Intelligent Facility Layout for Information Network System Infrastructure

Charles C. Willow

Management Information Systems & Management of Technology
Department of Management and Marketing
School of Business Administration
Monmouth University
West Long Branch, NJ 07764-1898
U. S. A.

ABSTRACT

Building a cellular yet effectively modular infrastructure is critical for major productive systems such as information and engineering, among others. In particular, flexible, optimized, and dynamic layouts of facilities are critical for successfully incorporating infrastructures for organizations. Application examples range from determining physical topologies for local area networks to cellular, agile manufacturing, in which the problem is closely associated with determining the optimal number of clusters of workgroups, comprising servers, wireless access points, PC's, printers, routers, firewalls, or automated welding machines.

A general-purpose neural-network application is suggested in this paper for real-time intelligent facility layout. Classical clustering techniques such as Hungarian algorithm, Group Technology, and BLOCPLAN have been proven to be ineffective, relative to the method outlined in this research. The effect of the development is illustrated with a simplified numerical example.

❑

Keywords: Facility Layout, Information Technology Infrastructure, Network Topology, Clustering, Neural Network, Feedforward Network, Perceptron, Linear Classifier.

1. Introduction

As the information era matures in the twenty-first century, the focus of technology development has shifted from heavy engineering such as manufacturing to information systems. Indeed, the two systems are different in many ways in that the former delivers tangible products, whereas the latter is concerned with virtual product development. However, this may not imply that the two are completely incompatible. In fact, they share a number of problems, among which are 'systems' and 'infrastructure' related (Willow and Yang 1997, Willow 2007a).

Building a cellular yet effectively modular infrastructure is critical for establishing the information system infrastructure. In particular, flexible, optimized, and dynamic layouts of facilities are critical for successfully incorporating infrastructures for organizations. Application examples range from determining physical topologies for local area networks to cellular, agile manufacturing, in which the problem is closely associated with determining the optimal number of clusters of workgroups, comprising servers, PC's, printers, routers, firewalls, or automated welding machines.

By far, the determination of 'physical topologies' for information networks, both wired and wireless, is perhaps the most important (Mueller *et al.* 2004), since networks serve as a foundation support technology for successful *e*-commerce or *m*-commerce systems. Figure 1 represents the problem context.

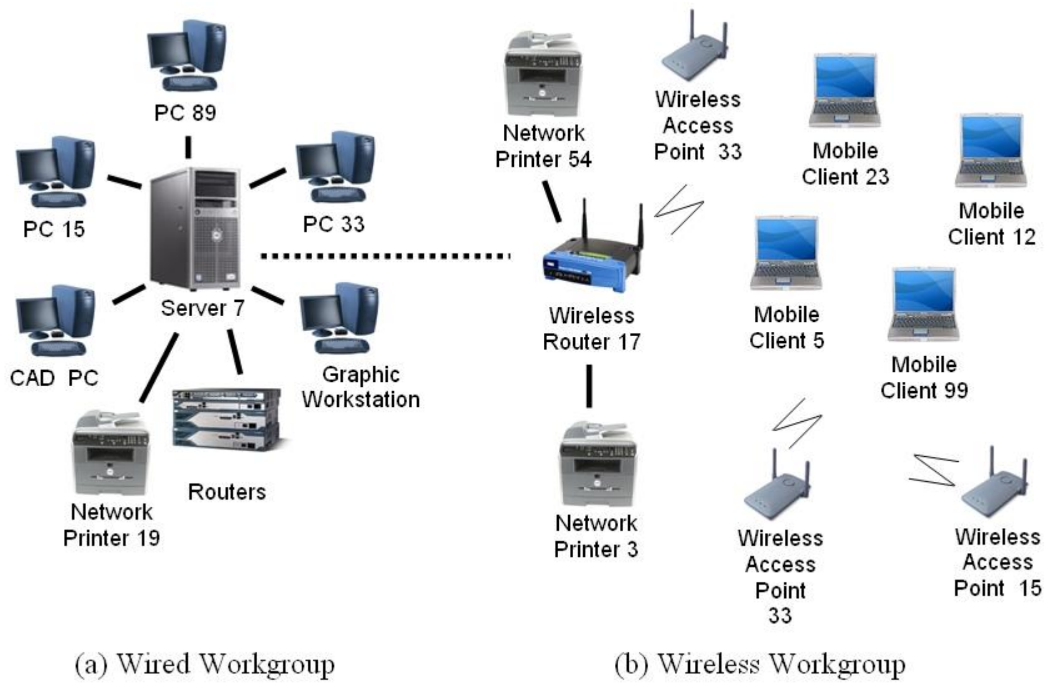


Figure 1. Information Network Workgroups

Each information network requires a *router*, which delivers data packets of information from the source node (*i.e.* sender) to the appropriate sink node (*i.e.* receiver). A server of the network is responsible for controlling and monitoring the network domain, often referred to as a 'workgroup'. As required, a group of subnetworks or subnets for short, for the server may be created by connecting it to another router (*viz.* daisy chaining), as indicated by dashed line(s) in Figure 1. For greater modularity and reliability, however, building recursive subnets is not recommended. Instead, dedicated server for each domain is considered the optimal. Given the increasing number of PC's, workstations, and other types of clients, however, allocation of these components/clients to the major service nodes such as servers, routers, network printers, and even firewalls, among others, is becoming prohibitively complex. The problem is compounded for wireless networks, in which reception of radio signals or infrared light waves must be accounted for, prior to generating clusters of workgroups for the network infrastructure. For a relatively large-scale organization, designing highly modular Local Area Networks (LAN) is imperative. LANs effect not only the reliability of overall information system, but more importantly, its scalability to Wide Area Network (WAN) and even Corporate Networks (CN).

A neural-network application is suggested for real-time intelligent facility layout. In consequence, on-line, mass-customized facility layouts are expected to be available to the information systems manager. Similar objectives for manufacturing were suggested by Tseng *et al.* (1997). Classical clustering techniques such as Hungarian algorithm, Group Technology, and BLOCPLAN have been proven to be ineffective, relative to the method outlined in this research. The effect of the development is illustrated with a simplified numerical example.

Group Technology (GT) is a philosophy that originated from manufacturing systems, in which similar parts are identified and grouped together to take advantage of their similarities in manufacturing and/or design (Ham *et al.* 1985, Groover 1987, Chang *et al.* 1991). Similar parts are arranged into part families, and each family would possess similar design and manufacturing characteristics. A robust and consistent part-classification-and-coding scheme is a prerequisite in implementing GT. To this end, each part must be identified based on its unique code.

Hungarian algorithm-based Product Family Architecture (PFA) and Family-Based Design (FBD) have long been the foundation for *Group Technology* (GT), which involves finite number of iterations of row and column exchanges of two-dimensional part-machine vector. The problem of determining *physical topologies* for various numbers of information networks is highly analogous to the part-family clustering for manufacturing. This indicates that alternative methods to GT, introduced in Willow (2002), may be applicable to information systems as well, and perhaps be accepted with higher effectiveness due to their need for real-time communications and integrated modules.

This paper presents an alternative to traditional applications to GT for generating optimal cluster or workgroups. A *neural network* is employed in forming the workgroups, in which a linear classifier is established with a classical feedforward network. The proposed method, through a number of numerical experiments, has been proven to outperform such conventional methods as Part Family Analysis (PFA) (Groover 1987, Chang *et al.* 1991) and BLOCPLAN (Donaghey 1991, Donaghey and Pire 1991).

Organization of the paper follows. In section 2, possible clustering with neural network is illustrated, followed by a linear classifier, constructed with perceptrons in section 3. The linear classifier is extended to accommodate nonseparable patterns in section 4. Illustration of the neural-network is provided numerically with an example in section 5. Section 6 summarizes the paper with conclusions and notes for future research.

2. Clustering with Neural Networks

Clustering part families is the objective of Group Technology (GT), which initiated a series of algorithm developments, including but not restricted to rank-order clustering by King, direct clustering, single-linkage clustering, and average-link clustering (Chang *et al.* 1991). Further, Donaghey (1991) introduced BLOCPLAN for facility layout planning to be used by large-scale plant as well as cellular manufacturing systems. However, these algorithms are associated with time-consuming iterations. Most importantly, major drawbacks of these methods are that:

- The algorithms are applicable exclusively for linearly separable patterns. Linearly *non-separable* patterns are thus under arbitrary control, resulting in great inaccuracies.
- There was no provision for user flexibility, such as inclusion of preferred number of clusters *prior to* the execution of these algorithms. In essence, less *a priori* information was available to the decision maker and/or management.

