Network Generating Attribute Grammar Encoding

Talib S. Hussain and Roger A. Browse
Queen's University, Kingston, ON, Canada

Appeared in 1998 IEEE International Joint Conference on Neural Networks (May 4-9, Anchorage, AK), vol 1, p.431-436.

Abstract

The development and theoretical analysis of neural network architectures may be improved with the availability of techniques which allow the systematic representation and generation of classes of architectures. Recent work on the genetic optimization of neural networks has led to new ideas on how to encode neural network architectures abstractly as grammars. Extending this approach, we have devised an encoding system that uses an attribute grammar in which the evaluation of both synthesized and inherited attributes within a generated parse tree provides the details of the connectivity of the network. Comparison with cellular encoding and the geometry-oriented variation of cellular encoding suggests that attribute grammar encoding is simpler, easier to use, and has more potential as a technique for effectively generating neural networks.

1. Introduction

Existing neural network architectures vary greatly in their structural form, learning styles, and functional characteristics. Comparative analyses of these architectures are often limited to empirical performance comparisons, usually involving only some architectures on a few tasks.

In the field of neural network research, there is a need for representational and analytic tools that integrate as broadly as possible across existing and possible architectures. Such tools would likely be of benefit to theoretical analysis, and in particular, an integrated representation that provided for the systematic generation of neural network architectures could have significance in the application of genetic optimization to neurocomputing systems.

Recent research in combining neural networks and genetic optimization has produced the novel method of encoding neural networks as the rules of a grammar [2-5,7-8].

This paper presents the Network Generating Attribute Grammar Encoding (NGAGE) technique. This technique employs the traditional concept of attribute grammars [1,6] as a means of depicting classes of neurocomputing architectures. The parse trees that result from applying the productions of these grammars represent individual network architectures in a way that identifies their major structural components.

We believe that the NGAGE technique will provide the basis for processes that will be capable of automatic generation, comparison, and search within classes of architectures, thereby simplifying neural network design efforts and facilitating the application of genetic algorithms.

2. Background

Cellular encoding [4,5] is a technique which represents neural networks as a tree of grammar transformations (encoding tree), and optimizes those trees using genetic programming operations. It provides a consistent framework for representing networks and a mechanism for automatic architecture generation. However, cellular encoding does not provide a straightforward relation between the encoding and the resultant networks, and it is somewhat awkward in its implementation. In particular, the network structure arising from the encoding tree is difficult to analyze since an identical sequence of transformations may produce highly different network components depending upon its location in the tree. Thus, cellular encoding does not provide a strong basis for theoretical comparisons between developed architectures and empirical analysis must be used.

Kodjabachian and Meyer [7] present research on a promising extension to cellular encoding which includes a high-level context-free grammar which imposes syntactic constraints upon the encoding tree. The restrictions constrain the class of architectures generated by the encoding and significantly decrease the genetic algorithm's search space. The approach has promise as an integrating framework in that different high level

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

constraints represent different classes of neural network architectures, and comparisons of constraints may be made. Within this approach, the relaxation of the highly restrictive constraints may lead to some of the same problems encountered in using the original cellular encoding technique.

The direction that we have taken in grammar-based encoding pursues similar objectives to the work of Gruau and Kodjabachian and Meyer. The representation power of attribute grammars allows the NGAGE system to avoid many of the difficulties of cellular encoding and to extend the utility of the approach.

An attribute grammar is an augmented context-free grammar in which each symbol in a grammar production has a set of attributes associated with it, and each production specifies how the attributes of its right hand symbols and/or left hand symbol are to be computed. A key benefit of attribute grammars which we wish to exploit is that they have a context-free component which allows for the straightforward generation of parse trees.

In a typical attribute grammar, an instance of the language is generated in three steps:

1. Assign attribute values to the starting symbol of the grammar, S, as necessary.
2. Beginning with S, apply a sequence of context-free productions of the grammar to obtain a parse tree.
3. Traverse the parse tree evaluating the attributes associated with each node in the tree. Inherited attributes are those whose values are passed downward in the parse tree, usually from values originally associated with the starting symbol. Synthesized attributes are computed based on attributes values of nodes that are deeper in the tree, usually originating from the terminal nodes.

3. NGAGE

The Network Generating Attribute Grammar Encoding (NGAGE) is based upon three main principles:

- A family of neural network architectures may be specified using an attribute grammar.
- A single neural network architecture may be represented as a sequence of grammar production rules. Thus, many architectures may be described using the same rules, with some using similar sequences of rules and some using highly different sequences.
- A specific neural network may be represented as a parse tree for which the attributes have been evaluated.

