Including control architecture in attribute grammar specifications of feedforward neural networks

Talib S. Hussain
Computing and Information Science Dept.
Queen's University
Kingston, Ontario
hussain@qucis.queensu.ca

Roger A. Browse
Computing and Information Science Dept.
Psychology Dept.
Queen's University, Kingston, Ontario
browse@qucis.queensu.ca

Abstract

An important problem in evolutionary computing is the design of genetic representations of neural networks that permit optimization of topology and learning characteristics. One promising approach for genetic representation of neural networks is the use of grammars to depict a process in which neural networks may be generated. Existing grammar representations of neural networks describe classes of networks with homogenous processing elements, simple fixed learning mechanisms and little organized topological structure. In previous research we have presented an attribute grammar representation for classes of networks with modular topology. Each parse tree generated by the grammar encodes a neural network specification which is subsequently executed by an interpreter. By expanding the grammar to include the control of the sequence of activity in the networks, we have been able to reduce the interpreter to a simple model of the operation of individual neurons in the networks. The expanded grammar may be used in an evolutionary computation to explore neural networks that vary in their architecture and learning behaviors.

1. Introduction

An important issue in the fields of neural networks and evolutionary computing is the development of methods for the specification of classes of neural networks. Neural network research could benefit from such techniques in that they may provide for improved theoretical analysis and comparison of network architectures. As well, such specification methods could lead to better scaling and the development of systematic relationships between problem decomposition and network modularity (Jacobs et al., 1991; Happel and Murre, 1994). Evolutionary computing could benefit from network specification methods in that they could provide a good basis for the encoding of topological and learning characteristics that are critical to the exploration of network solutions using genetic algorithms (Yao, 1993).

Recently, techniques have been developed that permit the use of grammars in the specification of classes of networks (Gruau, 1995; Kitano, 1990). One attractive feature of these systems is that each sequence of application of productions in the grammar coincides with a separate, but related neural network architecture. Thus, an evolutionary algorithm may search the space of possible architectures through its access to the rules of the grammar. However, these grammar-based approaches appear to have limited application in that they have thus far only been demonstrated to generate neural networks with fixed learning capabilities. Also, they have been limited to generating novel neural network architectures rather than the conventional architectures that have been useful in problem solving.

Cellular encoding (Gruau, 1995) is a grammar-based system whose productions all operate upon a single symbol type, termed a cell, and specify the expansion of those cells into connected groups of cells. The cells are considered nodes of the neural network once expansion is complete. While the idea of one network node expanding into several is intriguing, it does not permit the design of structured networks with distinctly different components.

This paper illustrates the use of attribute grammars for the genetic representation of neural networks (Hussain and Browse, 1998a, 1998b). We present an analysis of what makes the genetic representation of neural networks difficult, and propose a framework within which powerful new representations may be developed. We then present one particular attribute grammar which creates neural networks that are modular in structure and may vary

significantly in their learning behaviors. The advantages of our attribute grammar approach over cellular encoding are discussed.

2. Background

2.1 Neural interpreter complexity

Many different neural network algorithms have been proposed over the years. These models are mostly designed with the premise that a neural network consists of interconnected processing units, termed neurons or nodes, which process activation signals and update their internal structure based upon internal calculations (unsupervised networks) and/or upon externally provided feedback (supervised/reinforcement learning). However, between different models, there are great differences in the details of the networks, and in how those models are described. Thus, some networks, such as back-propagation, seem to be algorithmically simple, yet are actually quite complex when constructed with neurons that adhere to a strict model of neural computation (Hecht-Nielsen, 1990).

As a consequence of this great variation in network specifications, we face the dilemma that a representation must be selected which is powerful enough to encode all the important network characteristics are to be manipulated by the evolutionary process, yet which is still simple and structured enough to permit meaningful genetic manipulation.

A typical solution is to adopt a simple genetic description of a network and rely on a complex interpreter which is specific to one particular neural network architecture model to produce functioning individuals. The interpreter is an important yet often overlooked component of an evolutionary system for neural networks which has a strong relationship with the genetic description. The use of a complex interpreter keeps the genetic description simple, but it limits the range of neural network architectures that may be specified, and therefore searched.

Ideally, the interpreter should possess no more than a model of the operation of individual nodes within the neural networks. This permits the genetic representation great latitude in specifying classes of established networks, hybrid networks and even novel architectures without the introduction of boundaries between these classes within the genetic search.

