computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Journal
of Computing

INFORMATION-BASED ALGORITHMIC DESIGN OF A NEURAL NETWORK CLASSIFIER

Robert E. Hiromoto [1], Milos Manic [2]

[1] University of Idaho, Moscow, Idaho 83844-1010, USA, hiromoto@cs.uidaho.edu
[2] 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.*

Keywords: *Information-based complexity, artificial neural network, adaptive, non-adaptive, canonical form, perceptron, clusters.*

1. 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 relies 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 have 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 a problem. The number of iterations I_n measures the solution rate 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

$$S : F \to G \; \forall f \in F \qquad (1)$$

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 information $r(N)$ that does not exceed some error ε, where N is the computed information about f and

$$N(f) = \{L_i(f) \mid i = 0, 1, \ldots, n\}, \quad L_i \in \Lambda, \qquad (2)$$

then U is guaranteed to be an ε-*approximation*. This attention to information operations furnishes a comparison of algorithmic performance based on the information operator that is used. As an iterative process, Eqn. (2) can be formulated in two distinct forms: 1) non-adaptive information operators defined by

$$N(f) = \{L_0(f), L_1(f), \ldots, L_n(f)\}, \quad \forall f \in F \qquad (3)$$

where the partially computed information about f only depends upon the current iterative state. Thus within each iteration, the operation $L_i(f)$ defines independent processes that can be performed in parallel; or 2) adaptive information operators defined by

$$N(f) = \{L_0(f), L_1(f{:}\lambda_0), \ldots, L_n(f{:}\lambda_0,\lambda_1 \ldots \lambda_{n-1})\}, \qquad (4)$$

where λ_ls represent various combinations of previously learned information; and therefore, requires a sequential sweep throw each iteration step.

Within the context of the IBC representation, 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 possible application than originally intended. 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.

2. 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 has 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 direct 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].

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 re-mapping 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 canonical information flow description, the analysis is done at a higher level than that of *IBC*.

3. 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 are widely acknowledged 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 networks (ANNs) 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 simplistic approach to an artificial neuron can become very complex with numerous possibilities of combining these units and learning rules. The basic network is formed from an input layer, an output layer, and if needed a hidden layer of neuron nodes. Learning rules are conceptually easy to comprehend. Depending upon simultaneous or iterative data feed, the learning procedure is batch or incremental. These approaches are all well defined; however, the ambiguities of the problem domain make the process of building a universal ANN solution difficult to define. This difficulty can be understood in terms of the network parameters such as 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 is 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 a heuristic is one time-honored form of an information-based strategy to circumvent the learning process to achieve faster convergence. How one identifies and selects the appropriate information is not always clear. A few approaches distinguish themselves in this realm.

One approach known as “design by training” is very effective when it comes to applications where knowledge of a problem exists. For example, two-layered Counter Propagation Networks (CPN) are very effective with hetero-associative memory type of applications [7].

Another approach known as Genetic Algorithm Neural Networks (GANNs) relies on determination of neural network parameters by genetic search. The genetic approach is used to increase robustness in network training with respect to the problem of convergence [8-11].

Yet another approach refers to data mining of new information in existing data and articulating that knowledge as an extra part of the architecture. The representatives of such approaches are Functional Link and Polynomial Networks [12,13].

However, a universal approach to determining network parameters is not known. 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 following sections, 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}-matrix space and is shown to support computational parallelism.

4. PERCEPTRON

The Rosenblatt’s concept of a Perceptron neuron dates back to the 1958 [14]. The perceptron computes a single output from multiple real-valued inputs by forming a linear combination according to its input weights. Mathematically the actual net value can be written as

$$net = \sum_{i=1}^{n} w_i x_i + b \qquad (5)$$

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,

$$out = \phi(\sum_{i=1}^{n} w_i x_i + b) \qquad (6)$$

or in vector notation

$$out = \phi(W^T x + b) \qquad (7)$$

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. Feedback networks are not considered in this presentation.

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

$$\nabla W_i = \alpha \delta X_i$$

$$W_{i+1} = \Delta W_i + W_i \qquad (8)$$

$$\delta = d - o = (d - sign(Net_i))$$

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

The information-based complexity of Eqn. (4) represents an adaptive information operator where the *i*-th *net* result depends upon the previous *i*-1 sequential iterations. In the perceptron model, Eqn. (4) is overloaded in the sense that it represents both an approximation methods and an optimization search technique. In two-dimensions, x_i may be viewed as a two-dimensional vector that undergoes both a linear translation and rotation within a simple two-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 operators. The transition from *adaptive* to *non-adaptive* forms also implies the existence of a transformation from a sequential to a parallel algorithmic formalism. Fig. 1 and Table 1 show a simple pattern detection application of the perceptron training rule for a single neuron defined by Eqn. (8). A soft activation function is used and the effect of different learning constants α can be observed. Fig. 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 Fig. 1, two separate independent (orthogonal) degrees of freedom are present. If the *separation line* is taken as the basic geometric unit then the *line* undergoes two linearly independent motions: 1) translation and 2) rotation. It is through the learning procedure of determining ΔW_i where the coupling of these motions is performed. In addition, it is the value of α that dictates the ranges of rotations and the spacing between lines per iteration. Thus the effects 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, an information operator exists that splits the perceptron procedure into separate translation and rotation operations. Within this context, we examine 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.

