

Using *A Priori* Knowledge to Prestructure ANNs

George G. Lendaris* and Karl Mathia**

*Professor of Systems Science & Electrical Engineering

**Ph.D. Candidate in Electrical Engineering

Portland State University

P.O. Box 751, Portland, OR 97207

email: [lendaris, mathiak]@sysc.pdx.edu

**Research Engineer, Accurate Automation Corp., Chattanooga, TN

Abstract

The objective of the present work is to develop a constructive method that uses certain *a priori* information about a problem domain to design the starting structure of an artificial neural network (ANN). The method explored is based on a general systems theory methodology (here called GSM) that calculates a kind of structural information of the problem domain via analyzing I/O pairs from that domain. A modularized ANN structure is developed based on the GSM information provided. The notion of performance subset (PS) of an ANN structure is described, and extensive experiments on 3-input, 1-output Boolean mappings indicate that the resulting modularized-ANN design is 'conservative' in the sense that the PS of the modularized ANN contains at least all the mappings included in the GSM category used to design the ANN. Partial experiments on 5-input, 1-output Boolean functions indicate further success. The extended experimental results also suggest the possibility of using a measure of the learning curve of specified ANNs on a series of (in this case Boolean) functions to serve as a proxy measure for the complexity of those functions. This proxy measure seems to correlate well with a measure known as Boolean Length. Determining a function's Boolean Length is a non-trivial undertaking; perhaps it will turn out that training an ANN on the function and measuring its learning experience will be a useful measure of function complexity, and easier to determine than the function's Boolean Length.

1 Background

Neural networks [more properly, *Artificial* neural networks (ANNs)] represent an emerging computational paradigm whose design is based on hints taken from the study of biological brain. The three main computational attributes of biological brain we aspire to emulate are 1) massive parallelism, 2) obtaining information from the environment via **learning**, and 3) good generalization. There is a broad range of problem types for which this new paradigm holds great promise, for example, pattern recognition, system modeling, signal processing, etc. A characteristic these problems have in common is that we typically do not have sufficient explicit knowledge about the problem to directly design a technological device to solve it, so for such applications, the learning attribute of ANNs is very important.

In ANNs, learning (or training) involves a basic structure and a process of making modifications to the structure's parameters. The structure comprises computing elements (here called neurodes) and their connections. A neurode receives n signals from the environment and/or from other neurodes, each through a path 'weight' parameter. A weight parameter multiplies the signal going through it, and all such products are

summed together by the neurode (often called **net** input). Following this, the neurode performs a (usually nonlinear) mapping of **net** to yield the neurode's output value (the latter mapping is often called a transfer function in the neurode). See Figure 1. In most learning/training methods suggested in the literature to date, the parameter adjusted during training is the path weight value, with the basic ANN structure and the neurode transfer functions remaining fixed. The present paper discusses a candidate methodology for using *a priori* information to **prestructure** the ANN in a way that contributes to 1) reducing the training time required, and 2) improving the generalization performance of the resulting trained ANN.

A useful characterization is to represent the ANN as shown in Figure 4a, where the "box" performs a mapping of the inputs to the outputs. Once the inputs and outputs are defined, *conceptually*, there exists a **set of all possible mappings (SAPM)** from the input domain to the output range [e.g., for an n -binary-input, 1-binary-output context, there are 2^{2^n} possible mappings]. For each ANN structure with a given setting of its weight values, the ANN will perform exactly one of these possible mappings. Doing the mental experiment of scanning all possible weight-value combinations in the given ANN, and collecting all the individual

mappings performed by the ANN, we call the resulting collection of mappings the ANN's **performance subset (PS)** [17][15]. (Even in the binary case, if the number of inputs to the ANN exceeds approximately 30, it would be physically impossible to build an ANN whose **PS** contains all $2^{2^{30}}$ mappings, hence the name **subset**.)

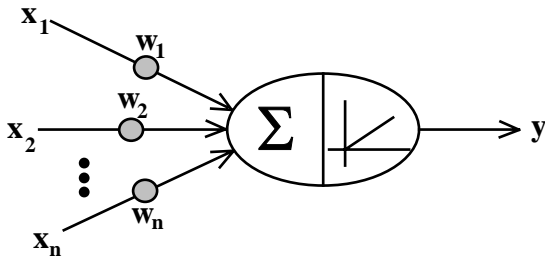


Figure 1. Basic neurode schematic.

