set necessary for successful generalization from successful training, as a function of the VC dimension and of some parameters specifying the accuracy and confidence level of the obtained generalization. For a variety of cases in which the VC dimension has been calculated or bounded, it is roughly $cn \log n$ where n is the number of trainable parameters in the net. Also the training expense may grow faster than linearly with the size of the training set. Consequently, superfluous parameters should be avoided and thus parsimonious parameterizations of neural nets are advantageous. The model of section 1.1 is an extremely parsimonious parameterization.

2.2.2 Graph Dynamics and Search

Aside from its ability to replace sequential search with parallel search, discussed in section 2.1, the crossover operation changes the set of moves available to the search procedure to include moves that would be highly nonlocal in a search procedure based on point mutation alone. This increases the accessible search space, and may result in better solutions. However, the crossover operation should exchange code units that are meaningful for the problem under consideration.

As pointed out by Farmer [6], connectionist systems such as neural networks are often formulated with a static connectivity graph but a second phase of research is introducing "graph dynamics" on a sparse connectivity graph. Examples in the neural network field include [4, 10, 23]. By expressing an $N \times N$ connection matrix **W** as a product or quadratic form of two $N \times M$ matrices, as in equations (1) or (3), we introduce a *code* for the graph corresponding to **W**. Thus our graph dynamics is actually "code dynamics", i.e. a dynamical system for the codes.

2.2.3 Expressiveness, Abstraction, and Prior Knowledge

An important property of any neural net is the degree to which it can express and manipulate abstract things and relationships. Such expressiveness is necessary if an experimenter is to incorporate high-level prior knowledge into the network. The idea of expressiveness is hard to quantify; we nevertheless propose to illustrate it by a discussion of grammars which generate objects at various levels of abstraction. For example, we could use a two-level grammar whose top-level rules translate each letter of an alphabet into a set of straight line segments and segments of curves arranged in a characteristic configuration, and whose bottom-level rules translate line and curve segments into individual pixels which are made available to a perceiver.

A connectionist neural net for letter recognition under this grammar might be constructed by devoting a group of neurons to each possible instance of an object in the grammar, and a bundle of connections to each pair of neuron groups whose objects are related by a rule in the grammar. In this way the grammar can be *mapped* into the neural net. The grammar has a definite *interpretation* (in terms of actual letters, as opposed to formal terms) and this interpretation can be enforced during the network's training by scoring all groups of neurons, rather than just the output neurons.

It is also possible to map an uninterpreted grammar into a neural net. For example an evolved classifier system may be regarded as an uninterpreted grammar. Also the grammar occurring in the full development model of [19], when augmented by equation (1), clearly maps into the resulting neural net but does not have any particular semantic interpretation. Yet another example is the code used in the genetic algorithm for evolving neural nets in [12], which is an artificial grammar of many-neuron "areas" and their "projections" to one another. By contrast with these systems, the neural nets for visual recognition problems presented in [17] are derived from interpreted grammars which describe the composition and shape of visual objects.

2.2.4 Grammars and Genetic Neural Nets

For genetic neural networks it is straightforward to map an uninterpreted grammar into the development process, hence indirectly into the net, as we will show.

In [18] we introduced a recursive parameterization of connection matrices which may be viewed as a development process. Large connection matrices are specified in terms of smaller ones:

$$W^{a,n}_{(i_1,i_2)(j_1,j_2)} = \sum_b P^{a,b}_{i_1,j_1} W^{b,n-1}_{i_2,j_2} \tag{4}$$

where (i_1, i_2) is a multi-index spanning the product space of the i_1 and i_2 index. a (or b) labels a vocabulary of connection graphs available in a given generation n (or $n-1$). $\mathbf{P}$ is called the "propagator" and if its entries are sparse, it can encode a grammar by which a term (a nascent connection graph) indexed by a is replaced with a set of terms (smaller graphs) indexed by b. For $i_1 = j_1$ these graphs are the connections within a netlet (a module which often has a specific function); for off-diagonal blocks $i_1 \neq j_1$ they are bundles of connections between netlets.

Likewise, in the present model equation (2) may be used to encode a growth grammar. Each term in the grammar is represented by a unit direction vector in the cells' internal space, and a grammar rule is an entry in **T** which encodes a transition from one such vector to another.

To map an interpreted grammar into a neural net with our model, we may augment the mapping just described. The details will be presented elsewhere.

3 CONCLUSION

We have sketched a model with three levels of organization which allows for the use of genetic algorithms in neural networks. This is accomplished by interposing a level representing development between the neural net level, which supports computation, and the genetic level, where the effects of crossover and point mutation act. The resulting system is cost-effective and expressive.

