

Prestructuring Neural Networks via Extended Dependency Analysis with Application to Pattern Classification

George G. Lendaris, Thaddeus T. Shannon, Martin Zwick
Systems Science Ph.D. Program, Portland State University, PO Box 751, Portland, OR

ABSTRACT

We consider the problem of matching domain-specific statistical structure to neural-network (NN) architecture. In past work we have considered this problem in the function approximation context; here we consider the pattern classification context. General Systems Methodology tools for finding problem-domain structure suffer exponential scaling of computation with respect to the number of variables considered. Therefore we introduce the use of Extended Dependency Analysis (EDA), which scales only polynomially in the number of variables, for the desired analysis. Based on EDA, we demonstrate a number of NN pre-structuring techniques applicable for building neural classifiers. An example is provided in which EDA results in significant dimension reduction of the input space, as well as capability for direct design of an NN classifier.

Keywords: neural networks, structure, pattern recognition, classifier, information theoretic reconstructability, extended dependency analysis, optical Fourier Transform.

1. INTRODUCTION

In any problem domain, the data associated with a given “problem” is not purely random, else the data is not informative. The term ‘structure’ is used by the statistics community to represent certain information about the non-random quality of the data. In the context of parametric statistics, a form of underlying model (i.e., a ‘structure’) is assumed, and procedures are followed to determine values for the parameters of the model to refine its fit to the data. In non-parametric statistics, while looser assumptions are made, the bottom line still is to discover a good structural representation of the data.

For the neural network (NN) context, we take the term ‘structure’ to mean the connection pattern and the number and type of neural elements. Quite aside from the learning aspects of an NN, its structure directly influences the NN’s computational capabilities. For example, in an input-output context, the NN’s structure defines the set of I/O mappings it is possible to implement with that NN as each of the possible combination of weight settings is instantiated.

While these two usages of the term structure are from two distinct domains of inquiry -- that of the statistician and that of the NN practitioner -- the *underlying* meaning of the term is really the same in both cases. Our challenge is to bring these two usages together in a pragmatic way, i.e., to effect a ‘structure matching’.

Assume we have a defined input domain (**I**) and output range (**O**), along with a set of training I/O pairs that (adequately) represent a “desired mapping” (DM) in a given problem domain. Further, assume for a moment that we successfully attain a reasonable structural model of the I/O data in the statistician’s sense. **Our key objective is to develop procedures one could follow to take the statistician’s structural information and use it to (pre)select an NN structure whose Performance Subset (described below) has the attributes of being big enough but not too big, AND for which there is assurance it contains the desired mapping.**

We describe such a procedure, including some prior results plus extensions reported here, and present results of applying the suggestions to a pattern classification problem.

2. COMMENTS ABOUT NNs

Consider the NN as a “black box” that performs a mapping of its inputs to its outputs. Once the input domain and output range are defined, conceptually, there exists a set of all possible mappings (**SAPM**) from one to the other. An NN structure with a given setting of its weight values will perform exactly one of these possible mappings. Doing the mental experiment of scanning all possible weight-value combinations in the given NN, and collecting all the individual mappings performed by the NN, we call the resulting collection of mappings the NN’s Performance Subset (**PS**) [22] [21].

In Fig. 1a, we symbolize the set of all possible maps **SAPM(I,O)** as the region defined by the outermost boundary, and symbolize the PS of some given NN structure as the region defined by the inner boundary (shaded area). In Fig. 1b, 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 mappings it is possible for a given NN structure to perform (over the set of all its weight values), it is not possible for the given NN structure to perform the mapping DM shown in Fig. 1b. Thus, no matter what weight-adjusting algorithm one uses, it would be impossible for that NN structure to ever learn the mapping at DM.

Several strategies exist for what might be done either before training and/or during training:

- 1) “move” the point DM until it is inside the given region of the PS (Fig. 1c),
- 2) “move” the region marked PS until it contains the desired mapping DM (Fig. 1d), and/or
- 3) increase the size of PS until it contains the point DM (Fig. 1e).

Strategy 1 corresponds to selecting a different representation schema for the inputs and /or outputs. Strategy 2 is accomplished by selecting a different NN structure. Two references describing approaches that (appear to) use this strategy on line are [13] and [24]. Strategy 3 is exemplified by the variety of methods that “grow” the starting NN structure during training. To date, most training strategies assume that DM is already contained within the PS of the starting NN structure (Fig.1f), and the job of the training algorithm is to converge upon the DM – indeed, typical convergence theorems state that a solution will be found *provided it exists*, and using the present vocabulary, this says *provided the DM is contained within the NN’s PS*.

While a seemingly safe design strategy would be to choose a structure with a very large PS, as this would increase the likelihood of it containing the DM associated with the given problem, some key difficulties would follow:

- 1) **Quantity-of-data issues:** larger PS corresponds to more adjustable parameters, and this entails requirement for larger training data sets; in many contexts, training data is limited.
- 2) **Training difficulties:** as the number of adjustable parameters increase, the time and difficulty of training increase dramatically.
- 3) **Quality-of-generalization issues:** if two NN structures learn the training data equally well, then the structure with the smaller PS has a better chance of generalizing better [22][8].

It follows that a better design strategy would be to choose a structure whose PS is big enough to contain the DM, but not much bigger, i.e., “big enough, but not too big”. While this statement is easy to make theoretically, from a pragmatic point of view, there are huge difficulties. The key difficulty lies in *knowing* how to choose a structure with assurance its PS contains the DM, AND that the PS meets this size constraint. The methods suggested herein provide such knowledge.

As an aside, the MLP structure (fully-connected feed-forward connection pattern with a single hidden layer of elements, and neural elements with a sigmoid activation function) is known to be a Universal Function Approximator []. In principle, with an infinite number of neural elements in the hidden layer, the Performance Subset (PS) of such a structure would comprise the entire SAPM for a given **I** and **O**. While this might sound appealing, all three of the difficulties listed above apply. But even aside from this, for any *practical* finite number of elements in the hidden layer, the corresponding PS will be substantially smaller than the SAPM, so the considerations illustrated in Fig. 1 must be addressed.

Our interest is in the possibility of using *a priori* structural information about the problem domain to constructively pre-structure an NN with assurance that its PS contains the desired mapping (Fig. 1f), while the overall size of its PS is as small as possible. The reason for the latter desire is that if a given NN structure learns the training data, then the smaller its PS, the better its chance for good generalization performance [22][8]. This latter property is indirectly achieved by those methods that do weight ‘pruning’ [25] [27]; they start with a NN structure whose PS is large enough to assure the 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 DM. While the results may be the same, the mechanics of our approach are different.