In what follows, the principles of NGAGE are demonstrated with a simple attribute grammar designed to reflect capabilities similar to those of basic cellular encoding. In particular, we develop our argument for neural network architectures without learning.

3.1 Sample attribute grammar

Using the notation of Alblas [1], the following is an attribute grammar for a subclass of layered feedforward networks (AG_1):

Non-terminals: S, M
Terminals: n
Start symbol: S
S initially has the attributes:
Inputs = $\{i_1,...i_x\}$
Outputs = $\{o_1,...,o_y\}$
Attributes: Inputs, Outputs, Nodes
Production and attribute evaluation rules:

start: S → M .
[(*inherited)*
Inputs **of** M := Inputs **of** S;
Outputs **of** M := Outputs **of** S;
]

n_seq: $M_0 \rightarrow M_1\ M_2$.
[(*inherited*)
Inputs **of** M_1 := Inputs **of** M_0;
Outputs **of** M_1 := Nodes **of** M_2;
Inputs **of** M_2 := Nodes **of** M_1;
Outputs **of** M_2 := Outputs **of** M_0;
(*synthesized*)
Nodes **of** M_0 := Nodes **of** $M_1 \cup$ Nodes **of** M_2;
]

n_par: $M_0 \rightarrow M_1\ M_2$.
[(*inherited*)
Inputs **of** M_1 := Inputs **of** M_0;
Outputs **of** M_1 := Outputs **of** M_0;
Inputs **of** M_2 := Inputs **of** M_0;
Outputs **of** M_2 := Outputs **of** M_0;
(*synthesized*)
Nodes **of** M_0 := Nodes **of** $M_1 \cup$ Nodes **of** M_2;
]

p_par: $M_0 \rightarrow M^L$.
[(*inherited*)
Inputs **of** M_j, j = 1..L := Inputs **of** M_0;
Outputs **of** M_j, j = 1..L := Outputs **of** M_0;
(*synthesized*)
Nodes **of** $M_0 := \cup$ {Nodes **of** M_j, j = 1..L };
]

t_sig: $M_0 \rightarrow n$.
[(*inherited*)
Inputs **of** n := Inputs **of** M_0;
Outputs **of** n := Outputs **of** M_0;
(*synthesized*)
Nodes **of** M_0 := {n};
]

where $\{i_1,...i_x\}$ and $\{o_1,...o_y\}$ denote the x input nodes and the y output nodes that exist independently of the derived network structure. The symbol n denotes a node created within the derivation. The value L shall be referred to as the *layer* parameter. Note that sub-scripts are only reference identifiers for the symbols within the productions, and not actually grammar symbols themselves. The use of the form <symbol>$^{<power>}$, as in the rule p_par, is not standard in attribute grammars. It represents the <symbol> repeated <power> times. The <power> term is consider constant for a given architecture. We refer to such a rule as a parameterised production rule.

The algorithm used in NGAGE to generate a neural network is the typical attribute grammar algorithm presented in section 2. Note that some productions of AG_1 are identical in their context-free component for simplicity of presentation. This property could be easily changed through the use of alternative non-terminal symbols and the inclusion of productions which map those symbols to the symbol M. In NGAGE, we further impose the *parameter rule constraint*, which requires the same sequence of production rules to be applied to every symbol that is generated from a parameterised rule. Thus, in the case of p_par, every symbol M_j is expanded in the same way.

3.2 Neural network encoding

The NGAGE system provides a representation of a neural network at several different levels.

First, the leaf nodes of the fully evaluated parse tree represent the resultant neural network. Each leaf depicts a neuron of the derived network and has, as a result of the evaluation of its attributes, a specification of all the other leaf nodes which form its Inputs and Outputs. Thus, the leaf nodes contain all the node and connectivity information required to construct a functional neural network. Figure 1 presents a sample neural network formed from the information in the leaf nodes of a parse tree generated using AG_1 with x = 3, y = 3, and *layer* = 4. All connections in the figure are directed from bottom to top.

