Dynamics of Projective Adaptive Resonance Theory Model: The Foundation of PART Algorithm

Yongqiang Cao and Jianhong Wu

***Abstract*—Projective adaptive resonance theory (PART) neural network developed by Cao and Wu recently has been shown to be very effective in clustering data sets in high dimensional spaces. The PART algorithm is based on the assumptions that the model equations of PART (a large scale and singularly perturbed system of differential equations coupled with a reset mechanism) have quite regular computational performance. This paper provides a rigorous proof of these regular dynamics of the PART model when the signal functions are special step functions, and provides additional simulation results to illustrate the computational performance of PART.**

***Index Terms*—Data clustering, differential equations, learning and adaptive systems, neural networks, pattern recognition.**

I. Introduction

DATA clustering is an unsupervised process of classifying patterns into groups (clusters), aiming at discovering structures hidden in a data set. Clustering problems have been studied extensively from different angles and using different approaches (for example, [21], [23], [7], [8], [12], [24], [27], [2], [1], [22], [16]), and new challenging arises as well due to the technology development. In particular, spectacular advances in information technology and large-scale computing are producing huge and very high dimensional data sets. These data sets arise naturally in a variety of contexts such as text/web mining, bioinformatics, imaging for diagnostics and surveillance, astronomy and remote sensing. The dimension of these data is in the hundreds or thousands (see, for example, [19]). In order to understand such high dimensional data sets, one needs to overcome many technical challenges. Traditional clustering methods do not work efficiently for data sets in such high dimensional spaces because of the inherent sparsity of data. In such data sets, for any given pair of points there exist at least a few dimensions on which the points are far apart from one another. For instance, for a data set that obeys the Gaussian distribution $N(\mu;\sigma^2 I_n)$, we have to pick at least 2^n random points from the data set in order to obtain just a few which are at distance less than $(1/2)\sigma\sqrt{n}$ from the center [14]. Therefore, a set of trillion points is still very sparse if it is in a thousand dimensional space! Furthermore, recent theoretical results [3] have shown that in a high dimensional space, the distance between every pair of points is almost the same for a wide variety of data distributions and distance functions. Under such circumstances, it even makes no sense to talk about proximity or clustering in the original full space of all dimensions. This is well known as the curse of dimensionality. This motivated the concept of subspace clustering whose goal is to find clusters formed in subspaces of the original high dimensional space.

CLIQUE [2] is the first well-known algorithm designed for automatically discovering clusters in different subspaces of a higher dimensional space. It uses a density-based approach [21], [15], [25]. To approximate the density of the data points, CLIQUE partitions the data space and finds the number of points that lie inside each cell (unit) of the partition. More precisely, the units are obtained by partitioning every dimension into ξ intervals of equal length. For a given subspace, a cross product of such intervals (one from each of dimensions of this subspace) is said to be a unit in the subspace. A unit is said to be dense if the fraction of total data points contained in the unit is above a density threshold τ. Then a maximal set of connected dense units in a given subspace forms a cluster in the subspace. Since a dense unit of k-dimensional subspace is also dense when projected into a $(k-1)$-dimensional subspace, CLIQUE therefore finds all dense units in k-dimensional subspace based on the dense units in $(k-1)$-dimensional subspace. It first finds all dense units in 1-dimensional subspace, then works from lower dimensional subspaces to higher ones to discover dense clusters in all subspaces. The time complexity of CLIQUE is therefore exponential in the highest dimensionality of any dense unit. Therefore, the time cost may be prohibitive when the dimensionality is high.

A fast subspace clustering algorithm PROCLUS was proposed by Aggarwal *et al.* in 1999 [1]. PROCLUS aims at finding projected clusters, each of which consists of a subset C of data points together with a subset D of dimensions such that the points in C are closely correlated in the subspace of dimensions D. PROCLUS uses a medoids-based optimization approach [23], [24] and combines the greedy technique and a locality analysis to find the set of dimensions associated with each medoid. Here a medoid is a representative object in a cluster to serve as the surrogate center for the cluster. The clustering quality (the criterion function to be optimized) is evaluated as the average Manhattan segmental distance from all points to their corresponding cluster centers. Here, the Manhattan segmental distance from a point to its cluster center is defined relative to the dimensions in which the cluster is formed. More precisely, for any two points $x_1 = (x_{11},\dots,x_{1d})$,

Manuscript received January 14, 2002; revised July 9, 2003. This work was supported in part by the Canada Research Chairs (CRC) Program, the Natural Sciences and Engineering Research Council of Canada (NSERC), and in part by the Mathematics for Information Technology and Complex Systems (MITACS).

Y. Cao is also with the Department of Cognitive and Neural Systems, Boston University, Boston, MA 02215 USA (e-mail: yqcao@bu.edu).

J. Wu is with the Laboratory for Industrial and Applied Mathematics, Department of Mathematics and Statistics, York University, Toronto, ON M3J 1P3, Canada; Fax: 1-416-7365757, E-mail: wujh@mathstat.yorku.ca.

Digital Object Identifier 10.1109/TNN.2004.824261

$x_2 = (x_{21}, \ldots, x_{2d})$, and a set of dimensions $D, |D| \le d$, the Manhattan segmental distance between x_1 and x_2 relative to D is given by $d_D(x_1, x_2) = \sum_{i \in D} |x_{1i} - x_{2i}|/|D|$. With two input parameters, the number of clusters k and the average dimensions l, PROCLUS works as follows. In the initialization phase, a small superset $\mathcal{M}$ of medoids is generated from a random sample of the data set, by using a greedy method such that the points (medoid candidates) in the superset $\mathcal{M}$ are well separated from each other. Then a set of k medoids from $\mathcal{M}$ is randomly chosen. In the iterative phase, the set of current medoids is updated by a hill climbing process: the bad medoids are replaced by random points from $\mathcal{M}$. The bad medoids are defined as the medoid of the cluster with the least size and the medoids of the clusters with sizes less than a predefined threshold. The dimensions associated with each medoid are determined by a locality analysis based on the simple idea that data points of a cluster are closer in a correlated dimension than in an uncorrelated dimension. More specifically, for each medoid m_i, let δ_i be the minimum distance from any other medoid to m_i. For each i, define the locality $\mathcal{L}_i$ to be the set of points which are within distance δ_i from m_i on the entire space. Then compute the average distance along each dimension from the points in $\mathcal{L}_i$ to m_i. Let X_{ij} be the average distance along dimension j, then normalize X_{ij} as $Z_{ij} = (X_{ij} - Y_i)/\sigma_i$, where $Y_i = (\sum_{j=1}^{d} X_{ij})/d, \sigma_i = \sqrt{\sum_j (X_{ij} - Y_i)^2/(d-1)}$, and d is the number of dimensions of the entire space. Finally, pick up the kl numbers with the least values of Z_{ij}, and the set D_i of dimensions associated with medoid m_i is then determined according to the rule that dimension $j \in D_i$ if Z_{ij} is selected. Of course, this is only an approximation to the dimensions associated with each medoid. The clusters are then formed by grouping every data point to its closest medoid according to the Manhattan segmental distance relative to the set of dimensions associated with the medoid. By simulations on synthetic data sets, Aggarwal *et al.* [1] showed that PROCLUS performed better than CLIQUE in term of quality and running time. However, a problem with this algorithm is how to determine its two input parameters, the number of clusters k and the average dimension l. Moreover, as the illustrative simulations in [4] showed, PROCLUS is sensitive to the choice of these input parameters. This imposes significant challenges for a user, as determining the number of natural clusters is one of the most difficult problems in cluster analysis. Besides, the estimation of the dimensions associated with each medoid may be quite inaccurate since the locality $\mathcal{L}_i$ usually contains many uncorrelated points. As a result, the clustering quality may be degraded.

In [4], we proposed an alternative approach based on a new neural network architecture projective adaptive resonance theory (PART) in order to provide a solution to the challenging high-dimensional clustering problem. The basic architecture of PART is similar to that of adaptive resonance theory (ART) neural networks which have been shown to be very effective in self-organized clustering in full dimensional spaces (for example, [7]–[13]). The adaptive resonance theory (ART) was first introduced by Grossberg in 1976 [17], [18] in order to analyze how brain networks can autonomously learn in real time about a changing world in a rapid but stable fashion, based on which Capenter and Grossberg [7], [8] developed two

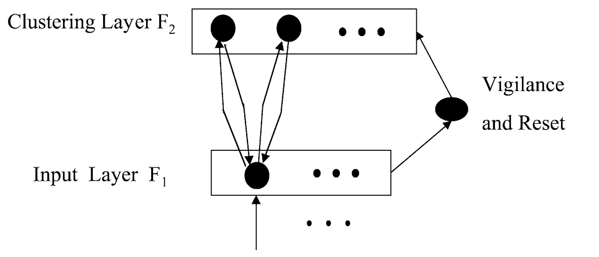

Fig. 1. Simplified configuration of ART architecture consisting of an input layer F_1, a clustering layer F_2 and a reset subsystem.

classes of ART neural network architectures ART1 and ART2 whose computational performance (dynamics) are described by systems of differential equations. ART1 self-organizes recognition categories for arbitrary sequences of binary input patterns, while ART2 does the same for either binary or anolog inputs. Some other classes of ART neural network architectures such as Fuzzy ART [12], ARTMAP [10], Fuzzy ARTMAP [13], Gaussian ARTMAP [26], and Distributed ART and Distributed ARTMAP [5], [6] were then developed with increasingly powerful learning and patten recognition capabilities in either an unsupervised or a supervised mode. Simply speaking, an ART network includes a choice process and a match process as its key parts. The choice process picks up the most likely category (cluster) for an input pattern. If the template of the chosen category is sufficiently similar to the input pattern to satisfy a predefined vigilance parameter ρ, then the category resonates and learns: its template is updated to respond to the new input pattern. Otherwise, the category is reset, and the next most likely category is chosen. If no existing category satisfies the match criterion, then a new category is recruited. Thus, ART incrementally produces categories necessary to represent clusters of input patterns. Fig. 1 shows the simplified configuration of an ART structure, which involves an input processing field (F_1 layer, also called input layer or comparison layer), a clustering field (F_2 layer, also called clustering layer or recognition layer), and a vigilance and reset subsystem. There are two sets of connections (each with its own weights) between each node in F_1 layer and each node in F_2 layer. F_1 layer is connected to F_2 layer by bottom-up weights while F_2 layer is connected to F_1 layer by top-down weights (also called templates). The connection weights between these two layers can be modified according to two different learning rules. The F_2 layer is a competitive layer which follows the winner-take-all paradigm: the node in F_2 with the largest net input becomes the candidate to learn the input pattern. Whether the candidate will learn the input pattern is decided by the vigilance and reset mechanism, which controls the degree of similarity of patterns placed in the same node (cluster).

The advantage of ART is that it does not assume the number of clusters in advance and allows the user to control the degree of similarity of patterns placed in the same cluster. Despite the great success of applying ART to clustering problems, our simulations reported in [4] show that the current ART architecture has to be modified to perform the task of subspace clustering in high dimensional data sets. In particular, ART focuses on similarity of patterns in the full dimensional space and thus may

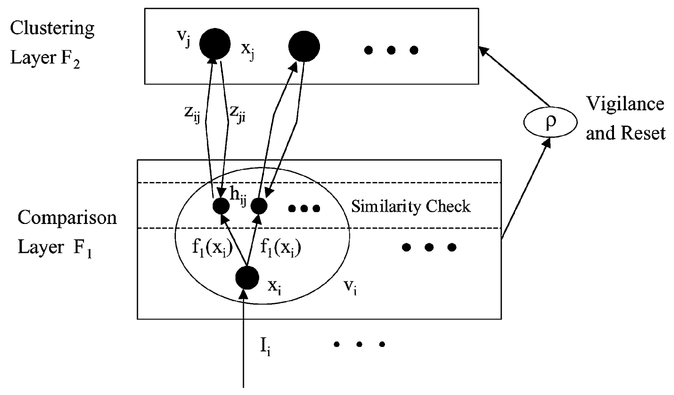

Fig. 2. PART architecture. In addition to the usual F_1 layer (input and comparison), F_2 layer (clustering) and a reset mechanism, there is a hidden layer associated with each F_1 layer node v_i for similarity check to determine whether the node v_i is active relative to an F_2 layer node v_j.

fail to find patterns formed in subspaces of higher dimensional space.

