May 8, 2007 9:55 AM

Ship to:
P.V. Balakrishnan

18115 CAMPUS WAY NE
BOTHELL, WA 98011

Campus: Bothell
User Name: sundar

Phone: 425 352-5384
Email: sundar@u.washington.edu

Reference No:

Standard / UW: Bothell

Need by: 08/05/2007

Maxcost: Free

Journal Title: Psychometrika

Volume: 59 **Issue:** 4 **Month/Year:** 1994 **Pages:** pp. 509#525

Article: Balakrishnan, P.V. (Sundar), Cooper, M.C., Jacob, V.S. and Lewis, P.A., A study of the classification capabilities of neural networks using unsupervised learning: A comparison with K-means clustering

Notes: English Only!

Ver:

UW-ILL #: DD267447

VOLUME 59
NUMBER 4
DECEMBER 1994

ISSN 0033-3123

Psychometrika

A JOURNAL OF QUANTITATIVE PSYCHOLOGY

THE PSYCHOMETRIC SOCIETY

Psychometrika

CONTENTS

VOLUME FIFTY-NINE **DECEMBER 1994** **NUMBER 4**

PSYCHOMETRIKA—VOL. 59, NO. 4, 509–525
DECEMBER 1994

A STUDY OF THE CLASSIFICATION CAPABILITIES OF NEURAL NETWORKS USING UNSUPERVISED LEARNING: A COMPARISON WITH *K*-MEANS CLUSTERING

P. V. (SUNDAR) BALAKRISHNAN

BUSINESS ADMINISTRATION PROGRAM
UNIVERSITY OF WASHINGTON, BOTHELL

MARTHA C. COOPER

DEPARTMENT OF MARKETING

VARGHESE S. JACOB

DEPARTMENT OF ACCOUNTING AND MIS
THE OHIO STATE UNIVERSITY

PHILLIP A. LEWIS

DEPARTMENT OF MARKETING
ROWAN COLLEGE OF NEW JERSEY

Several neural networks have been proposed in the general literature for pattern recognition and clustering, but little empirical comparison with traditional methods has been done. The results reported here compare neural networks using Kohonen learning with a traditional clustering method (*K*-means) in an experimental design using simulated data with known cluster solutions. Two types of neural networks were examined, both of which used unsupervised learning to perform the clustering. One used Kohonen learning with a conscience and the other used Kohonen learning without a conscience mechanism. The performance of these nets was examined with respect to changes in the number of attributes, the number of clusters, and the amount of error in the data. Generally, the *K*-means procedure had fewer points misclassified while the classification accuracy of neural networks worsened as the number of clusters in the data increased from two to five.

Key words: *K*-means, Kohonen, Monte Carlo simulation, neural networks, nonhierarchical clustering.

Introduction

Although the basic concept of neural networks (NNs) was envisioned as far back as the 1950's, the theory and application of the technique has only recently come into the forefront. Neural networks are particularly interesting because of their inherent claim to analyze noisy data and to deal with problems that have no clear cut solution, in addition to their critical differentiation from other techniques, namely their ability to learn.

Researchers are developing neural network models for various purposes, including managerial problems such as handling census data (Openshaw & Wymer, 1991), forecasting (Murtagh, 1991a), time-varying data (Rowher, 1991), and the traveling salesman

Requests for reprints should be sent to Martha Cooper, College of Business, The Ohio State University, 1775 College Road, Columbus, OH 43210-1399. Acknowledgements: Sara Dickson, Vidya Nair, and Beth Means assisted with the neural network analyses.

0033-3123/94/1200-93087$00.75/0

problem (Fort, 1991). Lippmann (1987) provides a basic overview of neural networks directed at pattern recognition. Neural networks in the general area of grouping or clustering have also been proposed (Ahalt, Krishnamurthy, Chen, Melton, 1990; Gath & Geva, 1988; Krishnamurthy, Ahalt, Melton, Chen, 1990; Osipenko, 1988; Wan, Wong, & Prusinkiewicz, 1988).

One author (Bezdek, 1991) has likened the level of activity in this field to a "feeding frenzy". Literally hundreds of articles have been written (see, for example, the *Journal of Neural Networks* or the *Proceedings of the International Conference on Neural Networks*) but little comparison with traditional techniques is available. Further, different applications of the same basic technique may produce different results leading to a lack of stability in the cluster solutions. For example, Balakrishnan, Cooper, Jacob, & Lewis (1992) obtained different cluster solutions using the Bucklin and Srinivasan (1991) coffee data on three different neural networks, all of which differed from the solutions obtained by using K-means clustering. The cluster sizes and their underlying characteristics changed as the number of training iterations changed. Similarly, even the K-means results differed depending on whether the algorithm was permitted multiple iterations and whether cluster seeds were input.

As mentioned earlier, increasingly, newer technologies, such as neural networks, are being considered as tools in a researcher's kit. However, before this shift away from the employment of more traditional statistical techniques to the newer technology gets underway in earnest, it is imperative that researchers demonstrate clearly the potential advantages of neural networks in dealing with clustering problems. In other words, one needs to document the specific advantages of neural networks as well as to identify the conditions under which they may be superior to traditional techniques. In this regard, Bezdek (1991) indicates that new algorithms need to be tested for such characteristics as complexity, convergence, stability, robustness, and performance validation.

Lack of comparative analysis is not unique to the neural network field. A similar situation occurred in the clustering area with many algorithms and stopping rules proposed without comparison with each other. Users had little guidance prior to the work of Milligan and Cooper (1985) and others (e.g., Dubes, 1987) about the potential effectiveness of thirty stopping rules for determining the number of clusters in a data set. Some work comparing neural networks with conventional methods has already begun for some algorithms (e.g., Bezdek, 1990; Bezdek & Hathaway, 1990; Srivastava & Murty, 1990.) Bezdek and Bobrowski (1990) and others have been evaluating fuzzy clustering algorithms, which can overlap with some neural network models.