In Figure 2a, we symbolize the set of all possible maps (**SAPM**) as the region defined by the outermost boundary, and symbolize the **PS** of some given ANN structure as the region defined by the inner boundary. In Figure 2b, we add a point **DM** to represent the mapping corresponding to a problem we wish solved (Desired Mapping). By our definition of **PS** as the collection of all those mappings it is possible for the given ANN structure to perform (over the set of all its weight values), **it is not possible** for the given ANN structure to perform the mapping **DM** shown in Figure 2b. Thus, no matter what weight-adjusting algorithm one attempts to use, it would be impossible for that ANN structure to ever learn the mapping at **DM**. So what is the ANN designer to do? Several strategies suggest themselves for what might be done either before training and/or during training: 1) "move" the point **DM** until it is inside the given region **PS** (Figure 2c), 2) "move" the region marked **PS** until it contains the desired mapping **DM** (Figure 2d), and/or 3) increase the size of **PS** until it contains the point **DM** (Figure 2e). Strategy 1 may be accomplished by the designer selecting a different representation schema for the inputs and/or outputs, and *de facto*, is accomplished any time a designer selects a representation of the problem such that the ANN structure the designer is working with successfully learns the desired mapping. Strategy 2 is accomplished by selecting a different ANN structure. The authors are aware of two references describing approaches (that appear to) use this strategy on line [13][20] (while this possible strategy was discussed even in the 1960's, a theoretical basis for such an approach is still in its infancy). Strategy 3 is exemplified by the variety of methods that "grow" the

starting ANN structure during training. To date, most training strategies assume that **DM** is already contained within the **PS** of the starting ANN structure (Figure 2f), and the job of the training algorithm is to converge upon **DM** -- indeed, typical convergence theorems state that a solution will be found *provided it exists*, and using the present vocabulary, this says *provided DM is contained in PS*.

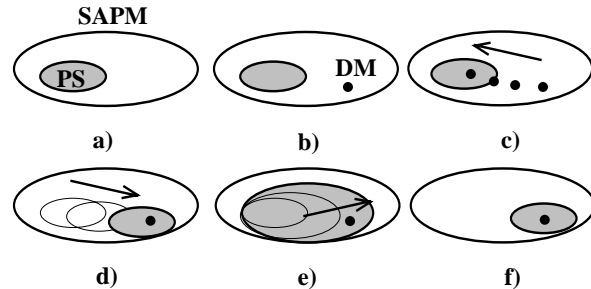


Figure 2. Set of all possible mappings (**SAPM**) and Performance Subset (**PS**) [shaded areas] (see text). The dot represents the desired mapping (**DM**).

The present paper is concerned with the possibility of using *a priori* information about the problem domain to constructively **prestructure** an ANN with assurance that its **PS** contains the desired mapping (Figure 2f), and in addition, with assurance that the size of its **PS** is relatively small. The reason for the latter desire is that if a given ANN structure learns the training data, then the smaller its **PS**, the better its chance for good generalization performance [17][6]. This latter property is the objective pursued by those methods that do weight 'pruning' [24][21]: they start with an ANN structure whose **PS** is large enough to assure inclusion of their **DM**, and then shrink the size of the **PS** in a principled way, making it smaller and smaller just to the stage before it no longer contains their **DM**.

In the quest to prestructure an ANN for the above stated reasons, there is motivation to move away from homogeneous structures to structures that comprise modules of smaller ANNs. Issues of concern include physical realizability of ANNs, scalability of training time for ANNs with large numbers of connections, and successful generalization. The modular approach entails smaller component ANN structures, hence easing these concerns. The ability to develop modularized ANNs is predicated upon the designer knowing (or discovering) enough about the problem domain to be able to decompose the given task into meaningful sub tasks. A collateral assumption is that these sub tasks will require fewer input and/or output signals, and therefore smaller structures to implement them. The designer must also achieve a good enough understand-

ing 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 ANN modules (and possibly other modules in hybrid systems) 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 ANN modules; how can training data for the (possibly artificial) sub-tasks be developed to train the ANN modules; and if the modules are trained individually, how can such trained ANN modules be put together, with perhaps additional training, for the larger ANN to solve the original (whole) task? In the present paper, the approach explored is to employ structural modeling methods from the literature of General Systems Theory to provide information to the designer about the structure of the problem domain, and this information is to be used to modularize an ANN prior to training with assurance that its **PS** contains the desired mapping DM.

