

On Matching ANN Structure to Problem Domain Structure

George G. Lendaris,* Martin Zwick** and Karl Mathia***

*Professor of Systems Science and Electrical Engineering

**Professor of Systems Science

***Graduate Student, Electrical Engineering

Portland State University, P.O. Box 751, Portland, OR 97207

ABSTRACT

To achieve reduced training time and improved generalization with artificial neural networks (ANN, or NN), it is important to use a reduced complexity NN structure. A "problem" is defined by constraints among the variables describing it. If knowledge about these constraints could be obtained a priori, this could be used to reduce the complexity of the ANN before training it. Systems theory literature contains methods for determining and representing structural aspects of constrained data (these methods are herein called GSM, general systems method). The suggestion here is to use the GSM model of the given data as a pattern for modularizing a NN prior to training it. The present work assumes the GSM model for the given problem context has been determined (represented here in the form of Boolean functions of known decompositions). This means that certain information is available about constraint among the system variables, and is used to develop a modularized NN. The modularized NN and an equivalent general NN (full interconnect, feed-forward NN) are both trained on the same data. Various predictions are offered: 1) The general NN and the modularized NN will both learn the task, but the modularized NN will learn it faster. 2) If trained on an (appropriate) subset of possible inputs, the modularized NN will perform better generalization than the general NN. 3) If trained on a non-decomposable function of the same variables, the general NN will learn the task, but the modularized NN will not. All of these predictions are verified experimentally. Future work will explore more decomposition types and more general data types.

INTRODUCTION

There is substantial motivation to move away from using neural networks (NNs) with homogeneous architectures to NN structures that comprise modules of smaller neural networks. Motivations include issues of physical realizability of NNs, scalability of training time for NNs with large numbers of connections, and successful generalization. The modules would be smaller, and thus ease these concerns. The ability to modularize a NN is predicated upon the designer knowing (or discovering) enough about the problem domain to be able to decompose the given task into meaningful subtasks. A collateral assumption is that these subtasks will require fewer input and/or output signals, and therefore smaller NNs to implement them. The designer must also achieve a good enough understanding of the internal structure of the problem domain and the specific task being implemented (e.g., in a control context, the internal structure of the overall control system) to determine how to connect the NN modules so together they can successfully solve the larger task.

Research questions associated with such an endeavor include: what guidance can be developed for "parsing" a task/problem domain; how can the results of such parsing be used to design appropriate NN modules; how can training data for the (possibly artificial) sub-tasks be developed to train the NN modules; and if the modules are trained individually, how can such trained NN modules be put together, with perhaps additional training, for the larger NN to solve the original (whole) task?

In a previous paper [Lendaris & Todd, 1992], these issues were approached via a problem context for which the researcher has full knowledge of the task structure [fact retrieval in a knowledge base, where the structure of the concept-type hierarchy is known to be a lattice]. In the present paper, the approach explored is to employ structural modeling methods from the literature of systems theory to provide information to the designer about the structure of the problem domain, and this information is to be used to modularize a NN prior to training it.

GENERAL APPROACH

The method under exploration is that of using techniques developed within the general systems community over the last 10-20 years for representing structural aspects of system properties [Lendaris, 1980; Klir, 1985; Warfield, 1976]. One of these representation schemas promises to be particularly useful in suggesting reduced-complexity connectivity patterns for NNs. For convenience herein, this schema and associated notation is called the General System Method (GSM). It is an outgrowth of Ashby's earlier work in constraint analysis [Ashby, 1964] and a number of systems researchers have contributed to its development and use [e.g., Klir, 1976, 1985; Krippendorff, 1979, 1986; Conant, 1980, 1988; Cavallo, 1981; Broekstra, 1981].

As success in this endeavor is accumulated, the method would inject a new step in the normally ad hoc process of choosing the architecture/training algorithm for a NN application. The method is to be a constructive one, to consist of three activities: 1) analyze data from the problem domain using exploratory (as opposed to confirmatory) multivariate techniques [this step potentially could be accomplished using NNs in a different mode], 2) use the results to select a structural model representation of the data from the GSM lattice (described later), and 3) design a (reduced-complexity) connectivity pattern for the NN using the derived structural model of the data. The intent is to have the resulting structure of the NN mirror the structure of the data, and thus a kind of structure matching (or, constraint matching) will have been accomplished.