A Neural Network (NN) typically processes large-scale problems in terms of dimensionality, amount of data handled, and the volume of simulation or neural hardware processing (Willow 2005). It emerged as an area of Artificial Intelligence (AI) (Russell *et al.* 2003) to mimic the human neurons in both perception and learning. It is interesting to note, however, that a conceivably disparate area within information science classified as 'knowledge representation' had brought the attention of researchers to pursue classes of 'computing and processing', such as neural networks. Object-oriented paradigm emerged as one of the better models for knowledge representation (Willow 1998a). In fact, the motivation for NN research was to seek an improved methodology in *machine learning*, and more specifically, in the area of *planning* algorithm, thereby augmenting the techniques available at the time. However, as the research progressed, more obstacles in emulating the human neurons were realized. Toward this end, the jargon NN at present, is more appropriate if it were to be replaced with *parallel, distributed simulation* (Willow, 1998b). Figure 2 illustrates taxonomic views of NN (Willow 2002), based on former studies by

Zurada (1992). Notice Figure 2 does not encompass an exhaustive list of available NN models to date.

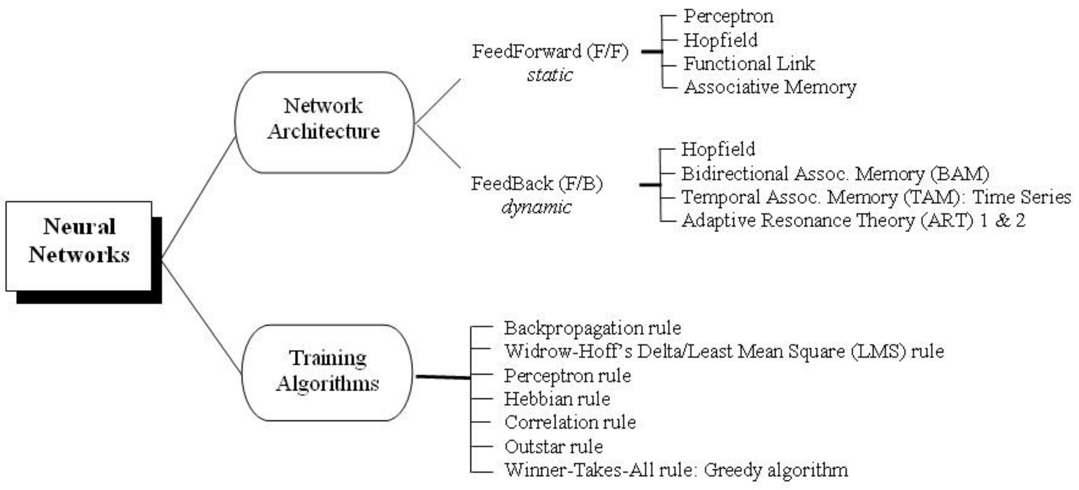


Figure 2. Classification of Neural Network Models

The concept of *feedback* plays a central role in learning for NN. Two different types of learning are to be distinguished: learning *with* supervision (*i.e.,* training) versus learning *without* supervision (Figure 3) (Willow 2002).

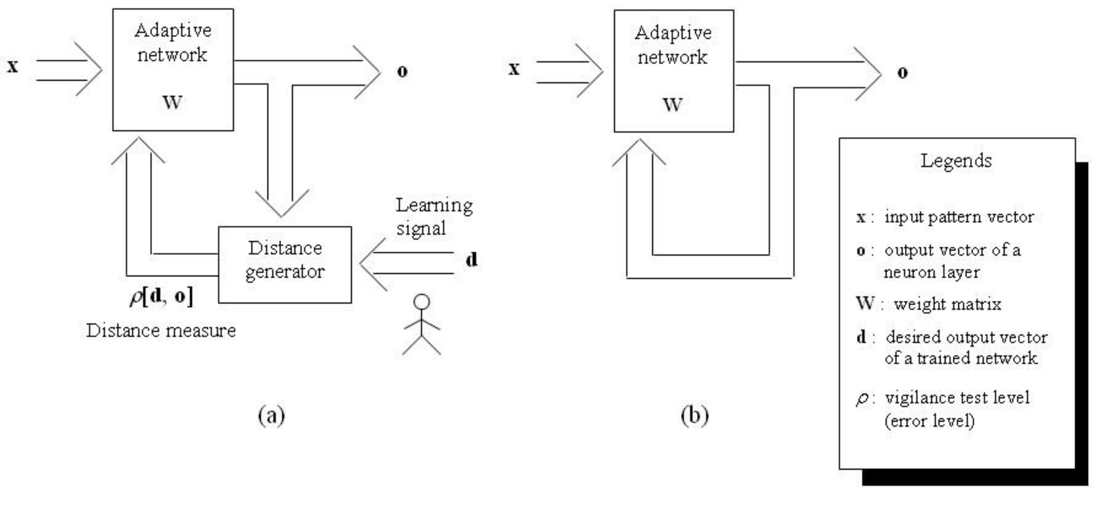


Figure 3. Learning Modes for Neural Networks:

(a) Supervised *versus* (b) Unsupervised

In *supervised* learning [Figure 3(a)], the desired response **d** of the system is provided by the teacher at each instant of time. The distance ρ[**d**, **o**] between the actual and the desired response serves as an error measure, and is used to correct network parameters externally.

Since adjustable weights are assumed, the teacher/supervisor may implement a reward-or-punishment scheme to adapt the network's weight matrix, **W**. This mode of learning is pervasive, and is used in many situations of natural learning. A set of input and output patterns called a *training set* is required for this learning mode. Often, the inputs, outputs, and the computed gradient are deterministic, however, the minimization of error proceeds over all its random realizations. As a result, most supervised learning algorithms reduce to stochastic minimization of error in multi-dimensional weight space (Willow 2005).

In learning *without* supervision [Figure 3(b)], the desired response (**d**) is not known; thus, explicit error information cannot be used to improve network behavior. Since no information is available as to correctness or incorrectness of response, learning must somehow be accomplished based on observations of responses to inputs of marginal or at times, no knowledge. Unsupervised learning algorithms use patterns that are typically redundant raw data having no labels regarding their class membership, or association. In this mode of learning, the network must discover for itself any possibly existing patterns, regularities, separating properties, *etc*. While discovering these, the network undergoes change of its parameters, which is called *self-organization*. Adaptive Resonance Theory (ART) is a good example of such a class. An excellent application of ART is suggested by Willow (2005, 2007c) for a classical email server management problem.

3. Linear Classifier Configuration with Perceptrons

A considerable number of Neural Network (NN) applications research to date have proposed highly complex mathematical models, and the authors have introduced various *ad hoc* algorithms inconsistent with their models. In effect, real-time response often is not achieved due to the complexity of the model itself. A single-layer perceptron with discriminant function suffices for identifying *clusters* of workgroups for information systems. A 'perceptron' is a single-layer feedforward NN with input and output layers and a threshold logic unit. Figure 4 follows to illustrate.

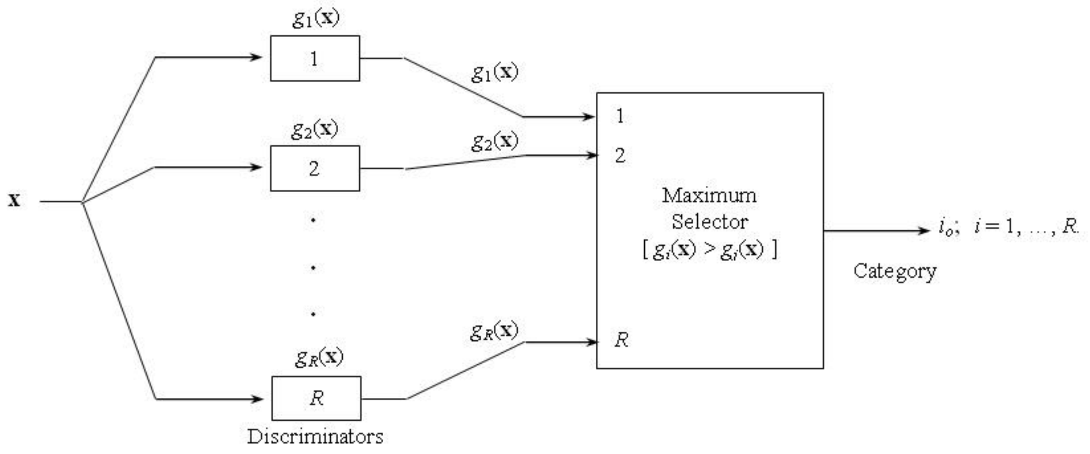


Figure 4. Cluster Classification with Single-Layer Perceptron

Here, R is the number of classes for the input pattern to be distinguished by using the discriminators. The cardinality for the input vector **x** is assumed to be n. A simplified example of a bi-class discriminator, better known as the dichotomizer (*i.e.*, R = 2) with an arity of 2 (n = 2) is illustrated in Figure 5.

