Information-Based Algorithmic Design

Robert E. Hiromoto
Department of Computer Science
University of Idaho
Moscow, Idaho 83844-1010
USA
hiromoto@cs.uidaho.edu

Milos Manic
Department of Computer Science
University of Idaho
1776 Science Center Drive
Idaho Falls, Idaho 83402
USA
misko@uidaho.edu

***Abstract*— An information-based design principle is presented that provides a framework for the design of both parallel and sequential algorithms. In this presentation, the notion of information (data) organization and canonical separation are examined and used in the design of an iterative line method for pattern grouping. In addition this technique is compared to the Winner Take All (WTA) method and shown to have many advantages.**

I. Introduction

Information is the basic building block of all processes whether biological or physical in nature. The design process in many engineering and scientific fields rely in one form or another on the organization of information, and its application to a process under investigation. However, once a system is designed much of the information complexity seems lost to the understanding of the applications oriented users.

The organization and presentation of information represent a basic starting point for the understanding of process driven systems. From a physical and mathematical perspective, the casting of a system into its canonical form is an essential analysis process that provides insight and simplicity in unraveling the underlying process or processes.

Although not surprisingly, the notion of canonical forms appears not to be appreciated outside of the theoretical realms. The solution of application problems or the research in extending these solution methods are many times led by past experience rather than a deeper formulation that relies on the information complexity that the problem exhibits and; thus, seek a canonical reformulation based on the interactions of the information that defines the problem and solution domains.

In terms of information, the present work is inspired by Joseph Traub et al. [1] in his work on *Information-Based Complexity* (IBC). IBC provides a different perspective on the analysis of numerical algorithms. Although, there has been some disagreements [2; 3] to IBC's contribution from the point of view of some in the numerical analysis (NA) community, IBC introduces the notion of *information operators*, where information is partially derived and used by a computation (an algorithm A that defines the information-based solution method) to solve the problem. The solution rate is measured by the number of iterations I_n to convergence. Formally, if F is a set of problem elements f and G the solution domain then the solution operator S is defined by

$$S : F \ \rightarrow \ G \ \ \forall f \ \in F \tag{1}$$

Briefly, the partial information about f is gathered by computing the information operations $L(f)$, where $L \in \Lambda$ and Λ denotes a collection of information operations that maybe computed. If U is the approximation to the solution S then the sharp lower bound on the worst case error of U is within some radius of informations $r(N)$ that does not exceed some error ϵ, where N is the information operator and $N(f) \ = \ \{L_i(F) \mid i \ = \ 1, 2, ..., n \ and \ L_i \in \Lambda\}$ then U is guaranteed to be an $\epsilon - approximation$. This attention to information operations furnishes a comparison of algorithmic performance based on the information operator that is used. In this regard, the notion of non-adaptive information operators (parallel use of partially computed information) or adaptive information operators (sequential use of partially computed information) can be compared formally.

Within the context of this presentation, the introduction of the *information operator* and *information operations* represents a novel and attractive approach to algorithm analysis and design in general, and speaks to a broader application than as applied in IBC. From an algorithmic point of view, the flow and manipulation of information is the very essence of an algorithm's design.

The *IBC*, though steeped in the analysis of computationally relevant information, limits itself to only the analysis. In the following sections, we explore this question, and in so doing provide an example where the analysis of information flow or the use of information operators when placed in a form of a *canonically mapped information flow* may yields more optimal algorithmic designs when possible.

II. Canonical Information Flow

Traditionally in mathematics, a canonical form of a function is a function that is written in the most standard, conventional, and logical way. In its standard form, examples include the Jordan normal form of matrices, the canonical prime factorization of positive integers, the decomposition of a permutation into a product of disjoint cycles, and the alignment of system of equations along an orthogonal basis function.

Intimately connected with these canonical forms is the simplest description of the underlying systemic properties that defines the function or process. Once transformed into its canonical form, the interdependence between parameters can be uncoupled to expose the full degrees of freedom.

From an algorithmic perspective, the transformation to canonical form also reduces the computational complexity of applying the information operations L_i as defined in *IBC*. Anyone who has attempted to prove Kepler's laws of planetary motion using Newton's equation for gravity when choosing the coordinate system of the Earth as the basis, no doubt is aware of the complications that are introduced.

