International Journal of Neural Systems, Vol. 29, No. 5 (2019) 1850052 (20 pages)

DOI: 10.1142/S0129065718500521

A Kernel Bayesian Adaptive Resonance Theory with A Topological Structure

Naoki Masuyama
Department of Computer Science and Intelligent Systems
Graduate School of Engineering, Osaka Prefecture University
1-1 Gakuen-cho Naka-ku, Sakai-Shi, Osaka 599-8531, Japan
masuyama@cs.osakafu-u.ac.jp

Chu Kiong Loo*
Department of Artificial Intelligence
Faculty of Computer Science and Information Technology
University of Malaya, 50603 Kuala Lumpur, Malaysia
**ckloo.um@um.edu.my*

Stefan Wermter
Department of Informatics
Faculty of Mathematics, Computer Science and Natural Sciences
University of Hamburg, Vogt-Koelln-Str. 30, 22527 Hamburg, Germany
wermter@informatik.uni-hamburg.de

Accepted 19 October 2018
Published Online 14 January 2019

This paper attempts to solve the typical problems of self-organizing growing network models, i.e. (a) an influence of the order of input data on the self-organizing ability, (b) an instability to high-dimensional data and an excessive sensitivity to noise, and (c) an expensive computational cost by integrating Kernel Bayes Rule (KBR) and Correntropy-Induced Metric (CIM) into Adaptive Resonance Theory (ART) framework. KBR performs a covariance-free Bayesian computation which is able to maintain a fast and stable computation. CIM is a generalized similarity measurement which can maintain a high-noise reduction ability even in a high-dimensional space. In addition, a Growing Neural Gas (GNG)-based topology construction process is integrated into the ART framework to enhance its self-organizing ability. The simulation experiments with synthetic and real-world datasets show that the proposed model has an outstanding stable self-organizing ability for various test environments.

Keywords: Unsupervised clustering; topology construction; adaptive resonance theory; kernel Bayes rule.

1. Introduction

In the recent Internet of Things (IoT) technology, massive amounts and different types of information are generated at any moment. However, it is difficult to extract useful information from this huge amount of data from various sources. In general, cluster analysis is one of the common approaches in several research fields such as statistics, machine learning and pattern recognition for extracting the hidden relationships from such a huge amount of data. k-means[1] and Expectation-Maximization (EM) algorithm[2] are typical types of unsupervised cluster learning algorithms. The Self-Organizing Map (SOM)[3] is another type of clustering which has a

*Corresponding author.

topological structure to visualize the structure of a data point. However, the k-means and the EM algorithm can only organize a predefined number of clusters, and an SOM tries to organize it by a single network even if there are multiple clusters. To handle clustering algorithms more adaptively, several types of growing networks have been introduced. Growing Cell Structure (GCS)[4] and Growing Neural Gas (GNG)[5] are significant models of growing networks which insert a new node to the region which has a maximum error. The Self-Organizing Incremental Neural Network (SOINN)[6] has successfully integrated the features of SOM and GNG.

In general, the incremental neural networks have the trade-off between the catastrophic forgetting and the ability to incrementally learn new knowledge, i.e. "plasticity-stability dilemma".[7] Plasticity can be achieved with incremental learning algorithms such as GNG and SOINN. However, these models cannot maintain the stability because they permanently insert new nodes at locations with high errors. In order to solve the plasticity-stability dilemma, ART[8] has been introduced. Fuzzy ART (FA),[9] Bayesian ART (BA)[10] and Kernel BA (KBA)[11] are considered as fundamental models in the ART family. TopoART[12] is an online hierarchical self-organizing incremental clustering algorithm which is based on FA. Although there are various advantages of TopoART, a major issue associated with the FA learning process is its sensitivity to statistical overlapping between the generated categories.[13] This sensitivity issue results in category proliferation (i.e. disordered generation of categories), which leads to a high computational cost, and reduction in clustering ability. One of the successful approaches to tackle the category proliferation problem and to improve clustering ability is the integration of Bayes' theorem to the ART architecture, namely BA. The significant properties of Bayes' Theorem-based ART are that the clusters are defined in Gaussian category which allows a category to grow and shrink by limiting a category hypervolume. However, Bayesian computation suffers from expensive computational cost due to a covariance matrix calculation. Thus, if the BA attempts to process a large number of samples with high-dimensional feature space, it is difficult to maintain feasible computational time, and the likelihood calculation is highly unstable, and global convergence is difficult to achieve.[14] KBA applies KBR[15] instead of the general Bayes' Theorem in BA, and a Correntropy[16]-based alternative similarity measurement called CIM[17] to solve the primitive issues of Bayesian computation which are described above. However, due to the original idea of ART, any input is considered as useful data for clustering. Therefore, ART-based models are potentially noise-sensitive.

In regard to self-organizing incremental clustering algorithms, such as SOINN and TopoART, the learning algorithm processes each sample as potentially informative and thus produces different cluster structures. This leads to variations in the generalization performance of cluster generation due to the proportion of error changes from one order of the sample presentation to the next one. In addition, the similarity measurement and a local error-based node insertion process are defined based on the Euclidean distance. Thus, the quality of a learned network (such as the distribution of clusters) is highly dependent on the presentation order of sample data in the learning sequence, and its calculation is unstable in the high-dimensional feature space.

In this paper, a new approach for a topological growing network, which is called Topological Kernel Bayesian ART (TKBA), is proposed to solve the primitive issues of TopoART and SOINN-based models. The cluster generation process in TKBA is performed by KBR and CIM. The KBR maintains the fast and stable computation even in a high-dimensional space due to a covariance-free Bayesian computation. The CIM is a kernel-based method from the information-theoretic learning perspective which localizes the cross information potential and quantifies the similarity between probability distributions of samples and clusters. Therefore, CIM provides a stable measurement even in a high-dimensional space. In TKBA, CIM is utilized for node insertion criteria, which provides the more generalized and stable matching criterion between samples and clusters compared with TopoART in the case of high-dimensional data and noisy environments. Furthermore, CIM is also utilized as a criterion to construct the topology networks, which contributes to the stability of topological networks. Based on the above features of KBR and CIM, TKBA achieves the stable topological cluster network generation and superior noise reduction abilities comparing with TopoART and SOINN-based models.

The main learning algorithm of TKBA is an extension of KBA.[11] Specifically, TKBA consists of the cluster generation process by KBA and a topology construction process. Therefore, TKBA also acquires the features of KBA, i.e., fast computation even in high-dimensional feature spaces, and the cluster generation process has a strong robustness against the influence of the order of given data. These features are further significant elements of the self-organizing ability in the topological models. By introducing the topological structure into KBA, the relationships between the clusters can be clearly understood. Thus, it is possible to develop an effective network construction process. In particular, we propose a new node deletion method based on topological connections and a CIM-based criterion and realize efficient cluster generation and topology construction. As mentioned above, although the learning algorithm in TKBA is similar to KBA, the cluster generation process is different due to the topology construction process. Moreover, the self-organization ability of TKBA is not only improved compared with KBA, but it is also successfully extended from functional perspectives.

This paper is divided as follows: Section 2 presents a literature review for conventional unsupervised clustering algorithms. Section 3 describes the details of TKBA. Section 4 presents simulation experiments in terms of a self-organizing capability with synthetic data, and a classification ability with real-world datasets. Concluding remarks are presented in Sec. 5.

2. Literature Review

Information modeling is based on computational intelligence attention from scientists and engineers. Typical supervised learning models are studied in several research fields due to their superior classification performance,[18–20] such as Support Vector Machine (SVM)[21] and Extreme Learning Machine (ELM).[22] Furthermore, recent innovative technologies representing deep learning[23] and their applications have been actively developed.[24–30] Besides, information modeling is applied to communication technologies and optimization methods.[31–34] However, these methods require well-structured labeled information for their learning algorithm. In contrast, unsupervised learning plays an important role in the information modeling with unlabeled information. Especially, it is regarded that the significance of cluster analysis will further increase in the growing IoT society, where a huge volume of information and data without structured labels are generated day by day.[35]

Typical types of unsupervised clustering algorithms are k-means[1] and the EM algorithm.[2] SOM[3] is one of the representative topological clustering algorithms which is utilized for data visualization. However, k-means and the EM algorithm are able to organize only a predefined number of clusters. In addition, SOM tries to organize the multiple separated clusters by a single network. GCS[4] and GNG[5] have successfully solved the problem in SOM by implementing a growing network architecture. Due to their superior performance, GNG-based clustering models have been integrated with several algorithms such as semi-supervised learning[36] and hierarchical clustering.[37] One of the problems of GNG is the excessive cluster creation. In response to this issue, Grow When Required (GWR)[38] showed an effective solution. That is, GWR inserts nodes whenever the state of the current network does not sufficiently match the input data. Another noteworthy approach is integrating GNG with CIM[16] which is called GNG-CIM.[39] CIM is a Correntropy[16]-based similarity measurement which is introduced from the information theoretic learning perspective. CIM can be considered as an alternative criterion to the Euclidean distance-based one. Adapting the clustering algorithm in the CIM sense is equivalent to reducing the localized cross information potential, and the theoretic information function that quantifies the similarity between two probability distributions based on the Gaussian kernel function. The insertion of new clusters is determined by the distribution density of existing clusters, therefore the excessive cluster insertion is suppressed.

A model combining the characteristics of SOM and GNG has also been proposed, which is known as SOINN.[6] SOINN is able to grow incrementally and to accommodate input patterns of (non)stationary data distributions, which are processed on the Euclidean distance-based similarity measurement and error-based node insertion criterion. SOINN has a great noise reduction ability due to its two layers architecture. However, the weights of the neurons are not stable and each layer has a high number of

relevant parameters. To tackle this problem, several types of SOINN-based models have been introduced. Enhanced SOINN (ESOINN)[40] simplifies the structure of SOINN to a single-layer model and reduces the number of predefined parameters. Furthermore, ESOINN has the capability that classes with a high-density overlap can be separated based on their distribution. Adjusted SOINN (ASOINN)[41] further reduces the number of parameters from ESOINN. Although these models maintain fast computation, the learned network distributions are unstable with respect to high-dimensional data because of its Euclidean distance-based network construction process. Nakamura and Hasegawa introduced the SOINN with Kernel Density Estimation (KDESOINN)[42] which can estimate the probability density function based on learned node distributions underlying the given information. Although KDESOINN achieves the good noise reduction and self-organizing capabilities, if the input data has a high dimensionality, the performance of KDESOINN is adversarily affected by the kernel density estimation (KDE) process.

GNG and SOINN achieve the ability of a learning algorithm that incorporates new knowledge into its growing network representation. Due to its adaptability and applicability, several types of research have applied the topological network to practical applications.[28,43] However, since these models insert new nodes permanently into the network, they have a potential to cause catastrophic forgetting. This trade-off is called the plasticity-stability dilemma. ART[8] is one of the representative approaches to solve the plasticity-stability dilemma. ART performs top-down learning expectations that are matched with bottom-up input. FA[9] is considered to be the leading incremental clustering algorithm in ART-based neural networks. TopoART[12] is an on-line hierarchical self-organizing incremental clustering algorithm which is based on FA. TopoART combines the advantages of ART and topology network learning, that enables the match-based fast stable learning and intrinsic self-organization which are inspired by the functions of the human brain. TopoART consists of two FAs which are called TopoARTa and TopoARTb respectively. TopoARTa performs clustering by FA using all the input data and TopoARTb generates clusters using only input data contributing to a cluster generation in TopoARTa. Although there are various advantages of FA, the major issue is its sensitivity to statistical overlap between the generated categories.[13] This sensitivity issue results in category proliferation (i.e. disordered generation of categories), which leads to high computational cost and a reduction in the classification accuracy. Several studies have been introduced to tackle the category proliferation problem and to improve the clustering and classification abilities.[44] One of the successful approaches is integrating the Bayes' Theorem to the ART architecture, namely BA[10] and KBA.[11] The significant properties of Bayes's Theorem-based ART are that the clusters are defined as Gaussian categories, which allows a category to grow and shrink by limiting a category hypervolume. Furthermore, the cluster activation with respect to ART learning is performed probabilistically and thus enables probabilistic inference using all the associated clusters from the given information.