We believe this model provides a framework which can be elaborated to describe important features of actual biological systems. A first step towards such elaboration has been taken for the development portion of the model. The model also provides a flexible means to explore the usefulness of ideas motivated by biology in computing.

References

[1] H. J. Antonisse and K. S. Keller. Genetic operators for high-level knowledge representation. In *Proceedings of the Second International Conference on Genetic Algorithms and their Applications.* Lawrence Erlbaum Associates, 1987.

[2] Eric Baum and David Haussler. What size net gives valid generalization? In *Neural Information Processing Systems 1*, page 81. Morgan Kaufmann, 1989.

[3] Leo W. Buss. *The Evolution of Individuality.* Princeton University Press, 1987.

[4] Scott E. Fahlman and Christian Lebiere. The cascade-correlation learning architecture. In *Neural Information Processing Systems 2*, pages 523–532. Morgan Kaufmann, 1990.

[5] J. D. Farmer, N. H. Packard, and A. S. Perelson. The immune system, adaptation and machine learning. *Physica D*, 22:187–204, 1986.

[6] J. Doyne Farmer. A rosetta stone for connectionism. *Physica D*, 42:153–187, 1990.

[7] S. Forrest. Implementing semantic network structures using the classifier system. In *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, 1985.

[8] T. Goto, P. Macdonald, and T. Maniatis. Early and late periodic patterns of *even-skipped* expression are controlled by distinct regulatory elements that respond to different spatial cues. *Cell*, 57:413–422, 1989.

[9] Stephen Jay Gould. *Ontogeny and Phylogeny.* Harvard University Press, 1977.

[10] Stephen Jose Hanson. Meiosis networks. In *Neural Information Processing Systems 2*, pages 533–541. Morgan Kaufmann, 1990.

[11] Kate Harding, Tim Hoey, Rahul Warrior, and Michael Levine. Autoregulatory and gap gene response elements of the *even-skipped* promoter of *drosophila. The EMBO Journal*, 8:1205–1212, 1989.

[12] Steven A. Harp, Tariq Samad, and Aloke Guha. The genetic synthesis of neural networks. Technical Report CSDD-89-I4852-2, Honeywell Corporate Systems Development Center, June 1989.

[13] John Holland. *Escaping Brittleness: The Possibilities of General Purpose Algorithms Applied to Parallel Rule-Based Systems*, chapter 20. Morgan Kaufmann, 1986.

[14] John Holland, Keith J. Holyoak, Richard E. Nisbett, and Paul R. Thagard. *Induction.* MIT Press, 1989.

[15] Klaus S. Lackner, Vernon D. Sandberg, and David H. Sharp. Data processing at the SSC with structured neural nets. Technical Report LA-UR-90-3774, Los Alamos National Laboratory, October 1990.

[16] Geoffrey F. Miller, Peter M. Todd, and Shailesh U. Hegde. Designing neural networks using genetic algorithms. In *Third International Conference on Genetic Algorithms*, pages 379–384, 1989.

[17] Eric Mjolsness, Charles Garrett, and Anand Rangarajan. A neural net for reconstruction of multiple curves with a visual grammar. Submitted to the International Joint Conference on Neural Networks '91 - Seattle., January 1991.

[18] Eric Mjolsness, David H. Sharp, and Bradley K. Alpert. Scaling, machine learning, and genetic neural nets. *Advances in Applied Mathematics*, 10:137–163, 1989.

[19] Eric Mjolsness, David H. Sharp, and John Reinitz. A connectionist model of development. Technical Report LA-UR-90-2327, Los Alamos National Laboratory, July 1990.

[20] A. S. Perelson. Immune network theory. *Immunolog. Rev.*, 110, 1989.

[21] Leslie Pick, Alexander Schier, Markus Affolter, Thomas Schmidt-Glenewinkel, and Walter J. Gehring. Analysis of the *ftz* upstream element: germ layer-specific enhancers are independently autoregulated. *Genes and Development*, 4:1224–1239, 1990.

[22] John Reinitz, Eric Mjolsness, and David H. Sharp. A connectionist model of the *drosophila* blastoderm. Technical Report LA-UR-90-3923, Los Alamos National Laboratory, November 1990.

[23] Manoel Fernando Tenorio and Wei-Tsih Lee. Self-organizing neural networks for the identification problem. In *Neural Information Processing Systems 1*, pages 57–64. Morgan Kaufmann, 1988.