While the above discussion is based on the view that the NN is performing a *mapping*, there are problem contexts in which *our* thinking characterizes the problem domain in terms of *decision making* -- for example, in the arena of pattern classification. When an NN is configured and trained to perform a pattern classification task, computationally, the NN itself is still doing whatever it does; however, how *we* think about the problem *as designers* changes how we embed what we know about the problem into the NN structure. In a problem domain in which our characterization is in terms of function approximation, this influences the way we define the representation of **I** and **O** that serve as inputs/outputs for the NN, and in addition how we craft the criterion function we use to assess the performance of the NN (e.g., during the training process). When we characterize a problem domain as one of pattern classification, clearly, the representations of **I** and **O**, and the performance criterion function stand to be substantially different. Accordingly, our design activities would proceed differently. The methodologies described in the sequel may be tailored to both situations.

3. GENERAL APPROACH

The methods described here are based upon techniques developed within the general systems community over the last 10-20 years for representing structural aspects of system properties [26][14][10]. One of the representation schemas available in the general systems literature has been demonstrated to be particularly useful in suggesting reduced-complexity connectivity patterns for NNs [16]-[21] (reduced-complexity implies a smaller Performance Subset, one of our objectives). For convenience, the general methodology being drawn from in the present line of research is called the General System Method (GSM). It is an outgrowth of earlier work in constraint analysis by Ashby [1], and a number of systems researchers have contributed to its development and use (e.g., [6][9][11][12][4][5][3][2]).

As success in the proposed approach is accumulated, the method would inject a new step in the normally ad hoc process

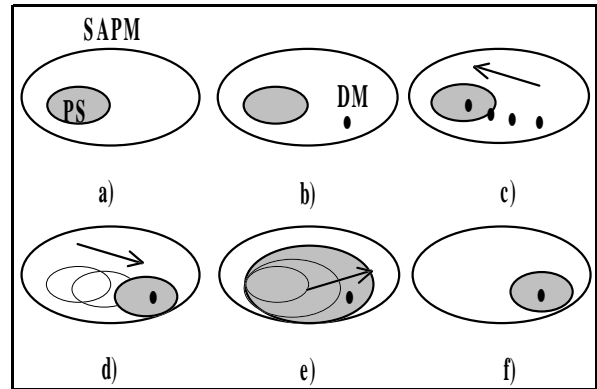


Figure 1.

of choosing the NN's structure and training algorithm for a NN application. This method is a constructive one, comprising 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 [12](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 *structure matching* (or, *constraint matching*) will have been accomplished. The procedures and experimental results for doing structure matching via *modularized NNs* in this context are reported in [17]-[21].

The focus of the present paper, however, addresses situations wherein a large number of measurements or features are given to the problem solver, and part of the task is to pare down the number of these measurements/features in some meaningful manner, in order to reduce the number of inputs to the NN. This aspect of the design task was not treated in the previous work. A method employed herein for accomplishing such *dimension reduction* is Extended Dependency Analysis [4][5], to be described in Section 5. A pattern classification context is used in Section 8 to demonstrate the ideas. First, however, a recapitulation of the previous work.

4. MODULARIZATION APPROACH

It turns out that the GSM *notation* 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 structure. 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 Fig. 2a. 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 Fig. 2b. 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.

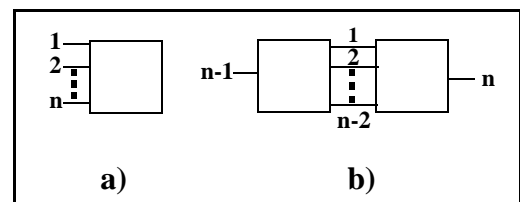


Figure 2.

In GSM, a lattice is used to organize the large number of combinations of non-decomposable sub-relations among n variables (e.g., see [12] 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, and the bottom of the lattice contains n boxes, each with 1 attached line (all n variables independent).

For the pre-structuring 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 internal 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.

4.1 Nominal Data.

The GSM methods considered herein work on 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 an exhaustive, top-down search is computationally feasible. Conant [5] has developed a bottom-up approach (described in Section 5) that incorporates heuristics, and demonstrated success for n on the order of 100 on a microcomputer, and with current technology, this number could possibly go up by a factor of 10.

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 quantitative data, and various multivariate tools offer significant computational power with which to extract structural information by exploiting the metric of the data. A key motivator for the present research topic is that the GSM method of Section 5 develops probability density information about the training data set as part of its process, and this information appears usable for the pre-structuring process. In the example of Section 8, it is demon-

stated that a direct reduced-complexity design can be accomplished with such information, one that meets the requirements stipulated in Section 1 that the NN's Performance Subset has the attributes of being "big enough but not too big", AND for which there is assurance it contains the desired mapping. In fact, we show an example wherein even the weight settings are obtained via this a priori information.

4.2 Experiments re. Modularization Approach.

For the modularization approach, an extensive set of experiments on 5-input and 7-input, 1-output structures were performed [21]. Since all of these structures have only one output, for simplicity, the output is here dropped from the notation. In GSM notation, a 3-input structure would thus be designated as an ABC structure [formally, an ABCD structure, where D is the output, which we are dropping here]. An example of this representation is shown in Fig. 3a. 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 for such a system (the SAPM defined in Section 2) has been widely studied, and much is known about them.

Assume two different structural types: i) non-decomposable, and ii) decomposable-into-two-relations with one shared variable. In GSM notation, the first is expressed as an ABC structure; the second as AB:BC, the latter notation making clear that B is the shared variable. Permutations of the input variables generate different, but topologically equivalent, mappings. In GSM, the ABC structure corresponds to a relation of maximum complexity – no reduction available (within this framework). The AB:BC structure represents the case where a (partial) decoupling of variables is possible, that is, ABC is decomposable into two sub-structures AB and BC which are not further decomposable.

If nothing were known *a priori* about a specific 3-input, 1-output binary mapping to be learned, then a candidate NN structure to put into the box of Fig. 3a could be a standard fully-connected MLP (feed-forward, with perhaps a single hidden layer, sigmoid activation function) [call this a 'non-decomposed structure']. However, if *a priori* knowledge were available that the mapping to be learned was of the AB:BC type, then an NN structure such as that shown in Fig. 3b could be used. In this case, we decompose the hidden layer into two sub-structures (two shaded boxes) which are not further decomposable. The number of inputs to each sub-structure is smaller than for the NN of Fig. 3a. We call this a decomposed structure, or alternatively, a modularized structure. For a 3-input/1-output system which has only 256 total possible mappings, this may seem trivial, but even moving up to just a 5-input system, the total number of possible maps is Order(billion)! Even for seemingly small numbers of inputs, it is physically not tractable to build NNs whose Performance Subset covers the entire set of possible mappings, so any constraints discovered in the data that can be translated into correlated constraints on the NN structure are most welcome.

