*Neural Networks*, Vol. 4, pp. 485–491, 1991
Printed in the USA. All rights reserved.

0893-6080/91 $3.00 + .00
Copyright © 1991 Pergamon Press plc

*ORIGINAL CONTRIBUTION*

# Clustering Characterization of Adaptive Resonance

LAURA IGNIZIO BURKE

Lehigh University

(*Received* 19 *March* 1990; *revised and accepted* 11 *March* 1991)

**Abstract**—*By exploiting the relationship between adaptive resonance networks in the fast learning mode and clustering algorithms, we seek to characterize and streamline their operations. We make the observation that a variant of* k*-means learning, Jancey's, actually provides a more apt counterpart to ART operation. In addition, since redundacy occurs in the search initiated by the reset mechanism in fast learning situations, we suggest an alternative strategy for the practical design of the network. The analogy with* k*-means based clustering leads to the identification of vigilance as a threshold-like measure comparable to parameters in these algorithms.*

**Keywords**—Adaptive resonance theory, Clustering algorithms, Reset mechanism, Fast learning, *K*-means clustering.

## 1. INTRODUCTION

The Adaptive Resonance Theory network due to Carpenter and Grossberg (Carpenter & Grossberg, 1987a, 1987b) has, on the one hand, been hailed as singularly pioneering. On the other, it remains enigmatic to some analysts with "nonneural" backgrounds. Herein we attempt to make accessible the ART architecture (specifically, ART2) by focusing on its clustering function and contrasting it with analogous conventional methods. The fast learning case for ART2 provides us with our specific focus. In particular, $k$-means (with varying $k$) clustering provides an apt counterpart to ART2. The analysis yields new insights into the ART2 search mechanism and motivates a practical modification of the system. Further, a discrete time, algorithmic view of ART2's operations sheds light on its implicit learning rate.

The first section briefly describes the $k$-means clustering algorithm and its relevant variants. This background material provides an environment for discussion of ART2 as a $k$-means emulator. Next, assuming a sequential ordering of operations, fast learning, and a discrete time framework, we describe ART2 and its distinguishing features. Section 3 analyzes the reset mechanism and describes its function via comparison with varying $k$-means clustering. The insight motivates adaptation of the search strategy. Section 4 addresses the previous comparison of ART2 with the single leader sequential clustering algorithm (Lippmann, 1987).

The paper is a revised and abridged version of a chapter of the author's thesis, completed while at University of California, Berkeley, under the advisement of Stuart Dreyfus and while holding a National Science Foundation Graduate Fellowship.

Requests for reprints should be sent to Laura Ignizio Burke, Dept. of Industrial Engineering, Lehigh University, Bethlehem, PA 18015.

## 2. ART: AN OUTGROWTH OF COMPETITIVE LEARNING

Grossberg described the competitive learning system in 1976 (Grossberg, 1976a, 1976b). He proved that such a network whose weights approached their equilibrium values on each trial could stably classify only limited sets of inputs if weights remained plastic indefinitely (Grossberg, 1976a). This observation gave rise to the stability/plasticity tradeoff: If an outside observer does not fix weights at some arbitrary time, then they cannot be guaranteed to converge to stability; i.e., to yield consistent classifications. One method to help stabilize systems that are unstable in general is slow learning, wherein weights are allowed to change very little on each trial (Carpenter & Grossberg, 1987a).

ART1 accomplishes stable fast or slow learning of categories of binary input patterns. We study more closely ART2, the analog counterpart of ART1, and the more apparently complex. In addition, our discussion is concerned only with the fast learning case. We focus on the function of a one-layer competitive learning system as a clustering device; specifically, the $k$-means clustering method. The extension to analogous clustering algorithms yields appropriate counterparts for ART2. In the next section, we provide background on the clustering problem and the $k$-means algorithm for nonhierarchical clustering. It also introduces the $k$-means with varying $k$ extension.

## 2.1. The Clustering Problem

A clustering technique seeks to find a grouping, or clustering, of an $N$-entity data set such that it optimizes some measure. For example, such a technique might attempt to minimize total within-cluster sum of squared distances. Thus input takes the form of $N$ $m$-vectors, or strings of attributes, and output is the cluster membership function,

$$a_{ij} = 1 \text{ if vector } \mathbf{x}_j \text{ is assigned to cluster } C_i, \quad 0 \text{ otherwise.} \tag{1}$$

Methods may be *hierarchical* or *nonhierarchical*. Hierarchical clustering methods yield a tree of divisions of the data. At the root of the tree lies the one-cluster grouping (all data in the same group). Going down the tree, different threshold levels yield partitions of the data into more and more clusters, until ultimately each entity forms its own cluster.