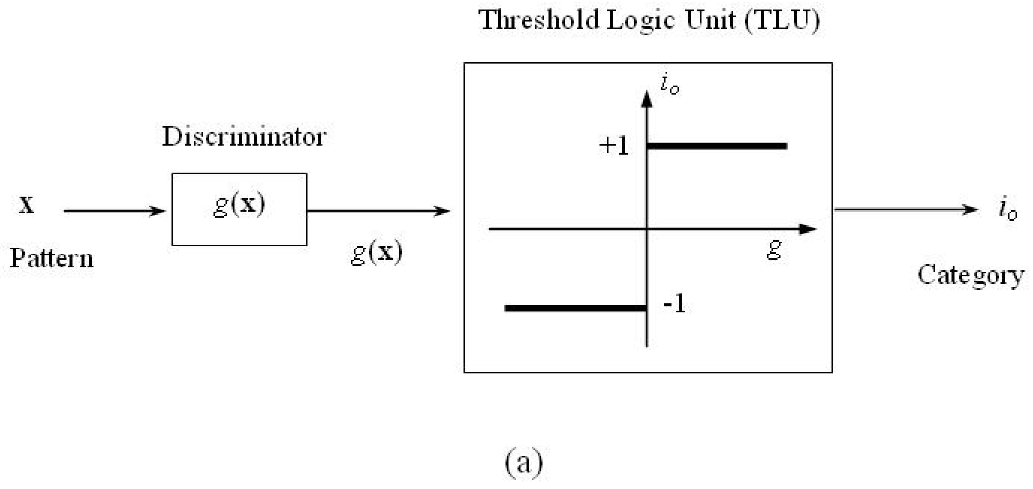


Figure 5. Dichotomizer in Single-Layer Perceptron:

(a) Number of Clusters, *R* = 2

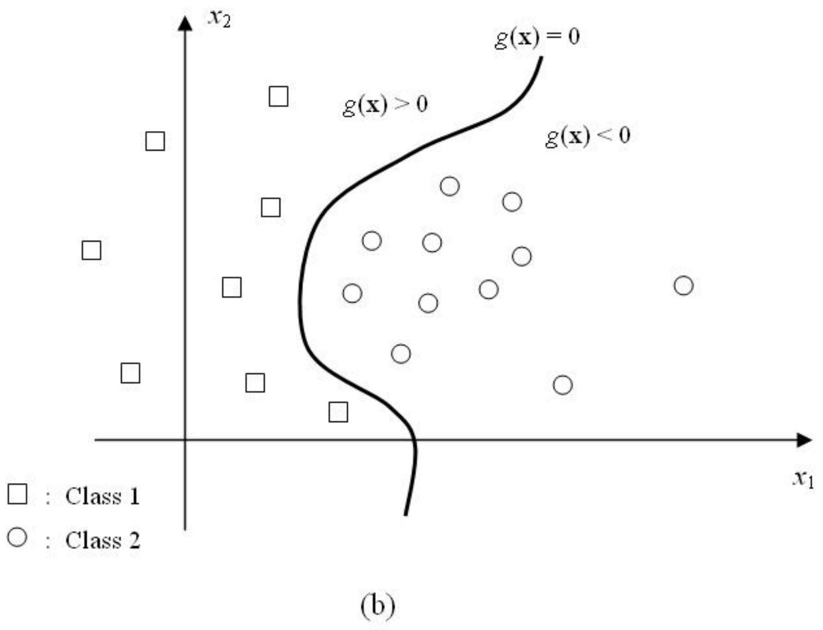


Figure 5. Dichotomizer in Single-Layer Perceptron *(cont'd)***:**

(b) Decision Surface for *n* = 2

It is precisely this nature of the feedforward perceptron aforementioned, by which the decision maker has complete flexibility to predetermine the desired number of clusters (*R*) prior to the execution of NN computing. This is expected to increase the overall productivity of building the information system infrastructure considerably, where major operations such as network design takes precedence over most others.

The discriminant function $g(\mathbf{x})$ forms a decision hyperplane in the E^2 space (*i.e., n* = 2), and is defined as

$$g(\mathbf{x}) \equiv g_1(\mathbf{x}) - g_2(\mathbf{x}) \tag{1}$$

for dichotomizers. Thus, the input pattern is ramified as class 1 if $g(\mathbf{x}) > 0$, and class 2, otherwise [*i.e.,* $g(\mathbf{x}) < 0$]. A key to classification with perceptrons is the determination of $g(\mathbf{x})$. A series of $g(\mathbf{x})$ can be simulated for the prescribed patterns in neural (*viz.* parallel) networks.

A single Threshold Logic Unit (TLU) with a *discrete* sigmoid function is realized for the dichotomizer, such that:

$$i_o = \operatorname{sgn}[g(\mathbf{x})] = \begin{Bmatrix} -1, & g(\mathbf{x}) < 0 \\ \textit{undefined}, & g(\mathbf{x}) = 0 \\ +1, & g(\mathbf{x}) > 0 \end{Bmatrix} \tag{2}$$

For *continuous* input vectors, the sigmoid or bipolar function becomes

$$\operatorname{sgn}[g(\mathbf{x})] = f(\text{net}) = \frac{2}{1+e^{-\lambda_{net}}} - 1, \tag{3}$$

where, λ is the gain or steepness factor for searching the optimum/optima. In Figure 6, f(net) is depicted with a range of λ values (Zurada 1992). Note that as $\lambda \to \infty$, the continuous function *converges* to the discrete sigmoidal function. The discrete version is almost always applicable for linear classification.

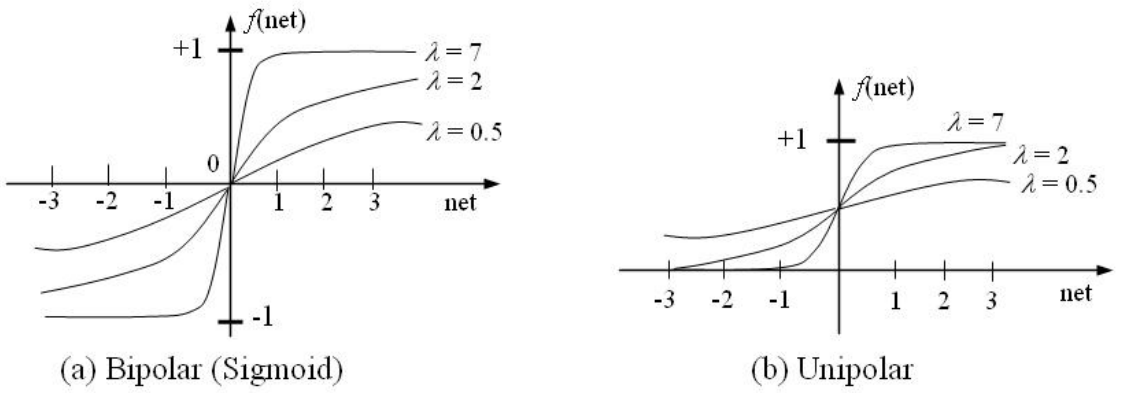


Figure 6. Continuous Activation Functions of a Neuron in TLU

An additional concept of classifier with *minimum-distance* follows in Figure 7. Notice the decision hyperplane is determined based on the following geometrical insights:

$g(\mathbf{x})$ should include the midpoint of the line segment $\overrightarrow{P_1P_2}$, given that vertices P_1 and P_2 represent the approximate center of gravity of class 1 and 2, respectively. In terms of *vectors*, $\mathbf{x_1}$ and $\mathbf{x_2}$ denote the center of gravity for each class.
$g(\mathbf{x})$ should be *normal* (*i.e.,* perpendicular) to the vector $\mathbf{x_1} - \mathbf{x_2}$.