In this paper, we specifically compare the nonhierarchical clustering capabilities of a class of neural networks using Kohonen learning with a K-means clustering procedure. That is, the focus of this study is on the ability of the procedures to correctly recover the known cluster structure in the data. Performance validation studies, in terms of the number of misclassified points, are conducted on data generated by a method used in the Milligan and Cooper (1985) study on clustering stopping rules. The remainder of the paper is organized as follows. The next section is an introduction to neural networks and their features as employed in this study. The methodology for this study is then outlined, followed by the data analysis. The final section contains the conclusions and limitations of this study, along with directions for future research.

Neural Networks

The concept of neural networks (NNs) originated with work on mammalian brains with the goal of designing systems to replicate their functioning. A neural network consists of two basic components: *processing elements* (which can have a local mem-

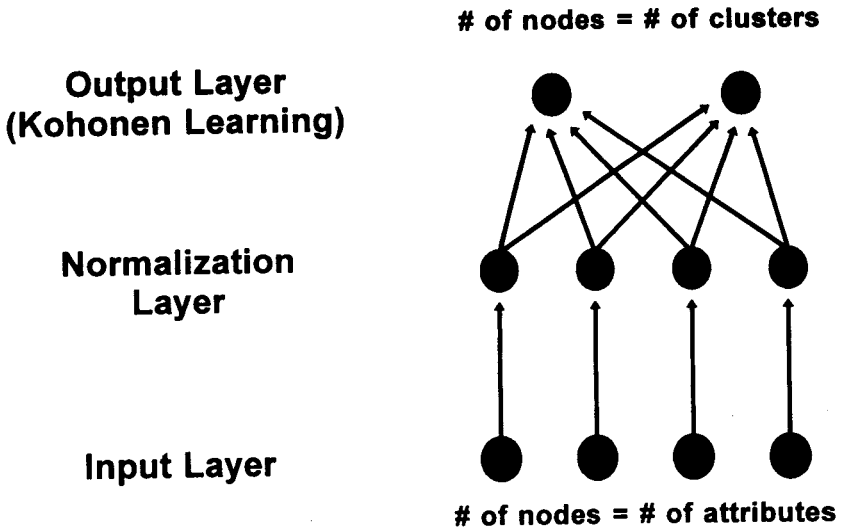

FIGURE 1.
Sample Net.

ory and perform localized information processing activities) and their *interconnections*. The processing element in a neural network can be considered akin to a neuron in a mammalian brain. The processing element receives a number of input signals and either generates a single output signal or does not generate any output signal (i.e., the output signal value is zero). This signal is then "sent" to one or more processing elements via the interconnections. Within the network each processing element corresponds to a node. For example, in Figure 1 there are ten processing elements or nodes.

Unlike conventional processing using a von-Neumann computer, the NN programmer does not specify an algorithm for the system to execute, instead the "programmer" specifies the structure of the neural network (Caudill, 1987). This is accomplished by defining the interconnection architecture between the processing elements, the rules that determine whether or not a processing element will fire (i.e., generate a signal) and the rules governing changes in the relative importance of the individual interconnections to a processing element's training data.

A typical neural network is organized into layers. At the lowest level, there is the input layer that contains the nodes through which the data are input into the network and at the top there is the output layer that generates the output interpreted by the user. Between the input layer and the output layer, there could be one or more layers, which are called the hidden layers. The output of the input layer is fed to the first hidden layer in the network and forms the input signals to this hidden layer. The output from the first hidden layer is then fed to the next and so on, until the signals reach the output layer which in turn generates a signal interpreted by the user.

In addition to the actual structure of the networks, (i.e., the number of layers and the interconnections), several other factors need to be specified before a neural network can be utilized. Since several sources discuss the concept of neural networks in detail (see for example, Hecht-Nielsen, 1990; Nelson & Illingworth, 1991; Rumelhart, McClelland, and the PDP Research Group, 1987), here we will only briefly discuss the generic neural network issues.

The first step in developing a neural network framework is specifying what the input layer nodes represent. For example, in the traveling salesman problem, these nodes represent cities. In the case of word identification, they could be letters. In market segmentation, they could be lifestyle attributes or demographic characteristics of individuals. Within the context of clustering, the input layer nodes represent the

attributes associated with the data and each node in the output layer corresponds to a cluster.

How the nodes are connected to each other constitutes the system's knowledge and determines how the neural network will respond to any arbitrary input. A connection between two nodes could be excitatory, in which case the connection between the nodes have positive weights; or they could be inhibitory, in which case the connection between the two nodes have negative weights. Thus, the weight w_{ij} is positive if node j excites node i and negative if j inhibits i. The strength of the connection is given by the absolute value of the weight w_{ij}. The pattern of connectivity can be represented by a weight matrix W in which w_{ij} represents the strength and sense (i.e., excitatory or inhibitory), of the connection between j and i.

The nodes in each layer need a framework to process the information. Each node takes in inputs and generates outputs. Except for the input layer, generally each node in a layer takes the output of the nodes to which it is connected from the previous layer and combines it with the weights of the connections, typically as the weighted sum of the outputs, to generate an input value for the node. If o_j represents the output of the j-th node connected to node i, then input $I_i = \sum_j w_{ij} o_j$, where the summation is over all nodes connected to node i from the previous layer. This combination of inputs to a node is referred to as the summation function and if it is a weighted sum, the summation function is called the "sum".

The input I_i is then transformed by the node i using a transform function that is usually nonlinear. A linear transform function within the Neuralworks Professional II package (Neuralware, 1991) is just the sum itself. The result of the transformation forms the output, o_i, of the node i. Whether this output o_i forms the input to a node k in the next layer depends on whether there is competition between the other nodes in the layer to which node i belongs. If there is competition within a layer, only the node generating the highest output value as a result of the transformation will generate an output to the next layer, all other nodes will generate zero as the output. If there is no competition between the nodes, all the nodes would generate an output and as discussed earlier, the outputs would be combined to form the input to the nodes in the next layer.

Another important factor in the design of neural networks is the learning rule used by the various layers in the network. There are two broad categories of learning: supervised and unsupervised. In supervised learning the data used for learning purposes has information about the "correct" output associated with each training or learning input data item. For the purposes of clustering-based research, we focused on unsupervised learning as the researchers are not aware of the correct classification of each data item. More specifically, we used Kohonen's learning rule and a variation on the theme implemented as the self-organizing-map in Neuralworks Professional II. A detailed discussion of the Kohonen approach can be found in Kohonen (1982a, 1982b; 1989). We now discuss the salient features of the unsupervised learning approach.

