A Comparison of Self-Organizing Neural Networks for Fast Clustering of Radar Pulses

Eric Granger and Yvon Savaria

Electrical and Computer Engineering Dept.

École Polytechnique de Montréal

P.O. Box 6079, Station "Centre-ville"

Montreal, Qc., Canada, H3C 3A7

{egranger, savaria}@grm94.polymtl.ca

Pierre Lavoie

Defence Research Establishment Ottawa

Department of National Defence

Ottawa, On., Canada, K1A 0Z4

Pierre.Lavoie@dreo.dnd.ca

Marc-André Cantin

Computer Science Dept.

Université du Québec Montréal

P.O. Box 8888, Station A

Montreal, Qc., Canada, H3C 3P8

cantin@info.uqam.ca

Abstract

Four self-organizing neural networks are compared for automatic deinterleaving of radar pulse streams in electronic warfare systems. The neural networks are the Fuzzy Adaptive Resonance Theory, Fuzzy Min-Max Clustering, Integrated Adaptive Fuzzy Clustering, and Self-Organizing Feature Mapping. Given the need for a clustering procedure that offers both accurate results and computational efficiency, these four networks are examined from three perspectives — clustering quality, convergence time, and computational complexity. The clustering quality and convergence time are measured *via computer simulation, using a set of radar pulses collected in the field. The effect of the pattern presentation order is analyzed by presenting the data not just in random order, but also in radar-like orders called burst and interleaved. Estimation of the worst-case running time for each network allows for the assessment of computational complexity.*

1 Introduction

The purpose of radar electronic support measures (ESM) is to search for, intercept, locate, and analyze radar signals in the context of military surveillance [1] [2] [3]. When a radar ESM system is illuminated by several radars, it typically relies on pulse repetition interval (PRI) parameters to deinterleave the intercepted pulse trains. Unfortunately, the multiplication of intricate PRI patterns in about every radar model has spurred an inordinate complexity in deinterleaving algorithms. Alternative approaches for clustering radar pulses are thus needed, ones that rely on direction of arrival, frequency, pulse width and other such parameters that can be obtained from individual pulses. The selection of parameters can vary greatly from one ESM system to the next due to installation, cost, size, tactical and other considerations. Yet, beyond parameter choice, clustering the intercepted radar pulses constitutes a challenging problem. This clustering must sustain a very high data rate since, in certain radar bands, the signal densities encountered can reach up to 10^6 radar pulses per second.

Self-organizing neural networks (SONNs) appear very promising for this type of clustering application, since they can cluster patterns autonomously, and lend themselves well to very high speed, parallel implementation [4] [5] [6]. Several innovative SONNs have been reported in the literature, each one demonstrating unique and interesting features. This paper presents a comparative study of four of them, all potentially suitable for solving very high throughput clustering problems. They are the Fuzzy Adaptive Resonance Theory [7], Fuzzy Min-Max Clustering [8], Integrated Adaptive Fuzzy Clustering [9], and Self-Organizing Feature Mapping [10].

The performance of these four neural networks is examined from three points of view — clustering quality, convergence time, and computational complexity. Indeed, in many practical applications, the accuracy of clustering results, and the computational efficiency of the clustering procedure are equally important. In this paper, *clustering quality* refers to the degree of similarity between the partitions (clusters) produced by a SONN, and a reference partition based on known category labels. This similarity is assessed by applying the Rand Adjusted [11] and the Jaccard [12] measures to partitions obtained by computer simulation. *Convergence time* is defined as the number of successive presentations of a finite input data set needed for a SONN's weight set to stabilize. This time is easily determined from computer simulation. *Computational complexity* is estimated from the maximum execution time required by a SONN algorithm to process one input pattern. In order to estimate this worst-case running time, we assume that the algorithm is implemented as a computer program running on an idealized random access machine (RAM) [13].

The main data set used in our simulations describes electromagnetic pulses transmitted by shipborne navigation radars. These pulses were collected from ashore by the Defence Research Establishment Ottawa (DREO) using a directional antenna, a superheterodyne tuner, a high-accuracy I/Q demodulator [14], and a two-channel, 10-bit digital oscilloscope.

The radars were observed one at a time to ensure that each file contained pulses from a single radar. The identity of each ship was obtained from a harbor control tower and used to label each file. These labels can serve as references to measure the effectiveness of clustering techniques for the application.

In this study, the outcome of numerous computer simulations are combined to yield an average similarity measure and convergence time. In addition to the Radar data set described above, two other sets, called Wine Recognition and Iris, are used for comparison. For each set, the patterns are presented in three statistically different orders: random, by burst, or interleaved in a radar-like manner. The data sets and presentation orders were selected to illustrate the commonalities and differences between the four neural networks. Simulation results and complexity estimates are analyzed with very high data rate clustering applications in mind.

The rest of this paper is organized as follows. In the next section, the main features of the four SONNs selected for this study are briefly outlined. In Section 3, the methodology used to compare these SONNs (namely, the performance measures, data sets, and data presentation orders) is described. Finally, the results are analyzed and discussed.

2 Self-Organizing Neural Networks

A clustering method suitable for radar ESM should have the following properties. First, it should not require prior knowledge of the number or characteristics of categories to be formed. Second, since the variable input arrival rate may reach 10^6 patterns per second, it should be able to cluster non-stationary streams of input patterns sequentially, without requiring their long-term storage. Lastly, the sequence of operations needed for implementing the method using current technology should lend itself well to high speed hardware realizations. Given that most of the popular, well-established classical [12] [15] [16] and fuzzy [17] [18] clustering algorithms require prior knowledge on either the number or the characteristics of clusters sought, several iterations with the data set, or storage of the entire data set in memory, none of them were considered for this comparison.

The unsupervised learning paradigm used in self-organizing neural networks (SONNs) is related to clustering, since it permits the assignment of adaptively defined categories to unlabeled patterns [6] [19]. SONNs appear promising for high data rate sequential clustering applications, since they can cluster patterns autonomously, in most cases without prior knowledge of the number of categories. Moreover, they do not require long-term storage of the input patterns, and permit adaptive determination of the shape, size, number, and placement of categories, while operating in parallel [8]. Self-organizing "neuro-fuzzy" networks have recently been developed, where fuzzy logic concepts are integrated into the SONN framework. Categories are then modeled as fuzzy sets that allow encoding the input scene's vagueness [20].

An important familly of SONNs is derived from the basic idea of competitive learning [21] [22] [23], a type of unsupervised learning. In short, competitive learning neural networks seek to determine decision regions in the input pattern space. The most elementary of these networks consists of a single layer of identical output neurons, each one fully connected to the input nodes with feedforward excitory weights, which encode the categories learned by the network. A category label is associated with each output neuron, whose features are represented by the numerical values of the set of weights connecting it to all input nodes. When a pattern is presented to the network's input nodes, it is propagated through the weights to output neurons, which enter a winner-take-all competition. The one with the strongest activation (the winner) is allowed to adapt its set of weights to incorporate the input's novel characteristics. Through this process, output neurons become selectively tuned to respond differently to given input patterns, and therefore learn to specialize for regions of the input space: they become feature detectors [24].

Four SONNs that are based on competitive learning were selected for this study: Fuzzy Adaptive Resonance Theory (FA) [7], Fuzzy Min-Max Clustering (FMMC) [8], Integrated Adaptive Fuzzy Clustering (IAFC) [9], and Self-Organizing Feature Mapping (SOFM) [10]. Although the classical ISODATA clustering method [12] [16] [25] [26] does not offer a practical solution to the problem (since it requires significant prior knowledge of the data), it is included as a reference point for clustering quality comparison, given its widespread use in statistical multivariate data analysis.

2.1 Overview of the four neural networks:

The first three SONNs (FA, FMMC and IAFC) subscribe to the basic Adaptive Resonance Theory (ART) [27] control structure. Essentially, ART networks categorize familiar inputs by adjusting previously learned categories, and create new categories dynamically in response to inputs different enough from those previously seen. A vigilance parameter ρ regulates the maximum tolerable difference between any two input patterns in a same category. Each one of these three SONNs integrates fuzzy logic concepts into the binary input ART1 [27] neural network processing framework.

Structurally, the FA neural network, proposed by Carpenter, Grossberg and Rosen [7], consists of two layers of neurons that are fully connected: an input layer $F1$ with $2M$ neurons (two per input feature) and an output layer $F2$ with N neurons (one per category). An adaptive weight value w_{ji}, represented as a real in the interval [0,1], is associated with each connection. The indices i and j denote the neurons that belong to the layers F1 and $F2$ respectively. For each neuron j of F2, the category prototype vector $\vec{w}_j = (w_{j1}, w_{j2}, ..., w_{j2M})$ contains the set of characteristics defining the category j. When complement coding is used, this vector may be interpreted as a hyperrectangle in the M-dimensional input space.

The FMMC neural network, proposed by Simpson [8], consists of 2 layers of neurons: an input layer $F1$ with M neurons (one per input pattern feature), and an output layer $F2$ with N neurons (one per category). Two weighted connections link every $F1$ neuron to an $F2$ neuron. For the $F2$ output neuron j ($j = 1, 2, ..., N$), the 2 sets of M connected weights encode a min ($\vec{u}_j = (u_{j1}, u_{j2}, ..., u_{jM})$) and a max ($\vec{v}_j = (v_{j1}, v_{j2}, ..., v_{jM})$) point. These two points define a hyperrectangle fuzzy set B_j, the network's representation of category j.

FA and FMMC are both fuzzy min-max clustering networks, whereby category prototype vectors are represented as fuzzy set hyperrectangles in the M-dimensional input space, each one entirely defined by a min and a max point [32]. Both accomplish fuzzification of the ART1 network by essentially replacing crisp logic AND operators with fuzzy logic AND (min) operators. Despite a superficial resemblance to FA, FMMC is a somewhat different network, yielding categories associated to hyperrectangles that never overlap one another. A contraction procedure is used to prevent the occurrence of hyperrectangle overlap.

IAFC is a fuzzy clustering algorithm proposed by Kim and Mitra [9] [28]. As with FMMC, IAFC contains an M neuron input $F1$ layer (one per feature), and an N neuron output $F2$

layer (one per category). These neurons are fully interconnected by bottom-up and top-down weights. The categories formed, however, are hyperspherical in shape and are represented as centroids in the M-dimensional input space. Two prototype vectors, $\vec{v}_j = (v_{j1}, v_{j2}, ..., v_{jM})$ and $\vec{b}_j = (b_{j1}, b_{j2}, ..., b_{jM})$, are associated with category j's centroid. The network's top-down weights v_{ji} encode the actual cluster centroids, whereas each bottom-up weight b_{ji} is a normalized mirror of v_{ji}, for $i = 1, 2, ..., M$ and $j = 1, 2, ..., N$.