2 General Approach

The method under exploration is based on techniques developed within the general systems community over the last 10-20 years for representing *structural* aspects of system properties [14][9][23]. One of these representation schemas promises to be particularly useful in suggesting reduced-complexity connectivity patterns for ANNs. For convenience herein, this schema and associated notation is called the General System Method (GSM). This method provides structural knowledge about a problem via a particular kind of (information theoretic) analysis of data from that problem domain. It is an outgrowth of Ashby's earlier work in constraint analysis [1] and a number of systems researchers have contributed to its development and use (e.g., [8][9][10][11][4][5][3][2]). 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 an ANN application.

The method is to be a constructive one, and 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 ANNs in a different mode], 2) use the results to select a structural model representation of the data from the GSM lattice of structures (described later), and 3) design a (reduced-complexity) connectivity pattern for the ANN using the derived

structural model of the data. The intent is to have the resulting structure of the ANN mirror the structure of the data, and thus a kind of *structure matching* (or, *constraint matching*) will have been accomplished. This avenue is intended to provide the desired assurance that the **performance subset (PS)** of the selected structure will contain the desired mapping DM (Figure 2f), and that the size of this **PS** is relatively small.

3 Basic Modularization Approach

The GSM notation 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 an ANN architecture. The notation is shown in Figure 3. 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). If the constraint among the n variables were further decomposable, say into two non-decomposable relations on $(n-1)$ variables each, then the representation consists of two boxes, as shown in Figure 3b. 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.

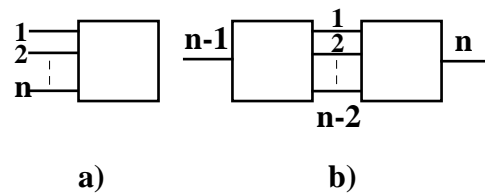


Figure 3. GSM Representation of Structures. **a)** Non-decomposable relation on n variables, decomposed into **b)** Two non-decomposable $(n-1)$ relations.

In GSM, a lattice is used to organize the large number of possible combinations of non-decomposable sub-relations among n variables [e.g., see [11], p.40]. The single node at the top of the lattice contains a box with n attached lines (all n variables dependent), 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 corresponds to a different degree of decomposition (hence a different level of complexity). 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 potential minor modifications, as a template for designing an ANN with substructures. The design principle is to 1) let each of the boxes in the selected GSM model represent a general (fully connected) sub-structure in the ANN, 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) [an example will be shown in Figure 4b]. By virtue of this constraint matching process, the complexity of the ANN architecture can be reduced, and hence, the training process should be easier, and the potential for good generalization higher.

3.1 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 [5] 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 ANN 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 for data beyond the nominal is the subject of future research.

4 Four-Variable Structure

The results reported here are based on an extensive set of experiments based on a four-variable (nominal-data) structure. In GSM notation, the latter is designated an ABCD structure. In our case, due to the input/output nature of our problem context, we impose the notion of causality, and consider this a 3-input, 1-output system, as shown schematically in Figure 4a. The data are all binary, thus this system is mathematically expressed as a Boolean function. The set of $2^{2^3} = 256$ possible Boolean functions (mappings) for such a system has been widely studied, and much is known about them. In the context of elementary cellular automata (ECA) for example, the 256 mappings are grouped into 88 equivalence classes [25]. This latter knowledge was used to select functions with known structural properties for the present ANN exploration.