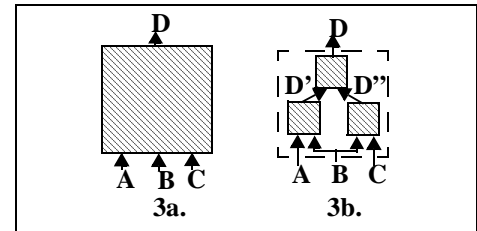


Figure 3.

For the experiments reported in [21], we defined a NN test structure corresponding to each level in the lattice of structures (for the number of input variables). The one for the top level in the lattice, i.e., non-decomposable (most complex), structure - is called NN **structure zero**. All test structure input modules were given $n+1$ hidden elements, where n is the number of inputs to the respective input module. Based on past work, we developed certain research expectations. First and foremost was that the test NN whose structure most closely resembles the underlying structure of the desired mapping would have the best generalization. In addition, a kind of 'neighborhood' relation was also conjectured, namely, that NN test structures 'close' to the optimal structure in the lattice would have better generalization than structures 'further away' in the lattice. Finally, we put forth a conjecture that using the same structure form, but with different input labelling would yield degraded generalization performance.

To test the generalization capabilities of pre-structured NNs, experiments were performed on a group of 5- and 7-input/1-output Boolean functions. Feedforward NNs with back-propagation-of-error training were used. The key variables in the experiments were *mappings with different underlying structures* to be learned, and the *structure of the NN* being trained. Details of the procedures are given in [21] (we note here, however, that over 5,000 experiments were run).

When selecting data for training any NN, it is important to choose a "representative" set of data. In some cases this is based on intuitive/heuristic procedures, and in other cases it is based on statistical considerations. As is well known, only to the extent that the training data actually captures the structure of the application domain, and the NN succeeds in capturing the structure in the data (i.e., a structure matching), can the NN be expected to generalize well. With the Boolean functions used in our experiments, it was possible to logically deduce representative training data by examining the precise structure of the mapping selected for training the NN. The more structured the selected mappings, the fewer the I/O pairs needed to completely specify the structural information. Such hand picked sets of I/O pairs served as "minimal training sets." Experiments were performed using these special training files to demonstrate that an NN whose structure matches the underlying structure of the mapping can successfully (i.e., with good generalization) extract the entire mapping from just the minimal training set, whereas the fully-interconnected (structure-zero) NN typically cannot.

In real-world scenarios, one is not usually able to hand pick data in the manner of the previous paragraph. Accordingly, we also ran a series of experiments based on a random sampling of the total set of possible I/O pairs. See [21] for details.

Test NNs were created for the 5- and 7-input cases, representing a lattice of structures with decreasing complexity. The

number of modules within the NN test structure were kept to a maximum of two. As a complexity measure, we focused on the number of shared variables between the two modules. The greater the number of shared variables, the greater the complexity of the structure. Specifically, for the 5-input case, the structures went from maximum complexity (ABCDE), to three shared variables (ABCD:BCDE), two shared variables (ABCD:CDE), and one shared variable (ABC:CDE)--the latter is shown in Fig. 4-- and no shared variables (ABC:DE). This lattice can also be viewed as a “family” of structures where the structures of lower complexity can be considered the descendants (or children) of more complex structures. With this view, one would expect any “parent” structure to be able to learn any mapping that its children can learn, while a child only has a *possibility* of learning a mapping of the same complexity as one of its parents. We found this to be true, however, it is in the area of generalization that the properly structured child *has an advantage* over its parents.

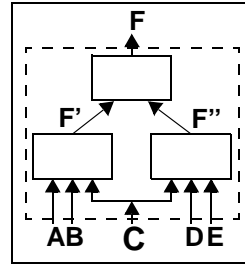


Figure 4.

The experiments consisted of training all the NN test structures on selected mappings whose underlying structure is known, and then testing each NN's generalization abilities on unseen I/O pairs for the given mappings.

4.3 Results of Modularization Experiments.

We summarize results given in [21], for the 7-input experiments. In experiments with the (hand-picked) minimal training sets, the NN whose structure matched that of the desired mapping was *always* able to outperform the other test structures. The winning NN test structure succeeded in extracting the needed information from the minimal training set to achieve 100% generalization performance. In all but one case, the structure-zero (non-decomposed) NN had the poorest generalization performance--as predicted based on the proposal here of matching NN structure to the structure underlying the I/O data.

In a stunning example of an ‘optimally’ prestructured [in the sense of having a small (smallest?) PS, with assurance it contains the DM -- cf. Section 2] NN's ability to learn the remaining information needed to perform the entire mapping, the 7-input test structure corresponding to a single shared variable (the least complex, most structured, mapping used in these experiments), was able to achieve **100%** generalization after being exposed to a hand-picked set comprising **only 18%** of the desired mapping's I/O pairs during training! Given these same I/O pairs, the structure-zero (non-decomposed) NN successfully generalized only 65% (recall that 50% is random). To get the structure-zero NN to improve its generalization to the 90% range required an exposure to 85% of the I/O pairs (randomly selected). We had previously reported a result for a 5-input NN [19]; wherein a level-1 test structure achieved 100% generalization on exposure to a 50% hand-picked train set, with the corresponding structure-zero NN performing 55% generalization. Keeping in mind that the likely candidate NN structure to use in the absence of *a priori* information is the fully connected (structure-zero) NN, these results further press home the idea of obtaining as much *a priori* information as possible for use in selecting the NN structure before training. Similar positive results are demonstrated for the other research predictions suggested earlier.

While the reported investigations ([17]-[21]) were based upon Boolean functions [because they have well documented properties that are utilized in the experimental design and interpretation], the long term objective is to develop a base for extending this approach to other classes of functions, specifically, to continuous ones.

5. EXTENDED DEPENDENCY ANALYSIS (EDA) METHODOLOGY

The key new results of this paper are based on application of the Extended Dependency Analysis (EDA) portion of the GSM to our structure-matching objective. Referring back to the lattice of structures described at the beginning of Section 4, as was mentioned, one procedure for determining the desired structure for a given data set starts at the top of the lattice and works down, until the desired structure is discovered (we call this top-down approach Reconstructability Analysis, however, some authors use this term more broadly to include bottom up approaches as well). This method suffers the combinatorial explosion syndrome, being doubly exponential in the number of variables. Extended Dependency Analysis (EDA) is a method developed to avoid this combinatorial explosion, through the use of a few simple heuristics that are only polynomial in the number of system variables. This approach starts from the bottom of the lattice and works up, and turns out being a very general technique for finding structural relations amongst variables, and is applicable to both dynamic analysis of time series and static analysis of panel data. This approach to structural modeling was developed to make it computationally feasible to model larger systems than is practical with traditional reconstructability analysis or dependency analysis, which are typically limited to systems under 10 variables. The basic EDA considers a single target (dependent variable) and searches for a set of other variables in the system that together significantly explain the target. This is essentially the analysis of a directed system, and is all that is required for a dynamic (time series) analysis. For static systems (panel data), basic EDA is carried out using each system variable as the target. The collection of candidate variable sets is then analyzed using a graph theoretic representation, producing a model for the entire system.