The objective of IAFC is to obtain improved nonlinear decision boundaries among closely located cluster centroids by introducing a new vigilance criterion and a new learning rule into ART1 processing. The new vigilance criterion combines a fuzzy membership value and the Euclidean distance to form more flexible categories. The new learning rule uses a fuzzy membership value, an intracluster membership value, and a function depending on the number of data set iterations with a Kohonen-type learning rule.

For our purposes, Kohonen's Self-Organizing Feature Mapping (SOFM) [10] [29] defines a mapping from the M-dimensional input space onto a two-dimensional grid of output neurons, whose size is user-defined. The SOFM method creates a vector quantizer within the neural structure by adjusting the weights of connections linking the M input neurons of the $F1$ layer to all the N' output $F2$ map neurons. A prototype vector $\vec{w}_j = (w_{j1}, w_{j2}, ..., w_{jM})$ is associated with every one of these map neurons. When an input pattern is presented, each map neuron computes the Euclidean distance of its prototype vector to the input. The closest prototype corresponds to the best-match map neuron. The weights connected to this neuron, as well as those of the other neurons in its neighborhood are adjusted to learn from the input. If the lateral map distance from the best-match neuron to the location of another one falls within the perimeter defined by the *neighborhood radius*, then this other neuron belongs to the neighborhood. Throughout learning, the neighborhood's radius, $r(t)$, and the learning rate, $\alpha(t)$, are progressively decreased in a linear way from their initial values, $r(0)$ and $\alpha(0)$, until they become equal to 1 and 0, respectively.

As learning evolves, weights become organized such that topographically close map neurons are selectively tuned to inputs that are physically similar. SOFM clustering is characterized by the formation of a topographic feature map in which the spatial coordinates of its neurons correspond to intrinsic features of the input patterns. Therefore, SOFM can be seen as a network producing an unsupervised nonlinear projection of the probability density function of the high-dimensional input data onto a two-dimensional display [10].

2.2 Modifications for radar pulse clustering:

Neither one of the four networks in its original form is suitable for sequential clustering of radar pulses in practical ESM systems. The aim of this subsection is to propose appropriate modifications.

FA, FMMC, and IAFC are ART-type networks, whereby output neurons are selected using a potentially demanding search process. The computational cost of the iterative category choice and corresponding expansion testing, through N output neurons, is $O(N^2)$. However, performing the category expansion test on all N output neurons, prior to a direct category choice, can significantly accelerate processing in cases where several output neurons would fail the expansion test. The computational cost of the resulting search process is $O(N)$.

The plasticity of ART-type networks like FA and FMMC has been well documented. When an input resembles none of the stored prototypes, an output neuron is committed, which amounts to creating a new category with the input as its prototype. For radar ESM applications, this implies that these networks can learn a new category from a single, or very few radar pulses. One drawback is that they can produce artifact categories, and thus possibly trigger false alarms, if corrupt pulses are encountered. In practice, some means of managing such false alarms would be required. Notice that IAFC is also an ART-type network, where plasticity is controlled by a learning rate parameter, λ, which varies with the number of data set iterations, and with the mismatch between inputs and winning category prototypes.

Since the environment around an ESM system changes over time as radars come and go, categories that have not been activated for a long time must be freed. In this paper, the data sets are sufficiently small that we need not bother with category removal. In practice however, category reuse would be necessary for maintaining an up-to-date representation of the environment, for accuracy, and for making efficient use of computing resources.

SOFM, and to some extent IAFC, permit stable learning of categories by progressively decreasing plasticity, that is, by reducing the influence of subsequent inputs on existing weights. Plasticity is controlled by the parameters $r(t)$ and $\alpha(t)$ in SOFM, and λ in IAFC. In both cases, the parameter values are gradually decreased over time, and the networks eventually loose their ability to react to new information. This approach is acceptable for batch data processing like in this paper. In practice, however, new radars can appear in the theater of operation at any time, and both IAFC and SOFM would require modification to learn continuously.

In FA, FMMC, and IAFC, input patterns are regrouped into a *variable* number of categories, each one represented by a prototype vector associated with a single output neuron.

By contrast, SOFM clusters inputs into a *fixed* number of output neurons, organized into a 2-dimensional topographic map. During training, the network adapts the prototype vectors for all its map neurons, and assigns an actual data cluster to a region in the map, which may contain one or more map neurons. This process raises the problem of locating categories in the map. Besides, it can be difficult to specify a feature map size when the potential solutions have a relatively small number (2 or 3) of clusters [30]. A number of interpretation procedures have been proposed to assist SOFM for clustering [5] [30]. In order to ease the interpretation of the feature maps in this paper, and to permit a fair comparison with the other ART-type SONNs, an additional mechanism was incorporated into SOFM's processing. This mechanism is activated after the initial topographical ordering phase has ended, when the map is being fine-tuned for statistical accuracy. Define a *category center* as a map neuron which has been assigned to more than ζ input patterns. All the other map neurons can be related to the category centers using a minimum Euclidean distance criterion. Thus, when an input pattern selects one of these other neurons as the best-match, the label of the category center neuron with the closest prototype vector is output by the modified network. This modification makes it possible to locate and count categories, whose number may vary as learning unfolds.

3 Comparison Method

The methodology used is based on three different performance measures — clustering quality, convergence time, and computational complexity. This methodology is different from that of other comparisons reported in literature (for example, [31] - [36]), in which clustering speed and response time are less critical than in our application. These measures are defined in this section.

3.1 Clustering quality:

A partition of n patterns into K groups defines a *clustering.* This can be represented as a set $A = \{a_1, a_2, ..., a_n\}$, where $a_h \in \{1, 2, ..., K\}$ is the category label assigned to pattern h. The degree of match between two clusterings, say A and B, may be compared by constructing a contingency table, as shown in Fig. 1. In this figure, c_{11} (c_{22}) is the number of pattern pairs that are in a same (different) cluster in both partitions. The value c_{21} is the number of pattern pairs that are placed in a same cluster by A, but in different clusters by B. The value c_{12} reflects the converse situation. The sum of all elements $m = c_{11}+c_{12}+c_{21}+c_{22} = n(n-1)/2$ is the total number of combinations of two out of n patterns. The four variables within the contingency table have been used to derive measures of similarity between two clusterings A and B [12] [15]. These measures are known in pattern recognition literature as external criterion indices, and are used for evaluating the capacity to recover true cluster structure. Based on a previous comparison of these similarity measures [37], the Rand Adjusted [11], defined by:

$$S_{RA}(A, B) = \frac{2(c_{11}c_{22} - c_{12}c_{21})}{2c_{11}c_{22} + (c_{11} + c_{22})(c_{12} + c_{21}) + c_{12}^2 + c_{21}^2} \tag{1}$$

and Jaccard statistic [12], defined by:

$$S_J(A, B) = \frac{c_{11}}{c_{11} + c_{12} + c_{21}} \tag{2}$$

have been selected to assess clustering quality for this study. It is worth noting that variable c_{22} does not appear in $S_J(A, B)$.

Since correct classification results are known for the data sets used, their patterns are all accompanied by category labels. These labels are withheld from the SONN under test, but they provide a reference clustering, R, with which a clustering produced by computer simulation, A, may be compared. Then, variables c_{11} and c_{22} in Fig. 1 represent the number of pattern pairs which are properly clustered together and apart, respectively, while c_{12} and c_{21} indicate the improperly clustered pairs. In this case, Eqs. 1 and 2 yield *scores* that describe the quality of the clustering produced by a SONN. Both the Rand Adjusted and Jaccard measures yield a score ranging from 0 to 1, where 0 denotes maximum dissimilarity, and 1 denotes equivalence. The closer a clustering A is to R, the closer the scores are to 1. Notice the dependence of these scores on the number of clusters in A and R.

3.2 Convergence time:

During a computer simulation, each complete presentation of an input data set to a SONN is called an *epoch*. Convergence time is conveniently measured by counting the number of epochs needed for a SONN to converge. Once convergence is reached, weight values remain constant during subsequent presentations of the entire data set in any order. This measure is independent from computational complexity (as defined in the following subsection), since an algorithm may require several epochs to converge using very simple processing, or vice-versa. The product of the two measures provides useful insight into the amount of processing required by each SONN to produce its best asymptotic clustering quality, while sustaining a desired clustering rate.

3.3 Computational complexity:

A first order approximation of the computational complexity for the SONN algorithms may be obtained by assessing their execution time on an idealized computer. Thus, the time complexity, T, combined with a fixed computing area C, allows for comparison of area-time complexities, CT. To that effect, assume that the SONN algorithms are implemented as computer programs running on a generic, single processor, random access machine (RAM) [13], where instructions are executed one after the other. This generic machine is capable of no more than one operation per cycle (i.e., neither VLIW or superscalar). Using the RAM model avoids the challenging task of accurately determining the performance of specific VLIW or superscalar machines, which is beyond the scope of this analysis.

Time complexity can be estimated from the maximum execution time required to process a single input pattern. The result is a total worst-case running time formula, T, which summarizes the behavior of a SONN algorithm as a function of two key parameters: the dimensionality of the input patterns, M, and the number of $F2$ output neurons, N. Specifically, T can be defined as the sum of the worst-case running times T_p for each operation p that is required to process an input [13]:

$$T = \sum_p T_p = \sum_p o_p \cdot n_p \tag{3}$$

where o_p is the constant amount of time needed to execute an operation p, and n_p is the number of times this operation is executed.

For simplicity, we assume that p can take one out of two values, 1 or 2, where o_1 is the time required to execute an elementary operation such as: $x+y$, $x-y$, $\max(x,y)$, $\min(x,y)$, $1-x$, $x<y$, etc., and o_2 is the time needed to compute a division x/y, a multiplication $x \cdot y$, a square root $\sqrt{x}$, or an exponent e^x. In addition, x and y are assumed to be real numbers represented by an integer with a b bit resolution, where b corresponds to the number of bits needed to represent each SONN's elementary values (i.e., synaptic weights, input pattern elements, and parameters) with a sufficient precision. Operations of type $p=2$ are more complex than those of type $p=1$, and their complexity, which is a function of b, depends on their specific implementation. Nevertheless, to simplify the analysis, we presume that $o_2 \simeq F(b) \cdot o_1 = F \cdot o_1$, where F remains as an explicit parameter in the complexity formulas.