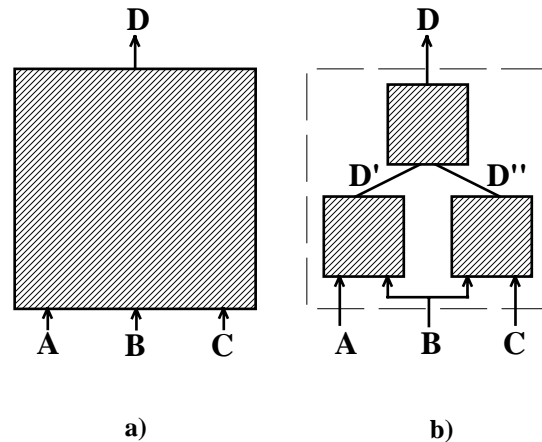


Figure 4. a) Non-decomposed and b) decomposed 3-input, 1-output system. The shaded boxes are implemented as separate ANNs.

The study focused on relations of two different *structural types*: 1) non-decomposable, and 2) decomposable into two relations with one shared variable. In GSM notation, type 1) is expressed as an ABCD structure, and type 2) as an ABD:ACD structure. We define D to represent the output variable of our causal system, and the ABD:ACD notation represents the case of A being the shared variable. Permutations of the input variables A, B, C generate a number of different but topologically equivalent mappings. In GSM, the ABCD structure corresponds to a relation of maximum complexity -- there is no reduction available (within this framework). The ABD:ACD structure, on the other hand, represents the case where there is a (partial) decoupling of variables possible: that is, ABCD may be decomposed into two sub-structures ABD and ACD which are not further decomposable. These two sub-structures are said to *share* variable A .

Consider now the task of designing an ANN to perform a 3-input, 1-output binary mapping. If nothing were known *a priori* about the specific mapping to be learned, then a typical candidate ANN structure to put into the box of Figure 4a would be a feed-forward type, with perhaps a single hidden layer, and **fully interconnected** [for present purposes, we call this a 'non-decomposed structure']. On the other hand, if *a priori* knowledge were available that the mapping to be learned was of the ABD:ACD type, then an ANN structure that would take into account such *a priori* information is shown in Figure 4b. In this case, we decompose the hidden layer into two sub-structures (the shaded boxes) which are not further decomposable. The number of inputs to each sub-structure is smaller than for the ANN of Figure 4a. [We call this a decomposed structure, or alternatively, a modularized structure.] For a 3-input system, with only 256 total possible mappings, this may seem trivial, but even moving up to just a 5-input system, the ANN related implications start becoming significant. For the 5-input case, the total number of possible maps is Order(billion)! As mentioned earlier, even for seemingly small numbers of inputs, it is physically not tractable to build ANNs whose performance subspace **PS** covers the entire set of possible mappings, so, any constraints discovered in the data that can be translated into correlated constraints on the ANN structure would be most welcome.

5 Experiments

A set of experiments was performed on the 256 possible 3-input, 1-output Boolean functions, using the partitioning into 88 equivalence classes mentioned in the previous section. A consistent exemplar of each of the classes was selected, yielding a set of 88 functions, and then 1760 experiments were run on these 88 functions [19]. Feedforward ANNs with back-propagation-of-error training were used. All experiments with each structure type used the same starting state, and the same training parameters. The key variable in the experiments were the different mappings to be learned. The training process was stopped at specified increments of training iterations, and the performance of the ANN was evaluated, via counting the number of bits of the output mapping that were correctly learned. The initial experiments reported in [18] focused on just those functions of the ABD:ACD decomposable type, all of which are of GSM structural type 4, and an equivalent number of *non*-decomposable functions (type ABCD), which are of GSM structural type 6. The examples from structural type 6 were selected intuitively to correspond in some plausible way to each

of the ABD:ACD functions of type 4. Those preliminary results (Figure 5) paved the way to the more extensive experiments described in this paper. These experiments included examples from all six structural types.

Figure 5. Training results for ANN and modularized ANN over seven 3-input, 1-output **a) decomposable** functions and **b) non-decomposable** functions.

First, experiments were run to determine the size hidden layer needed in the non-decomposed ANN structure to learn the examples taken from the ABCD class. We settled on a fully-connected, feed-forward structure with one hidden layer of 4 elements, and this led to a decomposed ANN structure (via removing selected connections) with each sub-structure in the hidden layer comprising 2 elements. The conjecture was that while the non-decomposed (more general) ANN would be able to learn all the mappings (i.e., both the ABCD types and the ABD:ACD types), the modularized ANN would not be able to learn the ABCD mappings. Further, it was conjectured that since the structure of the modularized ANN in some sense mirrored the known structure of the ABD:ACD mappings, the modularized ANN would be able to learn the ABD:ACD mappings 'more easily' than the more general ANN structure would, and (more importantly) it was conjectured that if the two structures were each trained on partial data from an ABD:ACD mapping, then the decomposed ANN would have better generalization performance than the more general ANN structure would (since its performance subset (**PS**) would be smaller).

