5-2019

Adaptive Resonance Theory (ART) for social media analytics

Lei MENG
Nanyang Technological University

Ah-Hwee TAN
Singapore Management University, ahtan@smu.edu.sg

Donald C. II WUNSCH
Missouri University of Science and Technology

Chapter 3
Adaptive Resonance Theory (ART) for Social Media Analytics

Abstract This chapter presents the ART-based clustering algorithms for social media analytics in detail. Sections 3.1 and 3.2 introduce Fuzzy ART and its clustering mechanisms, respectively, which provides a deep understanding of the base model that is used and extended for handling the social media clustering challenges. Important concepts such as vigilance region (VR) and its properties are explained and proven. Subsequently, Sects. 3.3–3.7 illustrate five types of ART variants, each of which addresses the challenges in one social media analytical scenario, including automated parameter adaptation, user preference incorporation, short text clustering, heterogeneous data co-clustering and online streaming data indexing. The content of this chapter is several prior studies, including Probabilistic ART [15] (©2012 IEEE. Reprinted, with permission, from [15]), Generalized Heterogeneous Fusion ART [20] (©2014 IEEE. Reprinted, with permission, from [20]), Vigilance Adaptation ART [19] (©2016 IEEE. Reprinted, with permission, from [19]), and Online Multimodal Co-indexing ART [17] (http://dx.doi.org/10.1145/2671188.2749362).

3.1 Fuzzy ART

Fuzzy ART, as briefed in Sect. 2.1.11, is featured in ART variants by its fuzzy operators and complement coding. It is the base model for the ART-based algorithms proposed in this book. This section presents the Fuzzy ART algorithm in detail.

3.1.1 Clustering Algorithm of Fuzzy ART

The architecture of Fuzzy ART (Fig. 3.1) consists of input field F_1 for receiving the input patterns and category field F_2 for the clusters. The generic network dynamics of Fuzzy ART are described as follows.

L. Meng et al., *Adaptive Resonance Theory in Social Media Data Clustering*, Advanced Information and Knowledge Processing,
https://doi.org/10.1007/978-3-030-02985-2_3

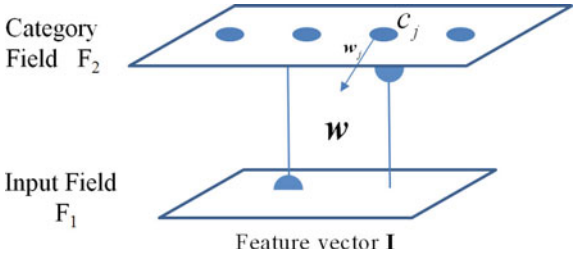

Fig. 3.1 Fuzzy ART architecture. ©2016 IEEE. Reprinted, with permission, from [19]

Input vectors: Let $\mathbf{I} = \mathbf{x}$ denote the input pattern in input field F_1. *Min-max normalization* is adopted to make the feature values of $\mathbf{I}$ be in [0, 1]. With complement coding [4], $\mathbf{x}$ is further concatenated with its complement vector $\bar{\mathbf{x}}$ such that $\mathbf{I} = [\mathbf{x}, \bar{\mathbf{x}}]$.
Weight vectors: Let $\mathbf{w}_j$ denote the weight vector associated with the jth cluster c_j ($j = 1, ..., J$) in category field F_2.
Parameters: The Fuzzy ART's dynamics are determined by choice parameter $\alpha \in (0, 0.01]$, learning parameter $\beta \in (0, 1]$ and vigilance parameter $\rho \in (0, 1)$.

The clustering process of Fuzzy ART has three key steps:

1. **Category Choice**: For each input pattern $\mathbf{I}$, Fuzzy ART calculates the choice function for all the clusters in category field F_2 and selects the most suitable cluster (winner) c_{j^*}, which has the largest value. The choice function for the jth cluster c_j is defined by

$$T_j = T(c_j, \mathbf{I}) = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|}, \tag{3.1}$$

   where the fuzzy AND operation $\wedge$ is defined by $(\mathbf{p} \wedge \mathbf{q})_i \equiv \min(p_i, q_i)$, and the norm $|.|$ is defined by $|\mathbf{p}| \equiv \sum_i p_i$.
2. **Template Matching**: The similarity between input pattern $\mathbf{I}$ and winner c_{j^*} is evaluated using a match function M_{j^*}, which is defined by

$$M_{j^*} = M(c_{j^*}, \mathbf{I}) = \frac{|\mathbf{I} \wedge \mathbf{w}_{j^*}|}{|\mathbf{I}|}. \tag{3.2}$$

   If the winner satisfies the vigilance criteria such that $M_{j^*} \geq \rho$, a resonance will occur, which leads to the prototype learning step. Otherwise, a new winner will be selected among the rest of the clusters in the category field. If no winner satisfies the vigilance criteria, a new cluster will be generated to encode the input pattern.
3. **Prototype Learning**: If c_{j^*} satisfies the vigilance criteria, its corresponding weight vector $\mathbf{w}_{j^*}$ will be updated through a learning function, defined by

$$\hat{\mathbf{w}}_{j^*} = \beta(\mathbf{I} \wedge \mathbf{w}_{j^*}) + (1 - \beta)\mathbf{w}_{j^*}. \tag{3.3}$$

3.1.2 Algorithm Analysis

As observed, Fuzzy ART adopts the choice and match functions T_j and M_{j^*} to measure the similarity between input patterns and clusters, and it uses a single ratio value, i.e. the vigilance parameter ρ, to limit the intra-cluster similarity. This unsupervised learning mechanism results in Fuzzy ART's linear time complexity of $O(n)$ and the fast convergence speed. In practice, Fuzzy ART usually achieves a reasonable clustering result in just the first epoch, and the cluster structure becomes stable in a few epochs.

Below is a proof of the learning mechanism of Fuzzy ART.

Property 3.1 *Using the category choice and template matching functions, each input pattern is categorized into the cluster with the best matching feature distribution.*

Proof Equation (3.1) shows that the similarity is calculated by the ratio of the intersection $|\mathbf{I} \wedge \mathbf{w}_j|$ and the corresponding cluster prototype $|\mathbf{w}_j|$. If the feature vector is interpreted using a histogram, the most similar feature distribution produces the largest value of $\frac{|\mathbf{I} \wedge \mathbf{w}_j|}{|\mathbf{w}_j|}$. The choice function in this way measures to which degree the cluster c_j is a subset of the input pattern $\mathbf{I}$. Thus, the category choice procedures select the cluster from which the feature distribution is the most similar to that of the input pattern.

Subsequently, the template matching procedure defined by Eq. (3.2) evaluates if the selected winner matches well with the feature distribution of the input pattern, controlled by the vigilance parameter ρ. With a reasonable setting of ρ, the clusters that do not match the feature distribution of the input pattern are rejected.

If all the existing categories are not fit for the input pattern, a new cluster is generated, and the prototypes are set by the features of the input pattern. In this way, each input pattern will be grouped into the best matching cluster.

Property 3.2 *The learning function defined by Eq. (3.3) incrementally identifies the key features from the input patterns.*

Proof The learning function defined by Eq. (3.3) consists of two components $\mathbf{I} \wedge \mathbf{w}_{j^*}$ and $\mathbf{w}_{j^*}$. The first component is the intersection between the input pattern and the cluster prototype, and the second one is the cluster prototype. It has been observed that whatever the value of the learning rate β, the values of the new cluster prototype, for each element of the weight feature vector $\mathbf{w}_{j^*,i}$, will not exceed the old one, making $|\hat{\mathbf{w}}_{j^*}| \leq |\mathbf{w}_{j^*}|$. That is, if the elements of the feature vector are inconsistent in values, the prototype learns a small value. In this way, the cluster prototype learns from the input pattern by suppressing the inconsistent features while preserving the key and consistent ones.

3.2 Geometric Interpretation of Fuzzy ART

This section presents an in-depth analysis of the clustering mechanism of Fuzzy ART by revealing its clustering behaviors in the feature space. The following sub-sections illustrate how the addition of complement coding changes the clustering behaviors of Fuzzy ART. The geometric interpretation of Fuzzy ART is also provided via an introduction to the concept of vigilance region (VR). Based on the above discoveries, an example is given to analyze the clustering dynamics of Fuzzy ART. Lastly, a discussion on Fuzzy ART's strengths and weaknesses in clustering is provided.

3.2.1 Complement Coding in Fuzzy ART

Complement coding [4] is employed in Fuzzy ART as a normalization method for the input patterns, which prevents cases in which the values of the weight vector of a cluster decrease to such a low level that the cluster is no longer representative of its category, and thus a set of new clusters must be generated to encode input patterns of this category; this is known as the problem of category proliferation.

However, complement coding prevents category proliferation by significantly changing the clustering mechanism of Fuzzy ART, converting the shapes of the clusters from open forms to hyper-octagons, termed vigilance regions (VRs) (see Fig. 3.2). The changes are illustrated below with mathematical proofs.

3.2.1.1 Effect of Complement Coding on Category Choice

Choice function Eq. (3.1) evaluates the degree to which the weight vector $\mathbf{w}_j$ of cluster c_j is a subset of input pattern $\mathbf{I}$. By employing complement coding in the ART learning system, weight vector $\mathbf{w}_j$ is concatenated by two parts, namely, the feature part that has the smallest value in each dimension among all the vertices of the weight hyper-rectangle and the complement part, of which the complement vector, in contrast to the feature part, has the largest value among vertices in each dimension. For example, in a 2D feature space, as shown in Fig. 3.2, the feature part of weight vector $\mathbf{w}_j$ is point a, and the corresponding complement part is the complement vector of point b.

With complement coding, it can be proven that the choice function essentially evaluates the similarity between the input pattern and the weight hyper-rectangle of the selected cluster.

Property 3.3 *Given the input pattern* $\mathbf{I} = (\mathbf{x}, \bar{\mathbf{x}})$*, weight vector* $\mathbf{w}_j = (\mathbf{a}, \bar{\mathbf{b}})$ *of cluster* c_j*, and* $\alpha \approx 0$*, choice function* T_j *considers the similarities between the original input pattern* $\mathbf{x}$ *and the weight hyper-rectangle of cluster* c_j*.*

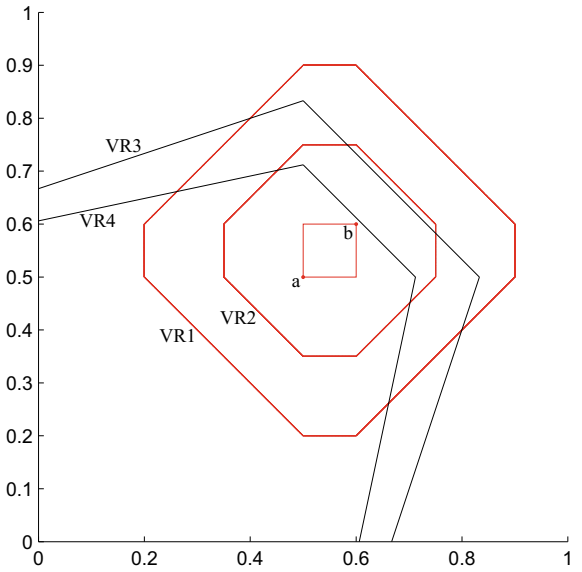

Fig. 3.2 Geometric display of a cluster and its vigilance regions (VRs) with and without complement coding in Fuzzy ART in 2D space. Without complement coding, point a denotes the weight vector of the cluster, and the corresponding VRs under vigilance parameters $\rho = 0.75$ and $\rho = 0.825$ are VR3 and VR4, respectively, as indicated by the black line. With complement coding, the weight vector is represented by the red rectangle with vertices a and b, and the corresponding VRs under vigilance parameters $\rho = 0.75$ and $\rho = 0.825$ are represented by the red octagons VR1 and VR2, respectively.

Proof

$$\begin{aligned} T_j &= \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|} \\ &= \frac{|\mathbf{x} \wedge \mathbf{a}| + |\bar{\mathbf{x}} \wedge \bar{\mathbf{b}}|}{|\mathbf{w}_j|} \\ &= \frac{|\mathbf{x} \wedge \mathbf{a}| + |\overline{\mathbf{x} \vee \mathbf{b}}|}{|\mathbf{a} + \bar{\mathbf{b}}|} \\ &= \frac{|\mathbf{a}|}{|\mathbf{a} + \bar{\mathbf{b}}|} \cdot \frac{|\mathbf{x} \wedge \mathbf{a}|}{\alpha + |\mathbf{a}|} + \frac{|\bar{\mathbf{b}}|}{|\mathbf{a} + \bar{\mathbf{b}}|} \cdot \frac{|\overline{\mathbf{x} \vee \mathbf{b}}|}{\alpha + |\bar{\mathbf{b}}|}. \end{aligned} \tag{3.4}$$

As shown in Eq. (3.4), the choice function evaluates both the degree to which $\mathbf{a}$ is a subset of $\mathbf{x}$ and the degree to which $\mathbf{x}$ is a subset of $\mathbf{b}$. The final choice value is obtained by their weighted summation, which is normalized by their respective norms.

Therefore, given $\mathbf{x}=(x_1, \ldots, x_m)$, $\mathbf{a} = (a_1, \ldots, a_m)$ and $\mathbf{b} = (b_1, \ldots, b_m)$, choice function T_j achieves its maximum for c_j when, for $\forall i \in [1, m]$, $a_i \leq x_i \leq b_i$. For

example, in Fig. 3.2, the choice function for this cluster achieves its maximum when the input pattern falls into the weight rectangle.

It may be concluded that the further the input pattern is from the weight hyper-rectangle, the lower the value that the choice function achieves for the cluster. Given that Eq. (3.1) and Eq. (3.2) share the same numerator, for $\forall \varepsilon \in (0, 1)$, $T_j = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|} = \varepsilon$ produces a VR-like hyper-octagon.

Therefore, with complement coding, the choice function evaluates the similarity of the input pattern to the weight hyper-rectangle of the selected cluster c_j.

3.2.1.2 Effect of Complement Coding on Template Matching

Match function Eq. (3.2) evaluates the degree to which the input pattern $\mathbf{I}$ is a subset of weight vector $\mathbf{w}_{j^*}$ of cluster c_{j^*}. In template matching, input pattern $\mathbf{I}$ is considered similar to the winner cluster c_{j^*} if

$$M_{j^*} = \frac{|\mathbf{I} \wedge \mathbf{w}_{j^*}|}{|\mathbf{I}|} \geq \rho. \tag{3.5}$$

The VR therefore is identified to show the extent to which an input pattern can be categorized into a specific cluster. Given the weight vector $\mathbf{w}_{j^*} = (w_1, \ldots, w_m)$ of cluster c_{j^*}, the vigilance parameter ρ, and an arbitrary input pattern $\mathbf{I} = (x_1, \ldots, x_m)$ in the Fuzzy ART system. If Fuzzy ART does not employ complement coding, Eq. (3.5) is equivalent to

$$\Sigma_{i=1}^{m} \min(x_i, w_i) - \rho \Sigma_{i=1}^{m} x_i \geq 0. \tag{3.6}$$

As shown in Fig. 3.2, when $m = 2$, Eq. (3.6) is an irregular polygon constructed by three functions and the horizontal and vertical axes.

In contrast, if Fuzzy ART employs complement coding, the number of dimensions of the feature space will be $\frac{m}{2}$. Therefore, Eq. (3.5) can be expressed as

$$\Sigma_{i=1}^{m} \min(x_i, w_i) \geq \frac{m\rho}{2}. \tag{3.7}$$

When $m = 4$, as shown in Fig. 3.2, the VR of c_{j^*} becomes a regular polygon, namely, an octagon.