Nonhierarchical methods also partition the data, but not in a way such that partitions are necessarily hierarchically related. A hierarchical method will yield some number $k_1$ clusters for some threshold, and for any $k < k_1$, clusters will be merged. Nonhierarchical methods do not guarantee that such cluster nesting will result.

Nonhierarchical clustering methods yield partitions of the $N$-entity data set which, though not hierarchically related, represent the minimum (or maximum) point of an error (or similarity) criterion (Anderberg, 1973, p. 166). Thus the analyst must decide, as in many hierarchical methods, upon a measure of distance (or, alternatively, similarity). The clustering algorithm attempts to minimize the total aggregated error of the distance measure in the final partition. Nonhierarchical methods suit larger problems than do hierarchical, since they demand only a fraction of the storage required by a hierarchical method's similarity matrix (Anderberg, 1973; Zupan, 1982). Moreover, data is processed serially rather than in batch, which facilitates the neural network comparison.

A standard assumption is that data points concentrate about the mean (or centroid) for their cluster. Define the cluster centroid for cluster $i$ as

$$m_i = \sum_j \frac{a_{ij}x_j}{n_i}, \tag{2}$$

where

$$n_i = \sum_j a_{ij}. \tag{3}$$

If clusters assume the shape of spheres or, more generally, ellipsoids, then define the within cluster scatter matrix for $i$ as

$$W_i = \sum_j a_{ij}(x_j - m_i)(x_j - m_i)^t, \tag{4}$$

where the superscript $t$ denotes transpose.

**W** surfaces frequently in classification and clustering literature. In addition to measuring within cluster scatter, the matrix's eigenvectors give the principal axes of the ellipsoid formed. The corresponding eigenvalues measure the relative scatter of data points along the axes. Thus, as Windham notes (Windham, 1987), a function of **W** serves most often as the criterion of which partitioning methods seek minima.

The criterion of interest herein is the trace criterion:

$$\text{minimize tr}(\mathbf{W}),\ \mathbf{W} = \sum \mathbf{W}_i. \tag{5}$$

Minimizing the trace criterion equates to performing the $k$-means algorithm. See Windham (1987) and Anderberg (1973) for discussion of alternative criteria. Section 2.3. focuses on $k$-means and its variants, and further elaborates. We note that the trace criterion is used also in feature selection procedures. Further, the trace criterion yields ellipsoid-shaped clusters.

## 2.3. *K*-Means Algorithm and Jancey's Variant

As described by Windham, the $k$-means algorithm represents a partitioning method which aims to minimize tr(**W**). Alternatively, it is a nearest-centroid sorting method which implicitly performs nonlinear optimization (Anderberg, 1973; see Burke, 1989, for formulation as a constrained nonlinear programming model). That is, the method finds prototype vectors to represent clusters which minimize the total within cluster sum of squared error,

$$E = \sum_i (x_i - w_{ij})^2 a_{ij}$$

where $\mathbf{x}$ is the input pattern, $\mathbf{w}_j$ is the prototype vector, and $a_{ij}$ is the membership function defined previously. Note that the cluster centroid for a cluster $j$, $\mathbf{m}_j$, minimizes this error. The primary advantage of a nonhierarchical method such as $k$-means comes from its ability to process data serially, so that large similarity matrices need not be stored.

The standard $k$-means algorithm, developed by MacQueen (1965), appears below. Possible variations on certain points follow.

1. For some choice of $k$, designate the first $k$ patterns of the input training set (called seed points or prototype vectors) as clusters having one member;

2. For each remaining pattern, find the cluster having the nearest centroid. Recompute the centroid of the "winning" cluster;
3. After assigning all patterns, perform an additional pass through the set by assigning each pattern to the cluster having the nearest centroid.

A convergent version results if more passes are made (Anderberg, 1973). The algorithm constructs a clustering which depends on the sequence of data units presented; however, MacQueen found the ordering has an insignificant effect when the clusters are well-separated (MacQueen, 1965).