Let us use the vocabulary introduced in Section 2 to discuss the 6 groupings of functions used in this study. As noted earlier, GSM structural type 6 refers to the most complex case, and thus these functions are expected to require the most general ANN structure, i.e.,

a fully connected one. The ABD:ACD functions used above are from type 4, and we have noted that the modularized ANN structure of Figure 4b works for these functions. The fully connected ANN structure selected has a performance subset (**PS**) that covers the entire set of 256 possible 3-input, 1-output Boolean mappings (as noted earlier, had the number of inputs been larger than approximately 30, this would not be physically possible). The modularized ANN structure used, however, has a smaller **PS**. The way the modularization was done in this case was to divide up the elements in the hidden layer equally to the two partitions. This can be considered in the same way as we discussed weight pruning earlier to infer that the **PS** of the modularized ANN is a reduced version of the more general ANN's **PS** (in this case, trivially so, since the larger **PS** includes the entire set of possible maps). A question of basic interest in the present research is how does the **PS** of the modularized ANN relate to the set of mappings associated with the various GSM structural types? We know that the **PS** of the modularized structure does not contain some of the mappings of type 6 (when we trained on those mappings that intuitively were among the "hardest" of these, the modularized ANN did not learn them). We also know that the **PS** of the modularized structure contains all of the mappings of type 4 (the ABD:ACD type), as it learned all of these. Since each structural type subsumes the lower types, we expect the **PS** of the modularized ANN structure designed according to structure type 4 requirements to include the mappings corresponding to the lower types. The experiments bore this out.

But, the full set of experiments show that the **PS** of the modularized ANN structure is in fact larger than just the mappings of GSM structural type 4. The modularized ANN was able to learn all the mappings of type 5, and further, some of the mappings of type 6. These results indicate that if we select a modularized ANN structure based on the GSM structural type inferred via GSM analysis of I/O pairs of data from the problem domain, then **the design is "conservative"** in the sense that its **PS** is at least big enough to contain the mappings of the inferred GSM structural type. The fact that the **PS** is larger than just the mappings contained in the inferred GSM set can be explained as follows: the mappings being explored are Boolean, i.e., all variables are binary. While the inputs to the ANN are binary, and the output neurode learns to give binary outputs, the hidden neurode values are not constrained (in our experiments) to binary values (these correspond to the synthetic variables D' and D" mentioned in Section 3). Accordingly, it is clear that the **PS** of the ANN prestructuring selected will be larger than the set

of mappings of the inferred GSM set. For the 3-input case the size of the ANN's **PS** reached up into structural type 6 (not all of it though). For a number of inputs n , larger than the 3 used here, the number of GSM structural types will be significantly larger than the 6 associated with the $n=3$ case.

Our analysis so far gives us hints suggesting the following speculation: the **PS** of the ANN designed via the GSM structural information (i.e. the selected structure in the GSM lattice) will contain functions within the "neighborhood" of the GSM structure identified. The term neighborhood here is intended to mean within approximately 2 levels further up from the one selected in the GSM lattice of structures. In the 4 variable case (3 inputs, 1 output), since the GSM lattice of structures is so small, the "neighborhood" reached into the top level of the lattice. However, for larger values of n , the GSM lattice will have significantly more levels, so the "neighborhood" could be a rather small fraction of the total range. Accordingly, the relative size of the prestructured ANN's **PS** will be a small fraction of the size of the collection of mappings up to the top of the lattice, and thus the difference between a general ANN's **PS** and that of a prestructured ANN will be greater, and thus the prestructuring will pay even better dividends than those already discussed for the $n=3$ case. We believe that the principle has been demonstrated, but there remains yet a significant amount of work to analyze even the 5-input case.