A significant challenge in subspace clustering is that the subspaces in which clusters are formed can not be identified in advance. One may attempt to find clusters in all possible subspaces and then to compare the results to obtain an optimal partition of the data set, but this is practically not feasible as the number of all possible subspaces $2^m - 1$ is intractably large for a data set with high dimension m. PART addresses this problem by introducing a selective output signaling mechanism to ART. Fig. 2 illustrates the basic PART architecture. Similar to ART architecture, F_2 layer of PART is a competitive layer which follows the winner-take-all paradigm. The principal difference between PART and ART lies in the functioning of the F_1 layer. In PART, F_1 layer selectively sends signals to nodes in F_2 layer. In other words, a node in F_1 layer can be active relative to some F_2 nodes but inactive relative to other F_2 nodes. To which F_2 node an F_1 node is active is determined by a similarity test between the corresponding top-down weight and the signal generated in the F_1 node. This similarity test plays a key role in the subspace clustering of PART. More precisely, we denote the nodes in F_1 layer (Comparison layer) by $v_i, i = 1, \ldots, m$; nodes in F_2 layer (Clustering layer) by $v_j, j = m+1, \ldots, m+n$; the activation of F_1 node v_i by x_i, the activation of F_2 node v_j by x_j; the bottom-up weight from v_i to v_j by z_{ij}, the top-down weight (also called template) from v_j to v_i by z_{ji}. The main difference between PART and ART is the introduction of the selective output signal h_{ij} (to be specified later) from a node v_i in F_1 to a committed node v_j in F_2 by a similarity check between the top-down weight z_{ji} and the signal $f_1(x_i)$ generated in v_i, where f_1 is a signal function to be specified later. The introduction of the selective output signaling mechanism in PART allows the signal generated in a node in the input layer to be transmitted to a node in the clustering layer only when the signal is similar to the top-down weight between the two nodes, and hence PART focuses on dimensions where information can be found. Like ART, the vigilance and reset mechanism in PART controls the degree of similarity of patterns in the same cluster, but the similarity measurement in PART is closely related to subspaces involved. In particular, the degree of similarity of patterns is controlled by both vigilance parameter and distance parameter which control the size of dimensions of the projected subspaces and the degree of similarity in a specific dimension involved, respectively. In contrast to PROCLUS, these vigilance and distance parameters are the only required input parameters for the PART algorithm. As our simulations on high dimensional synthetic data in [4] and the additional results reported in the Appendix of this paper showed, the PART algorithm, with a wide range of input parameters, enables us to find the correct number of clusters, the correct centers of the clusters and the sufficiently large subsets of dimensions where clusters are formed, so that we are able to fully reproduce the original input clusters after a reassignment procedure which reassigns every data point to its closest cluster center according to the distance on the subspace of the found dimensions.[1]

The PART algorithm developed in [4] is based on the assumptions that the model equations of PART (a large scale and singularly perturbed system of differential equations coupled with a reset mechanism) have quite regular computational performance described by the following dynamical behaviors, during each learning trial (when a constant input is imposed).

a) Winner-take-all paradigm: the F_2 node with the largest bottom-up filter input becomes the winner and only this node is activated after some finite time.
b) Selective output signals remain to be constants.
c) Synaptic weights are updated following specific formulae.
d) Set of dimensions of a specific projected cluster is nonincreasing in time.

The purpose of this paper is to provide a rigorous proof of the aforementioned dynamical behaviors of the model equations when the signal functions are step functions with thresholds, and to provide additional simulation results to further illustrate the algorithm. Our main results (Theorem 3.2) also provide parameter ranges where these dynamical behaviors hold (equivalently, where the PART algorithm works).

The rest of this paper is organized as follows. Section II describes the model and formulates all assumptions on parameters and the signal functions. Section III states the main results, followed by some explanations in terms of the PART algorithm. Section IV gives the proof of main results. Section V presents some further discussions and concluding remarks. We also summarize PART algorithm and give an illustrative example in the Appendix.

II. Dynamic Model of PART

We rewrite the dynamic model of PART described in [4] with precise time domain specification as follows. We refer to [4] for readers who are interested in how the model equations are derived, and we defer to the Appendix for a summary of PART algorithm.

Let $\Lambda_1 = \{1, \ldots, m\}$ denote the set of nodes in F_1 layer, and $\Lambda_2 = \{m+1, \ldots, p\}$ denote the set of committed nodes in F_2 layer. Recall that a node of the F_2 layer is called committed if it has learned some input patterns in previous learning traces, and a node is called noncommitted if it has not learned any input pattern.

[1]Here the correctness means the natural cluster structure which is hidden in a data set. In particular, in high dimensional synthetic data we considered here, the correctness means the input cluster structure.

The short term memory or activation (STM) equations for nodes in F_1 layer are given by

$$\epsilon_1 \frac{dx_i}{dt} = -x_i + I_i, \quad t \geq -1, \quad i \in \Lambda_1 = \{1, \ldots, m\} \tag{1}$$

where $0 < \epsilon_1 \ll 1, I_i$ is the (constant) input imposed on node v_i.

The STM equations for the committed nodes in F_2 layer are given by

$$\begin{aligned}\epsilon_2 \frac{dx_j}{dt} &= -x_j + [1 - Ax_j][g(x_j) + T_j] \\ &\quad - [B + Cx_j] \sum_{k \neq j, k \in \Lambda_2} g(x_k), \\ &\quad t \geq 0, j \in \Lambda_2 = \{m+1, \ldots, p\}\end{aligned} \tag{2}$$

where $0 < \epsilon_2 \ll 1, g$ is a signal function to be specified later, A, B, and C are nonnegtive constants, and the bottom-up filter input T_j satisfies

$$T_j = D \sum_{i \in \Lambda_1} z_{ij} h_{ij} \tag{3}$$

with a constant $D > 0$ and the selective output signal h_{ij} to be defined later.

The LTM (Long Term Memory or weight) equations for committed nodes v_j in F_2 layer are

$$\delta \frac{dz_{ij}}{dt} = f_2(x_j) \left[(1 - z_{ij}) L h_{ij} - z_{ij} \sum_{k \neq i, k \in \Lambda_1} h_{kj} \right] \quad t \geq 0 \quad i \in \Lambda_1 \quad j \in \Lambda_2 \tag{4}$$

$$\gamma \frac{dz_{ji}}{dt} = f_2(x_j)[-z_{ji}(t) + f_1(x_i)] \quad t \geq 0 \quad i \in \Lambda_1 \quad j \in \Lambda_2 \tag{5}$$

where $0 < \delta \ll \gamma = O(1)$, and $L > 0$ is a given constant, f_2 is a signal function.

The LTM equations for noncommitted winning node v_j in F_2 layer are

$$\delta \frac{dz_{ij}}{dt} = [1 - z_{ij}]L - z_{ij}(m-1) \quad t \geq 0 \quad i \in \Lambda_1 \tag{6}$$

$$\delta \frac{dz_{ji}}{dt} = -z_{ji} + f_1(x_i), \quad t \geq 0 \quad i \in \Lambda_1. \tag{7}$$

The term h_{ij} is related to the essentially new feature of PART: the selective output signaling mechanism. It is defined as follows:

$$h_{ij} = h_\sigma(d(f_1(x_i), z_{ji})) l_\zeta(z_{ij}) \quad t \geq 0 \quad i \in \Lambda_1 \quad j \in \Lambda_2 \tag{8}$$

where we take

$$d(a, b) = |a - b| \quad \text{for any} \quad a, b \in \mathbb{R} \tag{9}$$

and for a given constant c, h_c is given by

$$h_c(\xi) = \begin{cases} 1, & \text{if } \xi \leq c \\ 0, & \text{if } \xi > c \end{cases} \tag{10}$$

and l_c is given by

$$l_c(\xi) = 1 - h_c(\xi). \tag{11}$$

In the definition of (8), we require $\sigma > 0$ and

$$0 \leq \zeta < L/(L - 1 + m). \tag{12}$$

In the PART architecture, the above dynamical process is coupled with a reset mechanism: A winning F_2 node will be reset so that it will always be inactive during the remainder of the current trial if it does not satisfy some vigilance conditions. In particular, a winning (active) F_2 node v_j will be reset if at any given time $t \geq 0$, the degree of match r_j defined by

$$r_j = \sum_i h_{ij} \tag{13}$$

is less than a prescribed vigilance. Namely, reset occurs if and only if

$$r_j < \rho$$

here $\rho \in \{1, 2, \ldots, m\}$ is a vigilance parameter.

In what follows, we are going to make the following set of standing assumptions:

(H1). The constants A, B, and C satisfy

$$A = 1, \quad B = 0, \quad C > 0. \tag{14}$$

(H2). The signal function $f_1 : R \to R$ is nondecreasing and satisfies the Lipschitz condition

$$|f_1(x) - f_1(y)| \leq M|x - y|, \quad x, y \in \mathbb{R}. \tag{15}$$

(H3). The signal functions f_2 and g satisfy

$$f_2(x) = \begin{cases} 1, & \text{if } x \geq \eta \\ 0, & \text{if } x < \eta \end{cases} \tag{16}$$

$$g(x) = \begin{cases} 1, & \text{if } x \geq \theta \\ 0, & \text{if } x < \theta \end{cases} \tag{17}$$

where θ and η are two thresholds. Usually we require $\theta \leq \eta$.

Assumption (H1) is made for the sake of simlpifying the presentation, and is in fact a rescaling procedure so that the maximum level of activation of each node in F_2 layer is 1. A and C can be any positive constants, and B can be any nonnegative constant, as long as the ranges of signal functions and thresholds involved are modified accordingly. (H2) is a assumption satisfied by all signal functions used in the literature. (H3) requires all signal functions to be step functions with thresholds, and these reflect the on-or-off characteristic of neurons and make our already long proof relatively simple. We shall address the limitation of these assumptions in the final discussion section.

For convenience, we define some notations as follows. Define

$$T^{\max} = \frac{DmL}{L - 1 + m} \tag{18}$$

$$T^{\min} = \frac{DL}{L - 1 + m}. \tag{19}$$

We then let $T_{\max}$ and $T_{\min}$ be two arbitrarily given constants satisfying

$$T_{\max} \geq T^{\max} \tag{20}$$
$$T_{\min} \leq T^{\min}. \tag{21}$$

In other words, $T^{\max}$ and $T^{\min}$ are respectively the upper bound and lower bound of all possible nonzero bottom-up filter inputs T_j's under perfect learning (see Proposition 5.1).

III. Main Results

We start with the following technical Lemma, which plays an important role about the dynamical behaviors of the network.

Lemma 3.1: Consider the system

$$\begin{aligned}\epsilon\frac{dx_j}{dt} &= -x_j + [1 - Ax_j][g(x_j) + T_j] \\ &\quad - [B + Cx_j]\sum_{k\neq j, k\in\Lambda} g(x_k) \\ j &\in \Lambda = \{1, \dots, n\}\end{aligned} \tag{22}$$

where $\epsilon > 0$ and all T_j are constants, (A, B, C) and g satisfy conditions (14) and (17). Assume that there exists $J \in \Lambda$ such that $T_J/(1 + T_J + C) \leq \theta, 0 \leq T_j < T_J$ and $0 \leq x_j(0) \leq x_J(0) < \theta$ for all $j \neq J$. Then we have the following conclusions.

a) If $T_J/(1 + T_J) > \theta$, then we have the following results:
   i) For all $j \neq J, x_j(t) < \theta$ when $t \geq 0$, and
   $$\lim_{t\to\infty} x_j(t) = \frac{T_j}{1 + T_j + C};$$
   ii) $x_J(t)$ is increasing for $t \geq 0$, and
   $$\lim_{t\to\infty} x_J(t) = \frac{1 + T_J}{2 + T_J};$$
   iii) For any given $\eta \in [\theta, (1 + T_J)/(2 + T_J))$, there exists $T > 0$ such that $x_J(t) < \eta$ when $t < T$ and $x_J(t) \geq \eta$ when $t \geq T$. Moreover, we can make $T < 1$ if ϵ is sufficiently small.

b) If $T_J/(1 + T_J) \leq \theta$, then for all $j \in \Lambda$ and $t \geq 0, x_j(t) < \theta$.

Proof:

a) Note that $g(x_j(t)) = 0$ if $x_j(t) < \theta$. Therefore, when $x_j(t) < \theta$ for all $j \in \Lambda$, (22) becomes

$$\epsilon\frac{dx_j}{dt} = -x_j + [1 - x_j]T_j. \tag{23}$$

Under the assumption that $0 \leq x_j(0) \leq x_J(0) < \theta$ and $0 \leq T_j < T_J$ for $j \neq J$, we can solve linear system (23) and obtain that $x_j(t) < x_J(t)$ for $j \neq J$ as long as $x_k(s) < \theta$ for all $s \in [0, t)$ and $k \in \Lambda$.

