

Multi-label Classification via Adaptive Resonance Theory-based Clustering

Naoki Masuyama, *Member, IEEE*, Yusuke Nojima, *Member, IEEE*, Chu Kiong Loo, *Senior Member, IEEE*, and Hisao Ishibuchi, *Fellow, IEEE*

Abstract—This paper proposes a multi-label classification algorithm capable of continual learning by applying an Adaptive Resonance Theory (ART)-based clustering algorithm and the Bayesian approach for label probability computation. The ART-based clustering algorithm adaptively and continually generates prototype nodes corresponding to given data, and the generated nodes are used as classifiers. The label probability computation independently counts the number of label appearances for each class and calculates the Bayesian probabilities. Thus, the label probability computation can cope with an increase in the number of labels. Experimental results with synthetic and real-world multi-label datasets show that the proposed algorithm has competitive classification performance to other well-known algorithms while realizing continual learning.

Index Terms—Multi-label Classification, Continual Learning, Clustering, Adaptive Resonance Theory, Correntropy.

1 INTRODUCTION

THANKS to the recent advances of IoT technology, we can easily obtain a wide variety of data and utilize them to machine learning algorithms. Thus, the importance of continual learning is increasing for machine learning algorithms in order to efficiently utilize the data [1]. The requirement for continual learning is to handle both sequential learning and class-incremental learning without destroying the learned knowledge. In general, sequential learning is defined as a method that learns the data instance by instance, not in batches. Class-incremental learning is defined as a method that can deal with the situation where the number of classes (labels) increases during the learning process.

Since real-world phenomena and objects are complex and may have multiple semantics in nature, multi-label classification attracts a great deal of attention from machine learning and related fields such as web mining [2], rule mining [3], and information retrieval [4], [5]. In regard to multi-label classification algorithms, the sequential learning has realized by stream multi-label classification algorithms [6]. For those algorithms, however, data pre-processing such as normalization and standardization is often required. In addition, it is necessary for learning process to define the number of classes in advance. The class-incremental learning is theoretically feasible with the Bayesian approach for label probability computation in Multi-Label k -Nearest

Neighbor (ML- k NN) [7]. The label learning process independently counts the number of label appearances for each class and calculates the Bayesian probabilities. Thus, the label probability computation can cope with an increase in the number of labels. However, ML- k NN cannot cope with the sequential learning because k -NN requires the entire data before the learning process. Multi-label Learning with Emerging New Labels (MuENL) [8] has realized continual learning by constructing two classifiers for classifying instances and for detecting new labels. However, MuENL cannot perform continual learning in the non-stationary environment, i.e., the situation where new data distributions are sequentially provided.

In the case of single-label classification algorithms, several types of clustering-based classifiers capable of continual learning have been proposed [9], [10], [11], [12], [13], [14]. In particular, classifiers designed by an Adaptive Resonance Theory (ART)-based clustering algorithm have shown comparable classification performance to typical classification algorithms such as SVM and k -NN. The main feature of the above ART-based clustering algorithms is the use of the Correntropy-Induced Metric (CIM) [15] to a similarity threshold, which makes the self-organizing process fast and stable [11], [12], [13], [14].

In this paper, in order to realize a multi-label classification algorithm capable of continual learning, we propose Multi-Label CIM-based ART (MLCA) by integrating the Bayesian approach for label probability computation into the ART-based clustering with the CIM. Furthermore, we also propose two variants of MLCA by modifying the calculation method of the CIM to improve the classification performance of MLCA.

The contributions of this paper are summarized as follows:

- (i) A multi-label classification algorithm, called MLCA, is proposed by integrating a CIM-based ART and the Bayesian approach for label probability computation. MLCA computes the prior probability and likelihood

• N. Masuyama, and Y. Nojima are with the Graduate School of Engineering, Osaka Prefecture University, 1-1 Gakuen-cho Naka-ku, Sakai-Shi, Osaka 599-8531, Japan.

E-mails: {masuyama, nojima}@cs.osakafu-u.ac.jp

• C. K. Loo is with the Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia.

E-mail: ckloo.um@um.edu.my

• H. Ishibuchi is with the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China.

E-mail: hisao@sustech.edu.cn

Corresponding author: Hisao Ishibuchi (e-mail: hisao@sustech.edu.cn).

Manuscript received April 19, 2005; revised August 26, 2015.

by using nodes (representative points of training data) which have accumulated label counts while ML- k NN computes them by directly referencing to training instances.

- (ii) A new CIM-based ART is proposed by introducing an efficient node generation process and a bandwidth adaptation method for a kernel function in CIM. As a result, MLCA does not require any data pre-processing method such as normalization or scaling.
- (iii) Empirical studies show that MLCA and its variants have competitive classification performance to recent multi-label classification algorithms.
- (iv) The continual learning ability of MLCA is analyzed from multiple perspectives, and its usefulness and superiority are clarified.

The paper is organized as follows. Section 2 presents literature review for clustering algorithms and multi-label classification algorithms. Section 3 describes details of the proposed algorithm and modifications of the calculation method of the CIM. Section 4 presents extensive simulation experiments to evaluate the continual learning ability and classification performance of MLCA. Section 5 concludes this paper.

2 LITERATURE REVIEW

2.1 Clustering Algorithm

Cluster analysis is one of the widely applied approaches to extract hidden relation from data. Typical types of clustering algorithms are the Gaussian mixture model [16], k -means [17], and Self-Organizing Map (SOM) [18]. Although the above algorithms are quite simple and highly adaptable, the number of classes and network architectures are specified in advance. Growing Neural Gas (GNG) [19] and Self-Organizing Incremental Neural Network (SOINN) [9] are well-known growing self-organizing clustering algorithms that can overcome the drawbacks of the typical types of clustering algorithms. GNG and SOINN can adaptively generate topological networks corresponding to given data. However, since these algorithms permanently insert new nodes into their networks for memorizing new knowledge, they have a potential to forget learned knowledge (i.e., catastrophic forgetting). This trade-off is called the plasticity-stability dilemma [20]. A variant of GNG, called Grow When Required (GWR) [21] can avoid the plasticity-stability dilemma by adding nodes whenever the state of the current network does not sufficiently match the instance. One problem of GWR is that as the number of nodes in the network increases, the cost of calculating a threshold for each node increases, and thus the learning efficiency decreases.

A successful approach to avoid the plasticity-stability dilemma is the ART-based algorithms [22]. Because the ART-based algorithms realize sequential and class-incremental learning without the catastrophic forgetting, a number of the ART-based algorithms and their improvements are proposed in both supervised learning [23], [24], [25] and unsupervised learning [26], [27], [28], [29]. In the ART-based algorithms, a criterion of a new category (node) generation,

i.e., a similarity measurement between a node and an instance, has a great impact on the classification/clustering performance. Previous studies have shown that algorithms with the CIM [15] as a similarity measurement are capable of faster and more stable learning than other self-organizing clustering algorithms [11], [12], [13], [14].

2.2 Multi-label Classification

The multi-label classification algorithms are categorized into two approaches, namely, a problem transformation approach and an algorithm adaptation approach [30]. The problem transformation approach transforms a multi-label classification problem into multiple single-label classification problems. The problem transformation approach is further divided into two methods, namely, the Binary Relevance (BR) [31] and the Label Powerset (LP) [32]. The BR transforms a multi-label classification problem into multiple binary classification problems by decomposing multi-labels into multiple single labels. The LP transforms a multi-label classification problem into a multi-class classification problem by merging multi-labels into a single label. Various single-label classification algorithms have been used in the problem transformation approach thanks to its simplicity and applicability.

The algorithm adaptation approach extends existing single-label classification algorithms for handling multi-label classification problems. Various types of algorithm adaptation approach have been introduced based on k -NN [7], [33], decision tree [34], regression [35], Support Vector Machine (SVM) [36], and feed-forward neural networks [37], [38]. In order to achieve high classification performance, recent studies consider label distributions and their correlations in a label learning process [39], [40]. These algorithms, however, cannot cope with the situation where new label information is sequentially provided. ML- k NN [7] is a well-known algorithm adaptation method that integrates k -NN and the Bayesian approach for label probability computation. ML- k NN counts the number of relevant labels of neighbors for each instance in training data. Based on the counts of relevant labels, the likelihood and posterior probability of each label are computed by the Bayesian approach. Here, the computation of the Bayesian probability is individually performed in each label. Therefore, in theory, the number of labels to be learned can be increased/decreased during the label probability computation. One disadvantage of ML- k NN is that it cannot efficiently cope with the situation where new training instances are sequentially provided. Multi-Label Self-Adjusting k Nearest Neighbors (MLSA- k NN) [41] is capable of handling a stream multi-label classification by employing a self-adjusting window to detect a concept drift and to adaptively control a parameter k in k NN. MuENL [8] generates two classifiers for classifying instances and for detecting new labels. By using the two classifiers, MuENL realizes sequential learning and class-incremental learning, simultaneously. Although MuENL is capable of continual learning, MuENL requires a batch learning process in the initial stage of learning for constructing two classifiers for instances and labels. Moreover, MuENL cannot deal with the situation where new data distributions are provided in the non-stationary environment.

Several studies have realized multi-label classification by using a clustering algorithm. A typical type of clustering-based multi-label classification algorithm utilizes SOM [42], [43]. Although the learning process of SOM is performed in an unsupervised learning manner, the convergence of the SOM network is significantly slow and unstable. The online semi-supervised GNG for multi-label classification utilizes GNG as a base classifier [44]. Unlike SOM-based algorithms, this method adaptively generates a topological network corresponding to the given instances. However, as mentioned in the literature review for clustering in the previous subsection, GNG-based algorithms do not satisfy the requirements of continual learning. Multi-label Classification via Incremental Clustering (MCIC) [45] has realized continual learning by applying a continual clustering to a learning process. MCIC extracts and accumulates information from data by nodes through the continual clustering process that takes into account the arrival time of data. In addition, the continual clustering process also constructs and updates a distribution of labels in each node for a label estimation process. Some studies have employed Fuzzy ARTMAP [46] to a base classifier of the multi-label classification [47], [48], [49], [50]. ARTMAP is composed of two ART architecture to realize an explicit supervised learning process. Although Fuzzy ARTMAP has various advantages, there is a well-known problem, i.e., high sensitivity to statistical overlapping between the generated categories [51]. This sensitivity problem results in category proliferation (i.e., disordered generation of categories), which leads to a high

computational cost and deterioration in the classification performance. The recent ART-based algorithms in [47], [48], [49], [50] potentially have this problem.

3 PROPOSED ALGORITHM

In this section, first the theoretical background of the CIM is briefly described. Next, the proposed algorithm, namely, MLCA, is explained in detail. Then, two variants of MLCA are introduced by modifying the calculation method of the CIM. Table 1 summarizes the main notations used in this paper.