In effect, the information complexity can be viewed as a virtual complexity where the reduction to canonical form reorganizes the information to its simplest complexity. In this representation, *IBC* is certain to detect a more optimal algorithm.

Unfortunately, the adherence to canonical form tends to be lost or ignored when dealing with the actual implementation of an algorithm at the processor level. The art of computing appears more like an art than a rigorous set of well founded principles. Typically, an algorithm is assembled to fit the programming style or programming language that represents the fashions of the day. Algorithms are designed with little worry of cache utilization issues, problem sizes that are too large to remain in local memory, iterations schemes that maximize the inefficient manipulation of information, and so on. All of these examples are examples of the inefficient use of information that results in the notion that could be termed *virtual information complexity*.

In many optimization techniques, the reliance on randomness has played a significant role in the implementation of problem solutions that are intractable. Random treatment of problem solutions have proved to provide a convenient approach in surveying landscapes for optimization problems where the solutions space is vast and appears to follows no predetermined schedule or route. Monte Carlo techniques [4] are invaluable in the estimation of otherwise hard problems. However, in many situations the application of these approaches may be applied without merit but still used as an easy and straight forward (mindless) solution technique. The practical question to be asked is how can information be organized in a Monte Carlo approach in order to achieve a canonical form for information. Not surprisingly Sequential Monte Carlo techniques [5] have been proposed and studied, where *adaptive* information operations are applied to the Monte Carlo procedure to organize and more effectively utilize the previous iterated information. The value of reformulating information in terms of a canonical formulation should not be down graded as less important or orthogonal to the solution method [6; 7].

The approach proposed here introduces a notion of *Information-Based Algorithmic Design* where information flow of an algorithm is examined and then reformulated into a canonical mapping or an information remapping that better integrates the problem-solution domains. Rather than simply mapping a given algorithm to a particular processing unit, the task requires a fundamental analysis of the information complexity in terms of enhancing the specific information operator. In this approach a canonical information mapping is sought.

In the context of a canonical information flow description, the analysis is done at a higher level than that of *IBC*.

III. Background

In this presentation, the application of an information-based design approach for a neural network algorithm is considered. The importance of neural network applications and the advancement of their theory is widely acknowledge in both the academic and industrial communities. Entire conferences are held to disseminate the latest practices and techniques in optimization, search, and recognition problems. Although the neural network community has moved quite far from the anticipation that the science of neural networks might solve the fascinating mystery of the functional operation of the brain, the introduction of the artificial neural network (ANN) into the science of optimization techniques has had a serious impact on the solution of intractable problems.

The science of ANNs is still a challenging field. The basic ANN is simple but at the same time complicated. The basic network is formed from an input layer, an output layer, and if required a hidden layer of neuron nodes. Learning rules are conceptually easy to comprehend. Depending upon the application or non-application of supervisory rules, the learning procedure is incremental. These approaches are all well defined; however, the ambiguities of the problem domain makes the construction of a unique ANN difficult to define. This difficulty can be understood in terms of the number of inputs required for training, the number of initial nodes required for a given hidden layer, the relevance of the information contained within the input for training, the number of iterations required during the training process, etc. On the other hand, similar issues arise in other optimization techniques whether it be Genetic Algorithms or Monte Carlo techniques. So ANNs are not unique in these regards.

One intriguing question, which is the focal point of this presentation, is the role that information may play in facilitating and/or addressing some of the issues raised above. Clearly the use of heuristic is one time honored form of an information-based strategy to circumvent the learning process to achieve faster convergence. How does one identified and select the appropriate information is not always clear. Can an ANN be designed a priori without training? Is there a canonical form for neural network architectures

that is dictated solely by the problem specifications? If so how can it be realized?

In the sections that follow, a simple analysis of the perceptron neuron is presented within the context of its information-based complexity or information operators. This analysis then leads to a clustering algorithm whose associated architecture is uniquely defined in a general {n,m}-dimensional space and is shown to naturally support computational parallelism.