6 Generalization Experiment

In discussions related to selecting data for training an ANN, 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 of the former category might be character recognition: in this case, we know we should include at least some 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 ANN then captures the structure contained in the data, then the ANN 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, all the 3-input, 1-output experiments were carried out with a full training set, where the research objective was to observe the learning process. In these cases, generalization was

not at issue. But, when we do move on to consider the generalization question, the kind of knowledge available about the functions we are exploring is very useful. Since there is a definite (known) structure for the functions being learned, we are in a position to constructively design a "representative" subset of the possible input patterns for training the ANN that theoretically contains enough information to infer the entire mapping. After the ANN trains on this subset, to the extent that its structure really is tailored to the structure of the desired mapping, then we should expect the ANN to generalize well. The better the structural match, the closer to 100% generalization.

The 3-input case used for the rest of the experiments was judged too limiting for carrying out the desired generalization experiment, so we moved up to a 5-input, 1-output case. A set of four 5-input, 1-output functions, decomposable in a way indicated in Figure 6, were crafted for this experiment. By construction of these functions, we were able to select a training set comprising only 50% of the possible input patterns (i.e., just half the mapping) which we knew theoretically contained sufficient information from which to infer the total mapping. This set was used to train both a general (fully connected) 5-input ANN and a modularized 5-input ANN (cf. Figure 6). For the decomposed functions, both structures learned the training set perfectly. However, there was a big difference in the generalization tests: 1) the modularized ANN gave **perfect** responses for the I/O pairs not seen during training, and 2) the general ANN averaged only 55% correct responses (nearly random) on these test inputs. Also, four non-decomposable functions were selected, and both ANN structures trained on them, again using 50% of the possible input patterns. The general ANN learned the training set, while the modularized ANN did poorly. The modularized ANN did poorly at generalizing, and *so did the general ANN*. While these experiments used but a small fraction of the possible mappings in the 5-input, 1-output context, the experimental procedure of constructing a focused experiment and having this give results which support the hypothesis carries reasonable convincing power -- especially since it was constructed such that if the experiment gave negative results (counter-example), it would have significantly undermined the basic premise of the approach.

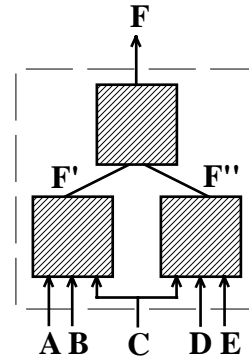


Figure 6. Modularized ANN, implementing a 5-input, 1-output ABCF:CDEF system.

A further observation from this experiment gives another encouraging datum. Let us recall from Section 1 the notion of **performance subset (PS)**, and in particular the size of **PS**. The general 5-input, 1-output ANN structure will have a relatively large **PS** (perhaps the entire set of 5-input, 1-output possible mappings), and the modularized ANN will have a **PS** whose size is smaller. We assume that both **PSs** contain the desired mapping DM, in this case the decomposable mapping(s) we crafted for the experiment. A well known phenomenon in training multi-layer perceptrons with the back-propagation-of-error algorithm is the tendency to get stuck in local minima. One of the pieces of folklore knowledge in the neural-networks arena may be paraphrased here as follows: the bigger the size of **PS**, the more complex the error surface in weight space, and hence the more likelihood of there being a local minimum ("valleys in the terrain") in which to get stuck. Stating this from a different perspective: a smaller relative size of the **PS** carries with it a smaller likelihood of there being a local minimum to get stuck in. The observation here was as follows: as expected, the modularized ANN was not able to learn the non-decomposable functions; however, while all of the decomposable functions selected were learned by the modularized ANN, in all the various experiments run, at least one of them were **NOT** learned by the general ANN, even with the same starting weight values. Based on such a small sample size, we can only speculate that the modularized ANN benefited from the reduction in the size of its **PS** in avoiding the local minima the general ANN got stuck in. If further research bears this out, then there is another substantial motivator at hand for using modularized ANNs of the type suggested here -- namely, those whose **PS** is assured to contain the desired mapping, and further, whose **PS** is of relatively small size.

Figure 7. Train results for four 5-input, 1-output Boolean functions (train set used 50% of each resp. mapping). Generalization results cited in text.