Note that for any $j \in \Lambda$, the unique equilibrium of (23) is $T_j/(1 + T_j)$. Since $T_J/(1 + T_J) > \theta, x_J(t)$ increases until its value exceeds θ. Therefore, there exists $t^o > 0$ such that $x_J(t^o) = \theta$ and $x_J(t) < \theta$ for $t \in [0, t^o)$. Therefore, we have $x_j(t) < \theta$ for all $j \neq J$ and $t \in [0, t^o]$.

Recall that $g(\theta) = 1$. From (23) we have, at $t = t^o$, that

$$\epsilon\frac{dx_J}{dt} = -x_J + [1 - x_J](1 + T_J) \tag{24}$$
$$\epsilon\frac{dx_j}{dt} = -x_j + [1 - x_j]T_j - Cx_j, \quad j \neq J. \tag{25}$$

From (24), $\dot{x}_J(t^o) > 0$. Therefore, there exists $\delta_J > 0$ such that $x_J(t) > \theta$ when $t \in (t^o, t^o + \delta_J)$. For $j \neq J$, from $x_j(t^o) < \theta$ and the continuity of $x_j(t)$, it follows that there exists $\delta_j > 0$ such that $x_j(t) < \theta$ for $t \in (t^o, t^o + \delta_j)$. Let $\delta_0 = \min_{k\in\Lambda}\delta_k$, then $x_J(t) > \theta$ and $x_j(t) < \theta$ for all $t \in (t^o, t^o + \delta_0)$ and $j \in \Lambda$ with $j \neq J$. Let

$$\begin{aligned} t^* = \sup\{s > t^o \colon x_J(t) > \theta \quad \text{and} \quad x_j(t) < \theta \text{ for} j \in \Lambda \\ \text{and } j \neq J, \text{for all } t \in (t^o, s)\}.\end{aligned}$$

Clearly, $t^* \geq t^o + \delta_0$. We claim that $t^* = \infty$.

By way of contradiction, we assume that $t^* < \infty$. Then $x_J(t^*) = \theta$ or $x_j(t^*) = \theta$ for some $j \in \Lambda$ with $j \neq J$. With the above definition of t^*, we have (24)–(25) hold for $t^o \leq t < t^*$. As the unique equilibrium of (24) is given by $E_J = (1 + T_J)/(2 + T_J) > \theta$, we can see that $x_J(t)$ is increasing for $t \in [t^o, t^*)$. Therefore, it is impossible that $x_J(t^*) = \theta$. On the other hand, the unique equilibrium of (25) $E_j = T_j/(1 + T_j + C) < T_J/(1 + T_J + C) \leq \theta$. For any $t \in [t^o, t^*)$, we have $\dot{x}_j(t) < 0$ if $x_j(t) > E_j$. Therefore, it is impossible that $x_j(t^*) = \theta$. This yields a contradiction. Therefore, $t^* = \infty$.

Since (24)–(25) hold for all $t \geq t^o$, we get

$$\lim_{t\to\infty} x_j(t) = E_j = \frac{T_j}{1 + T_j + C}, \ j \neq J$$
$$\lim_{t\to\infty} x_J(t) = E_J = \frac{1 + T_J}{2 + T_J}$$

and we easily verify the increasing property of $x_J(t)$. It also follows that for any given $\eta \in [\theta, E_J)$, there exists $T \geq t^o$ such that $x_J(t) < \eta$ when $t < T$, and $x_J(t) \geq \eta$ when $t \geq T$.

When ϵ is sufficiently small, from the following explicit expression of a solution of (23)

$$x_j(t) = \frac{T_j}{1 + T_j} + \left(x_j(0) - \frac{T_j}{1 + T_j}\right)e^{-\frac{1}{\epsilon}(1+T_j)t}$$

we have $t^o < 1/2$ such that $x_J(t^o) = \theta$. When $t \geq t^o$, from (23) we get

$$x_J(t) = \frac{1 + T_J}{2 + T_J} + \left(\theta - \frac{1 + T_J}{2 + T_J}\right)e^{-\frac{1}{\epsilon}(2+T_J)(t-t^o)}.$$

Therefore, for any given $\eta \in [\theta, E_J)$, if ϵ is sufficiently small, we have $t^o \leq T < 1$ such that $x_J(t) < \eta$ when $t < T$ and $x_J(t) \geq \eta$ when $t \geq T$.

b) For any $j \in \Lambda$, as $x_j(0) < \theta$, from (23) we get

$$\begin{aligned} x_j(t) &= \frac{T_j}{1+T_j} + \left(x_j(0) - \frac{T_j}{1+T_j}\right) e^{-\frac{1}{\epsilon}(1+T_j)t} \\ &< (1-\beta)\frac{T_j}{1+T_j} + \beta x_j(0) \\ &< \beta\theta + (1-\beta)\theta = \theta \end{aligned}$$

where $0 < \beta = e^{-(1/\epsilon)(1+T_j)t} \leq 1$.

This proves Lemma 3.1.

Note that an F_2 node v_j is active at time t if and only if $f_2(x_j(t)) > 0$. For every $j \neq J, x_j(t) < \theta \leq \eta$ implies that $f_2(x_j(t)) = 0$. Therefore, Lemma 3.1-(a)-(i) implies that all F_2 nodes v_j with $j \neq J$ are always inactive. Using the same argument, Lemma 3.1-(a)-(iii) implies that F_2 node v_J is activated after some finite time T. This is the winner-take-all paradigm.

Lemma 3.1 shows the winner-take-all paradigm in a simple case where all bottom-up filter inputs (T_j's) are fixed constants. The general case is addressed in the following main results.

Theorem 3.2: Consider the system of differential equations (1), (2), (4), (5) with $T_j(t)$ given by (3) and $h_{ij}(t)$ given by (8)–(12). Assume that $0 < \epsilon_1, \epsilon_2, \delta \ll 1, \gamma = O(1)$, and conditions (H1), (H2), and (H3) hold with θ and η satisfying

$$\frac{T_{\max}}{1+T_{\max}+C} < \theta < \frac{T_{\min}}{1+T_{\min}} \tag{26}$$

$$\theta \leq \eta < \frac{1+T_{\min}}{2+T_{\min}} \tag{27}$$

Assume also that there exists $J \in \Lambda_2$ such that

$$\begin{aligned} &0 \leq T_j(0) < T_J(0) \text{and } 0 \leq x_j(0) \\ &\leq x_J(0) < \theta \quad \text{for all} \quad j \neq J, \quad \text{and} \quad T_{\min} \\ &\leq T_J(0) \leq T_{\max}. \end{aligned}$$

Then when ϵ_1, ϵ_2, and δ are sufficiently small, the following results hold:

i) For $j \neq J$ and $t \geq 0, x_j(t) < \theta$ and $f_2(x_j(t)) = 0$;

ii) There exists T with $0 < T < 1$ such that $x_J(t) < \eta$ and $f_2(x_J(t)) = 0$ when $t < T$, and $x_J(t) \geq \eta$ and $f_2(x_J(t)) = 1$ when $t \geq T$;

iii) For any $i \in \Lambda_1, j \in \Lambda_2$ and $t \geq 0$,

$$h_{ij}(t) = h_{ij}(0). \tag{28}$$

(In other words, $h_{ij}(0) = 0$ implies $h_{ij}(t) = 0$ for all $t \geq 0$, and $h_{ij}(0) = 1$ implies $h_{ij}(t) = 1$ for all $t \geq 0$);

iv) For any $i \in \Lambda_1$ and $j \in \Lambda_2$ with $j \neq J, z_{ij}(t)$ and $z_{ji}(t)$ remain unchanged for all $t \geq 0$. But

$$\lim_{t\to\infty} z_{iJ}(t) = \begin{cases} 0, & \text{if } h_{iJ}(0) = 0 \\ \frac{L}{L-1+|X|}, & \text{if } h_{iJ}(0) = 1 \end{cases} \tag{29}$$

where

$$X = \{i \in \Lambda_1 | h_{iJ}(0) = 1\} \tag{30}$$

and $|X|$ denotes the number of elements of the set X, and

$$\lim_{t\to\infty} z_{Ji}(t) = f_1(I_i). \tag{31}$$

Moreover, writing $z_{ij}^{\epsilon_1,\epsilon_2,\delta}$ and $z_{ji}^{\epsilon_1,\epsilon_2,\delta}$ to indicate explicitly the dependence on $(\epsilon_1, \epsilon_2, \delta)$, we have

$$\lim_{\delta\to 0} z_{iJ}^{\epsilon_1,\epsilon_2,\delta}(1) = \begin{cases} 0, & \text{if } h_{iJ}(0) = 0 \\ \frac{L}{L-1+|X|}, & \text{if } h_{iJ}(0) = 1 \end{cases} \tag{32}$$

$$\lim_{\epsilon_1\to 0} z_{Ji}^{\epsilon_1,\epsilon_2,\delta}(1) = (1-\alpha)z_{Ji}(0) + \alpha f_1(I_i) \tag{33}$$

where $\alpha = 1 - e^{-1/\gamma}$;

v) For any $j \in \Lambda_2$, define $D_j(t) = \{i \in \Lambda_1 | l_\zeta(z_{ij}(t)) = 1\}$. Then

(va) $D_j(t) = D_j(0)$ for any $j \neq J$;

(vb) $D_J(t_2) \subseteq D_J(t_1)$ if $t_2 \geq t_1 \geq 0$;

(vc) $D_J(t) = D_J(1) = X$ for all $t \geq 1$, where X is defined in (30).

We defer the proof to Section 4.

Remark 3.3: Theorem 3.2 describes the dynamics of PART during a trial. In PART algorithm, F_2 layer (clustering layer) makes a choice by winner-take-all paradigm: The F_2 node with the largest bottom-up filter input is activated and therefore becomes the winner (a candidate to learn the input pattern), while all other F_2 nodes are inactive. (i) and (ii) in Theorem 3.2 verify that the winner-take-all paradigm is realized in the dynamics of PART. Recall that an F_2 node v_j is active at time t if and only if $f_2(x_j(t)) > 0$. (i) implies all F_2 nodes v_j with $j \neq J$ are always inactive (unless a reset process occurs). (ii) implies that F_2 node v_J is activated after some finite time T. In other words, the F_2 node (v_J) with the largest bottom-up filter input (T_J) becomes the winner after some finite time T. Therefore, (i) and (ii) implies the winner-take-all paradigm, the property (a) described in Section 1.

(iii) implies all the selective output signals remain to be constants. This is the property (b) described in Section 1. It follows from this property that the degree of match $r_j(t)(= \sum_i h_{ij}(t))$ keeps constant. This suggests an alternative implementation of PART net: reset in parallel all nonmatching F_2 nodes at $t = 0$, then select the winner from remaining F_2 nodes so that the winner must satisfy the match criterion. This will speed up the network computation since the serial search process (caused by repeated resets to successive winners in some situations, in particular, when a new pattern is input) is omitted.

Equations (32) and (33) (in (iv)) are the learning formulae used in PART algorithm, with $\alpha = 1 - e^{-1/\gamma}$ being the learning rate. This addresses the property (c) described in Section 1.

Recall that $D_j(t)$ is the set of dimensions of projected subspace associated with cluster C_j (represented by F_2 node v_j) at time t. (v) shows that the set of dimensions is nonincreasing during the learning. In particular, the resulting dimensions D_J of winner v_J after the learning trial corresponds to those F_1 nodes which are active relative to the winner v_J. This is property (d) described in Section 1. This nonincreasing property of dimensions contributes to stabilizing learning in response to arbitrary sequences of input patterns.

The following corollary gives a necessary and sufficient condition for an F_2 node v_j to be activated.

Corollary 3.4: Consider the system of differential equations (1), (2), (4), (5) with $T_j(t)$ given by (3) and $h_{ij}(t)$ given by

(8)–(12). Assume that $0 < \epsilon_1, \epsilon_2, \delta \ll 1, \gamma = O(1)$, and conditions (H1), (H2), and (H3) hold with θ and η satisfying

$$\frac{T^{\max}}{1+T^{\max}+C} < \theta < \frac{T^{\min}}{1+T^{\min}} \tag{34}$$

$$\theta \le \eta < \frac{1}{2-\theta}. \tag{35}$$

Assume also that there exists $J \in \Lambda_2$ such that $0 \le T_j(0) < T_J(0)$ and $0 \le x_j(0) \le x_J(0) < \theta$ for all $j \ne J$, and $T_J(0) \le T^{\max}$. Then when ϵ_1, ϵ_2, and δ are sufficiently small, we have the following results.

i) If $T_J(0) > \theta/(1-\theta)$, then all conclusions of Theorem 3.2 hold.

ii) If $T_J(0) \le \theta/(1-\theta)$, then for all $j \in \Lambda_2$ and $t \ge 0, x_j(t) < \theta$ and $f_2(x_j(t)) = 0$.