Jancey's variant is identical to the $k$-means algorithm except in the way it computes the new seed point (Jancey, 1966). Instead of computing the centroid in Step 2, it gives as the new seed point a value overshooting the new centroid which lies in the direction of the line from the old seed point to the new centroid. Apparently, the main objective of the approach was intended to speed convergence (Anderberg, 1973). Additionally, Jancey points out that since the clustering problem has many local minima, it is quite possible for the procedure to stabilize in a suboptimal solution. He suggests that allowing prototype vectors to move not just to centroids but "as far again in the same direction" (Jancey, 1966) encourages movement in a direction opposed to the gradient of the objective function. Such movement can enable prototype vectors to escape local minima. Of most importance with respect to ultimate neural implementation is Jancey's perception that the shift of prototype vectors should be changed from the centroid to "some quantity less dependent on numbers" of input patterns in a cluster.

Extensions of the $k$-means algorithm allow $k$, the number of clusters, to vary. Since the performance of the clusterer depends on how well $k$ is chosen, and since no method for determining the best $k$ exists, these *varying-$k$*-means algorithms (MacQueen, 1965) offer hope for better clustering. However, they require threshold distance values, the determination of which can prove equally difficult. The Isodata method is an implementation of a varying $k$-means algorithm (Ball & Hall, 1965).

## 3. ART2: A NEURAL CLUSTERING DEVICE

The primary function of an ART2 module is to carry out clustering, also called coding or categorization. More specifically, it emulates a varying-$k$-means method, and employs a prototype vector update more closely related to that used in Jancey's variant. Below we describe ART2 from this functional perspective. Figure 1 reproduces the ART2 architecture given by Carpenter and Grossberg. Figure 2 gives a simplified ART2 structure, which aids in summarizing the operations of ART2.

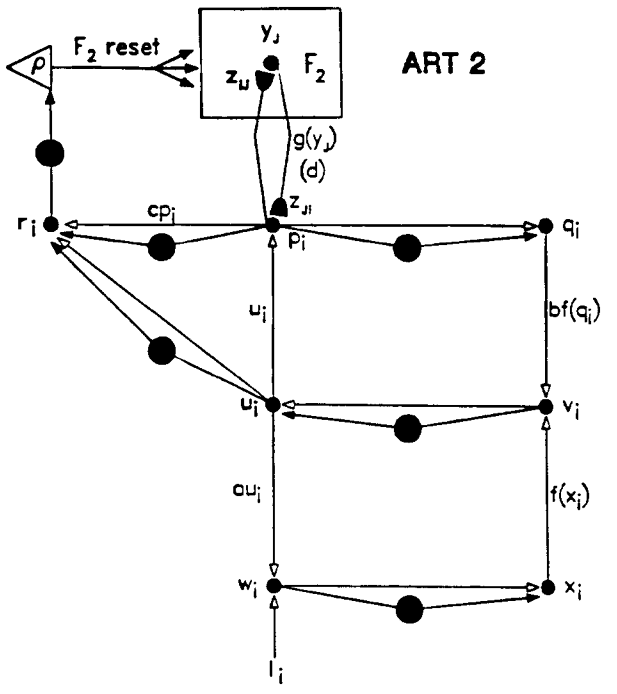

**FIGURE 1. A representation of the ART2 network of Carpenter and Grossberg (reprinted from Carpenter & Grossberg, 1987b).**

### 3.1. Operation of ART2

An input pattern undergoes extensive processing in ART2. Several parameters and constraints play important roles in this processing, as well as in operation of ART2. They are as follows:

- Each component of the **bottom-up** weight vector for all nodes $j$, $j = 1, \ldots, n$, denoted as $\mathbf{zu}_j$, must

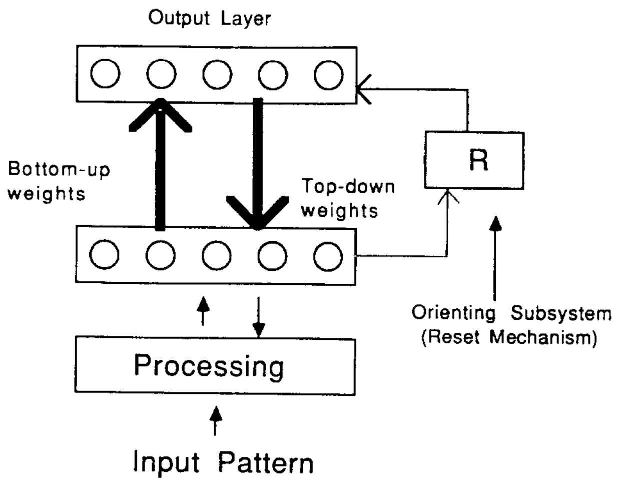

**FIGURE 2. Simplifed representation of ART2 network.**

initially obey

$$zu_{ij} \leq \frac{1}{\sqrt{m}} \tag{6}$$

where $m$ is the dimension of the imput patterns. The **top-down** weight vectors, denoted $z_j$, equal **0** at initialization.