3.2.2 Vigilance Region (VR)

Section 3.2.1 demonstrated the effect of complement coding on Fuzzy ART. It proved, in particular, that with complement coding the VR of a cluster becomes a hyper-octagon centered by the weight vector of the cluster, namely, the weight hyper-rectangle.

The shapes and functional behaviors of a VR depend on the use of complement coding. As shown in Fig. 3.2, with complement coding, the weight vector can be represented by a hyper-rectangle in the feature space. In this case, the VR is a hyper-octagon centered by the weight hyper-rectangle, and it shrinks as the cluster size expands (The behaviors of VR will be discussed in Sect. 3.2.3); otherwise, without complement coding, the VR is an irregular hyper-polygon with axes.

The vigilance region (VR) of a cluster, calculated from the vigilance criteria, is geometrically defined by a region associated to the cluster in the feature space, and it essentially determines how a cluster in the Fuzzy ART system recognizes similar patterns in the feature space. It also provides a geometric interpretation of the vigilance criteria in Fuzzy ART that the input patterns falling into VRs are considered to be similar to the corresponding clusters.

The following section analyzes the properties of the weight hyper-rectangle and VR of a cluster, which will be subsequently used to interpret the clustering process of Fuzzy ART in Sect. 3.2.3.

Property 3.4 *Given the weight vector* $\mathbf{w}_j = (w_1, \ldots, w_m)$ *of cluster* c_j *in the Fuzzy ART system with complement coding, the VR of* c_j *consists of* $3^{\frac{m}{2}} - 1$ *hyper-planes.*

Proof Similar to Eq. (3.4), given $\mathbf{w}_j = (a_1, \ldots, a_{\frac{m}{2}}, \bar{b}_1, \ldots, \bar{b}_{\frac{m}{2}})$, $\mathbf{I} = (\mathbf{x}, \bar{\mathbf{x}}) = (x_1, \ldots, x_{\frac{m}{2}}, \bar{x}_1, \ldots, \bar{x}_{\frac{m}{2}})$, Eq. (3.5) can by expressed as

$$\Sigma_{i=1}^{\frac{m}{2}} \min(x_i, a_i) + \Sigma_{i=1}^{\frac{m}{2}} \overline{\max(x_i, b_i)} \geq \frac{m\rho}{2}. \tag{3.8}$$

The m dimensional vector $\mathbf{w}_j$ is a hyper-rectangle in the $\frac{m}{2}$ dimensional space, and for $\forall i \in [1, \frac{m}{2}]$, $x_i \in [0, a_i) \cup [a_i, b_i) \cup [b_i, 1]$. Therefore, the feature space is divided into $3^{\frac{m}{2}}$ subsections.

Considering that Eq. (3.8) is an identical equation in the weight hyper-rectangle, the number of hyper-planes for constructing the VR is $3^{\frac{m}{2}} - 1$.

Property 3.5 *Patterns falling into the weight hyper-rectangle have the same value of match function defined by Eq. (3.2).*

Proof Given a cluster c_j and its weight vector $\mathbf{w}_j = (a_1, \ldots, a_{\frac{m}{2}}, \bar{b}_1, \ldots, \bar{b}_{\frac{m}{2}})$ and $\mathbf{I} = (x_1, \ldots, x_{\frac{m}{2}}, \bar{x}_1, \ldots, \bar{x}_{\frac{m}{2}})$ falling into the weight hyper-rectangle, it yields $\forall i \in [1, \frac{m}{2}]$, $a_i \leq x_i \leq b_i$.

In this case, according to Eq. (3.8), the value of the match function depends only on weight vector $\mathbf{w}_j$ such that $|\mathbf{I} \wedge \mathbf{w}_{j^*}| = \mathbf{w}_{j^*}$. Therefore, all the patterns in the weight hyper-rectangle have the same match value.

The situation may also be interpreted as all of those patterns having the same ℓ_1 distance to $\mathbf{a}$ and $\mathbf{b}$, as

$$\begin{aligned} |\mathbf{x} - \mathbf{a}| + |\mathbf{x} - \mathbf{b}| &= \sum_i (x_i - a_i) + \sum_i (b_i - x_i) \\ &= \sum_i (x_i - a_i + b_i - x_i) = \sum_i (b_i - a_i). \end{aligned} \tag{3.9}$$

Property 3.6 *Patterns falling into the weight hyper-rectangle of the winner do not result in the expansion of the weight hyper-rectangle during the learning step defined by Eq. (3.3).*

Proof In Property 3.4, if $\mathbf{I}$ falls into the weight hyper-rectangle of cluster c_j, $|\mathbf{I} \wedge \mathbf{w}_{j^*}| = \mathbf{w}_{j^*}$. In this case, Eq. (3.3) is equivalent to

$$\mathbf{w}_{j^*}^{(new)} = \beta \mathbf{w}_{j^*} + (1-\beta)\mathbf{w}_{j^*} = \mathbf{w}_{j^*}. \tag{3.10}$$

Therefore, weight vector $\mathbf{w}_{j^*}$ undergoes no change after encoding input pattern $\mathbf{I}$.

Property 3.7 *The weight hyper-rectangle of a cluster reflects the cluster size, which is controlled by the learning rate β.*

Proof Given input pattern $\mathbf{I} = (\mathbf{x}, \bar{\mathbf{x}})$, winner c_{j^*} and its corresponding weight vector $\mathbf{w}_{j^*} = (\mathbf{a}, \bar{\mathbf{b}})$, if $\mathbf{I}$ is categorized into c_{j^*}, $\mathbf{w}_{j^*}$ is updated according to Eq. (3.3) such that

$$\begin{aligned}
\mathbf{w}_{j^*}^{(new)} &= (\mathbf{a}^{(new)}, \bar{\mathbf{b}}^{(new)}) = \beta(\mathbf{I} \wedge \mathbf{w}_{j^*}) + (1-\beta)\mathbf{w}_{j^*} \\
&= \beta((\mathbf{x}, \bar{\mathbf{x}}) \wedge (\mathbf{a}, \bar{\mathbf{b}})) + (1-\beta)(\mathbf{a}, \bar{\mathbf{b}}) \\
&= \beta(\mathbf{x} \wedge \mathbf{a}, \overline{\mathbf{x} \vee \mathbf{b}}) + (1-\beta)(\mathbf{a}, \bar{\mathbf{b}}) \\
&= (\beta(\mathbf{x} \wedge \mathbf{a}) + (1-\beta)\mathbf{a}, \beta(\overline{\mathbf{x} \vee \mathbf{b}}) + (1-\beta)\bar{\mathbf{b}})).
\end{aligned} \tag{3.11}$$

From Eq. (3.11), it is observed that the update of weight vector $\mathbf{w}_{j^*}$ is essentially the movement of $\mathbf{a}$ and $\mathbf{b}$ towards the input pattern $\mathbf{I}$. Specifically, $\mathbf{a}$ moves toward $\mathbf{I}$ in the dimensions $\{i | x_i < a_i\}$, while $\mathbf{b}$ moves toward $\mathbf{I}$ in the dimensions $\{i | x_i > b_i\}$.

Therefore, when learning parameter β equals 1, the weight hyper-rectangle of c_{j^*} covers all the patterns in c_{j^*}, which indicates the boundaries of c_{j^*}. When $\beta < 1$, the weight hyper-rectangle expands toward the new patterns to some extent, making it unable to cover all the patterns. However, the weight hyper-rectangle may reflect the cluster size in a smaller scale.

Property 3.8 *The VR shrinks as the weight hyper-rectangle expands to control the minimum intra-cluster similarity.*

Proof As demonstrated in Property 3.4, a VR in the $\frac{m}{2}$ dimensional space is constructed by $3^{\frac{m}{2}} - 1$ functions, each of which is calculated using Eq. (3.8). Given the nature of the learning function defined by Eq. (3.3), which suppresses the values of features, following the definitions in Property 3.4, it yields $\forall i \in [1, \frac{m}{2}], a_i^{(new)} \leq a_i$ and $b_i^{(new)} \geq b_i$. Therefore, after the weight hyper-rectangle expands, the constant in the left part of Eq. (3.8) decreases. In this situation, functions in the subsections either remain the same or move toward the weight hyper-rectangle.

Interestingly, if an input pattern $\mathbf{I}$ causes the weight hyper-rectangle of a cluster c_j to expand, the function of the VR in the subsection to which $\mathbf{I}$ belongs will remain the same. Following the definitions in Property 3.4, if $\mathbf{I}$ causes the movement

of **a** in the ith dimension, it yields $x_i \leq a_i^{(new)} < a_i$. However, for this dimension, $min(x_i, a_i^{(new)}) = x_i$. Therefore, the function of the VR in that subsection is not related to the value of a_i. A similar conclusion can be drawn regarding the movement of **b**.

The shrinking of the VR can also be understood from another perspective. As the VR indicates the boundaries that the weight hyper-rectangle expands toward in all directions, when the weight hyper-rectangle expands in one direction, its distance from the VR is determined by the function in that direction, and the functions in other directions should shrink to meet this distance so that the updated VR remains a regular hyper-octagon centered by the weight hyper-rectangle.

3.2.3 Modeling Clustering Dynamics of Fuzzy ART Using VRs

Given the above properties of the weight hyper-rectangle and the VR, the clustering process of Fuzzy ART can be interpreted using a 2D example, as shown in Fig. 3.3.

Figure 3.3a depicts the evolution of a cluster in Fuzzy ART with the sequential presentation of I1(0.5, 0.6), I2(0.6, 0.5), I3(0.45, 0.45), and I4(0.43, 0.42), under the learning parameter $\beta = 1$ and the vigilance parameter $\rho = 0.825$. When the cluster has only one pattern I1, the weight rectangle R1 is situated exactly at point I1. In this case, the corresponding VR1 is a square diamond centered by I1. After the encoding of I2, R2 becomes a rectangle, and the corresponding VR becomes an octagon, which satisfies Property 3.4. During the presentation of the subsequent patterns, the weight rectangle expands to cover all of the patterns, which satisfies Property 3.7. It is notable that I4 lies directly on the edge of VR3 and, after learning from I4, VR4 overlaps with R4. Based on Properties 3.5 and 3.6, patterns falling into R4 have the same match function value, and this cluster will no longer expand. Also, the bottom-left edge of VR2-VR4, where I4 lies, never shrinks. This is because the weight rectangle always expands in this direction, which can be interpreted by the conclusion in Property 3.8.

Similarly, Fig. 3.3b shows the evolution of a cluster with the sequential presentation of I1(0.5, 0.6), I2(0.6, 0.5), I3(0.45, 0.45), I4(0.4, 0.41), and I5(0.4, 0.41) under $\beta = 0.6$ and $\rho = 0.825$. It is observed that, with a smaller learning parameter $\beta = 0.6$, R1 expands toward I2, but it cannot cover both I1 and I2 as shown in Fig. 3.3a. However, with a smaller sized R2, the corresponding VR2 covers a larger region than that depicted in Fig. 3.3a. Contrary to the behavior illustrated in Fig. 3.3a, a repeated presentation I5 of input pattern I4, as shown in Fig. 3.3b, still caused the cluster to learn. Therefore, when $\beta < 1$, the continuous presentation of the same pattern to the same cluster results in the gradual expansion of the weight rectangle of the cluster towards the input pattern. However, the cluster rectangle cannot cover that pattern due to the learning function of Fuzzy ART.

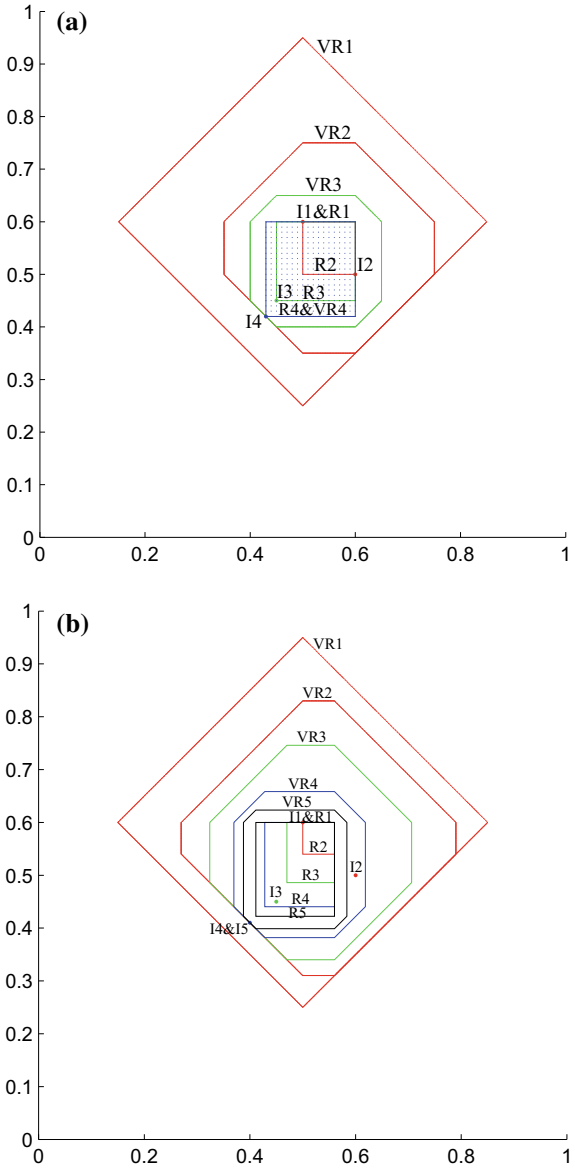

Fig. 3.3 2D example of the evolution of a cluster in Fuzzy ART under different learning parameter values **a** $\beta = 1$ and **b** $\beta = 0.6$. I1-I5 are the sequentially presented data objects, R1–R4 indicate the expansion of the cluster's weight rectangle, and VR1-VR4 indicate the corresponding VRs. © 2016 IEEE. Reprinted, with permission, from [19]

3.2.4 Discussion

The VR provides a geometric understanding of how Fuzzy ART works. As shown in Fig. 3.2, without complement coding, the VR of Fuzzy ART in a 2D space is an open

region, so the weight vector of the cluster denoted by point a may gradually move to the origin, which causes category proliferation. With complement coding, as shown in Fig. 3.3, the VR of a cluster in Fuzzy ART is a regular polygon, which shrinks as the cluster size expands. Therefore, Fuzzy ART with complement coding tends to partition the high-dimensional feature space into regions of hyper-rectangles.

The geometric interpretation of Fuzzy ART is also helpful for deducing and improving its limitations. First, given that the VR of a new cluster is usually much larger than the weight rectangle of the cluster and shrinks quickly after encoding the subsequent patterns, it may be difficult to cover a group of patterns using a single cluster, even if the VR covers all of the patterns. Secondly, a small VR may result in the generation of multiple clusters to cover a group of patterns. Thirdly, a large VR may incur an incorrect categorization of patterns, as the sequence of input patterns is unknown. Therefore, the performance of Fuzzy ART depends greatly on the value of vigilance parameter ρ, and the clustering results may differ with different sequences of input patterns.

3.3 Vigilance Adaptation ARTs (VA-ARTs) for Automated Parameter Adaptation

Clustering web multimedia data available on social websites has drawn much attention for social community discovery [16, 21], collective behavior analysis [27] and underlying topic discovery [15, 26]. However, the large-scale and complex nature of social media data raises the need to scale clustering techniques for big data and make them capable of automatically identifying data clusters with few empirical settings.

Fuzzy Adaptive Resonance Theory (Fuzzy ART) is a promising clustering using only the vigilance parameter ρ as a threshold for intra-cluster similarity. There have been studies in the literature on the adaptation or elimination of the vigilance parameter in the ART-based algorithms. But these studies introduce additional parameters, such as the number of clusters [9, 10] or the class labels for supervised learning [1, 3]. Therefore, adapting the vigilance parameter in ART under a pure clustering scenario without any additional information remains a challenge.