Various multivariate methods exist to perform exploratory structural analysis. These include Factor Analysis/ Principal Components Analysis, Multidimensional Scaling, Cluster Analysis, and Log-Linear Methods (which overlap substantially with information theoretic approaches). NNs have been shown to perform the equivalent of some of these methods. For some applications, the structure information obtained via such methods can be further refined, using, for example: Discriminant Analysis, Path Analysis, Covariance Structure Analysis, and Canonical Correlation Analysis.

BASIS OF APPROACH TO MODULARIZATION

The GSM notation (described below) that is used to represent structural attributes of constrained data, itself suggests a natural mechanism for transferring the constraint (structural) information into a modular design for a NN architecture. Notationally, for a constraint that is equivalent to a non-decomposable relation among a set of n variables, the GSM representation uses a square box with n lines attached (the lines may represent inputs or outputs, but pre-specification is not necessary), as shown in Figure 1a. If the constraint among the n variables were further decomposable, say into two non-decomposable relations of $(n-1)$ variables, then the representation consists of two boxes, as shown in Figure 1b. The lines connecting the two boxes correspond to the $(n-2)$ variables shared by the two sets of $(n-1)$ variables. The individual lines at the outside of each box represents the two non-common variables.

In GSM, a lattice is used to organize the large number of combinations of non-decomposable sub-relations among n variables [e.g., see Krippendorff, 1986, p.40]. The single node at the top of the lattice contains a box with n attached lines (all n variables independent), and the remainder of the lattice organizes all possible simpler structures, each representing a different possible constraint among the variables. Each level in the lattice contains all "descendants" of higher levels, where descent involves removal of one component (box) of a structure and restoration of any embedded relations in the component not already present in the remaining structure. The bottom of the lattice contains n boxes, each with 1 attached line (all n variables independent). For the present context, the suggested approach is to determine the lowest entry (simplest structure) in the lattice that describes (to an acceptable tolerance) the constraint in the given data. The collateral suggestion is to then use the GSM pattern selected from the lattice (interconnected boxes, with labeled connection lines), with minor modifications, as a template for designing a NN with substructures. The design principle is to 1) let each of the boxes in the selected GSM model represent a (general) sub-structure in the NN, 2) implement communication between the sub-structures via a synthetic variable that is introduced for each box, and 3) input these variables to an additional sub-structure introduced to accept these variables and to yield the system output variable(s). By virtue of this constraint matching process, the complexity of

the NN architecture can be reduced, and hence, the training process should be easier, and the potential for good generalization higher.

Nominal Data. For the case of qualitative (nominal, categorical) data, a constructive algorithm is used with GSM (using set- or information-theoretic measures of the variable combinations) to determine which entry in the GSM lattice is the desired one (typically, this means the simplest structure with acceptable reproduction of the data). The method, however, suffers the combinatoric problems associated with nominal data: a label must be stored for each state rather than exploiting a metric to summarize the states. As is well known, tree searches can be computationally expensive. Even for n as small as 10, it is doubtful that a direct, top down search is computationally feasible. Conant [1988] has developed a bottom up approach that incorporates heuristics, and has demonstrated success for n on the order of 100 on a microcomputer, and n of order 1000 may be possible for mainframes. This is encouraging.

The more usual case in NN applications is to have data that are ordinal, interval, or ratio scaled, rather than nominal. More information is inherently available in such data, and the previously mentioned multivariate tools offer significant computational power with which to extract structural information by exploiting the metric of the data. Such methods might allow, after calculating constraint information for the given data set, the use of this information to pick out (rather than search for) the corresponding set of structural models from the GSM lattice which are consistent with the data. Consideration of these ideas will be the subject of a future paper.

SUB-TASK EXPLORED IN THIS PAPER

To initiate this exploration, it was decided to examine a nominal-data causal system, represented in the form of a Boolean function. We start with a 3-input, 1-output system, as depicted in Figure 2. In GSM parlance, this is a 4-variable (ABCD) structure. The 4-variable structures used in this paper derive from [Zwick & Hsu, 1993], developed there in the context of structure and dynamics in elementary cellular automata (ECA). (In the ECA context, the 256 possible mappings are grouped into 88 equivalence classes and the number of different functions at various structure levels mentioned below refer to these classes). The lattice of 4-variable decompositions for such mappings comprises 6 levels, the ABCD structure being at the top level (not decomposable). The next level of complexity (level 5) is the decomposition type ABD/ACD/BCD, shown in Figure 3a using GSM notation. The next level of complexity (level 4) is the decomposition type ABD/ACD, shown in Figure 3b. (These are structural "types;" permutations of the variables generates a number of different but topologically equivalent specific structures.) For a causal system, we redraw Figure 3b as shown in Figure 3c. This latter schema motivates a way of modularizing a NN to learn/implement a level-4 mapping from ABC-->D (Figure 3d).