Proof:

i) We note that

$$\theta < \frac{T^{\min}}{1+T^{\min}}$$

implies

$$\frac{\theta}{1-\theta} < T^{\min}.$$

Therefore, we can choose $T_{\min}$ such that

$$\frac{\theta}{1-\theta} < T_{\min} < \min(T_J(0), T^{\min}).$$

Choose

$$T_{\max} = T^{\max}$$

we then have

$$\frac{T_{\max}}{1+T_{\max}+C} < \theta < \frac{T_{\min}}{1+T_{\min}}.$$

From

$$\theta \le \eta < \frac{1}{2-\theta}$$

it follows

$$\theta \le \eta < \frac{1}{2-\theta} < \frac{1+T_{\min}}{2+T_{\min}}.$$

Therefore, all conditions of Theorem 3.2 are satisfied. This proves (i).

ii) From $x_j(0) < \theta$ and the continuity of x_j, there is $t^* > 0$ such that when $t \in [0, t^*), x_j(t) < \theta$ for all $j \in \Lambda_2$. This implies $f_2(x_j(t)) = 0$ when $t \in [0, t^*)$ for all $j \in \Lambda_2$. Therefore (4) and (5) imply that $z_{ij}(t)$ and $z_{ji}(t)$ remain to be constants on $t \in [0, t^*)$. It follows that $h_{ij}(t)$ and $T_j(t)$ remain to be constants on $t \in [0, t^*)$. Noting that $T_J(0) \le \theta/(1-\theta)$ is equivalent to $T_J(0)/(1+T_J(0)) \le \theta$, using the same argument as in (b) of Lemma 3.1 we can get $x_j(t^*) < \theta$ for all $j \in \Lambda_2$. This process can be repeated until $t^* = \infty$. This proves (ii).

Remark 3.5: Although Corollary 3.4 gives a condition that

$$T_J(0) \le \frac{\theta}{1-\theta}$$

under which no F_2 node will be activated and therefore the winner-take-all paradigm fails, this condition can actually not hold under perfect learning (see Section 5 for a definition of perfect learning) based on the following reason: The second inequality of (34) and condition $T_J(0) \le \theta/(1-\theta)$ imply that

$$T_J(0) \le \frac{\theta}{1-\theta} < T^{\min}.$$

But it is impossible that $T_J(0) < T^{\min}$ under perfect learning (see Proposition 5.1).

IV. Proof of Main Theorem

The proof is technical and long, so we start with a short description of the main ideas and steps. First of all, we shall show that during a particular trial, the similarity measure (given by h_σ) between the output of an F_1 node and the corresponding component of the template of a target F_2 node remains to be a constant (persistence), this is the essential part in order to maintain the stability of the learning and categorizing process for each input. With this in mind, then the description of the dynamics in the F_2 layer (that decides the candidate for a given input) becomes relatively easy due to the "competitive exclusion principle": the total selective output signals from all F_1 nodes to all F_2 nodes have been decided already and thus the competitive nature of the F_2 layer selects the candidate based on the winner-take-all paradigm. This is (i) and (ii) of Theorem 3.2: only one node will be activated after a finite time. Once this is shown, the equations for the updating of the bottom-up and top-down weights become simple and decoupled ordinary differential equations that can be explicitly solved.

What makes the proof technically difficult is that we have to prove the "similarity measure persistence" and the "competitive exclusion principle" in two steps: we first show the "similarity measure persistence" holds true until F_2 layer selects its winner, and then we have to show simultaneously that the winner remains to be a winner and the similarity measures remain to be constants, and we do so by way of contradiction: violation of one of the above statements at some point will lead to a contradiction.

In order to prove Theorem 3.2, we need the following lemma.

Lemma 4.1: Consider (1) and assume f_1 satisfies (H2). Then for any given ξ, if ϵ_1 is sufficiently small then for $t \ge 0$

$$h_\sigma(|f_1(x_i(t)) - \xi|) = h_\sigma(|f_1(x_i(0)) - \xi|). \tag{36}$$

Proof: For all $t \ge -1$, from (1) it follows that

$$x_i(t) = I_i + (x_i(-1) - I_i)e^{-\frac{1}{\epsilon_1}(t+1)}. \tag{37}$$

We distinguish several cases.

Case a: $|f_1(I_i) - \xi| < \sigma$. Let $\sigma_0 = \sigma - |f_1(I_i) - \xi|$. If ϵ_1 is sufficiently small, then

$$|x_i(t) - I_i| = |x_i(-1) - I_i|e^{-\frac{1}{\epsilon_1}(t+1)} < \frac{\sigma_0}{2M}$$

for $t \ge 0$, where M is the Lipschitz constant in (15). Therefore,

$$|f_1(x_i(t)) - \xi| \le |f_1(I_i) - \xi| + |f_1(x_i(t)) - f_1(I_i)| < \sigma$$

for $t \geq 0$. Therefore,

$$\begin{aligned} h_\sigma(|f_1(x_i(t)) - \xi|) &= h_\sigma(|f_1(x_i(0)) - \xi|) \\ &= h_\sigma(|f_1(I_i) - \xi|) = 1. \end{aligned}$$

Case b: $|f_1(I_i) - \xi| > \sigma$. Using a similar argument to the above, we can show that

$$\begin{aligned} h_\sigma(|f_1(x_i(t)) - \xi|) &= h_\sigma(|f_1(x_i(0)) - \xi|) \\ &= h_\sigma(|f_1(I_i) - \xi|) = 0. \end{aligned}$$

Case c: $|f_1(I_i) - \xi| = \sigma$. We will prove the lemma when $f_1(I_i) < \xi$. The proof is similar when $f_1(I_i) > \xi$. We distinguish several subcases.

Case (ci): $x_i(-1) \geq I_i$ but $f_1(x_i(-1)) \leq \xi$. We have from (37) that $x_i(t) \geq I_i$ and $x_i(t)$ is decreasing. Therefore, $f_1(x_i(t)) \geq f_1(I_i)$ and $f_1(x_i(t)) \leq f_1(x_i(-1)) \leq \xi$. This also implies that

$$0 \leq |f_1(x_i(t)) - \xi| = \xi - f_1(x_i(t)) \leq \xi - f_1(I_i) = \sigma.$$

Therefore,

$$h_\sigma(|f_1(x_i(t)) - \xi|) = h_\sigma(|f_1(x_i(0)) - \xi|) = 1$$

for $t \geq 0$.

Case (cii): $x_i(-1) < I_i$. Clearly, $x_i(t) < I_i$ for all $t \geq -1$. Since f_1 is nondecreasing, $f_1(y) \leq f_1(I_i)$ for any $y < I_i$.

If $f_1(y) < f_1(I_i)$ for any $y < I_i$, then $f_1(x_i(t)) < f_1(I_i)$ for all $t \geq 0$. It follows that $|f_1(x_i(t)) - \xi| > \sigma$ for $t \geq 0$. Therefore,

$$h_\sigma(|f_1(x_i(t)) - \xi)|) = h_\sigma(|f_1(x_i(0)) - \xi|) = 0.$$

If there exists $y < I_i$ such that $f_1(y) = f_1(I_i)$, then when ϵ_1 is sufficiently small, we have $x_i(t) \geq y$ for $t \geq 0$. Therefore, $f_1(x_i(t)) = f_1(I_i)$ for $t \geq 0$. This implies that

$$h_\sigma(|f_1(x_i(t)) - \xi|) = h_\sigma(|f_1(x_i(0)) - \xi|) = 1$$

for $t \geq 0$.

Case (ciii): $f_1(x_i(-1)) > \xi$. Since $f_1(I_i) < \xi$, we have $f_1(I_i) < f_1(x_i(-1))$. Therefore, $x_i(-1) > I_i$ since f_1 is nondecreasing. We have from (37) that $x_i(t)$ is decreasing and tends to I_i as $t \to \infty$. Therefore,

$$f_1(I_i) \leq f_1(x_i(t)) \leq \xi$$

when ϵ_1 is sufficiently small and $t \geq 0$. This implies

$$|f_1(x_i(t)) - \xi| \leq \sigma$$

for $t \geq 0$. Therefore,

$$h_\sigma(|f_1(x_i(t)) - \xi|) = h_\sigma(|f_1(x_i(0)) - \xi|) = 1$$

for $t \geq 0$.

Summarizing the above discussions we conclude that (36) holds. This proves Lemma 4.1.

We can now give the proof of Theorem 3.2.

Proof of Theorem 3.2:

Step 1) We prove (i) & (ii). When $x_j(t) < \eta$, we have $f_2(x_j(t)) = 0$ by the definition of function f_2. From (4) and (5) it follows that when $x_j(t) < \eta$

$$z_{ij}(t) = z_{ij}(0), \quad z_{ji}(t) = z_{ji}(0)$$

for all $i \in \Lambda_1$ and $j \in \Lambda_2$. On the other hand, from Lemma 4.1, we have that when ϵ_1 is sufficiently small

$$h_\sigma(d(f_1(x_i(t)), z_{ji}(0))) = h_\sigma(d(f_1(x_i(0)), z_{ji}(0))).$$

Therefore, from (3) and (8), we have when $x_j(t) < \eta$

$$h_{ij}(t) = h_{ij}(0), \quad T_j(t) = T_j(0)$$

for all $i \in \Lambda_1$ and $j \in \Lambda_2$. Note that $T_J(0) \geq T_{\min}$ implies

$$\frac{T_J(0)}{1 + T_J(0)} \geq \frac{T_{\min}}{1 + T_{\min}} > \theta$$

and

$$0 \leq T_j(0) < T_J(0)$$

and

$$0 \leq x_j(0) \leq x_J(0) < \theta \leq \eta$$

for all $j \neq J$. Applying Lemma 3.1 we obtain that when ϵ_2 is small enough, then there exists $0 < T < 1$ such that $x_J(t) < \eta$ for $t < T, x_J(T) = \eta$, and $x_j(t) < \theta$ for all $t \leq T$ and $j \neq J$. In other words, we obtain that $f_2(x_J(t)) = 0$ for $t < T, f_2(x_J(T)) = 1$, and $f_2(x_j(t)) = 0$ for all $t \leq T$ and $j \neq J$. Therefore, when $0 \leq t \leq T, z_{ij}(t) = z_{ij}(0)$ and $z_{ji}(t) = z_{ji}(0)$ for all $j \in \Lambda_2$. From Lemma 4.1, when ϵ_1 is sufficiently small we have $h_{ij}(t) = h_{ij}(0)$ and $T_j(t) = T_j(0)$ for all $0 \leq t \leq T, i \in \Lambda_1$, and $j \in \Lambda_2$.

At t = T, we have for $j = J$,

$$\epsilon_2 \frac{dx_J}{dt} = -x_J + (1 - x_J)(1 + T_J) \tag{38}$$

$$\delta \frac{dz_{iJ}}{dt} = (1 - z_{iJ})Lh_{iJ} - z_{iJ} \sum_{k \neq i, k \in \Lambda_1} h_{kJ} \tag{39}$$

$$\gamma \frac{dz_{Ji}}{dt} = -z_{Ji} + f_1(x_i) \tag{40}$$

and for $j \neq J$

$$\epsilon_2 \frac{dx_j}{dt} = -x_j + (1 - x_j)T_j - Cx_j \tag{41}$$

$$\delta \frac{dz_{ij}}{dt} = 0 \tag{42}$$

$$\gamma \frac{dz_{ji}}{dt} = 0. \tag{43}$$

Let $\Lambda_3 = \{i \in \Lambda_1 | h_{iJ}(0) = 1\}$. Note that $h_{iJ}(T) = h_{iJ}(0)$. Then for $i \in \Lambda_3$, (39) becomes

$$\delta \frac{dz_{iJ}}{dt} = (1 - z_{iJ})L - z_{iJ} \sum_{k \neq i, k \in \Lambda_1} h_{kJ}. \tag{44}$$

Since $T_J(T) = T_J(0) \geq T_{\min}$ and $x_J(T) = \eta < (1 + T_{\min})/(2 + T_{\min})$, from (38) we have

$\dot{x}_J(T) > 0$. Therefore, there exists $\delta_J > 0$ such that $x_J(t) > \eta$ when $t \in (T, T+\delta_J)$. For $j \neq J$, from $x_j(T) < \theta$ and the continuity of $x_j(t)$, we have that there exists $\delta_j > 0$ such that $x_j(t) < \theta$ for $t \in (T, T+\delta_j)$. For $i \in \Lambda_3, h_{iJ}(T) = h_{iJ}(0) = 1$ implies $l_\zeta(z_{iJ}(T)) = 1$. It follows that $z_{iJ}(T) > \zeta$. Then from the continuity of $z_{iJ}(t)$, there exists $\delta_i > 0$ such that $z_{iJ}(t) > \zeta$ for $t \in (T, T+\delta_i)$. Let

$$\delta_0 = \min_{k \in \Lambda_2 \cup \Lambda_3} \delta_k$$

then for all $t \in (T, T+\delta_0)$

$$x_J(t) > \eta, \quad x_j(t) < \theta \quad (j \in \Lambda_2, j \neq J) \quad \text{and} \quad z_{iJ}(t) > \zeta \; (i \in \Lambda_3).$$

Let (T, t^*) be the maximal extension of $(T, T+\delta_0)$ such that

$$x_J(t) > \eta, \quad x_j(t) < \theta \quad (j \in \Lambda_2, j \neq J) \quad \text{and} \quad z_{iJ}(t) > \zeta \quad (i \in \Lambda_3).$$

We claim that $t^* = \infty$. Assume, by way of contradiction, that $t^* < \infty$. Then $x_J(t^*) = \eta$ or $x_j(t^*) = \theta$ (for some $j \in \Lambda_2$ and $j \neq J$) or $z_{iJ}(t^*) = \zeta$ (for some $i \in \Lambda_3$) (otherwise (T, t^*) can be extended further according the continuity of $x_J(t), x_j(t)$, and $z_{iJ}(t)$). Therefore, (38)–(43) hold for $T \leq t < t^*$.