3.4 Data set:

The data gathered by DREO consists of 800 radar pulses from 12 different shipborne navigation radars. As with any field trial, the recorded signals show imperfections in the radar transmitters, and exhibit distortion and noise due to the propagation channel and the collection equipment. For instance, the signal-to-noise ratio varies greatly from pulse to pulse, and from file to file (from 15 to 45 dB). This is due to the circular scan of the radars, and their varied power and distance from the collection site. Also, one file contains pulses with unusual characteristics indicative of a radar operating with a defective magnetron tube. These pulses were not removed, for they reflect anomalies sometimes encountered in the field.

After the trial, DREO reduced the dimensionality of the data to simplify computer simulation and eventual implementation of the clusterer. Thus, 15 real-valued features were extracted from each radar pulse. The feature extraction algorithm has its own limitations, and produced a few outliers, which were not removed. The final data set was then normalized using a linear transformation so that values in every dimension range between 0 and 1.

Note that special attention was paid to ensure that clustering the data set would be difficult enough to fully exercise the algorithms. One should therefore focus on their relative, rather than absolute, performance.

The number of pulses per file is nominally 50, but two files contain 100 and 200 pulses, respectively. An uneven number of pulses per radar is typical of the application. Indeed, some radars can transmit in excess of 300000 pulses/second in their pulse Doppler mode, and even a few thousand pulses/second in their medium PRI mode. The small number of pulses per file in the data set is a consequence of the collection equipment capabilities. Nevertheless, a train of 50 pulses corresponds to a few beam illuminations by a radar in medium PRI mode, and should be sufficient for ESM detection.

Fig. 2 shows the result of a two-dimensional principal component analysis (PCA) and linear discriminant analysis (LDA) projections for the data set. Although both PCA and LDA are linear projection methods, LDA benefits from *a priori* cluster label information. Clearly, the clusters cannot be described by the same statistics. Some are not Gaussian distributed, and they sometimes overlap one another in 2D.

The Radar set described above represents data from a stream of pulses collected for one realistic scenario, under which a SONN would be called upon for automatic deinterleaving. Even though the data set is representative, it is far from comprehensive: there are several types of radar ESM systems, numerous different theaters of operation, and an ever growing variety of radar sources. Consequently, two standard data sets, called Wine Recognition and

Iris, were used to broaden the scope of this research. A brief description, and the simulation results obtained on these sets can be found in Appendix A.

3.5 Data presentation order:

Since sequential clustering results depend on the order of presentation of the patterns, three types of presentation orders are considered in this comparison.

A *random* presentation order consists of random permutations of the input data set patterns, regardless of the class.

A *burst* presentation order is defined by a random sequence of groups of patterns called bursts, where each burst contains patterns from the same class. The specific patterns which form the burst, and the class associated with the burst are both selected at random (from classes with remaining patterns). The burst size is a random variable with a normal distribution, whose average is equal to 5 and whose standard deviation is equal to 3. Bursts of negative size are ignored.

A radar-like *interleaved* presentation order is obtained as follows. First, the patterns in each class are permuted at random. Then, four fictitious radar parameters are chosen at random for each class. Initial scan time (IS) and scan period (SP) are chosen from a uniform distribution between 0 and 1. Beamwidth (BW) and pulse repetition interval (PRI) are selected from uniform distributions between 0 and $0.25 \cdot SP$, and between 0 and $0.25 \cdot BW$, respectively. These parameters allow the computation of a time of arrival (TOA) for every pattern of each class. Sorting the patterns by TOA yields an interleaved, radar-like, order of presentation.

4 Comparison Results

4.1 Clustering quality and convergence time:

For simulation purposes, MATLAB programs were written for FA, FMMC, IAFC, and ISODATA algorithms. SOFM simulations were done using the "SOM_PAK" software[1]. Prior to each simulation, the normalized input patterns of the data set were organized according to one of the three statistical orders (random, burst or interleaved). The sequence of patterns was then repeatedly presented to the algorithms under test until weights remained stable for two successive epochs, yielding the clustering A. The data set presentation order was kept constant from one epoch to the next. After the simulation, the Rand Adjusted and Jaccard scores were computed, using the reference clustering, R, available for the data. The scores were stored along with the number of epochs needed for convergence. The results of 20 distinct simulations were combined to extract central tendencies, and to determine the effect of presenting the input patterns in different orders.

Simulation results for the Radar data set are given in Table 1. (Similar results obtained using the Wine Recognition, and Iris data are given in Appendix A.) The table contains the mean and standard deviation of the best scores that can be obtained from the four SONNs and ISODATA for all three statistical presentation orders. Convergence time, and parameter values[2] as obtained by trial and error are also shown. These parameter values always reflect the least amount of processing needed to produce the best clustering score.

For Table 1, the Rand Adjusted and Jaccard scores appear to describe similar trends.

[1] This software was obtained via anonymous FTP from cochlea.hut.fi.

[2] The reader is referred to the notation used for parameters in the original papers describing ISODATA, FA, FMMC, IAFC, and SOFM.

ISODATA confirmed our expectations by yielding the highest scores for all but one simulation, and thus offers a realistic performance target for the SONNs.

SOFM is the SONN that achieves the best overall scores, or rather score, since this network produces consistent clusterings that are the same across all presentation orders. This is a direct consequence of its slow convergence, combined with the map interpretation post-processing. This excellent score ($S_{RA}(A, R) = 0.95$) is comparable to those of ISODATA, but it is obtained after no less than 50 epochs. Remember that convergence speed is under user control, and it was set to yield the best possible score.

FA usually scores second best, achieving results that are significantly better with the interleaved presentation order ($S_{RA}(A, R) = 0.80$) than with either the random ($S_{RA}(A, R) = 0.61$) or the burst ($S_{RA}(A, R) = 0.59$) presentation orders. These scores are significantly lower than those of SOFM and ISODATA, but they are obtained after only 3 to 4 epochs, indicating that it is a very responsive network.

FMMC converges even faster than FA, and this may explain that it yields mixed results. Noteworthy is the low Rand Adjusted score ($S_{RA}(A, R) = 0.45$) obtained for the random order of presentation, in contrast to the corresponding Jaccard score ($S_J(A, R) = 0.92$), which is not as bad. Scores for the burst and interleaved orders are comparable to those of FA.

IAFC takes from about 17 to 22 epochs to converge, but yields scores that are no better than those of FA and FMMC. Nonetheless, the standard deviation of the scores is low, meaning that the results obtained are consistent from one simulation to the next.

An attempt was made to improve FA scores by means of a post-processing mechanism somewhat similar to SOFM's map interpretation procedure. Unfortunately, this did not yield higher overall scores, and thus the results are not shown in Table 1. It did, however, allow good scores to be obtained over a broader range of vigilance parameter values.

The results in Table 1 clearly indicate that the data presentation order has a significant impact on performance. With the exception of SOFM, the algorithms usually perform better as the input presentation order moves from the completely random case to the more structured ones, where patterns from a category may be presented together. The standard deviation of the scores are also dependent on the data presentation order. For instance, with FA and FMMC, the scores often vary more for the interleaved than for the burst, and more for the burst than for the random orders. In most cases, high scores appear to be correlated with high variance for the burst and interleaved presentation orders. With these orders, sequences are defined by mixtures of bursts of patterns from the same cluster. If we accept that scores depend on the size of the bursts, then the randomly defined burst sizes may explain the high variance of the scores. Burst sizes have no notable effect on convergence time.

4.2 Computational complexity:

For brevity, only the final results of our analysis are shown here. The reader is referred to Appendix B for further details. Table 2 shows the total worst-case running times per input pattern in a normalized format (T/o_1), when parameter F is the same for all operations of type $p = 2$. The corresponding growth rates, valid when the parameters M and $N \gg 1$, and the parameter F is constant, are also given. It turns out that the complexity of all SONNs is described by the same asymptotic growth rate, $O(NM)$.

Overall, the results in Table 2 indicate that FA's worst-case processing is the least sensitive to the parameters M, N and F. Its processing time is consumed mainly by the winner-take-all competitions between output neurons (choice function computations). Among the other networks, IAFC has the second lowest worst-case running time per input pattern, followed by SOFM and FMMC. Although FA, FMMC, and IAFC have similar ART-type control structures, FMMC incurs an overlap test and contraction procedure to eliminate hyperrectangle overlap, whereas IAFC incurs two membership measures (fuzzy and intra-cluster), and a supplementary category choice procedure based on Euclidean distances. SOFM processing is more straightforward than that of ART-type SONNs, since there is no category expansion test, nor search process. The selection of a winning map neuron has a complexity comparable to FMMC and IAFC choice functions, even though the number of map neurons, N', is usually greater than N (several map neurons may be needed to represent a single category with SOFM). However, the weight update step may involve several map neurons, instead of only one with ART-type SONNs. This m-learn strategy is compounded by the map interpretation post-processing.

The worst-case processing time as a function of the number of neurons N is an important consideration for implementation. As an example, suppose that one wishes to implement a SONN having $M = 16$ dimensions using a factor $F = 10$. Fig. 3 shows T/o_1 versus N for all four SONNs. The number of map neurons in SOFM is assumed to be three times the number of categories ($N' = 3N$). FA is clearly the fastest to process an input, followed by IAFC, SOFM and FMMC.

Another consideration is storage of weight values. FA and FMMC require $2MNb$ bits for storage of their weights. IAFC needs only half of that memory when input patterns are normalized. A memory size of $N'Mb$ bits is required with SOFM, but N' is normally greater than N (we have assumed a ratio of 3 to 1 so far).

Although this complexity analysis is based on a generic single processor random access machine, all these networks are suitable for parallel implementation using a multitude of VLSI architectures. The time complexity estimations obtained in our analysis then provide an indication of the semiconductor area required by the parallel hardware to meet a desired input pattern rate.

4.3 Discussion:

Simulation results on Radar data show that SOFM can yield excellent clustering scores if it is granted sufficient convergence time. By contrast, FA converges much faster, is computationally inexpensive, but yields lower scores than SOFM. In the case of FMMC and IAFC, the clustering quality is comparable to FA, but their processing requirements are significantly higher. These fundamental differences suggest that SOFM and FA could be attractive alternatives suitable for different radar ESM applications.