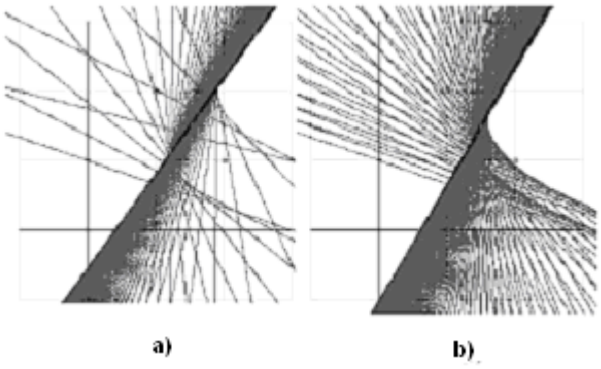

Fig. 1 - 2-D detection (a) α = 0.3 (b) α = 0.1

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

				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	

5. 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 output layer that performs a rotational search. Fig. 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 sets of nodes performs an orthogonal scan (one set in the *x* and another in *y* direction) of the search space. Each set of nodes therefore produces a set of stripes. The output of the first layer can be viewed as a set of intersections of these stripes. 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 Fig. 3.

The second, *output* layer (depicted in Fig. 2 as nodes with {x,y} inputs) performs a further reduction of the search space. In case of a 2-dimensional 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 empty rectangular sub-zones (associated with the stripped two-dimensional space). This layer performs diagonal striping across the search space. 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. In cases where the patterns are sparsely distributed, the computational search time for the initial space can be dramatically reduced if the value of δ is chosen appropriately.

The complexity of the proposed neural network architecture goes as follows. Assume a two-dimensional rectangular region of lengths l_x and l_y, where $l_k = l_{kr} - l_{kl}$ defines the right-left boundary extent of the space in the $k = x$ or y direction; and suppose that we select the orthogonal search space increments as δ_x and δ_y, such that, n searches

$$n_x = \frac{l_x}{\delta_x} \quad and \quad n_y = \frac{l_y}{\delta_y} \tag{9}$$

are performed incrementally along the x direction and y direction. An orthogonal search must be completed before the corresponding rotations can be performed. The results of the search network can be represented by an n_x x n_y matrix of cluster (pattern) positions. This matrix can be used in subsequent cluster analysis.

For an n x m input layer, the corresponding set of nodes consists of n x m Orthogonal Search Element (OSE) nodes, respectively (Fig. 4). These input node signals are intersected in pair-wise fashion. The n x m output layer nodes of the Rotational Search Element (RSE) refine the resulting signals.

The orthogonal search can be generalized to a d-dimensional space. For simplicity, assume that the region of interest is a higher dimensional cuboid with sides of length $l_1, l_2,..., l_d$. The corresponding architecture (not presented in this paper) would be embellished by n_x x n_y d-dimensional d hyperplanes. For an orthogonal search in the i-th dimension, n_i number of incremental hyperplane displacements is performed along the length l_i

$$n_i = \frac{l_i}{\delta_i}$$

where δ_i is the orthogonal search increment along the direction l_i as defined by Eqn. (9).