- $a = b = 10$. (7)
- $[(cd)/(1 - d)] \leq 1$. (8)

The lower ($F_1$) layer of ART2 normalizes the pattern, then suppresses noise, then renormalizes. A squashing function accomplishes the noise suppression; Carpenter and Grossberg suggest one of the two following:

$$f(x) = \begin{cases} 2\Theta x^2/(x^2 + \Theta^2) & \text{if } 0 \leq x < \Theta \\ 0 & \text{if } x \geq \Theta, \text{ or} \end{cases} \tag{9}$$

$$f(x) = \begin{cases} 0 & \text{if } 0 \leq x \leq \Theta \\ x & \text{if } x \geq \Theta. \end{cases} \tag{10}$$

The processed input pattern next enters the competitive stage of the system. As in competitive learning, a node at the second ($F_2$) layer receives as input the inner product of its weight vector with the processed input vector. The node having the greatest input "wins" the competition. To employ a clustering interpretation, the weight vector represents a prototype and the competition assigns the input pattern to the closest prototype.

ART2 next checks whether or not the weight vector of the winning node **sufficiently** matches the processed input pattern. If the match should prove insufficient, according to the vigilance parameter discussed below, then the winning node is "reset" by the **orienting subsystem** of the network. The reset node cannot participate in the coding (categorizing) of the present pattern. The $F_2$ node having the next greatest input becomes the winner; its match is checked, and so on. The vigilance parameter corresponds to threshold distance measures in Isodata and varying-$k$-means algorithms.

The equations below, given originally in Carpenter and Grossberg (1987b) show the values for the labeled nodes of the system pictured in Figure 1:

$$w(i) = I(i) + au(i) \tag{11}$$

$$x(i) = w(i)/\|\mathbf{w}\| \tag{12}$$

$$v(i) = f(x(i)) + bf(q(i)) \tag{13}$$

$$u(i) = v(i)/\|\mathbf{v}\| \tag{14}$$

$$p(i) = u(i) + dz_J(i) \tag{15}$$

(where $J$ denotes the node winning the competition)

$$q(i) = p(i)/\|\mathbf{p}\|. \tag{16}$$

Instead of discussing these in detail now, we continue our qualitative description.

Assume that a node has won the competition and satisfied the matching criterion. ART2 updates the weight vector of that node. Essentially, the update equation aligns the weight vector more closely with the input pattern it has coded:

$$\mathbf{z}_J \rightarrow \mathbf{u}/(1 - d). \tag{17}$$

The vector **u** is, however, a nonlinear function both of the current input and of the weight vector itself. The learned vector $\mathbf{z}_J$ is thus a function of all inputs that have activated nodes without reset, even with fast learning. The postprocessing steps involve eqns (15) and (16) above, and ultimately all the above equations.

## 3.2. Parameters, Constraints, Initialization

Several parameters affect ART2 operation. Carpenter and Grossberg suggest values for some of these, and provide ranges for others. The parameters $c$ and $d$ play an important role in the reset equation, subject to the constraint $cd/(1 - d) \leq 1$. In simulations, $c = 0.1$ and $d = 0.9$. Carpenter and Grossberg observe that stability requires that $a$ and/or $b$ be large relative to 1, and select $a = b = 10$ in ART2 simulations. The vigilance parameter, $\rho$, has a well-defined meaning but is difficult to specify *a priori*, as it defines the level of similarity of patterns within classes. We further discuss the problem of specifying vigilance in Section 3.4.

Each output layer unit has associated with it two weight vectors. Although they begin as distinct vectors for **uncommitted** nodes (nodes having not yet won a competition or **coded** an input pattern), as soon as the unit wins a competition (codes a pattern), the two vectors become identical.