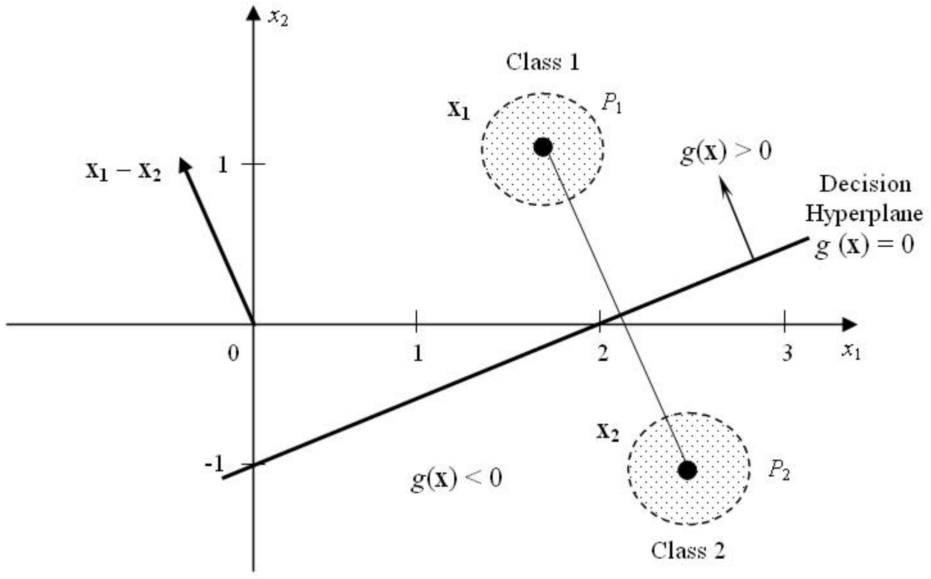


Figure 7. Minimum-Distance Classifier

The decision hyperplane can thus be expressed as the following.

$$g(\mathbf{x}) \;=\; (\mathbf{x_1} - \mathbf{x_2})^{\mathrm{T}}\mathbf{x} + \frac{1}{2}(\|\,\mathbf{x}_2\,\|^2 - \|\,\mathbf{x}_1\,\|^2) \;=\; 0 \tag{4}$$

It can also be seen that $g(\mathbf{x})$ implied here constitutes a hyperplane described by the equation

$$w_1x_1 + w_2x_2 + \ldots + w_nx_n + w_{n+1} = 0, \quad \text{or}$$
$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_{n+1} = 0 \tag{5}$$

or, briefly,

$$\begin{bmatrix} \mathbf{w} \\ w_{n+1} \end{bmatrix}^T \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = 0\,, \tag{6}$$

where, $\mathbf{w}$ is the $[n \times 1]$ weight (column) vector.

A widely accepted convention is to append formally a 1 (*cf.,* scale factor; unit normal vector) as the n+1th component of each pattern vector. The augmented pattern vector is now denoted by **y**, with n+1 rows, and is defined as

$$\mathbf{y} \equiv \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \tag{7}$$

The weighting coefficients w_1, w_2, …, w_{n+1} of the dichotomizer can now be obtained by comparing equations (4) and (5) as follows:

$$\mathbf{w} = \mathbf{x_1} - \mathbf{x_2}$$

$$w_{n+1} = \frac{1}{2}(\| \mathbf{x}_2 \|^2 - \| \mathbf{x}_1 \|^2) \tag{8}$$

The Euclidean distance between input pattern **x** and the prototype pattern vector $\mathbf{x}_i$ is expressed by the *norm* of the vector $\mathbf{x} - \mathbf{x}_i$ as

$$\| \mathbf{x} - \mathbf{x}_i \| = \sqrt{(\mathbf{x} - \mathbf{x}_i)^T (\mathbf{x} - \mathbf{x}_i)} \tag{9}$$

A minimum-distance classifier computes the distance from pattern **x** of unknown classification to each prototype. Then, the category number of that closest, or smallest distance, prototype is assigned to the unknown pattern. Calculating the squared distances from equation (9) yields

$$\| \mathbf{x} - \mathbf{x}_i \|^2 = \mathbf{x}^{\mathrm{T}}\mathbf{x} - 2\mathbf{x}_i^{\mathrm{T}}\mathbf{x} + \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_i, \qquad \text{for } i = 1, \ldots, R. \tag{10}$$

Hence, the discriminant function can now be summarized as

$$g(\mathbf{x}) = \mathbf{x}_i^{\mathrm{T}}\mathbf{x} - \frac{1}{2}\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_i, \qquad \text{for } i = 1, \ldots, R \tag{11}$$

$$= \mathbf{w}_i^{\mathrm{T}}\mathbf{x} + w_{i,\, n+1}, \qquad \text{for } i = 1, \ldots, R \tag{12}$$

The discriminant function coefficients that are weights $\mathbf{w}_i$ can thus be determined by letting

$$\mathbf{w}_i = \mathbf{x}_i \tag{13}$$

$$w_{i,\, n+1} = -\frac{1}{2}\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_i, \qquad \text{for } i = 1, \ldots, R \tag{14}$$

In summary and in essence, the decision surface, say S_{ij}, for the contiguous decision regions ξ_i and ξ_j are hyperplanes given by the equation

$$g_i(\mathbf{x}) - g_j(\mathbf{x}) = 0, \text{ or} \tag{15}$$

$$\mathbf{w}_i^{\mathrm{T}}\mathbf{x} + w_{i,\,n+1} - \mathbf{w}_j^{\mathrm{T}}\mathbf{x} - w_{j,\,n+1} = 0 \tag{16}$$

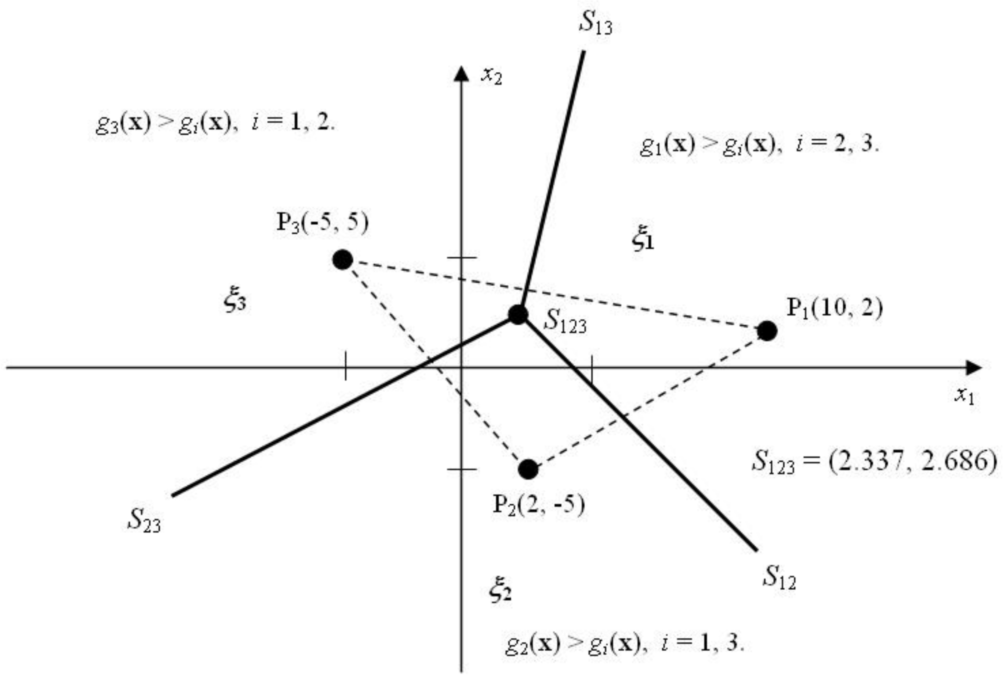
A numerical example follows to illustrate.

3.1. Numerical Example for Minimum-Distance Classifier

A linear classifier is designed in this example (n = 2, R = 3), in which the decision lines are generated using *a priori* knowledge on the center of gravity of the prototype points. Suppose the coordinates for the assumed prototype points are

$$\mathbf{x}_1 = \begin{bmatrix} 10 \\ 2 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 2 \\ -5 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} -5 \\ 5 \end{bmatrix},$$

as depicted in Figure 8.



S_{13}
x_2
$g_3(\mathbf{x}) > g_i(\mathbf{x}),\ i = 1, 2.$
$g_1(\mathbf{x}) > g_i(\mathbf{x}),\ i = 2, 3.$
P$_3$(-5, 5)
ξ_1
ξ_3
S_{123}
P$_1$(10, 2)
x_1
S_{123} = (2.337, 2.686)
S_{23}
P$_2$(2, -5)
S_{12}
ξ_2
$g_2(\mathbf{x}) > g_i(\mathbf{x}),\ i = 1, 3.$

Figure 8. Minimum-Distance Classifier Example

The *augmented* (*i.e.,* with $w_{i,\,n+1}$) weight vectors, based on equations (13) and (14) are

$$\mathbf{w}_i = \begin{bmatrix} \mathbf{x}_i \\ -\frac{1}{2}\mathbf{x}_i^T \mathbf{x}_i \end{bmatrix}; \qquad \mathbf{w}_1 = \begin{bmatrix} 10 \\ 2 \\ \hline -52 \end{bmatrix}, \quad \mathbf{w}_2 = \begin{bmatrix} 2 \\ -5 \\ \hline -14.5 \end{bmatrix}, \quad \mathbf{w}_3 = \begin{bmatrix} -5 \\ 5 \\ \hline -25 \end{bmatrix}$$

The corresponding linear discriminant functions are

$$\begin{aligned} g_1(\mathbf{x}) &= 10x_1 + 2x_2 - 52 \\ g_2(\mathbf{x}) &= 2x_1 - 5x_2 - 14.5 \\ g_3(\mathbf{x}) &= -5x_1 + 5x_2 - 25 \end{aligned}$$

Based on the computed discriminant functions, a minimum-distance classifier using the maximum selector can be completed for $R = 3$. Using equation (15), the decision lines can be calculated as follows.

$$\begin{aligned} S_{12}: g_1(\mathbf{x}) - g_2(\mathbf{x}) &= 8x_1 + 7x_2 - 37.5 = 0 \\ S_{13}: g_3(\mathbf{x}) - g_1(\mathbf{x}) &= -15x_1 + 3x_2 + 27 = 0 \\ S_{23}: g_3(\mathbf{x}) - g_2(\mathbf{x}) &= -7x_1 + 10x_2 - 10.5 = 0 \end{aligned}$$

The discriminant hyperplanes clearly indicate that there are indeed three linearly separable regions (ξ_1, ξ_2, and ξ_3), as shown in Figure 8.