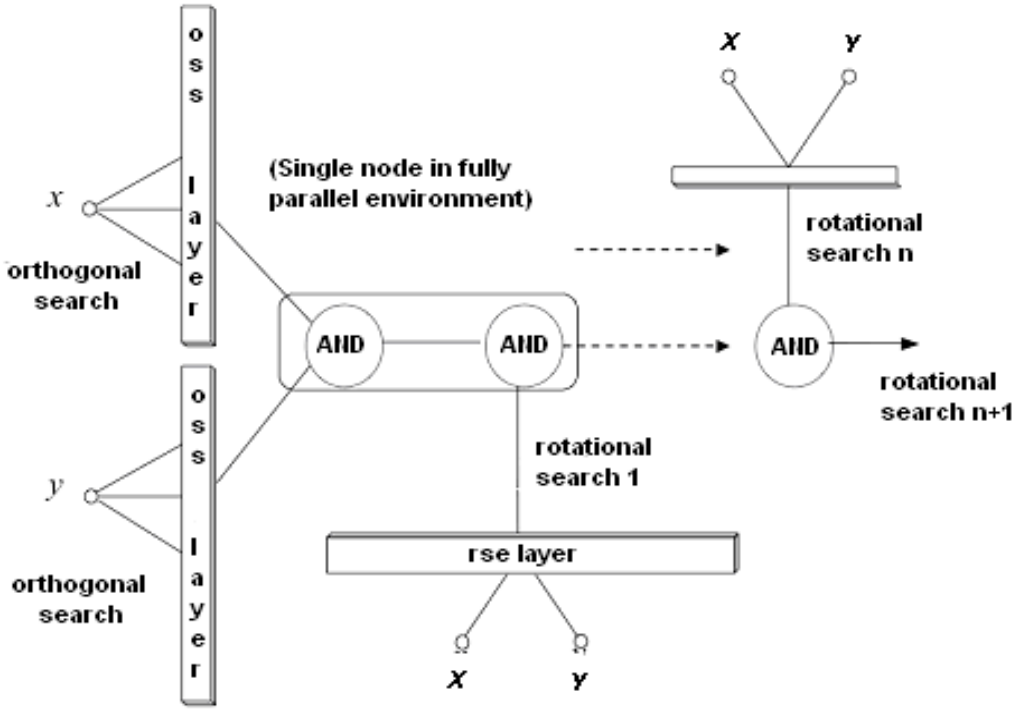

Fig. 2 - Basic two-layer neural network

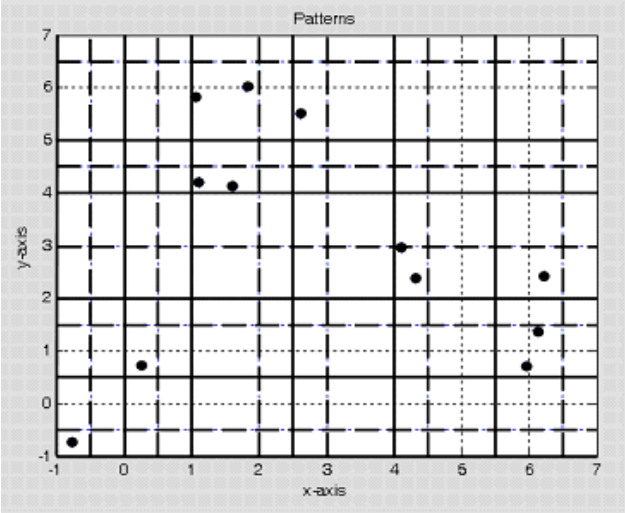

Fig. 3 - Output of first input layer

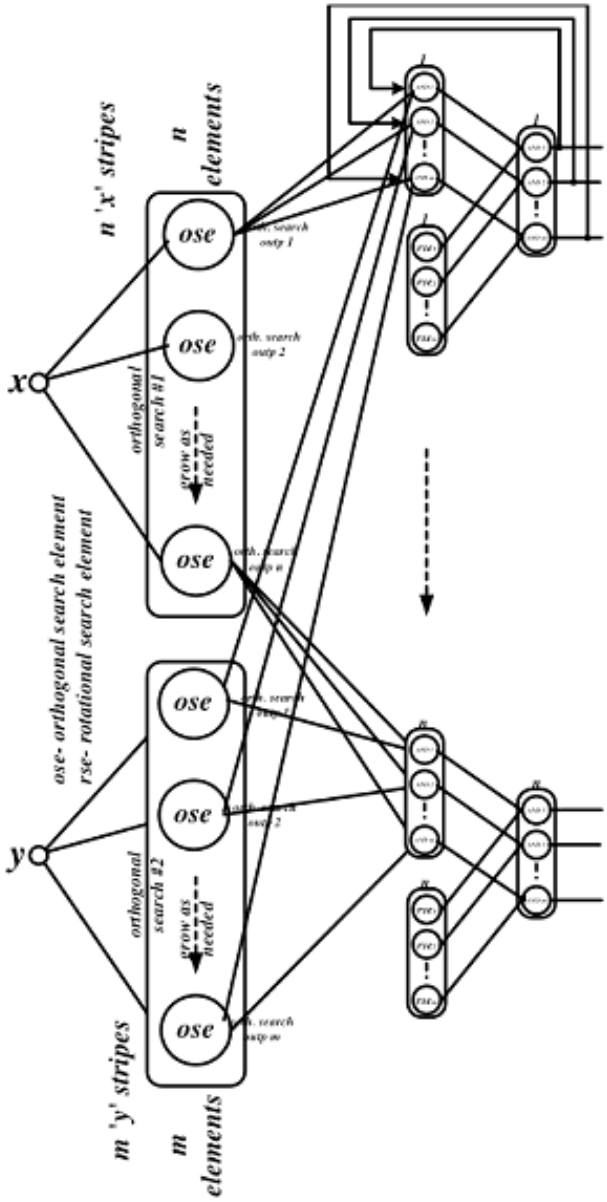

Fig. 4 - The OSE Architecture

The complexity for the total number of orthogonal search for all n_x x n_y hyperplanes; therefore, is