3. Principle of Topological Kernel Bayesian ART

In this section, firstly the fundamentals of KBR and CIM are described, then the learning algorithm of KBA is briefly introduced. Finally, a topology construction process of TKBA is presented.

3.1. *Kernel Bayes' Rule*

KBR has been introduced by Fukumizu *et al.*[15] as a nonparametric kernel method for realizing Bayes' rule. In KBR, a prior probability, a posterior probability, and a likelihood are all expressed as kernel means and covariance operators, which are learned nonparametrically in the Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$. Note that, in this case, the term "nonparametric" means that the probability density function estimation is data dependent and not determined *apriori*. Furthermore, the calculation of posterior probability is performed by straightforward matrix operations on RKHS, which means that the computational cost is proportional to the sample dimensions.[45]

Let us suppose that the kernel mean of posterior probability is calculated by observed samples $X = (x_1, x_2, \ldots, x_L)(x_l \in \Re^d)$ under the cluster distribution $\{(P_{(y_k)}, y_k)\}_{k=1}^{K}(P_{(y_k)} \in$ measure space $\mathcal{S}$, and $y_k \in$ measure space $\mathcal{R})$, where y denotes the existing cluster, and $P_{(y_k)}$ denotes the prior probability

of cluster y_k. Here, the cluster posterior probability $\hat{P}(y_k|x_l)$ is defined as follows:

$$\hat{P}(y_k|x_l)_{\mathrm{KBR}} = \frac{\hat{B}_{Q_{\mathcal{S}|k}}}{\sum_{k=1}^{K} \hat{B}_{Q_{\mathcal{S}|k}}}, \tag{1}$$

where $\hat{B}_{Q_{\mathcal{S}|r}}$ $(r = 1, 2, \ldots, K)$ is a Gram matrix representation of the kernel mean of the posterior probability, which is calculated as follows:

$$\hat{B}_{Q_{\mathcal{S}|k}} = \mathbf{k}_{\mathcal{S}}^{T} Z_{\mathcal{S}|\mathcal{R}} \mathbf{k}_{\mathcal{R}}(k). \tag{2}$$

Equation (2) is defined by the following equations:

$$\mathbf{k}_S = \left(\mathcal{K}_{\mathcal{S}}(\cdot, P_{(y_1)}), \ldots, \mathcal{K}_{\mathcal{S}}(\cdot, P_{(y_K)})\right)^T \in \mathcal{H}_{\mathcal{S}}, \tag{3}$$

$$\mathbf{k}_R = \left(\mathcal{K}_{\mathcal{R}}(\cdot, y_1), \ldots, \mathcal{K}_{\mathcal{R}}(\cdot, y_K)\right)^T \in \mathcal{H}_{\mathcal{R}}, \tag{4}$$

$$Z_{S|R} := \Lambda G_R \left((\Lambda G_R)^2 + \delta_K I\right)^{-1} \Lambda, \tag{5}$$

$$\Lambda = \mathrm{Diag}\left[\left(\frac{1}{L} G_S + \varepsilon_K I\right)^{-1} \times \sum_{l=1}^{L} \gamma \mathcal{K}_{\mathcal{S}}(P_{(y_k)}, x_l)\right], \tag{6}$$

where $\mathcal{K}$ denotes the positive definite kernels. In this paper, the Gaussian kernel function is utilized as the positive definite kernel $\mathcal{K}$, i.e. $\exp(-\|x-y\|^2/(2\sigma_{\mathrm{kbr}}^2))$ to maintain the fast convergence.[46] Here, the kernel bandwidth σ_{kbr} effects the sensitivity of KBR. ϵ_K and δ_K denote the regularization constants and γ represents the weighting factor. In the original paper of KBR,[15] these factors are set as $\epsilon_K = 0.01/K$, $\delta_K = 2\epsilon_K$, and $\gamma = 1.0$, respectively, where K denotes the number of clusters in the network.

G_S and G_R denote Gram matrices which have symmetric and positive semi-definite properties. Let $\Omega = (\mathbf{v}_1, \ldots, \mathbf{v}_n)$ be a set of arbitrary vectors, then the Gram Matrix G is defined as follows:

$$G = \begin{pmatrix} \mathcal{K}(\mathbf{v}_1, \mathbf{v}_1) & \ldots & \mathcal{K}(\mathbf{v}_1, \mathbf{v}_n) \\ \mathcal{K}(\mathbf{v}_2, \mathbf{v}_1) & \ldots & \mathcal{K}(\mathbf{v}_2, \mathbf{v}_n) \\ \vdots & \vdots & \vdots \\ \mathcal{K}(\mathbf{v}_n, \mathbf{v}_1) & \ldots & \mathcal{K}(\mathbf{v}_n, \mathbf{v}_n) \end{pmatrix}. \tag{7}$$

Gram matrices G_S and G_R in KBR are defined by Eq. (7) with $P_{(y_k)}$ and y_k, respectively. The details of the derivation of Eqs. (3)–(6) are referred in Fukumizu *et al.* (2013).[15]

The summary of KBR is presented in Algorithm 1.

Algorithm 1. Kernel Bayes' Rule[15]

Require:
the observed samples: $X = (x_1, x_2, \ldots, x_L)$ $(x_l \in \Re^d)$,
the existing clusters: $Y = (y_1, y_2, \ldots, y_K)$,
the cluster distribution: $\{(P_{(Y_j)}, Y_j)\}_{j=1}^{K}$,
Tikhonov regularization constants: ε_K, δ_K,
a weighting factor: γ,
and a kernel bandwidth for positive definite kernel: σ_{kbr}.

Ensure: a posterior probability $\hat{P}(y_k|x_l)_{\mathrm{KBR}}$.

1: Compute Gram matrices G_S and G_R using Eq. (7).
2: Compute a diagonal matrix Λ in Eq. (6).
3: Compute $Z_{S|R}$ in Eq. (5).
4: Compute a kernel mean of posterior probability $\hat{B}_{Q_{\mathcal{S}|r}}$ in Eq. (2).
5: Compute a posterior probability $\hat{P}(y_k|x_l)_{\mathrm{KBR}}$ in Eq. (1).

3.2. *Correntropy-Induced Metric*

Correntropy,[16] which is a generalized similarity measure between two sample vectors, is defined as follows:

$$C_\sigma(X, Y) = E[\kappa_\sigma(X - Y)], \tag{8}$$

where $X = (x_1, x_2, \ldots, x_L)$ and $Y = (y_1, y_2, \ldots, y_K)$ are arbitrary sample vectors. κ_σ is a kernel function that satisfies the Mercer's Theorem.[21] It induces RKHS and thus it can be defined as the dot product of the two random variables in the feature space as follows:

$$C(x_i, y_j) = E[\langle \phi(x_i) \rangle, \langle \phi(y_j) \rangle], \tag{9}$$

where ϕ denotes a nonlinear mapping from the input space to the feature space based on inner product operation as follows:

$$\kappa(x_i, y_j) = [\langle \phi(x_i) \rangle, \langle \phi(y_j) \rangle]. \tag{10}$$

In practical terms, correntropy can be described by the following equation due to the finite number of data L available:

$$\hat{C}_{L,\sigma} = \frac{1}{L} \sum_{i=1}^{L} \kappa_\sigma(x_i - y_j). \tag{11}$$

Correntropy is able to induce a metric, which is called CIM. CIM can be quantified as the similarity between two probability distributions as follows:

$$\mathrm{CIM}(X, y_j) = \left[\frac{1}{L}\sum_{i=1}^{L}\{\kappa_\sigma(0) - \kappa_\sigma(x_i - y_j)\}\right]^{\frac{1}{2}}. \tag{12}$$

It can be considered that CIM is a kernel-based similarity measurement which localizes the cross information potential and quantifies the similarity between the probability distributions of samples. In this paper, the Gaussian kernel function is utilized as the kernel function κ in Eq. (12), i.e. $\exp(-\|x - y\|^2/(2\sigma_{\mathrm{cim}}^2))$. Here, the kernel bandwidth σ_{cim} effects the sensitivity of CIM.

3.3. *Topological Kernel Bayesian ART*

The learning algorithm of TKBA is an extension of KBA.[11] TKBA consists of the learning algorithm of KBA and a topology construction process between clusters which are generated by KBA. The summary of the learning algorithm of KBA is presented in Algorithm 2. The learning algorithm of KBA is divided into three processes, namely (i) cluster choice, (ii) cluster match, and (iii) cluster learning, which are indicated in Algorithm 2 in lines 1–6, lines 7–17, and lines 9–10, respectively. TKBA integrates the topology construction process into KBA as a new process as follows:

Topology construction process defines topological connections between clusters represented of connected clusters that have similar/related information. In TKBA, the topology construction process consists of a cluster deletion and edge creation/deletion as follows:

(a) *Edge Creation*

Once a cluster match occurs and the 2nd winner cluster also satisfies the match criterion as $V_{J_l} \leq V_{\mathrm{MAX}}$, the 1st and 2nd winner clusters are connected by an edge. The 1st and 2nd winner clusters are determined by $J_l = \arg\max_{k\in K}\ [\hat{P}(y_k \,|\, x_l)_{\mathrm{KBR}}]$ in the cluster choice. Unlike GNG, the edges in TKBA do not have an age information.

For the sake of a stable topology construction, cluster deletion (b) and edge deletion (c) processes are performed by a predefined cycle λ.

(b) *Cluster Deletion*

As a cluster deletion criterion, the similarity between a cluster y_k which satisfies the cluster match (i.e. $V_{J_l} \leq V_{\mathrm{MAX}}$) and the sample x_l is defined by CIM as an error E_{cim} ([0, 1]) in the cluster. For the error E_{cim}, the initial value of the error $E_{\mathrm{cim}} = 1$ is given to a newly generated cluster, and E_{cim} becomes zero when $V_{J_l} = 0$. The error E_{cim} is updated by the following process during the cluster match:

$$E_{\mathrm{cim}}^{y_k} = \min(E_{\mathrm{cim}}^{y_k}, V_{J_l}), \tag{13}$$

where V_{J_l} is calculated by CIM.

Algorithm 2. Learning Sequence in KBA[11]

Require:
the samples: $X = (x_1, x_2, \ldots, x_L)$ $(x_l \in \Re^d)$,
the existing clusters: $Y = (y_1, y_2, \ldots, y_K)$ $(y_k \in \Re^d)$,
the number of samples that have accumulated by the cluster y_{J_l}: M_J,
a kernel bandwidth for CIM: σ_{cim},
a kernel bandwidth for KBR: σ_{kbr},
Tikhonov regularization constants in KBR: ε_K, δ_K,
a weighting factor in KBR: γ,
and a maximal hypervolume: V_{MAX},
Ensure: the updated of clusters Y

1: Input a vector x_l to network.
2: **if** There is no cluster in ART network **then**
3: Create a new cluster as $y_{K+1} = x_l$.
4: **else**
5: Compute a cluster posterior probability $\hat{P}(y_k|x_l)_{KBR}$ by KBR.
6: Compute an index of winner cluster $J_l = \arg\max_{k\in K}[\hat{P}(y_k|x_l)_{KBR}]$.
7: Compute $V_{J_l} = \kappa_{\sigma_{\mathrm{cim}}}(0) - \kappa_{\sigma_{\mathrm{cim}}}(\|x_l - y_{J_l}\|)$ by CIM.
8: **if** $V_{J_l} \leq V_{\mathrm{MAX}}$ **then**
9: Update the cluster $y_{J_l} = \frac{M_J}{M_J+1}y_{J_l} + \frac{1}{M_J+1}x_l$
10: Update the accumulated counter $M_j = M_j + 1$.
11: **else**
12: **if** All the clusters are failed with vigilance test **then**
13: Create a new cluster as $y_{K+1} = x_l$
14: **else**
15: Remove the cluster y_{J_l} from selection, and continue from step 6 with next candidate cluster
16: **end if**
17: **end if**
18: **end if**
19: **if** $l < L$ **then**
20: Continue from step 1 with $l \leftarrow l + 1$
21: **end if**

If the error E_{cim} is large, it means that there is no sample near the cluster. In the proposed model, the cluster deletion is executed if the cluster has an error E_{cim} larger than the square of V_{MAX}, namely:

$$E_{\text{cim}}^{y_k} > V_{\text{MAX}}^2. \tag{14}$$

Once the above condition is fulfilled, the cluster y_k is removed from the clusters Y.