An attribute of the EDA that turns out being especially useful in our NN design context is that in the process of determining the structural information (which is based on “binning” the continuous data to develop categorical data upon which the EDA operates), it also provides substantial information about the probability densities represented by the data. The latter may be incorporated in the NN design; a key benefit of such transition to the NN with this information is that the NN, in contrast to EDA, operates with the full metric (as opposed to categorical) problem domain.

5.1 Dynamic Analysis

The dynamic algorithm consists of a pair of heuristics labeled H2 and H3, together with some minimal form of reconstructability analysis. For each dependent variable in the analysis, H2 calculates the three-way transmission between the dependent variable and every possible pair of independent variables in the analysis. These values provide a measure of how much the knowledge of independent variables reduces uncertainty about the dependent variable. The pairs of independent variables are sorted based on these transmission values, and those which pass a chi-squared significance test are put into the set of candidate variables. The maximum size of the candidate set is a user-determined parameter in this algorithm. When more than this number of significant variables are found, the most significant variables are saved and the others are discarded.

After the exhaustive search of three-way transmissions, the candidate set is evaluated using whatever form of reconstructability analysis is practical. The intent at this stage is to weed out any variables that contribute nothing to reducing the uncertainty of the target given the high-order interaction of all the other variables in the candidate set.

The second heuristic, H3, takes the candidate set of variables from H2 and attempts to expand it by checking for higher and higher order relations involving the dependent variable, all the current candidate independent variables, and every individual non-candidate independent variable. This search allows variables into the explanatory set which are found to take part in higher order relationships to a significant degree (as determined by a chi-squared test) which were not found to take part in significant relationships in H2.

After H3 has added variables, reconstructability analysis is again applied to the candidate set to eliminate any surplus variables. The sequence of H3 and reconstructability analysis can be iterated several times until all possible additions to the candidate set have been tried.

Note that the weakness of this pair of heuristics is that for a significant high order interaction to be found, some triadic sub-interaction must be statistically significant. While it is certainly possible that such interactions could go undetected using this approach, the assumption is that in practice such instances are rare.

5.2 Static Analysis

Static analysis begins after a dynamic analysis is performed with each system variable as the target, producing a candidate set of variables for each system variable. This collection of candidate sets is then represented as a graph. Each node of the graph corresponds to a system variable, and two nodes are connected if they appear together in any candidate set. The structural model components are then determined by finding all the cliques in this graph.

Because of the limitations of heuristics H2 and H3, not all significant connections between nodes may be present in this graph. Two additional heuristics can be used to fill in these near cliques and candidate cliques. Heuristic H4 searches the graph for near cliques -- sets of nodes that are 80% to 90% interconnected. When such sets are found, the additional connections are added to the graph, and the resulting models evaluated using reconstructability analysis. Additional candidate cliques can be found by heuristic H5, which searches for cliques that have some minimum amount of overlap in their memberships. The union of two such overlapping cliques is then evaluated using reconstructability analysis. While these algorithms appear computationally expensive, the steps necessary for the 195-variable example in Section 8 required only 45 seconds on a 166MHz PC.

6. COMMENTS ABOUT PATTERN CLASSIFICATION

Generally speaking, the task of pattern classification/recognition is one of decision making. Problem contexts for this task are myriad, but typically fall into those of waveform classification and classification of geometric figures. Waveform classification includes what a doctor does when reading an EKG and pronounces "good" or "bad"; similarly, classification of geometric figures is what a baby does when viewing a room full of faces and points to one, saying "daddy". A precursor to the decision-making step is a set of measurement (sensor) data.

The mathematical construct of a representation (vector) space is created to house the measurement/feature data, and the collection of exemplars for each class forms a corresponding distribution in this space. Then, a collateral characterization of the classification task is one of estimating density functions in a high-dimensional space and dividing the space into regions, applying one of the class labels to each. For a given instance, a measurement vector is acquired, and a decision is made about which class the measurements are from. The boundaries separating each of the classes are called discriminant functions; a device that implements such discriminant functions is a classifier or recognizer, or in the case of NN implementation, a recognition network.

The probability of error is usually considered the key parameter in pattern recognition. The theoretically best classifier in the sense of minimizing probability of classification error (based on given distributions) is the Bayes Classifier. Due to difficulties in implementing the Bayes Classifier, approximations are employed. The simplest are parametric classifiers based on assumed mathematical forms ('structures' in our context) for either the density or discriminant functions. Even when mathematical forms can be assumed, the values of the parameters (to refine the fit of the model) must be estimated from available data. Issues concerning how the amount of data impacts classifier design/performance enters at this point. When no parametric structure can be assumed for the density functions, nonparametric methods must be employed, e.g., Parzen and k-nearest neighbor approaches, for estimating density functions. In general, the nonparametric approaches are very sensitive to choice of control parameter values, and are subject to giving heavily biased results.

One simple method for decision making in a high dimensional space, is to consider decision making as a dictionary search. All past experiences (learning samples) are stored in a memory (dictionary), and a test sample is classified to the class of the closest sample in the dictionary. This process is called the nearest neighbor classification rule. This classifier divides the space into regions in an often complex, sample-dependent way. This is a nonparametric classifier.

The neural-network approach is often characterized as being a model-free approach (non-parametric, in the statistician's vocabulary). However, we *do* start with a NN structure, and adjust weights (parameters). So while it may be true that the underlying model in the NN case may be substantially more general than those typically used in the statistical context, it is not technically correct to call NN approaches model free.

When the number of different measurement types (and thus the dimensionality of the measurement/representation space) needed to characterize a problem domain is large, the difficulty of designing a classifier increases enormously. This provides incentive to do some preprocessing of the original measurement data to extract important "features" from the observed samples. Such preprocessing can be thought of as performing a mapping from the n-dimensional measurement space to some lower-dimensional feature space. The mappings are not limited to any specific form and the class separability is used as the criterion to be optimized -- i.e., care is taken to not seriously reduce class separability. It is to be noted, that as long as the features are computed from the measurements, the features cannot carry more classification information than the measurements. Accordingly, the Bayes error in the feature space is never smaller than that in the measurement space. One method of dimension reduction is to decompose a given distribution into a number of clusters; the process of coming up with these clusters is called *clustering*, and in the NN context, *unsupervised learning*.