$$\Theta(n_x \times n_y)$$

The corresponding rotational complexity is of the same order $O(n_x \times n_y)$, where the big-O notation is used to emphasize that all enclosed volumes may or may not necessarily contain a pattern. The big-O here should, therefore, be understood as defining an upper bound.

The OSE node architecture is illustrated in Fig. 5. The OSE node consists of 3 neurons. Intersected signals from the 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 in Fig. 6. The RSE node also consists of 3 neurons, which performs a rotational search about the "stripped" areas. 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 by Fig. 6. The process of combining signals through a network is illustrated by Fig. 7, as a part of the complete network from Fig. 4.

6. PARALLEL ARCHITECTURE

The OSE architecture (Fig. 4) is parallel. The orthogonal searches can be performed sequentially or in parallel. Only synchronization points are required between the end of the parallel orthogonal searches and the beginning of the rotation phase. In other words, an orthogonal search must be completed before the corresponding rotations can be performed. This barrier synchronization is essential in identifying those strips that may possibly contain patterns in which rotations are required for verification. Assuming that all orthogonal searches complete in the same amount of time, the corresponding rotations can then be applied sequentially or in parallel, or in a data flow scheduled manner.

For a parallel architecture with a limited number of parallel processing units P_{cpu}, the maximum number of parallel steps (iterations) I_p required by the OSE architecture to complete the orthogonal search is given by

$$I_p = \left\lceil \frac{n_x \times n_y}{P_{cpu}} \right\rceil .$$

In a similar fashion, if the number of parallel processing units P_{cpu} limits the number of parallel rotations then a similar expression can be written down.

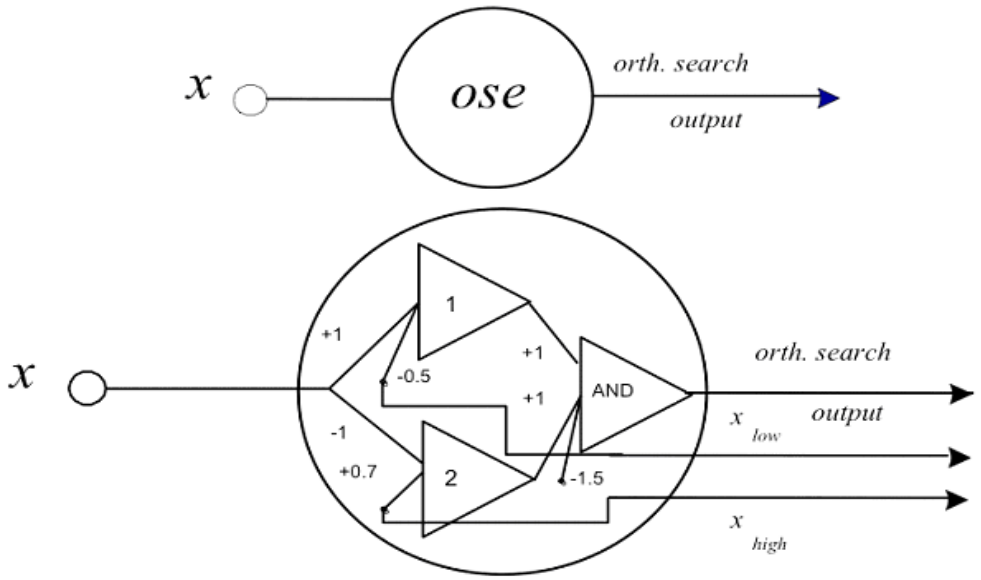

Fig. 5 - The OSE Architecture.