2.2 Attribute grammars

Context-free grammars are known to be fairly limited in their power of representation. Researchers in evolutionary computation have focused on CFG-based representations since they are simple to design and produce parse trees which are amenable to genetic manipulation. However, in developing a complex genetic representation of a neural network that permits a generic interpreter, a more powerful form of grammar is needed. We propose that attribute grammars are the ideal solution since they have a context-sensitive component yet produce the same useful parse trees as CFGs.

An attribute grammar (Knuth, 1968) is a context-free grammar augmented by the assignment of attributes and attribute values to the symbols of the grammar. A production rule specifies not only the replacement of symbols, but also the evaluation of the attributes of those symbols. The attributes may be used to pass information downward (inherited) or upward (synthesized) through the parse tree.

2.3 Hierarchical grammar design

Adopting a more powerful grammar is only the first step towards solving our dilemma in selecting a representation. In addition, we require the representation to be able to develop genomes with complex internal composition in order to permit a simpler interpreter and fully exploit the context-sensitivity of the attribute grammar. However, a typical genetic programming (GP) representation is supposed to obey the closure property defined by Koza (1992) which states that all symbols in a genetic representation may be interchanged and still result in valid parse trees. All GP representations obeying this property, such as cellular encoding, may be thought of context-free grammars (CFGs) that are non-hierarchical in their productions.

Recently, several researchers have relaxed this closure property to permit genotypes with a more complex internal composition. In one approach, termed strongly-typed genetic programming (STGP) (Haynes et al, 1996; Montana, 1993), a genome is derived from a hierarchical CFG. STGP is designed upon the premise that a genotype may require an internal structure with non-interchangeable components. These components may be reflected as different non-terminal symbols in

the grammar. New genetic operators are consequently required which preserve the independence of these types, and thereby preserve the syntactic (and semantic) validity of the resulting offspring.

We propose that these same principles may be applied to the design of hierarchical attribute grammars. Such grammars should be able to generate specifications of all levels of a neural network architecture.

3. System design

Our evolutionary system uses an attribute grammar to specify a class of neural networks. Each parse tree generated from the grammar depicts an individual neural network. The values of the attributes that are computed within the parse tree encode the connections among the nodes of the network along with the characteristics of the operation of the nodes. Our current grammar collects this information in the attributes of the root symbol of the tree to form a concise neural network specification. The system's neural interpreter is able to accept this specification and carry out the functions of the network.

Labeled non-terminals in the parse tree represent distinct structural components, which may be replaced through the subsequent application of productions using that non-terminal as the starting symbol, or which may be replaced through substitution from some other network whose parse tree contains the same symbol. Genetic operators may thus be designed in a similar fashion to those of STGP. At any time, the performance level of the network may be determined by evaluating the attributes of the parse tree and providing the resultant specification to the interpreter.

3.1 Neural interpreter

It has been our goal to develop the use of attribute grammars to the point that a complete neural specification can be created which requires only a highly generic interpreter. The design of our current interpreter has required the adoption of an atypical neural framework which focuses not on biological plausibility nor on simplicity of representation, but on the flexibility of the representational approach. Primary features of the framework are a generalized model of a neuron and the enforcement of a purely localized structure.

In our system, a neuron is considered a processing element which may receive signals of multiple types and may transmit signals of multiple types (see Figure 1). It may include arbitrary internal memory and internal functions that process the incoming signals, modify internal memory and produce output signals. This is in contrast to the typical neuron model in which there is only one signal type - activation.

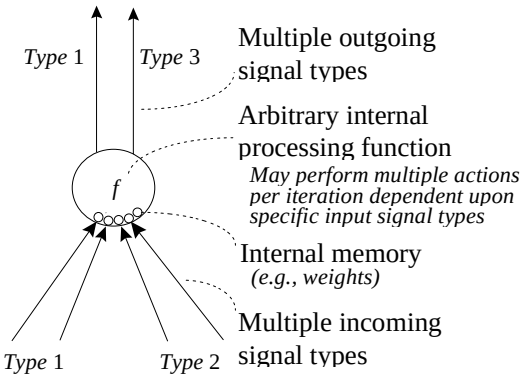

Figure 1: Basic neuron model

Further, we define a neural network to be a purely localized structure. Any computations that occur in the network must occur in the nodes, and all of these computations must be based solely upon information that a node receives as signals or stores in local memory. We shall term this information the domain of the node. Thus, for instance, all learning computations and operations must be performed by the nodes and must be local to the domain of each node.