All of the above considerations have been visited for the example classification task in Section 8. The context there is assigning a (predominant) land-use classification to a 2-Km. diameter patch of land, based on aerial imagery data. The first preprocessing step was to perform a 2-dimensional Fourier Transform on the photographic image. The motivation for this transformation is the possibility of effecting translation, scale and orientation invariance for an object in the scanned image. The Fourier transform data is reduced to 195 numbers by using specialized "sensors" in this domain. One approach is to work in this 195-dimensional measurement space, and proceed accordingly. Historically [15], another approach was to treat each 195-dimensional vector as patterns in themselves and develop characterizing features for them. This process yielded 54 features (known to be correlated), hence relocating the classifier design from a 195-dimensional space to a 54-dimensional space.

In Section 8, Extended Dependency Analysis (EDA) is used to significantly reduce the dimensionality needed for successful land-use pattern classification, for both, the 195 measurements on the Fourier Transformed scanned image, and the 54 features extracted from these measurements.

7. COUNTER-PROPAGATION NN PARADIGM

This hybrid neural network paradigm is explained here in preparation for the Pattern Classification example in Section 8. The originally proposed counterpropagation network [6] architecture involved a two-way flow (counter flow) of information intended to create a two directional lookup table. A simplified unidirectional network was introduced at the same time, and expanded upon thereafter[7], and is the one used herein. The simplified network is composed of two layers. The first layer comprises competitive "instars" trained using Kohonen learning. "Outstars" are emitted from the first layer to form the second layer, and the outstars are trained using Grossberg's outstar learning rule, after the first layer is finished being trained.

The training process consists of the Kohonen phase in which the weights of the instar layer are organized into prototype vectors through a clustering process. Each instar element then corresponds to the center of a cluster of input vectors. The training process is competitive, moving the closest existing prototype towards the presented input. All the standard difficulties associated with Kohonen learning can be encountered. The goal is that the prototypes arrange themselves in such a way that they are equiprobable in the nearest neighbor sense to the input vectors in the training set.

The second phase of training adjusts the outstar weight vectors associated with each prototype towards the desired output associated with the cluster center. Since the prototype layer is competitive, only one element at a time will have a non-zero value. This tends to make this training process relatively quick.

Trained Counterpropagation networks can be used in two different manners. In accretive mode, the network acts as a simple lookup table, wherein the value associated with the prototype closest to the input is returned. This mode is identical to the operation of the network during the second phase of training (though without a desired output specified). The second mode involves allowing two or more prototypes to "win", i.e. be near the input. In this case the outputs of each element are scaled so that they sum to one. This causes the network output to be the linear interpolation of the lookup table values for each of the winning prototypes. This second mode is appropriate when the desired output function is locally linearizable.

8. PATTERN CLASSIFICATION EXAMPLE

8.1 Description.

The example we use here to illustrate pre-structuring in a pattern classification context comes from one of the authors' archives contains a set of data from a NASA project of 25 years ago. The work was done in the Earth Observations Division at the Johnson Space Center in 1974 on Skylab II photographic imagery. A selected photographic plate of Phoenix, Arizona, had

been re-photographed with a circular aperture corresponding to a 2-Km. diameter circle on the ground, spaced so that the area originally photographed was covered by 433 of these circular “events”.

Each “event” photograph was then processed through equipment which produced an optical Fraunhofer Diffraction Pattern (known to be equivalent to the 2-dimensional Fourier Transform) of the event image. Two distinct sampling geometries were used to record the light intensity patterns of these FDP's: a radial wedge sampling and an annular ring sampling (this was based on the Diffraction Pattern Sampling methodology for pattern recognition developed by one of the authors in the late 1960's [23]). The wedge measurements provide direction information, and the annular-ring measurements provide spatial-frequency information. The geometries are centered on the optical axis of the system, which corresponds to a spatial frequency of zero. The Fourier Transform inherently provides translation invariance for items in the image, and the selected sampling geometries provide capability for size and rotation invariance. The objective of the study was to explore the ability of the Diffraction Pattern Sampling methodology to aid in land-use classification using aerial imagery. Measurements were taken through $100 \times 1.8^\circ$ wedge-shaped sensors, and 95 annular-ring sensors, resulting in 195 data points for each of the 433 events; these were the basis of further analysis.

During the 1974 project, 57 “features” were empirically extracted from the FDP data points[15]. By virtue of their ad hoc development, it was understood there would be substantial correlation among these features. A variety of methods were used to identify classification groupings for the original photographic events. These were boiled down from 96 to 5 land-use classes, and Fischer discriminant tests were developed and refined to yield 90+% classification accuracy, using as few as 9 of the features as input.

The data set in the authors' possession is incomplete, and NASA has long since discarded the original data. Nevertheless, data comprising the 54-features for 290 of the events were “resurrected” to provide coverage of the 5 land-use classes. The 5 classes are: Urban, Residential, Farm, Mountain, and Water; the frequency count of available examples: Urban, 126; Farm, 74; Water, 31; Residential, 30; and Mountain, 29. The corresponding original Wedge and Annular Ring data (195 measurements for each event) was only found for 177 of the photographic events. For explanation of the Diffraction Pattern Sampling methodology, see [23].

It was decided to use this Phoenix land-use data set as a basis for demonstrating the NN pre-structuring (structure matching) ideas of this paper. In particular, Extended Dependency Analysis is applied to this data to effect dimension-reduction (for both, the 54-feature data set and the 195-wedge-ring measurement data set), and to provide information useful in the NN pre-structuring, including an example of full NN design specification.

8.2 Extended Dependency Analysis (EDA) of the Pattern Classification Example.

The first step in applying EDA to a problem with quantitative data is to establish a scheme for converting the data to nominal/categorical values (a binning scheme). Such a scheme must include the number of bins to use for each variable, and a procedure for assigning observations to bins. While a host of possibilities exist, our preference is that an initial binning scheme should be both as unbiased and as computationally simple as possible. For the pattern classification task described in the previous section, we chose to express each variable using five bins and to assign observations to bins based on an equal observed interval scheme. For each variable i , we find the smallest and largest observed value in our training sample, a_i and b_i , respectively. The total observation interval for variable i is then $l_i = b_i - a_i$. We break this interval up into 5 equal subintervals of length $l_i/5$, denoted by S_{i1} through S_{i5} . These subintervals then constitute the bins we use in the categorical analysis. While not sophisticated, this scheme is computationally cheap to implement and in our example, turns out being rather effective.