From (42)–(43) and Lemma 4.1, we have $T_j(t) = T_j(0)$ when ϵ_1 is sufficiently small. Therefore, from (41),

$$x_j(t) \to \frac{T_j(0)}{1 + T_j(0) + C} < \theta.$$

Consequently for any $j \in \Lambda_2$ and $j \neq J$, it is impossible to have $x_j(t^*) = \theta$.

Now we show it is also impossible to have $x_J(t^*) = \eta$. According to (8), $h_{ij}(t) = 1$ implies

$$h_\sigma(d(f_1(x_i(t)), \quad z_{ji}(t))) = 1 \quad \text{and} \quad l_\zeta(z_{ij}(t)) = 1.$$

Note that $h_{ij}(T) = h_{ij}(0), z_{ij}(T) = z_{ij}(0)$ and $z_{ji}(T) = z_{ji}(0)$. From Lemma 4.1, when ϵ_1 is sufficiently small

$$h_\sigma(d(f_1(x_i(t)), \quad z_{Ji}(T))) = h_\sigma(d(f_1(x_i(T)), z_{Ji}(T))) = 1$$

for $i \in \Lambda_3$ and $T \leq t < t^*$. But from (40)

$$|z_{Ji}(t) - f_1(x_i(t))| \leq |z_{Ji}(T) - f_1(x_i(t))|$$

for $T \leq t < t^*$. It follows that $h_\sigma(d(f_1(x_i(t)), z_{Ji}(T))) = 1$ implies

$$h_\sigma(d(f_1(x_i(t)), z_{Ji}(t)) = 1.$$

Therefore, when ϵ_1 is sufficiently small

$$h_\sigma(d(f_1(x_i(t)), \quad z_{Ji}(t)) = 1$$

for all $i \in \Lambda_3$ and $T \leq t < t^*$. Furthermore, for $i \in \Lambda_3$ and $T \leq t < t^*$, from $z_{iJ}(t) > \zeta$ we have $h_{iJ}(t) = 1$. Therefore, (44) holds for all $i \in \Lambda_3$ and $T \leq t < t^*$.

Note that $\sum_{k \neq i, k \in \Lambda_1} h_{kJ}(t) \leq m - 1$, from (44) we get

$$\delta \frac{dz_{iJ}}{dt} \geq (1 - z_{iJ})L - z_{iJ}(m-1). \quad (45)$$

If there exists $i \in \Lambda_3$ such that $z_{iJ}(T) = z_{iJ}(0) \geq L/(L-1+m)$, we get from (45) that $z_{iJ}(t) \geq L/(L-1+m)$ for all $t \in (T, t^*)$. It follows that:

$$T_J(t) \geq \frac{DL}{L-1+m} \geq T_{\min}$$

for all $t \in [T, t^*)$.

If for all $i \in \Lambda_3$, $z_{iJ}(T) = z_{iJ}(0) < L/(L-1+m)$, then from (45), $z_{iJ}(t)$ is increasing, and hence $T_J(t)$ is increasing on $[T, t^*)$. Therefore

$$T_J(t) \geq T_J(T) = T_J(0) \geq T_{\min}$$

for all $t \in [T, t^*)$.

Therefore, we always have that $T_J(t) \geq T_{\min}$ for all $t \in [T, t^*)$. Consequently, from (38) we obtain, for all $t \in [T, t^*)$, that

$$\epsilon_2 \frac{dx_J}{dt} \geq -x_J + (1 - x_J)(1 + T_{\min}). \quad (46)$$

From (46) we get that, if $\eta \leq x_J(t) < (1+T_{\min})/(2+T_{\min})$, then $x_J(t)$ is increasing on $[T, t^*)$ and hence, it is impossible to have $x_J(t^*) = \eta$.

According to (45), as long as $\zeta < z_{iJ}(t) < L/(L-1+m), z_{iJ}(t)$ is increasing. Therefore, for any $i \in \Lambda_3$ it is impossible to have $z_{iJ}(t^*) = \zeta$. This leads to a contradiction. Therefore, $t^* = \infty$. This proves (i) and (ii).

Step 2) We now prove (iii). Note that for any $j \neq J, f_2(x_j(t)) = 0$. Therefore, from (4) and (5) we have $z_{ij}(t) = z_{ij}(0)$ and $z_{ji}(t) = z_{ji}(0)$. Then from Lemma 4.1, we obtain that when ϵ_1 is sufficiently small then $h_{ij}(t) = h_{ij}(0)$ for every $j \neq J$.

Using the proof in Step 1, we can show that $h_{iJ}(0) = 1$ implies $h_{iJ}(t) = 1$ for all $t \geq 0$, and $h_{iJ}(0) = 0$ implies $h_{iJ}(t) = 0$ for all $0 \leq t \leq T$. Therefore, we only need to show that $h_{iJ}(T) = 0$ implies $h_{iJ}(t) = 0$ for all $t > T$. From (8), $h_{iJ}(T) = 0$ implies either $l_\zeta(z_{iJ}(T)) = 0$ or $h_\sigma(d(f_1(x_i(T)), z_{Ji}(T))) = 0$. We now distinguish two cases.

Case A: $l_\zeta(z_{iJ}(T)) = 0$. It follows that $z_{iJ}(T) \leq \zeta$. Note that $h_{iJ}(T) = 0$. From (39) we get, at $t = T$, that

$$\delta \frac{dz_{iJ}}{dt} = -z_{iJ} \sum_{k \neq i, k \in \Lambda_1} h_{kJ}. \quad (47)$$

Note that $T_J(T) = T_J(0) \geq T_{\min} > 0$ and $h_{iJ}(T) = 0$ implies that

$$\sum_{k \neq i, k \in \Lambda_1} h_{kJ}(t) > 1.$$

Therefore, $\dot{z}_{iJ}(T) < 0$. So there exists $\delta_i > 0$ such that $z_{iJ}(t) < \zeta$ when $t \in (T, T+\delta_i)$. Let (T, t^*) be the maximal extension of (T, δ_i) contained in (T, ∞) such that $z_{iJ}(t) < \zeta$. Then (47) holds for all $t \in [T, t^*)$. We claim that $t^* = \infty$. Assume, by way of contradiction, that $t^* < \infty$. Then $z_{iJ}(t^*) = \zeta$. Otherwise, $[T, t^*)$ can be extended further by using the continuity of $z_{iJ}(t)$. On the other hand, it is impossible to have $z_{iJ}(t^*) = \zeta$ since $\dot{z}_{iJ}(t) < 0$ on $[T, t^*)$. Therefore, $t^* = \infty$. This implies $z_{iJ}(t) < \zeta$ for all $t > T$. Therefore, $h_{iJ}(t) = 0$ for all $t > T$.

Case B: $h_\sigma(d(f_1(x_i(T)), z_{Ji}(T))) = 0$. It follows that

$$d(f_1(x_i(T)), z_{Ji}(T)) > \sigma.$$

From the continuity of functions d, f_1, and z_{Ji}, we have that there exists $\delta_i' > 0$ such that $d(f_1(x_i(t)), z_{Ji}(t)) > \sigma$ for all $t \in [T, T+\delta_i']$. It follows that $h_{iJ}(t) = 0$ when $t \in [T, T+\delta_i']$. Therefore, (47) holds for all $t \in [T, T+\delta_i']$. Therefore, when δ is sufficiently small, we have $z_{Ji}(T+\delta_i') \le \zeta$. It follows that $l_\zeta(z_{iJ}(T+\delta_i')) = 0$. Using the same argument as in case A we can show that $h_{iJ}(t) = 0$ for all $t > T+\delta_i'$. Therefore, $h_{iJ}(t) = 0$ for all $t > T$. This proves (iii).

Step 3) We now prove (iv). Note that for any $j \neq J, f_2(x_j(t)) = 0$. Therefore from (4) and (5) we have $z_{ij}(t) = z_{ij}(0), z_{ji}(t) = z_{ji}(0)$ for all $t \ge 0$. From (28), we obtain that $h_{iJ}(0) = 0$ implies $h_{iJ}(t) = 0$, and $h_{iJ}(0) = 1$ implies $h_{iJ}(t) = 1$. Note that $f_2(x_J(t)) = 1$ for $t \ge T$, from (4) we have, when $t \ge T$, that

$$\delta \frac{dz_{iJ}}{dt} = (1 - z_{iJ})L - z_{iJ} \sum_{k\neq i, k\in\Lambda_1} h_{kJ}, \quad i \in X, \quad (48)$$

$$\delta \frac{dz_{iJ}}{dt} = -z_{iJ} \sum_{k\neq i, k\in\Lambda_1} h_{kJ}, \quad i \notin X. \quad (49)$$

From (5) we have, when $t \ge T$, that

$$\gamma \frac{dz_{Ji}}{dt} = -z_{Ji} + f_1(x_i). \quad (50)$$

Then (29) and (32) follow from (48) and (49). It is easy to see that (31) and (33) follows from (50) and (1).

Step 4) We now conclude the proof by verifying (v). For any $j \neq J, D_j(t) = D_j(0)$ follows from $z_{ij}(t) = z_{ij}(0)$. Clearly, $D_J(t) = D_J(0)$ when $t \le T$. Therefore, we only need to consider the case where $t \ge T$. For every $i \in \Lambda_1$ such that $l_\zeta(z_{iJ}(0)) = 0$, we have $h_{iJ}(0) = 0$. It follows that $i \notin X$. From (49), $z_{iJ}(t)$ is decreasing on $[T, \infty)$. Therefore, $l_\zeta(z_{iJ}(t)) = 0$ for all $t \ge 0$. In other words, $i \notin D_J(0)$ implies $i \notin D_J(t)$. For any $i \in \Lambda_1$ such that $l_\zeta(z_{iJ}(0)) = 1$ but $h_{iJ}(0) = 0$, from (49) we conclude that $z_{iJ}(t)$ is decreasing on $[T, \infty)$ and tends to 0 as $t \to \infty$. Therefore, there exists $t^* > T$ such that $z_{iJ}(t^*) = \zeta, z_{iJ}(t) > \zeta$ when $t < t^*$, and $z_{iJ}(t) < \zeta$ when $t > t^*$. It follows that $l_\zeta(z_{iJ}(t)) = 1$ when $t < t^*$, and $l_\zeta(z_{iJ}(t)) = 0$ when $t \ge t^*$. Moreover, if δ is sufficiently small, we have $t^* < 1$. For any $i \in \Lambda_1$ such that $l_\zeta(z_{iJ}(0)) = 1$ and $h_{iJ}(0) = 1$, from (48) we get $z_{iJ}(t) \to L/(L-1+|X|) > \zeta$. Therefore, $z_{iJ}(t) > \zeta$ for all $t \ge 0$. It follows that $l_\zeta(z_{iJ}(t)) = 1$ for all $t \ge 0$. In summary, we have shown that $D_J(t_2) \subseteq D_J(t_1)$ if $t_2 \ge t_1$. Noting that $h_{iJ}(0) = 1$ implies $l_\zeta(z_{iJ}(t)) = 1$ for all $t \ge 0, h_{iJ}(0) = 0$ and $l_\zeta(z_{iJ}(0)) = 0$ imply $l_\zeta(z_{iJ}(t)) = 0$ for all $t \ge 0$, and $h_{iJ}(0) = 0$ but $l_\zeta(z_{iJ}(0)) = 1$ implies $l_\zeta(z_{iJ}(t)) = 0$ for $t \ge 1$ when δ is sufficiently small, we conclude that $D_J(t) = D_J(1) = X$ for all $t \ge 1$ when δ is sufficiently small. This completes the proof of Theorem 3.2.