Furthermore, the clusters, in regions where the sample x_l input is infrequent, tend to be isolated from other clusters (i.e. there is no edge). It is considered that the isolated clusters are generated by noise samples. Therefore, an isolated cluster y_k, which does not have an emanating edge anymore, is also removed from the clusters Y.

(c) *Edge Deletion*

TKBA does not have an age factor for the edges, therefore, the edge deletion is performed only when an edge intersection is detected. In TKBA, the intersection edge is detected by the cross product-based detection algorithm,[47] which is applied to all the clusters that have an emanating edge. If an intersection is detected, the edge which has a maximum CIM is removed. Although the cross product-based detection algorithm[48,49] is mainly effective for the intersection in a three-dimensional space, the edge intersection is a significantly infrequent event in a high-dimensional space. Thus, TKBA utilizes the inner product-based intersection detection method which is detailed in Cormen[47] to reduce the computational load.

The summary of the learning sequence of TKBA is presented in Algorithm 3.

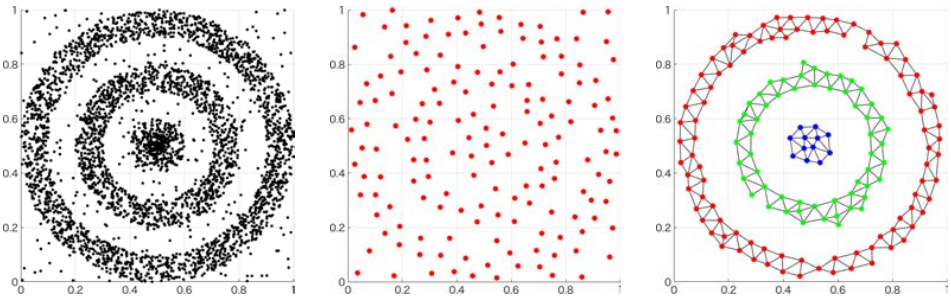
Figure 1 shows the examples of self-organizing results by KBA and TKBA. As described in Sec. 1, any input is considered as useful data for clustering due to the original idea of ART. Thus, KBA generates a lot of unnecessary clusters by the uniform Gaussian noise. In contrast, TKBA successfully organizes a concentric structure by the clearly separated topological networks. From this simple example, it can be seen that TKBA achieves to improve the self-organizing capability and to enhance the functionality of KBA by the topology construction process proposed in this section.

In case of utilizing the learned clusters Y for a classification task, an input sample is classified into a cluster y_k which has a minimum CIM between

Algorithm 3. Learning Sequence in TKBA

Require:
the samples: $X = (x_1, x_2, \ldots, x_L)$ $(x_l \in \Re^d)$,
the existing clusters: $Y = (y_1, y_2, \ldots, y_K)$ $(y_k \in \Re^d)$,
the number of samples that have accumulated at the cluster y_{J_l}: M_J,
a kernel bandwidth for CIM: σ_{cim},
a kernel bandwidth for KBR: σ_{kbr},
Tikhonov regularization constants in KBR: ε_K, δ_K,
a weighting factor in KBR: γ,
a maximal hypervolume: V_{MAX},
and a node insert cycle: λ.
Ensure: the updated clusters Y

Input a vector x_l to network.
if There is no cluster in ART network **then**
  Create a new cluster as $y_{K+1} = x_l$.
else
  Compute the cluster posterior probability $\hat{P}(y_k|x_l)_{KBR}$ by KBR.
  Compute an index of winner cluster $J_l = \arg\max_{k \in K} [\hat{P}(y_k|x_l)_{KBR}]$.
  Compute $V_{J_l} = \kappa_{\sigma_{\text{cim}}}(0) - \kappa_{\sigma_{\text{cim}}}(\|x_l - y_{J_l}\|)$ by CIM.
  if $V_{J_l} \leq V_{\text{MAX}}$ **then**
    Update the cluster $y_{J_l} = \frac{M_J}{M_J+1} y_{J_l} + \frac{1}{M_J+1} x_l$
    Update the accumulated counter $M_j = M_j + 1$.
    if the 2nd winner cluster also satisfies the Cluster Match criterion **then**
      Make an edge between 1st and 2nd winner clusters.
    end if
  else
    if All the clusters are failed with vigilance test **then**
      Create a new cluster as $y_{K+1} = x_l$
    else
      Remove the cluster y_{J_l} from selection, and continue from step 6 with next candidate cluster
    end if
  end if
end if
if l is multiple of a topology construction cycle λ **then**
  if The existing clusters Y satisfy $E_{\text{cim}}^{y_k} > V_{\text{MAX}}^2$ **then**
    Remove the cluster y_k and its edges.
  end if
  Remove all the isolated clusters.
  for $k \in \{1 \ldots K\}$ **do**
    Remove the longest edge from y_k, which makes the intersection.
  end for
end if
if $l < L$ **then**
  Continue from step 1 with $l \leftarrow l + 1$
end if



(a) Input (b) KBA (c) TKBA

Fig. 1. Self-organizing results of KBA and TKBA (4000 data points with 10% uniform Gaussian noise). The parameters of each model are as follows; KBA: a maximal hypervolume $V_{\mathrm{MAX}} = 0.15$, a kernel bandwidth for KBR $\sigma_{\mathrm{kbr}} = 0.1$, and a kernel bandwidth for CIM $\sigma_{\mathrm{cim}} = 0.1$, and TKBA: a maximal hypervolume $V_{\mathrm{MAX}} = 0.12$, a kernel bandwidth for KBR $\sigma_{\mathrm{kbr}} = 0.1$, a kernel bandwidth for CIM $\sigma_{\mathrm{cim}} = 0.1$, and a topology construction cycle $\lambda = 400$.

Algorithm 4. Inference Sequence of TKBA

Require:
the given samples $X' = (x'_1, x'_2, \ldots, x'_T)$ ($x'_t \in \Re^d$), and the parameter of clusters $Y = (y_1, y_2, \ldots, y_K)$.

Ensure: the nearest cluster y_k corresponding to the given sample x'_t

1: Input a vector x'_t to a topological network.
2: Compute CIM between x'_t and Y.
3: Select a cluster y_k which has a minimum CIM.
4: **if** $t < T$ **then**
5: Continue from step 1 with $t \leftarrow t + 1$
6: **end if**

the input sample and clusters. This procedure is summarized in Algorithm 4. In addition, by averaging the counts of label information of clusters in a topological network during a learning sequence, the label attribute of the topological network can be determined, and it will be utilized for supervised learning.

3.4. *Computational Complexity*

In this section, the complexity of typical unsupervised clustering algorithms is discussed. In addition, the complexity of representative supervised classification algorithms is also presented as a reference.

The complexity of an algorithm can generally be divided into two factors, i.e. (i) a computational complexity which deals with how long the algorithm is executed, and (ii) a space complexity which focuses on how much memory is used by an algorithm. In this paper, we focus on the computational complexity. It is represented by a Big-$\mathcal{O}$ notation with the symbols N, D, C, I, and L which denote the number of samples, the dimensions of the sample, the number of clusters, the number of iterations, and the size of batch samples, respectively.

As mentioned in Sec. 2, ASOINN is an extension model of ESOINN. It can be considered that the computational complexity of ASOINN shows to be equivalent to ESOINN, i.e. $\mathcal{O}(NC^2I)$, which is introduced in Asadi *et al.*[50] KDESOINN performs KDE with N samples for improving its clustering capability. The rest of the procedure in KDESOINN is the same as in ASOINN. The computational complexity of KDE indicates $\mathcal{O}(NDC)$,[51] therefore, the computational complexity of KDESOINN can be deduced as $\mathcal{O}(N^2DC^3I)$.

The computational complexity of FA has been assessed as $\mathcal{O}(NC^2I) + \mathcal{O}(NDCI)$ considering N samples with I iterations.[52] Here, $\mathcal{O}(NC^2I)$ is defined by a winner cluster searching process, and $\mathcal{O}(NDCI)$ is derived from a recursive calculation in cluster matching. In regard to TKBA, the complexity of a topology construction is low compared with a clustering process. Thus, the computational complexity of TKBA can be regarded as equivalent to FA. From a structural perspective, TopoART can be considered as a union of two FAs. An input sample for the second FA in TopoART is selected by the first FA, therefore, the second FA potentially has $\mathcal{O}(DC)$. As a result, the computational complexity of TopoART is defined as $\mathcal{O}(NDC^3I) + \mathcal{O}(ND^2C^2I)$.

The summary of the computational complexities is shown in Table 1. Compared to TKBA,

Table 1. Comparison of Computational Complexity. N, D, C, I, and L denote the # of samples, the dimensions of sample, the # of clusters, the # of iterations, and the size of batches, respectively.

Model	Order	Defined in
TKBA	$\mathcal{O}(NC^2I) + \mathcal{O}(NDCI)$	—
KDESOINN	$\mathcal{O}(N^2DC^3I)$	—
ASOINN	$\mathcal{O}(NC^2I)$	—
TopoART	$\mathcal{O}(NDC^3I) + \mathcal{O}(ND^2C^2I)$	—
k-means	$\mathcal{O}(NCDI)$	Ref. 53
kSVM	$\mathcal{O}(N^3)$	Ref. 54
ELM	$\mathcal{O}(NL^2 + L^3)$	Ref. 55

KDESOINN, ASOINN, and TopoART, ASOINN shows the lowest computational complexity. Being ART-based algorithms, TKBA and TopoART take an additional computational cost. However, TKBA is composed by a single layer, thus the computational complexity is lower than that of TopoART. Compared to TKBA and KDESOINN, the multipliers of N and C of KDESOINN are larger than of TKBA. Therefore, it is considered that TKBA has a lower computational complexity than KDESOINN.

4. Simulation Experiments

This section presents the simulation experiments for evaluating the self-organizing and classification abilities of TKBA. In this paper, SOINN-based models, i.e. KDESOINN[42] and ASOINN,[41] and the ART-based model, i.e. TopoART,[12] are utilized for comparison. Regarding TopoART, all evaluations are conducted with the second layer of TopoART, i.e. the TopoARTb network. The details of the simulation settings are presented in the respective sections.

4.1. *Effect of Parameters in TKBA*

Firstly, to provide a better understanding of parameters in TKBA, the self-organizing results with several parameter settings are presented. The dataset for this demonstration consists of 2D synthetic data as shown in Fig. 2. The dataset is divided into six distributions with 15k data samples each as A, B, C, D, E and F. Here, A and B satisfy 2D Gaussian distribution. C and D are concentric-ring distributions. E and F are sinusoidal distributions. In this experiment, we utilize the dataset shown in Fig. 2(a). The other datasets (Figs. 2(b)–2(d)) are utilized in the next section.

Table 2. Basic parameter settings for TKBA.