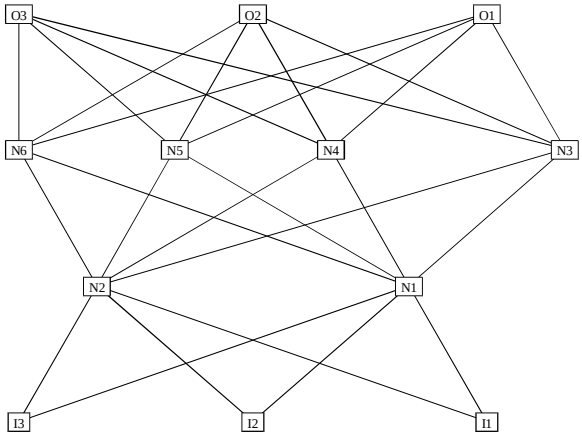

Figure 1: Sample Generated Neural Network

Second, the internal nodes of the final parse tree depict the structure of the derived neural network. For example, one internal node may have all the neurons in one layer as its descendants. Thus, even if the final network is very complex, its structure, as given by the internal nodes of the parse tree, may be very simple. The parse tree structure of the network from Figure 1 is presented in Figure 2.

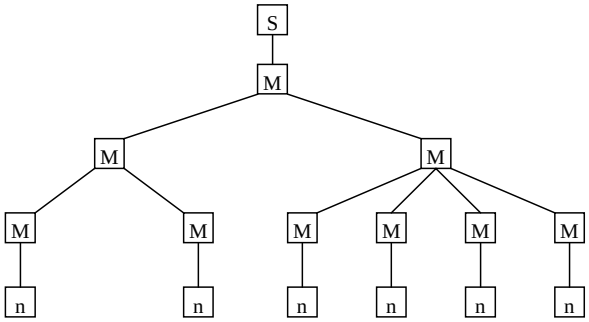

Figure 2: Sample Parse Tree

Third, the sequence of productions applied to the starting symbol represents a compact encoding of a network architecture. The rules may be represented as a *derivation tree*, in which the normal non-terminal rules (n_seq, n_par) have two children, the parameterised non-terminal rules (p_par) have only one child, and the terminal rules (t_sig) have no children. Through the use of a derivation tree as an encoding of a network architectures, search of architectures through genetic programming may be performed and comparisons of subtrees may be made to determine the structural similarity of different architectures. The derivation tree used to generate the network of Figure 1 is presented in Figure 3.

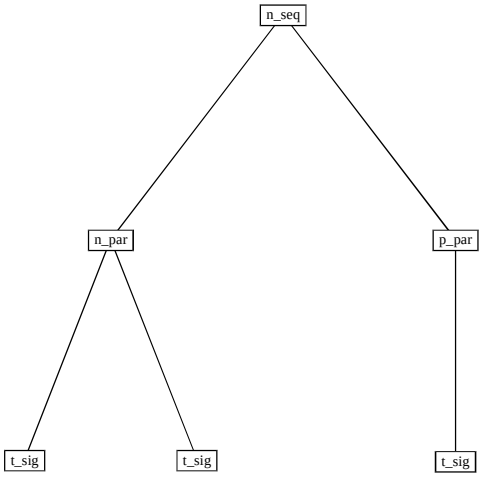

Figure 3: Sample Derivation Tree

Finally, a given NGAGE grammar, such as AG_1, represents a class of possible architectures. The interactions of the rules may be analyzed to determine the basic representation properties of all neural networks generated with that grammar. The family of architectures that is represented by AG_1 is discussed in the next section.

3.3 Implementation

The grammar and network generation code was programmed using MATLAB version 5.1. The network figures were generated automatically using GLAD, the Graph Layout Algorithm Display, developed at Queen's University.

4. Results

In this section, an analysis of the properties of the sample grammar AG_1 is presented, and a comparison between NGAGE and cellular encoding is made based on their usefulness as network generating and analysis techniques.

4.1 Representation properties

All networks generated with the AG_1 grammar will exhibit the following structural properties:

- Connected - All nodes will lie on a path between an external input and an external output; all input sites will have a path to at least one output site; and all output sites will have a path from at least one input site.
- Feedforward - The attribute evaluations of the rules do not allow for directed cycles in the final network.
- Feedforward Subclass - The grammar does not allow for the specification of all feedforward networks. Rather it represents an interesting subclass of feedforward networks in which nodes are either connected to all of the external inputs or none, and to all of the external outputs or to none.

As well, most neural networks architectures developed using AG_1 will have the tendency to be:

- Modular - Through the use of strict inheritance of connections and the application of n_par and p_par rules, networks may be developed with distinct structural modules. This is illustrated in Figure 4.
- Layered - Through the use of the parameterised production rule p_par, layered networks can easily be represented. This is also illustrated in Figure 4.