3.1 Correntropy and Correntropy-Induced Metric

Correntropy [15] provides a generalized similarity measure between two arbitrary instances $\mathbf{x} = (x_1, x_2, \dots, x_d)$ and $\mathbf{y} = (y_1, y_2, \dots, y_d)$ as follows:

$$C(\mathbf{x}, \mathbf{y}) = \mathbf{E}[\kappa_\sigma(\mathbf{x} - \mathbf{y})], \quad (1)$$

where $\mathbf{E}[\cdot]$ is the expectation operation, and $\kappa_\sigma(\cdot)$ denotes a positive definite kernel with a kernel bandwidth σ . The correntropy can be defined as follows:

$$\hat{C}(\mathbf{x}, \mathbf{y}) = \frac{1}{d} \sum_{i=1}^d \kappa_\sigma(x_i - y_i). \quad (2)$$

In this paper, we use the following Gaussian kernel in the correntropy:

$$\kappa_\sigma(x_i - y_i) = \exp\left[-\frac{(x_i - y_i)^2}{2\sigma^2}\right]. \quad (3)$$

TABLE 1: Summary of notations

Notation	Description
$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \dots, \mathbf{x}_\infty)$	A set of training instances
$\mathbf{x}_n = (x_{n,1}, x_{n,2}, \dots, x_{n,d})$	d -dimensional training instance (the n th instance)
$\mathbf{L} = (\mathbf{l}_1, \mathbf{l}_2, \dots, \mathbf{l}_n, \dots, \mathbf{l}_\infty)$	A set of relevant label sets for \mathbf{X}
$\mathbf{l}_n = (l_{n,1}, l_{n,2}, \dots, l_{n,N_l})$	A set of relevant label for \mathbf{x}_n
N_l	Dimension of the relevant label set for \mathbf{X}
$\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_K)$	A set of prototype nodes (the k th node)
$\mathbf{y}_k = (y_1, y_2, \dots, y_d)$	d -dimensional prototype node
$\mathbf{S} = (\sigma_1, \sigma_2, \dots, \sigma_K)$	A set of bandwidths for a kernel function
κ_σ	Kernel function with a bandwidth σ
CIM	Correntropy-Induced Metric
k_1, k_2	Indexes of the 1st and 2nd winner nodes
y_{k_1}, y_{k_2}	The 1st and 2nd winner nodes
V_{k_1}, V_{k_2}	Similarities between an instance x_n and winner nodes (y_{k_1} and y_{k_2})
V	Predefined similarity threshold
α_{k_1}	The number of instances that have accumulated by the node y_{k_1}
β_{k_1}	The number of labels that have accumulated by the node y_{k_1}
$\mathbf{y}_{k_1}^+$	Neighbor node of y_{k_1}
N_y	The predefined number of neighbor nodes for y_{k_1}
λ	Interval for adapting σ
H_i^+	Event that an instance has the i th label
H_i^-	Event that an instance does not have the i th label
E_i	Event that the label l_i has the highest frequency among the N_y neighbor nodes of y_{k_1}
$P(E H)$	Likelihood for a label probability computation
$P(H)$	Prior probability for a label probability computation
$P(H E)$	Posterior probability for a label probability computation
c	Label counter
$\mathbf{x}^* = (x_1^*, x_2^*, \dots, x_d^*)$	d -dimensional testing instance
\mathbf{l}^*	Predicted label vector of the test instance \mathbf{x}^*

A nonlinear metric called CIM is derived from the corentropy [15]. The CIM quantifies the similarity between two instances as follows:

$$\text{CIM}(\mathbf{x}, \mathbf{y}, \sigma) = \left[\kappa_\sigma(0) - \hat{C}(\mathbf{x}, \mathbf{y}) \right]^{\frac{1}{2}}, \quad (4)$$

where $\kappa_\sigma(0) = 1$ from (3). Here, thanks to the Gaussian kernel without a coefficient $\frac{1}{\sqrt{2\pi}\sigma}$ as defined in (3), a range of the CIM is limited to $[0, 1]$.

3.2 Learning Procedure

We use the following notations: Training instances are $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \dots, \mathbf{x}_\infty)$ where $\mathbf{x}_n = (x_{n,1}, x_{n,2}, \dots, x_{n,d})$ is a d -dimensional feature vector. Relevant label sets for \mathbf{X} are $\mathbf{L} = (\mathbf{l}_1, \mathbf{l}_2, \dots, \mathbf{l}_n, \dots, \mathbf{l}_\infty)$ where $\mathbf{l}_n = (l_{n,1}, l_{n,2}, \dots, l_{n,\infty})$ is a binary vector used to show a set of relevant labels for \mathbf{x}_n . The dimension of \mathbf{l}_n increases with the number of label types (classes) in the learned training instances. Note that MLCA is capable of continual learning, the algorithm can accept any number of training instances and labels. A set of prototype nodes in MLCA at the point of the presentation of instance \mathbf{x}_n is $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_K)$ ($K \in \mathbb{Z}^+$) where a node $\mathbf{y}_k = (y_1, y_2, \dots, y_d)$ has the same dimension as \mathbf{x}_n . Furthermore, each node \mathbf{y}_k has an individual bandwidth σ for the CIM, i.e., $\mathbf{S} = (\sigma_1, \sigma_2, \dots, \sigma_K)$.

The learning procedure of MLCA is divided into five parts: 1) initialization process for a bandwidth of a kernel function in the CIM, 2) winner node selection, 3) vigilance test, 4) node learning, and 5) label probability computation. Each of them is explained in the following subsections.

3.2.1 Initialization Process for a Bandwidth of a Kernel Function in the CIM

Similarity measurement between an instance and a node has a large impact on the performance of clustering algorithms. MLCA uses the CIM as a similarity measure. As defined in (4), the state of the CIM is controlled by a bandwidth σ of a kernel function which is a data-dependent parameter.

In general, the bandwidth of a kernel function can be estimated from λ instances belonging to a certain distribution [52], which is defined as follows:

$$\Sigma = U(F_\nu) \Gamma \lambda^{-\frac{1}{2\nu+d}}, \quad (5)$$

$$U(F_\nu) = \left(\frac{\pi^{d/2} 2^{d+\nu-1} (\nu!)^2 R(F)^d}{\nu \kappa_\nu^2(F) [(2\nu)!! + (d-1)(\nu!)^2]} \right)^{\frac{1}{2\nu+d}}, \quad (6)$$

where Γ denotes a rescale operator (d -dimensional vector) which is defined by a standard deviation of the d attributes among λ instances. ν is the order of a kernel. $R(F)$ is a roughness function. $\kappa_\nu(F)$ is the moment of a kernel. In this paper, we utilize the Gaussian kernel for the CIM. Therefore, $\nu = 2$, $R(F) = (2\sqrt{\pi})^{-1}$, and $\kappa_\nu^2(F) = 1$ are derived. The details of the derivation of (5) and (6) can be found in [52].

In MLCA, the initial state of σ in the CIM is defined by training instances. When a new node \mathbf{y}_{K+1} is generated from \mathbf{x}_n , a bandwidth σ_{K+1} is estimated from the past λ instances, i.e., $(\mathbf{x}_{n-\lambda}, \dots, \mathbf{x}_{n-2}, \mathbf{x}_{n-1})$, by using (5) and (6) with $\nu = 2$, $R(F) = (2\sqrt{\pi})^{-1}$, and $\kappa_\nu^2(F) = 1$, as follows:

$$\Sigma = \left(\frac{4}{2+d} \right)^{\frac{1}{4+d}} \Gamma \lambda^{-\frac{1}{4+d}}, \quad (7)$$

where Γ denotes a rescale operator (d -dimensional vector) which is defined by a standard deviation of the d attributes among the past λ training instances of MLCA. Here, Σ contains the bandwidth of each attribute.

In this paper, the median of Σ is selected as a representative bandwidth for the new node \mathbf{y}_{K+1} , i.e.,

$$\sigma_{K+1} = \text{median}(\Sigma). \quad (8)$$

3.2.2 Winner Node Selection

Once an instance \mathbf{x}_n is presented to MLCA, two nodes which have a similar state to the instance \mathbf{x}_n are selected, namely, winner nodes \mathbf{y}_{k_1} and \mathbf{y}_{k_2} . The winner nodes are determined based on the state of the CIM as follows:

$$k_1 = \arg \min_{k \in K} [\text{CIM}(\mathbf{x}_n, \mathbf{y}_k, \text{mean}(\mathbf{S}))], \quad (9)$$

$$k_2 = \arg \min_{k \in K \setminus \{k_1\}} [\text{CIM}(\mathbf{x}_n, \mathbf{y}_k, \text{mean}(\mathbf{S}))], \quad (10)$$

where k_1 and k_2 denote indexes of the 1st and 2nd winner nodes i.e., \mathbf{y}_{k_1} and \mathbf{y}_{k_2} , respectively. \mathbf{S} is a bandwidth for a kernel function of the CIM in each node.

Note that when there is no node in MLCA, the $(\lambda+1)$ th instance becomes the initial node (i.e., $\mathbf{y}_1 = \mathbf{x}_{\lambda+1}$). In the case, the bandwidth of \mathbf{y}_1 is estimated from the 1st to λ th instances in a set of training instances \mathbf{X} by (5)-(8), and the next instance is given without vigilance test until the 1st and 2nd winner nodes can be defined.

3.2.3 Vigilance Test

Similarities between an instance \mathbf{x}_n and the 1st and 2nd winner nodes are defined as follows:

$$V_{k_1} = \text{CIM}(\mathbf{x}_n, \mathbf{y}_{k_1}, \text{mean}(\mathbf{S})), \quad (11)$$

$$V_{k_2} = \text{CIM}(\mathbf{x}_n, \mathbf{y}_{k_2}, \text{mean}(\mathbf{S})). \quad (12)$$

The vigilance test classifies the relationship between an instance and a node into three cases by using a predefined similarity threshold V .

- Case I

The similarity between an instance \mathbf{x}_n and the 1st winner node \mathbf{y}_{k_1} is larger (i.e., less similar) than the similarity threshold V , namely:

$$V_{k_1} > V. \quad (13)$$

If (13) is satisfied, $V_{k_2} > V$ is also satisfied since $V_{k_2} > V_{k_1} > V$. Thus, a new node is defined as $\mathbf{y}_{K+1} = \mathbf{x}_n$, and the bandwidth σ_{K+1} is defined by (8).

Moreover, the following two counters α and β are updated. One counter α is the number of instances that have been accumulated by the node \mathbf{y}_{k_1} , which is updated as follows:

$$\alpha_{k_1} \leftarrow \alpha_{k_1} + 1. \quad (14)$$

The other counter β is the number of labels that have been accumulated by the node \mathbf{y}_{k_1} , which is updated as follows:

$$\beta_{k_1} \leftarrow \beta_{k_1} + \mathbf{l}_{k_1}. \quad (15)$$

- Case II

A similarity between an instance \mathbf{x}_n and the 1st winner

node \mathbf{y}_{k_1} is smaller (i.e., more similar) than the similarity threshold V , and a similarity between the instance \mathbf{x}_n and the 2nd winner node \mathbf{y}_{k_2} is larger (i.e., less similar) than the similarity threshold V , namely:

$$V_{k_1} \leq V, \text{ and } V_{k_2} > V. \quad (16)$$

If (16) is satisfied, node learning is performed. In addition, counters α_{k_1} and β_{k_1} are updated by (14) and (15), respectively.

- Case III

Similarities between an instance \mathbf{x}_i and the 1st and 2nd winner nodes are both smaller (i.e., more similar) than the similarity threshold V , namely:

$$V_{k_1} \leq V, \text{ and } V_{k_2} \leq V. \quad (17)$$

If (17) is satisfied, node learning is performed.

3.2.4 Node Learning

Different node learning is performed based on the results of the vigilance test.

If Case II, the state of the 1st winner node \mathbf{y}_{k_1} is updated as follows:

$$\mathbf{y}_{k_1} \leftarrow \mathbf{y}_{k_1} + \frac{1}{\alpha_{k_1}} (\mathbf{x}_n - \mathbf{y}_{k_1}). \quad (18)$$

When updating the node, the amount of change is divided by α_{k_1} , so the larger α_{k_1} becomes, the smaller the node position changes. This is based on the idea that the information around a node, where instances are frequently given, is important and should be held by the node.

If Case III, the state of the 1st winner node \mathbf{y}_{k_1} is updated by (18). In addition, all neighbor nodes of \mathbf{y}_{k_1} (i.e., $\mathbf{y}_{k_1}^+$) are also updated as follows:

$$\mathbf{y}_{k_1}^+ \leftarrow \mathbf{y}_{k_1}^+ + \frac{1}{N_y \alpha_{k_1}^+} (\mathbf{y}_{k_1} - \mathbf{y}_{k_1}^+), \quad (19)$$

where N_y is the predefined number of neighbor nodes for \mathbf{y}_{k_1} . $\alpha_{k_1}^+$ denotes the number of instances that have been accumulated by the node $\mathbf{y}_{k_1}^+$.

Equation (19) has the same concept as (18), but it should be less affected by the instance than \mathbf{y}_{k_1} because it is the neighbor node of \mathbf{y}_{k_1} . Thus, N_y is added as a coefficient.

3.2.5 Label Probability Computation

Similar to ML- k NN, MLCA employs the Bayesian approach for label probability computation. The prior probability and likelihood are updated if the condition for Case I or Case II of the vigilance test is satisfied. Note that ML- k NN computes the prior probability and likelihood by using training instances repeatedly, while MLCA computes the prior probability and likelihood by using nodes which have accumulated label counts. As a result, MLCA realizes continual learning by computing the prior probability and likelihood whenever an instance is given.