Unsupervised Learning

In unsupervised learning, the data used for learning purposes does not contain information on the "correct" output. In the current study, the cluster membership, for a given input vector is not provided in the training sample. The network, therefore, needs to learn on its own to classify each of the input vectors into the various clusters.

One of the popular learning approaches to unsupervised learning is the learning law ascribed to Kohonen and called the Kohonen Learning Law. This approach is more consistent with traditional clustering techniques. Although several researchers have

been associated with the law, Kohonen's work seems to have focused the research in the area leading to his name being associated with the work (Hecht-Nielsen, 1990).

If in a neural net there is a layer in which the nodes utilize Kohonen-learning, then the layer is typically called a Kohonen layer. Suppose there are n nodes in the Kohonen layer and each node receives m inputs. The m inputs would correspond to the m attributes of each data point in the data set. Associated with each input would be a weight. When the input vector enters the Kohonen layer, the processing elements compete to produce an output. The winner is determined based on some distance metric/function between the weight vector of the node and the input vector. The node having the closest weight vector to the input vector generates an output $o_i = 1$ and the others produce an output of zero. The node that wins incrementally modifies its weights using some learning scheme while the weights associated with the other nodes stay at their original values.

Formally, suppose the weight vector associated with node i is denoted as $\mathbf{w}_i = (w_{i1}, w_{i2}, \ldots, w_{im})^T$ and the input vector as $\mathbf{x} = (x_1, x_2, \ldots, x_m)^T$, where w_{ij} indicates the weight assigned to input x_j going into the node i. Each element in the Kohonen layer computes the distance between its weight vector and the input signal, thus $d_i = D(\mathbf{x}, \mathbf{w}_i)$, where the distance measurement function D is typically either the standard Euclidian or the spherical arc distance measurement. (The Euclidian distance between two vectors $\mathbf{x_1} = (x_{11}, \ldots, x_{1n})^T$ and $\mathbf{x_2} = (x_{21}, \ldots, x_{2n})^T$ is given by $(\sum_j((x_{1j} - x_{2j})^2)^{1/2}$ while the spherical arc distance is given by $1 - \mathbf{x_1} \cdot \mathbf{x_2}$ which is equal to $1 - \cos(\theta)$ when $\mathbf{x_1}$ and $\mathbf{x_2}$ are normalized to unit length.)

The node with the smallest d_i value wins the competition between the nodes (in that layer). In case of a tie, since the nodes are numbered from left to right, the node with the lowest processing element index number wins. Learning then takes place based on the following weight modification principle or the Kohonen learning law: $\mathbf{w_i}^{new} = \mathbf{w_i}^{old} + \alpha D(\mathbf{x}, \mathbf{w_i}^{old}) o_i$, where α is a constant, and $0 < \alpha \leq 1$. Since o_i can take on only zero or one values, the winning processing element adjusts its weight and the losing processing elements retains its old weight: $\mathbf{w_i}^{new} = \mathbf{w_i}^{old}$. As a result of learning the weight vector moves a fraction of the way to align itself with the input vector $\mathbf{x}$.

Using a neural net, therefore, is a two stage process. In the first stage the network is *trained* to recognize the data. It is during this stage that the system uses the learning rule. Once the network is trained, the second stage called the *recall* stage, involves using the trained network to classify the data points. Classification in the recall stage occurs by determining as before the distance between each of the input vectors and the weight vectors associated with each output node. Each input vector then is assigned to that node for which the distance between its weight vector and the input vector is the minimum.

Network Architecture

Our analysis primarily focused on two types of networks, both of which were created using the commercially available software, Neuralworks Professional II. The first network (see Figure 1) had three layers: an input layer with the number of input nodes equal to the number of attributes of the data set; a normalization layer, which normalizes the input vector; and an output layer which has the same number of nodes as the number of clusters in the data set. The output layer is the classification layer and uses the Kohonen learning rule during the learning process (Table 1). This type of network for ease of exposition will be referred to as a Type 1 network.

The second type of network (see Figure 2) has an input layer as in the first network.

TABLE 1

Network Characteristics

Layer	Characteristic	Value	
		Type 1	Type 2
Layer 1	Layer Function	Input	Input
	Summation Function	Sum	Sum
	Transfer Function	Linear	Linear
	Output Function	Direct	Direct
	Learning Rule	None	None
Layer 2	Layer Function	Normalization	SOM
	Summation Function	Norm-Mult	SOM
	Transfer Function	Linear	Linear
	Output Function	Direct	SOM
	Learning Rule	None	SOM
Layer 3	Layer Function	Output	Output
	Summation Function	Sum	Sum
	Transfer Function	Direct	Direct
	Output Function	One Highest	One Highest
	Learning Rule	Kohonen	None

However, here the hidden layer uses the self organizing map (SOM) learning rule as implemented in the Neuralworks Professional II package. This rule uses a conscience mechanism to prevent a particular node from assuming responsibility for the entire data set (DeSieno, 1988). In this approach, the conscience mechanism ensures that, along with the winning node, nodes around it within a prespecified neighborhood also update their weights. The neighborhood for the current study was set as one node on either side of the winning node. In addition, there is an output layer that is responsible for the classification. Each node in the output layer is connected to a specific region in the