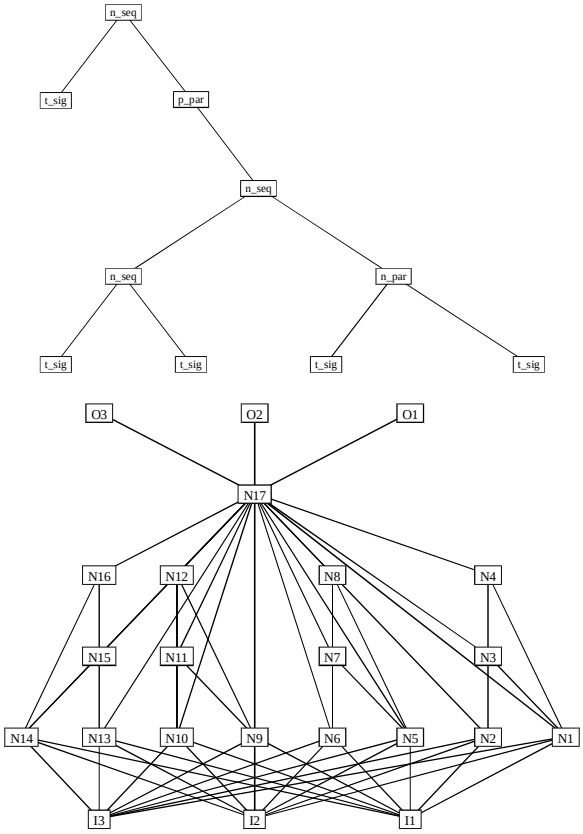

Figure 4: Derivation Tree and Generated Neural Network Exhibiting Modularity and Layering

4.2 Encoding properties

The encoding of an architecture in NGAGE, as reflected in the derivation tree of AG_1 productions, exhibits two properties that suggest it is a useful form of neural network representation.

Compactness:

A sequence of R rules of AG_1, excluding p_par, will produce networks which contain O(R) neurons and $O(R^2)$ connections. Thus, the architecture encoding, without p_par, is more compact than the final network by a square root factor.

A sequence of R rules of AG_1, including p_par, will produce networks which contain $O(L^R)$ neurons and $O(L^{2R})$ connections. Thus, the architecture encoding, with p_par, is more compact than the final network by an large exponential factor.

These results demonstrate that an attribute encoding of an architecture can provide significant savings in representation complexity. This result suggests that genetic optimization of NGAGE architecture encoding may be performed more efficiently and quickly than optimization of a direct network encoding.

Interpretability:

A second encoding property is a subjective one, and relates to how much insight the derivation can provide into the structural decomposition and functionality of the final neural network. We feel that the high compactness and potential for modularity provide the developer with a better idea of how the network decomposes its internal processing than a traditional network specification. In particular, in AG_1, the use of p_par can readily be interpreted as the construction of a layer of nodes in the final network.

The rule p_par using the *layer* parameter is somewhat analogous to the rule REC using the *life* parameter in Gruau's cellular encoding [4,5]. Both rules are used to indicate the repetition of structure in the network. However, p_par specifies a duplication to be applied to the subtree below the occurrence of p_par in the encoding tree, while REC specifies a duplication to be applied to the path leading down to the occurrence of REC in the encoding tree. We feel that the approach of p_par is more elegant and provides a representation of a repeated structure which is easier to extract from the tree.

4.3 Network generating properties

The NGAGE mechanism reflected by the AG_1 encoding exhibits two properties that suggests NGAGE is highly suitable as a tool for generating neural networks.

Consistency:

In the NGAGE system, the attribute grammar mechanism provides a fixed, consistent approach to how the encoding tree will generate a parse tree and how the attributes in that parse tree will be evaluated. As a result, the same subtree of production rules will be interpreted in exactly the same manner regardless of its location in the encoding tree, and it will give rise to structures that have similar structure - varying only with the differences in inherited attributes.

In cellular encoding, the encoding tree is interpreted in a dynamic fashion (Gruau allows looping and jumping within encoding tree [4]) and properties of cells are evaluated differently depending upon the order in which they are evaluated. As a result, the same sub-tree can be interpreted differently in different locations of the tree. Kodjabachian and Meyer's variation imposes a fixed traversal of the encoding tree, but the connectivity of the cells is still dependent upon the order in which the cells are evaluated. Thus, the same subtree will always be interpreted in the same way, but the connectivity that results may vary significantly over different locations of the subtree.