To update the prior probability and likelihood, an instance \mathbf{x}_n with a set of labels l_n , the 1st winner node \mathbf{y}_{k_1} and its N_y neighbor nodes are considered. Note that N_y denotes the number of neighbor nodes and is a predefined parameter in MLCA. Here, we consider the situation where

$(n - 1)$ instances have been given to MLCA. The likelihood $P(E|H)$ is computed as follows:

$$P(E_{n,i}|H_{n,i}^\phi) = \frac{(s + \mathbf{c}_i^\phi)}{\left[s \times (N_y + 1) + \sum_{j=0}^{N_y} \mathbf{c}_{i,j}^\phi \right]}, \quad (i \in N_l, \phi \in \{+, -\}), \quad (20)$$

where H_i^+ is the event that an instance has the i th label (i.e., $l_i = 1$), and H_i^- is the event that an instance does not have the i th label (i.e., $l_i = 0$). E_i is the event that the frequency of the label l_i among the N_y neighbor nodes of \mathbf{y}_{k_1} . N_l is a size of a label set. s is a smoothing parameter. In this paper, s is set to be 1 which yields the Laplace smoothing. A label counter c is defined as follows:

$$\begin{cases} \mathbf{c}_{i,j=g_i}^+ \leftarrow \mathbf{c}_{i,j=g_i}^+ + 1, & \text{if } l_{n,i} = 1 \\ \mathbf{c}_{i,j=g_i}^- \leftarrow \mathbf{c}_{i,j=g_i}^- + 1, & \text{otherwise} \end{cases}. \quad (21)$$

Here, g_i is a i th attribute of an N_l -dimensional counting vector $\mathbf{g} = (g_1, \dots, g_{N_l})$, which is defined as follows:

$$g_i = \sum_{j \in N_y} \beta_{i,j}, \quad (i \in N_l). \quad (22)$$

In order to make the maximum value of g_i the same as the number of neighbor nodes N_y , the following operation is performed.

$$g_i \leftarrow \text{round} \left[N_y \cdot \frac{g_i}{\max(\mathbf{g})} \right]. \quad (23)$$

The prior probability $P(H)$ is computed as follows:

$$P(H_i^+) = \frac{\left(s + \sum_{k=1}^K \beta_{k,i} \right)}{(s \times 2 + n)}, \quad (i \in N_l), \quad (24)$$

$$P(H_i^-) = 1 - P(H_i^+), \quad (i \in N_l), \quad (25)$$

where n denotes the number of instances that have been given.

The learning procedure of MLCA is summarized in Algorithm 1.

3.3 Label Prediction Procedure

We use the following notations: A testing instance is $\mathbf{x}^* = (x_1^*, x_2^*, \dots, x_d^*)$. A set of prototype nodes in MLCA after the learning procedure is $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_K\}$. In addition, each node \mathbf{y}_k has an individual bandwidth σ_k for the CIM, i.e., $S = (\sigma_1, \sigma_2, \dots, \sigma_K)$. Moreover, the likelihood $P(E|H)$, the prior probability $P(H)$, and the label counter β are utilized for the Bayesian approach.

First of all, the winner node \mathbf{y}_{k_1} for a test instance \mathbf{x}^* and its N_y neighbor nodes are determined. In the same manner as the learning procedure, a membership counting vector \mathbf{g} for the test instance \mathbf{x}^* is computed by (23). The posterior probability $P(H|E)$ for the test instance \mathbf{x}^* is defined by the Bayes rule as follows:

$$P(H_i^+ | E_{g_i}) = \frac{P(E_{g_i} | H_i^+) P(H_i^+)}{\sum_{\phi \in \{+, -\}} P(E_{g_i} | H_i^\phi) P(H_i^\phi)}, \quad (i \in N_l). \quad (26)$$

Algorithm 1: Learning procedure of MLCA**Input:**

- a training instance: $\mathbf{x}_n = (x_{n,1}, x_{n,2}, \dots, x_{n,d})$,
- a label set: $\mathbf{l}_n = (l_{n,1}, l_{n,2}, \dots)$,
- prototype nodes: $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_K)$ ($K \in \mathbb{Z}^+$),
- bandwidths of \mathbf{Y} : $\mathbf{S} = (\sigma_1, \sigma_2, \dots, \sigma_K)$,
- the number of instances that have been accumulated by nodes \mathbf{Y} : $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_K)$,
- the number of labels that have been accumulated by nodes \mathbf{Y} : $\boldsymbol{\beta} = (\beta_1, \beta_2, \dots, \beta_K)$,
- a label counter: c ,
- the number of neighbor nodes: N_y ,
- an interval for adapting σ : λ ,
- and a similarity threshold: V .

Output:

- updated prototype nodes: \mathbf{Y} ,
- updated bandwidths of \mathbf{Y} : \mathbf{S} ,
- the updated number of instances that have been accumulated by the nodes \mathbf{Y} : $\boldsymbol{\alpha}$,
- the updated number of labels that have been accumulated by the nodes \mathbf{Y} : $\boldsymbol{\beta}$,
- the updated label counter: c ,
- a prior probability: $P(H)$,
- and a likelihood: $P(E|H)$.

```

1 function LearningMLCA( $\mathbf{x}_n, \mathbf{l}_n, \mathbf{Y}, \mathbf{S}, \boldsymbol{\alpha}, \boldsymbol{\beta}, c, N_y, \lambda, V$ )
2   Input an instance  $\mathbf{x}_n$ .
3   Input a label set  $\mathbf{l}_n$ .
4   if The index  $n$  is multiple of  $\lambda$  then
5     | Compute a bandwidth for the CIM by (8).
6   if  $K < 1$  then
7     | Generate a new node as  $\mathbf{y}_{K+1} = \mathbf{x}_n$ .
8     | Assign a bandwidth  $\sigma_{K+1}$ .
9     | Update  $\alpha_{k+1}$  and  $\beta_{k+1}$  by (14) and (15), respectively.
10  else
11    | Compute the CIM by (4).
12    | Search indexes of winner nodes  $k_1$  and  $k_2$  by (9) and (10), respectively.
13    | if  $\text{CIM}(\mathbf{x}_n, \mathbf{y}_{k_1}, \text{mean}(\mathbf{S})) > V$  then
14      | Generate a new node as  $\mathbf{y}_{K+1} = \mathbf{x}_l$ .
15      | Compute a bandwidth  $\sigma_{K+1}$  which is defined by (8).
16      | Update  $\alpha_{k+1}$  and  $\beta_{k+1}$  by (14) and (15), respectively.
17      | Update similarities between an instance  $\mathbf{x}_n$  and nodes  $\mathbf{Y}$  by the CIM.
18      | Update a likelihood  $P(E|H)$  by (20).
19    | else
20      | Update a state of  $\mathbf{y}_{k_1}$  by (18).
21      | Update  $\alpha_{k_1}$  and  $\beta_{k_1}$  by (14) and (15), respectively.
22      | if  $\text{CIM}(\mathbf{x}_n, \mathbf{y}_{k_2}, \text{mean}(\mathbf{S})) \leq V$  then
23        | | Update the state of  $N_y$  neighbor nodes of  $\mathbf{y}_{k_1}$  by (19).
24        | | Update a likelihood  $P(E|H)$  by (20).
25    | Compute a prior probability by (24) and (25).
26  return  $\mathbf{Y}, \mathbf{S}, \boldsymbol{\alpha}, \boldsymbol{\beta}, c, P(E|H)$ , and  $P(H)$ .

```

A predicted label vector \mathbf{l}^* of the test instance \mathbf{x}^* is determined by a simple thresholding method as follows:

$$l_i^* = \begin{cases} 1, & \text{if } P(H_i^+ | E_{g_i}) > 0.5 \\ 0, & \text{otherwise} \end{cases}, \quad (i \in N_l). \quad (27)$$

Since the label prediction process is a binary classification, it is reasonable to specify the threshold to 0.5. ML- k NN also uses the same value.

The prediction procedure of MLCA is summarized in Algorithm 2.

3.4 Attribute Processing for the CIM

As shown in (2) and (3), the CIM in MLCA uses a common bandwidth σ even if the range of attribute values is different, and the average value as the similarity between an instance \mathbf{x}_n and a node \mathbf{y}_k . Therefore, if the range of attribute values is significantly different, a specific attribute may have a large impact on the value of the CIM when the common bandwidth σ is not appropriate for that attribute.

In this section, we propose two approaches in order to mitigate the above-mentioned effects: 1) one approach calculates the CIM by using an each individual attribute separately, and the average CIM value is used for similarity measurement, and 2) the other approach applies a clustering algorithm to attribute values, then attributes with similar value ranges are grouped. The CIM is calculated by using an each attribute group, and the average CIM value is used for similarity measurement.

3.4.1 Individual-based Approach

In this approach, the CIM is calculated by using an each individual attribute separately, and the average CIM value is used for similarity measurement. The similarity between an instance \mathbf{x}_n and a node \mathbf{y}_k is defined by the CIM^I as follows:

$$\text{CIM}^I(\mathbf{x}_n, \mathbf{y}_k, \boldsymbol{\sigma}_k) = \frac{1}{d} \sum_{i=1}^d [\kappa_{\sigma_{k,i}}(0) - C_I(x_{n,i}, y_{k,i})]^{\frac{1}{2}}, \quad (28)$$

$$C_I(x_{n,i}, y_{k,i}) = \kappa_{\sigma_{k,i}}(x_{n,i} - y_{k,i}), \quad (29)$$

where $\boldsymbol{\sigma}_k = (\sigma_1, \sigma_2, \dots, \sigma_d)$ is a bandwidth of a node \mathbf{y}_k . A bandwidth for the i th attribute is defined as follows:

$$\sigma_i = \left(\frac{4}{2+d} \right)^{\frac{1}{4+d}} \Gamma_i \lambda^{-\frac{1}{4+d}}, \quad (30)$$

where Γ_i denotes a rescale operator which is defined by a standard deviation of i th attribute values among the λ instances.

In this paper, MLCA with the individual-based approach is called MLCA-Individual (MLCA-I).

3.4.2 Clustering-based Approach

In this approach, for every λ instances, the clustering algorithm presented in Section 3.2 is applied to the attribute values. Each attribute value of λ instances is regarded as a one-dimensional vector and used as the input to the clustering algorithm. As a result, attributes with similar value ranges are grouped together, i.e., an instance $\mathbf{x}_n = (x_1, x_2, \dots, x_d)$ is transformed into $\mathbf{x}_n^C = (\mathbf{u}_{n,1}, \mathbf{u}_{n,2}, \dots, \mathbf{u}_{n,J})$ ($J \leq d$) by

Algorithm 2: Prediction procedure of MLCA

Input:

- a testing instance: $\mathbf{x}^* = (x_1^*, x_2^*, \dots, x_d^*)$,
- prototype nodes: $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_K)$ ($K \in \mathbb{Z}^+$),
- the bandwidths of \mathbf{Y} : $\mathbf{S} = (\sigma_1, \sigma_2, \dots, \sigma_K)$,
- the number of labels that have been accumulated by nodes \mathbf{Y} : $\boldsymbol{\beta} = (\beta_1, \beta_2, \dots, \beta_K)$,
- the number of neighbor nodes: N_y ,
- and a similarity threshold: V .

Output:

- a predicted label vector: \mathbf{l}^* .

```

1 function PredictMLCA( $\mathbf{x}^*, \mathbf{Y}, \mathbf{S}, \boldsymbol{\beta}, N_y, V$ )
2   Input an instance  $\mathbf{x}^*$ .
3   Compute similarities between an instance  $\mathbf{x}^*$  and
4   nodes  $\mathbf{Y}$  by the CIM.
5   Compute a membership counting vector  $\mathbf{g}^*$  by
6   (23).
7   Compute a posterior probability  $P(H|E)$  by (26).
8   Determine a predicted label vector  $\mathbf{l}^*$  by (27).
9   return  $\mathbf{l}^*$ .

```

the clustering algorithm, where \mathbf{u}_j represents the j th attribute group. The dimensionality of each attribute group is represented by $\mathbf{d} = (d_1, d_2, \dots, d_J)$ where d_j is the dimensionality of the j th attribute group.

In this approach, the similarity between an instance \mathbf{x}_n^C and a node \mathbf{y}_k is defined by the CIM^C as follows:

$$\text{CIM}^C(\mathbf{x}_n^C, \mathbf{y}_k^C, \boldsymbol{\sigma}_k^C) = \frac{1}{J} \sum_{j=1}^J [\kappa_{\sigma_j}(0) - \hat{C}_C(\mathbf{u}_j, \mathbf{v}_j)]^{\frac{1}{2}}, \quad (31)$$

$$\hat{C}_C(\mathbf{u}_j, \mathbf{v}_j) = \frac{1}{d_j} \sum_{i=1}^{d_j} \kappa_{\sigma_j}(u_i - v_i), \quad (32)$$

where $\mathbf{y}_k^C = (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_J)$ is a node \mathbf{y}_k , but its attributes are grouped by referencing to the attribute indexes of \mathbf{x}_n^C . A bandwidth σ_j is defined as follows:

$$\sigma_j = \frac{1}{d_j} \sum_{i=1}^{d_j} \left[\left(\frac{4}{2+d_j} \right)^{\frac{1}{4+d_j}} \Gamma_i \lambda^{-\frac{1}{4+d_j}} \right], \quad (33)$$

where Γ_i denotes a rescale operator which is defined by the standard deviation of the i th attribute value in the j th attribute group among the λ instances.