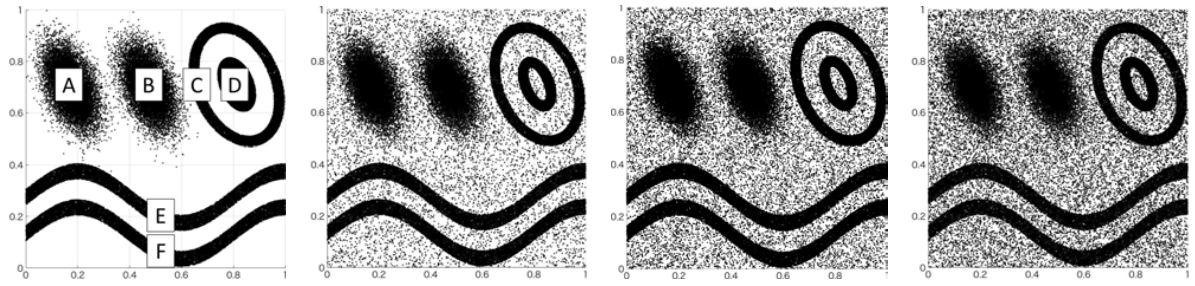
Model	Parameter		
TKBA	a kernel bandwidth for CIM σ_{cim}	:	0.025
	a kernel bandwidth for KBR σ_{kbr}	:	1.0
	a hypervolume V_{MAX}	:	0.25
	a topology construction cycle λ	:	400

TKBA has four significant parameters that have a strong influence on its performance. The basic parameters of TKBA are defined as shown in Table 2. Under this setting, the self-organizing result to the dataset in Fig. 2(a) is depicted in Fig. 3(b). The rest of the results in Fig. 3 is obtained if the parameter $\sigma_{\mathrm{cim}} = 0.015, 0.050, 0.100$ and 0.200. In Fig. 3, a red circle denotes generated clusters and a black line represents edges between clusters. Similar to Fig. 3, Figs. 4–6 show the effect of parameters σ_{kbr}, V_{MAX}, and λ, respectively. Here, same as in Fig. 3, the red circle denotes generated clusters and the black line represents edges between clusters. Based on Fig. 3, when the parameter σ_{cim} increases, the number of clusters decreases and the distance between clusters increases. In addition, each data distribution tends to combine itself. Therefore, σ_{cim} has a strong influence on the number of clusters. In contrast, as shown in Fig. 4, the influence of the parameter σ_{kbr} to the number of clusters is quite low. Focusing on Fig. 5, V_{MAX} has an influence on the number of clusters, however, its impact is not so high compared to σ_{cim}. Furthermore, since V_{MAX} has a role as a matching criterion, it is preferable that its value is fixed. The parameter λ also affects on the number of clusters, however, it turns out that its impact decreases if the λ has a larger value.

From the above discussion, although TKBA has four significant parameters, the parameter σ_{cim} plays an important role in the performance of TKBA. Therefore, we focus only on the parameter σ_{cim} in the following section.

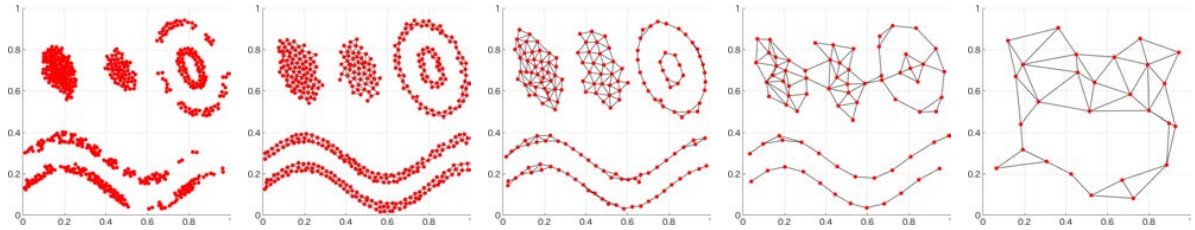
4.2. *Self-organizing Ability*

The self-organizing ability is examined with the same 2D synthetic data used in the previous section (Fig. 2). In this simulation, to evaluate the robustness of models, uniform random noise is added to the original dataset. In this research, a uniformly distributed random data is added to the data space as



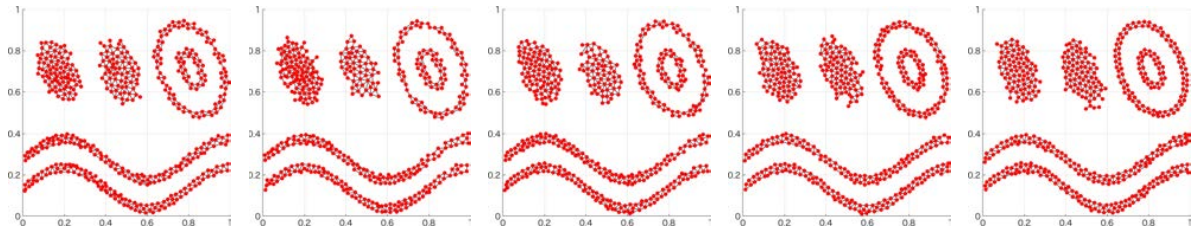
(a) Noise 0% (b) Noise 10% (c) Noise 20% (d) Noise 30%

Fig. 2. 2D synthetic dataset for self-organizing experiments. The dataset is divided into six parts as two Gaussian distributions (A and B), two concentric-ring distributions (C and D), and two sinusoidal distributions (E and F). Each distribution consists of 15k data points without noise. (a)–(d): different levels of additive noise.



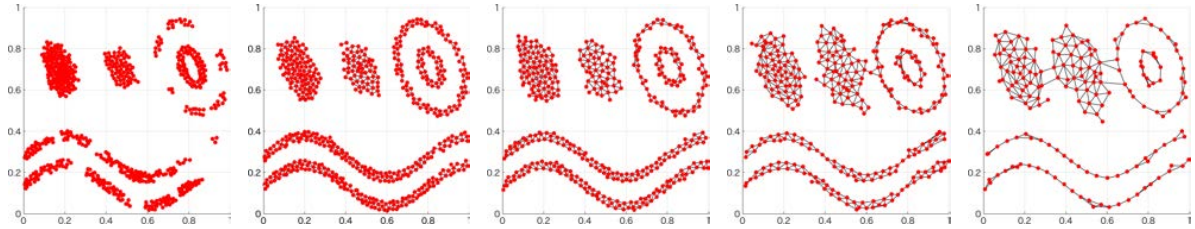
(a) $\sigma_{\text{cim}} = 0.015$ (b) $\sigma_{\text{cim}} = 0.025$ (c) $\sigma_{\text{cim}} = 0.050$ (d) $\sigma_{\text{cim}} = 0.100$ (e) $\sigma_{\text{cim}} = 0.200$

Fig. 3. Effect of parameter σ_{cim} in TKBA. The rest of the parameters in TKBA are the same as in Table 2.



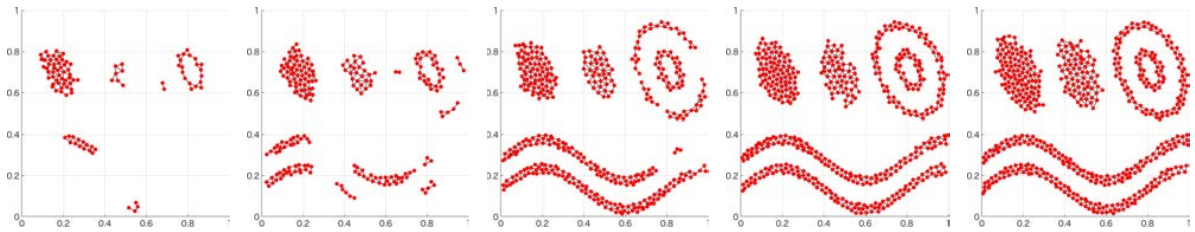
(a) $\sigma_{\text{kbr}} = 0.25$ (b) $\sigma_{\text{kbr}} = 0.50$ (c) $\sigma_{\text{kbr}} = 1.50$ (d) $\sigma_{\text{kbr}} = 3.00$ (e) $\sigma_{\text{kbr}} = 6.00$

Fig. 4. Effect of parameter σ_{kbr} in TKBA. The rest of the parameters in TKBA are the same as in Table 2.



(a) $V_{\text{MAX}} = 0.10$ (b) $V_{\text{MAX}} = 0.20$ (c) $V_{\text{MAX}} = 0.30$ (d) $V_{\text{MAX}} = 0.50$ (e) $V_{\text{MAX}} = 0.75$

Fig. 5. Effect of parameter V_{MAX} in TKBA. The rest of the parameters in TKBA are the same as in Table 2.



(a) $\lambda = 50$ (b) $\lambda = 100$ (c) $\lambda = 200$ (d) $\lambda = 500$ (e) $\lambda = 1000$

Fig. 6. Effect of parameter λ in TKBA. The rest of the parameters in TKBA are the same as in Table 2.

"noise" for distributions A, B, C, D, E, and F. Therefore, the noise in this experiment does not organize another distribution, and it is considered that the noise belongs to one of the six distributions. The simulation experiments are conducted in two environments, i.e. stationary and nonstationary environments. In the stationary environment, the data samples are randomly selected from the whole dataset. In the nonstationary environment, the distributions of A–F are sequentially exposed to the network. In the experiment, each data sample is exposed to the network only once.

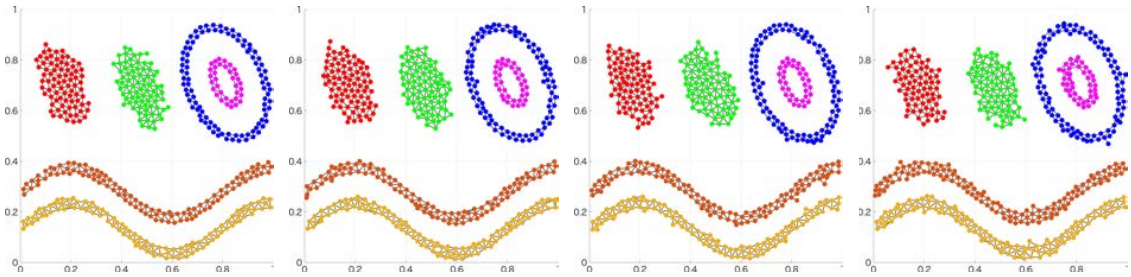
The parameter settings in each model for the self-organizing ability are summarized in Tables 2 and 3. The parameters in each model are tuned by empirically achieving the best Normalized Mutual Information (NMI),[56] Micro and Macro F-Scores[57] (i.e. all the results show 1.0) in the case of the dataset as shown in Fig. 7(a). The dataset in Fig. 7(a) does not contain any noise information, and therefore it is easy to set parameters to achieve the best results. The obtained parameters are utilized to problems with noise thereby the robustness of models can be estimated. In regard to KBR, the parameters follow Fukumizu *et al.* (2013)[15] as: regularization constants ϵ_K and δ_K, and a weighting factor γ are set as $\epsilon_K = 0.01/K$, $\delta_K = 2\epsilon_K$, and $\gamma = 1.0$, where K denotes the number of clusters in the network.

Firstly, the self-organizing results are shown. Figure 7–10 show the generated topological networks in TKBA, KDESOINN, ASOINN, and TopoART under a stationary environment, respectively. Focusing on Figs. 9 and 10, the noise reduction ability of ASOINN and TopoART is insufficient compared with TKBA and KEDSOINN. In Fig. 8, as the noise ratio increases, the topological network collapses in KDESOINN. In contrast, in Fig. 7, TKBA shows an outstanding noise reduction ability with a stable network organization.

Figures 11–14 show the generated topological networks in TKBA, KDESOINN, ASOINN, and TopoART, respectively under the nonstationary environment. Similar to that, under the stationary environment, ASOINN and TopoART suffer from sensitivity to noise in the topology construction. In Fig. 12, KDESOINN generates a topological network without collapsing as in Fig. 8(d). However, the clusters are connected between different distributions of the dataset. In contrast, same as that under stationary environment, TKBA shows a superior

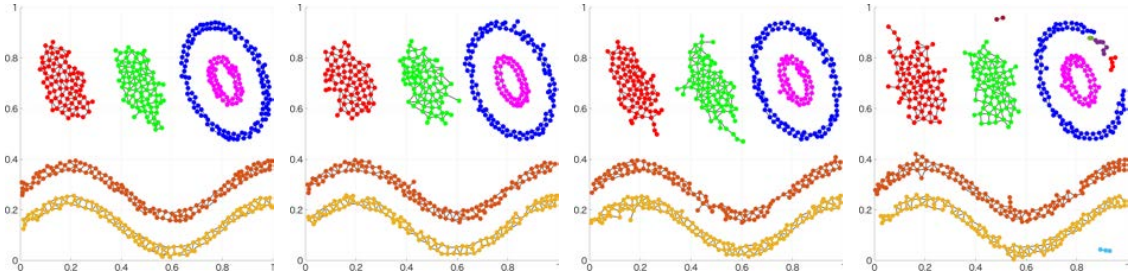
Table 3. Basic parameter settings for comparison models.