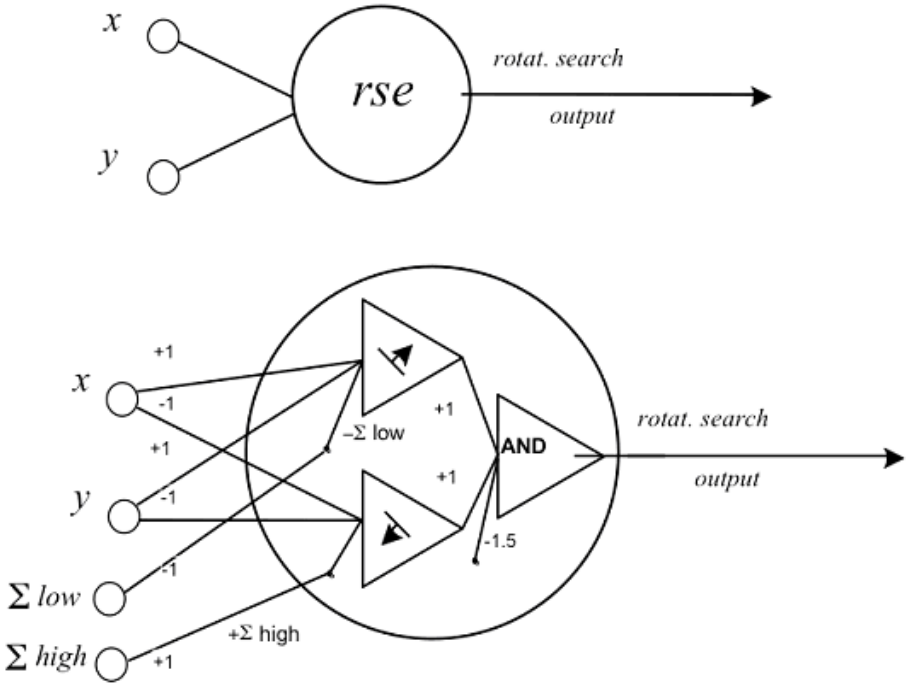

Fig. 6 - The RSE Architecture

7. ORTHOGONALITY

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 add 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 can also 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 [15,16] 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 [12,19-20], suffer from typical oscillation and flat spot problems when weights are chosen far from the solution.

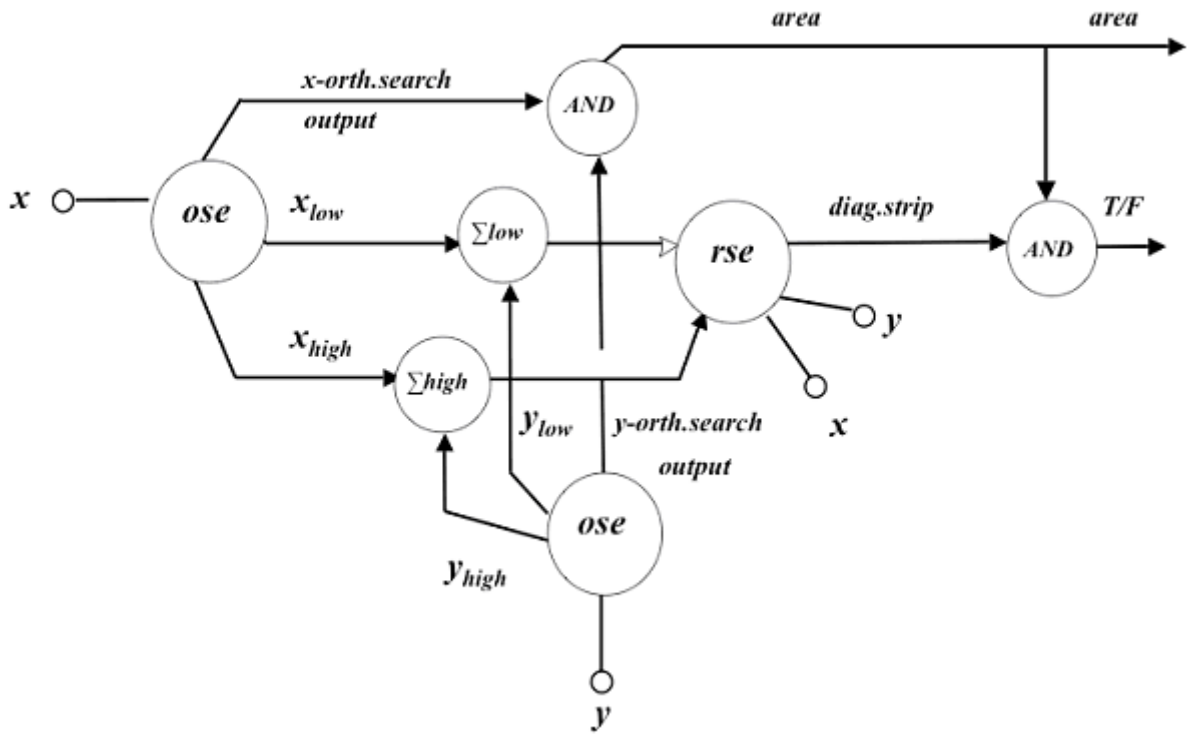

Fig. 7 - The combining signal network

8. HIGHER-DIMENSIONAL PROPERTIES

Given a number of patterns N_p, a volume per pattern (space pattern) density can be defined as

$$D_p = \frac{1}{S_d \cdot N_p}$$

where

$$S_d = \frac{d}{l_1 l_2 \cdots l_d}$$

is the space density, and l_i for i = 1, 2, ... , d is the length of each of the d-orthogonal directions.