In this paper, MLCA with the clustering-based approach is called MLCA-Clustering (MLCA-C).

The differences in attribute processing among the general approach, the individual-based approach (MLCA-I), and the clustering-based approach (MLCA-C) are depicted in Fig. 1.

The codes of MLCA, MLCA-I, and MLCA-C are available on GitHub¹.

1. <https://github.com/Masuyama-lab/MLCA>

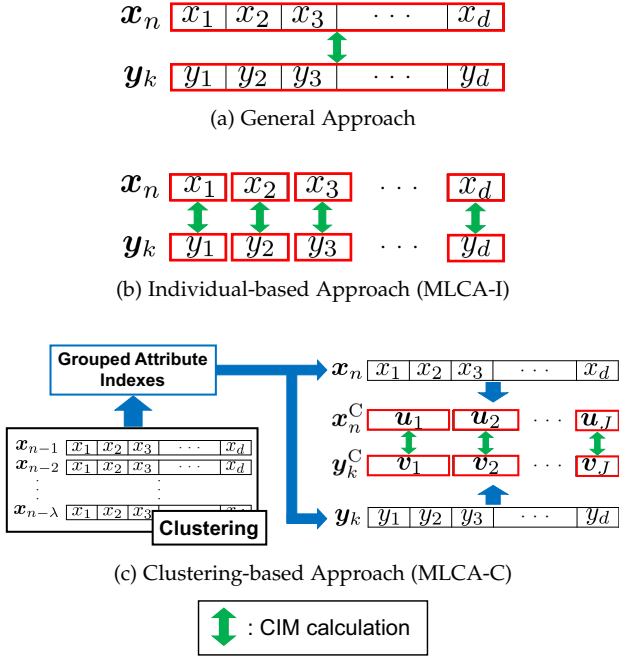


Fig. 1: Differences in the CIM calculation.

4 SIMULATION EXPERIMENTS

In this section, the ability of MLCA is evaluated from various perspectives. First, the continual learning ability of MLCA is analyzed with a two-dimensional synthetic multi-label dataset under the stationary and non-stationary environments. Next, the classification performance of MLCA is compared with other algorithms by using real-world multi-label datasets. Third, we evaluate the effect of a multi-epoch learning process to MLCA. Finally, we analyze the computational complexity of each algorithm.

4.1 Evaluation Metrics

We use the six metrics to evaluate the classification performance of multi-label classification algorithms [30].

- **Exact Match**
This is possibly the most strict performance metric. This metric measures whether a predicted set of labels for an instance is exactly equal to the true labels. The higher, the better.
- **F₁-score**
This is a harmonic mean between the Precision and the Recall, namely, a weighted measure of how many true labels are predicted and how many predicted labels are truly relevant. The higher, the better.
- **Macro-averaged AUC**
This is the area under the receiver operating characteristic curve. The Macro-averaged AUC is the arithmetic mean of the AUC for each label. This metric gives a better sense of the performance across all labels. The higher, the better.
- **Hamming Loss**
This metric computes the symmetric difference between the predicted and true labels, and divided by the total number of labels in a dataset. The lower, the better.

- **Ranking Loss**

This metric computes how many times a relevant label (a member of the true labels) appears ranked lower than a non-relevant label, namely, the average proportion of label pairs that are incorrectly ordered for an instance. The lower, the better.

- **Coverage**

This metric is defined as the distance to cover all possible labels assigned to an instance, namely, how many top-scored predicted labels are included without missing any true labels. The lower, the better. In this paper, we scaled the Coverage by the number of labels $N_l - 1$ thus the range of the Coverage is $[0, 1]$.

4.2 Continual Learning Ability

In theory, ART-based clustering is capable of learning new knowledge and preserving the learned knowledge without catastrophic forgetting by setting a fixed similarity threshold (i.e., a vigilance parameter in ART). Thus, MLCA can continually learn and preserve knowledge by adaptively generating nodes in response to changes of data distributions. Moreover, since MLCA has a fixed similarity threshold V and there is no node deletion process, MLCA does not inherently cause catastrophic forgetting.

In this section, we verify the continual learning ability of MLCA by using a two-dimensional synthetic multi-label dataset in the stationary and non-stationary environments. As a comparison algorithm, we apply MCIC [45] which is capable continual learning through a clustering process. Similar to MLCA, MCIC extracts and accumulates information from data by nodes. Therefore, MCIC is a competitive algorithm for MLCA. Note that, as mentioned in Section 2.2, MuENL [8] can perform sequential learning and class-incremental learning, i.e., continual learning. However, MuENL cannot deal with the situation where new data distributions are provided in the non-stationary environment, and thus MuENL cannot cope with the experimental conditions in this section.

Fig. 2 shows the two-dimensional synthetic multi-label dataset. The dataset consists of three distributions where each has 10,000 instances that follow a uniform distribution. In addition, as shown in Fig. 2b, seven types of label sets are defined. In this experiment, the instances are given to each algorithm in three different conditions: (1) all the instances are given at the same time (Fig. 2), (2) the three distributions are given in sequential order (Fig. 3), and (3) the seven distributions are given in sequential order (Fig. 4). The condition (1) is the stationary environment while the conditions (2) and (3) are the non-stationary environment. Here, the label information of each distribution is not changed. It should be noted that the given instances are presented as a sequence of three distributions in Fig. 3 whereas they are divided into seven distributions in Fig. 4. As a result, Fig. 4 is an easier problem than Fig. 3 because there are no overlap regions.

In the case that the three distributions are given in sequential order (Fig. 3), the instances in the overlapping regions are designed to contain label information of the already given distribution (e.g., $l_C = (1, 1)$) for generating a situation where the number of labels increases in a pseudo manner. Thus, as shown in Table 2, seven types of label sets

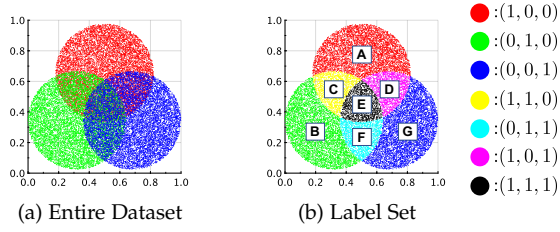


Fig. 2: Two-dimensional synthetic multi-label dataset and its label sets.

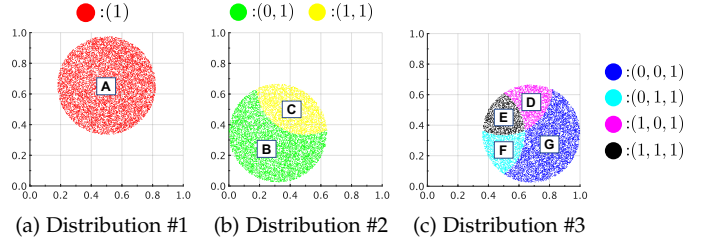


Fig. 3: Visualization of giving the three distributions in sequential order.

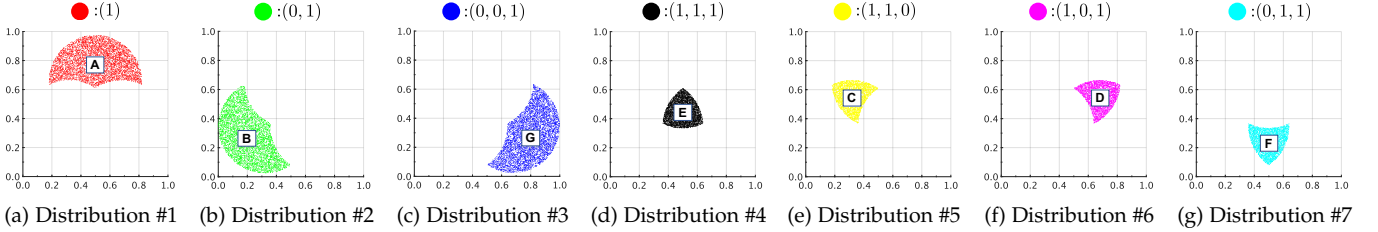


Fig. 4: Visualization of giving the seven distributions in sequential order.

TABLE 2: Transition of label sets during sequentially giving the three distributions in Fig. 3

Data	Subset	Transitions of Label Sets		
		Dist. #1	Dist. #2	Dist. #3
Distribution #1	A	$l_A = (1)$	$l_A = (1, 0)$	$l_A = (1, 0, 0)$
Distribution #2	B	—	$l_B = (0, 1)$	$l_B = (0, 1, 0)$
	C	—	$l_C = (1, 1)$	$l_C = (1, 1, 0)$
Distribution #3	D	—	—	$l_D = (1, 0, 1)$
	E	—	—	$l_E = (1, 1, 1)$
	F	—	—	$l_F = (0, 1, 1)$
	G	—	—	$l_G = (0, 0, 1)$

are defined after the three distributions are given. As the number of labels increases, the length of the label set also changes incrementally, e.g., $l_A = (1) \rightarrow l_A = (1, 0) \rightarrow l_A = (1, 0, 0)$. The similar label transition is occurred in the case of Fig. 4.

In order to analyze the continual learning capability, we evaluate the classification performance after each distribution is given. After learning the training instances of each distribution, the instances belonging to the learned distribution is used as test data. Namely, I) after learning the distribution #1, the classification performance is evaluated by using the test data of distribution #1. Next, II) after learning the distribution #2, the classification performance is evaluated by using the test data of the distributions #1 and #2. Then, III) after learning the distribution #3, the classification performance is evaluated by using the test data of the distributions #1, #2, and #3. We continue this procedure until all the distributions are given. We repeat the experiment 20 times with a different random seed to obtain consistent results. The parameters of MLCA are specified as follows: $N_y = 10$, $\lambda = 50$, and $V = 0.10$. The parameters of MCIC are specified as follows: $\delta = 0.1$, $K = 3$, $\lambda = 0.25$, $\beta_\mu = 2$, $\epsilon = \{0.011, 0.007, 0.005\}$, and a processing speed

is 10,000. Under the above parameter settings, MLCA and MCIC generate the similar number of nodes.

Table 3 shows classification performance after training the distributions in Figs. 2, 3, and 4. Focusing on MLCA, since high classification performance is maintained in the stationary and non-stationary environments, it is quantitatively shown that MLCA is capable of continual learning with a multi-label dataset. It is noteworthy that MLCA effectively accumulates the knowledge for the classification by using only about 900 to 2,700 nodes depending on the environment even if the distributions contains 30,000 instances in total. In contrast, the classification performance of MCIC is clearly inferior to MLCA even if the number of nodes is similar.

To analyze the above results in detail, Figs. 5-9 show the visualization of generated nodes and their label information in each algorithm. The results in Figs. 5-9 are a trial which showed the highest Exact Match in each algorithm among 20 trials. The color of a node indicates a label set that the node predicts. Once a testing instance is given, the nearest node predicts a label set for the testing instance corresponding to the color shown in the legend of Figs. 5-9.

Comparing Figs. 5a and 5b, MLCA can represent the seven distributions very well while MCIC fails to represent overlapped regions. Focusing on Figs. 6 and 7, MLCA and MCIC can properly preserve the information of the distribution #1. On the other hand, after learning the distributions #2 and #3, MLCA properly represents overlapped regions, but MCIC fails to do so. A similar tendency can be seen in Figs. 8 and 9. These results suggest that MLCA is capable of continuous learning in various environments while MCIC cannot accumulate and preserve information when distributions are adjacent or overlapping.

From the results in this section, it can be seen that MLCA adaptively generates nodes and incrementally learns label information from the given instances while maintaining the extracted knowledge.