IV. Perceptron

The Perceptron is a classical neuron that dates back to the 1958 [8]. The perceptron computes a single output from multiple real-valued inputs by forming a linear combination according to its input weights. Mathematically the actual output can be written as

$$net = \sum_{i=1}^{n} w_i x_i$$

where w_i and x_i are the vectors of weights and inputs, respectively. In general, each iteration of the inputs and corresponding weights may be passed through some nonlinear activation function ϕ and a bias b, such that,

$$out = \phi(\sum_{i=1}^{n} w_i x_i + b)$$

or in vector notation

$$out = \phi(\mathbf{W^T x} + \mathbf{b}) \tag{2}$$

Although a single perceptron is shown not to be a very general learning algorithm, it is the building block of a much larger and more practical multilayer perceptron (MLP) network that consists of a set of source nodes forming the input layer, one or more hidden layers of computation nodes, and an output layer of nodes. The input signal propagates through the network layer-by-layer in a *feedforward* fashion.

A supervised learning rule for a single perceptron neuron with a learning constant α is given by

$$\Delta W_i = \alpha \delta X_i$$

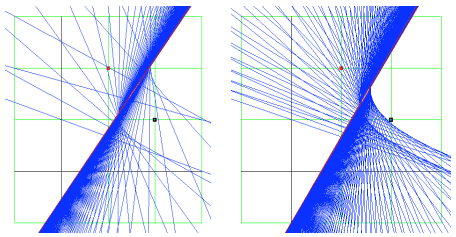
$$\Delta W_i = \alpha X_i(\mathbf{d} - sign(Net_i))$$

$$W_{i+1} = \Delta W_i + W_i$$

where

$$\delta = d - o$$

where d is the *desired* response and o is the actual output.



(a) (b)

Fig. 1. 2-Dimensional detection (a) $\alpha = 0.3$ (b) $\alpha = 0.1$

				Desired Output
Pattern1	1	2	+1	-1
Pattern2	2	1	+1	+1
Initial Weights	1	3	-3	
Final Weights (a)	1	-0.5	-0.5	
Final Weights (b)	1	-1	-0.5	

Table 1. A simple pattern detection example in two-dimensions

The information-based complexity of Equation (1) can be understood as an adaptive information operator where the i^{th} *net* result depends upon the previous $i-1$ sequentialized iterations. In a numerical analysis setting, such an information operator returns an approximation whose final value converges to within an ϵ of the answer. In the perceptron model, the information-based complexity of Equation (1) is overloaded in the sense that it represents both an approximation methods and an optimization search technique. In 2-dimensions, x_i may be viewed as a 2-dimensional vector that undergoes both a linear translation and rotation within a simple 2-dimensional region. This dual composition of transformations and approximation methods can readably be uncoupled into a much simpler *canonical* form that exposes these composite operations into pairs of non-adaptive information operations. The transition from adaptive to non-adaptive forms also implies the existence of a transformation from a sequential to a parallel algorithmic formalism. Figure 1 shows a simple pattern detection application of the perceptron training rule for a single neuron defined by Eqn. (1). A soft activation function is used and the effect of different learning constants α is displayed for a single problem specification (see Table 1).

Figure 1 illustrates two important features of the information operator as it is applied to the specific problem defined by Table 1. Upon closer examination of Figure 1, two separate independent (orthogonal) degrees of freedom are present. If the *line* is taken as the basic geometric unit then the *line* undergoes two separate but linearly independent motions: 1) translation and 2) rotation. It is through the learning procedure of determining ΔW_i where the coupling of these motions are performed. In addition, it is the value of α that dictates the ranges of rotations and the spacing between lines per iterations. Recall that Eqn. (1) defines an *adaptive* sequence of information updates, thus the ef-

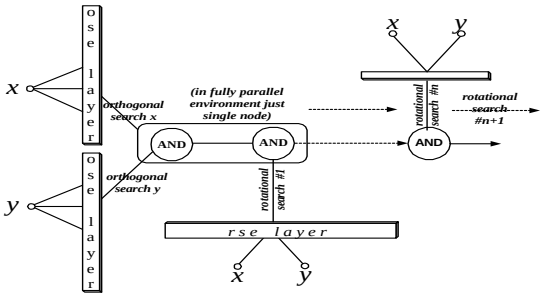


Fig. 2. Basic two-layer neural network.