In a parametric space of dimension d, similarities of patterns can be correlated by their locality within common clusters. As such, regions with anomalous patterns will be designated with low pattern densities. This suggests that searching around large clusters of related events could accelerate the search for unusual events. Although not discussed in this paper, this will introduce a possible learning strategy for adapting the search space parameter δ.

9. ORTHOGONAL SEARCH vs. WTA

The Center of Gravity (COG) algorithms such as the Kohonen WTA algorithm [21,22] 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, influencing the number of final clusters detected. The order in which patterns are applied can also influence the selection of the center of gravity for the final clusters. The weights determined by the patterns that have already been learned 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 1) patterns are fed to the WTA network a cluster at a time, and 2) the process of determining a

cluster center is not based on a weighted calculation. For patterns that are scattered throughout the search space, the result of each run of WTA method may dramatically differ depending on 1) the initial choice of all the parameters; and 2) the order in which patterns are applied. The initial parameter choice applies especially to the cluster radius chosen.

Ideal cases for WTA are problems with distinctively grouped patterns that are distributed at far distances. Here if the radius of attraction is much smaller than the distance between clusters, 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 have the negative effect of producing dramatically different clustered patterns for each of the different runs. These iterative approaches do little in learning 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 in which the 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. A repetitive orthogonal search would naturally increase pattern recognition resolution up to a single pattern. For this reason, the orthogonal search may be very effective 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. 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. Furthermore, the orthogonal search algorithm returns the same result with each run.

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 still be exploited. 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 algorithm.

In Fig. 8 and 9 an application of a COG (with α = 1) and the orthogonal approaches are illustrated, respectively. For the COG method, the possible clustering depends 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.

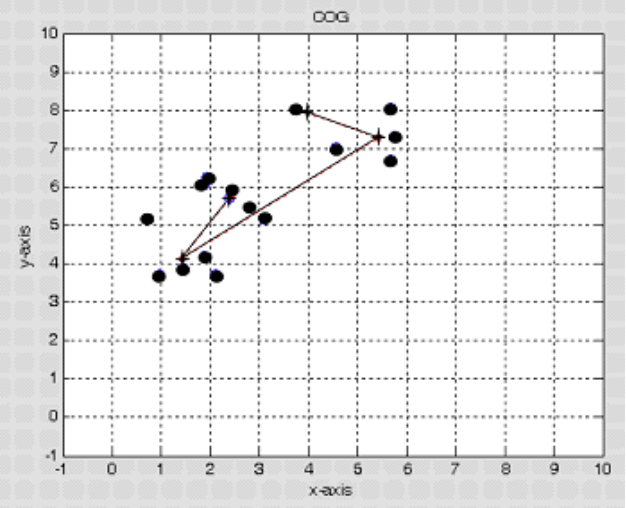

Fig. 8 - COG clustering

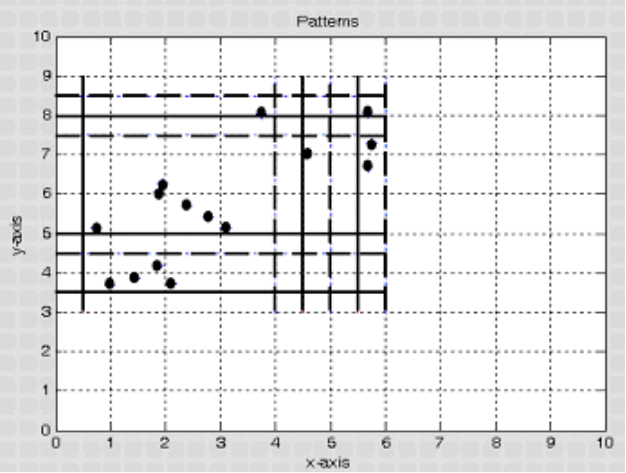

Fig. 9 - Orthogonal clustering

In the orthogonal approach the learning constant α does not even exist, hence the clustering is determined once and is never changed. The heavy

solid and heavy dashed black lines depict the two orthogonal searches. For a fixed radius of attraction, a solid black line that serves as one boundary and a corresponding dashed black line on the opposite boundary surround all patterns grouped into one cluster. 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. Fig. 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.