Consider the following example. The highest clustering score for the Radar data presented in the interleaved order is obtained using SOFM ($S_{RA}(A, R) = 0.95$). This is a very satisfactory score for radar ESM. However, SOFM converges very slowly and is computationally intensive. Assuming $M = 16$, $F = 10$ and $N' = 3 \cdot N = 30$ map neurons, it requires approximately 18000 elementary operations (T/o_1) for every input pattern. FA, on the other hand, achieves the second highest score for the interleaved radar data ($S_{RA}(A, R) = 0.80$). This is significantly lower than that of SOFM. But, assuming $M = 16$, $F = 10$ and $N = 10$ output neurons, FA requires just 2500 elementary operations per input pattern, and converges with almost 14 times fewer input pattern presentations than SOFM (3.6 epochs instead of 50).

Based on such numbers, SOFM could be well suited for radar ESM systems employed for long-range surveillance, intelligence, or targeting. In these tasks, accuracy prevails over timing: some delay can be tolerated in exchange for enhanced precision. Also, the high computational cost of SOFM may be less of a problem in shipborne or landbased installations, where equipment of some weight and size can be accommodated.

As for FA, it could better address the requirements of radar ESM systems used for threat alert. In such systems, reaction time must be minimized in order to engage timely protection measures against missiles, anti-aircraft artillery, intercept aircraft and other threats. The fast convergence of FA would thus be an indisputable asset. Also, the low computational complexity of FA would permit its use where space is scarce, and cost must remain low, like in small aircraft radar warning receivers.

5 Conclusion

A comparison of four self-organizing neural networks (SONNs) that are potentially suitable for high throughput clustering of radar pulses in ESM systems has been presented. These SONNs are the Fuzzy Adaptive Resonance Theory (FA), Fuzzy Min-Max Clustering (FMMC), Integrated Adaptive Fuzzy Clustering (IAFC), and Self-Organizing Feature Mapping (SOFM).

The SONNs have been compared from three standpoints — clustering quality, convergence time, and computational complexity. Computer simulations have been fed with a

Radar data set presented in several statistical orders. Convergence time has been measured by counting successive presentations of the data set prior to weight stabilization. Clustering quality has been assessed using the Rand Adjusted and Jaccard similarity measures.

Simulation results have shown that: (1) SOFM usually yields the best scores followed by FA, FMMC, and IAFC, (2) FMMC and FA converge the fastest, followed by IAFC and SOFM, (3) the data presentation order has a significant impact on the scores of ART-type networks, and their variability, and (4) SOFM, when granted a long convergence time, is insensitive to the data presentation order. The SONNs have then been studied from an algorithmic perspective to estimate worst-case running times, and consequently, computational complexity. Of the four SONNs, FA has been shown to have the lowest complexity, followed by IAFC, SOFM, and FMMC.

SOFM and FA emerge as suitable candidates for sorting pulses in radar ESM, albeit for different reasons. On the one hand, SOFM can achieve excellent clustering scores at the expense of a high complexity and potentially long convergence time. It would therefore be well suited for long range surveillance, intelligence and targeting. On the other hand, FA has the potential be to fast enough for use in threat alert systems. The selection of either SONN would ultimately depend on the specifics of the ESM application, and effect a tradeoff between clustering accuracy and computational efficiency.

Acknowledgments

This work was supported in part by the Defence Research Establishment Ottawa, the Canadian Microelectronics Corporation, le Fonds pour la Formation de Chercheurs et l'Aide la

Recherche (province of Quebec), and the Natural Sciences and Engineering Research Council of Canada.

References

[1] C. L. Davies and P. Hollands, "Automatic Processing for ESM," *Proc. IEE (UK)*, Vol. 129, Part. F, No. 3, pp. 164-171, June 1982.

[2] P. M. Grant and J. H. Collins, "Introduction to Electronic Warfare", *Proc. IEE (UK), Vol. 129, Part. F, No. 3, pp. 113-129, June 1982.*

[3] D. C. Schleher, Introduction to Electronic Warfare, Artech House, 1986.

[4] J. A. Anderson, M. T. Gately, P. A. Penz and D. R. Collins, "Radar Signal Categorization Using a Neural Network", *Proc. IEEE, Vol. 78, No. 10, pp. 1646-1656, 1990.*

[5] Behrooz Kamgar-Parsi, Behzad Kamgar-Parsi and J. C. Sciortino Jr., "Automatic Data Sorting Using Neural Network Techniques", Naval Research Laboratory Report NRL/FR/5720-96-9803, February 1996.

[6] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Approach to Machine Intelligence, Prentice-Hall, 1992.*

[7] G. A. Carpenter, S. Grossberg and D. B. Rosen, "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System", Neural Networks, Vol. 4, No. 6, pp. 759-771, 1991.

[8] P. Simpson, "Fuzzy Min-Max Neural Networks - Part2: Clustering", *IEEE Transactions on Fuzzy Systems, Vol. 1, No. 1, pp. 32-45, 1993.*

[9] Y. S. Kim and S. Mitra, "An Adaptive Integrated Fuzzy Clustering Model for Pattern Recognition", Fuzzy Sets and Systems, No. 65, pp. 297-310, 1994.

[10] T. Kohonen, *Self-Organization and Associative Memory, Springer-Verlag, 3rd edition, 1989.*

[11] L. Hubert and P. Arabie, "Comparing Partitions", Journal of Classification, Vol. 2, pp. 193-218, 1985.

[12] R. C. Dubes and A. K. Jain, *Algorithms for Clustering Data, Prentice Hall, 1988.*

[13] T. H. Corman, C. E. Leiserson and R. L. Rivest, Introduction to Algorithms, MIT Press, 1990.

[14] J. P. Y. Lee, "Wideband I/Q Demodulators: Measurement Technique and Matching Characteristics", *Proc. IEE: Radar, Sonar, Navigation, Vol. 143, No. 5, pp. 300-306, October 1996.*

[15] M. R. Anderberg, Cluster Analysis for Applications, Academic Press, 1973.

[16] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis, John Weiley, 1973.*

[17] J. C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, 1987.

[18] J. C. Bezdek, "Some Non-Standard Clustering Algorithms", *NATO ASI Series, Vol. G14, Developments in Numerical Ecology, P. Legendre and L. Legendre, Eds., Springer-Verlag, 1987.*

[19] *S. Haykin,* Neural Networks: A Comprehensive Foundation, Macmillan College Publishing Co., 1994.

[20] C.-T. Lin and C. S. George Lee, *Neural Fuzzy Systems: A Neuro-fuzzy Synergism to Intelligent Systems, Prentice-Hall, 1995.*

[21] *S. Grossberg, "Adaptive Pattern Classification and Universal Recoding: I. Parallel Development and Coding of Neural Detectors",* Biological Cybernetics, Vol. 23, pp. 121-134, 1976.

[22] S. Grossberg, "Adaptive Pattern Classification and Universal Recoding: II. Feedback, Oscillation, Olfaction, and Illusions", *Biological Cybernetics, Vol. 23, pp. 187-207, 1976.*

[23] *C. von der Malsberg, "Self-Organization of Orientation Sensitive Cells of the Striate Cortex",* Kybernetik, Vol. 14, pp. 85-100, 1973.

[24] D. E. Rumelhart and D. Zipser, "Feature Discovery by Competitive Learning", *Cognitive Science, Vol. 9, pp. 75-112, 1985.*

[25] *G. H. Ball and D. J. Hall, "ISODATA, an Iterative Method of Multivariate Analysis and Pattern Classification",* Proc. IFIPS Congress, 1965.

[26] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles, Addison-Wesley, 1974.*

[27] *G. A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine",* Computer, Vision, Graphics and Image Processing, Vol. 37, pp. 54-115, 1987.

[28] Y. S. Kim and S. Mitra, "Integrated Adaptive Fuzzy Clustering (IAFC) Algorithm", *Proc. 2nd Fuzz-IEEE, pp. 1264-1268, 1993.*

[29] *T. Kohonen, "Self-Organizing Formation of Topologically Correct Feature Maps",* Biological Cybernetics, Vol. 43, pp. 59-69, 1982.

[30] F. Murtagh, "Interpreting the Kohonen Self-Organizing Feature Map Using Contiguity-Constrained Clustering", *Pattern Recognition Letters, Vol. 16, pp. 399-408, 1995.*

[31] *S. Keyvan, L. C. Rabelo and A. Malkani, "Nuclear Reactor Condition Monitoring by Adaptive Resonance Theory",* Proc. IJCNN'92, Baltimore, USA, Vol. 3, pp. 321-8, 1992.

[32] Y. S. Kim and S. Mitra, "A Comparative Study of the Performance of Fuzzy ART-type Clustering Algorithms in Pattern Recognition", *Proc. SPIE: Intelligent Robots and Computer Vision XI, Boston, USA, pp. 335-341, November 1992.*

[33] *E. Pesonen, M. Eskelinen and M. Juhola, "Comparison of Different Neural Network Algorithms in the Diagnosis of Acute Appendicitis",* International Journal of Bio-Medical Computing, Vol. 40, Iss. 3, pp. 227-233, 1996.

[34] R. Peper, B. Zhang and H. Noda, "A Comparative Study of the ART2-A and the Self-Organizing Feature Map", *Proc. Int'l Joint Conf. on Neural Networks, pp. 1425-1429, 1993.*

[35] *C.-D. Wann and C. A. Thomopoulos, "Comparative Study of Self-Organizing Neural Networks",* Proc. IWANN'93, Sitges, Spain, pp. 316-21, June 1993.

[36] Z. Wang, A. Guerriero and M. De Sario, "Comparison of Several Approaches for the Segmentation of Texture Images", *Proc. SPIE: The Int'l Society of Optical Engineering, Vol. 2424, pp. 580-591, 1995.*

[37] *G. W. Milligan, S. C. Soon and L. M. Sokol, "The Effect of Cluster Size, Dimensionality, and the Number of Clusters on Recovery of True Cluster Structure",* IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 5, pp. 40-47, 1983.

[38] R. A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems", *Annals of Eugenics, Vol. 7, 1936.*

[39] S. Lawrence, C. L. Giles, A. C. Tsoi and A. D. Back, " Face Recognition: A Convolutional Neural-Network Approach", IEEE Trans. on Neural Networks, Vol. 8, No. 1, pp. 98-113, 1997.