There are 7 different 3-variable Boolean functions at level 4. There are 47 such functions at the top (non-decomposable) level of the lattice. The 7 decomposable functions at level 4, and 7 non-decomposable functions from level 6 were selected for the first set of experiments. The second set of experiments deals with Boolean functions of 5 variables.

All experiments were carried out using a feed-forward architecture with backpropagation-of-error training. Two NN structure types were used: a) a full-interconnect (general, or non modularized) NN, comprising one hidden layer and a single output; and b) a modularized structure, patterned after Figure 3d. We used the simplest modularization possible, namely, keeping the same number of elements in the NN, and just partitioning the inputs to the hidden layer, rather than the full interconnect. A stylized representation of the NN structure used for the 5-input case is shown in Figure 4 (for this 5-input case, the general NN had 6 elements in the hidden layer, with full interconnect). Even this very simple modularization demonstrates the kinds of gains possible by the suggested process of patterning the NN structure after the GSM model of problem-domain structure.

The decomposable functions were used to train the general NN and the modularized NN. Both architectures successfully learned these decomposable functions. The prediction was that the NN whose structure already "matched" the problem-domain structure should be able to learn

the task with greater ease than the general NN. This prediction was confirmed, as the modularized NN learned the task(s) with significantly fewer training cycles.

A representative set of non-decomposable functions were used to train the general and modularized NNs. The general NN successfully learned the non-decomposable functions, and (as predicted) the modularized NN did not.

EXPERIMENTAL RESULTS

Since we are working with Boolean functions, the "desired" output for the NN is 1 or 0. The output of an element was judged successful for a 1 when it became $\geq .8$, and successful for a 0 when it became $\leq .2$. "Snapshots" of the NN performance were taken at various stages for each of the functions used for training. Their respective performance numbers were averaged, and these numbers used for subsequent comparisons.

A. THREE INPUT VARIABLES.

The 7 decomposable functions from level 4 were used to train both the general NN (4 hidden elements) and the modularized NNs, in the latter cases, taking care to hook up the correct variable as the shared one for each of the 7 functions (2 share A, 2 share C, and 3 share B). The learning performances are compared in Figure 5. As predicted, the modularized NN learned the tasks in substantially fewer cycles than did the general NN.

Seven non-decomposable (level-6) functions were selected, and these were also used to train both the general NN and the modularized NN. The averaged learning performances are compared in Figure 6. As predicted, the general NN learned the tasks, but the modularized NN was not able to do so (trained out to 100,000 cycles). A further prediction was that the general NN should find it easier to learn a decomposable function than to learn a non-decomposable function. Figures 5 & 6 show that it took the general NN approximately 1400 and 3000 cycles, respectively, to accomplish these tasks -- again confirming the prediction. [Three different random starting conditions were run for each function. Figures 5 & 6 are based on the best performance for each function.]

B. FIVE INPUT VARIABLES

To demonstrate whether the earlier results obtain in a more complex situation, experiments were carried out using a 5-input Boolean function. In GSM parlance, this corresponds to an ABCDEF structure. The decomposition lattice for ABCDEF structures is significantly larger than for the ABCD structures described earlier; the (ABCF/CDEF) structure was selected. A constructive method was developed to create Boolean functions with this kind of structure, and a few such functions were designed and used to train both a 5-input general NN and a 5-input modularized NN. A dramatically larger fraction of the Order(billion) 5-input Boolean functions are non-decomposable compared to the decomposable ones. Thus, the experimental procedure to define a non-decomposable function was to just pick a function randomly. Four such functions were used to train a 5-input general NN and a 5-input modularized NN (cf. Figure 4). The same random weight initialization was used for each experiment in the set; a second random initialization was used for a second set of experiments. This yielded 8 experiments for averaging, to compare with the 7 used in the 3-input case. As expected, the 5-input non-decomposable functions were not learned by the modularized NN. However, while all of the decomposable functions selected were learned by the modularized NN, in all the various experiments run, at least one of them were NOT learned by the general NN, even with the same starting weight values. This provides an even stronger result than for the 3-input case. Figure 7 shows the performance for these experiments; note that the general NN's average performance for the decomposable functions it did learn would be 11,000 cycles; also, the bottom curve reaches 100% at 40,000 cycles. A summary of the results is given in Figure 8.