TABLE 3: Results of classification performance for continual learning under environments in Figs. 2, 3, and 4

Algorithm	Parameter	Metric	Dist. #1	Dist. #2	Dist. #3	Dist. #4	Dist. #5	Dist. #6	Dist. #7
MLCA	$V = 0.1$	Exact Match	0.972 (0.001)	—	—	—	—	—	—
		Hamming Loss	0.010 (0.000)	—	—	—	—	—	—
		Number of Nodes	868.7 (16.6)	—	—	—	—	—	—
	$V = 0.1$	Exact Match	1.000 (0.000)	0.958 (0.001)	0.909 (0.001)	—	—	—	—
		Hamming Loss	0.000 (0.000)	0.014 (0.000)	0.031 (0.001)	—	—	—	—
		Number of Nodes	703.3 (10.6)	1183.5 (14.0)	1552.4 (17.0)	—	—	—	—
	$V = 0.1$	Exact Match	1.000 (0.000)	0.999 (0.000)	0.997 (0.000)	0.994 (0.001)	0.957 (0.002)	0.965 (0.004)	0.930 (0.008)
		Hamming Loss	0.000 (0.000)	0.000 (0.000)	0.001 (0.000)	0.003 (0.000)	0.015 (0.001)	0.012 (0.002)	0.024 (0.003)
		Number of Nodes	595.4 (13.2)	1165.5 (17.1)	1741.0 (19.7)	1954.1 (21.8)	2169.6 (20.4)	2407.7 (21.8)	2658.5 (24.4)
MCIC	$\epsilon = 0.011$	Exact Match	0.342 (0.039)	—	—	—	—	—	—
		Hamming Loss	0.222 (0.002)	—	—	—	—	—	—
		Number of Nodes	804.6 (9.3)	—	—	—	—	—	—
	$\epsilon = 0.007$	Exact Match	1.000 (0.000)	0.799 (0.001)	0.392 (0.230)	—	—	—	—
		Hamming Loss	0.000 (0.000)	0.068 (0.000)	0.212 (0.065)	—	—	—	—
		Number of Nodes	770.7 (8.7)	1320.9 (7.8)	1749.3 (16.8)	—	—	—	—
	$\epsilon = 0.005$	Exact Match	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	0.234 (0.000)	0.143 (0.000)	0.249 (0.000)	0.334 (0.000)
		Hamming Loss	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)	0.511 (0.000)	0.286 (0.000)	0.250 (0.000)	0.222 (0.000)
		Number of Nodes	620.0 (8.1)	1236.1 (11.2)	1865.4 (13.2)	2133.4 (15.1)	2377.3 (17.5)	2625.7 (17.0)	2869.2 (16.6)

The standard deviation is indicated in parentheses.

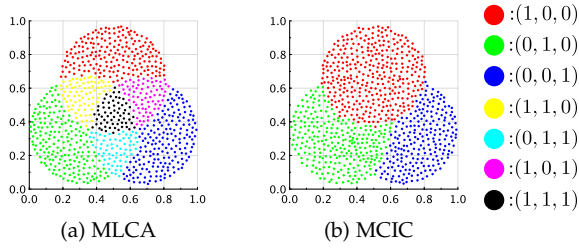


Fig. 5: Visualization of self-organizing results in the case the three distributions are given at the same time.

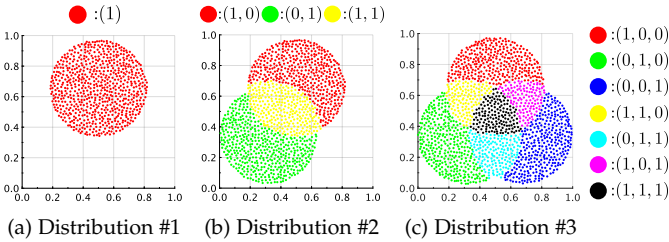


Fig. 6: Visualization of nodes in MLCA in the case the three distributions are given in sequential order.

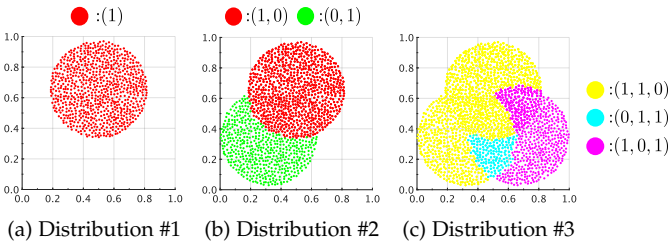


Fig. 7: Visualization of nodes in MCIC in the case the three distributions are given in sequential order.

In this section, since the label information of each distribution is not changed, the learning ability has been verified only for the knowledge that is represented by node po-

sitions/distributions. Regarding the label information (the prior probabilities and likelihood) of nodes in MLCA, it is sequentially updated based on the frequency of a label appearance. Therefore, although the location or distribution of nodes do not change, the meaning of the knowledge may change in the case where the label distribution is changed. This is known as concept drift.

From another point of view, the label forgetting nature of MLCA may lead to superior classification performance for stream data with concept drift than other algorithms. This is an interesting future research topic.

4.3 Quantitative Analysis

This section presents a comparison on the classification performance of MLCA, MLCA-I, and MLCA-C with that of MCIC [45], MuENL [8], mLODM [40], GLOCAL [39], MLSA- k NN [41], and ML- k NN [7] by utilizing real-world multi-label datasets.

The source code of MCIC ², MuENL ³, mLODM ⁴, GLOCAL ⁵, MLSA- k NN ⁶, and ML- k NN ⁷ are provided by authors.

4.3.1 Datasets

We use 16 real-world multi-label datasets that six numerical and six categorical datasets from the Mulan repository [53] and two numerical and two categorical datasets from the Extreme Classification repository [54]. Table 4 shows the statistics of the datasets. During our experiments, the training instances in each dataset are presented to each algorithm only once. In regard to pre-processing for datasets, mLODM and GLOCAL need [0, 1] scaling to maintain high classification performance, while other algorithms do not need the scaling. Therefore, we prepare the [0, 1] scaled data for mLODM and GLOCAL. For other algorithms, we use the raw data with no pre-processing.

2. <https://github.com/vu-luong/MCIC>
 3. https://www.lamda.nju.edu.cn/code_MuENL.ashx
 4. https://www.lamda.nju.edu.cn/code_mLODM.ashx
 5. https://www.lamda.nju.edu.cn/code_Glocal.ashx
 6. <https://github.com/canoalberto/MLSAkNN>
 7. http://www.lamda.nju.edu.cn/code_MLkNN.ashx

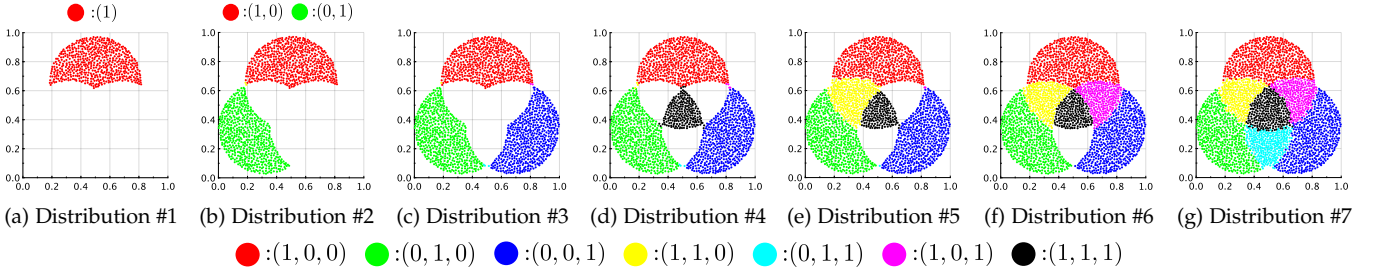


Fig. 8: Visualization of nodes in MLCA in the case the seven distributions are given in sequential order.

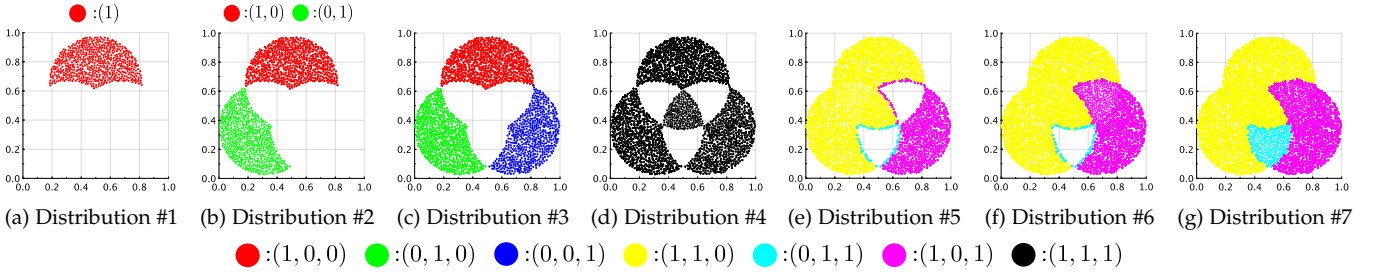


Fig. 9: Visualization of nodes in MCIC in the case the seven distributions are given in sequential order.

TABLE 4: Statistics of multi-label datasets

Dataset	Number of Instances	Number of Attributes		Number of Labels
		Numerical	Categorical	
Small-scale				
Flags	194	10	9	7
Emotions	593	72	0	6
Birds	645	258	2	19
Image	2,000	294	0	5
Scene	2,407	294	0	6
Yeast	2,417	103	0	14
VirusGO	207	0	749	6
GpositiveGO	519	0	912	4
Genbase	662	0	1,186	27
Medical	978	0	1,449	45
PlantGo	978	0	3,091	12
Langlog	1,460	0	1,004	75
Large-scale				
EURLex-4K	19,348	5,000	0	3,993
Mediamill	43,907	120	0	101
Bibtex	7,395	0	1,836	159
Delicious	16,105	0	500	983

4.3.2 Parameter Specifications

MLCA, MLCA-I, MLCA-C, and all the comparison algorithms have parameters which have an impact on the classification performance. This section presents parameter specifications of each algorithm in detail.

We use grid search to specify parameter values of each algorithm. Before grid search, we separate a dataset into training instances and test instances. The training instances are 90% of the dataset, and the testing instances are the remaining 10%. The testing instances are used only for the performance evaluation of the designed classifier (i.e., they are not used for parameter specifications). During grid search, we train an algorithm with the training instances, and test the algorithm by using the training instances again. Thus, we obtain a parameter setting which shows the high-

est classification performance to the training instances. Once the best parameter setting, which is an optimal to the training instances, is specified, the classification performance of the algorithm with the best parameter setting is evaluated by using the testing instances. Since grid search for specifying the parameters does not use the testing instances, the generalization ability of each algorithm can be properly evaluated in Section 4.3.4.

For specifying the best parameter setting, we calculate the Exact Match in each parameter specification for the training instances in each datasets listed in Table 4 except for large-scale datasets due to a time-consuming training process. Regarding the parameters for the large-scale datasets, we apply the same specification with Langlog dataset because it has the largest number of labels among small-scale datasets.

We repeat the evaluation 20 times (i.e., 2×10 -fold Cross Validation (CV)) with training instances selected by different random seeds. Although six evaluation metrics were introduced in Section 4.1, the Exact Match is used for specifying the parameters since it is the most strict evaluation metric. Table 5 summarizes the parameters of all the algorithms. In the following, the settings and results of grid search are explained in detail by separating MLCA and its variants and comparison algorithms.