[40] B. Moore, "ART1 and Pattern Clustering", *Proc. 1988 Connectionist Models Summer School, Touretski, Hinton and Sejnowski, Eds., Morgan Kaufmann Publishers, pp. 174-185, 1988.*

A Simulation Results for the Wine Recognition and Iris Data Sets

The *Wine Recognition* data set[3] is the result of a chemical analysis of wines from 3 different cultivars. The problem consists in determining the origins of 178 wine samples, where 13 attributes characterize the constituents found in each sample. The well-known *Iris* data set [38] contains 150 flowers belonging to 3 species of iris flowers: satosa, versicolor and virginica. Each species contains 50 flowers that are characterized by 4 features: petal and sepal length and width.

Simulation results for the Wine Recognition and Iris data sets are given in Tables 3 and 4, respectively. As with the Radar data, SOFM most often achieves the highest SONN scores, usually followed by FA. The only exceptions are found when the Wine and Iris data are presented in the interleaved order, in which case FA scores the best. IAFC performs almost as well as FMMC for the Wine data, and surpasses FMMC for the Iris data. In this last case, IAFC almost ties FA for second best. Overall, IAFC appears to be more sensitive than the others to the distribution of the patterns in input space.

It is worth noting that the gap between SOFM scores and those of the other three SONNs is not as wide as with the Radar data. Besides, the three ART-type networks usually perform better when data are presented in more structured, radar-like orders. This holds true to the extent that FA scores slightly higher than ISODATA for Iris data presented in the interleaved order.

The SONN convergence times appear to vary in a proportional manner from one data

[3]This set was obtained from the UCI Machine Learning Repository (www.uci.edu/~mlearn).

set to the next: FMMC always converges with the least amount of epochs, followed closely by FA, then by IAFC and SOFM. On average, FA and FMMC converge slightly faster for the Iris data than with the Wine data, and faster with the Wine data than the Radar data. IAFC and SOFM converge in much fewer epochs with the Wine data than either the Iris or Radar data. This discrepancy may be due to a sensitivity to data structures with overlapping clusters. This is especially true for IAFC, whose learning rate, λ, scales according to fuzzy and intra-cluster membership functions [9]. Notice that its convergence for the Iris data requires more epochs than for the Radar data set. In any case, there seems to be little correlation between convergence time and presentation order.

B Analysis of Computational Complexity

In this appendix, details of a computational complexity analysis are presented[4] for each SONN algorithm. First, the maximum time required to execute every operation of type p is derived ($T_p = o_p \cdot n_p$). Table 5 gives a breakdown of time complexities, organized according to algorithm steps. The time complexity of a step is equal to the product of its cost per iteration and the number of iterations it needs. Notice how the values for some of these steps are obtained by adding the contribution of several sub-steps (shown directly below). Then, the sum of these time complexities yields a total worst-case running time per input pattern, $T = \sum_p T_p$, and a corresponding growth rate. The resulting complexity polynomial is expressed in a normalized format, T/o_1, in terms of network parameters M, N and F, and provides an estimate of computational complexity. The following cost values o_p are used throughout the analysis:

- o_1: the time required to execute elementary, low-level operations (i.e., $x + y$, $x - y$, $\max(x, y)$, $1 - x$, $x < y$, etc.). The values x and y are real numbers represented with a b bit resolution.

- $o_{2,q}$: the time needed to compute a division x/y ($o_{2,1}$), a multiplication $x \cdot y$ ($o_{2,2}$), a square root $\sqrt{x}$ ($o_{2,3}$), or an exponent e^x ($o_{2,4}$). Notice that operations of type $p = 2$ are subdivided into operations of type $q = \{1, 2, 3, 4\}$ on account for their dissimilar complexities. The complexity of these operations is a function of b ($o_{2,q} = o_{2,q}(b)$), and they differ according to specific implementations. However, in order to ease the

[4] A similar type of analysis has been conducted by Lawrence et al. [39] to approximate the computational complexity of various tasks done by a face recognition system. This type of analysis is important in the context of implementing real-time systems, where complexity scales according to system parameters.

analysis and the comparison of results, we assume that $o_{2,1} \simeq o_{2,2} \simeq o_{2,3} \simeq o_{2,4} \simeq F(b) \cdot o_1 = F \cdot o_1$, where parameter F reflects the use of higher complexity operations in the analysis. This is a reasonable assumption if dedicated hardware is available to support the requested operations.

As explained previously, this complexity analysis assumes that the SONN algorithms are implemented using a random access machine (RAM) model of computation [13]. They are also supposed to be executed in an order permitting efficient reuse of data (e.g., no values are computed twice). Input patterns are considered to be normalized, and all weight and parameter values are presumed to be initialized prior to the presentation of any data. The time associated with these last two operations is ignored since it is comparable for all the SONNs. The SONNs form clusters progressively by sequential processing of input data patterns. ISODATA is an iterative procedure that requires all the input data at once to form a predefined number of clusters. Since ISODATA cannot be analyzed in the same way as the SONNs, it is omitted from the complexity analysis.

B.1 Fuzzy ART (FA):

In Table 5, the breakdown of time complexities shows that operation B, the computation of the output neuron choice functions, is potentially the most costly with FA. The sequence of the FA search process (operation C) has been modified according to Subsection 2.2. The vigilance test is computed only once for all N output neurons prior to category choice. Similar modifications to FMMC and IAFC were also assumed. The total worst-case running time

T_{FA} corresponds to the sum of the elements multiplied in the "Time complexity" column:

$$\begin{aligned} T_{FA} &= o_1 M + N[o_1(6M+1) + o_{2,1}] + 2o_1 N + 4M(o_1 + o_{2,2}) \\ &= NM(6o_1) + N(3o_1 + o_{2,1}) + M(5o_1 + 4o_{2,2}) \\ &= o_1(6NM + 3N + 5M) + o_{2,1}(N) + o_{2,2}(4M) \end{aligned}$$

If parameter F is comparable for all type $p = 2$ operations, $o_{2,1} \simeq o_{2,2} \simeq F \cdot o_1$, then:

$$T_{FA} = o_1 \cdot [6NM + NF + 4MF + 3N + 5M] \tag{4}$$

The rate of growth of T_{FA} when F is constant and $M, N \gg 1$ is $O(NM)$.

B.2 Fuzzy Min-Max Clustering (FMMC):

As with FA, the computation of output neuron membership functions (operation A) is potentially very demanding. FMMC's simple hyperrectangle expansion procedure (operation C) is equivalent to FA's fast learning rule (with $\beta = 1$). However, its combination of overlap test and contraction procedure (operations D and E), are also notably time consuming. The total worst-case running time T_{FMMC} is:

$$\begin{aligned} T_{FMMC} &= N[M(9o_1 + 2o_{2,2}) + o_{2,1}] + N[3o_1 M + 2o_1 + o_{2,1}] + 2o_1 M + \\ &\quad (N-1)12o_1 M + (N-1)[M(o_1 + o_{2,1})] \\ &= NM(25o_1 + o_{2,1} + 2o_{2,2}) + 2N(o_1 + o_{2,1}) - M(11o_1 + o_{2,1}) \\ &= o_1(25NM + 2N - 11M) + o_{2,1}(NM + 2N - M) + o_{2,2}(2NM) \end{aligned}$$

If parameter F is comparable for all type $p = 2$ operations, $o_{2,1} \simeq o_{2,2} \simeq F \cdot o_1$, then:

$$T_{FMMC} = o_1 \cdot [3NMF + 25NM + 2NF - MF + 2N - 11M] \tag{5}$$

The rate of growth of T_{FMMC} when F is constant and $M, N \gg 1$ is $O(NM)$.

B.3 Integrated Adaptive Fuzzy Clustering (IAFC):

The computation of the Euclidean distances and related values in operation A (between input $\vec{a}$ and the prototype vectors $\vec{v}_j$) is potentially very time consuming. These values are needed at several instances in IAFC's processing. For example, fuzzy membership values are needed for an alternate choice procedure (see operation C.2). The complexity required for the choice functions and the search process is comparable to that of FMMC. Its centroid update (operation D) is as costly as the prototype vector update of FA. The total worst-case running time T_{IAFC} is:

$$\begin{aligned} T_{IAFC} &= N[M(o_1 + o_{2,1}) + o_1 + o_{2,1} + o_{2,2} + o_{2,3}] + N[o_{2,2}M] + N[4o_1 + 2o_{2,1} + o_{2,2} + o_{2,4}] + \\ &\quad 2o_{2,2}M + 9o_1 + 3o_{2,1} + 6o_{2,2} \\ &= NM(o_1 + o_{2,1} + o_{2,2}) + N(5o_1 + 3o_{2,1} + 2o_{2,2} + o_{2,3} + o_{2,4}) + M(2o_{2,2}) + 9o_1 + 3o_{2,1} + 6o_{2,2} \\ &= o_1(NM + 5N + 9) + o_{2,1}(NM + 3N + 3) + o_{2,2}(NM + 2N + 2M + 6) + o_{2,3}(N) + o_{2,4}(N) \end{aligned}$$

If parameter F is comparable for all type $p = 2$ operations, $o_{2,1} \simeq o_{2,2} \simeq o_{2,3} \simeq o_{2,4} \simeq F \cdot o_1$, then:

$$T_{IAFC} = o_1 \cdot [2NMF + NM + 7NF + 2MF + 5N + 9F + 9] \tag{6}$$

The rate of growth of T_{IAFC} when F is constant and $M, N \gg 1$ is $O(NM)$.

B.4 Self-Organizing Feature Mapping (SOFM):

The SOFM processing is more straightforward than that of the ART-type SONNs, since there is no search process. Most of the workload is concentrated in 3 operations: A, B and D. The selection of a winning map neuron has a complexity comparable to the FMMC and IAFC choice functions, even though N' is usually greater than N (several map neurons may be needed to represent a single category with SOFM). Notice also the weight update required for several map neurons, instead of just one with ART-type SONNs. The total worst-case running time T_{SOFM} is:

$$\begin{aligned}
T_{SOFM} &= N'[M(o_1 + o_{2,1}) + 2o_1 + o_{2,3}] + N'[M(2o_1 + o_{2,2}) + 3o_1 + 2o_{2,1} + o_{2,3}] + \\
&\quad 2(o_1 + o_{2,1} + o_{2,2}) + N'[M(o_1 + o_{2,1}) + o_1 + o_{2,3}] - M(o_1 + o_{2,1}) + o_1 - o_{2,3} \\
&= N'M(4o_1 + 2o_{2,1} + o_{2,2}) + N'(6o_1 + 2o_{2,1} + 3o_{2,3}) - M(o_1 + o_{2,1}) + 3o_1 + 2o_{2,1} + 2o_{2,2} - o_{2,3} \\
&= o_1(4N'M + 6N' - M + 3) + o_{2,1}(2N'M + 2N' - M + 2) + o_{2,2}(N'M + 2) + o_{2,3}(3N' - 1)
\end{aligned}$$

If parameter F is comparable for all type $p = 2$ operations, $o_{2,1} \simeq o_{2,2} \simeq o_{2,3} \simeq F \cdot o_1$, then:

$$T_{SOFM} = o_1 \cdot [3N'MF + 4N'M + 5N'F - MF + 6N' - M + 3F + 3] \tag{7}$$

The rate of growth of T_{SOFM} when F is constant and $M, N' \gg 1$ is $O(N'M)$.