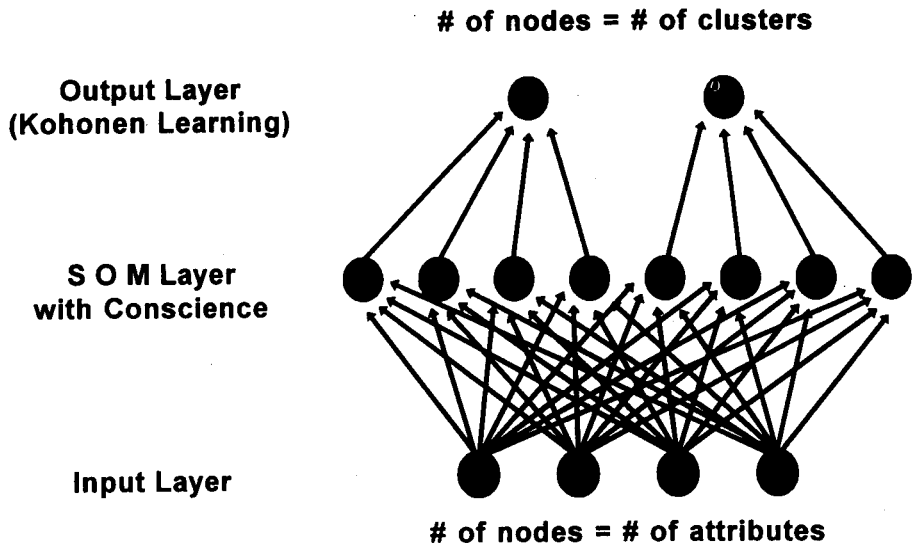

FIGURE 2.
Single Layer Self Organizing Map with Conscience.

hidden layer (Figure 2). This SOM-layer network is labelled Type 2 to distinguish it from the earlier network.

An Example

We now briefly illustrate the process of learning for the Type 2 network. Consider a data set consisting of fifty points (i.e., individuals or objects), each of which is described along four attributes. Our objective then is to classify each one of the points into one of two clusters. To this end, we therefore, create a net with four nodes in the input layer and two nodes in the output layer. The SOM layer is built with eight nodes. As shown in Figure 2, each node in the output layer is uniquely assigned to four nodes in the SOM layer.

The first step in the process is to *train* the network. The initial weights (randomly assigned) associated with the SOM layer are shown in Table 2. The system randomly picks a data point from the data set, in this case the data point is $\mathbf{x} = (50.22, 50.55, 32.3, 31.44)$, which represents the four attributes ($m = 4$). The network then computes the Euclidean distance $D_i = ((x_1 - w_{i1})^2 + (x_2 - w_{i2})^2 + (x_3 - w_{i3})^2 + (x_4 - w_{i4})^2)^{1/2}$, $i = 5, \ldots, 12$, for each of the eight nodes in the SOM layer (Neuralware, 1991). Since we have a conscience mechanism, the node with the minimum adjusted distance wins during learning, where the adjusted distance, $D_i' = D_i - B_i$. The bias, $B_i = \gamma(1/N - F_i)$, where N is the number of processing elements in the SOM layer (eight in this case) and F_i is the frequency with which the processing element i has historically won. Here the value of γ was set at 2.0 for the first 1000 iterations and then decreased to 1.5 for the next 1000 iterations and finally to 1.0 for the last 1000 iterations. At the beginning of the learning process $F_i = 1/N$, so $B_i = 0$. Table 2 depicts the result of the first data point passing through the network. Here the winner is node 12, and since it is the first iteration, the distance and adjusted distance are the same. Also note that since the system updates the nodes in the neighborhood of the winning processing element, the weights of node 11 are also updated, while all the other processing elements retain their original weights.

Now for the second iteration, the frequencies are modified based on the following relationship, for the winning node $F_{i\,\text{new}} = F_{i\,\text{old}} + \beta\,(1.0 - F_{i\,\text{old}})$ and for all the other nodes in the layer, $F_{i\,\text{new}} = F_{i\,\text{old}} + (0.0 - F_{i\,\text{old}})$. The value of β was decreased

TABLE 2

Example - Weight Changes During Learning

Iteration	[1]IN	Input Value	Weights of SOM layer nodes							
			5	6	7	8	9	10	11	12
Iteration 0	1	0	-.08	-.06	-.02	-.03	.03	.07	-.03	.01
	2	0	.1	.002	.02	.03	.03	-.01	.03	.08
[2]IW	3	0	.05	-.03	.03	-.05	.03	-.08	-.01	.05
	4	0	.08	-.06	.05	.02	-.06	-.06	.08	.01
Iteration 1	1	50.22	-.08	-.06	-.02	-.03	.03	.07	10.02	10.05
	2	50.55	.1	.002	.02	.03	.03	-.01	10.13	10.17
	3	32.3	.05	-.03	.03	-.05	.03	-.08	6.45	6.5
	4	31.44	.08	-.06	.05	.02	-.06	-.06	6.35	6.29
[3]Dist.	----	-------	84.26	84.38	84.31	84.33	84.29	84.33	84.3	84.25
[4]AD	----	-------	84.26	84.38	84.31	84.33	84.29	84.33	84.3	84.25*
Iteration 2	1	42.74	-.08	-.06	-.02	-.03	.03	8.61	16.56	16.59
	2	49.00	.1	.002	.02	.03	.03	9.79	17.91	17.94
	3	42.26	.05	-.03	.03	-.05	.03	8.39	13.61	13.65
	4	37.3	.08	-.06	.05	.02	-.06	7.41	12.54	12.49
[3]Dist	----	--------	85.98	86.12	86.04	86.07	86.03	86.08	69.44	69.40
[4]AD	----	-------	85.95	86.09	86.02	86.04	86.01	86.06	69.41*	69.58
Iteration 2000	1	39.00	16.42	15.60	13.01	17.32	34.30	44.89	49.44	50.94
	2	46.92	29.09	30.71	33.74	38.56	43.70	46.61	47.30	48.43
	3	43.71	57.14	55.60	49.66	44.35	40.81	41.00	41.07	42.27
	4	33.90	29.67	29.43	28.02	30.89	31.66	32.09	33.31	37.00
[3]Dist.	----	-------	32.03	31.17	30.32	27.57	7.96	7.93	10.79	12.51
[4]AD	----	-------	32.10	31.15	30.23	27.49	7.87*	8.05	10.94	12.46

*Indicates minimum distance, corresponding SOM layer node wins
[1]IN - Input node, [2]IW - Initial Weights, [3]Dist. - Distance, [4]AD - Adjusted distance

from .1 to .08 and then to .05 for each set of 1000 iterations. The winning node, in this case node 12, gets its frequency increased while the others have their frequency reduced. Therefore, for all the processing elements other than 12, the distance has been reduced by B_i to obtain the resulting D_i'. The conscience mechanism, therefore, attempts to adjust the distances so that those processing elements that are not winning with an average frequency are given a better chance to win than those that are winning at an above average frequency.