- MCLA and its Variants

MLCA, MLCA-I, and MLCA-C have three parameters, i.e., the number of neighbor nodes N_y , an interval λ for adapting σ in the CIM, and a similarity threshold V . Among those parameters, the similarity threshold V has a large impact on the classification performance. Therefore, we perform grid search for V in increments of 0.05 over the range of $V = [0.05, 0.95]$ while fixing $N_y = 10$ and $\lambda = 50$. The parameters N_y and λ are the same specification as in [55]. In addition, the case of $V = 0.01$ is also considered. This is because that MLCA with a smaller V will tend to generate

TABLE 5: Parameter specifications of each algorithm

Algorithm	Parameter	Value	Grid Range	Description
MLCA	V	Grid Search	{0.01, 0.05, 0.10, ..., 0.95}	Similarity threshold
	λ	50	—	Interval for adapting σ in the CIM
	N_y	10	—	The number of neighbor nodes
MLCA-I	V	Grid Search	{0.01, 0.05, 0.10, ..., 0.95}	Similarity threshold
	λ	50	—	Interval for adapting σ in the CIM
	N_y	10	—	The number of neighbor nodes
MLCA-C	V	Grid Search	{0.01, 0.05, 0.10, ..., 0.95}	Similarity threshold
	λ	50	—	Interval for adapting σ in the CIM
	N_y	10	—	The number of neighbor nodes
MCIC	θ	Grid Search	{0.100, 0.200, 0.300, 0.400, 0.495}	Boundary of a cluster
	λ	Grid Search	{0.00, 0.05, 0.10, ..., 0.50}	Decay control parameter
	δ	0.1	—	Parameter for label set learning
	K	3	—	The number of nearest neighbors
	β_{mu}	3	—	Threshold of a mature weight
MuENL	λ_1	Grid Search	{1.0E-6, 1.0E-5, ..., 1.0E-0}	Trade-off parameter of ranking loss for a multi-label classifier
	λ_2	Grid Search	{1.0E-6, 1.0E-5, ..., 1.0E-0}	Trade-off parameter of l_2 regularization for a multi-label classifier
	I	20	—	The maximum iteration for updating a weight
mlODM	θ	Grid Search	{0.10, 0.20, ..., 0.90}	Approximation parameter for block coordinate descent
	μ	Grid Search	{0.10, 0.20, ..., 0.90}	Trade-off parameter for block coordinate descent
	γ	0.5	—	Bandwidth of a kernel function in rank-SVM
	C	1	—	Cost for rank-SVM
	I	4	—	The maximum number of iterations
GLOCAL	λ_3	Grid Search	{0.0, 1.0E-6, 1.0E-5, ..., 1.0E-0}	Regularization parameter for a global and local manifold
	λ_4	Grid Search	{0.0, 1.0E-6, 1.0E-5, ..., 1.0E-0}	Regularization parameter for a global and local manifold
	λ_1	1	—	Regularization parameter for a global and local manifold
	λ_2	0.125	—	Regularization parameter for a global and local manifold
	k	20	—	Dimension of a latent space
	g	3	—	The number of clusters of k -means
MLSA- k NN	m_{max}	Grid Search	{200, 400, ..., 2,000}	The maximum size of the window
	$kAdj_{max}$	Grid Search	{20, 40, ..., 200}	History length to specify k value of k NN
	m_{min}	50	—	The minimum size of the window
	p	1	—	Penalty ratio
	r	0.5	—	Reduction ratio
ML- k NN	k	3	{3, 6, ..., 30}	The number of nearest neighbors

more nodes which yields higher classification performance than that with a larger V .

The detailed results of grid search for the similarity threshold V in MLCA and its variants are shown in Fig.1 of the supplementary file.

In this paper, the parameters of MLCA and its variants are not specified for each dataset, but the same settings are applied to all datasets. Note that the parameters of the comparison algorithms are specified for each dataset in order to achieve the maximum classification performance. To specify the appropriate parameters for all datasets, we adopt the Friedman test and Nemenyi post-hoc analysis [56] to conduct statistical comparisons among the different parameter specifications ($V = [0.01, 0.45]$ in Fig. 1 of the supplementary file) by using results on the Exact Match of all datasets. The Friedman test is used to test the null hypothesis that all algorithms perform equally. If the null hypothesis is rejected, the Nemenyi post-hoc analysis is then conducted. The Nemenyi post-hoc analysis is utilized for all pairwise comparisons based on the ranks of results on the Exact Match of all datasets. The difference in the performance of two algorithms is treated as statistically significant if the p -value defined by the Nemenyi post-hoc analysis is smaller than the significance level. Here, the null hypothesis is rejected at the significance level of 0.05 both in the Friedman test and the Nemenyi post-hoc analysis.

Fig. 10 shows critical difference diagrams for different parameter specifications of each algorithm. A better specification has lower average ranks, i.e., on the right side of a critical distance diagram. In theory, the parameter specifications within a critical distance (i.e., a red line) do not have a statistically significance difference [56]. From the results in Fig. 10, parameter values to be used in comparisons of classification performance by testing instances are specified as follows: $V = 0.01$ and 0.30 for MLCA, $V = 0.01$ and 0.15 for MLCA-I, and $V = 0.01$ and 0.25 for MLCA-C. As mentioned earlier, MLCA and its variants utilize the above V values for all the datasets in the classification experiments.

• Comparison Algorithms

When grid search was used in the original paper where each algorithm was proposed, we use the same grid search with the same range of parameter values for each algorithm. Otherwise, we choose one or two parameters which have the largest effects on the classification performance. The details are as follows:

MCIC: The range of the decay controlled parameter λ is the same range in the original paper. Although the boundary of a cluster θ is fixed in the original paper, it has a large effect to the clustering performance. The range of θ is theoretically defined as $0 < \theta < 0.5$.

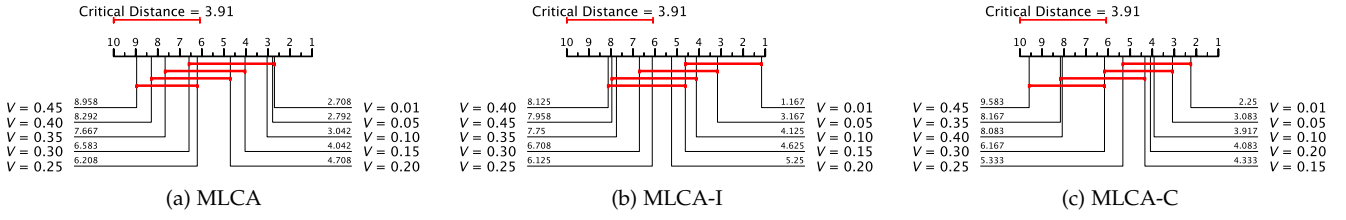


Fig. 10: Critical difference diagram for the Exact Match by different parameter specifications.

TABLE 6: Parameter specifications for each dataset by grid search

Dataset	MCIC		MuENL		mlODM (Scaling)		mlODM (Raw)		GLOCAL (Scaling)		GLOCAL (Raw)		MLSA- k NN		ML- k NN
	θ	λ	λ_3	λ_4	θ	μ	θ	μ	λ_3	λ_4	λ_3	λ_4	m_{\max}	$kAdj_{\max}$	k
Flags	0.100	0.00	1.0	1.0	0.90	0.30	0.10	0.30	0.0	1.0	0.0	0.0	200	20	3
Emotions	0.100	0.00	1.0E-6	1.0E-6	0.70	0.10	0.70	0.30	1.0E-4	1.0	1.0E-6	1.0E-6	200	20	3
Birds	0.100	0.00	1.0	1.0	0.70	0.40	0.10	0.10	1.0E-1	1.0	0.0	0.0	200	20	3
Image	0.100	0.00	1.0E-6	1.0E-6	0.70	0.40	0.70	0.40	1.0E-5	1.0E-4	1.0E-5	1.0E-4	200	20	6
Scene	0.100	0.00	1.0	1.0	0.90	0.10	0.90	0.10	0.0	0.0	1.0E-2	1.0	200	20	3
Yeast	0.100	0.00	1.0	1.0	0.90	0.30	0.50	0.10	1.0	1.0E-1	0.0	0.0	600	20	3
VirusGO	0.100	0.00	1.0E-1	1.0E-6	0.70	0.10	0.70	0.10	0.0	0.0	0.0	0.0	200	20	3
GpositiveGO	0.495	0.45	1.0	1.0	0.90	0.10	0.90	0.10	0.0	0.0	0.0	0.0	200	20	3
Genbase	0.200	0.00	1.0	1.0E-6	0.10	0.10	0.10	0.10	0.0	0.0	0.0	1.0E-6	600	20	3
Medical	0.100	0.20	1.0	1.0	0.10	0.10	0.10	0.10	0.0	1.0E-4	1.0E-6	1.0E-4	200	20	3
PlantGO	0.100	0.00	1.0	1.0E-6	0.20	0.10	0.20	0.10	1.0E-3	1.0E-4	1.0E-3	1.0E-4	400	20	6
Langlog	0.100	0.10	1.0	1.0	N/A	N/A	N/A	N/A	1.0E-1	1.0E-5	1.0E-2	1.0E-3	800	20	3
EURLex-4K	0.100	0.10	1.0	1.0	N/A	N/A	N/A	N/A	1.0E-1	1.0E-5	1.0E-2	1.0E-3	800	20	3
Mediamill	0.100	0.10	1.0	1.0	N/A	N/A	N/A	N/A	1.0E-1	1.0E-5	1.0E-2	1.0E-3	800	20	3
Bitbtex	0.100	0.10	1.0	1.0	N/A	N/A	N/A	N/A	1.0E-1	1.0E-5	1.0E-2	1.0E-3	800	20	3
Delicious	0.100	0.10	1.0	1.0	N/A	N/A	N/A	N/A	1.0E-1	1.0E-5	1.0E-2	1.0E-3	800	20	3

N/A indicates that an algorithm could not build a predictive model within 5 days under the available computational resources.

MuENL: In the original paper, grid search is performed to $\lambda_1, \lambda_2 \in \{0.001, 0.01, 0.1, 1\}$. In this paper, we set a wider range that includes the above one.

mlODM: The parameters and their grid ranges are the same as in the original paper.

GLOCAL: The parameters and their grid ranges are the same as in the original paper.

MLSA- k NN: In the original paper, the authors mentioned that a window size $kAdj_{\max}$ and the maximum window size m_{\max} need to be set appropriately values in advance depending on datasets. In the original paper, these parameters are fixed as $kAdj_{\max} = 100$ and $m_{\max} = 1,000$. In this paper, grid search was performed in the ranges around these values.

ML- k NN: In the original paper, grid search was performed in the range of $k = \{8, 9, 10, 11, 12\}$. However, there was no significant difference in the classification performance. In this paper, we set wider range for finding an appropriate value.

Table 5 summarizes the parameters for grid search and their ranges. Table 6 shows the parameters that indicate the highest Exact Match for the training instances in each dataset as determined by grid search. N/A indicates that an algorithm can not build a predictive model within 5 days under the available computational resources. In this paper, we assign the worst evaluation value to each metric if an algorithm can not build a predictive model.

Since MuENL has 10 parameters, we only consider three parameters because the rest of the parameters are related

to class-incremental learning that is not performed in this section. Regarding mlODM and GLOCAL, there is a large difference in the classification performance depending on whether a dataset is pre-processed or not. In this paper, therefore, we use the [0, 1] scaled data and raw data without pre-processing in the learning process of mlODM and GLOCAL, and denote them as mlODM (Scaling), mlODM (Raw), GLOCAL (Scaling), and GLOCAL (Raw), respectively. Note that MCIC and MuENL are capable of continual learning while mlODM, GLOCAL, MLSA- k NN and ML- k NN cannot deal with it.

4.3.3 Experimental Conditions

We evaluate the classification performance of each algorithm by using datasets in Table 4 and parameter specifications in Table 5. For small-scale datasets, we repeat the evaluation 20 times (i.e., 2×10 CV) with a different random seed for obtaining consistent averaging results. In this section, each algorithm is trained by using the same training instances with the one in Section 4.3.2, and 10% of instances, which are not used in Section 4.3.2, are used as testing instances. For the large-scale datasets, indexes for training and testing data are provided in [54]. Thus, we use those indexes for training and testing in each algorithm. Similar to Section 4.3.2, the Friedman test and Nemenyi post-hoc analysis [56] are utilized. The Friedman test is used to test the null hypothesis that all algorithms perform equally. If the null hypothesis is rejected, the Nemenyi post-hoc analysis is then conducted. The Nemenyi post-hoc analysis

is used for all pairwise comparisons based on the ranks of results over all the evaluation metrics of all datasets. Here, the null hypothesis is rejected at the significance level of 0.05 both in the Friedman test and the Nemenyi post-hoc analysis. All computations are carried out on Matlab 2020a with 2.2GHz Xeon Gold 6238R processor and 768GB RAM.

4.3.4 Experimental Results

We compare the classification performance of each algorithm by using a critical difference diagram defined by the Nemenyi post-hoc analysis. The detailed results of six evaluation metrics for each dataset are summarized in Tables 1-4 on the supplementary file.

Fig. 11 shows a critical difference diagram based on the classification performance for all the datasets. A better specification has lower average ranks, i.e., on the right side of a critical distance diagram. In theory, the parameter specifications within a critical distance (i.e., a red line) do not have a statistically significance difference [56].

The results are derived from all the evaluation metrics for all the datasets. Therefore, we regard that MLCA ($V=0.01$) shows excellent classification performance on various datasets. MLCA-I ($V=0.01$) and MLCA-C ($V=0.01$) are also superior algorithms than other comparison algorithms. Focusing on MLCA and its variants with a larger V specification, MLCA-C ($V=0.25$) shows a lower rank than MLCA-I ($V=0.15$) and MLCA ($V=0.30$). It means that MLCA-C is capable of maintaining high classification performance while compressing information.