When the desired output vector **d** is known *a priori*, training the network possibly through feedback may begin, as depicted in Figure 9. The notion of feedback NN was conceptualized in Figure 3(a).

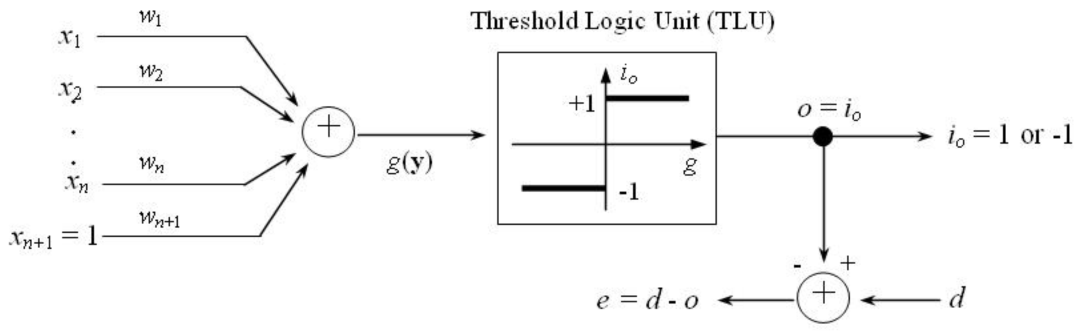


Figure 9. Linear Classifier with Training

In essence, perceptron models of NN, introduced in section 2, can make a significant improvement over the conventional methods for clustering based on Hungarian algorithm. Various extensions to Hungarian algorithm have many limitations, among which is the determination of the number of possible clusters, *R*. The decision maker has no control over *R*, which is *generated* by complete iterations of the algorithm. By contrast, perceptron allows the flexibility of selecting the *desired* number of clusters *a priori*, preferably selected by the decision maker.

4. Nonseparable Patterns

For the linearly *nonseparable* input pattern(s) **x**, there is a possibility of a network system client(s) such as a PC and workstations being classified under *arbitrary* workgroups. One objective of the Group Technology (GT) is to minimize such anomalies, which was not fully realized in previously introduced algorithms. To accommodate this problem, additional layer(s) to the feedforward networks is proposed in this section. Each layer is perceived as a '*set*'. Additional layers therefore, expedite further mappings to the other sets accordingly. Notice the object of GT is to generate mutually exclusive or *disjoint* clusters for all components under the network design.

An intermediate layer (of neurons) between the input and output layers is introduced, dedicated to pattern-to-image transformation (*i.e.,* filtering). In set-theoretic terms, a double mapping is in effect (Figure 10).

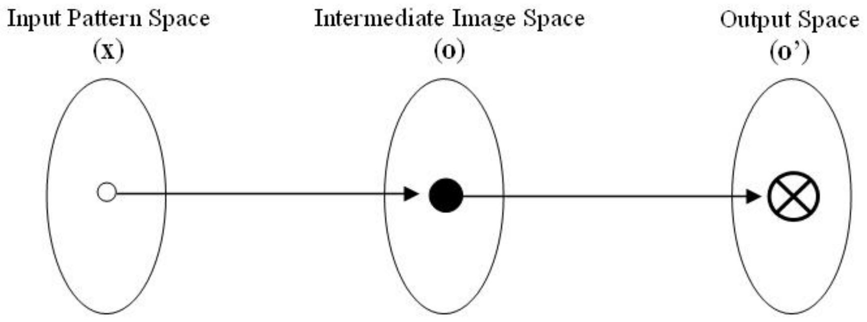
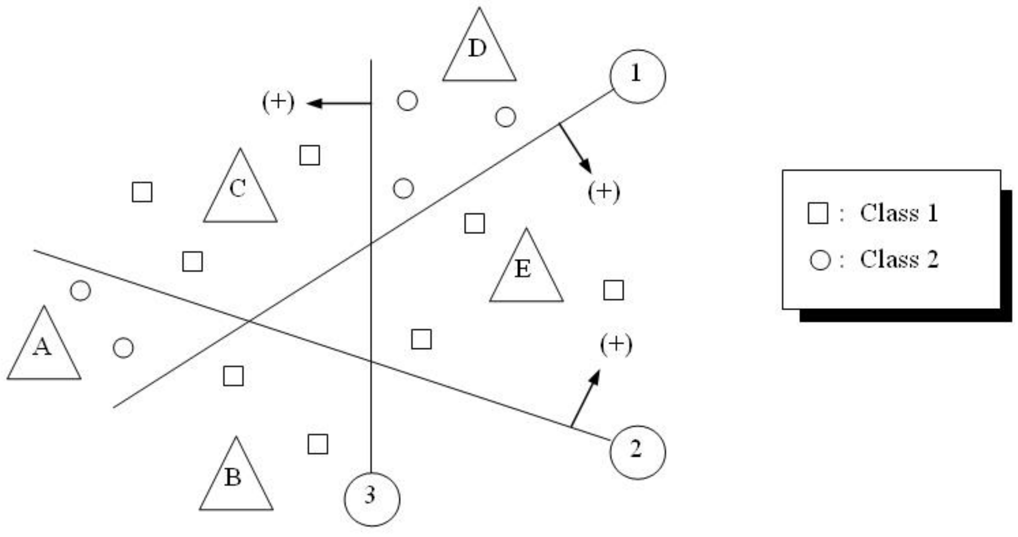


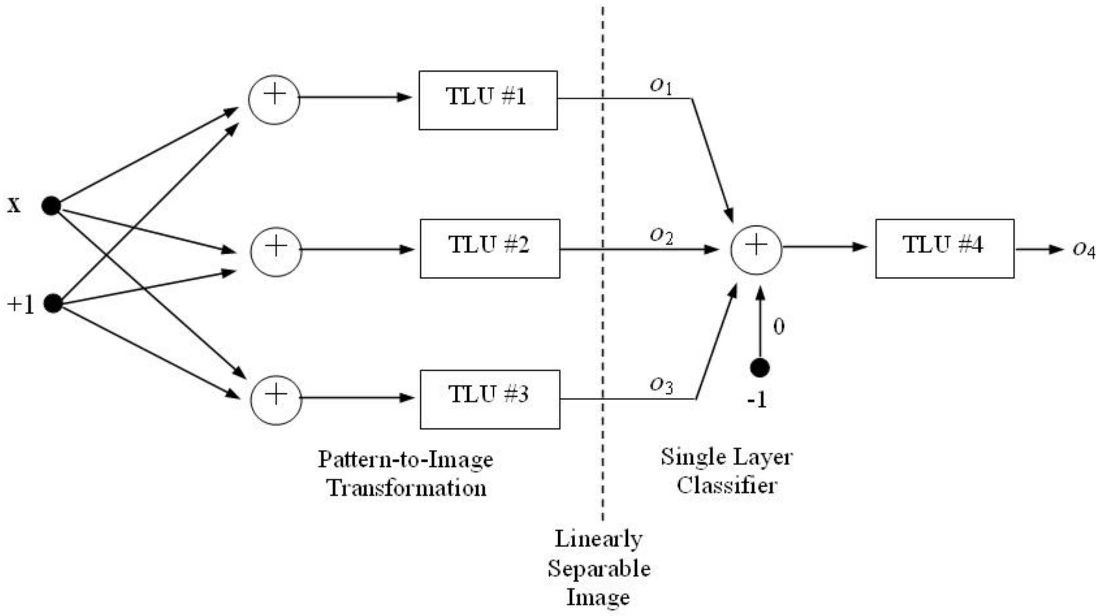
Figure 10. Set Mappings of Linearly Nonseparable Patterns

Suppose two sets of patterns $\xi 1$ and $\xi 2$ are to be classified into two categories. The example patterns are shown in Figure 11(a). In short, *R* = 2, but the patterns are linearly nonseparable. Three arbitrarily selected partitioning surfaces 1, 2, and 3 have been shown in the pattern space, *x*. The partitioning has been done in such a way that the pattern space now has compartments containing only patterns of a single category. Moreover, the partitioning surfaces are hyperplanes in pattern space E^n. The partitioning of Figure 11(a) is also nonredundant (*i.e.* implemented with minimum number of lines). It corresponds to mapping the *n*-dimensional original pattern space *x* into the three-dimensional image space, *o*.