7 Proxy Measures for Complexity?

The 88 functions studied in these experiments were sorted according to the 'learning difficulty' experienced by the 1) fully-interconnected ANN and 2) the modularized ANN. In addition, the same functions were sorted according to four different measures in the literature dealing with Boolean functions, namely, the already discussed GSM structural type (range 1-6) [11], and in addition the Lambda-count (range 0-4) [11], Fluency (range 1-9) (theory developed in [25], original idea by Ashby[22]), and Boolean Length (range 1-10) -- this is the minimum number of Boolean operations necessary to represent the binary string (related to Kolmogorov complexity) [25]. Values of these four complexity measures for the 88 Boolean functions selected for our experiments appear in [25]. In comparing these sorted lists, except at the two ends (where all but Lambda basically concurred), it turned out that there was little, if any, correlation between the orderings given by the 4 published measures of complexity. However, there was a good correlation in the orderings provided by the Boolean Length measure and the 'learning difficulty' assigned to each of the ANN structures investigated. The suggestion based on these observations is that certain measures of the learning curve of specified ANN structures might be used as a proxy measure for the complexity of certain classes of functions [19]. The learning curve may be characterized by (at least) two of its attributes: the transient portion and the steady state level. Here the steady state level can be characterized by the number of bits learned (max. of 8 for the 3-input, 1-output case), and the transient by the number of training cycles required by the ANN structure to accomplish the learning. Other qualities of the transient suggest themselves via visual analysis, but have not yet been reduced to quantitative expressions. While the investigation here was based upon Boolean functions that have well documented properties in the literature, there might be a basis for developing this approach to

other classes of functions. Thus we have a turn of events. Instead of lamenting the difficulty an ANN has in learning a given task, we might be able to use the learning experience of specified ANN structures as proxy measures for the complexity of certain classes of functions.

8 Related Extensions

These considerations suggest yet another extension. As was mentioned earlier, the research reported here assumes that GSM structural analysis would pick out of the GSM lattice of structures a structure that (best, in some sense) captures the structural information available from the given I/O data pairs. However, it was also noted that this GSM process suffers a combinatorial problem in searching the lattice. Proposed heuristic methods offer some relief [5]. Turning our perspective around, we could think in terms of creating a collection of prototype ANN modularized structures, one for each layer in the lattice (some ANN structures may cover more than one layer in the lattice). In a set of controlled experiments, one could catalog the learning curves for each prototype ANN structure for a range of functions selected from above and below the prototype's position in the lattice. The functions from below would result in an "ease-of-learning" characteristic, and the functions from above would result in a "difficulty (or impossible)-to-learn" characteristic. Then, develop a strategy to use the set of prototype ANN structures to determine which level in the GSM lattice best represents the data being studied. This strategy would include observing the learning curve of selected prototype ANN structures, and use this information to decide whether to move up or down the lattice. A bracketing kind of strategy could be appropriate. It could turn out that this way of finding the desired GSM lattice level could be done in a more tractable way than is presently possible via extant GSM methods. Further, this could provide a two-stage process for ANN development. Use this hypothesized strategy to do preliminary tests on the data using ANNs in order to pick out which ANN structure to then commit to more extensive training for developing a solution to the problem at hand.

8 Conclusion

To recapitulate, the assumption is that we start with a set of I/O data for the problem domain, specifically here, binary I/O data. Next, we submit this data to what we are calling GSM structural analysis, and this analysis assigns a structural type to the data (ranging from type 6 to type 1 for the 4-variable case). This

structure number is a kind of complexity measure (6 being the most complex), relating to the decomposability of the Boolean function inferred to underlie the given data. This yields a rather coarse coding of the possible 256 functions being considered, and as might be expected, within each category, there will be a gradation of the degree of complexity, if we had a finer way to measure it. Nevertheless, the premise here is that even this rather coarse measure can be put to good use in modularizing ANNs -- where 'good' here relates to improved learning speed, and more importantly, improved generalization performance