GENERALIZATION

In discussions related to selection of data for training a NN, the advice usually offered is that one must choose a "representative" set of data. In some cases this is to be based on

intuitive/heuristic procedures, and in some others, based on statistical considerations. An example in the former category might be character recognition: in this case, we are to include examples of each of the characters to be recognized. In the second category, we are told to make sure that the pdf of the input data is appropriately sampled. In general, the idea is that to the extent the data actually captures the structure of the application domain, and to the extent the NN then captures the structure contained in the data, then the NN can be expected to generalize well, that is, to provide correct responses for inputs not seen during training.

The experiments described above provide a context for making explicit these ideas. So far, we have used all the possible input patterns during training, thus generalization was not at issue. However, since there is definite (known) structure in the functions being learned, we ought to be able to choose a "representative" subset of the possible input patterns for training the NN, and after it trains on this subset, expect the NN to generalize well -- in fact, we should be able to design an experiment to demonstrate 100% generalization, based on a substantially reduced training set.

Using the same method that was used to construct the decomposable in the first place, a training set comprising only 50% of the possible input patterns (i.e., just half of the mapping) was crafted that in principle contained all the information needed for a modularized NN to learn the entire function. This training set was used to train the general NN and the modularized NN. As predicted, after learning the training set, the modularized NN gave 100% generalization performance. Also as predicted, the general NN performed significantly poorer in its attempts to generalize -- average of 55% correct responses on the inputs not seen during training.

FUTURE WORK

The objective of the work reported above was to demonstrate that once we determine in the GSM lattice of decomposition types the lowest entry that describes (to an acceptable tolerance) the constraint in the given data, a beneficial modularizing of an NN can be accomplished using the corresponding GSM model as a template. For the above experiments, the "lowest entry" did not have to be discovered, rather, functions were constructed whose decompositions were known a priori. It was demonstrated that a modularized NN fashioned after the GSM model does indeed exhibit the properties expected when the NN's structure matches the problem domain structure. Further research will explore GSM structures with more than one shared variable, and with more than two modules. Assuming the exhibited properties continue to hold, there is great motivation to pursue the next (more difficult) step of developing a method for determining the appropriate GSM model that is to be used for modularizing the NN when the function underlying the data is not, as in the present case, already known. A particularly challenging context for application of these ideas will be problems based on interval or ratio scaled data -- a progression from the nominal type explored so far.

CITED REFERENCES

- Ashby, R., "Constraint Analysis of Many-Dimensional Relations," in Prog. in Bio-Cybernetics II, Wiener&Schade(eds.),'65.
- Broekstra, G. "C-Analysis of C-Structures," International Journal of General Systems, v17, pp33-61,'81.
- Cavallo, R. and G. Klir, "Reconstructability Analysis," International Journal of General Systems, v7, pp7-32,'81.
- Conant, R., "Structural Modeling Using a Simple Info Measure." Int Journal of Gen Systems, v11, no 6, pp721-730,'80.
- Conant, R., "Extended Dependency Analysis of Large Systems," International Journal of General Systems, v14, pp97-123,'88.
- Klir, G., "Identification of Generative Structures in Empirical Data," Int Journal of General Systems, v3, pp89-104,'76.
- Klir, G., Architecture of Systems Problem Solving, Plenum Press, 1985.
- Krippendorff, K., "On the Identification of Structures in Multivariate Data by the Spectral Analysis of Relations," Proceedings, 23rd Annual Meeting of the Society of General Systems Research, 1979.
- Krippendorff, K., Information Theory: Structural Models for Qualitative Data, Sage University Paper No. 62, SAGE, 1986.
- Lendaris, G., "Structural Modeling, A Tutorial Guide," IEEE Sys, Man, and Cybernetics, vSMC-10, no12, pp807-840, Dec'80.
- Lendaris, G. and D. Todd, "Use of Structured Problem Domain to Explore Development of Modularized Neural Networks," Proceedings of the IJCNN-92, Baltimore, pp III-869 to 874, IEEE, June, 1992
- Warfield, J.N., Societal Systems: Planning, Policy and Complexity, Wiley, 1976.
- Zwick, M. and H. Shu, "Reconstructability Analysis and the Dynamics of Elementary Cellular Automata," in prep 1993.

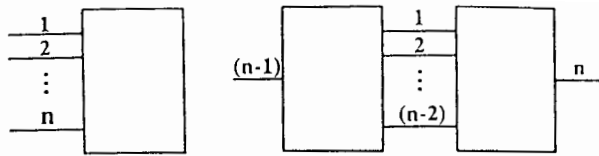


Figure 1. a) Non-decomposable relation on n variables, b) Two non-decomposable $(n-1)$ relations on n variables.