Model	Parameter		
KDESOINN	node insert cycle λ	:	50
	maximum age of edge age_{MAX}	:	20
	parameter for threshold ρ	:	0.02
ASOINN	node insert cycle λ	:	200
	maximum age of edge age_{MAX}	:	10
	parameter for node deletion c	:	0.5
TopoART (a, b)	node insert cycle τ	:	(300, 300)
	learning rate β	:	(1.00, 0.65)
	parameter for node deletion ϕ	:	(2, 2)
	vigilance parameter ρ	:	(0.90, 0.95)



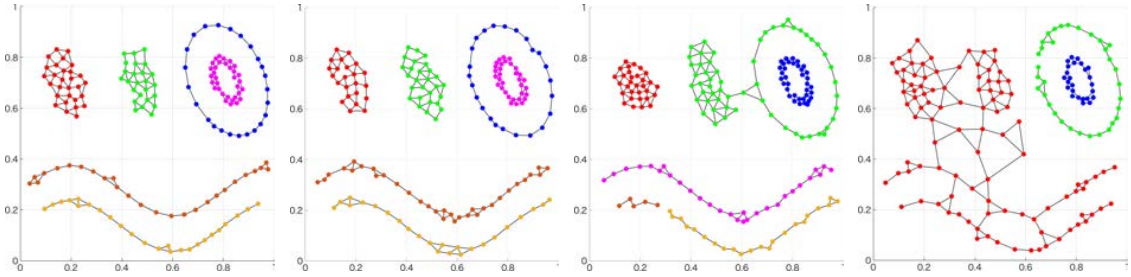
(a) Noise 0% (b) Noise 10% (c) Noise 20% (d) Noise 30%

Fig. 7. Topology construction of TKBA with the stationary environment.



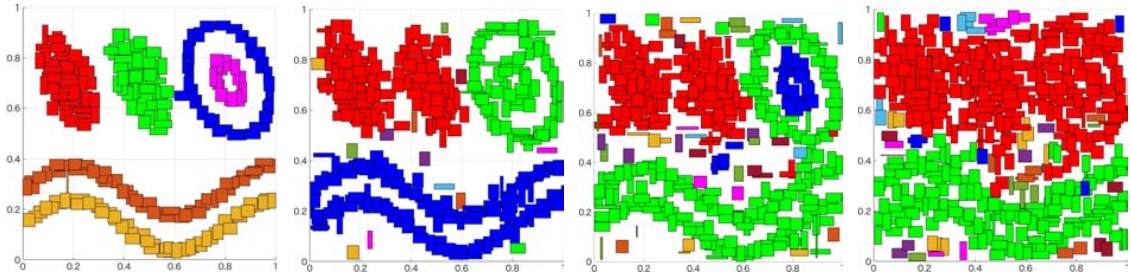
(a) Noise 0% (b) Noise 10% (c) Noise 20% (d) Noise 30%

Fig. 8. Topology construction of KDESOINN with the stationary environment.



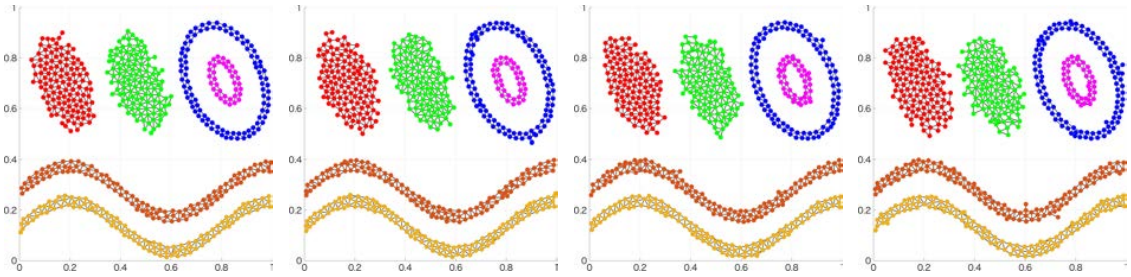
(a) Noise 0% (b) Noise 10% (c) Noise 20% (d) Noise 30%

Fig. 9. Topology construction of ASOINN with the stationary environment.



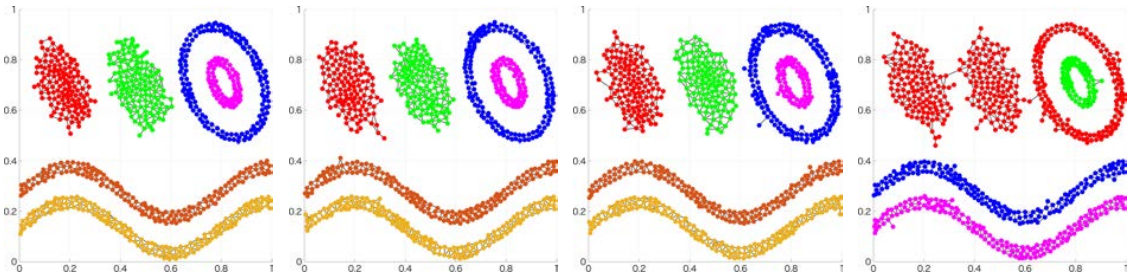
(a) Noise 0% (b) Noise 10% (c) Noise 20% (d) Noise 30%

Fig. 10. Topology construction of TopoART with the stationary environment.



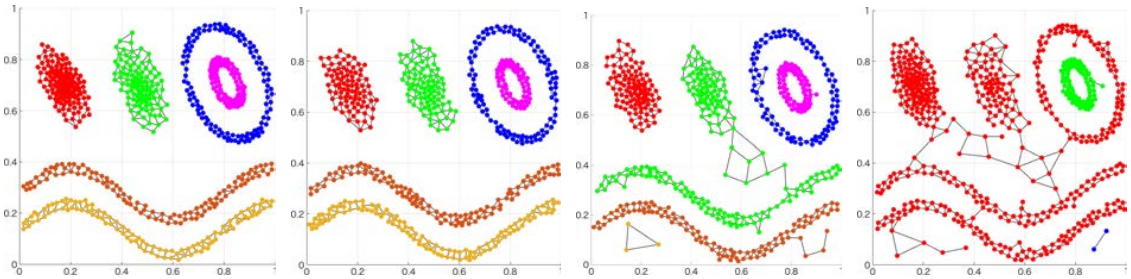
(a) Noise 0% (b) Noise 10% (c) Noise 20% (d) Noise 30%

Fig. 11. Topology construction of TKBA with the nonstationary environment.



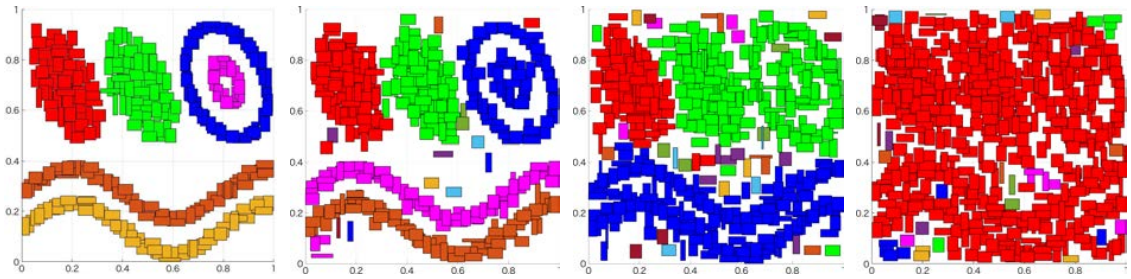
(a) Noise 0% (b) Noise 10% (c) Noise 20% (d) Noise 30%

Fig. 12. Topology construction of KDESOINN with the nonstationary environment.



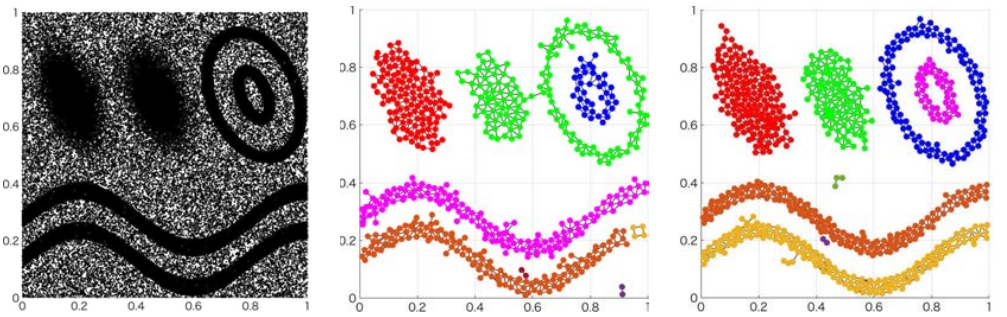
(a) Noise 0% (b) Noise 10% (c) Noise 20% (d) Noise 30%

Fig. 13. Topology construction of ASOINN with the nonstationary environment.



(a) Noise 0% (b) Noise 10% (c) Noise 20% (d) Noise 30%

Fig. 14. Topology construction of TopoART with the nonstationary environment.



(a) Noise 50 % (b) Stationary Environment (c) Non-stationary Environment

Fig. 15. Examples of failure topology construction of TKBA. (a) input data with 50% noise added, (b) data was given in stationary, and (c) data was given in nonstationary.

self-organizing ability with a strong noise reduction performance.

Figure 15 shows examples of a failure topology construction of TKBA with 50% noise added. As the noise ratio rises, the distribution of the generated clusters in TKBA becomes unstable. In addition, similar to KDESOINN and ASOINN, TKBA also tends to combine clusters which have different distributions, or generates useless clusters.

Compared with the generated topological networks in stationary and nonstationary environments, SOINN-based models, namely ASOINN and KDESOINN, generate networks that have huge gaps (in terms of the density of clusters) depending on the input order of data samples. On the other hand, ART-based models like TKBA and TopoART generate a similar topology in each environment. It can be seen that the ART-based approach has the superiority in the robustness of the self-organizing ability for the input order of sample data.

Secondly, the quality of topological networks is discussed from the statistical perspective. In this paper, the quality of a network means how well data can be represented by the generated networks. In regards to the quality assessment of the learned topological network which is generated from the dataset containing noise, the data samples without noise are exposed to the network, and the nearest class from each data sample is searched to calculate NMI and Micro and Macro F-Scores. In this experiment, we assigned the label information to each data distribution A to F as a class 1–6, respectively for calculating NMI, Micro, and Macro F-Scores. In addition, to reduce the bias resulting from the random sampling of training data, 10-fold cross-validation is utilized. Moreover, all the experiments are conducted in 20 trials to obtain the consistent averaging results. In addition, the Wilcoxon signed-rank test[58] is employed to determine whether one algorithm has a statistical significance difference and the null hypothesis is rejected at the significant level of 0.05.

The results are summarized in Table 4. The generated topological networks are clearly separated into six distributions as shown in Figs. 7 and 11. TKBA shows the highest score in each measurement. Furthermore, the standard deviation of each score indicates an outstanding stability of the self-organizing capability of TKBA. The superiority to the stability of ART-based models is also shown by the number of clusters and classes, which have similar results in stationary and nonstationary environments.

From the above results, TKBA has robust self-organizing abilities both in noisy environments and the influence of the input order of sample data.

4.3. *Classification Ability*

This section presents the comparison of TKBA, KDESOINN, ASOINN, and TopoART in terms of the classification performance, the robustness, and the processing time per sample of each model by utilizing 12 real-world datasets from the UCI repository of machine learning databases.[59] The datasets in this experiment are summarized in Table 5. In addition, the results of k-means[1] are shown as a standard unsupervised classification algorithm. Note that the value of k in k-means is set as the number

Table 4. Quality assessment of self-organizing ability on the synthetic dataset. A symbol † represents TKBA has a statistically significant difference ($p < 0.05$) by the Wilcoxon signed-rank test on NMI, and Micro/Macro- F-score.