To illustrate the changes in weights and distances, the results at 2000 iterations are also shown in Table 2. Note that Nodes 5, 10, and 11 were winning at a higher than average frequency, hence their adjusted distances are higher then the actual distances.

Also note the substantial decrease in the distance measure with increased learning. For these nets the training was done for 3000 iterations after which recall was performed. During the recall phase the node with the minimum value of D_i (unadjusted) wins in the SOM layer and each of the 50 data points are assigned to a cluster by the network. With this background in mind, the methodology to assess the relative classification accuracy of the neural nets is described next.

Methodology

The data for this study were generated using Milligan's (1985) procedure, which has since been used for several comparisons of both hierarchical and non-hierarchical clustering algorithms (e.g., Cooper & Milligan, 1988; Milligan, 1980; Milligan & Cooper, 1985). The data sets varied by the following design factors: the number of clusters (4), the number of attributes (3), and the amount of error perturbation (3). The comparison of the Kohonen Learning method with the K-means clustering method was conducted as a full-factorial experimental design. A total of one hundred and eight data sets consisting of fifty points each were generated for the $4 \times 3 \times 3$ design with three replications of each cell. Each of these data sets with known cluster memberships was then clustered using the various Kohonen neural networks as well as the K-means algorithm.

Data Generation

The data were generated as truncated, multivariate normal distributions in Euclidean space. The data sets were created to exhibit the commonly accepted features of clusters, namely internal cohesion and external isolation (Cormack, 1971). *Internal cohesion* was accomplished by requiring all data points to be within the boundaries of the cluster on all dimensions. Boundaries were three standard deviations across for all dimensions. *External isolation* was defined as having the first dimension be nonoverlapping. Clusters were separated using $f(S_1 + S_2)$, where f is the separation factor and S_1 and S_2 are the standard deviations of clusters 1 and 2, respectively, on the first dimension (see Milligan, 1985). The value of f was randomly selected from a uniform distribution bounded by .25 and .75. Other dimensions were permitted to overlap. Thus, there is guaranteed cluster separation on the first dimension only in the no-error condition.

In the low and high error conditions, the data point coordinates were then perturbed to distort the true distances between points. This error perturbation simulates noise in the data, such as with data collection or measurement error. In the low error condition, overlap did not occur on the first dimension but the separation was very small. The high error condition did have some cluster overlap on the first dimension and possibly on the other dimensions. New interpoint distances were calculated using $[\sum_i (A_{ij} - A_{ik} - \delta\varepsilon_{ijk})^2]^{1/2}$, where A_{ij} and A_{ik} are the original coordinates values for observation j and k on dimension i. The number of measures (i.e., attributes) used to characterize each data set was varied so that all points in a data set were described by a 4, 6, or 8 dimensional space. The number of true clusters in each data set was 2, 3, 4, or 5 with a roughly equal distribution of points in each cluster. The error term is generated from a univariate normal distribution with a mean of zero and a standard deviation determined by taking the average of the standard deviations on dimension i for the two clusters containing points j and k. The multiplication factor, δ, had values of 1 or 2 which determined the low and high error levels, respectively (Milligan, 1985).

Kohonen Networks

Three different Kohonen neural networks were used on the test data sets. The output layers were constrained to the number of clusters known to be in the data, (i.e., 2, 3, 4, or 5). Different networks were developed for each number of clusters and number of attributes combination. Unsupervised learning was used since the objective of clustering is to find the structure in the data. The true number of clusters was used for the final output layer as a means of direct comparison with the *K*-means results. The basis for determining misclassification was the true and known cluster membership of the data point.

Normalization of the input data is required to prepare the Kohonen algorithm for use on the nearest neighbor classification (NeuralWare, 1991). Normalization involves mapping the input vector $\mathbf{x}$ onto a sphere, so $\|\mathbf{x}\| = 1$. The output of a node before competition occurs in the Kohonen layer is given by the dot product $\mathbf{w} \cdot \mathbf{x}$ in vector form or $\|\mathbf{w}\| \, \|\mathbf{x}\| \cos(\theta)$. Since the implementation is designed to have $\|\mathbf{w}\| = 1$ and as a result of normalizing the input vector $\mathbf{x}$, the node needs to compute only the angle between the weight and input vectors. The node having the smallest angle then wins the competition. The data were normalized using the multiplicative normalization, NormMult option provided in the package. It forces each vector to end on the unit sphere by extending or shortening the input vectors.

Additionally, the effect of scaling the data before it was classified was also studied using the Type 1 networks. The minmax routine available in the Neural Nets Professional II scaled the data between -1 and $+1$. For each attribute the lowest and highest value in the data range are mapped to the lowest and highest values in the scale, while the other data values are linearly mapped within the scale range.

The Type 1 networks, without scaling the data (NN1) or after scaling the data (NN2), were run at 1000 iterations. The learning rate α was kept constant at 0.2 for these two networks. The Type 2 networks (illustrated in Figure 2), were run at 3000 iterations (SOM3000). Here the parameters were varied with the number of iterations. For the first 1000 iterations the learning rate α was kept at .2, the bias rate γ was kept at 2, and the frequency modification rate β was set at .1. For the next 1000 iterations the parameters were reduced to .15, 1.5 and .08, and for the last 1000 iterations the values were .1, 1.0 and .05 for α, γ, and β respectively. Decreasing the values of the parameters as the network learns helps minimize the oscillation of weights.

K-Means Clustering

In nonhierarchical clustering, as in the neural networks under investigation, the user specifies the number of clusters and the algorithm assigns points to the clusters based on some criterion. For the purpose of comparison of the classification accuracy of the neural nets described above, the clustering of the simulated data sets was also conducted using the FASTCLUS procedure provided in the current release of SAS (1990). Based, in part, on the *leader algorithm* (Hartigan, 1975) and the *K-means algorithm* (MacQueen, 1967), FASTCLUS uses the Euclidean distances between observations to sort data into disjoint clusters. The FASTCLUS program was used as it is widely available and is representative of the commonly available *K*-means procedures.