4. Attribute grammar representation of neural networks

We present an attribute grammar (FFS_1) which represents a class of feedforward, supervised neural networks with learning. In FFS_1, we have extended our previous work (Hussain and Browse, 1998a, 1998b) and taken steps in the direction of a fully generic interpreter by excluding from the interpreter any requirements to perform network-level operations. The grammar itself includes nodes which control the stages of firing of levels of nodes within the network to ensure the proper sequence of feedforward and learning operations. These control nodes carry out a different operation than the processing nodes, but those differences are only instances of the

basic neuron model, and thus do not require special treatment by the neural interpreter.

The grammar uses three distinct types of non-terminals in order to incorporate the control behavior of the network directly into the grammar. Space precludes a complete description of the FFS_1, but the context-free portion of the entire grammar is given (see Figure 2), three productions are presented in detail (see Figure 3), and the meanings of the symbols are summarized (see Figure 4).

S → i o f N
Specifies a network's external I/O.
N → C M
Adds a control structure to a topology of basic nodes.
C → s C | t C | z_a | z_b
Specifies a topology of control nodes.
M → seq(**M**, **M**) | par(**M**, **M**) | **n**
Specifies a topology of basic nodes.

Figure 2: Synopsis of CFG portion of FFS_1

N → C M
(inherited)
C.*nodes_to_control* = **M**.*all_nodes*
C.*existing_controls* = {}
(synthesized)
N.*all_nodes* = **M**.*all_nodes* ∪ **C**.*new_nodes*
N.*in_nodes* = **M**.*in_nodes*
N.*out_nodes* = **M**.*out_nodes*
N.*visible_controls* = **C**.*visible_controls*
N.*connections* = **M**.*connections* ∪ **C**.*connections*

C_1 → s C_2
(inherited)
C_2.*nodes_to_control* = C_1.*nodes_to_control*
C_2.*existing_controls* = {**s**} ∪ C_1.*existing_controls*
(synthesized)
C_1.*visible_controls* = {**s**} ∪ C_2.*visible_controls*
C_1.*new_nodes* = {**s**} ∪ C_2.*new_nodes*
C_1.*connections* = C_2.*connections* ∪ {connect all 'S'-producing nodes in C_1.*existing_controls* to **s**} ∪ {connect all 'A'-producing nodes in C_1.*nodes_to_control* to **s**} ∪ {connect **s** to all 'F'-producing nodes in C_1.*nodes_to_control* }

C → z_a
(synthesized)
C.*visible_controls* = {**z_a**}
C.*new_nodes* = {**z_a**}
C.*connections* = {connect all 'S' and 'T'-producing nodes in **C**.*existing_controls* to **z_a**} ∪ {connect **z_a** to all nodes in **C**.*nodes_to_control*}

Figure 3: Three detailed productions of FFS_1

S	Start symbol: Describes a complete network.
N	Represents a functioning neural module.
M	Represents a topology of processing neurons
C	Represents a topology of control neurons.
i,**o**,**f**	Externally accessible input, output and feedback ports, respectively.
n	A node which computes 'A' signals until it receives a 'Z' signal; computes 'F' signals only when it receives an 'S' signal and until it receives a 'Z' signal; and modifies internal weights only when it receives a 'Z' signal.
s	A node which outputs an 'S' signal if none of its incoming 'A' signals have changed since the previous cycle.
t	A node which outputs a 'T' signal if none of its incoming 'F' signals have changed since the previous cycle.
z_a	A node which outputs a 'Z' signal if it receives both an 'S' and 'T' signal.
z_b	A node which always outputs a 'Z' signal.

Figure 4: Meanings of grammar symbols

Inspection of the grammar as presented above immediately reveals that the resulting performance of each node will depend upon what control signals it receives, and that the performance of the entire network will depend upon the control architecture produced by the grammar. For instance, a control structure with no **z_a** or **z_b** node will produce no learning, while one with a **z_b** node will produce continuous learning.