This section describes such an approach, called vigilance adaptation ART (VA-ART), originally investigated in [18, 19], which allows clusters to have their own ρ values and make them self-adaptable by leveraging the distribution of vigilance regions (VRs) in the feature space. Three variants of VA-ART will be discussed, including AM-ART, CM-ART, and HI-ART, which are named after their adopted methods for adapting the vigilance parameter ρ, i.e. the activation maximization rule (AMR), the confliction minimization rule (CMR), and the hybrid integration rule (HIR).

The following sub-sections discuss the three VA-ARTs and include an experimental analysis of four real-world social media datasets. It is observed that AM-ART, CM-ART, and HI-ART are more robust than Fuzzy ART to the initial vigilance

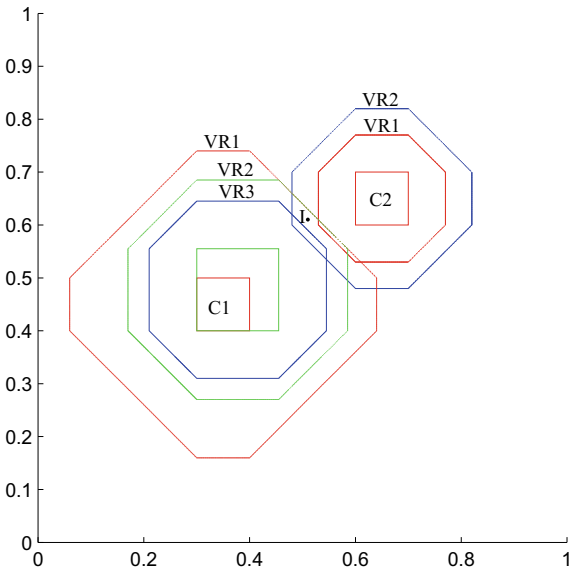

Fig. 3.4 A 2D example of how AMR adapts the vigilance parameters of two clusters in Fuzzy ART with complement coding. © 2016 IEEE. Reprinted, with permission, from [19]

value, and they usually achieve better or comparable performance and a much faster speed than state-of-the-art clustering algorithms that also do not require a predefined number of clusters, which have been discussed in Sect. 2.5.3.

3.3.1 Activation Maximization Rule

The Activation Maximization Rule (AMR) comes from the observation that, with a small vigilance value, input patterns are likely to incur resonances for the same cluster. Alternatively, a large vigilance value may cause the input patterns for all clusters in the category field to reset, requiring the creation of a new cluster. Therefore, AMR is proposed to restrain the continuous activation of the same cluster and promote the activation of clusters that usually incur resets.

Specifically, AMR adapts the vigilance parameter ρ_{j^*} of the winner c_{j^*} when

(1) **Resonance occurs**: $\hat{\rho}_{j^*} = (1 + \sigma)\rho_{j^*}$;
(2) **Reset occurs**: $\hat{\rho}_{j^*} = (1 - \sigma)\rho_{j^*}$.

The restraint parameter $\sigma \in [0, 1]$ controls the degree to which the vigilance parameter increases or decreases. With a small σ, AMR incurs small changes in the vigilance values of clusters, so the performance of ART may still depend on the initial value of the vigilance parameter. In contrast, a large σ may help to make AM-ART more robust to the initial vigilance value but may result in unstable vigilance values of clusters, which could increase the risk of pattern mis-categorization.

Figure 3.4 illustrates how AMR works. C1 and C2 are two clusters with different values of vigilance parameters. When the input pattern I is presented, C2 is the first winner. However, C2 incurs a reset due to its small VR. Subsequently, the next

winner, C1, encodes input pattern I. Without AMR, the VR of C2 remains the same, and that of C1 shrinks from VR1 to VR2. Therefore, if another input pattern close to I is presented, it will be mis-categorized to C1 again. However, with AMR, the VR of C2 expands from VR1 to VR2, and that of C1 shrinks from VR1 to VR3. In this case, C2 can successfully encode input pattern I. If the initial vigilance value is large, AMR may increase the VRs of clusters to alleviate the over-generation of clusters. Therefore, AMR may help to improve the clustering performance of Fuzzy ART when the initial vigilance value is not suitable.

Notably, AMR may also help to even out the sizes of two very close clusters by quickly shrinking the VR of the cluster that encodes more patterns, which may help to prevent the generation of small clusters and the over-generalization of cluster weights.

3.3.2 Confliction Minimization Rule

The Confliction Minimization Rule (CMR) minimizes the overlap between VRs of close clusters to produce better cluster boundaries. CMR is based on the idea that, in Fuzzy ART, the incorrect recognition of patterns is usually caused by a small vigilance value, so the VR of a cluster may cover patterns from other classes. Therefore, well-partitioned boundaries between clusters can minimize the risk of mis-categorization.

Specifically, CMR in Fuzzy ART has three key steps:

1. **Candidate Selection**: Select all winner candidates $\mathscr{C}_w = \{c_j | M_j \geq \rho\}$ in category field F_2 through the match function defined by Eq. 3.2. If no candidates are selected, CMR stops;
2. **Winner Identification**: Identify the winner c_{j^*} from all candidates through the choice function defined by Eq. 3.1 such that $j^* = \arg\max_j T_j$;
3. **Confliction Minimization**: Update the vigilance parameters of all winner candidates except the winner $\{c_j | c_j \in \mathscr{C}_w \wedge j \neq j^*\}$ using $\hat{\rho}_j = M_j + \Delta$ ($\Delta \approx 0$ is a positive value).

CMR requires Fuzzy ART to first identify all winner candidates to the input pattern through the match function. After the winner is identified in the second step, the vigilance values of all other candidates are increased to slightly higher than their respective match values. In this way, the winner is more likely to encode the subsequent input patterns that are close to the current input pattern, and the overlap between the VRs of those candidates will decrease. However, when the initial vigilance value is high, unlike AMR, CMR cannot alleviate the over-generation of clusters.

Contrary to AMR, which requires no changes to the clustering procedures of Fuzzy ART, CMR requires a change in the sequence of category choice and template matching steps. Therefore, to employ CMR in Fuzzy ART, the category choice and template matching steps should be replaced using the CMR procedures.

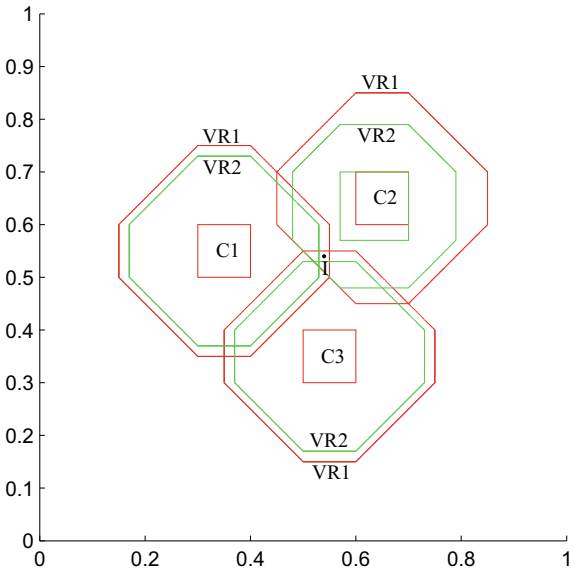

Fig. 3.5 2D example of how CMR adapts the vigilance values of clusters in order to reduce overlap between their VRs. © 2016 IEEE. Reprinted, with permission, from [19]

Figure 3.5 illustrates the ability of CMR to reduce the overlap between the VRs of clusters. C1-C3 are three clusters with corresponding VRs denoted by VR1, and I is an input pattern falling in the overlap between the VRs of all the clusters. C2 encodes the input pattern I, and its VR shrinks from VR1 to VR2. Without CMR, the overlap between all three clusters does not decrease. While with CMR, the VRs of C1 and C3 shrink from VR1 to VR2 accordingly. Therefore, the overlap undergoes significant reduction. However, the improved VRs still cannot scale the boundaries between the clusters well because Fuzzy ART cannot decide which cluster best fits the patterns falling within the overlapping areas based on existing knowledge learned from the patterns.

3.3.3 Hybrid Integration of AMR and CMR

AMR and CMR are inspired by different considerations for ART and have different mechanisms when embedding in ART, so they cannot be simply combined into a single framework. However, the ideas of AMR and CMR can be simultaneously integrated. Specifically, AMR essentially rewards the clusters that have larger choice values than the winner but incur resets due to a large vigilance value, while penalizing the clusters that incur resonances to avoid a potentially low vigilance value. In contrast, CMR minimizes the overlap between the VRs of clusters. Therefore, the objectives of both AMR and CMR could be achieved using a single framework.

The implementation of the hybrid method, called the Hybrid Integration Rule (HIR), may follow the procedures of either AMR or CMR. Following AMR, after

the winner is identified, HIR will subsequently discover all the winner candidates and apply CMR to minimize the overlap of their VRs. Following CMR, after the winner is identified, HIR will subsequently search for all the clusters with choice values that are equal to or greater than the winner and decrease their vigilance values according to AMR.

For time efficiency, HIR is implemented according to the procedures of CMR, as listed below:

1. **Candidate Selection**: Select all winner candidates $\mathscr{C}_w = \{c_j | M_j \geq \rho\}$ in category field F_2 through the match function Eq. 3.2. If no candidates are selected, for $\forall c_j \in F_2$, set $\hat{\rho}_j = (1-\sigma)\rho_j$, and HIR stops;
2. **Winner Identification**: Identify the winner c_{j^*} from all candidates through the choice function Eq. 3.1 such that $j^* = \arg\max_j T_j$. Set $\hat{\rho}_{j^*} = (1+\sigma)\rho_{j^*}$;
3. **Confliction Minimization**: Update the vigilance parameters of all winner candidates $\{c_j | c_j \in \mathscr{C}_w \wedge j \neq j^*\}$, except the winner, through $\hat{\rho}_j = M_j + \Delta$;
4. **Activation Maximization**: Search in the remaining clusters to identify the set of clusters $\mathscr{R}_c = \{c_j | c_j \in F_2 \wedge c_j \notin \mathscr{C}_w \wedge T_j \geq T_{j^*}\}$ and for $\forall c_j \in \mathscr{R}_c$, set $\hat{\rho}_j = (1-\sigma)\rho_j$.

HIR identifies all the neighboring clusters of the input pattern to minimize the overlap of their VRs while simultaneously increasing the vigilance value of the winner and decreasing those values of the clusters that have equal or larger choice values but incur resets. Therefore, HIR takes advantage of both AMR and CMR.

3.3.4 Time Complexity Analysis

Given an input pattern $\mathbf{x}$, Fuzzy ART undergoes the procedures including

1. **Complement coding**: that augments $\mathbf{I} = \mathbf{x}$ to $\mathbf{I} = [\mathbf{x}, \bar{\mathbf{x}}]$. It has a time complexity of $O(n_f)$, where n_f denotes the number of features.
2. **Cluster matching**: that performs category choice and template matching procedures through

$$T_j = T(c_j, \mathbf{I}) = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|},$$
$$M_{j^*} = M(c_{j^*}, \mathbf{I}) = \frac{|\mathbf{I} \wedge \mathbf{w}_{j^*}|}{|\mathbf{I}|},$$

   which are defined in Eqs. (3.1)–(3.2) and have a time complexity of $O(n_c n_f)$, respectively. n_c denotes the number of clusters.
3. **Prototype learning**: that either creates a new cluster c_j with a weight vector $\mathbf{w}_j = \mathbf{I}$ or updates the weights $\hat{\mathbf{w}}_{j^*}$ of the winning cluster c_{j^*} by

$$\hat{\mathbf{w}}_{j^*} = \beta(\mathbf{I} \wedge \mathbf{w}_{j^*}) + (1 - \beta)\mathbf{w}_{j^*},$$

which is defined in Eq. (3.3) and has a time complexity of $O(n_f)$.

Therefore, given n_i input patterns, the overall time complexity of Fuzzy ART is $O(n_i n_c n_f)$.

AM-ART requires the vigilance values of the selected winners to be adapted, incurring a time complexity of $O(n_c)$; CM-ART reverses the procedures of category choice and template matching and adapts the vigilance parameter values of all winner candidates, of which the time complexity is $O(n_c)$; HI-ART integrates the procedures of AM-ART and CM-ART. Therefore, the three VA-ARTs have the same time complexity as Fuzzy ART, i.e. $O(n_i n_c n_f)$.

3.3.5 Experiments

This section presents an experimental study of VA-ARTs using real-world social media datasets to illustrate (1) how the Fuzzy ART variants work, (2) how to select suitable parameters, (3) what properties the VA-ARTs have, and (4) how their clustering performance is compared with state-of-the-art ones.

3.3.5.1 Datasets

To evaluate the consistency in the performance of VA-ARTs for clustering different types of social media data, four real-world social media datasets are selected for experiments, including

- **NUS-WIDE dataset** [6] consists of 269,648 Flickr images with their raw surrounding text and ground-truth labels from 81 concepts. 10,800 images were used in total from nine classes, including dog, bear, cat, bird, flower, lake, sky, sunset, and wedding, each of which contains 1,200 images. Each image was represented as a 426-D vector by concatenating three types of visual features, including Grid Color Moment (225 features), Edge Direction Histogram (73 features) and Wavelet Texture (128 features).
- **20 Newsgroups dataset** [13] consists of approximately 20,000 messages from 20 different netnews newsgroups, each of which contains nearly 1,000 documents. 9,357 documents were collected from 10 classes, including alt.atheism, comp.graphics, comp.windows.x, rec.sport.baseball, rec.sport.hockey, sci.med, sci.space, misc.forsale, talk.politics.guns, and talk.politics.misc, from the processed MATLAB version of the 20news-bydate dataset.[1] Regarding the feature extraction, any words that occurred less than 30 times were filtered, and each doc-

[1] http://qwone.com/~jason/20Newsgroups/.

ument was represented by a bag-of-words vector of 6,823 features, weighted by the term frequency-inverse document frequency (tf-idf) algorithm.

- **Corel5K dataset** [7] consists of 4,999 images from 50 equal-sized classes. The whole dataset was utilized for experiments, and 426 visual features were extracted for image representation, as used in the NUS-WIDE dataset.
- **BlogCatalog dataset** [28] consists of the friendship network and the raw blog data (blog content, category, and tags) of 88,784 social network users. A polished version of the dataset was used as processed in [16]. Specifically, the blog content of 10,000 users from 10 equal-sized classes was collected, including travel, music, writing, sports, shopping, computers, finance, film, fashion, and books. By filtering the infrequent words, each user is represented by a 5685-D vector, of which the features are weighted by the tf-idf algorithm.

3.3.5.2 Parameter Selection

Like Fuzzy ART, the proposed VA-ARTs, i.e. AM-ART, CM-ART, and HI-ART, share the parameters α, β, and ρ. Beyond those, AM-ART has the restraint parameter σ, CM-ART has the parameter Δ, and HI-ART has both. $\alpha = 0.01, \beta = 0.6, \sigma = 0.1$, and $\Delta = 0.01$ are consistently used throughout the experiments. Such settings have been demonstrated in past efforts [16, 18, 25] to make the Fuzzy ART variants achieve robust performance.

The vigilance parameter ρ is essentially a ratio value that controls the minimum intra-cluster similarity of patterns, so its value is data-dependent. However, an empirical method [16] has shown that a suitable value of ρ typically results in the generation of a few small clusters with tens of patterns, typically 10% of the total number of the generated clusters. The experiments conducted in this section followed this method to select the initial value of ρ.