As one of the procedure options in FASTCLUS, the cluster means can replace the cluster seeds and the observations can thus be reassigned to clusters based on their closeness to the new seeds. This process of seed replacement continues until there is little or no change in cluster seeds from one iteration to the next.

TABLE 3

Significance Levels by Cluster, Attributes and Error Level of Misclassification

Variable	K-means	NN1	NN2	SOM
Overall Model	.244	.152	*.001	*.000
Number of Clusters	.75	*.000	*.000	*.000
Number of Attributes	.147	.989	.140	.375
Error Level	* .001	.812	.297	.288
Error Level x Number of Clusters	.882	.459	.414	*.002
Error Level x Number of Attributes	.174	.725	.358	*.048
Number of Clusters x Number of Attributes	.661	.892	.375	.115
Error Level x Number of Clusters x Number of Attributes	.732	.715	.110	.409

* Significant at .05 level

Analysis

The results of the four clustering techniques employed on each of the 108 data sets were examined in two ways. First, the overall performances of the techniques were evaluated and compared in terms of the number of observations misclassified by the different techniques. Second, the clustering results were submitted to a series of analyses of variance (ANOVAs) using the general linear models procedure (GLM) available in the current edition of SAS (1990) to determine the effect of the data characteristics on the clustering results. Repeated measures were not assumed in any of the analyses since each algorithm was applied independently to the data sets, resulting in a uniquely generated set of misclassifications. Thus, each of the analyses was performed on 108 data sets, representing a fully-crossed factorial design of 3 (no, low, or high error) × 3 (four, six, or eight attributes) × 4 (two, three, four, or five clusters), with three replications per cell. The overall results of these analyses are presented in Tables 3 and 4.

The FASTCLUS procedure recovered the correct cluster structure of the error-free data with no more than five iterative recomputations of the cluster seeds. The majority of the low- and high-error data sets were also correctly classified by the FASTCLUS procedure. It is instructive to note that only a few of the data sets were poorly classified which increased the average number of misclassifications across all data sets.

TABLE 4

Mean Misclassifications of Observations
by Error, Number of Clusters, and Number of Attributes

Variable	Level	K-Means (in %)	NN1 (in %)	NN2 (in %)	SOM3000 Network (in %)
Overall Average		1.23	9.12	8.86	12.21
Error	None	0	9.44	10.84	10.66
	Low	.22	8.06	6.84	12.72
	High	3.78	9.88	8.94	13.56
Number of Clusters	2	1.48	2.00	2.60	2.96
	3	.66	6.82	2.00	3.78
	4	2.00	8.44	14.22	17.34
	5	1.18	19.26	16.66	25.18
Number of Attributes	4	1.84	8.84	11.72	11.94
	6	2.06	11.72	8.12	13.78
	8	.12	8.78	6.78	11.22

The ANOVA model was not significant overall ($p = .244$) as reported in Table 3. Only the main effect for the level of error in the data would have been significant at the .01 level ($N = 108$, $p < .001$). The mean misclassification of observations was 0.0%, 0.22%, and 3.78% for the no-error, low-error, and high-error data sets, respectively. Increasing the number of attributes improved cluster recovery, though not significantly ($p = .147$). Previous research results that indicate that more, relevant attributes should lead to better recovery (e.g., Milligan, 1980). The K-means method was generally insensitive to the number of clusters with only 1–2% of the points misclassified as reported in Table 4.

The Type 1 NN1 network used NormMult normalization *without scaling the data*. The 1000 iteration results are reported in Table 3 and 4. The overall model was not significant. There would only have been a significant main effect for the number of true clusters present in the data sets. As the number of true clusters increased, the networks were increasingly unable to recover the appropriate cluster structure with misclassifications increasing from 2.00% to 19.26%.

The Type 1 NN2 uses the same normalization procedure as NN1 but *scales the data* between −1 and +1. When the results were analyzed, the model for 1000 iterations was significant at the $p < .001$. Only the main effect for the number of clusters was

found to be significant at $p = .0001$. Whereas the NN1 data produced an ascending line of misclassifications as the number of clusters increased, the scaled data in NN2 produced a stair-step pattern (see Table 4). There was a sharp break between three and four clusters in the NN2 data sets. NN2 data had better recovery for the three cluster solutions and more misclassifications for the four cluster data compared with NN1 results.

The Type 2 network illustrated in Figure 2 was also tested. Results were similar to or slightly less positive than those for the basic Kohonen networks. The mean percentage of misclassification of observations for the Self-Organizing Map with conscience (SOM3000) network are presented in Table 4. The level of misclassification for the SOM3000 was similar to the two basic Kohonen networks examined for the number of clusters and the number of attributes. The stair step pattern of NN2 was again exhibited across the number of clusters. Some researchers have pointed out the potential problems with SOMs (Huntsberger & Ajjimarangee, 1990) and the results reported in Murtagh (1991b) suggest that the SOM may not be appropriate for some clustering problems. However, it was expected in this case that the use of the conscience mechanism would improve the clustering results, especially as the data sets had more or less an equal number of points per cluster. In this regard, Balakrishnan, et al. (1992) for instance had found that incorporating a conscience mechanism tended to result in clusters of relatively equal size.

The analysis of the SOM3000 clustering results produced one main effect for the number of clusters ($p = .0001$). Two significant two-way interactions occurred between the level of error and the number of attributes ($N = 108$, $p = .048$) and for the level of error and the number of true clusters ($N = 108$, $p = .002$).

Conclusions and Directions for Future Research

The purpose of the study was to compare a class of neural networks and a *K*-means algorithm on known data sets with known cluster structure. The results of the study are summarized below:

1. The *K*-means method performed well on the experimental data sets. Overall, the *K*-means method, as represented in FASTCLUS, demonstrated the best performance, compared with the neural networks studied. Misclassification of observations in the data sets for the *K*-means method was between zero and two percent, except for the high error condition.
2. Error in the data consistently affected the performance of the *K*-means method. Solution quality was excellent for the error-free data. However, as error was added to the data, the solution quality did deteriorate. This pattern, it is important to note, did not hold for the neural networks. That is, the neural networks investigated were less sensitive to the kind of error under investigation, although their average misclassification rate was higher.
3. The number of clusters in the data consistently affected cluster recovery in the neural networks unlike for the *K*-means. The Kohonen algorithm had a tendency to overcluster as the number of clusters increased. This tendency was more pronounced for the scaled data (NN2). A close inspection of the classification outputs indicates that for some data sets, two true clusters were combined by the neural networks into one large cluster.
4. A suggested improvement to mitigate the above problem of over clustering (i.e., collapsing a number of different true clusters into one larger cluster) on the Kohonen model is to give it a "conscience". The use of such a conscience mechanism is to ensure that any one node does not take an overly large pro-

portion of the data points. In other cases, this addition was found to have improved model results. However, in the present study, the conscience mechanism did not improve results.