C Summary of SONN Algorithms

C.1 Fuzzy ART (FA):

The FA's functionality can be described as a five steps algorithm:

1. Weights and parameters initialization: Initially, all the neurons of F2 are uncommitted, and all weight values w_{ji} are initialized to 1. An F2 neuron becomes committed when it is selected for an input $\vec{a}$, then the corresponding weights w_{ji} can take real values in the interval [0,1].

2. Input vector complement coding: When a new M-element input vector $\vec{a} = (a_1, a_2, ..., a_M)$ (where each element a_i is a real number in the interval [0,1]) is presented to the network, it undergoes a preliminary *complement coding* [7]. This gives a network input vector $\vec{I}$ of $2M$ elements such that: $\vec{I} = (\vec{a}; \vec{a}^c) = (a_1, a_2, ..., a_M; a_1^c, a_2^c, ..., a_M^c)$, with $a_i^c = 1 - a_i$. This coding is recommended to prevent a category proliferation problem that may occur in analog ART networks [40].

3. Category choice: With the presentation of an input $\vec{I}$ to F1, the *choice function* $T_j(\vec{I})$ is calculated for each neuron j in F2:

$$T_j(\vec{I}) = \frac{|\vec{I} \wedge \vec{w}_j|}{\alpha + |\vec{w}_j|} \tag{8}$$

where $|\cdot|$ is the norm operator ($|\vec{x}| = \sum_{i=1}^{2M} |x_i|$), $\wedge$ is the fuzzy logic AND operator ($\vec{I} \wedge \vec{w}_j = (\min(I_1, w_{j,1}), \min(I_2, w_{j,2}), ..., \min(I_{2M}, w_{j,2M}))$), and α is a user-defined *choice parameter* such that $\alpha > 0$. F2 is a winner-take-all competitive layer, where the winner is the neuron $j = J$ with the greatest value of activation T_j for the input $\vec{I}$ ($T_J = \max\{T_j : j = 1...N\}$).

If the same T_J value is obtained by two or more neurons, the one with the smallest index j wins. The winning neuron J is retained for Step 4.

4. Vigilance test: This step serves to compare the similarity between the prototype vector of the winning neuron $\vec{w}_J$ and input $\vec{I}$, against a user-defined *vigilance parameter* ρ, through the following test:

$$\frac{|\vec{I} \wedge \vec{w}_J|}{|\vec{I}|} \geq \rho \tag{9}$$

where $\rho = [0, 1]$. This comparison is carried out on layer F1: the winning neuron J transmits its learned expectation, $|\vec{I} \wedge \vec{w}_J|$, to F1 for comparison with $\vec{I}$. If the vigilance test (Eq. 9) is passed, then J becomes *selected*, and it is allowed to adapt its prototype vector (Step 5). Otherwise, J is deactivated for $\vec{I}$: T_J is set equal to -1 for the duration of the current input presentation. The algorithm searches through the remaining F2 layer neurons (Steps 3 and 4), until some other neuron J passes the vigilance test. If no committed neuron from the F2 layer can pass this test, an uncommitted F2 neuron is selected to undergo prototype vector update, and becomes committed.

5. Prototype vector update: The prototype vector of the winning neuron J is updated according to:

$$\vec{w}'_J = \beta(\vec{I} \wedge \vec{w}_J) + (1 - \beta)\vec{w}_J \tag{10}$$

where β is a user-defined *learning rate parameter* such that $\beta = [0, 1]$. The algorithm can be set to slow learning, with $0 < \beta < 1$, or to fast learning, with $\beta = 1$. Once this update step is accomplished, the network reactivates all F2 neurons, and can process a new input vector.

C.2 Fuzzy Min-Max Clustering (FMMC):

The FMMC network's algorithm may be described with four steps:

1. Category hyperrectangle and parameter initialization: All the category neurons of F2 are considered to be uncommitted, and all hyperrectangles B_j are initialized such that $\vec{u}_j = \vec{1}$ and $\vec{v}_j = \vec{0}$.

2. Hyperrectangle expansion: Let $B_j = \{\vec{a}, \vec{u}_j, \vec{v}_j, b_j\}$ define the category hyperrectangle fuzzy set for F2 neuron j, where $\vec{a} = (a_1, a_2, ..., a_M)$ is an input pattern, and $b_j = b_j(\vec{a}, \vec{u}_j, \vec{v}_j) \in [0, 1]$ is a membership function defined by:

$$b_j(\vec{a}, \vec{u}_j, \vec{v}_j) = \frac{1}{M} \sum_{k=1}^{M} [1 - f(a_k - v_{jk}, \gamma) - f(u_{jk} - a_k, \gamma)] \tag{11}$$

The ramp threshold function $f(\cdot)$ is defined such that $f(x, \gamma) = 1$ if $x \cdot \gamma > 1$, $f(x, \gamma) = x \cdot \gamma$ if $0 \leq x \cdot \gamma \leq 1$, and $f(x, \gamma) = 0$ if $x \cdot \gamma < 0$. When a new input vector $\vec{a} = (a_1, a_2, ..., a_M)$ is presented to the network, function b_j measures the degree to which it belongs to hyperrectangle j. The closer $\vec{a}$ is to the hyperrectangle points, the closer b_j is to 1, subject to a *sensitivity parameter* γ. For a set of committed category neurons in F2, the closest hyperrectangle B_J that can expand (if needed) to include $\vec{a}$ is identified. The winner $j = J$ is the category in which $\vec{a}$ yields the greatest degree of membership ($b_J = \max\{b_j : j = 1...N\}$). This selected category hyperrectangle is then submitted to an expansion test:

$$\frac{1}{M} \sum_{k=1}^{M} [\max(v_{Jk}, a_k) - \min(u_{Jk}, a_k)] \leq \theta \tag{12}$$

where $\theta \in [0,1]$ is a user-defined *maximum hyperrectangle size* parameter. If this last constraint is met, then expansion learning takes place: the min and max points of the selected category hyperrectangle are adjusted so that $u'_{Jk} = \min(u_{Jk}, a_k)$ and $v'_{Jk} = \max(v_{Jk}, a_k)$ for $k = 1, 2, ..., M$. Otherwise, alternate committed category hyperrectangles may be selected. If all else fails, a new category hyperrectangle is committed, and is assigned $\vec{a}$.

3. Overlap test: A hyperrectangle represents the portion of a category fuzzy set to which the corresponding patterns have full membership. Hyperrectangles are not allowed to overlap, although the portions of the fuzzy sets with partial membership do overlap. After each hyperrectangle expansion process, the overlap of B_J is quantified with respect to the other existing hyperrectangles B_h ($h = 1, 2, ..., N, h \neq J$) for the following 4 cases, along each of the M dimensions with (a): $v_{hk} > v_{Jk} > u_{hk} > u_{Jk}$, (b): $v_{Jk} > v_{hk} > u_{Jk} > u_{hk}$ (for partial hyperrectangle overlap), (c): $v_{Jk} > v_{hk} \geq u_{hk} > u_{Jk}$, and (d): $v_{hk} > v_{Jk} \geq u_{Jk} > u_{hk}$ (for full hyperrectangle overlap), where $k = 1, 2, ..., M$.

4. Hyperrectangle contraction: Overlap between the selected hyperrectangle B_J and another one, B_h, is eliminated using a contraction process on a dimension-by-dimension basis. For the 4 cases described in Step 3, the overlap is eliminated with (a): $u'_{hk} = v'_{Jk} = (u_{hk} + v_{Jk})/2$, (b): $u'_{Jk} = v'_{hk} = (u_{Jk} + v_{hk})/2$, (c): $u'_{Jk} = v_{hk}$, (d): $v'_{Jk} = u_{hk}$, for $k = 1, 2, ..., M$. Then, the network reactivates all F2 neurons, and is ready to repeat steps 2 to 4 with a new input pattern.

C.3 Integrated Adaptive Fuzzy Clustering (IAFC):

The IAFC network's algorithm can be summarized with the following four step procedure:

1. Weights and parameters initialization: Initially, all the neurons of F2 are uncommitted, and all weight values v_{ji} and b_{ji} are initialized to 0. An F2 neuron becomes committed when it is selected for an input $\vec{a}$, then the corresponding weights v_{ji} can take real values greater than 0. The weights b_{ji} are always normalized versions of the v_{ji}.

2. Category choice: When a new input pattern $\vec{a} = (a_1, a_2, ..., a_M)$ is presented, it is normalized. The resulting input pattern $\vec{I} = (I_1, I_2, ..., I_M)$ is applied to the network's F1 neurons, through bottom up weights b_j, and a winner-take-all competition occurs among F2 neurons. The dot product Y_j is found for each F2 neuron j:

$$Y_j = \sum_{i=1}^{M} I_i \cdot b_{ji} = \vec{I} \cdot \vec{b}_j^T \tag{13}$$

The output neuron $j = J$ that receives the largest activation is selected ($Y_J = \max\{Y_j : j = 1, 2, ..., N\}$). By using this correlation-type choice function, the winner is decided by the angle between $\vec{I}$ the $\vec{b}_j$s. This may cause misclassification since the input-centroid distance is not considered. In such cases, the Euclidean distance may be a better alternative for category choice. For a category choice, IAFC uses the following combined similarity measure. After using Eq. 13, and selecting a winner J, the fuzzy membership value μ_J of the input pattern in cluster J is computed:

$$\mu_J = \frac{\left(\frac{1}{||\vec{a}-\vec{v}_J||^2}\right)^{\frac{1}{m-1}}}{\sum_{j=1}^{N}\left(\frac{1}{||\vec{a}-\vec{v}_j||^2}\right)^{\frac{1}{m-1}}} \tag{14}$$

where $m \in [1, \infty)$ is the weight exponent (experimentally set to 2 [9]), and $||\cdot||$ is the Euclidean distance. If μ_J is less than σ, the algorithm activates an alternate procedure to find a winner: a cluster whose centroid $\vec{v}_j$ is closest to $\vec{a}$ according to the Euclidean distance ($Y'_J = \min\{|||\vec{a} - \vec{v}_j|| : j = 1, 2, ..., N\}$). Therefore, if σ is set low, the angle between $\vec{I}$ and the $\vec{b}_j$s becomes dominant in the choice of a winner, otherwise the Euclidean distance is dominant in this choice.