3.3.5.3 Robustness to Vigilance Parameter

This section evaluates the performance of AM-ART, CM-ART, and HI-ART on improving the robustness of Fuzzy ART to the vigilance parameter ρ. The performance was measured in terms of purity [30] and the number of clusters generated. Here, purity measures how well the algorithm recognizes the data objects of the same class, and the number of clusters measures how well the algorithm partitions the dataset with the lowest network complexity. The performance of the NUS-WIDE and 20 Newsgroups datasets were reported in Fig. 3.6, and similar observations were found in the experiments on the Corel5K and BlogCatalog datasets.

As shown in Fig. 3.6a, when $\rho < 0.4$, AM-ART, CM-ART, and HI-ART performed much better in purity and identified more clusters than Fuzzy ART. When $\rho > 0.7$, all algorithms achieved a comparable performance; however, as shown in Fig. 3.6b, the higher purity was achieved by increasing the network complexity. Meanwhile, AM-ART and HI-ART generated significantly fewer clusters than Fuzzy ART and CM-ART. These findings indicated that AMR, CMR, and HIR enabled the

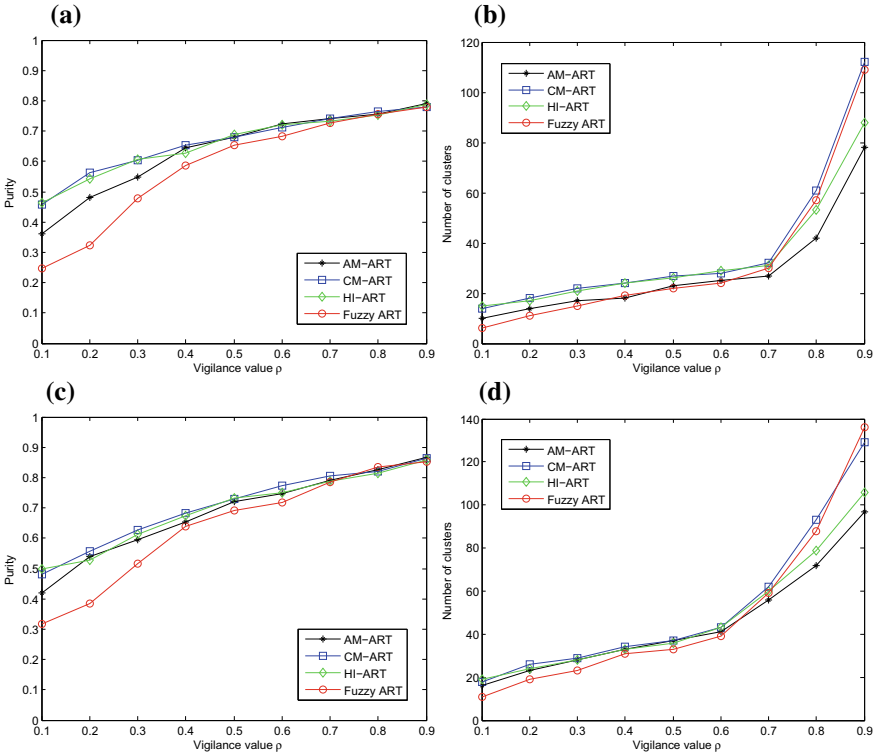

Fig. 3.6 Sensitivity of AM-ART, CM-ART, HI-ART, and Fuzzy ART to the vigilance parameter ρ measured by purity and the number of generated clusters on the NUS-WIDE (**a** and **b**) and the 20 Newsgroups (**c** and **d**) datasets. © 2016 IEEE. Reprinted, with permission, from [19]

proposed algorithms to be more robust than Fuzzy ART to the vigilance parameter, especially when the initial vigilance value is low. AMR can effectively simplify the generated cluster network when the initial vigilance value is high. More importantly, HI-ART has the advantage of both AM-ART and CM-ART, which demonstrates the viability of developing hybrid methods for vigilance adaptation. Similar findings can be observed in Fig. 3.6c, d.

A case study was further conducted to provide a deeper understanding of how the proposed algorithms work by analyzing the clusters generated by each algorithm. As shown in Fig. 3.7a, b, under $\rho = 0.2$, AM-ART, CM-ART and HI-ART identified more smaller clusters with better coherence than Fuzzy ART. These facts explain the lower performance of Fuzzy ART. In contrast, as illustrated in Fig. 3.7c, d, when $\rho = 0.9$, all algorithms generated clusters of similar quality, while HI-ART and AM-ART generated far fewer small clusters than CM-ART and Fuzzy ART. This explains why they can generate fewer clusters than the other algorithms and demonstrates the effectiveness of AMR in simplifying the network with a high vigilance value.

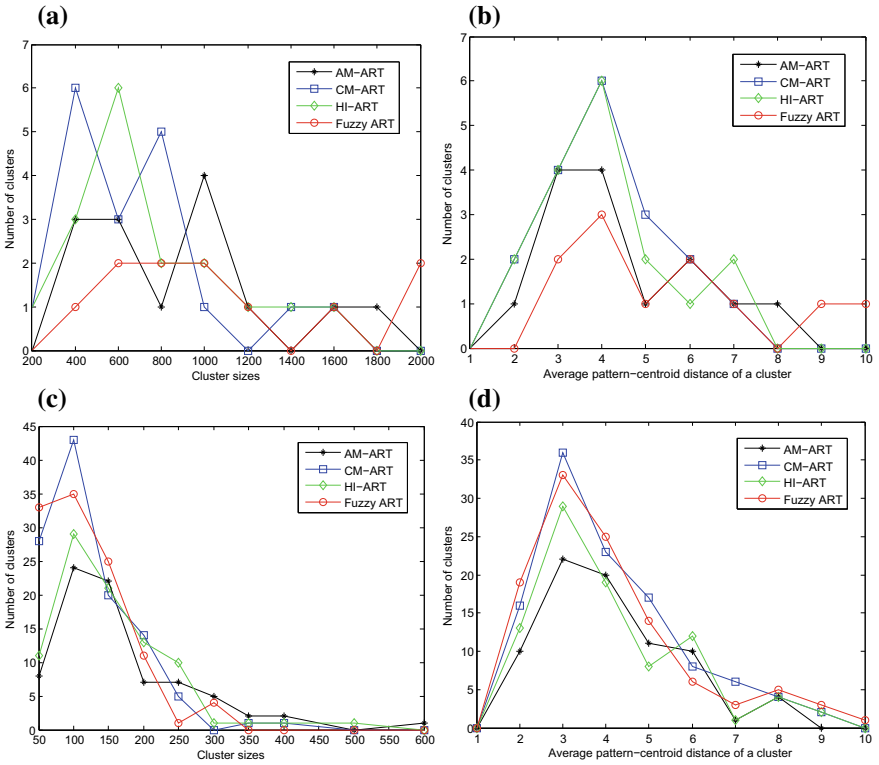

Fig. 3.7 Distributions of clusters generated by AM-ART, CM-ART, HI-ART, and Fuzzy ART on the NUS-WIDE dataset in terms of cluster size and average pattern-centroid distance under $\rho = 0.2$ (**a** and **b**) and $\rho = 0.9$ (**c** and **d**). © 2016 IEEE. Reprinted, with permission, from [19]

3.3.5.4 Convergence Analysis

This section presents a study on the convergence property of the VA-ARTs algorithms and Fuzzy ART. Their performance on the NUS-WIDE and the 20 Newsgroups datasets under $\rho = 0.7$ and 0.6, respectively, was reported, and similar findings were observed on the other datasets.

As shown in Fig. 3.8, all algorithms experienced large changes during the first six rounds. This circumstance is likely due to the generation of new clusters. CM-ART and HI-ART usually obtain comparable convergence speeds, which are faster than AM-ART and Fuzzy ART. This is because CMR promotes shrinking of the VRs of neighboring clusters by reducing their overlap, resulting in the fast stabilization of cluster assignments. AM-ART usually converges slower than Fuzzy ART, because AMR increases the vigilance value of the competitive winner candidates and decreases that of the winner so that patterns may jump across those winner candidates when they are presented multiple times. HI-ART converged faster than Fuzzy ART during the first rounds of iterations due to CMR but achieved a convergence

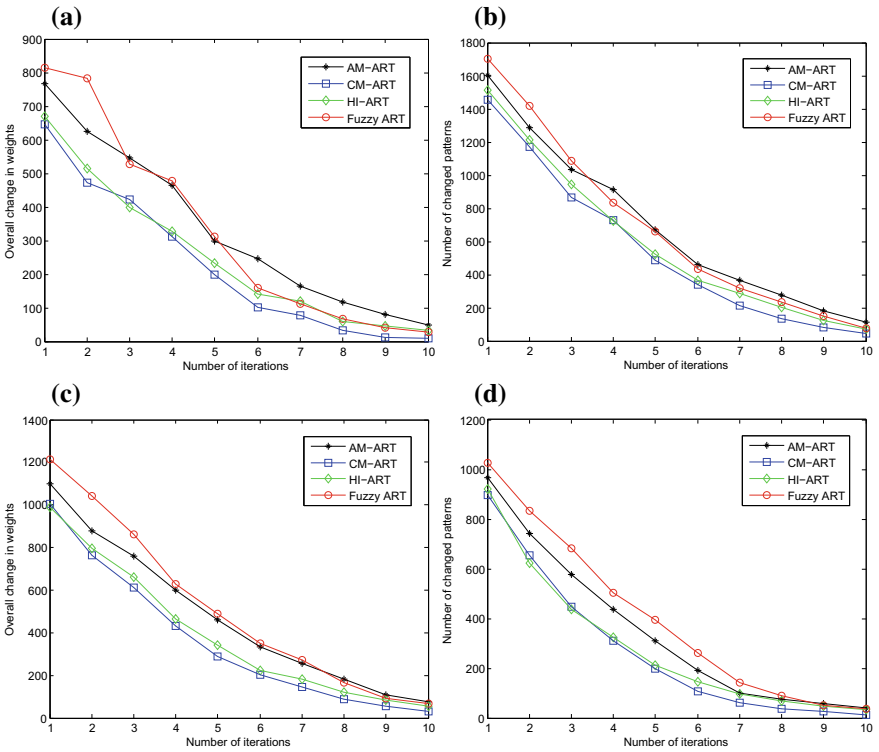

Fig. 3.8 Convergence analysis of AM-ART, CM-ART, HI-ART, and Fuzzy ART measured by the change in weights and the number of patterns moving across clusters in each iteration on the NUS-WIDE (**a** and **b**) and 20 Newsgroups (**c** and **d**) datasets. © 2016 IEEE. Reprinted, with permission, from [19]

speed similar to that of Fuzzy ART after the network became stable due to AMR. Interestingly, in contrast to its performance on the NUS-WIDE dataset, AM-ART converged faster than Fuzzy ART on the 20 Newsgroups dataset. This may be due to the larger dispersion of patterns in the feature space, which caused the increased size of the VRs to have less of an effect.

3.3.5.5 Clustering Performance Comparison

This section compares the clustering performance of AM-ART, CM-ART, and HI-ART to existing clustering approaches that also automatically identify the number of clusters in data, including DBSCAN, affinity propagation, Cluster$_{dp}$, and fuzzy ART. All algorithms were implemented in MATLAB. Hierarchical and genetic clustering approaches are not considered here because they require heavy computation and are not scalable for large-scale datasets.

Min-max normalization was applied to the datasets because the ART-based algorithms required the input values to be in the range of [0, 1]. Experimental results indicated that the normalization of data had an unobvious effect on the performance of other algorithms. To ensure a fair comparison, practical parameter tuning strategies were utilized for the algorithms. For DBSCAN, the minimum cluster size $minPts$ was determined by evaluating the sizes of the small clusters generated by Fuzzy ART under high vigilance values $\rho \in [0.7, 0.9]$. Subsequently, the method suggested in [8] was followed to select the search radius ε, namely, to plot the k-distance graph (k is the value of $minPts$) and choose the "bend" value. For Affinity propagation, the preference value p was selected using the MATLAB function "preferenceRange.m".[2] The values of $dampfact$, $convits$ and $maxits$ were first set to the suggested values by the authors and then changed with respect to the preference value p to ensure convergence. For Cluster$_{dp}$, the search radius d_c was set to the value of ε used in DBSCAN as both have the same meaning, and the cluster centers were selected from the decision graph that produced the best performance.

Three external clustering performance measures were used, including purity [30], class entropy [11] and the Rand index [29]. Purity evaluates the precision aspect, i.e., how well an algorithm recognizes patterns belonging to the same class, and a higher value indicates better performance. Class entropy evaluates the recall aspect, i.e., how well an algorithm partitions the dataset with the minimum number of clusters, and a lower value indicates better performance. The Rand index considers both aspects. Internal performance measures, such as the sum-of-squared error (SSE), were not used because they make assumptions based on cluster shapes, so they are not suitable to evaluate the performance of DBSCAN and Cluster$_{dp}$.

The performance of each algorithm was first reported under different parameter settings in terms of the Rand index on all datasets, which provides an overall picture of the performance of each algorithm. Specifically, for the ART-based algorithms, the curve of performance was plotted as a function of the vigilance parameter ρ; for DBSCAN, the curve was plotted as a function of the minimum cluster size $minPts$; for Affinity Propagation, the curve was plotted as a function of the preference value p; for Cluster$_{dp}$, the curve was plotted as a function of the search radius d_c. The other parameters of each algorithm were fixed or tuned as aforementioned so that the best performance was achieved under each condition of the functions. Additionally, for DBSCAN and the ART-based algorithms whose results may be affected by the input data sequence, the performance was the mean value obtained by repeating the experiments ten times with different sequences of patterns, while that of Affinity Propagation and Cluster$_{dp}$ was obtained in a single run. The results are shown in Fig. 3.9. To facilitate the comparison, the x-axis values of each algorithm were normalized to be in the range of [0, 1]. The performance of the ART-based algorithms typically increased with respect to the increase in the vigilance value ρ, which indicated that better performance can be achieved by setting a higher intra-cluster similarity threshold to some extent. However, a vigilance value that is too high would result in a deteriorated performance by the high network complexity, as shown in Fig. 3.9a, b.

[2] http://genes.toronto.edu/index.php?q=affinity%20propagation.

Furthermore, HI-ART and CM-ART usually outperform AM-ART and Fuzzy ART when the vigilance value is low, consistent with this book's findings as presented in Sect. 3.3.5.3. It is notable that Fuzzy ART achieved a very low performance when $\rho < 0.3$, which was caused by the fact that all patterns in the Corel5K dataset were clustered into a single cluster. In contrast, AM-ART achieved an improved performance on this case while CM-ART and HI-ART had a big improvement over Fuzzy ART. Compared with the ART-based algorithms, DBSCAN could achieve a stable performance when the values of $minPts$ were near the best setting. However, it was observed that the best parameter value varied with different datasets, and the best performance of DBSCAN was typically lower than these achieved by the ART-based algorithms. Affinity Propagation could perform comparably to the ART-based algorithms and achieved a more stable performance under different parameter settings, especially in Fig. 3.9a, d. However, the performance of Affinity Propagation could fluctuate a lot, as shown in Fig. 3.9b, c, making it difficult to manually select the best parameter settings. Cluster$_{dp}$ typically performed the worst among all algorithms.