Hence, each of the decision hyperplanes 1, 2, or 3 is implemented by a single *discrete* perceptron with suitable weights, and the transformation of the pattern space to the image space is performed by the network as displayed in Figure 11(b). Note only the first layer of discrete perceptrons responding with o_1, o_2, and o_3 is involved in the discussed space transformation.



(a) Partitioning in the Pattern Space



(b) Double-mapped Layered Network from part (a)

Figure 11. Classification of Linearly Nonseparable Patterns

Since the arrows point toward the *positive* side of the decision hyperplane in the pattern space, each of the seven compartments from Figure 11(a) may be mapped into one of the vertices of the [-1, 1] cube. As a consequence, the result of the mapping for the patterns from the figure is illustrated in Figure 12, with image space o_1, o_2, and o_3 corresponding compartment labels at corners.

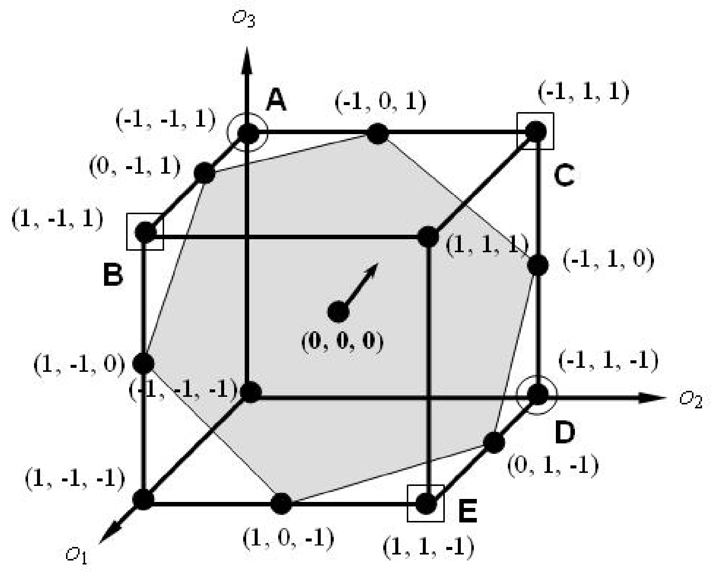


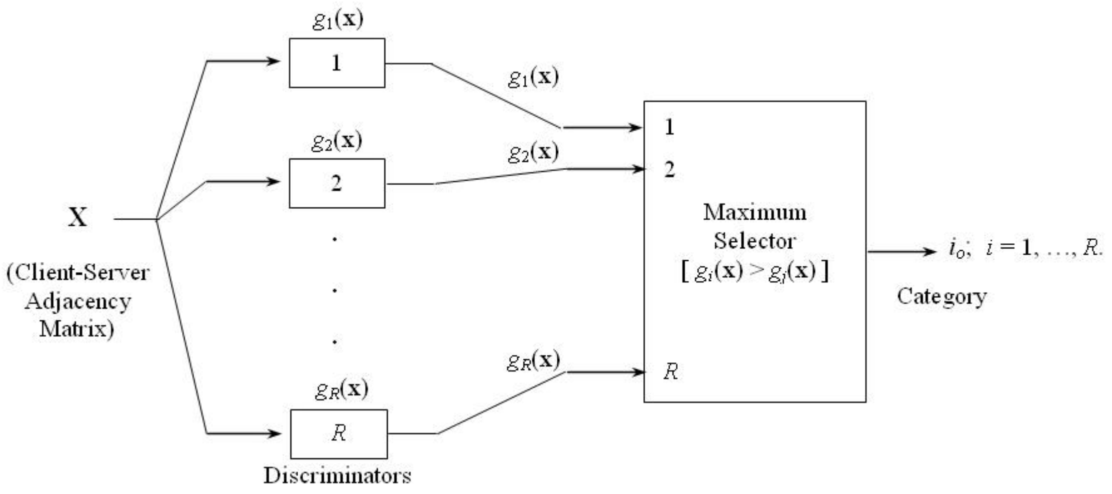
Figure 12. Classification in the Image Space by Intermediate Perceptrons

The patterns of class 1 from the original compartments *B*, *C*, and *E* are mapped into vertices (1, -1, 1), (-1, 1, 1), and (1, 1, -1), respectively. In turn, patterns of class 2 from compartments *A* and *D* are mapped into vertices (-1, -1, 1) and (-1, 1, -1), respectively. This shows that in image space *o*, the patterns of class 1 and 2 are easily separable by a plane arbitrarily selected, such as the one in Figure 12, having equation $o_1 + o_2 + o_3 = 0$. The single discrete perceptron in the output layer with inputs o_1, o_2, and o_3, zero bias, and the output o_4 is now able to provide the correct final mapping of patterns into classes as follows.

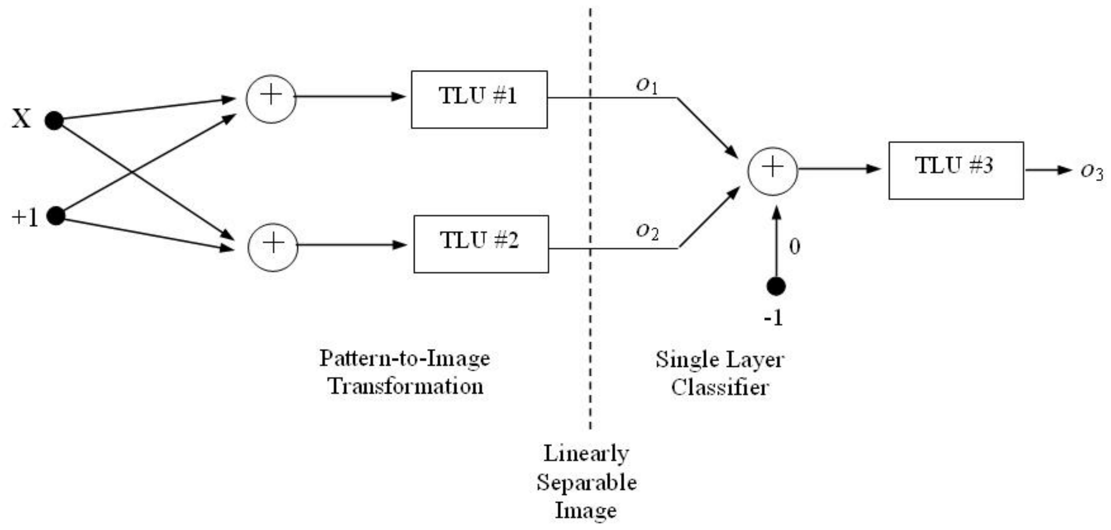
$$o_4 = \begin{Bmatrix} \operatorname{sgn}(o_1 + o_2 + o_3) > 0: \text{ class 1} \\ \operatorname{sgn}(o_1 + o_2 + o_3) < 0: \text{ class 2} \end{Bmatrix} \tag{17}$$

5. Neural Network Architecture for Linearly Separable Clustering

Thus far, insights to developing a new methodology in Group Technology (GT) have been investigated. A *neuro*-system encompassing a *single-layer perceptron* architecture is proposed for forming the network workgroups, with the flexibility of selecting the desired number of classes *a priori*. Upon completion of the classification, *linear separability* of the input patterns will be verified. Further, for linearly inseparable outputs, *re-classification* for the input pattern vectors will be sought with an augmented neuro-architecture. That is, a *multiply layered feedforward network* will be employed until complete linear separability is achieved. The cardinality of the client-server (adjacency) matrix in most network systems is of the second order (*i.e.,* two-dimensional; $n = 2$; flat-file structure). However, the suggested perceptron model could easily accommodate problems with higher order (*i.e.* recursion). For example, a complex network configuration problem might involve classifying its clients and servers into workgroup clusters, which are then required to form another set of clusters of floor plans, and in turn to encompass many buildings. Figure 13 follows to illustrate.



(a) Single-Layer Perceptron Classifier



(b) Double-Layer Feedforward Classifier ($n = 2$)

Figure 13. Proposed Framework for Workgroup Clusters of Network System

A simplified numerical example follows to illustrate the framework and method introduced in this paper.