3. Vigilance test: The choice of winning cluster J is validated according to the following test:

$$e^{-\gamma \cdot \mu_J} \cdot ||\vec{a} - \vec{v}_J|| \leq \tau \tag{15}$$

where γ is a factor that controls the shape of clusters formed (normally set to 1 [9]), and τ is the *vigilance parameter*. This similarity measure considers the Euclidean distance between $\vec{a}$ and $\vec{v}_J$, and the relative degree of belonging of $\vec{a}$ to every committed category. If the winning J satisfies the test, this cluster's centroid is updated (Step 4). Otherwise, neuron J is deactivated, and another cluster is selected as winner (Steps 2 and 3). If no cluster can pass the vigilance test, an uncommitted F2 neuron is selected for centroid update.

4. Update the winning centroids: The centroid of the selected category is updated according to:

$$\vec{v}'_J = (1 - \lambda) \cdot \vec{v}_J + \lambda \cdot \vec{a} \tag{16}$$

where $\lambda = f(l) \cdot \pi(\vec{a}, \vec{v}_J, \tau) \cdot \mu_J^2$. The value l is the number of iterations of the data set, $f(l) = 1/\{k(l-1)+1\}$ is a decreasing function of l, and $\pi(\vec{a}, \vec{v}_J, \tau)$ is the intra-cluster membership value of $\vec{a}$ to the winner J, defined by:

$$\pi(\vec{a}, \vec{v}_J, \tau) = \begin{cases} 1 - 2(||\vec{a} - \vec{v}_J||/\tau)^2 & \text{if } 0 \leq ||\vec{a} - \vec{v}_J|| \leq \tau/2, \\ 2(1 - ||\vec{a} - \vec{v}_J||/\tau)^2 & \text{if } \tau/2 \leq ||\vec{a} - \vec{v}_J|| \leq \tau, \\ 0 & \text{if } ||\vec{a} - \vec{v}_J|| \geq \tau \end{cases}$$

A normalized version of $\vec{v}'_J$ ($\vec{b}'_J$) is also calculated for subsequent processing. The previously deactivated output neurons are enabled, and a new input pattern is accepted at Step 2.

C.4 Self-Organizing Feature Mapping (SOFM):

The SOFM algorithm used in this paper is summarized in terms of the following 4 steps:

1. Weights and parameter initialization: Weights $\vec{w}_j(0) = (w_{j1}(0), w_{j2}(0), ..., w_{jM}(0))$ from each map neuron to all the input neurons are initialized to small random numbers. The initial learning rate $\alpha(0)$ and neighborhood radius $r(0)$ are also set prior to training. For every neuron j of the map, the *assigned input count* $c(j)$ is reset to 0.

2. Map neuron selection: At time t, the input pattern $\vec{a}(t) = (a_1(t), a_2(t), ..., a_M(t))$ is compared with each map neuron prototype vector $\vec{w}_j(t)$ using the Euclidean distance $||\vec{a}(t) - \vec{w}_j(t)||$. The best-matching map neuron J, the one that minimizes the Euclidean distance, is selected according to:

$$J = \arg\min\{||\vec{a}(t) - \vec{w}_j(t)|| : j = 1, 2, ..., N\} \tag{17}$$

The assigned input count of the best-match map neuron is incremented ($c(J) = c(J) + 1$).

3. Weights updating: The input $\vec{a}(t)$ is mapped onto the neuron J relative to all other $\vec{w}_j(t)$ values. During learning, the neurons that are topographically close to neuron J's location in the map (within a neighborhood) activate each other to learn from the input. Weights connected to J and to other neurons in its neighborhood become more alike through learning [30], thus more responsive to $\vec{a}(t)$. They are adjusted according to the learning rule:

$$\vec{w}_j(t+1) = \vec{w}_j(t) + n_{Jj}(t)[\vec{a}(t) - \vec{w}_j(t)] \tag{18}$$

where $n_{Jj}(t)$ is the neighborhood *kernel* that is a function defined over time, and the map neurons. The kind of kernel used in this paper is named *bubble*, whose definition considers a set of map neurons neighboring the best-match neuron J. This set is denoted $N_J(t)$, and it contains the set of map neurons considered to be in the neighborhood of map neuron J at time t. Then, $n_{Jj}(t) = \alpha(t)$ if neuron j belongs to $N_J(t)$, and $n_{Jj}(t) = 0$ otherwise, where $\alpha(t)$ is the *learning rate* parameter $(0 < \alpha(t) < 1)$. Specifically, the set $N_J(t)$ is a function of $d_{Jj} = |r_J - r_j|$, the lateral map distance of neuron J to neuron j, which is a Euclidean measure in the R^2 output space. The size of the set $N_J(t)$ is regulated by the threshold parameter $r(t)$. If $d_{Jj} \leq r(t)$, the map neuron j belongs to J's neighborhood, and it is allowed to learn from $a(t)$. Then, parameters $\alpha(t)$ and $r(t)$ are adjusted: they are progressively decreased in a slow and linear way from their initial values, $r(0)$ and $\alpha(0)$, to 1 and 0, respectively. If $\alpha(t) \geq 0.5 \cdot \alpha(0)$, then the feature map may be interpreted (Step 4). [?? This can anso be one or a few passes of the data set.]

4. Map interpretation: Every map neuron j whose assigned input count $c(j)$ is greater than ζ, a user-defined *category assignment* parameter, is interpreted as a category center. If $c(J)$ is smaller, or equal to ζ, find the closest category center K according to:

$$K = \arg\min_k \{||\vec{w}_k(t+1) - \vec{w}_J(t+1)||\} \tag{19}$$

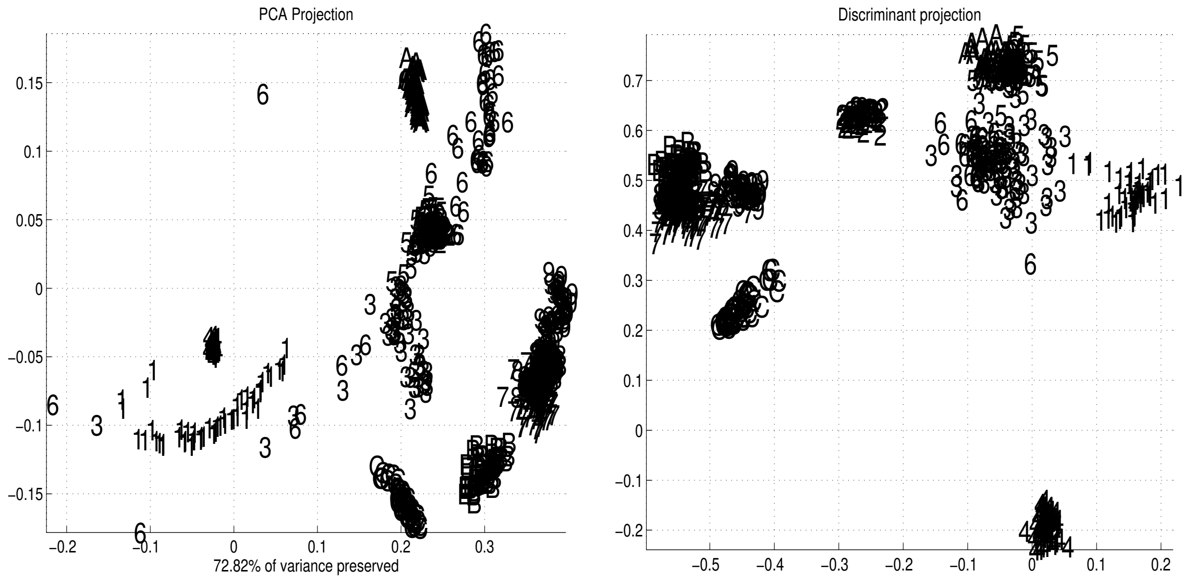
At this stage in training, this search may be bound to J's adjacent map neurons, given the topographical relationships. Then, input $\vec{a}(t)$ is implicitly assigned to K without additional weight updating. The integer discrete-time coordinate t is incremented, and Steps 2 to 4 are repeated with data set patterns until $\alpha(t)$ is reduced to 0, and the weight values are fixed, at which time a feature map has been formed.

$12c_{11}$ $12c_{12}$

$12c_{21}$ $12c_{22}$

		Clustering A	
		Same	Different
Clustering B	Same		
	Different		

Figure 1: Contingency table used to compare two clusterings.



(a) Principle component analysis (PCA)

(b) Linear discriminant analysis (LDA)

Figure 2: Two-dimensional projections for the Radar data set.

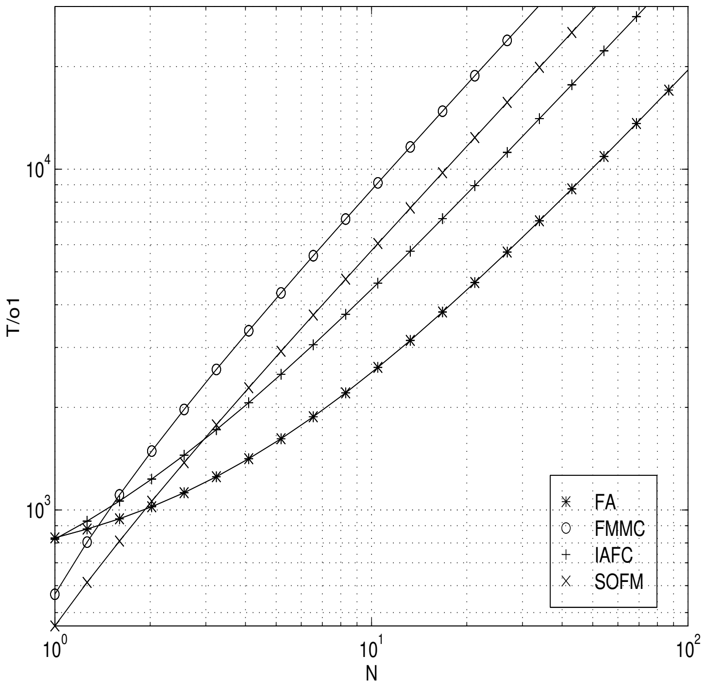


Figure 3: Worst-case running time (T/o_1) versus N, assuming that $M = 16$ and $F = 10$. The number of map neurons in SOFM is set to $N' = 3 \cdot N$.