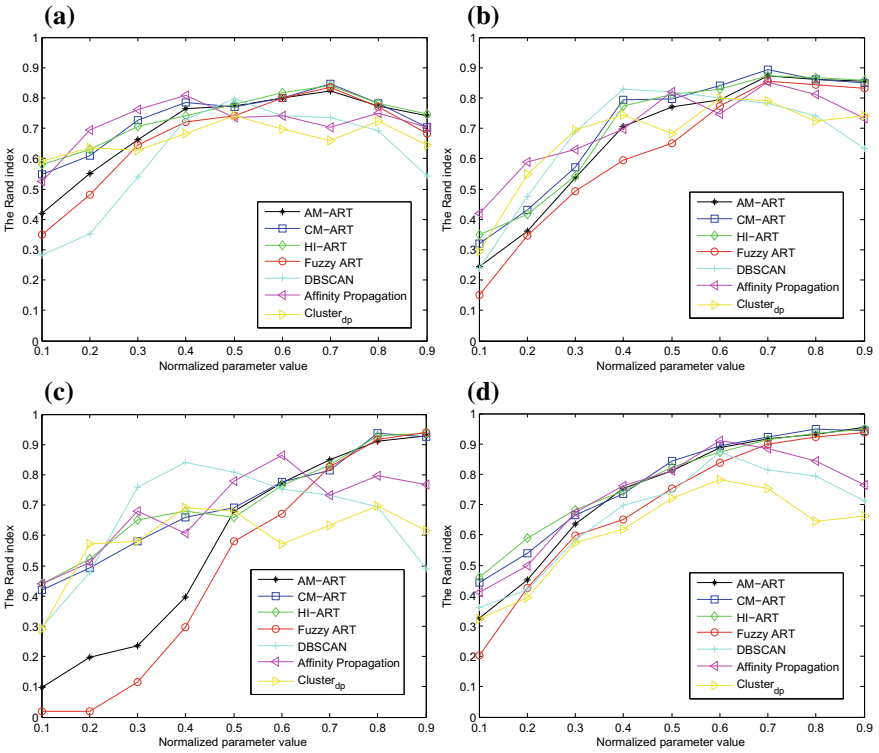

Fig. 3.9 Clustering performance comparison of the proposed algorithms and four baseline algorithms under different parameter settings measured by the Rand index on the **a** NUS-WIDE, **b** 20 Newsgroups, **c** Corel5K and **d** BlogCatalog datasets. © 2016 IEEE. Reprinted, with permission, from [19]

Although a fairly stable performance was achieved in Fig. 3.9a, its best performance is almost 10% lower than those achieved by other algorithms. This could be caused by the noisy features of the patterns so that the neighboring relationship between patterns belonging to the same cluster may not be well-reflected by the calculated distance. Additionally, Cluster$_{dp}$ suffered from the problem of selecting qualified cluster centers from the decision graph on all datasets. In the experiments conducted, almost all patterns were in a mass while few of them satisfied the requirements of having many neighbors and a long distance to other more qualified cluster centers.

A case study was further conducted on the best clustering results of all algorithms achieved in Fig. 3.9 by comparing their performances in terms of purity, class entropy, and the Rand index. For DBSCAN and the ART-based algorithms, the means and standard derivations obtained from ten runs were reported, and their differences were further measured by the t-test. As shown in Table 3.1, the proposed CM-ART and AM-ART typically obtained the best performances across all datasets in terms of purity and the Rand index, which was usually significantly better than that achieved by DBSCAN, Affinity Propagation, and Cluster$_{dp}$ at the significant level $p = 0.001$. Fuzzy ART usually performs comparably to the proposed algorithms and exhibits the best performance on the Corel5K dataset in terms of purity and the Rand index. However, it did not perform significantly differently than CM-ART at the significant level $p = 0.1$. It was observed that Affinity Propagation and DBSCAN usually obtain the best performance of class entropy, which indicated that the ART-based algorithms may have to generate more clusters to guarantee a higher quality of clusters. This may be due to the irregular distributions of patterns in the feature space resulted by the noisy patterns. Furthermore, the proposed AM-ART, CM-ART, and HI-ART usually achieve a performance that is not significantly different to the best performance in terms of class entropy at the significant level $p = 0.05$. The above findings revealed that the proposed algorithms usually perform better than or comparable to the existing algorithms in terms of purity and the Rand index, and also perform reasonably well in terms of class entropy.

3.3.5.6 Case Study on Noise Immunity

The noisy and diverse nature of social media data raises a challenge for the robustness of clustering algorithms to noise. Here, noise is not only defined by the noisy patterns that are isolated from clusters of the same class, but also defined by the noisy features that result in the noisy or ill-represented patterns. This section reports the performance of VA-ARTs and the baselines on noisy data.

To quantitatively evaluate the effectiveness of these algorithms on noisy data, a widely used method was followed to add noise to different proportions of the original data to produce noisy datasets at different noisy levels. Specifically, the Matlab function $y = awgn(x, snr)$ was used to add additive white Gaussian noise to the data collected from the NUS-WIDE dataset, where x and y are the original and the generated noisy data patterns respectively, and snr is the signal-to-noise ratio. $snr = 20$ was empirically set to ensure that the generated noisy patterns would

Table 3.1 The best clustering performance of DBSCAN, Affinity Propagation (AP), Cluster$_{dp}$, Fuzzy ART, AM-ART, CM-ART, and HI-ART on the four datasets in terms of purity, class entropy, and the Rand index. ©2016 IEEE. Reprinted, with permission, from [19]

		DBSCAN	AP	Cluster$_{dp}$	Fuzzy ART	AM-ART	CM-ART	HI-ART
NUS-WIDE	Purity	0.6598 ± 0.015	0.6827	0.6193	0.7264 ± 0.026	0.7313 ± 0.023	$\mathbf{0.7436 \pm 0.023}$	0.7348 ± 0.025
	Class entropy	0.7188 ± 0.011	**0.7063**	0.7497	0.7287 ± 0.024	0.7148 ± 0.022	0.7266 ± 0.026	0.7159 ± 0.021
	Rand index	0.7970 ± 0.012	0.8084	0.7408	0.8305 ± 0.027	0.8244 ± 0.019	$\mathbf{0.8461 \pm 0.026}$	0.8419 ± 0.023
20 Newsgroups	Purity	0.7084 ± 0.017	0.7225	0.6518	0.7165 ± 0.027	0.7476 ± 0.022	$\mathbf{0.7735 \pm 0.019}$	0.7491 ± 0.024
	Class entropy	$\mathbf{0.5604 \pm 0.016}$	0.5779	0.5978	0.5679 ± 0.027	0.5873 ± 0.021	0.6081 ± 0.024	0.5936 ± 0.026
	Rand index	0.8303 ± 0.013	0.8522	0.7907	0.8527 ± 0.021	0.8745 ± 0.023	$\mathbf{0.8918 \pm 0.018}$	0.8794 ± 0.024
Corel5K	Purity	0.6792 ± 0.012	0.6926	0.5708	$\mathbf{0.7983 \pm 0.026}$	0.7627 ± 0.018	0.7863 ± 0.022	0.7715 ± 0.020
	Class entropy	0.4940 ± 0.009	0.5358	0.5639	0.5216 ± 0.016	$\mathbf{0.4758 \pm 0.021}$	0.5391 ± 0.017	0.5034 ± 0.015
	Rand index	0.8408 ± 0.018	0.8639	0.6977	$\mathbf{0.9391 \pm 0.024}$	0.9284 ± 0.014	0.9380 ± 0.019	0.9369 ± 0.021
BlogCatalog	Purity	0.7762 ± 0.017	0.8023	0.7129	0.8431 ± 0.017	$\mathbf{0.8635 \pm 0.013}$	0.8599 ± 0.023	0.8492 ± 0.016
	Class entropy	0.5121 ± 0.019	**0.4889**	0.5307	0.5321 ± 0.022	0.5003 ± 0.021	0.5218 ± 0.019	0.4963 ± 0.017
	Rand index	0.8720 ± 0.016	0.9120	0.7836	0.9361 ± 0.014	$\mathbf{0.9561 \pm 0.018}$	0.9492 ± 0.018	0.9484 ± 0.015

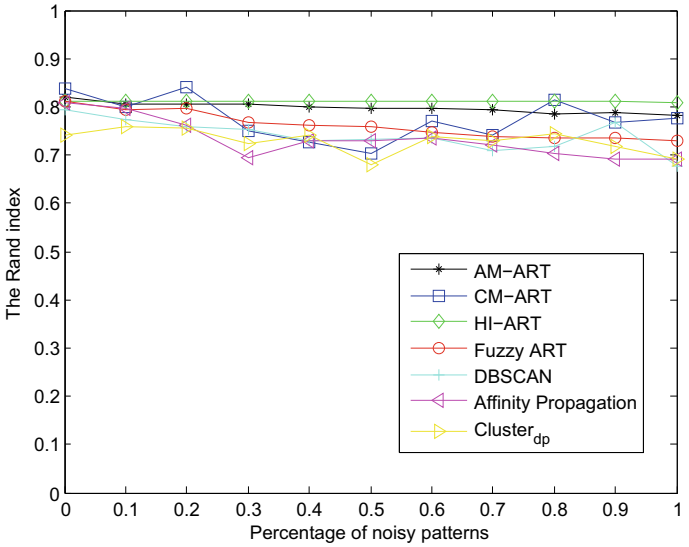

Fig. 3.10 Performance of AM-ART, CM-ART, HI-ART, and the algorithms in comparison on noisy data generated from the NUS-WIDE dataset. © 2016 IEEE. Reprinted, with permission, from [19]

generally blur, but not break, the original distribution of patterns to a certain extent. For each class of data patterns, the same number of patterns was randomly selected to add noise. In total, ten noisy datasets were generated with different proportions of noisy patterns.

The average performance of all algorithms obtained from ten runs on the original and ten noisy datasets were reported in Fig. 3.10. Regarding the ART-based algorithms, Fuzzy ART has a relatively stable decrease in performance when applied to noisier datasets while AM-ART, CM-ART, and HI-ART behave differently. AM-ART and HI-ART show a much better robustness than Fuzzy ART, especially HI-ART whose performance is almost not affected by the noisy patterns; while the performance of CM-ART fluctuates on different noisy datasets. An investigation of the generated cluster structures found that the performance of Fuzzy ART decreased mainly because of the increase in the generated clusters, while the clusters generated by Fuzzy ART still had a high quality in terms of precision; AM-ART and HI-ART alleviate this case by generating higher-quality, but much fewer, clusters than Fuzzy ART; the performance of CM-ART was affected by the case when the noisy patterns were selected as cluster centers. This produced much more complex cluster boundaries and resulted in the over-generation of clusters. However, by incorporating both AMR and CMR, HI-ART can largely alleviate this problem. In comparison, the performance of DBSCAN, Affinity Propagation, and Cluster$_{dp}$ also decreased and fluctuated along with the increase in the percentage of noisy data. It demonstrated the robustness of the proposed AM-ART and HI-ART to noise.

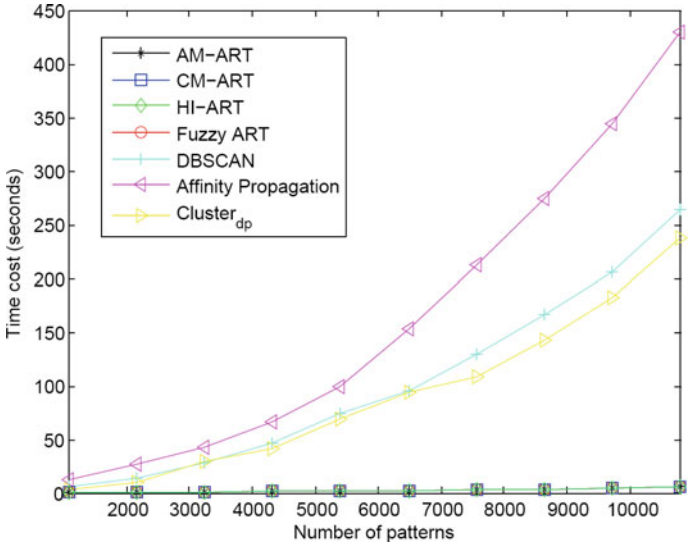

Fig. 3.11 Time cost of AM-ART, CM-ART, HI-ART, and the algorithms in comparison on the NUS-WIDE dataset. © 2016 IEEE. Reprinted, with permission, from [19]

3.3.5.7 Time Cost Comparison

This section presents an evaluation on the time cost of the proposed algorithms and the baselines on the NUS-WIDE dataset with respect to the increase in the number of input patterns. Specifically, to ensure an unbiased evaluation, the 10,800 patterns in the dataset were divided into 10 subsets, each of which contained 1,080 patterns of equally sized subsets from the nine classes and tested the time cost of each algorithm by incrementally adding a subset at each time. To ensure a fair comparison, the parameter settings of each algorithm followed those used in the previous section, but they were slightly tuned to force them to generate the same number of clusters. All algorithms were run on a 3.40 GHz Intel(R) Core (TM) i7-4770 CPU with 16 GB RAM. Figure 3.11 illustrates that, compared with Affinity Propagation, DBSCAN and Cluster$_{dp}$, the time cost of the four ART-based algorithms was much faster and increased slightly as the number of input patterns increased. It is notable that AM-ART, CM-ART, HI-ART and Fuzzy ART were able to cluster 10,800 patterns in 6 s. This demonstrates the scalability of the ART-based algorithms for big social media datasets. Moreover, the largest difference in their time cost was only less than 0.2 s, which demonstrates that the incorporation of AMR, CMR and HIR into Fuzzy ART incurs little computation.

3.4 User Preference Incorporation in Fuzzy ART

In social media analytics, clustering is usually used as a tool to discover the information that is of interest to users. However, the diverse needs and subjective knowledge of users may result in different needs for the clustering results. As discussed in Sect. 2.2, group and pairwise label constraints are commonly-used for gathering additional information for semi-supervised clustering. However, most of the existing algorithms incorporate such information for either learning new representation, distance measures of data or making the cluster assignment of the data objects as close to the user preferences as possible. Therefore, user preferences may not be obvious in the final clustering results of these algorithms.

This section illustrates an architecture extended from Fuzzy ART to incorporate user preferences, which essentially creates predefined clusters to partition the feature space before clustering happens. Beyond that, we show that this architecture not only incorporates the user-provided information to enhance the clustering quality, but also provides the flexibility for users to directly control the degree of topic mining, thus creating the user-desired clustering results. In a real-world application, Chaps. 4 and 5 will illustrate how to incorporate such methods into ART-based clustering algorithms and how it improves the performance.

3.4.1 General Architecture

Figure 3.12 depicts the general architecture of the extended Fuzzy ART for incorporating user preference. As observed, it has two channels, where $\mathbf{x}^a$ received by the left channel is the data object's feature vector, and $\mathbf{x}^b$ is the vector of user preferences. Specifically, $\mathbf{x}^b$ is a multi-hot vector, encoding user-provided semantic labels for the given data object. The provided semantic labels may be class labels, descriptive tags etc. Notably, the preference channel has a direct match rather than the two-way similarity measure, because the preference vector is used as a guide for the splitting and merging of data clusters.