Carpenter and Grossberg further specify initial weight vectors by observing properties realized by them. They note that a system will more likely form new categories if the bottom-up vector **zu** is closer to its limit, and that such an initial choice actually helps the system to stabilize. In a system without reset, taking

$$zu_J(i) \ll 1/\sqrt{m} \tag{18}$$

can greatly impact results by forming fewer classes than $n$, the maximum number allowed for. Carpenter and Grossberg also recommend a variety of choices for initial weights within constraints: learned patterns, randomly distributed patterns, or uniform vectors. These suggestions differ from typical neural network initial weight vectors (small random vectors) and typical "seed points" for $k$-means clustering (input patterns themselves, random vectors, or means of specific groups).

The ability to choose normalized uniform initial weight vectors allows real-time users to benefit from having initial system configurations which lead to reliable performance. Moreover, the incremental advantage of using random vectors over uniform does not support their use. Once a node codes a pattern, the new weights (both top-down and bottom-up) in no way incorporate original weight vectors.

Thus initial bottom-up vectors merely facilitate the system startup. In reality, the first vector coded by a unit serves as the weight vector (or seed point) for the corresponding class. Such a process has prompted comparisons of ART with the single leader sequential clustering algorithm; Section 4 further discusses the comparison.

## 3.3. Clustering Analysis of Reset and Ensuing Search in ART2

The reset function in ART2 essentially checks that the "winning" node at the output layer actually matches the input pattern as closely as desired. The user inputs the vigilance parameter, $0 \le p \le 1$, to quantify "closely enough." If the prototype for class $j$, its weight vector $\mathbf{z}_j$, does not sufficiently match input $\mathbf{x}$, then $j$ is reset and cannot participate in the ensuing repeat of the competitive phase.

Reset occurs, if the reset function, $R$, falls below $p$. Carpenter and Grossberg carefully orchestrate $R$, the initial weights of the system, and the system parameters to ensure the following:

1. When learning has just begun, ART2 will not reset a new, uncommitted node; i.e., learning can commence
2. As learning progresses and nodes become committed, mismatches will trigger resets; i.e., **learning increases mismatch sensitivity**.

The reset function is:

$$R = \frac{\sqrt{(1 + c)^2 + 2(1 + c)\,\|cd\mathbf{z}_j\|\cos(u, \mathbf{z}_j) + \|cd\mathbf{z}_j\|^2}}{1 + \sqrt{c^2 + 2c\,\|cd\mathbf{z}_j\|\cos(u, \mathbf{z}_j) + \|cd\mathbf{z}_j\|^2}} \tag{19}$$

(Carpenter & Grossberg, 1987b, eqn (25)).

We note that, by definition of the $\mathbf{z}_j$ with fast learning,

$$\|\mathbf{z}_j\|^2 = [1/(1 - d)]^2 \tag{20}$$

for all nodes $j$ which have coded at least one pattern (i.e., for all committed $j$).

When the winning node at level $F_2$ fails to satisfy the reset equation, i.e., if $R < p$, then the node with the next highest input at the competitive level becomes the new winner. If several categories exist, the search for a satisfactory node may be somewhat prolonged. Further, if all nodes have learned at least one pattern, there exists the possibility that no node will suffice. Thus, in the fast learning case, it seems that a modified search is called for.

Carpenter and Grossberg construct $R$ as a function of $\mathbf{u}$ and $\mathbf{z}_J$, where $\mathbf{u}$ is the normalized, processed input vector and $\mathbf{z}_J$ is the weight vector associated with winning node $J$. As noted before, however, if $J$ has previously coded any pattern, then its weight vector was

$$z_J(i) = u(i)/(1 - d), \quad i = 1, \ldots, m. \tag{21}$$

at the end of each trial in which node $J$ is chosen. Moreover, $\|\mathbf{u}\| = 1$; thus, $\|\mathbf{z}_J\| = [1/(1 - d)]$. Also, observe that

$$\cos(u, z_J) = \frac{u^t\mathbf{z}_J}{\|u\|\|\mathbf{z}_J\|}, \tag{22}$$

or

$$\cos(u, \mathbf{z}_J) = \frac{u^t\mathbf{z}_J}{[1/(1 - d)]} \tag{23}$$

at the end of the trial. In addition, the ART2 $F_1 \rightarrow F_2$ input is proportional to $\mathbf{u}^t z_j$ for all nodes $j$. Thus, $\cos(\mathbf{u}, \mathbf{z}_J)$ is proportional to the input to the coded nodes $J$, in the fast learning situation.