Having converted the quantitative measurements and features into categorical values using the above method, we divided the available 54-feature data set for 290 photographic events into two data sets, one comprising 147 events for training, and one comprising 143 events for testing generalization. For the available 195-measurement data set for 177 events, we again created two data sets, one comprising 100 events for training, and one comprising 77 for testing generalization. For either set of data, it is clear that there are not enough training samples to carry out a statistically significant analysis. Fortunately, information theoretic analysis techniques can proceed based on measures of reduction in uncertainty, even in the absence of statistical significance. While not a substitute for properly applied statistical methods, uncertainty-based techniques allow one to proceed in a *principled manner* in cases where one just doesn't have enough data. In such cases, however, the latter stages of the EDA methodology, which rely on significance tests, are not used, and thus limit analysis to the application of the H2 heuristic.

The H2 analysis amounts to sequentially constructing approximate joint probability distributions for two independent variables (features or measurements) and the dependent variable (classifier output) based on the training data. These distributions can be decomposed into conditional distributions for each class (a step we will take in the next section), and allow us to calculate the degree to which “class” is explained by each possible pair of features or measurements.

The H2 heuristic yields an ordering of the feature or measurement variables by their ability to reduce uncertainty in knowing the class of a sample, based on their pair-wise interaction with each of the other features or measurements. The results for the 54-feature analysis are in Table 1, and the results for the 195-measurement analysis are in Table 2. In addition to the order-

ing, EDA gives us transmission values for each pair. These values are the reduction in the uncertainty of the sample's class based on knowing the sample's value for that pair of features or measurements. Finally, EDA also gives us the uncertainty reduction from knowing the sample's values for all the variables in the dependency set. The size of the dependency set is chosen by the user, and a choice of size n selects the top n variables of the ordered list. In our analysis we found that a dependency set of six variables was adequate to completely reduce the uncertainty (to a value of zero) in the 54-feature case. For the 195-measurement case, this is accomplished with 8 variables.

Table 1: Results of EDA on 54 Feature Variables

Uncertainty of the Dependent Variable (Class):2.0092 bits. 'Transmission' is amount reduced from this value.			
Independent Variable	Best Partner	Transmission (bits)	Reduces Uncertainty by:
Feature 23	Feature 48	1.8868	93.91%
Feature 48	Feature 23	1.8868	93.91%
Feature 27	Feature 48	1.8840	93.77%
Feature 29	Feature 48	1.8813	93.63%
Feature 32	Feature 48	1.8542	92.28%
Feature 44	Feature 48	1.8510	92.12%
Feature 52	Feature 23	1.8038	89.77%
Feature 16	Feature 48	1.7661	87.90%
Feature 38	Feature 48	1.7191	85.56%
Feature 50	Feature 23	1.6952	84.37%

Table 2: Results of EDA on 195 Measurement Variables

Uncertainty of the Dependent Variable (Class): 2.3197 bits.'Transmission' is amount reduced from this value.			
Independent Variable	Best Partner	Transmission (bits)	Reduces Uncertainty by:
Wedge 37	Ring 48	2.1708	93.58%
Ring 48	Wedge 37	2.1708	93.58%
Ring 49	Wedge 83	2.1699	93.54%
Wedge 83	Ring 49	2.1699	93.54%
Ring 52	Wedge 37	2.1679	93.45%
Wedge 70	Ring 49	2.1663	93.39%
Ring 47	Wedge 83	2.1636	93.27%
Wedge 84	Ring 52	2.1600	93.11%
Ring 53	Wedge 37	2.1534	92.83%
Wedge 69	Ring 48	2.1518	92.76%

It is worth noting that our ability to reduce the uncertainty of a sample's class to zero in both analyses is context specific – we have both information-rich feature and measurement data, and small training sample sizes. While one would usually wish to reduce the uncertainty to zero, in many cases this will not be possible using only a small number of variables.

It is interesting (indeed, intuitively pleasing) that the pairs selected by the EDA for the Wedge-Ring measurement data always turn out containing one wedge and one ring. As we might expect, the information carried by the two types of observations is complimentary. It is also interesting that the rings selected are closely grouped in one frequency band.

To demonstrate the structural analysis component described in Section 4 as applied to this pattern classification task, we carried out an abbreviated Reconstructability Analysis based on the results of the EDA for the 54-feature data set. We fed in the binned ABCDEZ contingency table as input to reconstructability analysis [28][31] (the A-E letters correspond to the features 23, 48, 27, 29, & 32; and Z corresponds to the dependent variable -- class). First we searched over loopless models, looking for the best single predictor of Z, the best pair predictors, and the best triple predictors. The best single feature was 48, the 2nd best was 23. The best pair of features was 23/48, and the 2nd best pair was 27/48. The best triplet of features was 23/48/42; the 2nd best triplet was 48/29/32.

We then ran a search over all models in the lattice (this is possible for 5 variables with the features binned into 3 values each), including those with loops, and looked for models with high $(I+S)/2$, where I = information content, normalized from 0 to 1, and S = simplicity = 1 - complexity, where complexity is degrees of freedom, normalized to a 0 to 1 range. Models BDZ:EZ and BZ:DZ:EZ had the highest $(I+S)/2$, namely .957 and .958, respectively; these reduced $u(Z)$ by 86.7% and 85.3%, respectively. This suggested a choice of B, D, and E as the optimum predictors of Z. Going from our original ABCDEZ model to a

BDEZ model accomplishes a huge reduction in our degrees of freedom from 1214 to 134, while only modestly diminishing the 90.7% uncertainty reduction to 86.8%. The full scale search allowed us to simplify our BDEZ model still further, to BDZ:EZ, with $df = 70$ and 86.7% uncertainty reduction, or to BZ:DZ:EZ, with $df = 66$ and 85.3% uncertainty reduction. Of these two models, BDZ:EZ is preferable. Note that in this model, BD predicts Z and E also predicts Z, but these predictions are separate (the two predictions are merged using the maximum uncertainty principle); this is different from all three independent variables jointly predicting Z as they do in model BDEZ (Zwick, 1996). Compared to the original ABCDEZ model which has $df = 1214$, our final model, BDZ:EZ has $df = 70$ with only 4% loss of uncertainty reduction.

8.3 Pre-structuring NNs based on EDA results for the Pattern Classification Example.

The information theoretic structural analysis in the preceding section was done using categorical data. In general, potentially useful variation in the data is likely discarded in the transformation from quantitative to nominal/categorical variables. However, making use of the information obtained via the structural analysis to effect a pre-structuring of a neural network, puts us in the position of embedding the nominal-data results back into a quantitative context with a natural ordering and metric. In principle, there should be more explanatory power in the quantitative domain than in the nominal domain. Thus this process of obtaining certain structural information via a categorical analysis, pre-structuring the NN based on this information, and then letting the NN operate in its natural quantitative way, allows us the potential of regaining some "information" that we might have left behind when we binned the data.