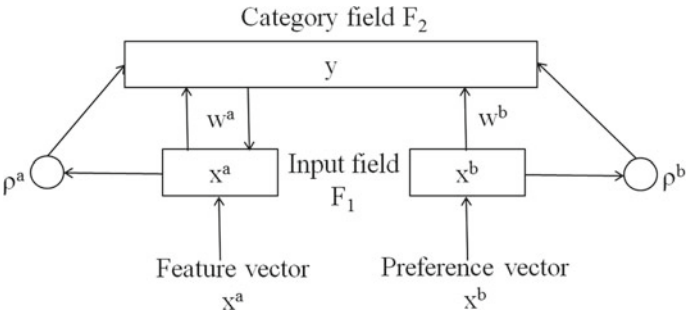

Fig. 3.12 The architecture of two-channel Fuzzy ART for incorporating user preferences

Taking advantage of the incremental clustering nature of Fuzzy ART, this architecture incorporates such user preferences by creating pre-defined clusters using both feature and preference vectors. Its general procedures include

1. Selecting one or more data objects belonging to the same group, with $\mathscr{I}_i = \{I_1, \ldots, I_n\}$ the normalized feature vectors, and provides the corresponding semantic labels $G_i = \{g_1, \ldots, g_p\}$.
2. Given the clusters $\mathscr{C} = \{c_1, \ldots, c_j\}$ in category field F_2 and the semantic lexicon $\mathscr{G}$, $\mathbf{x}^a$ and $\mathbf{x}^b$ are constructed. $\mathbf{x}^a = mean(I_1, \ldots, I_n)$. Obtaining the new lexicon $\hat{\mathscr{G}} = \mathscr{G} \bigcup G_i = \{g_1, \ldots, g_m\}$, $\mathbf{x}^b = [x_1, \ldots, x_m]$, where $x_i = 1$ if $g_i \in G_i$ and $x_i = 0$ otherwise.
3. Applying complement coding to $\mathbf{x}^a$ such that $\mathbf{x}^a = [\mathbf{x}^a, \mathbf{1} - \mathbf{x}^a]$, and presenting $\mathbf{x}^a$ and $\mathbf{x}^b$ to the input field F_1.
4. Using the match function, i.e. Eq. (3.2), to select the clusters $\mathscr{C}_w$ passing vigilance criteria according to $\mathbf{x}^b$.
5. Using the choice and match functions, i.e. Eqs. (3.1) and (3.2), to find the best-matching cluster c_{j^*} according to $\mathbf{x}^a$.
6. If c_{j^*} exists, updating cluster weights by $\mathbf{w}_{j^*} = mean_{\mathbf{I}_i \in c_{j^*}}(\mathbf{I}_i)$. Otherwise, creating a new cluster c with $\mathbf{w}^a = \mathbf{x}^a$ and $\mathbf{w}^b = \mathbf{x}^b$.
7. Repeating the above procedures until all user-specified groups of data objects are presented.

3.4.2 Geometric Interpretation

The above approach proposed for Fuzzy ART to incorporate user preferences works by partitioning the feature space using the user-specified data groups in the pre-clustering stage. The matching in the channel of $\mathbf{x}^b$ provides users with direct control over the degree of fine-grained topic mining, determined by ρ^b. Additionally, it allows users to flexibly present one group of data objects in multiple rounds of selection. Data objects in the channel of $\mathbf{x}^a$ serve as seeds to partition the feature space, and ρ^a controls the degree of the generalization power of clusters, where data groups with vigilance regions (VRs) far away from each other will result in the generation of multiple clusters to encode their respective feature distributions.

The incorporation of user preferences makes Fuzzy ART a semi-supervised learning algorithm, which takes in the user preferences in the form of prior knowledge to initialize a cluster structure before clustering. Essentially, the user may identify data groupings wherein the data objects in the same group are deemed to be similar to each other. As such, these predefined clusters can then be treated as a user-defined projection from the feature space to the category space. During the subsequent clustering process, these user-defined clusters can be further generalized by recognizing and learning from similar input patterns, while new clusters can still be created automatically for novel patterns dissimilar to the existing clusters. By incorporating user preferences, the predefined clusters help create better cluster structures than those use only pure data driven clustering.

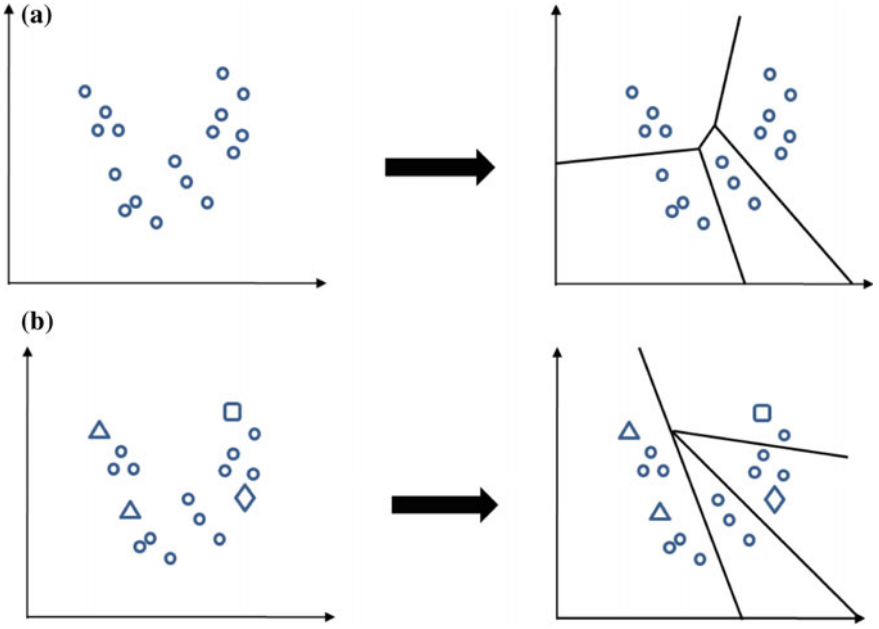

Fig. 3.13 An example illustrating the influence of user preferences on the clustering results of Fuzzy ART. **a** One possible clustering result of Fuzzy ART—original data shown on the left while the result is on the right; **b** Changes in clustering result after receiving user preferences, i.e. the "triangle," "rectangle" and "diamond"

Figure 3.13 gives an example that geometrically shows how this approach works and is able to make users in-the-loop to obtain desired clusters. Specifically, Fig. 3.13a shows a typical clustering result of Fuzzy ART on the data points in the feature space. As observed, without user preference, Fuzzy ART incrementally processes the data objects and partitions them according to the predefined distance measures. In contrast, as shown in Fig. 3.13b, with the user-specified data objects in three classes (i.e. the "triangle," "rectangle" and "diamond"), the data objects that were originally partitioned in two clusters (in Fig. 3.13a) are clustered together with the connection of "triangles," while those originally in the same cluster are separated due to the "rectangle" and "diamond".

3.5 Probabilistic ART for Short Text Clustering

Images shared on social networking websites are usually diverse, making it difficult to organize or search using the images themselves. As shown in Fig. 3.14, images in the same class, even of the same semantics, can have very different appearances, because of color difference, angle of photo capturing, and background context etc.

Fig. 3.14 An example on two sets of images with surrounding text (after filtering stop words) shared on social networking websites. The words shared across the two sets are circled

Luckily, these images are usually accompanied by rich surrounding text such as a title and user comments, which may describe their semantics. This motivates the tag-based approach for image clustering [22].

However, as observed in Fig. 3.14, such text is typically short and noisy—because of personal knowledge and interests. The surrounding text left by users may be quite diverse and even irrelevant. Another problem is that the words in the surrounding text usually appear only once, and the small number of keywords may be buried by much noisier words. More importantly, some meaningful words may be shared by different categories. This may cause difficulties when measuring the similarity between such images. Currently, the two challenges remaining in this task include the identification of keywords for data clusters and the robustness to noisy words.

This section describes a Fuzzy ART variant, i.e. Probabilistic ART, for short text clustering. Unlike Fuzzy ART, which updates cluster weights by depressing unstable feature values, it models the representation, i.e. weight vector, of clusters using the probabilistic distribution of word occurrences. To achieve this, Probabilistic ART does not employ complement coding, and it has a new learning function. Chapter 4 will further describe how to use it as one of the core algorithms for tag-based web image organization. This algorithm may be applied to other scenarios, such as tweet mining.

3.5.1 Procedures of Probabilistic ART

Probabilistic ART shares the procedures of Fuzzy ART, i.e. category choice, template matching, and prototype learning, which are detailed in Sect. 3.1. However, it has own data representation and prototype learning methods, making the above procedures have different meanings, as illustrated below:

- **Data Representation**: Given a word lexicon of all distinct words from short text $\mathscr{G} = \{g_1, \ldots, g_m\}$ and the word list of a data object $G = \{g_1, \ldots, g_p\}$, Probabilistic ART represent the data object using a multi-hot vector $\mathbf{x} = [x_1, \ldots, x_m]$,

defined as

$$x_i = \begin{cases} 1 & if\ x_i \in G \\ 0 & otherwise \end{cases}.$$

Note that complement coding is not used, since the learning function changes and the theory of vigilance regions (VRs) does not apply here.

This representation is a point in the textual feature space of m dimensions, and more common words in two given vectors lead to a shorter distance. Traditional methods for word weighting, such as tf-idf or word/neural embedding are not adopted, because words with short text cannot provide sufficient statistical information [12], which may result in feature vectors with a flat distribution and low values.

- **Similarity Measure**: Probabilistic ART also uses the choice and match functions (Eqs. (3.1) and (3.2)) for selecting the best-matching cluster c_{j^*}. However, considering that the two functions measure the intersection of vector histograms and the cluster weight vector is modeled by the probability of occurrences, the similarity measure used in Probabilistic ART essentially measures the degree of the match in terms of the keywords of clusters, instead of using the theory of VR.
- **Prototype Modeling**: As aforementioned, Probabilistic ART models the cluster prototype, i.e. weight vector, using the probability of word occurrences. It makes the cluster weight $\mathbf{w}$ of cluster c_j a word distribution, reflecting the importance of words to c_j. In this way, the keywords, or semantics, of a data cluster are naturally obtained during the clustering process. The following section, i.e. Sect. 3.5.2, explains why the learning function of Fuzzy ART fails in this case and illustrates the derivation of the learning function for the cluster weight $\hat{\mathbf{w}}$.

3.5.2 Probabilistic Learning for Prototype Modeling

As demonstrated in Property 3.2 of Sect. 3.1.2, the learning function of Fuzzy ART, i.e. Eq. (3.3), models cluster prototypes by stably suppressing the rare and unstable components while preserving the key and frequent ones. However, when learning from the multi-hot features of short text, a set of noise-induced mismatched words will erode the values of the key ones in the cluster weights. Besides, the sub-key words cannot be preserved, which may lead to the generation of extra clusters that represent the same topics.

Based on the above consideration, Probabilistic ART uses a new learning function that models cluster weights using probabilistic distribution of word occurrences, so that the weights of noisy words are suppressed while the key and sub-key ones are preserved.

Given a cluster c_j with a cluster weight vector $\mathbf{w}_j = [w_{j,1}, \ldots, w_{j,m}]$ and l data objects therein, denoted as $\mathscr{T} = \{\mathbf{t}_1, \ldots, \mathbf{t}_l\}$ where $\mathbf{t}_i = [t_{i,1}, \ldots, t_{i,m}]$, the probability of occurrence of the kth word t_k in cluster c_j can be calculated by its frequency:

$$w_{j,k} = p(t_k|c_j) = \frac{\sum_{i=1}^{l} t_{i,k}}{l}. \tag{3.12}$$

In this way, the weight prototype of cluster c_j is represented by the probability of word occurrences, i.e. $\mathbf{w}_j = [p(t_1|c_j), \ldots, p(t_m|c_j)]$.

Subsequently, the sequential factor is introduced and denoted in Eq. 3.12 by $p_l(t_k|c_j)$ as the state for time l. Assuming a new data object $\mathbf{t}_{l+1}$ is assigned to cluster c_j, the relationship between the states of time l and $l+1$ can be derived by

$$p_{l+1}(t_k|c_j) = \frac{\sum_{i=1}^{l+1} t_{i,k}}{l+1} = \frac{l}{l+1} p_l(t_k|c_j) + \frac{t_{l+1,k}}{l+1}. \tag{3.13}$$

As such, the general form of learning function for $w_{j,k}$ is defined by

$$\hat{w}_{j,k} = \frac{l}{l+1} w_{j,k} + \frac{t_{l+1,k}}{l+1}, \tag{3.14}$$

where l is the number of data objects in cluster c_j, and $t_{l+1,k}$ is the kth element of the input data object $\mathbf{t}_{l+1}$. Considering $\mathbf{t}_{l+1,k}$ equals either 0 or 1, the learning function for cluster weight $\mathbf{w}_j = [w_{j,1}, \ldots, w_{j,m}]$ can be further simplified as

$$\hat{w}_{j,k} = \begin{cases} \eta w_{j,k}, & t_{l+1,k} = 0 \\ \eta(w_{j,k} + \frac{1}{l}), & otherwise \end{cases}, \tag{3.15}$$

where $\eta = \frac{l}{l+1}$.

3.6 Generalized Heterogeneous Fusion ART (GHF-ART) for Heterogeneous Data Co-Clustering

Social media data is usually associated with rich meta-information, such as an article with images included or a video with a textual description and user comments. Such heterogeneous data describes the single data object in different views, naturally leading to the question: Is it possible to more effectively represent the data objects by utilizing multimodal data instead of a single modality?

Existing studies, as described in Sect. 2.3, typically treat it as a multi-objective optimization problem, i.e. finding the cluster partition that leads to minimized objective function values across different modalities. However, the increased computational complexity incurred by multimodal features and the method for weighting the heterogeneous feature modalities are still open challenges.

This section describes Generalized Heterogeneous Fusion ART (GHF-ART) as a solution for clustering data objects represented by multimodal features. Compared to related work, GHF-ART stands out due to its linear time complexity, weighting

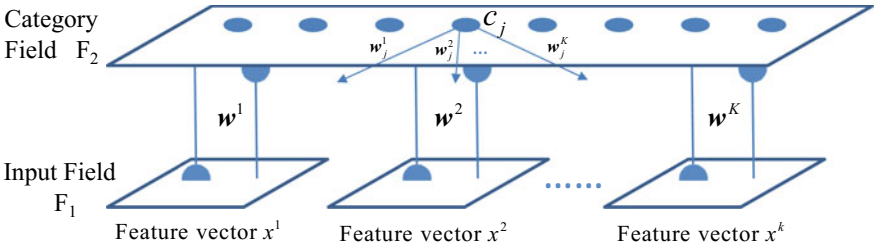

Fig. 3.15 The architecture of Generalized Heterogeneous Fusion ART. Vigilance connections are omitted for visual simplicity. © 2014 IEEE. Reprinted, with permission, from [20]

method for feature modalities and the ability to use multiple clusters to encode data objects that are similar in partial feature modalities. Chapters 5 and 6 will illustrate using GHF-ART for clustering composite multimedia data objects and heterogeneous social networks.

3.6.1 General Architecture

Generalized Heterogeneous Fusion ART (GHF-ART) is a variant of Fusion ART that shares its architecture, as shown in Fig. 3.15. As observed, this architecture has multiple input fields and just one category field, and it is a general architecture for simultaneously learning with multimodal feature mappings. It extends Fusion ART [24] (see Sect. 2.3.6) from two aspects: First, GHF-ART allows different feature channels to have their own data representation and cluster weight learning functions; secondly, an unsupervised method, called *robustness measure*, is utilized to weight feature modalities in the overall similarity measure. These extensions make GHF-ART able to receive a different type of data pattern in each input feature channel and to fuse different types of features for an effective similarity measure.

Whereas most current algorithms [2, 5, 14, 23] employ global optimization methods for heterogeneous data fusion, GHF-ART performs heterogeneous data co-clustering using a multi-channel self-organizing neural network, i.e. Fusion ART. In essence, GHF-ART simultaneously learns the multi-dimensional mappings across multiple feature spaces to the category space. The clustering process of GHF-ART thus incrementally partitions input feature spaces and maps them to the category space, forming regions of clusters. The vigilance parameters ρ^k for each independent channel allow GHF-ART to threshold the minimum intra-cluster similarity across feature channels, making it different from existing algorithms which may group the data objects that are similar in most feature channels but dissimilar in a few ones.