5. Scaling the data for the neural networks changed the pattern of results but did not substantially improve the results.
6. The clustering literature suggests that cluster recovery increases with an increase in the number of attributes (Milligan & Cooper, 1987). The rationale being that the added dimensions gives greater opportunity for triangulating on the correct solution. However, the number of attributes generally was not significant when compared with the number of clusters and the error levels in this experiment. The *K*-means results are in the anticipated direction with fewer misclassifications as the number of attributes increased, but the neural networks do not exhibit this same pattern.
7. The *K*-means algorithm converged in relatively few iterations, (i.e., in five or fewer).

Limitations

The experimental data employed in the study were created to be representative of a number of different conditions using a data generation methodology well-accepted in the clustering literature. However, the analyses conducted on the data sets have at least two limitations. First, the number of data points may be considered relatively small when compared to those commonly used for neural networks. Also, the FASTCLUS procedure was designed to be used with data sets having more than 100 observations. According to the SAS User's Guide, FASTCLUS solutions for smaller data sets are vulnerable to order effects. However, this was not found to be a problem for the error-free data in this study. Second, conducting the analysis on error-free data and knowing the correct number of clusters in the data may have favorably biased the results as the most serious problem in the *K*-means approach is the selection of initial cluster seeds (see Milligan, 1980). Using the correct number of clusters and error-free data significantly reduced the problem of selecting the seeds. Analyses conducted on data with skewed distributions or other forms of error would likely result in a lower hit rate. Despite these limitations, the *K*-means algorithm performed well and is widely available to researchers.

Although the Neural Networks chosen for this study did not perform well, several contributing factors should be considered that would provide the basis for future research in the area. (We are indebted to an anonymous referee for motivating the points discussed here.) It should first be noted that given the unsupervised learning approach used in the basic Kohonen and SOM networks, these networks are a natural starting point to explore for use in clustering. However, at the same time it should be kept in mind that these networks were originally designed for vector quantization and topology-preserving mappings, respectively. Consequently, Kangas, Kohonen, and Laaksonen (1990) warn that these networks are susceptible to misclassification problems when they are required to separate the signals into categories. They suggest that for pattern recognition or other decision processes it is possible to significantly improve recognition accuracy by "fine tuning". Unfortunately, such a "fine tuning" process often requires additional information. These may take the form of more data points that may not often be available in many situations involving data collected from behavioral sciences experiments. More importantly, the information needed to minimize misclassification involves employing multiple codebook (i.e., weight) vectors to represent each cluster. These multiple vectors, they suggest, that are needed are the vectors defining the boundaries of the clusters rather than the typical centroids. Unfortunately, in the

context of applied research, the very boundaries of the clusters are typically not known and have to be determined.

Another limitation of the study is the structure of the data that was used. Since all of the data used for the study consisted of hyperellipsoidal clusters, it would tend to favor the *K*-Means approach. On the other hand data from cluster patterns that are nonhyperellipsoidal may favor NNs as opposed to *K*-means. However, the focus of this study on hyperellipsoidal clusters is motivated by the belief that in the behavioral and social sciences the definition of cluster has come to be closely associated with the hyperellipsoidal pattern.

Additionally, it must be noted that the Kohonen learning algorithm does not deterministically converge, hence as long as the learning rate (α) is positive the weights will be updated leading to weight oscillation. Thus, there is no guarantee that when learning is terminated that the weight vectors represent the true centroid of the cluster. One way to get around this is to gradually decrease the value of α to zero as the learning progresses to freeze the centroids. In this study the learning rate was kept constant for the Type 1 networks, while it was decreased over a number of iterations in the Type 2 networks.

Future Directions

This paper has begun to examine the interrelationships between one kind of neural network and a traditional kind of cluster analysis in line with the suggestions of Bezdek (1991). Future work should include:

1. Additional comparisons of neural networks with traditional methods across a wider variety of settings, methods, and data. For example, the results presented here may not be generalizable to other types of error, such as outliers or random noise dimensions.
2. Different network construction. The number of nodes in the hidden layers and the number of hidden layers may affect results.
3. One difficulty with the neural networks was the merging of two or more true clusters in the final solution. Once a large cluster is obtained, it would be instructive to subject these points to a separate network analysis to determine if they can be further broken down correctly into smaller-sized clusters.
4. Other neural networks that are designed to perform clustering kinds of analyses should be analyzed.
5. This paper has addressed one of these characteristics of model validation: performance validation. The other characteristics, such as complexity, convergence, stability, and robustness should be addressed in future research. In addition, comparing the various techniques on real data, such as in Balakrishnan, et al. (1992), should be continued.

References

Ahalt, S. C., Krishnamurthy, A. K., Chen, P., & Melton, D. E. (1990). Competitive learning algorithms for vector quantization. *Neural Networks, 3,* 277–290.

Balakrishnan, P. V., Cooper, M. C., Jacob, V. S., & Lewis, P. A. (1992). A cluster analysis of brand choice data using neural networks. In V. S. Jacob & H. Pirkul (Eds.), *Proceedings of the Twentieth International Business School Computer Users Group* (pp. 365–375). Columbus, OH.

Bezdek, J. C. (1990). A note on generalized self-organizing network algorithms. *Proceedings of the Society of Photo Optical Instrumentation Engineers Applications of Artificial Intelligence VII* (pp. 260–267). Bellingham, WA: Society of Photo Optical Instrumentation Engineers.