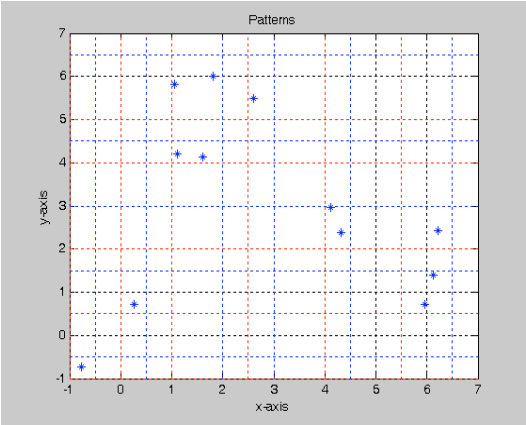


Fig. 3. Output of first input layer.

fects of ΔW_i and α suggest that the dependences between subsequent updates is an artifact of the organization of the information operator rather than the information required for convergence. In other words, a *non-adaptive* information operator exists that splits the orthogonal motions into separate translation and rotation operations. One immediate consequence of this approach is the capability of *non-adaptive* information operators to readily support parallel execution. Within this context, this presentation examines the consequences of a *canonical* re-formulation of the perceptron's order of rule application. In the following sections, a *canonical* neural network emerges that exhibits a fixed network complexity per iteration level and defines a sparse solution matrix.

V. A Canonical Perceptron Model

The orthogonality of the proposed neural network architecture consists of two essential layers: one input layer that performs an orthogonal search, and one hidden layer that performs rotational search. Figure 2 illustrates such an architecture that is applied to a two-dimensional space.

The first input layer performs an orthogonal pass through a search space in the x and y directions. This layer consists of two sets of nodes (in two-dimensions) that can be executed in parallel. Each node performs one orthogonal scan (in the x and y directions) of the search space. The output of the first input layer can be viewed as a pairwise intersection of the *possible* output signals. In the simple example of a two-dimensional scan space, each of these sets performs a y-horizontal and x-vertical striping of the search space, that results in a set of rectangular areas that may possibly contain the patterns as illustrated in Figure 3. In cases where the patterns are sparsely distributed, the computational complexity of searching the initial space can be dramatically reduced.

The second, *hidden* layer (depicted in Figure 2 as nodes with {x,y} inputs) performs a further reduction of the search space. This layer is similar to the first input layer but differs by a rotation as defined by the {x,y} coordinate pairs. This layer is necessary in order to uniquely eliminate all empty rectangular sub-zones (associated with the stripped two-dimensional space). This layer performs diagonal striping across the search space. Finally, the output signals from first and second layer are intersected, which results in a final set of non-empty clusters. Though further layers are not necessary, each additional layer will only sharpen the cluster of patterns within the space, hence improving clustering resolution.

The resolution of the pattern depends directly on scanning step size δ. The smaller the step size of δ, the better is the resolution. The lower boundary of this search is recognition of the whole set of patterns as belonging to a single cluster, while the upper boundary is recognition of clusters with single pattern belonging to it.

The complexity of the proposed neural network architecture goes as following. For a general n x m input layer, the corresponding set of nodes consists of n x m orthogonal search element (OSE) nodes, respectively (Figure 4). These input node signals are intersected in pairwise fashion using n x m AND nodes. The resulting signals are compared with the n x m hidden layer of Rotational Search Element (RSE) nodes (Figure 4), using the same number of AND neurons.

The OSE node architecture is illustrated by Figure 5. The OSE node consists of 3 neurons. Intersected signals from first two neurons result in "stripped" areas, for both dimensions of an orthogonal search space. X-low and X-high (Y-low and Y-high), are signals extracted and used in RSE nodes. The RSE node architecture is illustrated by Figure 6. The RSE node also consists of 3 neurons. Intersected signals from first two neurons result in a "stripped" area, this time performing a rotational search. The sum of signals X-low and X-high (Y-low and Y-high), is used for the biasing of the first two neurons in the node, as illustrated