Distinctiveness:

In AG_1, the structural decomposition of the network is encoded distinctly through the placement of rules in the encoding tree. A subtree of rules tends to reflect a specific structural component. The NGAGE technique thus provides a representation that appears quite suitable for genetic optimization. The key benefit NGAGE has over cellular encoding in this respect is that cross-over operators are more likely to preserve useful structural decomposition of evolved networks since that structure is represented explicitly as subtrees in the encoding tree rather than implicitly as paths in the tree.

5. Conclusions

We have presented a new grammar encoding of neural networks which is simple to understand, easy to implement, and is potentially a powerful tool for generating and searching among neural network architectures. Future work shall involve designing more extended grammars which emphasize further the creation of hierarchical, modular structures, performing an empirical comparison between networks generated using NGAGE and cellular encoding, as well as formulating existing network architectures using the attribute grammar

encoding. Eventually, we expect that highly heterogeneous, complex networks may be created which will exhibit useful scaling properties.

As an example of how easily NGAGE may be extended to represent other classes of networks, consider the following grammar, AG_2, which generates a subclass of fully recurrent networks, such as the Hopfield network:

Non-terminals: S, H
Terminals: n
Start symbol: S
Attributes: Connect, Nodes
Production and attribute evaluation rules:

S → H
 [(*inherited*)
  Connect **of** H := Nodes **of** S;
  (*synthesized*)
  Nodes **of** S := Nodes **of** H;
 ]

H_0 → H_1 n
 [(*inherited*)
  Connect **of** H_1 := Connect **of** H_0;
  Connect **of** n := Connect **of** H_0 - {n};
  (*synthesized*)
  Nodes **of** H_0 := Nodes of $H_1 \cup$ {n};
 ]

H → n
 [(*inherited*)
  Connect **of** n := Connect **of** H - {n};
  (*synthesized*)
  Nodes of H := {n};
 ]

After a parse tree is generated from the rules of AG_2, the synthesized attribute 'Nodes' collects all the generated leaf nodes, permitting each leaf node to inherit the full set of nodes to which it should be connected, as the inherited attribute 'Connect'. The fact that each node is fully connected within the network is expressed locally in the productions that generate the leaf nodes. Global connectivity is accomplished through the evaluation of the attributes.

A key issue not addressed in this paper is that of encoding learning rules; we have focused primarily upon structural attributes. NGAGE will likely be able to represent the learning rules associated with the operation of the network nodes. Localized learning operations will be particularly straightforward, and more global strategies can be represented through the attribute structure.

A recent result on the behavior of genetic programming by Poli and Langdon [9] suggests that in the evolutionary optimization of trees in which upper level nodes represent high-level form, the higher-level structure of the solution is optimized first by the GP, and the details are optimized subsequently. Our future work will endeavor to determine if similar behavior is exhibited in the evolution of networks based on NGAGE. Such behavior, if present, will provide support for the automatic selection of architectures in NGAGE.

References

[1] Alblas, H. (1991) "Introduction to attribute grammars," *Attribute Grammars, Applications and Systems*. H. Alblas and B. Melichar (Eds.), New York: Springer-Verlag, p. 1-15.

[2] Friedrich, C.M. & Moraga, C. (1996) "An evolutionary method to find good building blocks for architectures of artificial neural networks," *Proceedings of the Sixth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, p. 951-956.

[3] Friedrich, C.M. & Moraga, C. (1997) "Using genetic engineering to find modular structures and activation functions for architectures of artificial neural networks," *Proceedings of the Fifth Fuzzy Days*, p. 150-161.

[4] Gruau, F. (1994) *Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm* Ph.D. Thesis, l'Ecole Normale Superieure de Lyon.

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

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

[7] Kodjabachian, J. & Meyer, J-A. (1997) "Evolution and development of modular control architectures for 1-D locomotion in six-legged animats," Available at www.biologie.ens.fr/perso/meyer/publications.html

[8] Luke, S. & Spector, L. (1996) "Evolving graphs and networks with edge encoding: Preliminary report," *Late-Breaking Papers of the Genetic Programming '96 Conference*.

[9] Poli, R. & Langdon, W.B. (1997) "An experimental analysis of schema creation, propagation and disruption in genetic programming," *Proceedings of the Seventh International Conference on Genetic Algorithms*, p.18-25.