In Fig. 9, the bounded or stripped areas that contain the clustered patterns enclose areas that are filled with pattern-less regions. For this reason, 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 in part the motivation to formulate the cluster positions in a sparse matrix representation.

As a simple example, Fig. 11 illustrates the rotation about a stripped area, where the lines of rotation (dashed) are given by $y = -x + 1.5$ and $y = -x + 2.7$. The rotations in this example are chosen to be at 45°s to the orthogonal {x,y} but this is not a requirement.

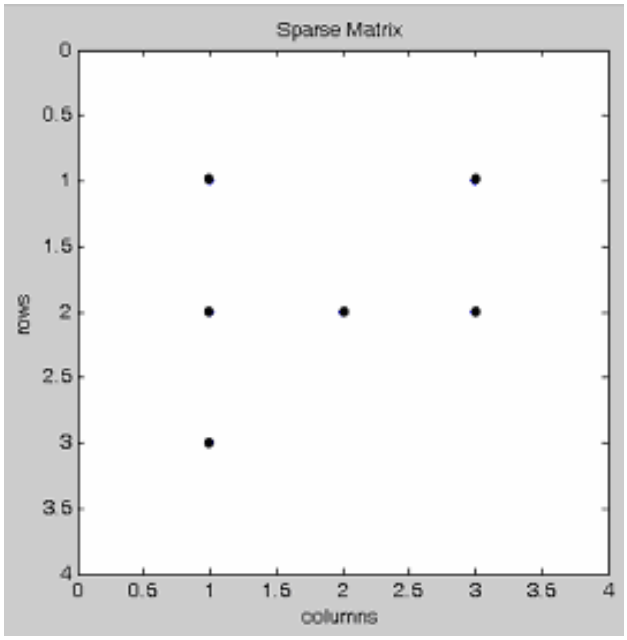

Fig. 10 - Sparse matrix representation

10. ARCHITECTURAL STRUCTURE

The orthogonal search neural network architecture is an unsupervised, feed-forward 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 two layers are necessary, additional layers can be added to enhance the sharpness of detecting, refining and smoothing cluster boundaries within the search space.

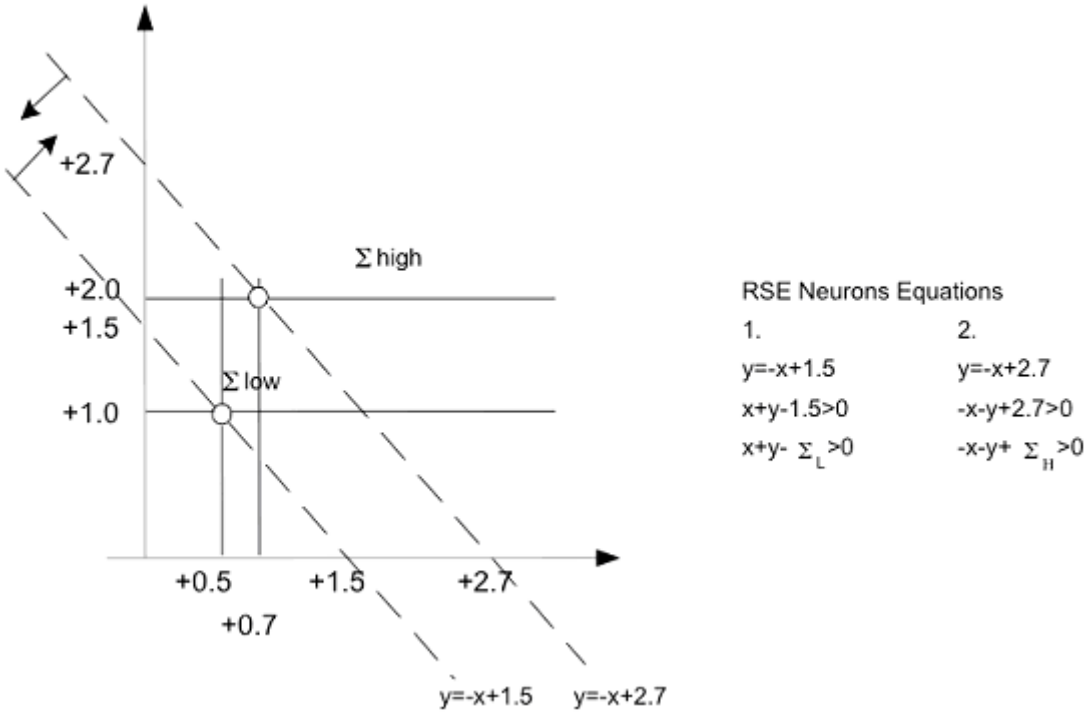

Fig. 11 - Rotational Example

The first layer combines orthogonal search signals in the $\{n \times m\}$-matrix space, and their outputs are 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 Eqn. (1), this new *canonically* simplified orthogonal architecture uniquely defines without ambiguities the number of nodes required within each layer of an $\{n \times m\}$-matrix network.