5. Conclusions

Our approach, as illustrated by FFS_1, allows a neural network topology to be represented through the application of attribute grammar rules and the evaluation of the attributes of the symbols in the resultant parse tree. This approach, in contrast to cellular encoding, exhibits several advantages. Firstly, it provides a clear representation of the structure of networks generated with the grammar. FFS_1 collects the entire specification of the network into a few attribute of the root. Cellular encoding, by contrast, never presents a concise neural representation. The user is required to perform a complex traversal of the parse tree in order to inspect what the parse tree represents.

Secondly, cellular encoding produces a small space of possible architectures. It only

permits one type of node and does not explicitly represent the learning characteristics of those nodes. Also, it requires a complex neural interpreter since all the details concerning what a node does and how it does it are not encoded in the grammar. In FFS_1, the specification of the control architecture by the grammar and the inclusion of multiple types of nodes permits a greater variety of network architectures to be modeled.

Finally, within cellular encoding, identical subtrees do not always expand in identical ways as they may be dependent on the context of the rest of the tree for the development of network connections. This characteristic emerges primarily from the application of cellular encoding's REC and WAIT productions, and affects the behavior of the genetic operators used to optimize genetic codes. In particular, crossover will not transfer fixed meaning in such a representation. FFS_1 introduces a stricter notion of semantic identity to the subtrees in the genetic representation and a more explicit notion of a constituent module.

There are a variety of simple ways in which a grammar such as FFS_1 may be modified to explore new possibilities. For instance, new productions which change the hierarchy or introduce new structures may easily be incorporated to extend or constrain the class of networks represented by the grammar. The attribute grammar can also be made slightly more interesting through the assignment of probabilities to the various production rules. During the random generation of an individual, the rules applicable to a given non-terminal under expansion may be selected probabilistically. This has the effect of biasing the generation process to encouraging certain structural forms in the final neural network. Note that the class of networks represented by the grammar is not affected, only the frequency with which certain networks are generated. In population-based genetic search, this has the effect of biasing the relative frequency of certain genes in the population.

Our research is continuing to extend the representational capabilities of our grammar, and we are exploring a complete attribute grammar for a localized version of the back-propagation algorithm, as presented by Hecht-Nielsen (1990). Our preliminary findings indicate that FFS_1 may be modified to produce back-propagation networks through changes solely to the grammar productions and terminal symbol characteristics. No changes to the neural interpreter are needed. The introduction of the control architecture into the grammar itself has reduced the role of the interpreter and has increased the network architecture details available to manipulation by the evolutionary process.

Acknowledgments

The research reported in this paper was conducted with financial support from the Natural Science and Engineering Research Council of Canada.

References

Gruau, F. (1995) "Automatic definition of modular neural networks," *Adaptive Behavior*, *3*, p. 151-183.

Happel, B.L.M. and Murre, J.M.J. (1994) "Design and evolution of modular neural network architectures," *Neural Networks*, *7*, p. 985-1004.

Haynes, T.D., Schoenefeld, D.A. and Wainwright, R.L. (1996) "Type inheritance in strongly typed genetic programming," Chapter 18 in K.E. Kinnear, Jr. and P.J. Angeline (Eds.), *Advances in Genetic Programming 2*. Cambridge, Mass: MIT Press.

Hecht-Nielsen, R. (1990) *Neurocomputing*. Reading, Mass.: Addison-Wesley.

Hussain, T.S. and Browse, R.A. (1998a) "Network generating attribute grammar encoding," *1998 IEEE International Joint Conference on Neural Networks*, p. 431-436.

Hussain, T.S. and Browse, R.A. (1998b) "Basic properties of attribute grammar encoding," *Late-Breaking Papers of the Third Annual Genetic Programming Conference.*

Jacobs, R.A., Jordan, M.I., Nowlan, S.J. and Hinton, G.E. (1991) "Adaptive mixtures of local experts," *Neural Computation*, *3*, p. 79-87.

Kitano, H. (1990) "Designing a neural network using genetic algorithm with graph generation system," *Complex Systems*, *4*, p.461-476.

Knuth, D. E. (1968) "The semantics of context-free languages," *Mathematical Systems Theory*, *2*, p. 127-145.

Koza, J.R. (1992) *Genetic Programming.* Cambridge, Mass: MIT Press.

Montana, D.J. (1993) "Strongly Typed Genetic Programming," *BBN Tech Report #7866*.

Yao, X. (1993) "Evolutionary artificial neural networks," *International Journal of Neural Systems*, *4*, p. 203-222.