V. Discussions

Theorem 3.2 requires that there exists $J \in \Lambda_2$ such that $T_{\min} \le T_J(0) \le T_{\max}$, and $0 \le T_j(0) < T_J(0), 0 \le x_j(0) \le x_J(0) < \theta$ for all $j \neq J$. It is natural to ask whether these assumptions can be satisfied in a sequence of arbitrary trials. To answer this problem, let us introduce a new terminology. A winning F_2 nodes v_J is called to have perfect learning at a trial if either v_J is a committed node and, at the end of the trial z_{iJ}, satisfies

$$z_{iJ} = \begin{cases} 0, & \text{if } h_{iJ}(0) = 0 \\ \frac{L}{L-1+|X|}, & \text{if } h_{iJ}(0) = 1 \end{cases} \quad (51)$$

or v_J is a noncommitted node and, at the end of the trial z_{iJ}, satisfies

$$z_{iJ} = \frac{L}{L-1+m} \quad (52)$$

for all $i \in \Lambda_1 = \{1, \ldots, m\}$, where X is defined in (30). We can now state the following result.

Proposition 5.1: Assume an F_2 node v_j has perfect learning in last trial. Then at the beginning of the current trial, either

$$T_{\min} \le T^{\min} \le T_j(0) \le T^{\max} \le T_{\max}$$

or

$$T_j(0) = 0$$

for any input pattern.

Proof: Since v_j has perfect learning in last trial, $z_{ij}(0)$ satisfy (51) or (52). Noting that $h_{ij}(0)$ only take value 0 or 1, from (3) we have

$$\begin{aligned} T_j(0) &\le D \sum_i z_{ij}(0) \le D|X|L/(L-1+|X|) \\ &\le DmL/(L-1+m) = T^{\max} \le T_{\max} \end{aligned}$$

(note that $|X| \le m$).

On the other hand, $T_j(0) > 0$ implies that there exists $i \in \Lambda_1$ such that $h_{ij}(0) = 1$, and hence $l_\zeta(z_{ij}(0)) = 1$. Therefore, $z_{ij}(0) > \zeta$. Therefore, either $z_{ij}(0) = L/(L-1+|X'|)$ or $z_{ij}(0) = L/(L-1+m)$ according to (51) and (52) (Here

X' denotes the set X of last trial). Since $|X'| \le m$, we have $z_{ij}(0) \ge L/(L-1+m)$. Therefore

$$T_j(0) \ge Dz_{ij}(0) \ge DL/(L-1+m) = T^{\min} \ge T_{\min}.$$

This proves Proposition 5.1.

The assumption that $0 \le x_j(0) \le x_J(0) < \theta$ for all $j \ne J$ can be satisfied by setting the initial activations $(x_j(0))$ of all F_2 nodes to 0. This can be easily achieved by an additional mechanism which resets all F_1 and F_2 nodes to the inactive states when an input pattern is ceased to be presented to the network.

The assumption that there exists $J \in \Lambda_2$ such that $0 \le T_j(0) < T_J(0)$ for all $j \in \Lambda_2$ with $j \ne J$ is not always satisfied. When this assumption is not satisfied it is possible that more than one committed F_2 nodes are activated or no committed F_2 node is activated. In the case where no committed F_2 node is activated, a noncommitted F_2 node will be selected as the winner. In the case where more than one committed F_2 nodes are activated, one usually has to increase the vigilance gradually. At each given vigilance level, some F_2 nodes with lower degree of match are reset until either a unique F_2 node with the largest degree of match becomes the winner (when there is a unique active F_2 node with the largest degree of match) or all active F_2 nodes are reset and then a noncommitted F_2 node is selected as the winner (when there are two or more active F_2 nodes with the same largest degree of match).

The following proposition gives a sufficient condition under which it is impossible that more than one F_2 nodes are activated at the same time.

Proposition 5.2: Consider the system

$$\begin{aligned}\epsilon_2 \frac{dx_j}{dt} &= -x_j + [1 - Ax_j][g(x_j) + T_j] \\ &\quad - [B + Cx_j] \sum_{k \ne j, k \in \Lambda_2} g(x_k), \\ &\quad t \ge 0, \quad j \in \Lambda_2 \end{aligned} \tag{53}$$

where $\epsilon_2 > 0$ and all $0 \le T_j(t) \le T_{\max}, (A, B, C)$ and g satisfy conditions (14) and (17) with θ and η satisfying

$$\frac{1 + T_{\max}}{2 + T_{\max} + C} < \theta < 1 \tag{54}$$

$$\eta > \theta \tag{55}$$

where $T_{\max}$ satisfies (20). Assume that $x_j(0) < \theta$ for every $j \in \Lambda_2$. Then for every given $t \ge 0$, it is impossible to have $j_1, j_2 \in \Lambda_2$ with $j_1 \ne j_2$ such that $x_{j_1}(t) \ge \eta$ and $x_{j_2}(t) \ge \eta$.

Proof: By way of contradiction, we assume that there are $t' > 0$ and $j_1, j_2 \in \Lambda_2$ with $j_1 \ne j_2$ such that $x_{j_1}(t') \ge \eta$ and $x_{j_2}(t') \ge \eta$. Then, according to the assumption $x_j(0) < \theta$ and the continuity of $x_j(t)$, there must be t^* with $0 < t^* < t'$ such that $x_{j_1}(t) \ge \theta$ and $x_{j_2}(t) \ge \theta$ for $t^* \le t \le t'$ but either $x_{j_1}(t^*) < \eta$ or $x_{j_2}(t^*) < \eta$. Without loss of generality, we assume that $x_{j_1}(t^*) < \eta$. According to (53) and ($A = 1, B = 0, C > 0$), we have $x_j(t) \le 1$ for all $t \ge 0$ and $j \in \Lambda_2$. Note that for $t^* \le t \le t'$

$$\sum_{k \ne j_1, k \in \Lambda_2} g(x_k(t^*)) \ge g(x_{j_2}(t^*)) = 1$$

and

$$\sum_{k \ne j_2, k \in \Lambda_2} g(x_k(t^*)) \ge g(x_{j_1}(t^*)) = 1.$$

From (53) we get

$$\begin{aligned}\epsilon_2 \frac{dx_j}{dt} &= -x_j + [1 - x_j](1 + T_j) - Cx_j \sum_{k \ne j, k \in \Lambda_2} g(x_k) \\ &\le -x_j + [1 - x_j](1 + T_{\max}) - Cx_j \\ &\quad t^* \le t \le t', \quad j = j_1, j_2. \end{aligned} \tag{56}$$

Therefore, from $\theta > (1 + T_{\max})/(2 + T_{\max} + C)$ we obtain $\dot{x}_{j_1}(t) < 0$ whenever $x_{j_1}(t) \ge \theta$. In other words, $\dot{x}_{j_1}(t) < 0$ for $t^* \le t \le t'$. Note that $x_{j_1}(t^*) < \eta$. Therefore, it is impossible to have $x_{j_1}(t') \ge \eta$. This is a contradiction. This proves Proposition 5.2.

Throughout this paper, we assume that signal functions are step functions with thresholds. This reflects the on-or-off characteristic of a single neuron, and makes the PART algorithm user friendly in the sense that users are required to pick up the thresholds instead of picking up signal functions. This is clearly an approximation of the concept of similarity. In other words, whenever these signal functions are used to evaluate whether the output signal of a F_1 node is similar to the corresponding component of the template, we assume there is only "yes" or "no" answer. This is clearly an approximation of a fuzzy concept of similarity. It remains to be investigated in the future whether the same dynamical behaviors are still true or not if we use general signal functions. It is also desirable to extend PART to deal with categorical and fuzzy data.

We note that, similar to the traditional ART systems, the "top-down" weights observe a fast-commit/slow-recode dynamics. However, in PART these weights track the inputs (due to (33)) rather than tracking a matched pattern, as is characteristic of most ART systems where the matched pattern often tends to code a low dimensional projection of a set of inputs (a major task of the PART algorithm). It would be interesting to investigate how this tracking the input itself affects the convergence of top-down weight. More importantly, examining the alternative dynamic of tracking a matched pattern should be a very useful direction for future investigations. We wish to express our great appreciation for the Associate Editor for pointing out this direction, and for many valuable comments that lead to significant improvement of the manuscript.

Appendix A
Summary of PART Algorithm

This Appendix summarizes the discrete PART algorithm which is derived from the PART dynamics described in this paper. The original form of the algorithm was presented in [4].

A. F_1 *Activation and Computation of Selective Output Signals* h_{ij}

We take the signal function f_1 as the identical function $f_1(x_i) = x_i$, then at an equilibrium where $x_i = I_i, f_1(x_i) = I_i$, where $I = (I_1, \ldots, I_m)$ is an input pattern. Therefore, h_{ij} is computed by (8) with σ being a distance vigilance parameter and ζ a small threshold usually taken as 0. In particular, we have

$$h_{ij} = \begin{cases} 1 & \text{if } |f_1(x_i) - z_{ji}| \le \sigma \text{ and } z_{ij} > \zeta \\ 0 & \text{otherwise.} \end{cases} \quad \text{(A-1)}$$

B. F_2 Activation and Selection of Winner

We compute the bottom-up filter input T_j to the committed F_2 node v_j by (3) with $D = 1$ and then select the winner according to the rule as follows:

Let $\Gamma = \{T_k : F_2$ node v_k is committed and has not been reset on the current trial$\}$, then node v_J is a winner either if $\Gamma \neq \emptyset$ and $T_J = \max \Gamma$, or if $\Gamma = \phi$ and node v_J is the next noncommitted node in F_2 layer.

C. Vigilance and Reset

A winning F_2 node v_J is reset if the matching degree r_J defined by (13) is less than a vigilance parameter ρ. Otherwise, the winner passes the vigilance test and is ready to learn the input pattern.

D. Learning

For the winner v_J which has passed the vigilance test, update its bottom-up weights z_{iJ} and its top-down weights z_{Ji} as follows:

If v_J is a committed winning F_2 node, then

$$z_{iJ}^{\text{new}} = \begin{cases} L/(L-1+|X|), & \text{if } h_{iJ} = 1 \\ 0, & \text{if } h_{iJ} = 0 \end{cases} \quad \text{(A-2)}$$

$$z_{Ji}^{\text{new}} = (1-\alpha) z_{Ji}^{\text{old}} + \alpha I_i \quad \text{(A-3)}$$

where $0 \le \alpha \le 1$ is the learning rate, $|X|$ denotes the number of elements in the set $X = \{i : h_{iJ} = 1\}$. Therefore, $r_J = |X|$.

If v_J is a noncommitted F_2 node, then

$$z_{iJ}^{\text{new}} = L/(L-1+m) \quad \text{(A-4)}$$

$$z_{Ji}^{\text{new}} = I_i. \quad \text{(A-5)}$$

E. Dimensions of Projected Clusters

Each committed F_2 node v_j represents a projected cluster C_j. The set D_j of the associated dimensions of the projected cluster C_j is determined by $l_\zeta(z_{ij})$ according to the following formula:

$$\text{The dimension } i \in D_j \text{ if and only if } l_\zeta(z_{ij}) = 1. \quad \text{(A-6)}$$

F. Outlier Node

Theoretically, F_2 layer can have arbitrarily many nodes in the PART architecture, therefore it can categorize arbitrarily many input patterns. But due to the resource restriction, the number of nodes in F_2 layer has to be limited. On the other hand, in many clustering problems, there are always some data points (called outliers) which do not cluster well. Therefore, we add a special node in PART module, called outlier node. We simply put into the outlier node all data points that cannot be clustered into F_2 nodes.