9 References

- [1.] Ashby, R., "Constraint Analysis of Many-Dimensional Relations," in *Proc. in Bio-Cybernetics II*, Wiener & Shacde (eds.), 1965.
- [2.] Broekstra, G., "C-Analysis of C-Structures," *Internat Jour of Gen Systems*, vol 7, pp. 33-61, 1981.
- [3.] Cavallo, R. & G. Klir, "Reconstructability Analysis," *Internat Jour of Gen Sys*, vol 7, pp7-32, 1981.
- [4.] Conant, R., "Structural Modeling Using a Simple Information Measure," *Internat Jour of Gen Sys*, vol 6, pp. 721-730, 1980.
- [5.] Conant, R., "Extended Dependency Analysis of Large Systems," *Internat Jour of Gen Sys*, vol 14, pp. 97-123, 1988.
- [6.] Hertz, J., Krogh, A. & R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, 1991.
- [7.] Kalman, B.L., Kwasny, S.C. & A. Abella, "Decomposing Input Patterns to Facilitate Training," *Proceedings of World Congress on Neural Networks Portland (WCNN-93)*, pp. III.503-III.506 Earlbaum, July, 1993.
- [8.] Klir, G., "Identification of Generative Structures in Empirical Data," *Internat Jour of Gen Sys*, vol 3, pp. 89-104, 1976.
- [9.] Klir, G., *Architecture of Systems Problem Solving*, Plenum Press, 1985.
- [10.] 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.
- [11.] Krippendorff, K., *Information Theory: Structural Models for Qualitative Data*, Sage University Paper No. 62, SAGE, 1984.
- [12.] Langton C.G., Taylor C. , Farmer J.D. & S. Rasmussen (eds.), *Artificial Life II*, Addison-Wesley, 1992.
- [13.] Lee, T-C, *Structure Level Adaptation for Artificial Neural Networks*, Bluer Academic, 1991.
- [14.] Lendaris, G.G., "Structural Modeling, A Tutorial Guide," *IEEE Systems, Man and Cybernetics*, vol SMC-10, no12, pp. 807-840, Dec. 1980.
- [15.] Lendaris, G.G., "A Proposal for Indicating Quality of Generalization when Evaluating ANNs," *Proc. of IJCNN 1990*, San Diego, IEEE, 1990.
- [16.] Lendaris, G.G. & D. Todd, "Use of Structured Problem Domain to Explore Development of Modularized Neural Networks," *Proc. of the IJCNN-92, Baltimore*, pp. III-869-874, IEEE, June 1992.
- [17.] Lendaris, G.G. & G.L. Stanley, 1965, "Structure and Constraint in Discrete Adaptive Networks," *Proc. of the Nat Electronics Conf*, vol. XXI, pp. 500-505.
- [18.] Lendaris, G.G., M. Zwick & K. Mathia, "On Matching ANN Structure to Problem Domain Structure," *Proc. of the World Congress on Neural Networks - WCNN-93 Portland*, Earlbaum/INNS,1993
- [19.] Mathia, K., "Under What Conditions Do Modularized Neural Networks Learn Boolean Functions?," Project Report SySc 651, Portland State University, Portland, Oregon, March 1994.
- [20.] Odrj, S.V., Petrovacki, D.P. & G.A. Krstonosic, "Evolutional Development of a Multilevel Neural Network," *Neural Networks*, vol. 6, pp. 583-595, Pergamon, 1993.
- [21.] Reed, R., "Pruning Algorithms--A Survey," *IEEE Trans. on Neural Networks*, vol. 4, no 5, Sept 1993, pp. 740-747.
- [22.] Walker, C.C. & W.R. Ashby, "On Temporal Characteristics of Behavior in Certain Complex Systems," *Kybernetik*, vol. 3, no. 2, pp. 100-108, 1966. [Reprinted in *Mechanisms of Intelligence: Ross Ashby's Writings on Cybernetics*, R. Conant (Ed), Intersystems Publications, pp.93 -110, 1981.
- [23.] Warfield, J.N. *Societal Systems: Planning, Policy and Complexity*, Wiley, 1976.
- [24.] Weigend, A.S., Rumelhart, D.E. & B.A. Huberman, "Back Propagation, weight-elimination and time series prediction," in *Proc. 1990 Connectionist Models Summer School*, D.Touretzky, J. Elman, T Sejnowski & G. Hinton, Eds., 1990, pp. 105-116.
- [25.] Zwick, M., Shu, H. & B. Gifford, "Complexity and Dynamics