Input Sequence	Noise Rate	Measurement	TKBA	KDESOINN	ASOINN	TopoART
Stationary	10 [%]	# of Clusters (SD)	520.100 (7.473)	564.400 (8.540)	255.900 (10.597)	333.300 (10.105)
		# of Classes (SD)	6.000 (0.000)	6.200 (0.789)	6.450 (0.686)	33.950 (7.163)
		NMI (SD)	**0.998** (0.000)	0.991 (0.021)	0.993 (0.015)	0.843†(0.102)
		Micro F-score (SD)	**1.000** (0.000)	0.983†(0.053)	0.991 (0.037)	0.688†(0.179)
		Macro F-score (SD)	**1.000** (0.000)	0.977†(0.070)	0.988†(0.050)	0.513†(0.192)
	20 [%]	# of Clusters (SD)	541.900 (9.597)	576.400 (14.864)	256.050 (13.028)	417.100 (9.341)
		# of Classes (SD)	6.200 (0.422)	6.500 (0.707)	5.500 (0.827)	48.100 (6.828)
		NMI (SD)	**0.998** (0.000)	0.997 (0.001)	0.942†(0.045)	0.520†(0.318)
		Micro F-score (SD)	**1.000** (0.000)	0.999 (0.000)	0.866†(0.102)	0.405†(0.201)
		Macro F-score (SD)	**1.000** (0.000)	0.942†(0.074)	0.823†(0.134)	0.242†(0.185)
	30 [%]	# of Clusters (SD)	562.400 (8.682)	596.700 (14.024)	270.350 (11.744)	481.350 (8.940)
		# of Classes (SD)	6.300 (0.483)	6.900 (1.524)	3.850 (1.226)	43.250 (7.468)
		NMI (SD)	**0.998** (0.000)	0.934†(0.070)	0.753†(0.141)	0.245†(0.301)
		Micro F-score (SD)	**0.999** (0.000)	0.849†(0.165)	0.593†(0.166)	0.167†(0.170)
		Macro F-score (SD)	**0.971** (0.060)	0.709†(0.171)	0.506†(0.184)	0.076†(0.090)
Nonstationary	10 [%]	# of Clusters (SD)	609.000 (10.100)	829.200 (28.397)	753.900 (25.713)	376.700 (8.417)
		# of Classes (SD)	6.000 (0.000)	6.400 (0.699)	7.550 (1.572)	24.900 (3.523)
		NMI (SD)	**0.999** (0.000)	0.992 (0.021)	0.963†(0.045)	0.919†(0.064)
		Micro F-score (SD)	**1.000** (0.000)	0.983†(0.053)	0.916†(0.101)	0.816†(0.142)
		Macro F-score (SD)	**1.000** (0.000)	0.935†(0.087)	0.856†(0.125)	0.650†(0.159)
	20 [%]	# of Clusters (SD)	626.800 (10.952)	819.200 (12.026)	730.650 (22.051)	453.850 (10.742)
		# of Classes (SD)	6.000 (0.000)	6.200 (0.919)	6.950 (1.932)	41.650 (5.304)
		NMI (SD)	**0.998** (0.000)	0.954†(0.064)	0.818†(0.128)	0.658†(0.192)
		Micro F-score (SD)	**1.000** (0.000)	0.900†(0.140)	0.669†(0.153)	0.444†(0.153)
		Macro F-score (SD)	**1.000** (0.000)	0.832†(0.184)	0.589†(0.172)	0.264†(0.146)
	30 [%]	# of Clusters (SD)	643.800 (9.259)	812.400 (17.315)	721.100 (20.455)	513.900 (9.492)
		# of Classes (SD)	6.700 (1.059)	5.900 (1.524)	6.200 (1.508)	32.150 (5.887)
		NMI (SD)	**0.992** (0.021)	0.913†(0.089)	0.761†(0.107)	0.125†(0.240)
		Micro F-score (SD)	**0.983** (0.053)	0.816†(0.183)	0.567†(0.166)	0.200†(0.068)
		Macro F-score (SD)	**0.935** (0.087)	0.705†(0.216)	0.472†(0.163)	0.063†(0.046)

Table 5. Configurations of UCI dataset.

Dataset	Classes	Attributes	Samples
Iris	3	4	150
Skin	2	4	245,057
Shuttle	7	9	58,000
Poker Hand	10	11	10,25,010
Wine	3	13	178
Zoo	7	16	101
Letter	26	16	20,000
Thyroid	2	21	7200
Ionosphere	2	34	351
Sonar	2	60	208
Optdigits	10	64	5620
Semeion	10	256	1593

of classes in each individual UCI dataset. Furthermore, the results of typical supervised classification algorithms, i.e. kernel SVM (kSVM)[60] and ELM,[22] are also shown.

The parameters of each model are summarized in Table 6. The rest of the parameters are the same as those in Tables 2 and 3. In addition, the parameters of KBR are also the same as those in Sec. 4.1. The parameters in Table 6 are tuned by preliminary experiments utilizing the fewer number of samples in each dataset. We repeatedly calculate with different parameter settings and adopt the parameters that obtained the highest NMI. Since parameter settings are changed in an arbitrarily fixed range when changing the parameter condition, there is a possibility of finding an even better parameter setting by sophisticated optimization algorithms.

Similar to the self-organizing ability assessment, 10-fold cross-validation method is utilized with 20 trials for obtaining consistent averaging results. In addition, during the network learning sequence, each data sample is shown 100 times. Although several

Table 6. Parameter settings for classification experiment on UCI datasets. The rest of the parameters of each model are the same as Tables 2 and 3.

Dataset	TKBA σ_{cim}	KDESOINN ρ	ASOINN c	TopoART (ρ_{a}, ρ_{b})	kSVM σ_{ksvm}	ELM # of Hidden Nodes
Iris	0.50	1.00	1.00	(0.550, 0.775)	1.2	50
Skin	0.10	10.50	0.50	(0.500, 0.750)	1.2	500
Shuttle	0.37	0.31	0.40	(0.980, 0.990)	1.4	200
Poker Hand	0.40	5.00	1.50	(0.700, 0.850)	1.0	500
Wine	2.50	2.30	0.50	(0.200, 0.600)	3.5	1500
Zoo	3.50	8.00	0.30	(0.550, 0.775)	4.5	1000
Letter	1.90	3.70	0.11	(0.540, 0.770)	2.0	500
Thyroid	2.10	3.00	0.10	(0.840, 0.920)	1.0	50
Ionosphere	1.80	0.50	1.00	(0.020, 0.510)	3.5	2500
Sonar	5.60	2.50	1.00	(0.010, 0.505)	3.5	2500
Optdigits	4.70	5.20	0.70	(0.0050, 0.5025)	4.0	1400
Semeion	16.90	15.20	1.50	(0.900, 0.950)	9.0	12,000

datasets provide training and test data independently, both data are integrated randomly and cross-validation is applied to the entire data. Furthermore, the Wilcoxon signed-rank test is employed to determine whether one algorithm has a statistically significant difference and the null hypothesis is rejected at a significant level of 0.05.

Table 7 shows the results of classification performance. Focusing on ASOINN, it can be seen that the classifier generation process is unstable due to the fact that the standard deviation is larger than that of other models, even if each measurement result generally shows a high score. KDESOINN shows smaller standard deviations than those of ASOINN. However, the model is likely to generate excessive clusters in the network. Regards to TopoART, especially in the case of high-dimensional data, the model has the tendency to generate excessive classes much more than other models, due to the shortcomings of FA. In contrast, TKBA shows higher measurement scores than other models while maintaining small standard deviations and a smaller number of clusters and classes. In addition, the values of the Micro and Macro F-score show similar results. It can be stated that the model has a higher stability than comparison models for a class-imbalance problem, except for the Poker Hand dataset which is however an imbalanced dataset.

To compare the robustness and adaptability of the models, Adjusted Rand Index (ARI)[61] is considered. Let n_{ij} be the number of samples that are in both class u_i and cluster v_j. Let $n_{i\cdot}$ and $n_{\cdot j}$ be the number of samples in class u_i and cluster v_j, respectively. The ARI is defined as follows:

$$\text{ARI} = \frac{\sum_{i,j}\binom{n_{ij}}{2}\left[\sum_i\binom{n_{i\cdot}}{2}\sum_j\binom{n_{\cdot j}}{2}\right]\Big/\binom{n}{2}}{\frac{1}{2}\left[\sum_i\binom{n_{i\cdot}}{2}+\sum_j\binom{n_{\cdot j}}{2}\right] - \left[\sum_i\binom{n_{i\cdot}}{2}+\sum_j\binom{n_{\cdot j}}{2}\right]\Big/\binom{n}{2}}. \quad (15)$$

In general, a higher ARI shows that the model has a better performance.

In Fig. 16, it is difficult for TopoART to handle high-dimensional samples. In contrast, TKBA shows the highest ARI for the majority of datasets. Therefore, it can be seen that TKBA has superior robustness and adaptability to different types of problems.

In Fig. 17, the processing time per sample for each model is summarized. In general, as the number of clusters and classes of the model and the dimensionality of samples increase, the processing time increases. The processing time of TopoART for the Poker Hand dataset shows the longest processing time. This is because TopoART has generated excessive clusters. From Fig. 17, it can be seen that TKBA is able to maintain a fast processing even in case of high-dimensional samples.

In summary, from the results in this section, it can be stated that TKBA is able to perform the fast and stable computation with outstanding noise reduction capability even in a high-dimensional space. Furthermore, TKBA shows superior robustness for different types of tasks. Thus, TKBA is a successful approach for enhancing the capabilities of the topological growing network algorithms.

Table 7. Classification performance of TKBA, KDESOINN, ASOINN and TopoART. A symbol † represents TKBA has a statistically significant difference ($p < 0.05$) by the Wilcoxon signed-rank test on NMI, and Micro-/Macro-F-score. The results of k-means (with an ideal k), kSVM and ELM are for the references.