The step-by-step PART algorithm is presented in detail as follows:

TABLE I
LIST OF PARAMETERS FOR THE PART ALGORITHM

PARAMETER	PERMISSIBLE RANGE	SAMPLE VALUE
L	$L > 1$	2
α	$0 \le \alpha \le 1$	0.1
ζ	$0 \le \zeta < L/(L-1+m)$	0
ρ	$1 \le \rho \le m$	NA
σ	$\sigma \ge 0$	NA

```
PART Algorithm
0. Initialization:
   Take the number $m$ of nodes in $F_1$ layer as the number of dimensions
   of input data.
   Choose the number $n$ of nodes in $F_2$ layer is much larger than the
   expected number of clusters.
   Set the internal parameters $L$, $\alpha$ and $\zeta$ and maximum iteration times $M$.
   Choose the external input parameters $\rho$ and $\sigma$.
1. Set all $F_2$ nodes as being noncommitted.
2. For each data point in input data set, do Steps 2.1–2.6.
   2.1. Compute $h_{ij}$ for all $F_1$ nodes $v_i$ and committed $F_2$ nodes $v_j$. If all
        $F_2$ nodes are noncommitted, go to Step 2.3.
   2.2. Compute $T_j$ for all committed $F_2$ nodes $v_j$.
   2.3. Select the winning $F_2$ node $v_J$. If no $F_2$ node can be selected, put
        the data point into outlier and then continue to do Step 2.
   2.4. If the winner is a committed node, compute $r_J$, otherwise go to Step 2.6.
   2.5. If $r_J \ge \rho$, go to Step 2.6, otherwise reset the winner $v_J$ and go
        back to Step 2.3.
   2.6. Set the winner $v_J$ as the committed, and update the bottom-up
        and top-down weights for winner node $v_J$.
3. Repeat Step 2 $M$ times.
4. For each cluster $C_j$ in $F_2$ layer, compute the associated dimension set
   $D_j$.
```

The time cost of PART algorithm is $O(mnNM)$, where m is the number of dimensions of data space, n is the number of F_2 nodes, N is the number of all data points and M is the number of iterations. Our simulations on high dimensional synthetic data showed that the categories in F_2 layer became stable after only a few iterations. Therefore, M is usually a small number (less than 10 in our simulations).

Table I gives the permissible range and the suggested sample values of parameters in the above PART algorithm. For large data sets, some random data points may be correlated in several dimensions. However, it is very unlikely that a large number of random data points are correlated in a large set of dimensions. Therefore, we should choose ρ large enough to eliminate the randomness, but smaller than or equal to the number of dimensions of any possible projected cluster. As we do not know in advance the numbers of dimensions of projected clusters, we can choose ρ as small as possible, but large enough to eliminate the randomness.

We now provide a small-scale example to illustrate the PART algorithm. Consider the data set $\{(1, 2, 3, 4), (2, 3, 4, 5), (3, 4, 5, 6), (4, 5, 6, 7), (6, 7, 8, 9), (1, 2, 3, 4), (3, 4, 5, 6), (2,3,4,5), (6,4,5,6), (4,2,3,1), (6,7,1,2), (4,5,6,7)\}$ of 12 points in 4-dimensional space. Note that a few points appear twice. We apply PART algorithm with $\rho = 1, \sigma = 0.5, L = 2, \zeta = 0$, and $\alpha = 0.1$. The choice of $\rho = 1$ requires two points must be close to each other with respect to at least one dimension, and the choice of $\sigma = 0.5$ is made since all numbers involved in the data set are integers. We set $m = 4$ (the dimension of data space) and $n = 12$ (the maximal number of clusters).

Step 1) $I = (1, 2, 3, 4)$. Since no F_2 node is committed yet, it is natural to select v_5 (the first node in F_2 layer) to learn the input pattern. We have bottom-up weights $z_{5(bu)}^{\text{new}} = (z_{i5}^{\text{new}})_{i=1}^4$ with

$$z_{i5}^{\text{new}} = \frac{2}{2 - 1 + 4} = \frac{2}{5}$$

and the top-down weights

$$z_{5(\text{td})}^{\text{new}} = (z_{5i}^{\text{new}})_{i=1}^4 = (1, 2, 3, 4).$$

Step 2) $I = (2, 3, 4, 5)$. In this case,

$$h_5 = (h_{15}, h_{25}, h_{35}, h_{45}) = (0, 0, 0, 0)$$

since $|I_i - z_{5i}| = 1 > 0.5 = \sigma, \quad i = 1, \ldots, 4$. We have $r_5 = 0 < 1 = \rho$ and hence we select the next noncommitted F_2 node v_6 to learn the pattern. We have

$$z_{6(\text{bu})}^{\text{new}} = (z_{i6}^{\text{new}})_{i=1}^4 = \left(\frac{2}{5}, \frac{2}{5}, \frac{2}{5}, \frac{2}{5}\right)$$
$$z_{6(\text{td})}^{\text{new}} = (z_{6i}^{\text{new}})_{i=1}^4 = (2, 3, 4, 5).$$

Similarly, we have

Step 3) $I = (3, 4, 5, 6)$. v_7 is the noncommitted F_2 node to learn the pattern

$$z_{7(\text{bu})}^{\text{new}} = \left(\frac{2}{5}, \frac{2}{5}, \frac{2}{5}, \frac{2}{5}\right)$$
$$z_{7(\text{td})}^{\text{new}} = (3, 4, 5, 6).$$

Step 4) $I = (4, 5, 6, 7)$. v_8 is the noncommitted F_2 node to learn the pattern

$$z_{8(\text{bu})}^{\text{new}} = \left(\frac{2}{5}, \frac{2}{5}, \frac{2}{5}, \frac{2}{5}\right)$$
$$z_{8(\text{td})}^{\text{new}} = (4, 5, 6, 7).$$

Step 5) $I = (6, 7, 8, 9)$. v_9 is the noncommitted F_2 node to learn the pattern

$$z_{9(\text{bu})}^{\text{new}} = \left(\frac{2}{5}, \frac{2}{5}, \frac{2}{5}, \frac{2}{5}\right)$$
$$z_{9(\text{td})}^{\text{new}} = (6, 7, 8, 9).$$

Step 6) $I = (1, 2, 3, 4)$. We have
$h_5 = (1, 1, 1, 1)$ since $|I_i - z_{5i}| = 0 < \sigma$ and $z_{i5} = 2/5 > 0$,
$T_5 = \sum_{i=1}^4 z_{i5} h_{i5} = 8/5$
$h_6 = h_7 = h_8 = h_9 = (0, 0, 0, 0)$ since $|I_i - z_{ji}| = 1 > \sigma$ for $i = 1, 2, 3, 4, j = 6, 7, 8, 9$
$T_6 = T_7 = T_8 = T_9 < T_5$.

So v_5 is the winning F_2 node. Since $r_5 = 4 \geq 1 = \rho$ we conclude that v_5 learns the input pattern

$$z_{5(\text{bu})}^{\text{new}} = \left(\frac{2}{5}, \frac{2}{5}, \frac{2}{5}, \frac{2}{5}\right)$$
$$z_{5(\text{td})}^{\text{new}} = (1 - \alpha) z_{5(\text{td})}^{\text{old}} + \alpha I$$
$$= 0.9(1, 2, 3, 4) + 0.1(1, 2, 3, 4) = (1, 2, 3, 4).$$

Similarly, we have

Step 7) $I = (3, 4, 5, 6)$. v_7 is the winning F_2 node and learns the input pattern

$$z_{7(\text{bu})}^{\text{new}} = \left(\frac{2}{5}, \frac{2}{5}, \frac{2}{5}, \frac{2}{5}\right)$$
$$z_{7(\text{td})}^{\text{new}} = (3, 4, 5, 6).$$

Step 8) $I = (2, 3, 4, 5)$. v_6 is the winning F_2 node and learns the input pattern

$$z_{6(\text{bu})}^{\text{new}} = \left(\frac{2}{5}, \frac{2}{5}, \frac{2}{5}, \frac{2}{5}\right)$$
$$z_{6(\text{td})}^{\text{new}} = (2, 3, 4, 5).$$

Step 9) $I = (6, 4, 5, 6)$. We have
$h_5 = h_6 = h_8 = (0, 0, 0, 0)$
$h_7 = (0, 1, 1, 1)$
$h_9 = (1, 0, 0, 0)$
$T_7 = \sum_{i=1}^4 z_{i7} h_{i7} = 6/5$ (using $z_{i7} = (z_{7(\text{bu})}^{\text{new}})_i$ in Step 3)
$T_9 = \sum_{i=1}^4 z_{i9} h_{i9} = 2/5$ (using $z_{i9} = (z_{9(\text{bu})}^{\text{new}})_i$ in Step 5).
$T_9 < T_7$, so v_7 is the winning F_2 node.
$r_7 = 3 \geq 1$, so v_7 learns the input pattern

$$z_{7(\text{bu})}^{\text{new}} = \left(0, \frac{2}{4}, \frac{2}{4}, \frac{2}{4}\right) = \left(0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$$
$$z_{5(\text{td})}^{\text{new}} = (1 - \alpha) z_{5(\text{td})}^{\text{old}} + \alpha I$$
$$= 0.9(3, 4, 5, 6) + 0.1(6, 4, 5, 6) = (3.3, 4, 5, 6).$$

Step 10) $I = (4, 2, 3, 1)$. We have
$h_5 = (0, 1, 1, 0)$
$h_6 = h_7 = h_9 = (0, 0, 0, 0)$
$h_8 = (1, 0, 0, 0)$
$T_5 = \sum_{i=1}^4 z_{i5} h_{i5} = 4/5$ (using $z_{i5} = (z_{5(\text{bu})}^{\text{new}})_i$ from Step 6)
$T_8 = \sum_{i=1}^4 z_{i8} h_{i8} = 2/5$ (using $z_{i8} = (z_{8(\text{bu})}^{\text{new}})_i$ from Step 4).
$T_8 < T_5$, so v_5 is the winning F_2 node.
$r_5 = 2 \geq 1$, so v_5 learns the input pattern

$$z_{5(\text{bu})}^{\text{new}} = \left(0, \frac{2}{3}, \frac{2}{3}, 0\right)$$
$$z_{5(\text{td})}^{\text{new}} = (1 - \alpha) z_{5(\text{td})}^{\text{old}} + \alpha I$$
$$= 0.9(1, 2, 3, 4) + 0.1(4, 2, 3, 1) = (1.3, 2, 3, 3.7).$$

Similarly, we have v_9 learns the input pattern $(6,7,1,2)$ and v_8 learns the input pattern $(4,5,6,7)$, and we obtain five projected clusters

$$C_5 = \{(4,2,3,1),(1,2,3,4),(1,2,3,4)\}$$
$$C_6 = \{(2,3,4,5),(2,3,4,5)\}$$
$$C_7 = \{(6,4,5,6),(3,4,5,6),(3,4,5,6)\}$$
$$C_8 = \{(4,5,6,7),(4,5,6,7)\}$$
$$C_9 = \{(6,7,8,9),(6,7,1,2)\}$$

the corresponding subspaces are $(0, 1, 1, 0)$, $(1, 1, 1, 1)$, $(0,1,1,1)$, $(1,1,1,1)$, and $(1,1,0,0)$, where 1 and 0 indicate that the corresponding dimension is used or not used respectively.

APPENDIX B
A SIMULATION EXAMPLE

This Appendix provides a simulation example on high dimensional synthetic data. We refer to [4] for more simulation examples and performance comparisons with other algorithms.

The high dimensional synthetic data are generated via the method introduced by Aggarwal *et al.* in 1999 [1]. In particular, after the number of clusters, the number of data points in each cluster and dimensions associated with each cluster have been generated, the data points for a given cluster i are generated as follows: The coordinates of the points on the noncluster dimensions are generated uniformly at random, the coordinates of the points projected onto a cluster dimension j follow a normal distribution with the mean at the respective coordinate of the anchor point, and with the variance given by $(s_{ij} \cdot r)^2$, where r is a fixed spread parameter and s_{ij} is a scale factor chosen from $[1, s]$ uniformly at random. We use $r = s = 2$ in our data generation. We refer to [4] and [1] for more details about the data generation process.