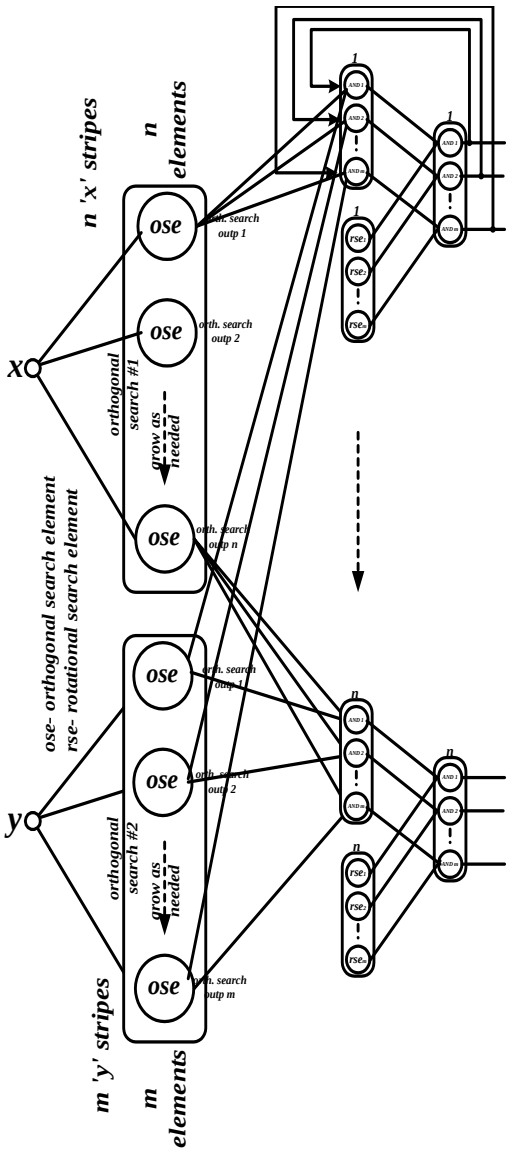


Fig. 4. The OSE Architecture.

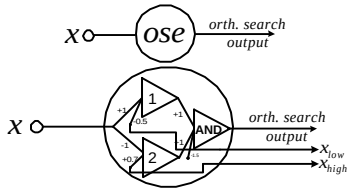


Fig. 5. The OSE Architecture.

by Figure 6. The process of combining signals through a network is illustrated by Figure 7, as a part of the complete network from Figure 4. Summation and AND neuron implementation is not explained in this paper as trivial.

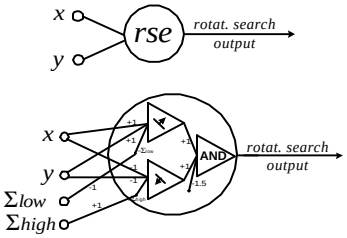


Fig. 6. The RSE Architecture.

As a neural network approach to clustering, the orthogonal search algorithm can be used regardless of dimensionality of search space.

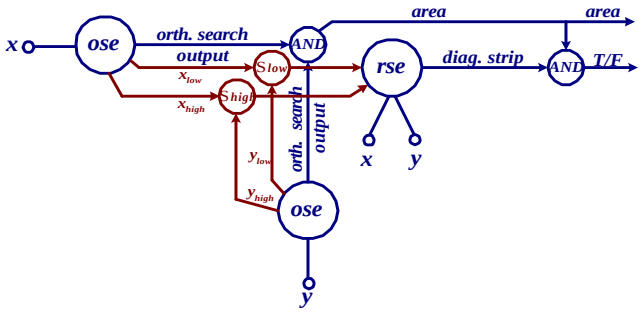


Fig. 7. The process of combining signals through a network.

VI. Orthogonality and Preconditioning

The orthogonality of the proposed neural network lends itself to the notion of "grow as needed," the principle of the cascade correlation network architecture [15]. The canonical combinations of orthogonal translations and rotations adds units of nodal layers to the network as required to achieve a certain degree of clustering resolution.

The algorithm developed above represents a canonical formulation of a clustering technique; however, it does have the capability to be used as a preconditioning search algorithm regardless of the dimensionality of the search space. As a preconditioning process, the orthogonality of the proposed algorithm can simplify the initial stages for deducing specific properties for a given search space. This acquired knowledge may ensure more accurate application of neural

network algorithms that are characterized by a high dependence on the starting parameterization set chosen. Algorithms such as Levenberg-Marquardt algorithm [9; 10] are examples of this dependence. They are proven to be very fast when the initial weight-set is chosen close to a solution but otherwise almost always fail to converge. Other algorithms based on gradient search, such as Error Back Propagation [11; 12; 13; 14], suffer from typical oscillation and flat spot problems when weights are chosen far from the solution.