Regarding comparison algorithms, ML- k NN shows a superior classification performance than other algorithms except for MLCA and its variants. In Fig. 11, mLODM (Scaling) and GLOCAL (Scaling) are better than mLODM (Raw) and GLOCAL (Raw), respectively. Since MCIC, MuENL, and MLSA- k NN are capable of handling streaming data, their classification performance is inferior to algorithms that perform batch learning such as ML- k NN, mLODM, and GLOCAL.

In order to discuss the features of each algorithm in detail, Figs. 12 and 13 show critical difference diagrams defined by numerical datasets and categorical datasets, respectively. In the case of numerical datasets (Fig. 12), the classification performance of MLCA and its variants are still superior to comparison algorithms. MLCA-I ($V=0.01$) shows the lowest rank. In addition, MLCA-I ($V=0.15$) also shows a good performance. Thus, we regard that MLCA-I is suitable for numerical data sets. On categorical datasets (Fig. 13), ML- k NN and GLOCAL perform very well in contrast to the case of numerical datasets. However, these algorithms have an obvious drawback compared to MLCA and its variants because they require all the training instances in advance. In terms of MLCA and its variants, MLCA ($V=0.01$) shows the lowest rank. The performance of MLCA and MLCA-I is greatly affected by a specification of a similarity threshold V . In contrast, the difference in performance between MLCA-C ($V=0.01$) and MLCA-C ($V=0.25$) is small. Therefore, we regard that MLCA-C has stable classification performance on nominal datasets.

Table 7 shows the average number of generated nodes after learning the training instances. Focusing on MLCA and MLCA-I, these algorithms generate only a very small

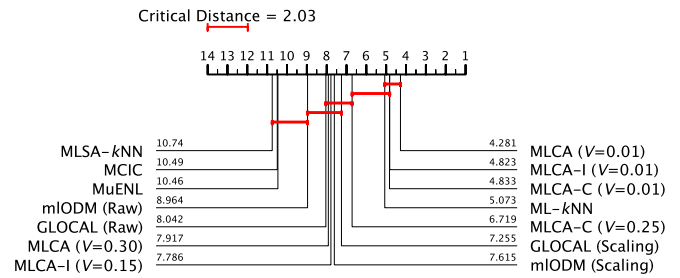


Fig. 11: Critical difference diagram based on the classification performance for all the datasets.

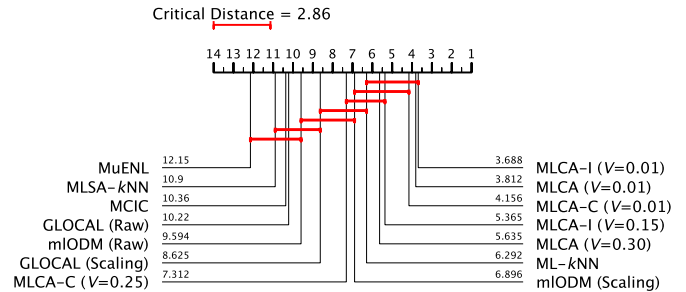


Fig. 12: Critical difference diagram based on the classification performance for numerical datasets.

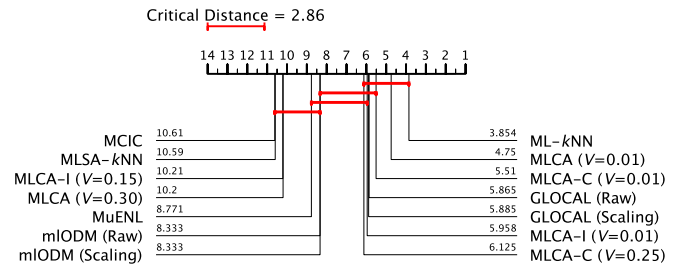


Fig. 13: Critical difference diagram based on the classification performance for categorical datasets.

number of nodes in the case of nominal datasets, especially when a similarity threshold V is large. On the other hand, MLCA-C can maintain the sufficient number of nodes for classification even in the case of nominal datasets. Thus, MLCA-C is considered to be a strong algorithm for nominal datasets, and this property can be seen in Figs. ??-??.

From the results in Figs. 11-13, and Table 7, the characteristics of MLCA and its variants can be analyzed as follows:

- MLCA**
 This algorithm can be the first-choice algorithm because it shows stable and high classification performance for both numerical and nominal datasets. In other words, it has an advantage if the attribute type of the dataset is unknown. Furthermore, in the case of a numerical dataset, MLCA shows high classification performance regardless of a specification of the similarity threshold V . This means that MLCA can maintain high classification performance and high information compression performance, simultaneously.

TABLE 7: Average number of nodes generated by MLCA, MLCA-I, and MLCA-C

Dataset	MLCA		MLCA-I		MLCA-C	
	$V = 0.01$	$V = 0.30$	$V = 0.01$	$V = 0.15$	$V = 0.01$	$V = 0.25$
Flags	174.6 (1.00)	163.9 (0.94)	170.5 (0.98)	88.2 (0.51)	167.7 (0.96)	19.1 (0.11)
Emotions	533.7 (1.00)	532.9 (1.00)	533.7 (1.00)	532.1 (1.00)	533.7 (1.00)	244.6 (0.46)
Birds	580.5 (1.00)	580.5 (1.00)	580.5 (1.00)	146.0 (0.25)	580.0 (1.00)	170.6 (0.29)
Image	1800.0 (1.00)	1597.1 (0.89)	1800.0 (1.00)	1736.8 (0.96)	1800.0 (1.00)	1729.0 (0.96)
Scene	2159.0 (1.00)	2076.3 (0.96)	2159.0 (1.00)	2151.0 (0.99)	2159.0 (1.00)	2148.0 (0.99)
Yeast	2175.3 (1.00)	2123.1 (0.98)	2175.3 (1.00)	2168.8 (1.00)	2175.3 (1.00)	2015.6 (0.93)
VirusGO	159.3 (0.86)	11.2 (0.06)	86.3 (0.46)	5.2 (0.03)	159.3 (0.86)	122.1 (0.66)
GpositiveGO	412.3 (0.88)	28.3 (0.06)	183.0 (0.39)	8.8 (0.02)	412.3 (0.88)	308.8 (0.66)
Genbase	192.0 (0.32)	1.7 (0.00)	31.9 (0.05)	1.0 (0.00)	192.0 (0.32)	130.5 (0.22)
Medical	871.0 (0.99)	55.0 (0.06)	450.9 (0.51)	14.0 (0.02)	871.0 (0.99)	736.5 (0.84)
PlantGO	775.6 (0.88)	39.5 (0.04)	301.5 (0.34)	12.5 (0.01)	766.4 (0.87)	644.0 (0.73)
Langlog	1130.6 (0.86)	761.4 (0.58)	1116.7 (0.85)	513.1 (0.39)	1130.6 (0.86)	1118.4 (0.85)

The value in the parentheses indicates the ratio to the number of generated nodes against the number of training instances.

• **MLCA-I**

This algorithm shows the outstanding classification performance on the numerical datasets. On the other hand, the classification performance on the nominal datasets is low in comparison with the results on to numerical datasets. Therefore, MLCA-I has an advantage if the dataset contains only numerical attributes.

• **MLCA-C**

This algorithm shows stable and high classification performance for both numerical and nominal datasets although it is not as good as MLCA. It is notable that MLCA-C stably has small rank values for both nominal and numerical datasets even when a specification of a similarity threshold V is large. Therefore, MLCA-C can achieve high classification performance and high information compression for both numerical and nominal datasets.

Table 8 summarizes the characteristics of MLCA and its variants based on the above analysis.

4.4 Effects of a Multi-Epoch Learning Process

MLCA, MLCA-I, and MLCA-C utilize generated nodes as classifiers. This means that the clustering performance on the training instances has a huge impact on the classification performance. The nodes of MLCA, MLCA-I, and MLCA-C are adaptively and continually generated/updated by the given instances. Therefore, it is possible to improve the clustering performance by learning the training instances in multiple epochs, and consequently to improve the classification performance of MLCA, MLCA-I, and MLCA-C. This feature is one of the advantages of MLCA, MLCA-I, and MLCA-C against the other compared algorithms.

We only apply the datasets listed in Table 4 except for the large-scale datasets due to a time-consuming training process. Fig. 14 shows results of the Exact Match of MLCA, MLCA-I, and MLCA-C with the learning of the training instances for 1 to 10 epochs. The conditions of this experiment are the same as in Section 4.3.

The following observation is obtained: As the number of epochs increases, the value of the Exact Match increases or remains roughly the same in most cases except for the

TABLE 8: Summary of characteristics of MLCA and its variants

Algorithm	Similarity Threshold V	Classification Performance	
		Numerical	Nominal
MLCA	Small	Very High	Very High
	Large	Medium	Low
MLCA-I	Small	Very High	Low
	Large	Medium	Low
MLCA-C	Small	High	High
	Large	Medium	Medium

Birds dataset. In particular, it is effective for categorical data. Therefore, we regard that the multi-epoch learning process is generally beneficial for MLCA, MLCA-I, and MLCA-C.

4.5 Computational Complexity

This section presents the computational complexity of MLCA, MLCA-I, MLCA-C, and comparison algorithms. Specifically, MLCA and its variants are analyzed in detail.

Regarding MLCA, the computational complexity of each process is as follows: For computing a bandwidth of a kernel function in the CIM is $\mathcal{O}(\frac{n}{\lambda}d)$ (line 5 in Alg. 1), for computing the CIM is $\mathcal{O}(ndK)$ (line 11 in Alg. 1), for sorting the result of the CIM is $\mathcal{O}(K \log K)$ (line 12 in Alg. 1), and for computing the label probability is $\mathcal{O}(N_l N_y)$ (line 24 in Alg. 1). Here, n is the number of training instances, λ is an interval for adapting σ , K is the number of nodes, N_l is a size of a label set, and N_y is the predefined number of neighbor nodes for y_{k_1} . Thus, the total computational complexity of MLCA is $\mathcal{O}(\frac{n}{\lambda}d + ndK + K \log K + N_l N_y)$.

In terms of MLCA-I and MLCA-C, the difference of a training process is only in CIM^I and CIM^C which is defined in (28) and (31), respectively. Since CIM^I considers an individual attribute of a training instance separately, it takes $\mathcal{O}(nd^2 K)$. CIM^C applies a clustering approach to attributes of a training instance every λ instances. Thus, $\mathcal{O}(\frac{n}{\lambda}(\frac{n}{\lambda}d + ndK + K \log K))$ is additionally required. As a result, the computational complexity of MLCA-I and MLCA-C are $\mathcal{O}(\frac{n}{\lambda}d + nd^2 K + K \log K + N_l N_y)$ and $\mathcal{O}((\frac{n}{\lambda} + 1)(\frac{n}{\lambda}d + ndK + K \log K) + \frac{n}{\lambda}N_l N_y)$, respectively.

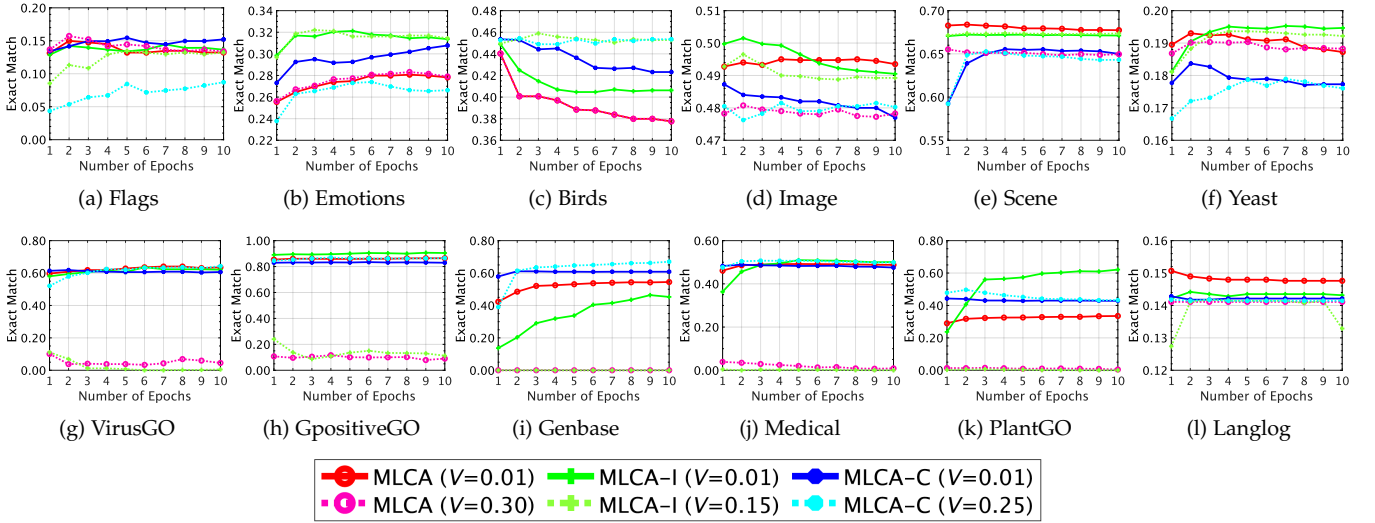


Fig. 14: Results of the Exact Match of MLCA, MLCA-I, and MLCA-C with multiple epochs.