Dataset	Measurement	TKBA	KDESOINN	ASOINN	TopoART	k-means	kSVM	ELM
Iris	# of Clusters (SD)	21.830 (1.843)	39.175 (4.754)	44.610 (3.421)	18.310 (1.434)	3.000 (0.000)	3.000 (0.000)	3.000 (0.000)
	# of Classes (SD)	6.870 (0.548)	2.640 (0.595)	5.560 (1.857)	4.625 (1.380)	2.950 (0.137)	3.000 (0.000)	3.000 (0.000)
	NMI (SD)	**0.802** (0.102)	0.753†(0.025)	0.753†(0.073)	0.695†(0.128)	0.718 (0.105)	0.963 (0.046)	0.875 (0.118)
	Micro-F-score (SD)	**0.891** (0.072)	0.666†(0.010)	0.766†(0.107)	0.695†(0.103)	0.804 (0.098)	0.963 (0.046)	0.943 (0.057)
	Macro-F-score (SD)	**0.884** (0.072)	0.556†(0.012)	0.714†(0.150)	0.621†(0.138)	0.775 (0.133)	0.963 (0.046)	0.942 (0.058)
Skin	# of Clusters (SD)	444.580 (11.094)	2.400 (0.758)	226.540 (12.592)	84.780 (1.135)	2.000 (0.000)	0.000 (0.000)	2.000 (0.000)
	# of Classes (SD)	12.140 (1.859)	1.000 (0.000)	13.760 (3.137)	2.920 (1.043)	2.000 (0.000)	2.000 (0.000)	2.000 (0.000)
	NMI (SD)	**0.806** (0.034)	0.000†(0.000)	0.671†(0.049)	0.000†(0.000)	0.000 (0.000)	0.974 (0.004)	0.982 (0.002)
	Micro-F-score (SD)	**0.974** (0.008)	0.792†(0.003)	0.946†(0.013)	0.792†(0.003)	0.792 (0.003)	0.998 (0.000)	0.999 (0.000)
	Macro-F-score (SD)	**0.962** (0.011)	0.442†(0.001)	0.924†(0.017)	0.442†(0.001)	0.442 (0.001)	0.997 (0.001)	0988 (0.000)
Shuttle	# of Clusters (SD)	532.200 (10.186)	414.767 (13.805)	181.433 (11.150)	406.967 (9.496)	7.000 (0.000)	7.000 (0.000)	7.000 (0.000)
	# of Classes (SD)	7.900 (1.202)	8.600 (2.015)	7.433 (2.107)	172.300 (13.108)	4.967 (0.535)	7.000 (0.000)	7.000 (0.000)
	NMI (SD)	0.679 (0.013)	0.718 (0.064)	**0.764** (0.084)	0.689 (0.093)	0.471 (0.060)	0.994 (0.002)	0.997 (0.002)
	Micro-F-score (SD)	0.927 (0.004)	0.931 (0.023)	**0.959**†(0.018)	0.926 (0.039)	0.853 (0.018)	0.994 (0.002)	0.997 (0.002)
	Macro-F-score (SD)	0.434 (0.061)	0.576†(0.082)	**0.602**†(0.090)	0.580†(0.077)	0.418 (0.087)	0.994 (0.002)	0.997 (0.002)
Poker Hand	# of Clusters (SD)	4.667 (1.963)	140.900 (12.844)	364.167 (7.706)	1000.000 (0.000)	10.000 (0.000)	9.000 (0.000)	9.000 (0.000)
	# of Classes (SD)	4.667 (1.963)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	8.867 (0.755)	9.000 (0.000)	9.000 (0.000)
	NMI (SD)	**0.000** (0.001)	**0.000**†(0.000)	**0.000**†(0.000)	**0.000**†(0.000)	0.00 (0.000)	0.037 (0.004)	0.023 (0.003)
	Micro-F-score (SD)	0.462 (0.048)	**0.502**†(0.002)	**0.502**†(0.002)	**0.502**†(0.002)	0.502 (0.002)	0.529 (0.007)	0.537 (0.006)
	Macro-F-score (SD)	**0.093** (0.013)	0.077†(0.005)	0.077†(0.005)	0.077†(0.005)	0.078 (0.006)	0.198 (0.029)	0.140 (0.012)
Wine	# of Clusters (SD)	23.000 (2.169)	72.675 (4.834)	53.470 (4.157)	27.980 (1.968)	3.000 (0.000)	3.000 (0.000)	3.000 (0.000)
	# of Classes (SD)	5.680 (1.041)	4.330 (1.485)	8.530 (2.991)	5.820 (2.338)	2.990 (0.032)	3.000 (0.000)	3.000 (0.000)
	NMI (SD)	**0.811** (0.118)	0.637†(0.226)	0.670†(0.184)	0.107†(0.124)	0.698 (0.015)	0.985(0.026)	0.973 (0.036)
	Micro-F-score (SD)	**0.915** (0.062)	0.709†(0.185)	0.809†(0.133)	0.425†(0.069)	0.830 (0.125)	0.985 (0.026)	0.973 (0.036)
	Macro-F-score (SD)	**0.916** (0.062)	0.659†(0.231)	0.793†(0.159)	0.264 (0.108)	0.810 (0.160)	0.985 (0.026)	0.973 (0.036)
Zoo	# of Clusters (SD)	12.967 (1.894)	8.633 (3.610)	32.133 (2.825)	20.300 (1.057)	7.000 (0.000)	7.000 (0.000)	7.000 (0.000)
	# of Classes (SD)	6.000 (0.799)	1.067 (0.211)	7.333 (1.7666)	11.433 (1.904)	5.700 (1.005)	7.000 (0.000)	7.000 (0.000)
	NMI (SD)	**0.911** (0.064)	0.014†(0.044)	0.867 (0.103)	0.905 (0.072)	0.851 (0.104)	0.922 (0.110)	0.933 (0.073)
	Micro-F-score (SD)	0.805 (0.094)	0.409†(0.044)	0.794 (0.115)	**0.820** (0.121)	0.814 (0.119)	0.922 (0.110)	0.933 (0.073)
	Macro-F-score (SD)	0.573 (0.094)	0.105†(0.026)	0.611†(0.185)	**0.674**†(0.181)	0.653 (0.204)	0.922 (0.110)	0.933 (0.073)
Letter	# of Clusters (SD)	251.833 (7.371)	271.633 (9.276)	373.367 (30.648)	493.500 (11.923)	26.000 (0.000)	26.000 (0.000)	26.000 (0.000)
	# of Classes (SD)	40.633 (2.427)	2.133 (0.890)	19.100 (4.544)	39.400 (6.935)	25.367 (0.801)	26.000 (0.000)	26.000 (0.000)
	NMI (SD)	**0.519** (0.022)	0.060†(0.045)	0.310†(0.049)	0.118†(0.015)	0.463 (0.017)	0.889 (0.051)	0.852 (0.019)
	Micro-F-score (SD)	**0.382** (0.026)	0.052†(0.008)	0.175†(0.038)	0.055†(0.006)	0.350 (0.018)	0.889 (0.051)	0.852 (0.019)
	Macro-F-score (SD)	**0.382** (0.026)	0.017†(0.012)	0.188†(0.045)	0.023†(0.009)	0.300 (0.025)	0.889 (0.051)	0.852 (0.019)
Thyroid	# of Clusters (SD)	308.200 (7.794)	434.460 (12.431)	550.040 (35.302)	196.120 (10.921)	3.000 (0.000)	3.000 (0.000)	3.000 (0.000)
	# of Classes (SD)	25.600 (1.487)	25.160 (2.108)	26.840 (2.589)	26.150 (5.254)	3.000 (0.000)	3.000 (0.000)	3.000 (0.000)
	NMI (SD)	0.078 (0.089)	0.016†(0.043)	**0.116** (0.121)	0.026†(0.047)	0.000 (0.000)	0.946 (0.010)	0.951 (0.005)
	Micro-F-score (SD)	0.928 (0.003)	0.926 (0.001)	**0.929** (0.004)	0.926†(0.001)	0.926 (0.001)	0.951 (0.010)	0.946 (0.028)
	Macro-F-score (SD)	0.366 (0.003)	0.328†(0.021)	**0.397**†(0.083)	0.333†(0.025)	0.320 (0.000)	0.946 (0.010)	0.951 (0.005)
Ionosphere	# of Clusters (SD)	22.185 (2.148)	48.585 (4.783)	45.075 (3.917)	47.495 (2.099)	2.000 (0.000)	2.000 (0.000)	2.000 (0.000)
	# of Classes (SD)	6.045 (0.953)	2.565 (0.919)	5.755 (2.017)	18.765 (3.001)	2.000 (0.000)	2.000 (0.000)	2.000 (0.000)
	NMI (SD)	**0.378** (0.163)	0.151†(0.101)	0.073†(0.145)	0.179†(0.100)	0.113 (0.123)	0.957 (0.034)	0.902 (0.047)
	Micro-F-score (SD)	**0.832** (0.045)	0.699†(0.046)	0.678†(0.073)	0.715†(0.048)	0.702 (0.069)	0.957 (0.034)	0.902 (0.047)
	Macro-F-score (SD)	**0.795** (0.062)	0.535†(0.103)	0.472†(0.154)	0.578†(0.099)	0.598 (0.172)	0.957 (0.034)	0.902 (0.047)
Sonar	# of Clusters (SD)	12.500 (2.076)	13.075 (1.901)	38.215 (3.948)	29.120 (1.098)	2.000 (0.000)	2.000 (0.000)	2.000 (0.000)
	# of Classes (SD)	5.430 (1.244)	5.635 (0.815)	9.585 (2.856)	16.275 (3.664)	2.000 (0.000)	2.000 (0.000)	2.000 (0.000)
	NMI (SD)	0.108 (0.109)	**0.142**†(0.113)	0.121†(0.114)	0.074†(0.088)	0.074 (0.092)	0.847 (0.067)	0.765 (0.090)
	Micro-F-score (SD)	0.622 (0.100)	0.639(0.081)	**0.640**†(0.097)	0.568†(0.109)	0.615 (0.078)	0.847 (0.067)	0.765 (0.090)
	Macro-F-score (SD)	0.569 (0.100)	0.573†(0.114)	**0.607** (0.119)	0.531†(0.127)	0.595 (0.096)	0.847 (0.067)	0.765 (0.090)
Optdigits	# of Clusters (SD)	271.280 (9.188)	844.920 (18.958)	82.460 (7.967)	495.220 (9.824)	10.000 (0.000)	10.000 (0.000)	10.000 (0.000)
	# of Classes (SD)	12.420 (1.515)	2.940 (1.381)	11.640 (2.292)	51.000 (7.182)	9.980 (0.063)	10.000 (0.000)	10.000 (0.000)
	NMI (SD)	**0.837** (0.024)	0.218†(0.161)	0.772†(0.055)	0.086†(0.038)	0.652 (0.036)	0.987 (0.006)	0.923 (0.012)
	Micro-F-score (SD)	**0.822** (0.045)	0.162†(0.049)	0.767†(0.106)	0.118†(0.015)	0.659 (0.048)	0.987 (0.006)	0.923 (0.012)
	Macro-F-score (SD)	**0.787** (0.045)	0.088†(0.052)	0.739†(0.124)	0.048†(0.021)	0.608 (0.066)	0.987 (0.006)	0.923 (0.012)
Semeion	# of Clusters (SD)	48.767 (4.762)	283.167 (15.790)	12.900 (2.724)	115.967 (7.350)	10.000 (0.000)	10.000 (0.000)	10.000 (0.000)
	# of Classes (SD)	10.733 (1.456)	15.467 (3.058)	5.400 (1.032)	115.867 (7.348)	10.000 (0.000)	10.000 (0.000)	10.000 (0.000)
	NMI (SD)	0.581 (0.036)	**0.602** (0.052)	0.462†(0.063)	0.376 (0.183)	0.480 (0.051)	0.960 (0.017)	0.748 (0.042)
	Micro-F-score (SD)	**0.571** (0.046)	0.552 (0.093)	0.368†(0.066)	0.373†(0.181)	0.507 (0.407)	0.960 (0.017)	0.748 (0.042)
	Macro-F-score (SD)	0.484 (0.046)	**0.507**†(0.107)	0.260†(0.063)	0.337†(0.196)	0.457 (0.060)	0.960 (0.017)	0.748 (0.042)

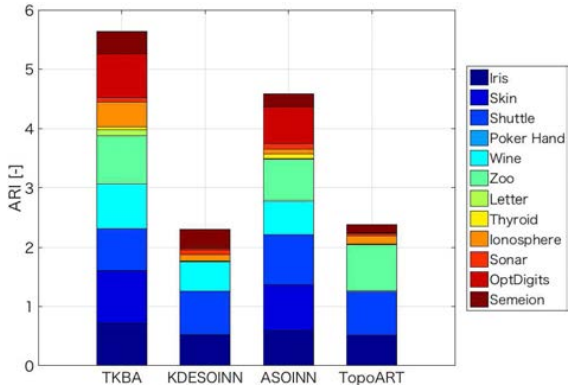


Fig. 16. Comparison of ARI. A larger ARI means that a model is more stable to a problem.

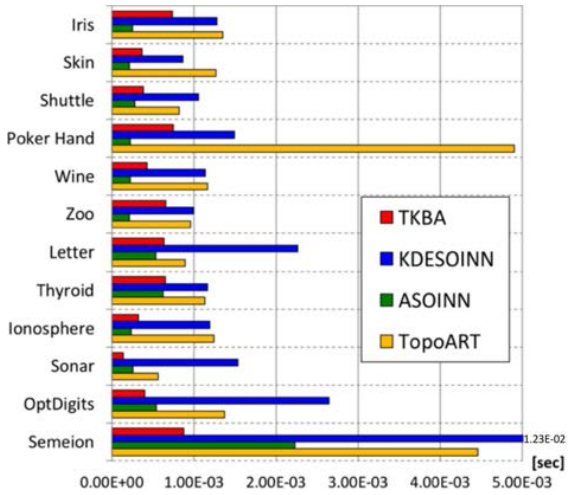


Fig. 17. Comparison of processing times per sample.

5. Conclusions

In this paper, a new unsupervised topological clustering algorithm is introduced by combining the ART-based topological growing network and the kernel framework. TKBA successfully integrates the Bayesian kernel approach, the generalized similarity measurement and the topology construction process in the ART framework.

The results of the self-organizing experiments with the synthetic dataset showed that TKBA is able to perform noise reduction and stable topology construction. In addition, classification experiments with real-world datasets have revealed that TKBA has the capability to deal with several types of data while maintaining superior robustness, adaptability and fast computation. In summary, the typical problems of self-organizing growing network models can be dealt with by TKBA as follows:

— The self-organizing ability of TKBA has a robustness for different order of given data.
— TKBA achieves fast computation in the high-dimensional space due to its kernel framework.
— TKBA acquires high noise reduction capability due to CIM.