Bezdek, J. C. (1991). Pattern recognition, neural networks, and artificial intelligence. In M. Trivedi, *Proceedings of the Society of Photo Optical Instrumentation Engineers Applications of Artificial Intelligence VII* (pp. 1–12). Bellingham, WA: Society of Photo Optical Instrumentation Engineers.

Bezdek, J. C., & Bobrowski, L. (1990). Matching 'boxy' data: Clustering with the Minkowski norms. *Proceedings of Third International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems* (pp. 295–299). Paris, France: Ministere de la Recherche et de la Technologie.

Bezdek, J. C., & Hathaway, R. J. (1990). Generalized regression and clustering. *Proceedings of the International Conference on Fuzzy Logic and Neural Networks* (pp. 575–578). Iisuka, Japan: Kyusu Institute of Technology.

Bucklin, R. E., & Srinivasan, V. (1991). Determining interbrand substitutability through survey measurement of consumer preference structures. *Journal of Marketing Research, 27* (February), 58–71.

Caudill, M. (1987). Neural networks primer part I. *AI Expert, 8* (December) 46–52.

Cooper, M. C., & Milligan, G. W. (1988). The effect of measurement error on determining the number of clusters in cluster analysis. In W. Gaul & M. Schader (Eds.), *Data analysis, decision support and expert knowledge representation in marketing and related areas of research* (pp. 319–328). Berlin: Springer Verlag.

Cormack, R. M. (1971). A review of classification. *Journal of the Royal Statistical Society, Series A, 134*, 321–367.

DeSieno, D. (1988). Adding a conscience to competitive learning. *Proceedings of the International Conference on Neural Networks* (pp. 117–124). San Diego, CA: IEEE Society.

Dubes, R. C. (1987). How many clusters are best?—An experiment. *Pattern Recognition, 20*(6), 645–663.

Fort, J. C. (1991). Kohonen algorithm and the traveling salesman problem. In F. Murtagh (Ed.), *Neural networks for statistical and economic data* (pp. 113–118). Dublin: Munotec. (Supported by the Statistical Office of the European Communities, Luxembourg)

Gath, I., & Geva, A. B. (1988). Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 11*(7), 773–781.

Hartigan, J. A. (1975). *Clustering algorithms*. New York: John Wiley and Sons.

Hecht-Nielsen, R. (1990). *Neurocomputing*. Reading, MA: Addison-Wesley.

Huntsberger, T. L., & Ajjimarangee, P. (1990). Parallel self-organizing feature maps for unsupervised pattern recognition. *International Journal of General Systems, 16*(4), 357–372.

Kangas, J. A., Kohonen, T. K., & Laaksonen, J. T. (1990). Variants of self-organizing maps. *IEEE Transactions on Neural Networks, 1*(1), 93–99.

Kohonen, T. (1982a). Self-organized formation of topologically correct feature maps. *Biological Cybernetics, 43*, 59–69.

Kohonen, T. (1982b). Analysis of a simple self organizing process. *Biological Cybernetics, 44*, 135–140.

Kohonen, T. (1989). *Self organization and associative memory*. (3rd edition). New York: Springer-Verlag.

Krishnamurthy, A. K., Ahalt, S. C., Melton, D. E., & Chen, P. (1990). Neural networks for vector quantization of speech and images. *IEEE Journal on Selected Areas in Communications, 8*, 1449–1457.

Lippmann, R. P. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine, 4*, pp. 4–22.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1*, 281–297.

Milligan, G. W. (1980). An examination of the effect of six types of error perturbation on fifteen clustering algorithms. *Psychometrika, 45*, 325–342.

Milligan, G. W. (1985). An algorithm for generating artificial test clusters. *Psychometrika, 50*, 123–127.

Milligan, G. W., & Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika, 50*, 159–179.

Milligan, G. W., & Cooper, M. C. (1987). Methodology review: Clustering methods. *Applied Psychological Measurement, 11*(4), 329–354.

Murtagh, F. (1991a). A short survey of neural networks for forecasting. In F. Murtagh (Ed.), *Neural networks for statistical and economic data* (pp. 87–94). Dublin: Munotec. (Supported by the Statistical Office of the European Communities, Luxembourg)

Murtagh, F. (1991b). *Neural networks for classification and discrimination* (working paper). Munchen, Germany: Space Telescope—European Coordinating Facility, European Southern Observatory, Karl-Schwarzschild-Strasse 2, D-8046 Garching bei Munchen. (To be included in P. Arabie, L. J. Hubert, & G. De Soete (Eds.), *Combinatorial data Analysis*)

Nelson, M. M., & Illingworth, W. T. (1991). *A practical guide to neural networks*. Reading, MA: Addison Wesley.

NeuralWare. (1991). *Neuralworks Professional II/PLUS* (Manual). Pittsburgh, PA: Author.

Openshaw, S., & Wymer, C. (1991). A neural net classifer system for handling census data. In F. Murtagh (Ed.), *Proceedings Neural Networks for Statistical and Economic Data* (pp. 73–86). Dublin: Munotec Systems. (Supported by the Statistical Office of the European Communities, Luxembourg)

Osipenko, V. V. (1988). Solution of a double clusterization problem with the use of self-organization. *Soviet Journal of Automation and Information Sciences, 21*(3), 77–82.

Rowher, R. (1991). Neural networks for time-varying data. In F. Murtagh (Ed.), *Proceedings Neural Networks for Statistical and Economic Data* (pp. 59–69). Dublin: Munotec Systems. (Supported by the Statistical Office of the European Communities, Luxembourg)

Rumelhart, D. E., McClelland, J. L., & the PDP Research Group (1987). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*. Cambridge, MA: MIT Press.

SAS Institute. *SAS User's Guide: Statistics*, (Version 5 Edition). (1990). Cary, NC: Author.

Srivastava, A., & Murty, M. N. (1990). A comparison between conceptual clustering and conventional clustering. *Pattern Recognition, 23*(9), 975–981.

Wan, S. J., Wong, S. K. M., & Prusinkiewicz, P. (1988). An algorithm for multidimensional data clustering. *ACM Transactions on Mathematical Software, 14*(2), 153–162.

Manuscript received 1/8/93
Final version received 10/20/93