VII. Orthogonal Search vs WTA

The Center of Gravity (COG) algorithms such as the Kohonen WTA algorithm [16; **?**] are highly dependent on the initial choice of parameters: the order of patterns applied; the initial configuration of the architecture; the initial weight-set; and the selected radius of attraction. The initial weight-set, if not judicially selected, may bias the centers of gravities and result in obstructing the learning of new patterns; thereby, reducing the possible number of final clusters detected. The order in which patterns are applied can influence the selection of the center of gravity for the final clusters. The weights determined by already seen (learned) patterns limit the mobility towards unseen patterns. In addition, the number of neurons initially used to construct the neural network also influences the final clustering of patterns. For example, too larger a number of initial neurons used in the construction of a network can result in the over-learning (over-fitting) of a problem, which could result in a larger number of particularly small clusters. On the other hand, too small a number of neurons may prevent the network from learning the relationship between new clusters resulting in less resolution.

The WTA approach is particularly sensitive to the distribution of patterns in the search space. For patterns that are already grouped, the WTA approach performs satisfactorily. This assumes that a priori knowledge about a problem's organization exists and is used. The result of each run of the WTA algorithm is, therefore, expected to be the same when patterns are fed to the WTA network a cluster at a time. For patterns that are scattered throughout the search space, the result of each run of WTA method may dramatically differ depending on initial choice of all the parameters. This applies especially to the order in which patterns are applied, as well as with the cluster radius chosen.

An ideal case for WTA are problems with very small, distinctively grouped patterns that are distributed at far distances. Here if the radius of attraction is much smaller than the distance between clustered patterns, the WTA approach is likely to return fast and repeatable results.

Even though different variations of the WTA approach may rely upon a single iteration through all the patterns, more general WTA algorithm may require a number of iterations. Although sometimes computationally very fast, the former WTA approaches has the negative effect of producing dramatically different clustered patterns for each of the different runs. This unsupervised approach does little to target a learning constant α that learns to anticipate the possible cluster positions. As a consequence, the knowledge gained from any one application of the WTA method does not guarantee an improvement on subsequent applications. In essence, the careful selection of the starting parameters is key criteria to the performance of the WTA method.

In contrast to the WTA algorithm, the orthogonal search algorithm is *deterministic* in the sense that the algorithm returns the same clustering of patterns, irrespective of the order that patterns are shown to the network. Hence, as additional patterns are subsequently added to the search space, no previous information about patterns already processed is lost. This property distinguishes the advantages of the orthogonal approach over the WTA method, and underscores the importance of formulating information-based operations in an orthogonal (independent) fashion.

The orthogonal search algorithm may result in a larger number of clusters; some of which may contain only a single pattern. For this reason, the orthogonal search may be more suitable for detecting patterns, rather than clusters. However, this is not a limitation. The resolution of the pattern depends directly on the scanning step size δ. The smaller the step size δ, the better is the resolution. At the lower end of a search boundary, the entire set of patterns is recognized as belonging to a single cluster, while at the upper end of a search boundary the recognition of clusters allows for a cluster containing a single pattern. Unlike the WTA method, the orthogonal search algorithm does not rely on the use of a learning constant, even though it is an unsupervised method.

Both the WTA and the orthogonal approaches generalize easily to higher-dimensional problems. In higher-dimensions, the orthogonal search may prove to be slower than WTA; however, the parallel and deterministic nature of the orthogonal search method can be exploited to maximize computational efficiency. In addition, the orthogonal search approach has the advantage of decoupling the problem domain into subspaces that can be explored systematically. This is done through the recursive application of the RSE architectural unit layer, where each pair of dimensions is investigated individually. The one most important architectural aspect of the orthogonal search approach is the recursive application of this fundamental RSE unit layer as illustrated in Figure 2.

In Figures 8 and 9 an application of a COG (with $\alpha = 1$) and the orthogonal approaches are illustrated, respectively. For the COG method, the possible clustering depend upon the value of α, so that the example of patterns used is susceptible to several different clustering possibilities depending upon the value selected for α. The red line in Figure

8 merely indicates the order of pattern scanning applied in this COG example.