In the experiment reported below, we use an input data file with 20 000 data points in a 100-dimensional space, which has six clusters generated in 20, 24, 17, 13, 16, and 28-dimensional subspaces respectively. Table II shows the associated dimensions (the subspace where cluster is formed) and the number of data points of each cluster in the input file. The data points are presented in random order in the experiment. We report the results, with emphasis on four different aspects: number of clusters found, dimensions found, centers of clusters found, and the contingency table [20] of input clusters (original clusters) and output clusters (clusters found). Table III, Table IV, and Fig. 3 show the simulation results with $\rho = 10$ and $\sigma = 9$ (Note that since we use a different distance function in the PART algorithm which is described in Appendix A, the value of σ is much larger than the ones in [4]). Note that in the reported results we have treated as outliers the data points in the F_2 categories with very small sizes (less than 1.2% of total data points in this experiment). The simulation results show that the PART algorithm succeeds in finding the exact number of original clusters and in finding almost exact centers of all original clusters. The dimensions found are not identical to those of the original clusters, but these found dimensions are contained as subsets of the associated dimensions of original clusters. These subsets are sufficiently large so that, after a further reassignment procedure, we are able to reproduce the original clusters from the found cluster centers, the found number of clusters and the found dimensions. Table V show the reassignment results. The reassignment procedure reassigns every data point to

TABLE II
DIMENSIONS AND NUMBERS OF DATA POINTS OF INPUT CLUSTERS FOR A DATA SET IN A 100-DIMENSIONAL SPACE

Input	Dimensions	Points
1	1, 6, 10, 12, 15, 17, 31, 36, 37, 38, 45, 46, 52, 54, 58, 61, 67, 79, 81, 99	5250
2	1, 3, 5, 6, 8, 10, 11, 13, 15, 16, 17, 18, 19, 20, 27, 28, 30, 38, 59, 68, 70, 84, 85, 92	1944
3	3, 5, 10, 11, 12, 15, 16, 18, 23, 37, 43, 52, 58, 61, 74, 81, 99	4652
4	2, 3, 6, 9, 12, 13, 15, 31, 43, 51, 70, 91, 95	3006
5	1, 2, 3, 8, 9, 11, 12, 13, 16, 18, 28, 41, 48, 51, 74, 82	2676
6	1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 20, 30, 40, 51, 54, 57, 58, 74, 77, 84, 90, 99, 100	1472
Outliers	-	1000

TABLE III
DIMENSIONS AND NUMBERS OF THE DATA POINTS OF OUTPUT CLUSTERS

Output	Dimensions	Points
1	10, 12, 17, 37, 46, 58, 61, 79, 81, 99	5144
2	5, 8, 13, 15, 16, 18, 30, 70, 85, 92	1878
3	3, 5, 10, 12, 15, 23, 43, 58, 74, 99	4412
4	2, 6, 9, 12, 15, 31, 43, 70, 91, 95	2716
5	1, 2, 3, 9, 13, 16, 18, 28, 41, 74	2608
6	1, 3, 5, 9, 15, 51, 57, 58, 84, 90	1186
Outliers	-	2056

TABLE IV
CONTINGENCY TABLE OF INPUT CLUSTERS AND OUTPUT CLUSTERS. ENTRY (i, j) DENOTES THE NUMBER OF DATA POINTS THAT ARE COMMON TO OUTPUT CLUSTER i AND INPUT CLUSTER j. THE TABLE SHOWS THAT THE ORIGNAL INPUT CLUSTER STRUCTURE IS SUCCESSFULLY IDENTIFIED

Output\Input	1	2	3	4	5	6	Outliers	Sums
1	5144	0	0	0	0	0	0	5144
2	0	1878	0	0	0	0	0	1878
3	0	0	4412	0	0	0	0	4412
4	0	0	0	2716	0	0	0	2716
5	0	0	0	0	2608	0	0	2608
6	0	0	1	0	0	1185	0	1186
Outliers	106	66	239	290	68	287	1000	2056
Sums	5250	1944	4652	3006	2676	1472	1000	20000

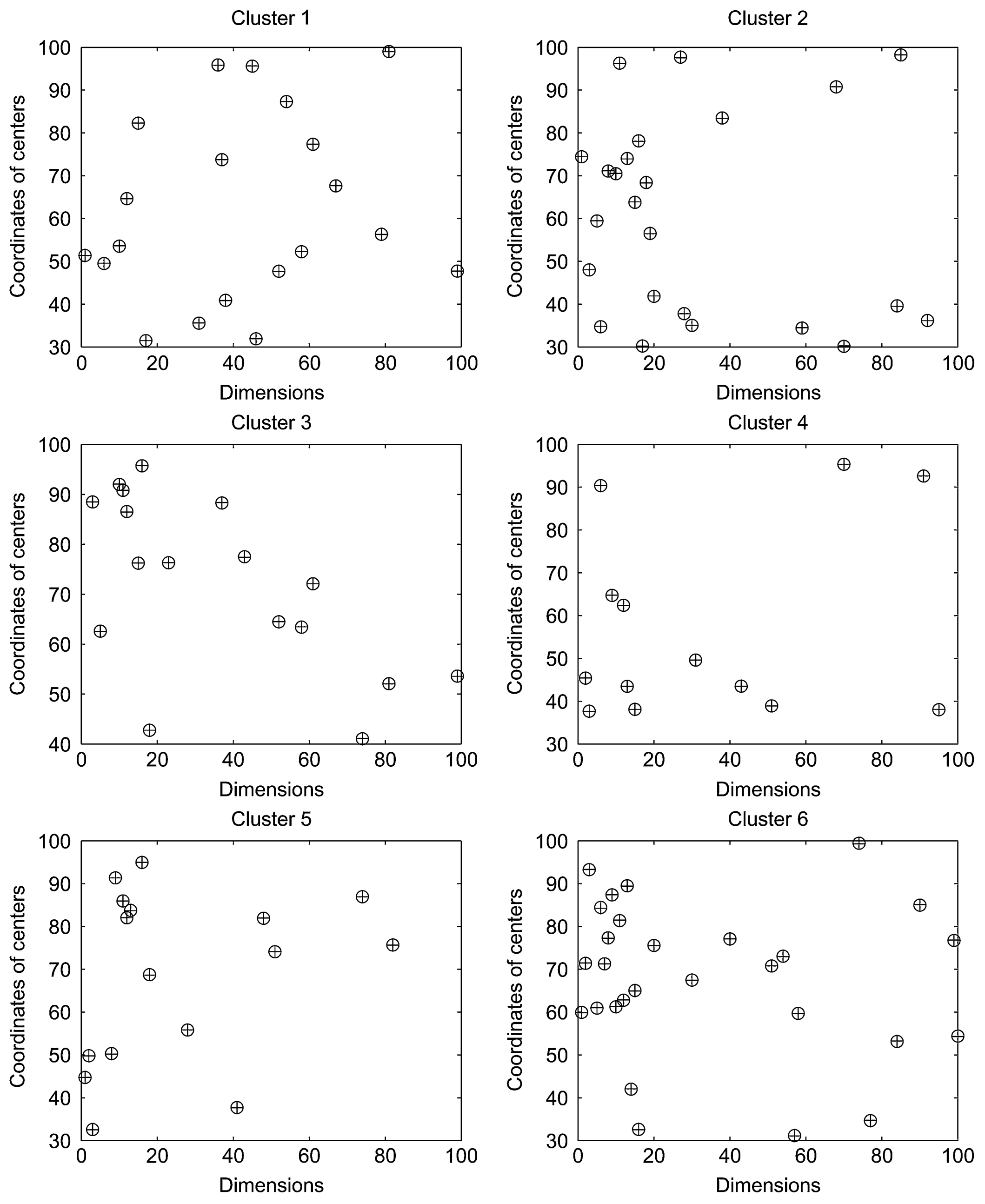

Fig. 3. Comparison of centers of output clusters with original clusters in associated dimensions. 'O' denotes the coordinate of the center of an original cluster, and '+' denotes the coordinate of the center of an output cluster, in the corresponding dimension. This figure shows that the centers of output clusters coincide with the centers of input clusters in the associated dimensions in which clusters are formed.

TABLE V
REASSIGNMENT RESULT ACCORDING TO THE FOUND CLUSTER CENTERS AND DIMENSIONS. THE RESULT SHOWS THAT THE ORIGINAL INPUT CLUSTERS ARE FULLY REPRODUCED

Output\Input	1	2	3	4	5	6	Outliers	Sums
1	5250	0	0	0	0	0	234	5484
2	0	1944	0	0	0	0	194	2138
3	0	0	4651	0	0	0	125	4776
4	0	0	0	3006	0	0	148	3154
5	0	0	0	0	2676	0	123	2799
6	0	0	1	0	0	1472	176	1649
Sums	5250	1944	4652	3006	2676	1472	1000	20000

its closest cluster center according to the Manhattan segmental distance relative to the found dimensions. Here, the Manhattan segmental distance is defined as follows: For any two points $x_1 = (x_{11}, \ldots, x_{1d}), x_2 = (x_{21}, \ldots, x_{2d})$, and a set of dimensions $D, |D| \leq d$, the Manhattan segmental distance between x_1 and x_2 relative to D is $d_D(x_1, x_2) = \sum_{i \in D} |x_{1i} - x_{2i}|/|D|$.

REFERENCES

[1] C. C. Aggarwal, C. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park, "Fast algorithms for projected clustering," in *Proc. SIGMOD'99*, 1999, pp. 61–72.
[2] R. Agrawal, J. Gehrke, D. Gunopilos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," in *SIGMOD'98*, 1998, pp. 94–105.

[3] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is 'nearest neighbors' meaningful?," in *Proc. 7th Int. Conf. Database Theory (ICDT'99)*, 1999, pp. 217–235.

[4] Y. Cao and J. Wu, "Projective ART for clustering data sets in high dimensional spaces," *Neural Networks*, vol. 15, pp. 105–120, 2002.

[5] G. A. Carpenter, "Distributed learning, recognition, and prediction by ART and ARTMAP neural networks," *Neural Networks*, vol. 10, pp. 1473–1494, 1997.

[6] G. A. Carpenter, B. L. Milenova, and B. W. Noeske, "Distributed ARTMAP: A neural network for fast distributed supervised learning," *Neural Networks*, vol. 11, pp. 793–813, 1998.

[7] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Comput. Vision, Graphics, Image Processing*, vol. 37, pp. 54–115, 1987.

[8] ——, "ART2: Self-organization of stable category recognition codes for analog input patterns," *Appl. Opt.*, vol. 26, pp. 4919–4930, 1987.

[9] ——, "ART3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures," *Neural Networks*, vol. 3, pp. 129–152, 1990.

[10] G. A. Carpenter, S. Grossberg, and J. H. Reynolds, "ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network," *Neural Networks*, vol. 4, pp. 565–588, 1991.

[11] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "ART2-A: An adaptive resonance algorithm for rapid category learning and recognition," *Neural Networks*, vol. 4, pp. 493–504, 1991.

[12] ——, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 759–771, 1991.

[13] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Networks*, pp. 698–713, June 1992.

[14] S. Dasgupta, "Learning mixtures of Gaussians," in *Proc. IEEE Symp. Foundations Computer Science*, New York, 1999, pp. 634–644.

[15] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining (KDD'96)*, Portland, OR, 1996, pp. 226–231.

[16] B. Everitt, S. Landau, and M. Leese, *Cluster Analysis*, 4th ed, London: Oxford University Press, 2001.

[17] S. Grossberg, "Adaptive pattern classification and universal recoding, I: Parallel development and coding of neural feature detectors," *Biological Cybernetics*, vol. 23, pp. 121–134, 1976.

[18] ——, "Adaptive pattern classification and universal recoding, II: Feedback, expectation, olfaction, and illusions," *Biological Cybernetics*, vol. 23, pp. 187–202, 1976.

[19] R. L. Grossman, K. Chandrika, P. Kegelmeyer, V. Kumer, and R. R. Namburu, *Data Mining for Scientific and Engineering Applications*. Boston, MA: Kluwer Academic, 2001.

[20] L. Hubert and P. Arabie, "Comparing partitions," *J. Classification*, vol. 2, pp. 193–218, 1985.

[21] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ: Prentice-Hall, 1988.

[22] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, pp. 264–323, 1999.

[23] L. Kaufman and P. Rousseeuw, *Finding Groups in Data—An Introduction to Cluster Analysis*. New York, NY: Wiley, 1990.

[24] R. T. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," in *Proc. 20th VLDB Conf.*, Santiago, Chile, 1994, pp. 144–155.

[25] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications," *Data Mining and Knowledge Discovery*, vol. 2, pp. 169–194, 1998.

[26] J. R. Williamson, "Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps," *Neural Networks*, vol. 9, pp. 881–897, 1996.

[27] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *Proc. SIGMOD'96*, pp. 103–114, 1996.

Yongqiang Cao received the B.S. degree in computational mathematics from Beijing University, Beijing, China, in 1991, the M.S. degree in systems engineering from Dalian University of Technology, Dalian, China, in 1994, and the Ph.D. degree in applied mathematics from York University, Toronto, Canada, in 2003.

Currently, he is a NSERC Postdoctoral Fellow in the Department of Cognitive and Neural Systems at Boston University, Boston, MA. His primary research interests include neural networks, pattern recognition, data mining, and biological and machine vision.

Jianhong Wu received the B.Sc. degree in computational mathematics and the M.Sc. and Ph.D. degrees in applied mathematics from Hunan University, Hunan, China, in 1982, 1984, and 1987.

He is currently a full Professor of Mathematics and a Canada Research Chair in Applied Mathematics (Tier I) at York University, Canada. He is one of the founding members for the Laboratory for Industrial and Applied Mathematics. His current research interests include nonlinear dynamics, delay differentil equations, neural networks for pattern recognition and associative memory, mathematical biology and empdemiology.