Table 1: Summary of Best Simulation Results for the Radar Data Set.

ALGORITHM: **(parameter values)** · *presentation order*	**PERFORMANCE MEASURES** **Clustering quality** $S_{RA}(A,R)$ Mean	Std. dev.	$S_J(A,R)$ Mean	Std. dev.	**Conv. time** Number of epochs
ISODATA: $(2 \leq L \leq 4; \forall N_c; .1 \leq \theta_c \leq .2; .1 \leq \theta_s \leq .5; 14 \leq \theta_N \leq 22)$					
· *random*	0.93	0.122	0.99	0.043	$I \geq 15$
· *burst*	0.96	0.124	0.99	0.041	$I \geq 15$
· *interleaved*	0.96	0.163	0.99	0.062	$I \geq 15$
Fuzzy ART (FA): $(.85 \leq \rho \leq .95; \beta = 1, \alpha = 1)$					
· *random*	0.61	0.056	0.93	0.041	3.8
· *burst*	0.59	0.109	0.93	0.049	3.2
· *interleaved*	0.80	0.130	0.96	0.050	3.6
Fuzzy Min-Max Clustering (FMMC): $(\gamma = 5; .01 \leq \theta \leq .25)$					
· *random*	0.45	0.062	0.92	0.038	2.2
· *burst*	0.67	0.064	0.94	0.062	2.0
· *interleaved*	0.72	0.099	0.95	0.067	2.0
Integrated Adaptive Fuzzy Clustering (IAFC): $(\gamma = 1;\ m = 2;\ 0 \leq \sigma \leq .6;\ .1 \leq k \leq 1;\ .5 \leq \tau \leq .6)$					
· *random*	0.62	0.002	0.88	0.001	17.6
· *burst*	0.63	0.060	0.89	0.025	20.9
· *interleaved*	0.66	0.046	0.90	0.021	21.2
Self-Organizing Feature Mapping (SOFM): $(5 \times 5 \text{ map}; 2 \leq \zeta \leq 16; .03 \leq \alpha(0) \leq .5; 2 \leq r(0) \leq 3)$					
· *random*	0.95	0	0.99	0	50
· *burst*	0.95	0	0.99	0	50
· *interleaved*	0.95	0	0.99	0	50

Table 2: Summary of the Computational Complexity Estimations

SONN:	**COMPUTATIONAL COMPLEXITY**	
	Worst-case running time (T/o_1)	**Growth rate**
FA:	$6NM + NF + 4MF + 3N + 5M$	$O(NM)$
FMMC:	$3NMF + 25NM + 2NF - MF + 2N - 11M$	$O(NM)$
IAFC:	$2NMF + NM + 7NF + 2MF + 5N + 9F + 9$	$O(NM)$
SOFM:	$3N'MF + 4N'M + 5N'F - MF + 6N' - M + 3F + 3$	$O(N'M)$

Table 3: Summary of Best Simulation Results for the Wine Recognition Data Set.

ALGORITHM: **(parameter values)** · *presentation order*	**PERFORMANCE MEASURES** **Clustering quality** $S_{RA}(A,R)$ Mean	$S_{RA}(A,R)$ Std. dev.	$S_J(A,R)$ Mean	$S_J(A,R)$ Std. dev.	**Conv. time** Number of epochs
ISODATA: $(L=1; N_c \geq 5; .3 \leq \theta_c \leq .6; .1 \leq \theta_s \leq .4; \theta_N \geq 8)$					
· *random*	0.70	0.060	0.84	0.021	$I \geq 4$
· *burst*	0.76	0.095	0.87	0.032	$I \geq 4$
· *interleaved*	0.76	0.099	0.87	0.031	$I \geq 4$
Fuzzy ART (FA): $(\beta = 1; \alpha = 1; .35 \leq \rho \leq .5)$					
· *random*	0.26	0.056	0.68	0.013	2.9
· *burst*	0.29	0.077	0.70	0.020	3.0
· *interleaved*	0.49	0.110	0.77	0.031	2.8
Fuzzy Min-Max Clustering (FMMC): $(4 \leq \gamma \leq 5; .6 \leq \theta \leq .75)$					
· *random*	0.07	0.012	0.67	0.001	2.1
· *burst*	0.18	0.013	0.69	0.004	2.0
· *interleaved*	0.32	0.112	0.71	0.031	2.0
Integrated Adaptive Fuzzy Clustering (IAFC): $(\gamma = 1;\ m = 2;\ .6 \leq \sigma \leq .9;\ .1 \leq k \leq .8;\ .4 \leq \tau \leq .6)$					
· *random*	0.13	0.013	0.68	0.003	6.5
· *burst*	0.13	0.009	0.68	0.002	7.6
· *interleaved*	0.18	0.009	0.70	0.002	6.8
Self-Organizing Feature Mapping (SOFM): $(3 \times 3 \text{ map}; 2 \leq \zeta \leq 11; .01 \leq \alpha(0) \leq 1; 1 \leq r(0) \leq 2)$					
· *random*	0.48	0	0.73	0	14
· *burst*	0.50	0	0.73	0	14
· *interleaved*	0.50	0	0.73	0	14

Table 4: Summary of Best Simulation Results for the Iris Data Set.

ALGORITHM: **(parameter values)** · *presentation order*	**PERFORMANCE MEASURES** **Clustering quality** $S_{RA}(A,R)$ Mean	Std. dev.	$S_J(A,R)$ Mean	Std. dev.	**Conv. time** Number of epochs
ISODATA: $(1 \leq L \leq 3; \forall N_c; 1 \leq \theta_c \leq 1.5; \theta_s < 1.2; 15 \leq \theta_N \leq 22)$					
· *random*	0.72	0.054	0.83	0.030	$I \geq 6$
· *burst*	0.72	0.063	0.84	0.035	$I \geq 6$
· *interleaved*	0.73	0.068	0.89	0.036	$I \geq 6$
Fuzzy ART (FA): $(.7 \leq \rho \leq .75;\ \alpha = 1;\ \beta = 1)$					
· *random*	0.68	0.077	0.82	0.042	2.3
· *burst*	0.68	0.085	0.83	0.047	2.6
· *interleaved*	0.75	0.109	0.86	0.042	1.8
Fuzzy Min-Max Clustering (FMMC): $(4 \leq \gamma \leq 5; .4 \leq \gamma \leq .55)$					
· *random*	0.53	0.078	0.78	0.034	1.8
· *burst*	0.60	0.103	0.79	0.036	2.1
· *interleaved*	0.68	0.218	0.82	0.045	1.6
Integrated Adaptive Fuzzy Clustering (IAFC): $(\gamma = 1;\ m = 2;\ .2 \leq \sigma \leq .5;\ .3 \leq k \leq .7;\ .8 \leq \tau \leq 1)$					
· *random*	0.68	0.046	0.81	0.019	26.3
· *burst*	0.68	0.016	0.83	0.009	24.7
· *interleaved*	0.72	0.033	0.84	0.012	23.0
Self-Organizing Feature Mapping (SOFM): $(3 \times 3 \text{ map}; 7 \leq \zeta \leq 12; .07 \leq \alpha(0) \leq .3; 1 \leq r(0) \leq 3)$					
· *random*	0.69	0	0.84	0	45
· *burst*	0.69	0	0.84	0	45
· *interleaved*	0.69	0	0.84	0	45

Table 5: Breakdown of Time Complexities.

SONN	**TIME COMPLEXITY**	
Steps of the algorithm:	**Cost per iteration**	**Number of iterations**
FA		
A. Complement coding:	$o_1 M$	1
B. Choice functions:	$o_1(6M+1)+o_{2,1}$	N
C. Search process:		
C.1. vigilance test	o_1	N
C.2. category choice	o_1	N
D. Prototype vector update:	$4M(o_1+o_{2,2})$	1
FMMC		
A. Membership functions:	$M(9o_1+2o_{2,2})+o_{2,1}$	N
B. Search process:		
B.1. expansion test	$3o_1 M+o_1+o_{2,1}$	N
B.2. hyperrectangle choice	o_1	N
C. Hyperrectangle expansion:	$2o_1 M$	1
D. Overlap test:	$12o_1 M$	$N-1$
E. Hyperrectangle contraction:	$M(o_1+o_{2,1})$	$N-1$
IAFC		
A. Pre-computations:		
A.1. $\|\|\vec{a}-\vec{v}_j\|\|,\ \forall j$	$M(o_1+o_{2,1})+o_{2,3}$	N
A.2. $1/(\|\|\vec{a}-\vec{v}_j\|\|^2),\ \forall j$	$o_{2,1}+o_{2,2}$	N
A.3. $\sum_{j=1}^{N} 1/(\|\|\vec{a}-\vec{v}_j\|\|^2)$	o_1	N
B. Choice functions	$o_{2,2}M$	N
C. Search process:		
C.1. vigilance test	$2o_1+o_{2,1}+o_{2,2}+o_{2,4}$	N
C.2. membership test	$o_1+o_{2,1}$	N
C.3. category choice	o_1	N
D. Centroid vector update:	$2o_{2,2}M+9o_1+3o_{2,1}+6o_{2,2}$	1
SOFM		
A. Map neuron selection:		
A.1. Euclidean distances	$M(o_1+o_{2,1})+o_{2,3}$	N'
A.2. best-match choice	o_1	N'
A.3. increment its input count	o_1	N'
B. Update prototype(s):		
B.1. define neighborhood	$3o_1+2o_{2,1}+o_{2,3}$	N'
B.2. adjust weights	$M(2o_1+o_{2,2})$	N'
C. Modify $\alpha(t)$ and $r(t)$:	$2(o_1+o_{2,1}+o_{2,2})$	1
D. Map interpretation:		
D.1. find category centers	o_1	N'
D.2. relate neurons	$M(o_1+o_{2,1})+o_{2,3}$	$N'-1$
D.3. category choice	o_1	1