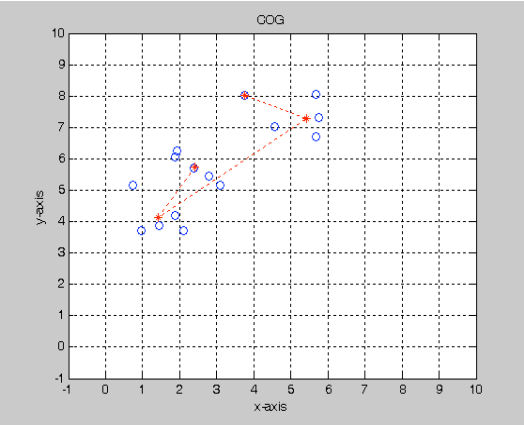


Fig. 8. COG clustering.

The orthogonal approach is independent of α so that the clustering is determine once and is never changed. The two orthogonal searches are depicted by red and blue lines. For a fix radius of attraction, all patterns grouped in one cluster are surrounded by a red line that serves as one boundary and a corresponding blue line on the opposite boundary. As a consequence of the cluster invariance for the orthogonal approach, a matrix representation of the cluster arrangement can be formulated. In this formulation, as the patterns are clustered into larger groups the matrix becomes sparse and thus the cluster locations can easily be manipulated during subsequent analysis. It should be noted that the rotational lines are not drawn in the figure; however, the rotations are applied to verify or eliminate patterns that do or do not occupy positions defined by the initial orthogonal search. In fact, this is the motivation to formulate the cluster positions in a sparse matrix representation.

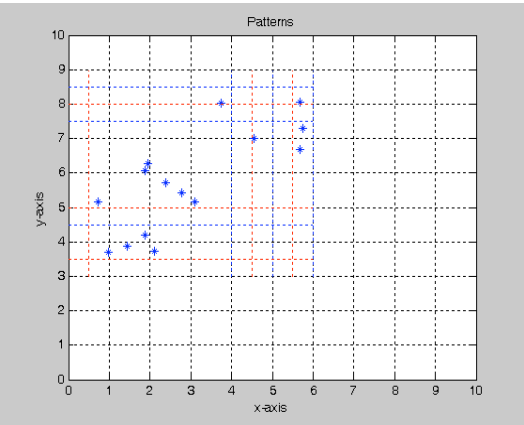


Fig. 9. Orthogonal clustering.

Figure 10 pictures the corresponding matrix associated with the results of the orthogonal scanning technique. For the orthogonal approach, this representation is fixed and provides a concise formulation of the clustered space. It is very gratifying to realize the over all simplistic structures that emerge from this canonical formalism.

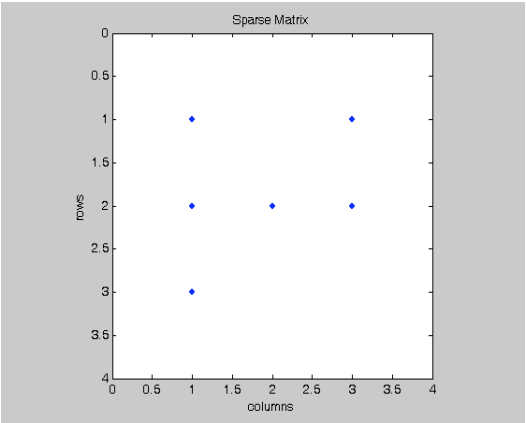


Fig. 10. Sparse matrix representation.

VIII. On the Canonical Structure of the Orthogonal Architecture

The orthogonal search neural network architecture is an unsupervised, feedforward type of network. The network is recursively applied to the search space defined by the problem domain in two- or higher-dimensions. The architecture is built from two basic layers that are combined recursively as it is applied to the search space. Although only four layers are necessary, additional layers can be added to enhance the sharpness of detecting, refining and smoothing cluster boundaries within the search space. The first two layers combine orthogonal search signals in the {n x m}-dimensional space, and their outputs combined with the rotational searches applied in the next layer. At each of these levels, the computational dependence allows for and defines the parallel aspects of the architecture. Within this architectural framework, a highly parallel implementation is easily achievable. This property is the result of the non-adaptive nature of the information operators defined by this architecture. Rather than the original formulation of the perceptron model where the information operations are defined by Equation (1), this new *canonically* simplified orthogonal architecture uniquely defines without ambiguities the number of nodes required within each layer of an {n x m}-dimensional network.