11. CONCLUSION

The notion of information-based algorithmic design is an abstraction that offers potential to achieve a *canonical formulation* of solution techniques. 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 $\{n \times m\}$-layers that make up the basic building blocks of the network. The recursive application of this basic ANN block results in finer overall resolution. It is important to realize that the proposed ANN architecture is deterministic. The number of nodes in the input layer and in the secondary rotational (hidden) layer is specified by the dimensionality of the problem space; as well as, the degree of parallelism chosen. What remains undetermined is the orthogonal displacement δ that defines the incremental stride taken along each orthogonal direction. However, this ambiguity can be recast in an adaptive and constructive way that allows for a variable stride step size to rapidly sweeping through each orthogonal direction. The details of this adaptive approach are left for a future work. The *non-adaptive* nature of the proposed algorithm exhibits a *canonical structure* that is computationally parallel and specifies uniquely the number of neural nodes within each layer as required to define the architecture exactly. The nature of the parallelism allows for a divide-and-conquer approach for limited number of processors where regions of the search space can be subdivided and scheduled as processors become available. In addition, the advantages of high-level parallelism can be captured in hardware. As such, embedded systems can be designed to enhance the efficiency of this algorithm.

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 a 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 $\{n \times m\}$ search, the second layer of rotations can assimilate the knowledge discovered by the first layer in a parallel fashion as well. The final result is a clustered space.

12. REFERENCES

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

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

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

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

[5] Doucet, N. de Freitas, and N. Gordon. Sequential Monte Carlo Methods in *Practice, Springer* 2001.

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

[7] Hecht-Nielsen, R. Counter-Propagation networks, *IEEE First International Conference on Neural Networks,* Volume II, 1987.

[8] V. Maniezzo. Genetic evolution of the topology and weight distribution of neural networks, *IEEE Transactions on Neural Networks,* 1994, Vol. 5, No.1, pp. 39-53.

[9] S. Mizuta , T. Sato, D. Lao, M. Ikeda, T. Shimizu. Structure design of neural networks using genetic algorithms, *Complex Systems,* 2001, 13, pp. 161-175.

[10] K. Balakrishnan , V. Honavar. Evolutionary

design of neural architectures – preliminary taxonomy and guide to literature, Artificial Intelligence Group, Iowa State University, Ames, *Tech. Rep. CS TR#98-01*, 1995.

[11]Manic, M. Wilamowski, D. Robust Neural Network Training Using Partial Gradient Probing, *IEEE Int. Conf. on Industrial Informatics, INDIN 2003, August 21-24, Banff, Alberta, Canada.*

[12]J. M. Zurada. Introduction to Artificial Neural Systems, West Publishing Company 1992.

[13]Wilamowska, K., Manic, M. Unsupervised pattern clustering for data mining, *IECON'01 - 27. Annual Conference of the IEEE Industrial Electronics Society*, Denver, Colorado, Nov 29 to Dec 2, 2001, pp.1862-1867.

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

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

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

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

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

[19]Sejnowski T.J., Rosenberg, C.R. Parallel Networks that Learn to Pronounce English Text, *Complex Systems*, 1987, Vol. 1, 145-168.

[20]Kohonen, T. Self-organized formation of topologically correct feature maps, in *Biological Cybernetics*, 1982, 43:59-69.

[21]Kohonen, T. Self-Organization and Associative Memory, *Springer-Verlag*, 1988 2nd Ed. New York.

***Dr. Robert E. Hiromoto**, received his Ph.D. degree in Physics from University of Texas at Dallas. He is professor of computer science at the University of Idaho. His areas of research include the automated flight formation of Unmanned Aerial Vehicles, Information-based design of sequential and parallel algorithms, decryption techniques using set theoretic estimation, and parallel graphics rendering algorithms for cluster-based systems.*

***Dr.Milos Manic**, IEEE Senior Member, received his Ph.D. degree in Computer Science from University of Idaho, Computer Science Dept. He received his M.S. and a Dipl.Ing. in Computer Science and Electrical Engineering from the University of Nis, Faculty of Electronic Engineering, Serbia. He is an assistant professor at the UI Computer Science Dept. and adjunct faculty with the UI ECE Dept. He is also a program director for University of Idaho CS & ECE programs in Idaho Falls, and serves as an IEEE IES AdCom member and webmaster. His areas of research include artificial neural networks, fuzzy logic, and performability of fault tolerant systems.*