The reset function should increase with $\cos(\mathbf{u}, \mathbf{z}_J)$. After all, the closer $\mathbf{u}$ is to $\mathbf{z}_J$, the less desirable is reset. However, if $R$ does increase with $\cos(\mathbf{u}, \mathbf{z}_J)$ then the search process becomes redundant in fast learning, where weights reach asymptote on each trial. In this case, since the winning node $J$ has maximum input no other node $j$ can have $\cos(\mathbf{u}, \mathbf{z}_j) > \cos(\mathbf{u}, \mathbf{z}_J)$. Thus, the search ensuing after a mismatch triggers reset involves nodes which cannot satisfy the reset inequality, if our conjecture that $R$ increases with $\cos(\mathbf{u}, \mathbf{z}_j)$ holds. The conjecture is proven in (Burke, 1989) for the fast learning case; it can also be seen from inspection of Figure 7 of (Carpenter & Grossberg, 1987b) and by simple differentiation of the function $F(x) = R^2$, where $x = \cos(u, z)$.

Carpenter and Grossberg seek to automate search. That is, parameters $c$ and $d$ together with initial weights and the reset function allow the network to "know" if a winning node is committed and thus requires a true check for reset. In our analysis, we separate the committed node from the uncommitted since the upshot (for fast learning) of Carpenter and Grossberg's goal is that only committed nodes undergo reset checks. The complex approach results from biological evidence precluding the marking of nodes (or their creation *in vivo* (Grossberg, 1976a)). Marking nodes with the label *committed* or *uncommitted* is the solution suggested here to facilitate efficient search. The next section puts forth the practical alternative search strategy which accomplishes the same end as does the original.

### 3.4. Alternative Strategy: Varying-*k*-Means Interpretation

An alternative search strategy might consist of merely resetting the node $J$ found inadequate as a "match" for input $\mathbf{u}$, and immediately forming a new category for $\mathbf{u}$; i.e., by committing an uncommitted node. Carpenter and Grossberg's orienting subsystem accomplishes this goal, but requires the blind search described above. Moreover, we can further restrict the number of nodes to $n_{\max}$ allowed at layer $F_2$, and form a "residue" of rejected patterns for any vigilance level. Thus, if an incoming pattern triggers reset and $n_{\max}$ committed nodes exist at layer $F_2$, we consider the pattern unfamiliar at this point. Such an approach circumvents the problem of creating numerous groups consisting of only one or two outliers. Recall that such a pragmatic approach substitutes wired-in knowledge (labels of uncommitted or committed) for the automation devised by Carpenter and Grossberg.

Our approach uncovers underlying relations between ART2 and the varying-*k*-means algorithm. The reset function partially obscured the comparison, but our analysis and simplified strategy further illuminates similarities. The *k*-means algorithm with varying $k$ prescribes an association and mismatch strategy much like ART2's. A parameter dictating the desired closeness of a match functions similarly to $\rho$, and any mismatched inputs go to the residue. The original ART2 structure does not provide for a residue, which implies that the user must specify the number of clusters, $k$. If $k$ can be chosen arbitrarily large, we might consider the problem of specifying $\rho$ equivalent to the dual of specifying $k$.

With the aid of a residue, the specification of $\rho$ becomes easier. If a disproportionately large number of patterns are sent to the residue, we can reduce the vigilance level. Thus we might start with a relatively high $\rho$ and reduce it only as much as necessary to achieve a desirably small (or perhaps nonexistent) residue. The lack of precision in the procedure follows from its relationship with clustering.

## 4. THE LEADER ALGORITHM AND ART2

ART2 shares characteristics with the single leader sequential clustering algorithm, as pointed out by Lippmann (1987). We consider the varying *k*-means algorithm to be the more fundamentally analogous. However, some interesting similarities exist between the leader algorithm and ART2, and we summarize and refine them here.

The leader algorithm assigns inputs to categories with "indecent haste" (Hartigan, 1975); though fast, it usually serves only to provide an initial, quick clustering. It forms a class for the first input and denotes that input as the leading case for class 1, $L(1)$. The distance between the next input and $L(1)$ must fall below a threshold $T$, chosen beforehand by the user; otherwise the new input forms a new class and becomes $L(2)$. The algorithm continues in this way, assigning new inputs to classes based on distances to the leading case for the class and forming new classes for dissimilar inputs.