5.1. Example: Neural Network Applications to Workgroup Determination

Given the following 'client-server matrix' in two dimensions (*i.e., n* = 2) (Ganz *et al.* 2004),

Table 1. Initial Client-Server Matrix

Client / Server	PC-1	PC-2	PC-3	PC-4	PC-5	PC-6	PC-7	PC-8	PC-9	PC-10
Server-01	X	X	X	X	X					
Image-01	X				X					
Scanner-01		X	X	X						
Printer-01	X				X					

Plotter-01		X	X							
Server-02						X	X	X	X	X
Image-02						X				X
Scanner-02		X			X	X	X	X	X	X
Printer-02							X		X	
Plotter-02								X		

it can be regenerated in two-dimensional *coordinates*, appropriate for processing in neural net as the input pattern vector. Numerical *proximity* or *acquaintance* values are determined for each axis/dimension, usually from the Information System/Information Technology (IS/IT) dimension system or decision makers. Below, a sample pattern vector for the given matrix is presented, based on utility measures.

Table 2. Input Pattern Vector

x_2 / x_1	2.1	2.4	2.8	4.1	4.2	5.1	7.2	9.3	14.1	14.2
2.3	$\mathbf{X}_1$	$\mathbf{X}_2$	$\mathbf{X}_3$	$\mathbf{X}_4$	$\mathbf{X}_5$					
1.1	$\mathbf{X}_6$				$\mathbf{X}_7$					
1.2		$\mathbf{X}_8$	$\mathbf{X}_9$	$\mathbf{X}_{10}$						
3.2	$\mathbf{X}_{11}$				$\mathbf{X}_{12}$					
3.4		$\mathbf{X}_{13}$	$\mathbf{X}_{14}$							
4.2						$\mathbf{X}_{15}$	$\mathbf{X}_{16}$	$\mathbf{X}_{17}$	$\mathbf{X}_{18}$	$\mathbf{X}_{19}$
4.3						$\mathbf{X}_{20}$				$\mathbf{X}_{21}$
5.1		$\mathbf{X}_{22}$			$\mathbf{X}_{23}$	$\mathbf{X}_{24}$	$\mathbf{X}_{25}$	$\mathbf{X}_{26}$	$\mathbf{X}_{27}$	$\mathbf{X}_{28}$
5.2							$\mathbf{X}_{29}$		$\mathbf{X}_{30}$	
5.3								$\mathbf{X}_{31}$		

Thus, the following pattern matrix (*i.e.,* 31 column vectors) is generated:

$$\underset{(2\times 31)}{\mathbf{X}} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_9, \mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{13}, \mathbf{x}_{14}, \mathbf{x}_{15}, \mathbf{x}_{16}, \mathbf{x}_{17}, \mathbf{x}_{18}, \mathbf{x}_{19}, \mathbf{x}_{20}, \mathbf{x}_{21}, \mathbf{x}_{22}, \mathbf{x}_{23}, \mathbf{x}_{24}, \mathbf{x}_{25}, \mathbf{x}_{26}, \mathbf{x}_{27}, \mathbf{x}_{28}, \mathbf{x}_{29}, \mathbf{x}_{30}, \mathbf{x}_{31}]$$

where,

$\mathbf{x}_1 = [2.3,\ 2.1]^T$, $\mathbf{x}_2 = [2.3,\ 2.4]^T$, $\mathbf{x}_3 = [2.3,\ 2.8]^T$,

$\mathbf{x}_4 = [2.3,\ 4.1]^T$, $\mathbf{x}_5 = [2.3,\ 4.2]^T$, $\mathbf{x}_6 = [1.1,\ 2.1]^T$,

$\mathbf{x}_7 = [1.1,\ 4.2]^T$, $\mathbf{x}_8 = [1.2,\ 2.4]^T$, $\mathbf{x}_9 = [1.2,\ 2.8]^T$,

$\mathbf{x}_{10} = [1.2,\ 4.1]^T$, $\mathbf{x}_{11} = [3.2,\ 2.1]^T$, $\mathbf{x}_{12} = [3.2,\ 4.2]^T$,

$\mathbf{x}_{13} = [3.4,\ 2.4]^T$, $\mathbf{x}_{14} = [3.4,\ 2.8]^T$, $\mathbf{x}_{15} = [4.2,\ 5.1]^T$,

$\mathbf{x}_{16} = [4.2,\ 7.2]^T$, $\mathbf{x}_{17} = [4.2,\ 9.3]^T$, $\mathbf{x}_{18} = [4.2,\ 14.1]^T$,

$\mathbf{x}_{19} = [4.2,\ 14.2]^T$, $\mathbf{x}_{20} = [4.3,\ 5.1]^T$, $\mathbf{x}_{21} = [4.3,\ 14.2]^T$,

$\mathbf{x}_{22} = [5.1,\ 2.4]^T$, $\mathbf{x}_{23} = [5.1,\ 4.2]^T$, $\mathbf{x}_{24} = [5.1,\ 5.1]^T$,

$\mathbf{x}_{25} = [5.1,\ 7.2]^T$, $\mathbf{x}_{26} = [5.1,\ 9.3]^T$, $\mathbf{x}_{27} = [5.1,\ 14.1]^T$,

$\mathbf{x}_{28} = [5.1,\ 14.2]^T$ $\mathbf{x}_{29} = [5.2,\ 7.2]^T$, $\mathbf{x}_{30} = [5.2,\ 14.1]^T$,

$\mathbf{x}_{31} = [5.3,\ 9.3]^T$

Assuming these patterns are to be classified into two new workgroups due to physical facility constraints (*i.e.* $R = 2$), a (single) discriminant function, $g(\mathbf{x})$ is generated. Note that the number of clusters, R, was selected by the decision maker such as the information systems administrators in advance.

The *minimum-distance classification* scheme is to be applied. In Figure 14, the pattern vectors are projected on the E^2 Cartesian plane, where the center of gravity for each class is estimated.

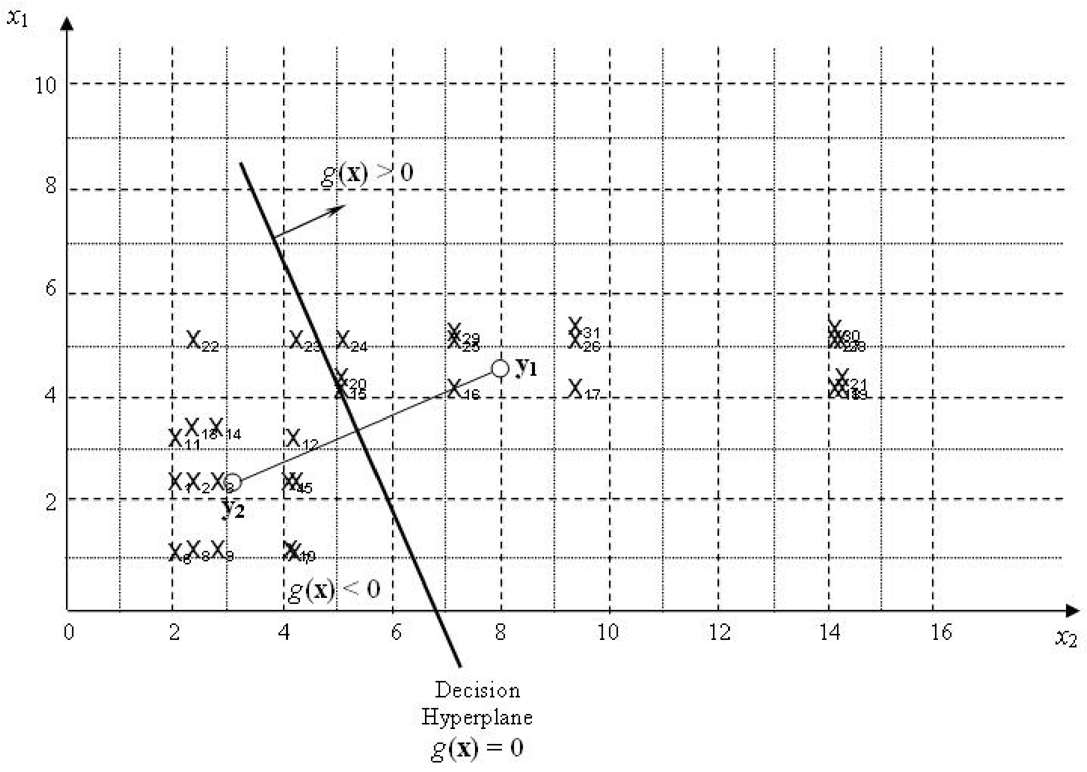


Figure 14. Minimum-Distance Classifier for the Example

$\mathbf{y_1}$ and $\mathbf{y_2}$ are the center of gravity of the two possible clusters, respectively:

$$\mathbf{y_1} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4.7 \\ 8 \end{bmatrix}, \qquad \mathbf{y_2} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2.3 \\ 3.1 \end{bmatrix}$$