The analysis in the previous section shows that all the information needed to correctly classify all the samples in the training data sets is contained in the derived dependency sets, providing a remarkable degree of dimension reduction in both the feature and measurements spaces: from 54 down to 6 in feature space, and from 195 down to 8 in measurement space.

While this dimensionality reduction is very attractive, we should note that just as we left information behind when we transformed our quantitative variables into nominal values, we leave potentially valuable information behind in the nominal domain if we use the results of EDA purely for dimension reduction. The conditional probability distributions that are developed in the EDA can additionally play the role of an approximate Bayes classifier for the nominal variables. Since our goal in constructing a neural classifier is to approximate the Bayes classifier for the quantitative variables, a natural suggestion is to take advantage of this approximate Bayes classifier in the nominal domain, at least as a starting point for the classifier design in the quantitative domain.

A wide variety of NN architectures can be used for implementing a pattern classifier. Our previous work with pre-structuring Multi-Layer Perceptrons (MLPs) for function approximation [17]-[21] suggests an equivalent prestructuring approach for this pattern classification problem. Use of MLPs in this problem context suggests a trade-off between parsimony of structure and complexity/increased training time. Various other NN paradigms are also appropriate candidates into which we might apply the structural information obtained; these could include Probabilistic Neural Nets (PNNs), Adaptive Resonance (ART) based networks, and many others. The Counterpropagation NN paradigm described in Section 7 turns out being a particularly simple one for implementing the conditional probability distributions approximated in EDA. Experiments with the MLP and the Counterpropagation paradigms are described in the next three subsections.

8.3.1 Results with MLP Paradigm.

Extended work with the 54-feature data set using MLPs has found that a single-hidden-layer network with 54 inputs, 8 hidden-layer processing elements with sigmoid transfer functions, and 5 output elements is capable of perfectly learning the training samples and produces 92% - 95% correct classification on the generalization events. In the current exploration, we reduced the number of inputs to the NN based on the EDA to just those 6 features in the dependency set, and ran a number of experiments. In addition, the information obtained from the Reconstructability Analysis using the top 5 of the 6 variables of this same dependency set suggested further experiments with modular NNs (cf. Section 4). In all cases we found that as long as we followed the clues provided by the information theoretic analyses, we could greatly reduce the number of free parameters in our networks while maintaining classification rates in the high 90% range for training samples, and low 90% for generalization samples. The BDZ:EZ structure cited in Section 8.2 suggested a NN structure with two modules, one with inputs from two features (designated BD in this format), the other with a single input (designated E). Four elements were employed in each of the modules; the corresponding non-modularized NN employed 8 elements in the hidden layer, having a full interconnect with the input variables.

8.3.2 Results with Counterpropagation Paradigm.

A straight forward implementation of a Counterpropagation NN for our problem context could proceed as follows: Provide as many elements in the prototype layer (first layer) as there are exemplars in the training set, and set the instar weights for each prototype element equal to one of the input vectors. This approach would (almost) guarantee 100% classification for the training samples (assuming enough memory is available for all the prototypes). In our case, networks instantiated in this manner did learn the training samples perfectly, and generalized at a 92% rate for the 54-feature data set and at a 95% rate for the 195-measurement data set. Based on the EDA, we can make two initial simplifications to this implementation. First, reduce the number of inputs as above for the MLPs. Second, select the *number* of elements to allow in the prototype layer based on the number of non-zero cells in the frequency table calculated for the dependency set during EDA. Since the number

of non-zero cells in the frequency table is bounded by the number of samples in the training data set, and is probably significantly less, we have a reasoned basis for selecting a smaller number of prototype elements than training samples (a component of our desired pre-structuring process).

In the 54-feature case, we found that our six-way frequency table for the dependency set had 74 non-zero cells as opposed to 147 samples. This significant reduction in prototype elements still allowed us to train perfectly and produced a generalization rate of 93%. The 195-measurement case had 83 non-zero samples in the eight-way frequency table of its dependency set as opposed to 100 training samples. The 8-input 83-prototype networks we trained typically achieved a 99% correct rate on the training samples and a 91% generalization rate.

8.3.3 Results With Fully Pre-structured NNs

Now we would like to directly translate our joint probability distribution for the nominal variables into a neural classifier design for the quantitative features and measurements. In the following discussion, we use the subscript k to designate a particular observation, subscript i to designate a particular variable, and subscript j to refer to a particular nominal value. For the k -th sample we have a feature or observation vector $\mathbf{X}_k = (X_{k1}, X_{k2}, \dots, X_{kD})$, where D is the number of variables in the dependency set. In the EDA, each X_{ki} takes on one of n_i nominal values v_{ij} , based on the binning scheme used. The binning scheme we introduced above can be expressed as

$$v_{ki} = v_{ij} \text{ if } X_{ki} \in [a_i + (j-1)l_i, a_i + jl_i).$$

In the quantitative feature or measurement space, we can refer to the intervals that generate the nominal values by specifying the center of the interval c_{ij} . For our simple binning we have

$$c_{ij} = a_i + (j-0.5)l_i.$$

We can partition our D dimensional quantitative space by taking the direct sum of the intervals defined on the individual variables. Each region in the partition can be referred to by specifying the region's center $\mathbf{C}_j = (c_{1j}, c_{2j}, \dots, c_{Dj})$. For our binning scheme, the set of points \mathbf{C}_j gives a Voronoi partition for the space.

Since we know the most likely class of any sample falling into a particular region based on the EDA, we can implement a classifier in the quantitative space using a nearest neighbor approach on the \mathbf{C}_j . This is equivalent to specifying a set of conditional probability distributions that are uniform on each region defined by the binning scheme, then taking the most likely class. While assuming a uniform probability distribution across each region is a coarse approach to take, it is indicative of the kind of information we lose when we transform from quantitative to nominal variables. What is most important to realize is that we can retrieve the coarse probability distribution from the nominal space and use it as a starting point for estimating a more accurate distribution in the quantitative space. Using this kind of information allows us to explore pre-structuring a NN in new ways, in addition to that of the modularization method described in Section 4.

As an example, we can take the vectors \mathbf{C}_j that partition our quantitative space and use them directly as instars to prototype elements in a Counterprop NN. Thus, the prototypes are completely specified by the binning scheme and the actual range of variable values observed. The outstar weights from each prototype element may be directly assigned, using the conditional probabilities of the classes in that region developed in the EDA. If there are no training examples in a given partition, the corresponding prototype element in the Counterprop NN may be dropped. This pre-structuring procedure results in a fully functional NN!