Improving the interpretability and selectivity of a huge amount of information makes it possible to further extend a scope of availability for big data of the IoT society.[62] The interpretability and selectivity of information could be enhanced by compressing/expanding the information to an arbitrary granularity. One of the approaches to realize it is to apply a hierarchical architecture to a model. Thus, as future work, a hierarchical architecture will be introduced to TKBA for further improvement of its performance in terms of functionality and ability. From the functional perspective, TKBA is able to compress/expand input information in a more flexible way. The noise reduction and stable self-organizing capabilities are one of the expected ability improvements. As an algorithmic improvement, an adaptive parameter optimization should be considered to reduce the number of parameters that need to be adjusted.

Acknowledgments

This research was supported by Ministry of Education, Culture, Sports, Science and Technology - JAPAN (MEXT) Leading Initiative for Excellent Young Researchers (LEADER). Frontier Research Grant (Project No. FG003-17AFR) from University of Malaya, ONRG grant (Project No: ONRG-NICOP-N62909-18-1-2086) from office of Naval Research Global, UK and the Georg Forster Research Fellowship for Experienced Researchers from Alexander von Humboldt-Stiftung/Foundation also support this research.

References

1. S. Lloyd, Least squares quantization in PCM, *IEEE Trans. Inf. Theor.* **28**(2) (1982) 129–137.
2. A. P. Dempster, N. M. Laird and D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *J. R. Stat. Soc.* (1977) 1–38.

3. T. Kohonen, Self-organized formation of topologically correct feature maps, *Biol. Cybernet.* **43**(1) (1982) 59–69.
4. B. Fritzke, Growing cell structures — a self-organizing network for unsupervised and supervised learning, *Neural Netw.* **7**(9) (1994) 1441–1460.
5. B. Fritzke, A growing neural gas network learns topologies, *Adv. Neural Inf. Process. Syst.* **7** (1995) 625–632.
6. S. Furao and O. Hasegawa, An incremental network for on-line unsupervised classification and topology learning, *Neural Netw.* **19**(1) (2006) 90–106.
7. G. A. Carpenter and S. Grossberg, The art of adaptive pattern recognition by a self-organizing neural network, *Computer* **21**(3) (1988) 77–88.
8. S. Grossberg, Competitive learning: From interactive activation to adaptive resonance, *Cognitive Sci.* **11**(1) (1987) 23–63.
9. G. A. Carpenter, S. Grossberg and D. B. Rosen, Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system, *Neural Netw.* **4**(6) (1991) 759–771.
10. B. Vigdor and B. Lerner, The Bayesian ARTMAP, *IEEE Trans. Neural Netw.* **18**(6) (2007) 1628–1644.
11. N. Masuyama, C. K. Loo and F. Dawood, Kernel Bayesian ART and ARTMAP, *Neural Netw.* **98** (2018) 76–86.
12. M. Tscherepanow, Topoart: A topology learning hierarchical art network, in *Int. Conf. Artificial Neural Networks*, (Springer, 2010), pp. 157–167.
13. S. Marriott and R. F. Harrison, A modified fuzzy artmap architecture for the approximation of noisy mappings, *Neural Netw.* **8**(4) (1995) 619–641.
14. S. Anatolyev, R. Khabibullin and A. Prokhorov, An algorithm for constructing high dimensional distributions from distributions of lower dimension, *Econ. Lett.* **123**(3) (2014) 257–261.
15. K. Fukumizu, L. Song and A. Gretton, Kernel Bayes' rule: Bayesian inference with positive definite kernels, *J. Mach. Learn. Res.* **14**(1) (2013) 3753–3783.
16. W. Liu, P. P. Pokharel and J. C. Príncipe, Correntropy: Properties and applications in non-Gaussian signal processing, *IEEE Trans. Signal Process.* **55**(11) (2007) 5286–5298.
17. R. Chalasani and J. C. Principe, Self-organizing maps with information theoretic learning, *Neurocomput.* **147** (2015) 3–14.
18. M. Ahmadlou and H. Adeli, Enhanced probabilistic neural network with local decision circles: A robust classifier, *Integr. Comput.-Aided Eng.* **17**(3) (2010) 197–210.
19. M. H. Rafiei and H. Adeli, A new neural dynamic classification algorithm, *IEEE Trans. Neural Netw. Learn. Sys.* **28**(12) (2017) 3074–3083.
20. M. H. Rafiei, W. H. Khushefati, R. Demirboga and H. Adeli, Supervised deep restricted Boltzmann machine for estimation of concrete, *ACI Mater. J.* **114**(2) (2017) 237–244.
21. V. Vapnik, *The Nature of Statistical Learning Theory* (Springer Science, Business Media, 2013).
22. G.-B. Huang, Q.-Y. Zhu and C.-K. Siew, Extreme learning machine: Theory and applications, *Neurocomput.* **70**(1–3) (2006) 489–501.
23. W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu and F. E. Alsaadi, A survey of deep neural network architectures and their applications, *Neurocomput.* **234** (2017) 11–26.
24. Y. Zeinali and B. A. Story, Competitive probabilistic neural network, *Integr. Comput.-Aided Eng.* **24**(2) (2017) 105–118.
25. M. Koziarski and B. Cyganek, Image recognition with deep neural networks in presence of noise–dealing with and taking advantage of distortions, *Integr. Comput.-Aided Eng.* **24**(4) (2017) 337–349.
26. V. Nagaraj, A. Lamperski and T. I. Netoff, Seizure control in a computational model using a reinforcement learning stimulation paradigm, *Int. J. Neural Syst.* **27**(7) (2017) 1750012.
27. Y.-Z. Lin, Z.-H. Nie and H.-W. Ma, Structural damage detection with automatic feature-extraction through deep learning, *Comput.-Aided Civil Infrastruct. Eng.* **32**(12) (2017) 1025–1046.
28. X. Li, Y. Bai, Y. Peng, S. Du and S. Ying, Nonlinear semi-supervised metric learning via multiple kernels and local topology, *Int. J. Neural Syst.* **28**(2) (2018) 1750040.
29. E. López-Rubio, M. A. Molina-Cabello, R. M. Luque-Baena and E. Domínguez, Foreground detection by competitive learning for varying input distributions, *Int. J. Neural Syst.* **28**(5) (2018) 1750056.
30. U. R. Acharya, S. L. Oh, Y. Hagiwara, J. H. Tan and H. Adeli, Deep convolutional neural network for the automated detection and diagnosis of seizure using eeg signals, *Comput. Biol. Med.* **100** (2018) 270–278.
31. G. Zhang, H. Rong, F. Neri and M. J. Pérez-Jiménez, An optimization spiking neural p system for approximately solving combinatorial optimization problems, *Int. J. Neural Syst.* **24**(5) (2014) 1440006.
32. L. Pan, G. Păun, G. Zhang and F. Neri, Spiking neural p systems with communication on request, *Int. J. Neural Syst.* **27**(8) (2017) 1750042.
33. M. A. Nabian and H. Meidani, Deep learning for accelerated seismic reliability analysis of transportation networks, *Comput.-Aided Civil Infrastruct. Eng.* **33**(6) (2018) 443–458.
34. H. Hashemi and K. Abdelghany, End-to-end deep learning methodology for real-time traffic network management, *Comput.-Aided Civil Infrastruct. Eng.* **33**(10) (2018) 849–863.
35. A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou and A. Bouras, A survey of clustering algorithms for big data: Taxonomy

and empirical analysis, *IEEE Trans. Emerg. Topics Computing* **2**(3) (2014) 267–279.
36. O. Beyer and P. Cimiano, Online semi-supervised growing neural gas, *Int. J. Neural Syst.* **22**(5) (2012) 1250023.
37. E. López-Rubio, E. J. Palomo and E. Dominguez, Bregman divergences for growing hierarchical self-organizing networks, *Int. J. Neural Syst.* **24**(4) (2014) 1450016.
38. S. Marsland, J. Shapiro and U. Nehmzow, A self-organising network that grows when required, *Neural Netw.* **15**(8) (2002) 1041–1058.
39. N. Masuyama and C. K. Loo, Growing neural gas with correntropy induced metric, in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)* (IEEE, Athens, Greece, 2017), pp. 1–7.
40. S. Furao, T. Ogura and O. Hasegawa, An enhanced self-organizing incremental neural network for online unsupervised learning, *Neural Netw.* **20**(8) (2007) 893–903.
41. F. Shen and O. Hasegawa, A fast nearest neighbor classifier based on self-organizing incremental neural network, *Neural Netw.* **21**(10) (2008) 1537–1547.
42. Y. Nakamura and O. Hasegawa, Nonparametric density estimation based on self-organizing incremental neural network for large noisy data, *IEEE Trans. Neural Netw. Learn. Syst.* **28**(1) (2017) 8–17.
43. P. H. Niknam, B. Mokhtarani and H. Mortaheb, Prediction of shockwave location in supersonic nozzle separation using self-organizing map classification and artificial neural network modeling, *J. Nat. Gas Sci. Eng.* **34** (2016) 917–924.
44. G. A. Carpenter and W. D. Ross, Art-emap: A neural network architecture for object recognition by evidence accumulation, *IEEE Trans. Neural Netw.* **6**(4) (1995) 805–818.
45. S. Fine and K. Scheinberg, Efficient SVM training using low-rank kernel representations, *J. Mach. Learn. Res.* **2**(Dec) (2001) 243–264.
46. S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz and G. Rätsch, Kernel PCA and de-noising in feature spaces, in *Proc. 1998 Conf. Adv. Neural Inf. Process. Syst.* (MIT press, Cambridge 1999), 536–542.
47. T. H. Cormen, *Introduction to Algorithms* (MIT Press, 2009).
48. A. J. Hanson, Geometry for n-dimensional graphics, *Graph. Gems IV* **443** (1994) 149–170.
49. E. López-Rubio, Improving the quality of self-organizing maps by self-intersection avoidance, *IEEE Trans. Neural Netw. Learn. Syst.* **24**(8) (2013) 1253–1265.
50. R. Asadi, H. Sabah Hasan and S. Abdul Kareem, *Review of Current Online Dynamic Unsupervised Feedforward Neural Network Classification*, Proceedings of the International Conference on Advances in Computer Science and Electronics Engineering (CSEE, KL, Malaysia, 2014), pp. 21–28.
51. M. Wand, Fast computation of multivariate kernel estimators, *J. Computat. Graph. Stat.* **3**(4) (1994) 433–445.
52. T. Burwick and F. Joublin, Optimal algorithmic complexity of fuzzy art, *Neural Process. Lett.* **7** (1998) 37–41.
53. D. Arthur and S. Vassilvitskii, How slow is the k-means method? in *Proc. twenty-second Annual Symposium Computational Geometry*, (ACM, NY, USA, 2006) pp. 144–153.
54. A. Abdiansah and R. Wardoyo, Time complexity analysis of support vector machines (SVM) in LibSVM, *Int. J. Comput. Appl.* **128**(3) (2015) 28–34.
55. A. Akusok, K.-M. Björk, Y. Miche and A. Lendasse, High-performance extreme learning machines: A complete toolbox for big data applications, *IEEE Access* **3** (2015) 1011–1025.
56. A. Strehl and J. Ghosh, Cluster ensembles — A knowledge reuse framework for combining multiple partitions, *J. Mach. Learn. Res.* **3**(Dec) (2002) 583–617.
57. M. Sokolova and G. Lapalme, A systematic analysis of performance measures for classification tasks, *Inf. Process. Manag.* **45**(4) (2009) 427–437.
58. F. Wilcoxon, Individual comparisons by ranking methods, *Biometrics Bull.* **1**(6) (1945) 80–83.
59. M. Lichman, UCI machine learning repository [http://archive. ics. uci. edu/ml] University of California, Irvine, School of Information and Computer Sciences (2013).
60. B. E. Boser, I. M. Guyon and V. N. Vapnik, A training algorithm for optimal margin classifiers, in *Proc. Fifth Annual workshop on Computational Learning Theory* (ACM 1992), pp. 144–152.
61. L. Hubert and P. Arabie, Comparing partitions, *J. Classif.* **2**(1) (1985) 193–218.
62. C. P. Chen and C.-Y. Zhang, Data-intensive applications, challenges, techniques and technologies: A survey on big data, *Inf. Sci.* **275** (2014) 314–347.