Therefore, a discriminant function $g(\mathbf{x})$ that is *normal* to the $\mathbf{y_1}$ - $\mathbf{y_2}$ vector is generated as shown in Figure 14. For verification, the discriminant function in matrix form is

$$g(\mathbf{x}) = \begin{bmatrix} \mathbf{w} \\ w_{n+1} \end{bmatrix}^T \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = 0$$

$$= \begin{bmatrix} 2.4 \\ 4.9 \\ 35.59 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = 2.4x_1 + 2.9x_2 + 35.59 = 0$$

where,

$$\mathbf{w} \equiv [w_1, w_2, \ldots, w_n]^T = \mathbf{y_1} - \mathbf{y_2} = [2.4,\ 4.9]^T, \quad \text{and}$$

$$w_{n+1} = \frac{1}{2}(\| \mathbf{y}_2 \|^2 - \| \mathbf{y}_1 \|^2) = \frac{1}{2}\left[\left(\sqrt{2.3^2 + 3.1^2}\right)^2 - \left(\sqrt{4.7^2 + 8.0^2}\right)^2 \right] \cong 35.59$$

The following part families are thus formed:

$g(\mathbf{x}) > 0$: *Class 1* = {**X**$_{15}$, **X**$_{16}$, **X**$_{17}$, **X**$_{18}$, **X**$_{19}$, **X**$_{20}$, **X**$_{21}$, **X**$_{22}$, **X**$_{24}$, **X**$_{25}$, **X**$_{26}$, **X**$_{28}$, **X**$_{29}$, **X**$_{30}$, **X**$_{31}$}

$g(\mathbf{x}) < 0$: *Class 2* = {**X**$_{1}$, **X**$_{2}$, **X**$_{3}$, **X**$_{4}$, **X**$_{5}$, **X**$_{6}$, **X**$_{7}$, **X**$_{8}$, **X**$_{9}$, **X**$_{10}$, **X**$_{11}$, **X**$_{12}$, **X**$_{13}$, **X**$_{14}$, **X**$_{23}$, **X**$_{27}$}

In comparison, similar clusters of workgroups are generated by using one of the conventional techniques (*i.e.,* King's algorithm), as illustrated in Table 3 (Chang *et al.* 1991). Notice how Scanner-02 caused linear non-separability under the conventional scheme.

Table 3. Generation of Part Families with King's Algorithm

	PC-1	PC-4	PC-3	PC-5	PC-2	PC-6	PC-10	PC-7	PC-9	PC-8	W$_i$	Rank
Image-01	1			1							18	1
Printer-01	1			1							18	1
Plotter-01			1		1						40	3
Scanner-01		1	1		1						44	4
Server-01	1	1	1	1	1						62	5
Scanner-02				**1**	**1**	1	1	1	1	1	2032	10
Image-02						1	1				192	6
Printer-02								1	1		768	7
Plotter-02										1	1024	8
Server-02						1	1	1	1	1	1984	9
Scanner-02						**1**	**1**	**1**	**1**	**1**	2032	10
Weight	2	4	8	16	32	64	128	256	512	1024		

Should the patterns be linearly *nonseparable*, reconfiguring the network in a *multi-layer* feedforward architecture is necessary, as emphasized in section 4. Mathematical software capable of solving parallel network problems such as Mathematica (Wolfram 1991) and/or MatLab/SIMULINK (Donnelly 1995) may be useful.

6. Conclusions

This paper presented an alternative approach to generating workgroup clusters for information system networks. Generation of linearly separable patterns or clusters is equally important for information systems as well as for conventional business entities such as project planning and/or manufacturing. A single-layer peceptron neural net, along with multi-layer feedforward nets where applicable, have been selected as the major vehicle. Comparisons were made with inflexible Hugarian-based algorithms, associated with iterations of row and column exchanges.

By incorporating the suggested models of this paper into building 'intelligent software agents' (Willow 2005, 2006b, 2007a, 2007b, 2007c), a *comprehensive* Information Systems Management (ISM) may be achieved, allowing real-time network designs as well as frequent modifications to their topologies. ❑

References

Booch, G. (1991) *Object-Oriented Design with Applications,* Benjamin and Cummings Inc., New York.

Chang, T-C., Wysk, R. A., and Wang, H-P. (1991) *Computer-Aided Manufacturing,* Prentice-Hall Inc., New Jersey.

Donaghey, C. (1991) *BLOCPLAN Manual,* University of Houston Press, Houston.

Donaghey, C., and Pire, V. (1991) "Solving the Facility Layout Problem with BLOCPLAN," *in Factory Automation and Information Management*, CRC Inc., Boca Raton.

Donnelly, K. (1995) *MATLAB Manual: Computer Laboratory Exercises*, Harcourt College Publishers, Berkeley, CA.

Fichman, R, and Kemerer, C. (1992) Object-Oriented and Conventional Analysis and Design Methods – Comparison and Critique. *IEEE Transactions on Computers*, **22**, 22-39.

Ganz, A., Ganz, Z., and Wongthavarawat, K. (2004) *Multimedia Wireless Networks: Technologies, Standards and QoS*, Prentice-Hall, New York.

Groover, M. P. (1987) *Automation, Production Systems, and Computer-Integrated Manufacturing*, Prentice-Hall Inc., New Jersey.

Gunter, C. A., and Mitchell, J. C. (1994) *Theoretical Aspects of Object-Oriented Programming*, MIT Press, Cambridge, MA.

Ham, I., Hitomi, K., and Yoshida, T. (1985) *Group Technology: Applications to Production Management*, Kluwer Academic Inc., Boston.

Mueller, S., and Ogletree, T. W. (2004) *Upgrading and Repairing Networks, 4/E.*, Prentice-Hall, New Jersey.

Russell, S., and Norvig, P. (2003) *Artificial Intelligence: A Modern Approach, 2/E.,* Prentice-Hall, New Jersey.

Tseng, M. M., and Jiao, J. (1997) A Framework of Design for Mass Customization. *Proceedings of the 2nd Annual International Conference on Industrial Engineering Applications and Practice*, San Diego, California, U.S.A., 85-90.

Voss, G. (1991) *Object-Oriented Programming: An Introduction*, McGraw-Hill Inc., New York.

Willow, C. C. (1998a) An Object-Oriented Database for Information Management in Computer Integrated Manufacturing. *ACM Transactions on Information Systems*, **34**, 12-39.

Willow, C. C. (1998b) Neuro-Fuzzy Systems in Engineering Design. *Decision Sciences Report*, Department of Decision and Information Sciences, University of Houston, U.S.A.

Willow, C. C. (2002) A Feedforward Multi-layer Neural Network for Cell Formation in Computer Integrated Manufacturing. *Journal of Intelligent Manufacturing*, **13**, 75-87.

Willow, C. C. (2005) A Neural Network-Based Agent Framework for Mail Server Management. *International Journal of Intelligent Information Technologies*, **1** (4), 36-52.

Willow, C. C. (2006a) Qualitative Decision Making with Integrated Systems Design Methodology. *Journal of Engineering and Technology Management*, Accepted for Publication and In Print [JET-M MS#2005-03-3738].

Willow, C. C. (2006b) A Conceptual Framework for Agent-based Information Resource Management. *Proceedings of the 2006 International Conference on Embedded and Ubiquitous Computing: Springer Lecture Notes in Computer Science 4096* (Eds: Sha, Edwin; Han, Sung-K.; Xu, Cheng-Z.; Kim, Moon H.; Yang, Laurence T.; and Xiao, B.), Seoul, South Korea, Springer and IEEE Computer Society, 4096, 171-181, August 1-4, 2006.

Willow, C. C. (2007a) Neural Network-based Multiple Agent System for Web Server Management. MU WP No. W06-01, Management Information Systems, School of Business Administration, Monmouth University, NJ.

Willow, C. C. (2007b) Neural Network-based Multiple Agent System for Application Server Management. MU WP No. W06-02, Management Information Systems, School of Business Administration, Monmouth University, NJ.

Willow, C. C. (2007c) Mail Server Management with Intelligent Agents. *Advances in Intelligent Information Technologies*, **1**, 313-335, Idea Group Publication.

Willow, C. C.., and Ro, I-K. (1991) Integrated Approaches to Loading and Dispatching Problems in Flexible Manufacturing Systems. *International Journal of Production Research*, **2**, 112-137.

Willow, C. C.., and Mayer, R. J. (1996) A Relational Approach to Managing the Data in Computer Integrated Manufacturing. *Productions and Operations Management*, **2**, 102-139.

Willow, C. C., and Yang, T. (1997) An Object-Oriented Simulation for Determining the Material Handling System Capacity in Computer Integrated Manufacturing. *Proceedings of the 1st International Conference on Industrial Engineering Practices*, Houston, Texas, October, 1997.

Wolfram, S. (1991) *Mathematica: A System for Doing Mathematics by Computer*, 2nd Edition, Addison-Wesley, Boston.

Zurada, J. M. (1992) *Introduction to Artificial Neural Systems*, West Inc., California.