Unlike ART2, the leading case for a class (i.e., the weight vector) does not change after the first assignment. The leader algorithm requires only one pass through the data (which accounts for its speed). It does not possess properties of contrast enhancement or noise suppression, nor does it attempt to update and improve its "leading cases."

Improvements on the leader algorithm include conducting many passes through the data, which eliminates the system's dependence on ordering of inputs. The difficulty of choosing the threshold $T$ mirrors the problem of determining the vigilance parameter in ART2. A relationship exists, however, between $T$ and the resulting number of clusters. If the leader algorithm produces $K$ clusters with some $T$ for a specific ordering of inputs, then, for any ordering, it will produce no more than $K$ clusters for threshold $2T$ (Hartigan, 1975).

ART2 and the leader algorithm share a number of characteristics. Both process inputs serially, and both utilize a threshold for deciding when to form new clusters. Lippmann made the comparison in a paper which is frequently cited and referenced for such analogies. In extending the notion of a clustering counterpart to ART2, however, we build on Lippmann's analysis by suggesting the varying *k*-means algorithm as an apt counterpart.

## 5. CONCLUSION

Loose analogies between competitive learning and *k*-means clustering have surfaced throughout the neural network literature, although little serious attention has followed. This paper has reviewed the actual *k*-means algorithm for clustering, in all of its simplicity. We viewed the ART2 network, a mathematically and biologically elegant extension of competitive learning, as an implementation of a clustering algorithm. However, we would do the system a great disservice if we tried to characterize it as merely a parallel implementation of *k*-means learning. Rather, we aimed to utilize clustering ideas to better understand and potentially improve the ART2 network.

First, as indicated in Section 3, ART2 functions similarly to a *k*-means with varying $k$ algorithm. Significantly, however, it does not perform a centroid computation as in these algorithms. Rather, in a manner similar to Jancey's variant, the weight vector update causes the new weight vector to become more aligned with the most recently coded input pattern, and the change may "overshoot" the centroid. The similarity, together with our demonstration that a shortcut search may replace the automated original without altering its function, led to the proposed revised search strategy and its subsequent role in determining the vigilance level. Elsewhere, the results of the adapted ART2 network appear (see Burke, 1989, 1990). In this engineering application, the system facilitated classification of sensor signals indicating level of tool wear to 95% accuracy.

Intriguingly complex and elegantly simple, ART2 shares properties with the single leader clustering algorithm, as well as the varying $k$-means method. But the *dissimilarity* between ART2 and clustering algorithms may play a large role in endowing it with provably "brain-like" properties, especially stability in the face of a changing environment and plastic weights. Even in engineering applications, the realization of such ideals will yield the dominant systems in automating intelligent decision-making.

## REFERENCES

Anderberg, M. (1973). *Cluster analysis for applications*. New York: Academic Press.

Ball, G., & Hall, D. (1965). *Isodata: A novel method of data analysis and pattern classification*. AD699616, Stanford Research Institute, Menlo Park, CA.

Burke, L. (1989). *Automated identification of tool wear states in machining processes: An application of self-organizing neural networks*. Unpublished Ph.D. dissertation, University of California, Berkeley.

Burke, L. (1990). An unsupervised neural network approach to tool wear identification. Submitted to *Institute of Industrial Engineers Transactions*, in press.

Carpenter, G., & Grossberg, S. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics and Image Processing*, **37**, 54–115.

Carpenter, G., & Grossberg, S. (1987b). ART2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, **26**(23), 4919–4946.

Grossberg, S. (1976a). Adaptive pattern classification and universal recording: I. Parallel development and coding of neural feature detectors. *Biological Cybernetics*, **23**, 121–134.

Grossberg, S. (1976b). Adaptive pattern classification and universal recording: II. Feedback, expectation, olfaction, illusions. *Biological Cybernetics*, **23**, 187–202.

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

Jancey, R. (1966). Multidimensional group analysis. *Australian Journal of Botany*, **14**, 127–130.

Lippmann, R. (1987). An introduction to computing with neural networks. *IEEE ASSP Magazine*, **4**, 4–21.

MacQueen, J. (1965). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Symposium on Mathematical Statistics and Probability, 5th, Berkeley*, **1**, (pp. 281–297). Berkeley, CA: University of California Press.

Windham, M. (1987). Parameter modification for clustering criteria. *Journal of Classification*, **4**, 191–214.

Zupan, J. (1982). *Clustering of large data sets*. New York: Research Studies Press.