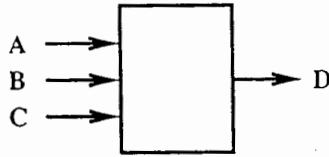


Figure 2. A 3-input, 1-output system. In GSM, called a 4-variable relation (ABCD).

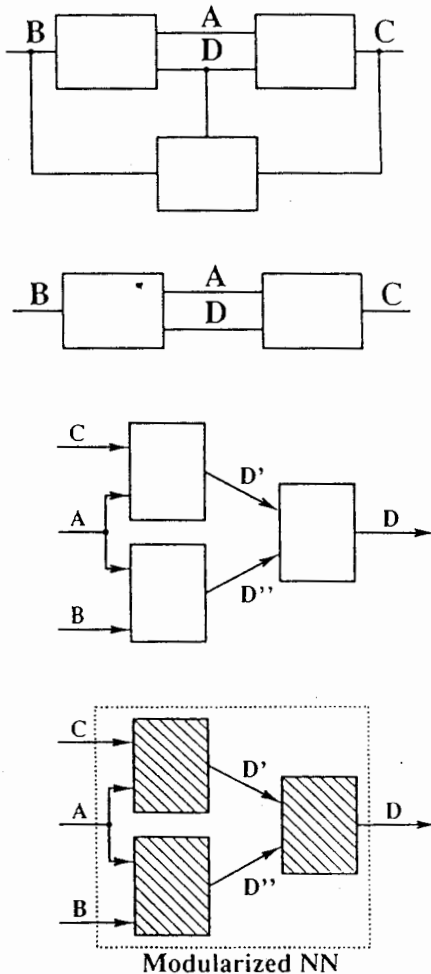


Figure 3. a) ABD/ACD/BCD decomposition type, b) ABD/ACD decomposition type, c) Causal version of ABD/ACD, d) Modularized NN implementing structure of 3c.

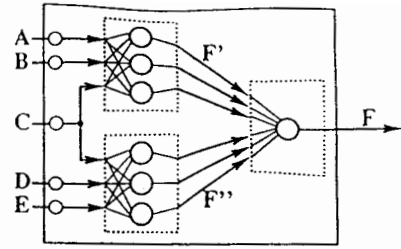


Figure 4. Stylized NN structure for 5-input decomposable function.

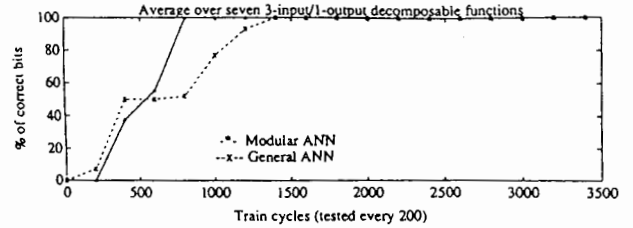


Figure 5. Train results for general NN and for modularized NN, averaged over 7 3-input decomposable functions.

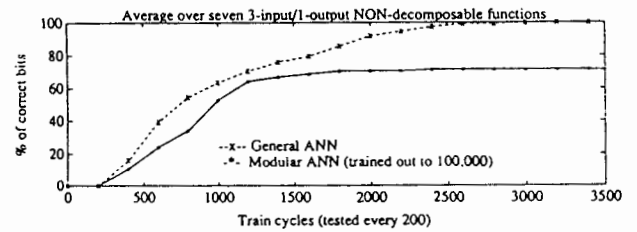


Figure 6. Train results for general NN and for modularized NN, averaged over 7 3-input non-decomposable functions.

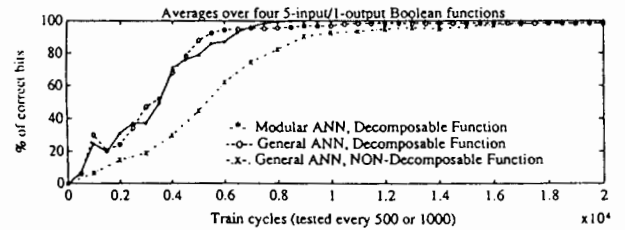


Figure 7. Train results for 5-input Boolean functions.

	3-Input Case	5-Input Case	
	# of Train Cycles to reach 100%	# of Train Cycles to reach 100%	# of functions NOT learned
Modular NN Decomp. Fn.	800	8,500	0
General NN Decomp. Fn.	1,400	11,000 (for those learned)	1
General NN NON-Decomp.	3,000	40,000	0

Figure 8. Summary of train results for 3-input and 5-input experiments.