3.6.2 Clustering Procedures

The clustering procedures of GHF-ART follow the general procedures of ART, i.e. category choice, template matching, and prototype learning, but they also extend the base functions to handle multimodal feature channels. Additionally, an update on the weight values for different feature channels is performed after the cluster assignment of each input data object. The details are illustrated below:

1. **Data Representation**: Let $\mathbf{I} = \{\mathbf{x}^k|_{k=1}^K\}$ denote the normalized multi-channel input data object, where $\mathbf{x}^k$ is the feature vector for the kth feature channel. Note that, with complement coding [4], $\mathbf{x}^k$ in the input field F_1 is further augmented with a complement vector $\bar{\mathbf{x}}^k$ such that $\bar{\mathbf{x}}^k = 1 - \mathbf{x}^k$.
2. **Category Choice**: Given a cluster c_j with weight vectors $\{\mathbf{w}_j^k|_{k=1}^K\}$, the choice function used in GHF-ART measures the weighted average of similarities across the K feature channels, defined as

$$T(c_j, \mathbf{I}) = \sum_{k=1}^{K} \gamma^k \frac{|\mathbf{x}^k \wedge \mathbf{w}_j^k|}{\alpha + |\mathbf{w}_j^k|}, \tag{3.16}$$

   where γ^k, called the contribution parameter, is the weight value for the kth feature channel. $\gamma^k = \frac{1}{K}$ makes $T(c_j, \mathbf{I})$ average the similarities across all feature channels. The following sub-section illustrates how to use the *robustness measure* to make γ^k be self-adaptable.
3. **Template Matching**: After identifying the cluster with the highest value, denoted as the winner c_{j^*}, the multi-channel version of the match function is used to evaluate to which degree the input pattern $\mathbf{I}$ is a subset of c_{j^*} in terms of each feature channel k, defined as

$$M(c_{j^*}, \mathbf{x}^k) = \frac{|\mathbf{x}^k \wedge \mathbf{w}_{j^*}^k|}{|\mathbf{x}^k|}. \tag{3.17}$$

   Note that in GHF-ART, a resonance occurs only when the match values of all the K feature channels satisfy the vigilance criteria $M(c_{j^*}, \mathbf{x}^k) \geq \rho^k$ $(k = 1, \ldots, K)$. Otherwise, a reset occurs to select a new winner from the rest of the clusters in the category field F_2.
4. **Prototype Learning**: As aforementioned, GHF-ART allows different feature channels to have their own data representation and learning functions, making it possible to learn effective feature distribution for pattern representation of each feature modality. As such, when input features in all the feature channels satisfy the vigilance criteria, their corresponding weight vectors $\mathbf{w}_{j^*}^k$ $(k = 1, \ldots, K)$ are updated through the respective learning functions, which can be the original learning function of ART, i.e. Eq. (3.3), for images and text articles, or that of Probabilistic ART, i.e. Eq. (3.15), for short text.

5. **Adaptive Feature Weighting**: In addition to prototype learning, the assignment of an input pattern also leads to the update of feature channel weights, i.e. the contribution parameters γ^k, using the robustness measure as described in the following Sect. 3.6.3.

3.6.3 Robustness Measure for Feature Modality Weighting

The contribution parameter γ^k specifies the weighting factor given to each feature channel during the category choice process. Intuitively, the feature channel which is more robust in distinguishing the classes of the patterns should have a higher weight. It motivates the development of a "*robustness measure*" to learn data-specific robustness of feature modalities from the input data itself rather than following an empirical setting.

3.6.3.1 General Method

A robust feature modality consistently reveals the common features shared by the data objects of the same class, which is why the robustness of feature modalities can be measured by the intra-cluster scatters, i.e. the distance between cluster weights and features of the data objects therein. Consider a cluster c_j with weight vectors $\{\mathbf{w}_j^1, \ldots, \mathbf{w}_j^K\}$ and its members $\mathscr{I} = \{\mathbf{I}_1, \ldots, \mathbf{I}_L\}$ where $\mathbf{I}_i = \{\mathbf{x}_i^1, \ldots, \mathbf{x}_i^K\}$ for $i = 1, \ldots, L$, the intra-cluster scatter, termed *Difference*, for the kth feature vector is first defined as follows:

$$D_j^k = \frac{\frac{1}{L}\sum_l |\mathbf{w}_j^k - \mathbf{x}_l^k|}{|\mathbf{w}_j^k|}. \tag{3.18}$$

Subsequently, the overall difference of one feature vector can be evaluated by averaging the difference of all the clusters, defined by:

$$D^k = \frac{1}{J}\sum_j D_j^k, \tag{3.19}$$

where J is the number of clusters. Therefore, the *robustness* of the kth feature modality can be measured by

$$R^k = \exp(-D^k). \tag{3.20}$$

When D^k is 0, R^k becomes 1, indicating that this feature modality can properly represent data objects belonging to the same class. In contrast, when D^k is very large, R^k approaches zero. This expression implies that the feature modality with a higher difference is not robust and is less reliable. Thus, in a normalized form, the contribution parameter γ^k for the kth feature channel can be expressed by

$$\gamma^k = \frac{R^k}{\sum_{k=1}^{K} R^k}. \tag{3.21}$$

This equation shows the rule for tuning the contribution parameter during the clustering process. Initially, the contribution parameter is given by equal weights based on the intuition that the powers of all features are the same. Subsequently, the value of γ^k changes along with the encoding of the input patterns.

3.6.3.2 Incremental Version of Robustness Measure

The update of the contribution parameters γ^k using the *robustness measure* occurs after each resonance, i.e. the cluster assignment for each input data object. As such, Eqs. (3.18)–(3.21) are computationally expensive. This incurs the need for an incremental method that updates γ^k based on the values of the last round.

GHF-ART encodes input data by either updating the existing cluster weights or creating a new cluster. These two actions produce different levels of changes to *Difference* and *Robustness*. Therefore, the update functions are considered in two cases:

- **Resonance in an existing cluster**: Assume the input data object is assigned to an existing cluster c_j with L data objects. In this case, only the change of D_j^k should be considered. Based on Eq. (3.18), the *Difference* for the kth feature channel of cluster c_j is computed by

$$\begin{aligned}\hat{D}_j^k &= \frac{\frac{1}{L+1}\sum_{l=1}^{L+1}|\hat{\mathbf{w}}_j^k - \mathbf{x}_l^k|}{|\hat{\mathbf{w}}_j^k|} \\ &= \frac{1}{(L+1)|\hat{\mathbf{w}}_j^k|}\left(\sum_{l=1}^{L}|\mathbf{w}_j^k - \mathbf{x}_l^k + \hat{\mathbf{w}}_j^k - \mathbf{w}_j^k| + |\hat{\mathbf{w}}_j^k - \mathbf{x}_{L+1}^k|\right).\end{aligned} \tag{3.22}$$

  The next important step is to introduce $|D_j^k|$ to Eq. (3.22) by separating out $|\mathbf{w}_j^k - \mathbf{x}_l^k|$. It leads to the problem of determining whether $\mathbf{w}_j^k - \mathbf{x}_l^k$ and $\hat{\mathbf{w}}_j^k - \mathbf{w}_j^k$ share the same positive or negative signs.

  Solving this problem requires tricks obtained from data distribution and the corresponding learning functions. For example, the learning function of ART, i.e. Eq. (3.3), teaches that $|\hat{\mathbf{w}}_j^k| < |\mathbf{w}_j^k|$, and in most cases $|\mathbf{w}_j^k| < |\mathbf{x}_l^k|$ when the cluster weights become stable. In this case, Eq. (3.22) can be approximated using

$$\begin{aligned}\hat{D}_j^k &\leq \frac{1}{(L+1)|\hat{\mathbf{w}}_j^k|}\left(\sum_{l=1}^{L}|\mathbf{w}_j^k - \mathbf{x}_l^k| + L|\hat{\mathbf{w}}_j^k - \mathbf{w}_j^k| + |\hat{\mathbf{w}}_j^k - \mathbf{x}_{L+1}^k|\right) \\ &\leq \frac{\eta}{|\hat{\mathbf{w}}_j^k|}\left(|\mathbf{w}_j^k|D_j^k + |\mathbf{w}_j^k - \hat{\mathbf{w}}_j^k| + \frac{1}{L}|\hat{\mathbf{w}}_j^k - \mathbf{x}_{L+1}^k|\right).\end{aligned} \tag{3.23}$$

  where $\eta = \frac{L}{L+1}$.

Similarly, as in the short text clustering scenario described in Sect. 3.5, it is known that $\mathbf{x}$ is usually sparse, keywords possess only a small portion and the learning function, i.e. Eq. (3.15), models the probability of word occurrences. As such, when the cluster weights become stable, they are likely to have $|\mathbf{w}_j^k| < |\mathbf{x}_l^k|$ and $|\hat{\mathbf{w}}_j^k| < |\mathbf{w}_j^k|$, since $|\mathbf{x}_l^k|$ contains multiple 1's and $|\mathbf{w}_j^k|$ usually have the weights of sub-key and noisy words decreased.

After the update for all the feature channels, the new contribution parameters γ^k can then be obtained by calculating Eqs. (3.18)–(3.21). In this way, the computational complexity reduces from $O(n_i n_f)$ to $O(n_f)$, where n_f denotes the dimension of the feature channels and n_i denotes the number of documents.

- **Generation of new cluster**: When generating a new cluster, the *Difference* of the other clusters remain unchanged. Therefore, the addition of a new cluster just introduces a proportion change to the *Robustness*, according to Eq. (3.19). Considering the robustness R^k $(k = 1, \ldots, K)$ for all the feature channels, the update equation for the kth feature channel is defined as:

$$\hat{\gamma}^k = \frac{\hat{R}^k}{\sum_{k=1}^{K} \hat{R}^k} = \frac{(R^k)^{\frac{J}{J+1}}}{\sum_{k=1}^{K} (R^k)^{\frac{J}{J+1}}}, \tag{3.24}$$

3.6.4 Time Complexity Analysis

GHF-ART has been demonstrated to have a linear time complexity of $O(n_i n_c n_f)$ in [20], where n_i is the number of data objects, n_c is the number of clusters and n_f is the number of feature dimensions.

Specifically, it depends on the following two steps:

1. **Search for suitable clusters**: that calculates the choice and match functions by

$$T(c_j, \mathbf{I}) = \sum_{k=1}^{K} \gamma^k \frac{|\mathbf{x}^k \wedge \mathbf{w}_j^k|}{\alpha + |\mathbf{w}_j^k|},$$

$$M(c_{j^*}, \mathbf{x}^k) = \frac{|\mathbf{x}^k \wedge \mathbf{w}_{j^*}^k|}{|\mathbf{x}^k|},$$

   which are defined in Eqs. (3.16)–(3.17) and have a time complexity of $O(n_c n_f)$.
2. **Update of contribution parameters**: that contains two cases: (1) the input pattern is grouped into one of the existing clusters, and (2) a new cluster is generated for the input pattern. As illustrated in Sect. 3.6.3, for the first case, the new contribution parameter is calculated by Eqs. (3.19)–(3.21) and Eq. (3.23). The time complexity of Eq. (3.23) is $O(n_f)$ and that of Eqs. (3.19)–(3.21) is $O(1)$. For the second case, the contribution parameter is updated according to Eq. (3.24), of which the time complexity is $O(1)$.

3.7 Online Multimodal Co-indexing ART (OMC-ART) for Streaming Multimedia Data Indexing

The retrieval of social media data plays an important role in social network services, such as searching for image boards of interest on Pinterest, finding answers to Python programming questions on Github community portal, or looking for products on e-commerce websites. An intractable issue for this task is the processing of continuous incoming data, which will create a great burden to re-generating the indexing base, i.e. vector representation of data, for the search engine. Another challenge is the effective representation of multimedia data with multimodal information, which has been intensively discussed in both Sects. 2.3 and 3.6.

This section discusses Online Multimodal Co-indexing ART (OMC-ART) which tackles the aforementioned challenges using a clustering approach. By extending GHF-ART with the online learning capability, OMC-ART incrementally processes the input multimodal data object and generates a two-layer hierarchical indexing base. The first layer is the clusters with generalized feature distribution and the key features of the data objects therein, while the second one includes data objects as leaves. Compared to related work, OMC-ART does not require any ground-truth information and can perform online learning of streaming data for updating the existing indexing base without re-visiting past data for re-creation. Chapter 7 will show how to use OMC-ART to build search engines that are capable of online indexing multimodal data and retrieving them flexibly using image, text or both.

3.7.1 General Procedures

Online Multimodal Co-indexing Adaptive Resonance Theory (OMC-ART) is a variant of GHF-ART with the advantages of low time complexity, effective multimodal data fusion, an incremental clustering manner and no need for a predefined number of clusters. Beyond that, OMC-ART incorporates an online data normalization method to be able to perform online learning, and it manipulates the cluster structure to make it an indexing base for effective and efficient searches.

Besides the general steps of GHF-ART, OMC-ART introduces two more steps for the online adaptation of cluster structures and the creation of an indexing base, as illustrated below:

1. **Data Representation**: OMC-ART uses the same method as GHF-ART for data representation, i.e. $\mathbf{I} = \{\mathbf{x}^k|_{k=1}^K\}$ for a multi-channel input data object, and it applies complement coding to the channels following the Fuzzy ART clustering theories. Note that the min/max values for each entry of $\mathbf{x}^k$, denoted by $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$, are identified and compared with those of past data objects. New min/max values will incur an update on $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$, and $\mathbf{I}$ will be normalized using the new min/max values $\hat{\mathbf{x}}_{min}$ and $\hat{\mathbf{x}}_{max}$.

2. **Online Normalization**: If $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ are updated, past data objects and weight vectors need to be updated too. A direct re-computation for updated weight vectors is time consuming. To solve this problem, OMC-ART adopts an online normalization method, without information loss, to perform an incremental update of the vectors of past data objects and weight values. The details are presented in Sect. 3.7.2.
3. **Cluster Generation**: OMC-ART follows the same procedures of GHF-ART to create data clusters, i.e. category choice, template matching, prototype learning and adaptive feature weighting, as described in Sect. 3.6.2.
4. **Salient Feature Discovery**: The clusters of OMC-ART serve as a natural two-layer hierarchical indexing base of data objects, where the first layer contains cluster weights reflecting the feature distribution of the data objects therein, while the second layer includes data objects in their respective clusters. As such, the salient features of each cluster can facilitate quick targeting of search queries. Section 3.7.3 describes the criteria for identifying the salient features of clusters after the clustering process.

3.7.2 Online Normalization of Features

GHF-ART may not be directly applicable to online learning because the *min-max normalization* requires the maximum and minimum values of each feature to normalize the feature vectors of the data objects so they have values in [0, 1]. To address this issue, OMC-ART employs an online adaptation method that updates the normalized feature distribution vectors $\mathbf{x}$ of the data objects and cluster weights $\mathbf{w}$ to exactly what they should be when an input data object incurs a change in such values, as respectively defined by Eqs. (3.25) and (3.26) below,

$$\hat{\mathbf{x}} = \frac{\mathbf{x}_{max} - \mathbf{x}_{min}}{\hat{\mathbf{x}}_{max} - \hat{\mathbf{x}}_{min}}\mathbf{x} + \frac{\mathbf{x}_{min} - \hat{\mathbf{x}}_{min}}{\hat{\mathbf{x}}_{max} - \hat{\mathbf{x}}_{min}}, \tag{3.25}$$

$$\hat{\mathbf{w}} = \frac{\mathbf{x}_{max} - \mathbf{x}_{min}}{\hat{\mathbf{x}}_{max} - \hat{\mathbf{x}}_{min}}\mathbf{w} + \frac{\mathbf{x}_{min} - \hat{\mathbf{x}}_{min}}{\hat{\mathbf{x}}_{max} - \hat{\mathbf{x}}_{min}}, \tag{3.26}$$

where $\mathbf{x}$ and $\hat{\mathbf{x}}$ denote the feature vector of a data object and its updated version, respectively. Similar definitions apply to $\mathbf{w}$ and $\hat{\mathbf{w}}$, $\mathbf{x}_{min}$ and $\hat{\mathbf{x}}_{min}$, and $\mathbf{x}_{max}$ and $\hat{\mathbf{x}}_{max}$.