When implemented this for the land-use classification example. With the 54-feature data set, there results a 6-input, 74-prototype-element network that is 100% correct on the training samples and generalizes at 93%. For the 195-Wedge-Ring-measurement data set, there results a 8-input, 83-prototype-element network that is 100% correct on the training samples and generalizes at a 91% rate.

9. CONCLUSION

Our objective has been to continue development of methods for pre-structuring NNs based on structural information extracted from analysis of available problem-domain data using GSM (information- and set-theoretic based) methods. Previous work focused on using the GSM tools to develop a modularized NN kind of prestructuring in a function-approximation context. The current work focused on issues in the pattern recognition context. These include reducing the number of inputs to the NN (dimension reduction) and then, based on the binning process used (required by GSM), pre-selecting the number of prototype elements in a Counterpropagation style NN. The previous GSM tools used are computationally intractable for this context, and so, we selected Extended Dependency Analysis as the GSM technique for the current work. This pre-structuring yielded an NN that trained 99%-100% and generalized 91%-93%. These results are encouraging. Finally, the probability density information extracted from the data during the EDA process, was used to directly set all the weights in the Counterprop NN, resulting in a fully functional NN with 100% correct training, and also 91%-93% generalization. The usual reasons for implementing the classifier using neural networks apply, even in this latter context where the full design is obtained via information gathered in the EDA methodology. In addition, however, embedding the EDA derived information into a NN provides a means for moving from a categorical space (into which one is obliged to operate for applying EDA) back into a continuous, metrical space.

10. REFERENCES

- [1] Ashby, R., "Constraint Analysis of Many-Dimensional Relations," in *Prog. in Bio-Cybernetics II*, Wiener & Schade, eds. '65.
- [2] Broekstra, G. "C-Analysis of C-Structures," *Int Journal of General Systems*, v17, pp33-61, '81.
- [3] Cavallo, R. and G. Klir, "Reconstructability Analysis," *Int Journal of General Systems*, v7, pp7-32, '81.
- [4] Conant, R., "Structural Modeling Using a Simple Info Measure." *Int Journal of General Systems*, v11, no 6, pp721-730, '80.
- [5] Conant, R., "Extended Dependency Analysis of Large Systems," *Int Journal of General Systems*, v14, pp97-123, '88.
- [6] Hecht-Nielsen, R., "Counterpropagation networks", *Applied Optics*, **26**, 4979-4984, 1987.
- [7] Hecht-Nielsen, R., "Applications of counterpropagation networks", *Neural Networks*, **1**, 131-139, 1988.
- [8] Hertz, J., Krogh, A. & R. G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, 1991.
- [9] Klir, G., "Identification of Generative Structures in Empirical Data," *Int Journal of General Systems*, v3, pp89-104, '76.
- [10] Klir, G., *Architecture of Systems Problem Solving*, Plenum Press, 1985.
- [11] 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.
- [12] Krippendorff, K., *Information Theory: Structural Models for Qualitative Data*, Sage University Paper No. 62, SAGE, '86.
- [13] Lee, T-C, *Structure Level Adaptation for Artificial Neural Networks*, Bluer Academic, 1991.
- [14] Lendaris, G.G., "Structural Modeling, A Tutorial Guide," *TRANSACTIONS IEEE Sys, Man, and Cybernetics*, vol SMC-10, no12, pp807-840, Dec, 1980.
- [15] Lendaris, G.G. & S.B. Chism, *Land-Use Classification of Skylab S-190B Photography Using Optical Fourier Transform Data*, NASA LBJ Space Center, no LEC-5633, March, 1975.
- [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., M. Zwick & K. Mathia, "On Matching ANN Structure to Problem Domain Structure", *Proceedings of World Congress on Neural Networks'93 (WCNN-93, Portland)*, Earlbaum/INNS, July 1993.
- [18] Lendaris, G. & K Mathia, "Using A Priori Knowledge to Prestructure ANNs", *Australian Journal of Intelligent Information Systems*, vol 1, no 1, March 1994.
- [19] Lendaris, G. & K. Mathia, "On Prestructuring ANNs Using A Priori Knowledge", *Proceedings of World Congress on Neural Networks'94 (WCNN-94, San Diego)*, Earlbaum/INNS, June 1994.
- [20] Lendaris, G., "Prestructuring ANNs via A Priori Knowledge", *Proc. SPIE International Conference*, SPIE Press, Apr '95.
- [21] Lendaris, G., A. Rest & T. Misley, "Improving ANN Generalization Using A Priori Knowledge to Pre-Structure ANNs", *Proceedings of International Conference on Neural Networks'97 (ICNN'97)*, IEEE Press, July 1997.
- [22] Lendaris, G.G. & G.L. Stanley, "Structure and Constraint in Discrete Adaptive Networks," *Proc. of the National Electronics Conf*, vol XXI, pp. 500-505, 1965.
- [23] Lendaris, G. and G.L. Stanley, "Diffraction Pattern Sampling for Pattern Recognition," *Proceedings of IEEE*, v 58, No 2, pp. 198-216, February 1970. [A major paper].
- [24] Odin, S.V., Odri, S.V., Petrovacki, D.P. & G.A. Krstonosic, "Evolutional Development of a Multilevel Neural Network," *Neural Networks*, vol. 6, pp 583-595, Pergamon, 1993.
- [25] Reed, R., "Pruning Algorithms-A Survey," *IEEE Trans. on Neural Networks*, vol. 4, no 5, pp. 740-747, Sept 1993.
- [26] Warfield, J.N., *Societal Systems: Planning, Policy and Complexity*, Wiley, 1976.
- [27] Weigend, A.S., Rumelhart, D.E. & B.A. Huberman, "Backpropagation, weight-elimination and time series prediction," *Proc. 1990 Connectionist Models Summer School*, D. Touretzky, J. Elman, T. Sejnowski & G. Hinton, Eds., 1990
- [28] Zwick, M., H. Shu & R. Koch, "Information-Theoretic Mask Analysis of Rainfall Time-Series Data", *Adv in Sys Sci & Apps*, vol 1, pp. 154-159, 1995.
- [29] Zwick, M. "Control Uniqueness in Reconstructability Analysis", *International Journal of General Systems*, vol 24, pp. 151-162, 1996.
- [30] Zwick, M. "Complexity Reduction in State-Based Modeling," *Proc International Conf on Complex Systems*, Nashua, NH, Oct 1998.
- [31] Zwick, M. & M. Daniels, "OCCAM: A Reconstructability Analysis Program", in preparation, 1999.