IX. Conclusion

The notion of information-based algorithmic design is an abstraction that may potentially provide a means to

achieve a canonical formulation of the solution technique. In this presentation, the information operator associated with the perceptron learning algorithm is separated into two independent components and used in a non-adaptive formulation that defines an ANN architecture with unambiguous number of nodes per translation and rotation layers. Specifically, the basic design of the proposed ANN network defines three {n x m}-dimensional layers that make up the basic building blocks of the network. The recursive application of this basic ANN block results in finer overall resolution. In addition, the non-adaptive nature of the proposed algorithm exhibits a canonical structure that is computational parallel and specifies uniquely the number of neural nodes within each layer as required to define the architecture exactly.

Both the WTA and the orthogonal algorithms belong to the unsupervised type of learning, where learning the desired outcome (number of clusters) is not known ahead of time. The orthogonal search algorithm excels at detecting patterns rather than clusters. However with predefined search step it can also produce clustering of the pattern space. An advantage of the orthogonal algorithm is the simultaneous execution of the two sets of input layer nodes. Once the input layers have completed their orthogonal {n x m}-dimensional search, the second layer of rotations can assimilate the knowledge discovered by the first layer in a parallel fashion as well. The intersection of these three layers, executed in parallel on three (sets of) nodes will result in a clustered space.

References

[1] J. F. Traub, G. W. Wasilkowski, and H. Woźniakowski (1988) "Information-Based Complexity," Academic Press Series In Computer Science And Scientific Computing Archive.

[2] B. N. Parlett (1992) "Some Basic Information on Information-Based Complexity Theory," Bulletin of the American Mathematical Society Vol. 26, No. 1, pp. 3-28.

[3] J. F. Traub and H. Woźniakowski (1992) "Perspectives on Information-Based Complexity," Bulletin of the American Mathematical Society Vol. 26, No. 1, Pages 29-52.

[4] M. H. Kalos and P. A. Whitlock (1986) "Monte Carlo Methods, Volume I: Basics," Wiley-Interscience Publications, John Wiley and Sons, New York.

[5] A. Doucet, N. de Freitas, and N. Gordon (2001) "Sequential Monte Carlo Methods in Practice," Springer.

[6] M. Gunzburger, R. E. Hiromoto, and M. Mundt (1996) "Analysis of a Monte Carlo Boundary Propagation Method," Journal of Computers and Math. with Applic. Vol. 31, No. 6, pp. 61-70, (1996).

[7] R.E. Hiromoto and R.G. Brickner (1991) "Empirical Results of a Hybrid Monte Carlo Method for the Solution of Poisson's Equation," Applications of Supercomputers in Engineering, Boston, Mass., August 22-25.

[8] F. Rosenblatt (1958) "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," Psychological Review, v65, No. 6, pp. 386-408.

[9] Levenberg, K. (1944) "A Method for the Solution of Certain Problems in Least Squares," Quart. Appl. Math. 2, 164-168.

[10] Marquardt, D. (1963) "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," SIAM J. Appl. Math. 11, 431-441.

[11] Rumelhart, D.E., McClelland, J.L. (1986) "Parallel Distributed Processing: Explorations in the Microstructure of Cognition," Vol.1, Cambridge, Ma, MIT Press.

[12] Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986) "Learning Internal Representation by Error Propagation," Parallel Distributed Processing, Vol.1, pp.318-362, MIT Press, Cambridge, MA.

[13] J. M. Zurada (1992) "Introduction to Artificial Neural Systems," West Publishing Company.

[14] Sejnowski T.J., Rosenberg, C.R. (1987) "Parallel Networks that Learn to Pronounce English Text," Complex Systems 1:145-168.

[15] Fahlman & Lebiere (1990) "The Cascade-Correlation Learning Architecture," in Advances in Neural Information Processing Systems 2, D.Touretzky, ed., San Mateo, CA, Morgan Kaufmann, pp.524-532.

[16] Kohonen, T. (1982) "Self-organized formation of topologically correct feature maps," in Biological Cybernetics, 43:59-69.

[17] Kohonen, T. (1988) "Self-Organization and Associative Memory," 2nd Ed. New York, Springer-Verlag.