OMC-ART may handle data streams in both a single or a batch of data objects. As an online algorithm, the initial maximum and minimum values $\mathbf{x}_{max}^{(1)}$ and $\mathbf{x}_{min}^{(1)}$ should be carefully considered when the first data stream has only one data object. In this case, without the loss of generalization, $\mathbf{x}_{max}^{(1)} = \mathbf{x}^{(0)}$ and $\mathbf{x}_{min}^{(1)} = \mathbf{x}^{(0)} - \mathbf{1}$, where $\mathbf{x}^{(0)}$ is the original value of $\mathbf{x}$ without normalization. Below presents the proof of Eqs. (3.25) and (3.26).

Theorem 3.1 *Considering a feature x of the data object* $\mathbf{x}$ *that has been normalized by N rounds of maximum and minimum values $\{x_{max}^{(n)}, x_{min}^{(n)}\}_{n=1}^{N}$, the value of x with n round of normalization $x^{(n)}$ can be inferred directly by that of $x^{(n-1)}$ by Eq. (3.25).*

Proof Given $x_{max}^{(n)}$ and $x_{min}^{(n)}$,

$$x^{(n)} = \frac{x^{(0)} - x_{min}^{(n)}}{x_{max}^{(n)} - x_{min}^{(n)}}, \tag{3.27}$$

$$x^{(n-1)} = \frac{x^{(0)} - x_{min}^{(n-1)}}{x_{max}^{(n-1)} - x_{min}^{(n-1)}}. \tag{3.28}$$

By substituting $x^{(0)}$ in Eq. (3.27) and using the expression of $x^{(0)}$ derived from Eq. (3.28),

$$x^{(n)} = \frac{x_{max}^{(n-1)} - x_{min}^{(n-1)}}{x_{max}^{(n)} - x_{min}^{(n)}} x^{(n-1)} + \frac{x_{min}^{(n-1)} - x_{min}^{(n)}}{x_{max}^{(n)} - x_{min}^{(n)}}. \tag{3.29}$$

Theorem 3.2 *Without loss of generalization, given a weight w of the weight vector* $\mathbf{w}$ *of cluster c, denoted by $w^{(N)}$, which learns from a set of feature values $\{x_n\}_{n=1}^{N}$ of N data objects and has been updated N times using the min/max values, denoted by $\{x_{max}^{(N)}, x_{min}^{(N)}\}_{n=1}^{N}$. If a new input data object d_{N+1} introduces $x_{max}^{(N+1)}$ and $x_{min}^{(N+1)}$, the adapted weight value $w^{(N+1)}$ can be derived by Eq. (3.26).*

Proof Suppose cluster c is the first cluster generated by Fuzzy ART, the weight vector $\mathbf{w}$ of c is thus set by

$$w^{(1)} = x_1^{(1)}. \tag{3.30}$$

According to the learning function of Fuzzy ART, i.e. Eq. (3.3), the value of w after the presentation of $x_n^{(n)}$ is

$$w^{(n)} = \begin{cases} w^{(n-1)} & if\ x_{n-1}^{(n)} \geq w^{(n-1)} \\ (1-\beta)w^{(n-1)} + \beta x_n^{(n)} & otherwise \end{cases}. \tag{3.31}$$

Based on the above Eqs. (3.30) and (3.31), it can be inferred that

$$w^{(N)} = c_1 x_1^{(N)} + \cdots + c_N x_N^{(N)}, \tag{3.32}$$

where c_i is a real-valued coefficient computed by the multiplication of 0, β, and $1-\beta$, and $c_1 + \cdots + c_N = 1$. Therefore, when $x_{max}^{(N+1)}$ and $x_{min}^{(N+1)}$ are introduced,

$$w^{(N+1)} = c_1 x_1^{(N+1)} + \cdots + c_{N+1} x_{N+1}^{(N+1)}. \tag{3.33}$$

By denoting Eq. (3.29) by $x^{(n)} = a^{(n)}x^{(n-1)} + b^{(n)}$, Eq. (3.33) is further derived as

$$\begin{aligned} w^{(N+1)} &= a^{(N+1)}(c_1 x_1^{(N)} + \cdots + c_N x_N^{(N)}) + (c_1 + \cdots + c_N)b^{(N+1)} \\ &= a^{(N+1)}w^{(N)} + b^{(N+1)}. \end{aligned} \tag{3.34}$$

This proof also holds for any algorithms that update cluster weights using a linear combination of the data objects therein, such as Probabilistic ART using Eq. (3.15) for short text clustering.

3.7.3 *Salient Feature Discovery for Generating Indexing Base of Data*

The cluster structure produced by OMC-ART can serve as a natural indexing base where each cluster contains data objects belonging to the same topic, and the cluster weights reveal the importance of the features of the individual clusters. It is achieved by both the use of vigilance parameter ρ^k, which does not limit the number of clusters but thresholds the intra-cluster similarity, and the incorporation of the learning functions of Fuzzy ART and Probabilistic ART which discover the key features by preserving or increasing the values of the key features while decreasing those of the noisy features.

In the scenario of a multimedia search, a query may first search for the matching clusters sharing similar feature/topic distributions, instead of the whole database. Selecting salient features for query matching is an important step, which may save the computation and avoid matching with noisy features. Given a cluster c_j with weight vectors $\{\mathbf{w}_j^k\}_{k=1}^K$ produced by OMC-ART, the set of salient features for each channel k, denoted by $\mathscr{K}_j^k$, is obtained based on the following criterion:

$$\mathscr{K}_j^k = \left\{ f_m^k | w_{j,m}^k > \frac{1}{M}\sum_{i=1}^{M} w_{j,i}^k \right\}, \tag{3.35}$$

where f_m^k is the mth feature of the kth feature channel, and M is the corresponding number of features.

The proposed criterion selects the features with values above average as the key features. It follows the idea that the high dimensional features are usually sparse and noisy, especially for the surrounding text of images. Therefore, the proposed method may filter the features, providing little information, while keeping those that are useful for indicating the difference between clusters.

3.7.4 Time Complexity Analysis

OMC-ART has been proven to have a total time complexity of $O(n_i n_c n_f)$ in [17], where n_i denotes the number of data objects, n_c denotes the number of clusters and n_f denotes the total number of features.

As described in Sect. 3.7.1, given a stream of totally n_i data objects, OMC-ART has three key steps:

1. **Online normalization of features**: that applies min-max normalization to the input data, which has a total time complexity of $O(n_i n_f)$ for all data. A change in the bound values of features x_{max} and x_{min} will incur a computational cost of $O((n_i + n_c)n_f)$ in the worst case for updating the values of past data and cluster weights, as defined in Eqs. (3.25)–(3.26).
2. **Clustering using GHF-ART**: that has been demonstrated in Sect. 3.6.4 to have an overall time complexity of $O(n_i n_c n_f)$.
3. **Indexing based generation**: that updates the existing indexing base with the input data and the new cluster weights, which has a time complexity of $O((n_i + n_c)n_f)$.

3.8 Discussion

This chapter presents a theoretical analysis of Fuzzy ART, an implementation of adaptive resonance theory (ART) with fuzzy operators, and a class of Fuzzy ART variants for addressing the challenges in social media data clustering. Fuzzy ART is chosen as the base model mainly because of its low time complexity, model extensibility, no need to set the number of clusters and incremental clustering manner, making it possible to handle big and complex social media data streams of heterogeneous types of information.

The theoretical interpretation of Fuzzy ART using vigilance region (VR) in Sect. 3.2 first offers a deep understanding of how this algorithm works and what its limitations are for clustering. Subsequently, in the following sections, the extensions of Fuzzy ART are illustrated, which respectively address the following problems:

1. **How to make the hyper-parameters that require manual settings self-adaptable**: It essentially requires a clustering algorithm to be able to identify the shared key features of data groups and partition them in the feature space with clusters of suitable shapes and intra-cluster scatters. Section 3.3 describes how to make clusters created by Fuzzy ART have their own threshold, i.e. vigilance parameter ρ, for intra-cluster similarity, using the theory of VR.
2. **How to produce clusters according to users' preferences**: It requires not only semi-supervised learning, which incorporates the association between data objects into consideration, but also generating the clusters that users want to obtain. Section 3.4 presents a solution based on a two-channel Fuzzy ART, which uses the user-provided seeding data groups to partition the feature space as predefined clusters and expand them during the clustering process.

3. **How to cluster short text having little statistical information**: It requires the ability of a clustering algorithm to identify the keywords/semantics of data groups and model the individual group's semantic distribution. Section 3.5 presents Probabilistic ART, which adopts new data representation and learning functions to discover the features of data clusters and model their semantic distributions, i.e. weight vectors, using the probability of word occurrences.
4. **How to fuse multimodal information to cluster composite multimedia data objects**: It requires a clustering algorithm with the ability to find appropriate representations and similarity measures for heterogeneous data and fuse the decisions of individual data modality for the overall similarity measure. Section 3.6 introduces the Generalized Heterogeneous Fusion ART (GHF-ART), which allows different feature modalities to have their own feature representation and learning methods and uses a method to adaptively weight different feature channels when fusing similarities measured by different channels for the overall similarity measure.
5. **How to index streaming multimodal data for retrieval**: It requires a clustering algorithm to be able to perform online learning and index multimodal data with efficient representations, i.e. an indexing base. Section 3.7 describes the Online Multimodal Co-indexing ART (OMC-ART), which extends GHF-ART by including the online learning capability and builds the indexing base using the generated cluster structure.

Compared to the existing approaches in the literature, the ART-based algorithms are superior in terms of linear time complexity and light parameter tuning. By using the two-way similarity measure and the intra-cluster similarity threshold ρ, they do not need the manual setting of the number of clusters and can produce a reasonable cluster structure even in the first epoch of data presentation. The theory of adaptive resonance has a fast, simple and extensible learning mechanism, making it a proper base model for incorporating new theories and approaches to address social media clustering challenges. Part II will illustrate how to use the algorithms introduced in this chapter to address problems in real-world social media mining tasks.

Beyond the ART variants and their target scenarios as described in this chapter, the advances of social tools for online communication and the development of novel machine learning and natural language processing techniques will lead to new challenges and requirements for clustering algorithms. These changes will boost the development of new ART variants incorporating with new methods to further improve the fundamental theory of ART for clustering, such as the over-generation of small clusters when the data representation is noisy; or with new theories on data embedding, such as word2vector and deep learning, to help with improved data representation to alleviate data sparsity and noisy feature problems. However, these algorithms may also encounter challenges when learning from social media data, requiring in-depth investigation and research.

References

1. Amorim DG, Delgado MF, Ameneiro SB (2007) Polytope ARTMAP: pattern classification without vigilance based on general geometry categories. IEEE Trans Neural Netw 18(5):1306–1325
2. Bekkerman R, Jeon J (2007) Multi-modal clustering for multimedia collections. In: CVPR, pp 1–8
3. Carpenter GA, Grossberg S, Reynolds JH (1991) ARTMAP: supervised real-time learning and classification of nonstationary data by a self-organizing neural network. Neural Netw 4(5):565–588
4. Carpenter GA, Grossberg S, Rosen DB (1991) Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. Neural Netw 4(6):759–771
5. Chen Y, Wang L, Dong M (2010) Non-negative matrix factorization for semisupervised heterogeneous data coclustering. TKDE 22(10):1459–1474
6. Chua T, Tang J, Hong R, Li H, Luo Z, Zheng Y (2009) NUS-WIDE: A real-world web image database from national university of singapore. In: CIVR, pp 1–9
7. Duygulu P, Barnard K, de Freitas JF, Forsyth DA (2002) Object recognition as machine translation: learning a lexicon for a fixed image vocabulary. In: ECCV, pp 97–112
8. Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, pp 226–231
9. He J, Tan AH, Tan CL (2002) Art-c: A neural architecture for self-organization under constraints. In: Proceedings of international joint conference on neural networks (IJCNN). Citeseer, pp 2550–2555
10. He J, Tan AH, Tan CL (2004) Modified art 2a growing network capable of generating a fixed number of nodes. IEEE Trans Neural Netw 15(3):728–737
11. He J, Tan AH, Tan CL, Sung SY (2003) On quantitative evaluation of clustering systems. Clustering and information retrieval. Kluwer Academic Publishers, pp 105–133
12. Hu X, Sun N, Zhang C, Chua TS (2009) Exploiting internal and external semantics for the clustering of short texts using world knowledge. In: Proceedings of ACM conference on information and knowledge management, pp 919–928
13. Lang K (2005) Newsweeder: learning to filter netnews. In: Proceedings of international conference machine learning, pp 331–339
14. Long B, Wu X, Zhang Z, Yu PS (2006) Spectral clustering for multi-type relational data. In: ICML, pp 585–592
15. Meng L, Tan AH (2012) Semi-supervised hierarchical clustering for personalized web image organization. In: Proceedings of international joint conference on neural networks (IJCNN), pp 1–8
16. Meng L, Tan AH (2014) Community discovery in social networks via heterogeneous link association and fusion. In: SIAM international conference on data mining (SDM), pp 803–811
17. Meng L, Tan AH, Leung C, Nie L, Chua TS, Miao C (2015) Online multimodal co-indexing and retrieval of weakly labeled web image collections. In: Proceedings of the 5th ACM on international conference on multimedia retrieval. ACM, pp 219–226. https://doi.org/10.1145/2671188.2749362
18. Meng L, Tan AH, Wunsch DC (2013) Vigilance adaptation in adaptive resonance theory. In: Proceedings of international joint conference on neural networks (IJCNN), pp 1–7
19. Meng L, Tan AH, Wunsch DC (2016) Adaptive scaling of cluster boundaries for large-scale social media data clustering. IEEE Trans Neural Netw Learn Syst 27(12):2656–2669
20. Meng L, Tan AH, Xu D (2014) Semi-supervised heterogeneous fusion for multimedia data co-clustering. IEEE Trans Knowl Data Eng 26(9):2293–2306
21. Papadopoulos S, Kompatsiaris Y, Vakali A, Spyridonos P (2012) Community detection in social media. Data Min Knowl Discov 24(3):515–554
22. Papadopoulos S, Zigkolis C, Kompatsiaris Y, Vakali A (2011) Cluster-based landmark and event detection for tagged photo collections. IEEE Multimed Mag 18(1):52–63

23. Rege M, Dong M, Hua J (2008) Graph theoretical framework for simultaneously integrating visual and textual features for efficient web image clustering. In: Proceedings of international conference on world wide web, pp 317–326
24. Tan AH, Carpenter GA, Grossberg S (2007) Intelligence through interaction: towards a unified theory for learning. LNCS 4491:1094–1103
25. Tan AH (1995) Adaptive resonance associative map. Neural Netw 8(3):437–446
26. Tan AH, Ong HL, Pan H, Ng J, Li Q (2004) Towards personalised web intelligence. Knowl Inf Syst 6(5):595–616
27. Tang L, Liu H (2009) Scalable learning of collective behavior based on sparse social dimensions. In: CIKM, pp 1107–1116
28. Wang X, Tang L, Gao H, Liu H (2010) Discovering overlapping groups in social media. In: ICDM, pp 569–578
29. Xu R, II DCW (2011) BARTMAP: a viable structure for biclustering. Neural Netw 24(7):709–716
30. Zhao Y, Karypis G (2001) Criterion functions for document clustering: experiments and analysis. Technical report, department of computer science. University of Minnesota