Table 9 summarizes the computational complexity of MLCA, MLCA-I, MLCA-C, and comparison algorithms. Here, variables in the computational complexity of comparison algorithms are as follows:

MCIC: T is a time period defined as $T = \frac{1}{\lambda} \log(\frac{\beta_\mu}{(\beta_\mu - 1)} + 1)$. Here, n is the number of instances, λ and β_μ are the parameters of MCIC. K_p and K_o are the number of mature and immature clusters of MCIC, respectively.

MuENL: N_l is the size of a label set, n is the number of instances, and d is the dimension of instances. Note that the complexity of MuENL in Table 9 shows a pairwise label ranking classifier, not including a label incremental learning process.

mIODM: n is the number of instances, N_l is the dimension of a current relevant label, and I is the number of iterations of an optimization process.

GLOCAL: n is the number of instances, n_m is the number of instances of a partitioned training instances, k is a rank of a label matrix which satisfies $k < N_l$.

MLSA- k NN: d is the dimension of a training instance, N_l is a size of a label set, m_{\max} is the maximum size of the window, and m_{\min} is the minimum size of the window.

ML- k NN: n is the number of training instances, d is the dimension of a training instance, k is the number of nearest neighbors of k -NN, and N_l is a size of a label set. Here, $\mathcal{O}(n^2d)$ is for k -NN computation, and $\mathcal{O}(nkN_l)$ is for the label probability computation.

With respect to the computational complexity of each algorithm, MLSA- k NN and MCIC shows their superior computational efficiency than the other algorithms. MLCA, MLCA-I, MLCA-C, and ML- k NN have moderate computational efficiency. These algorithms do not dramatically increase the computational complexity even when the number of instances and labels in a dataset is large. In contrast, MuENL, mIODM, and GLOCAL have a polynomial complexity in terms of the number of instances or the size of a label set. Therefore, we consider that these algorithms are time-consuming and could not generate a predictive model in a valid time, especially for the large-scale datasets with a

large number of training instances and labels.

TABLE 9: Summary of computational complexity

Algorithm	Computational Complexity	Ref.
MLCA	$\mathcal{O}(\frac{n}{\lambda}d + ndK + K \log K + N_lN_y)$	—
MLCA-I	$\mathcal{O}(\frac{n}{\lambda}d + nd^2K + K \log K + N_lN_y)$	—
MLCA-C	$\mathcal{O}((\frac{n}{\lambda} + 1)(\frac{n}{\lambda}d + ndK + K \log K) + \frac{n}{\lambda}N_lN_y)$	—
MCIC	$\mathcal{O}(n + \frac{n}{T}(K_p + K_o))$	—
MuENL	$\mathcal{O}(ndN_l^2)$	[8]
mIODM	$\mathcal{O}(nN_l^2I)$	[40]
GLOCAL	$\mathcal{O}(n^2 + n_m^2 + kn)$	[57]
MLSA- k NN	$\mathcal{O}(m_{\max}d + m_{\max}N_l + m_{\max} \log_2 \frac{m_{\max}}{m_{\min}})$	[41]
ML- k NN	$\mathcal{O}(n^2d + nkN_l)$	[58]

The training and testing time on a CPU are summarized in Tables 5 and 6 of the supplementary file.

5 CONCLUDING REMARKS

This paper proposed a multi-label classification algorithm capable of continual learning by extending our preliminary research [55], namely MLCA. In addition, two variants of MLCA were proposed by modifying the calculation method of the CIM, namely MLCA-I and MLCA-C. The proposed algorithms consist of two components: The CIM-based ART and the Bayesian approach for label probability computation. Because both components can deal with a situation where new training instances and corresponding labels are sequentially provided, the proposed algorithms can realize continual learning. The results of extensive experiments from qualitative and quantitative perspectives showed that MLCA has competitive classification performance to other well-known algorithms while maintaining the continual learning ability. Furthermore, the results also showed that the performance of MLCA can be enhanced by modifying the calculation method of the CIM.

The ability to adapt to concept drift [59] and to handle mixed numerical and categorical data [60] are significant factors for clustering-based algorithms capable of continual

learning. A future research topic is to examine the performance of the proposed algorithms under concept drift and to improve them. It is also an important future research topic to modify the proposed algorithms for mixed datasets with both numerical and categorical attributes.

ACKNOWLEDGMENT

This research was supported by Ministry of Education, Culture, Sports, Science and Technology - JAPAN (MEXT) Leading Initiative for Excellent Young Researchers (LEADER). The Universiti Malaya Impact-oriented Interdisciplinary Research Grant Programme (IIRG) - IIRG002C-19HWB, Universiti Malaya Covid-19 Related Special Research Grant (UMCSRG) CSRG008-2020ST from University of Malaya. National Natural Science Foundation of China (Grant No. 61876075), Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), The Stable Support Plan Program of Shenzhen Natural Science Fund (Grant No. 20200925174447003), Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531).

REFERENCES

- [1] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 57–71, 2019.
- [2] J. Zhang, C. Li, D. Cao, Y. Lin, S. Su, L. Dai, and S. Li, "Multi-label learning with label-specific features by resolving label correlations," *Knowledge-Based Systems*, vol. 159, pp. 148–157, 2018.
- [3] H. Han, M. Huang, Y. Zhang, X. Yang, and W. Feng, "Multi-label learning with label specific features using correlation information," *IEEE Access*, vol. 7, pp. 11 474–11 484, 2019.
- [4] L. He, L. Xie, H. Shu, and S. Hu, "Discrete semi-supervised learning for multi-label image classification and large-scale image retrieval," *Multimedia Tools and Applications*, vol. 78, no. 17, pp. 24 519–24 537, 2019.
- [5] C. Zhang and Z. Li, "Multi-label learning with label-specific features via weighting and label entropy guided clustering ensemble," *Neurocomputing*, vol. 419, pp. 59–69, 2021.
- [6] X. Zheng, P. Li, Z. Chu, and X. Hu, "A survey on multi-label data stream classification," *IEEE Access*, vol. 8, pp. 1249–1275, 2019.
- [7] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.
- [8] Y. Zhu, K. M. Ting, and Z.-H. Zhou, "Multi-label learning with emerging new labels," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 10, pp. 1901–1914, 2018.
- [9] S. Furao and O. Hasegawa, "An incremental network for on-line unsupervised classification and topology learning," *Neural Networks*, vol. 19, no. 1, pp. 90–106, 2006.
- [10] F. Shen and O. Hasegawa, "A fast nearest neighbor classifier based on self-organizing incremental neural network," *Neural Networks*, vol. 21, no. 10, pp. 1537–1547, 2008.
- [11] N. Masuyama, C. K. Loo, and F. Dawood, "Kernel Bayesian ART and ARTMAP," *Neural Networks*, vol. 98, pp. 76–86, 2018.
- [12] N. Masuyama, C. K. Loo, and S. Wermter, "A kernel Bayesian adaptive resonance theory with a topological structure," *International Journal of Neural Systems*, vol. 29, no. 5, p. 1850052 (20 pages), 2019.
- [13] N. Masuyama, C. K. Loo, H. Ishibuchi, N. Kubota, Y. Nojima, and Y. Liu, "Topological clustering via adaptive resonance theory with information theoretic learning," *IEEE Access*, vol. 7, pp. 76 920–76 936, 2019.
- [14] N. Masuyama, N. Amako, Y. Nojima, Y. Liu, C. K. Loo, and H. Ishibuchi, "Fast topological adaptive resonance theory based on correntropy induced metric," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2019, pp. 2215–2221.
- [15] W. Liu, P. P. Pokharel, and J. C. Principe, "Correntropy: Properties and applications in non-Gaussian signal processing," *IEEE Transactions on Signal Processing*, vol. 55, no. 11, pp. 5286–5298, 2007.
- [16] G. J. McLachlan, S. X. Lee, and S. I. Rathnayake, "Finite mixture models," *Annual Review of Statistics and its Application*, vol. 6, pp. 355–378, 2019.
- [17] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.
- [18] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.
- [19] B. Fritzke, "A growing neural gas network learns topologies," *Advances in Neural Information Processing Systems*, vol. 7, pp. 625–632, 1995.
- [20] G. A. Carpenter and S. Grossberg, "The ART of adaptive pattern recognition by a self-organizing neural network," *Computer*, vol. 21, no. 3, pp. 77–88, 1988.
- [21] S. Marsland, J. Shapiro, and U. Nehmzow, "A self-organising network that grows when required," *Neural Networks*, vol. 15, no. 8, pp. 1041–1058, 2002.
- [22] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," *Cognitive Science*, vol. 11, no. 1, pp. 23–63, 1987.
- [23] S. C. Tan, J. Watada, Z. Ibrahim, and M. Khalid, "Evolutionary fuzzy ARTMAP neural networks for classification of semiconductor defects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 933–950, 2014.
- [24] A. L. Matias and A. R. R. Neto, "OnARTMAP: A fuzzy ARTMAP-based architecture," *Neural Networks*, vol. 98, pp. 236–250, 2018.
- [25] A. L. Matias, A. R. R. Neto, C. L. C. Mattos, and J. P. P. Gomes, "A novel fuzzy ARTMAP with area of influence," *Neurocomputing*, vol. 432, pp. 80–90, 2021.
- [26] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, no. 6, pp. 759–771, 1991.
- [27] B. Vigdor and B. Lerner, "The Bayesian ARTMAP," *IEEE Transactions on Neural Networks*, vol. 18, no. 6, pp. 1628–1644, 2007.
- [28] L. Wang, H. Zhu, J. Meng, and W. He, "Incremental local distribution-based clustering using Bayesian adaptive resonance theory," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3496–3504, 2019.
- [29] L. E. B. da Silva, I. Elnabarawy, and D. C. Wunsch II, "Distributed dual vigilance fuzzy adaptive resonance theory learns online, retrieves arbitrarily-shaped clusters, and mitigates order dependence," *Neural Networks*, vol. 121, pp. 208–228, 2020.
- [30] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1819–1837, 2013.
- [31] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.
- [32] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Random k -labelsets for multilabel classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 7, pp. 1079–1089, 2010.
- [33] H. Liu, X. Wu, and S. Zhang, "Neighbor selection for multilabel classification," *Neurocomputing*, vol. 182, pp. 187–196, 2016.
- [34] A. Clare and R. D. King, "Knowledge discovery in multi-label phenotype data," in *Principles of Data Mining and Knowledge Discovery*, 2001, pp. 42–53.
- [35] A. Osojnik, P. Panov, and S. Džeroski, "Multi-label classification via multi-target regression on data streams," *Machine Learning*, vol. 106, no. 6, pp. 745–770, 2017.
- [36] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, vol. 14, no. 7, 2001, pp. 681–687.
- [37] N. Zhang, S. Ding, and J. Zhang, "Multi layer ELM-RBF for multi-label learning," *Applied Soft Computing*, vol. 43, pp. 535–545, 2016.
- [38] H. Zhang, J. Yang, G. Jia, S. Han, and X. Zhou, "ELM-MC: multi-label classification framework based on extreme learning machine," *International Journal of Machine Learning and Cybernetics*, pp. 1–14, 2020.
- [39] Y. Zhu, J. T. Kwok, and Z.-H. Zhou, "Multi-label learning with global and local label correlation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 6, pp. 1081–1094, 2018.

- [40] Z.-H. Tan, P. Tan, Y. Jiang, and Z.-H. Zhou, "Multi-label optimal margin distribution machine," *Machine Learning*, vol. 109, no. 3, pp. 623–642, 2020.
- [41] M. Roseberry, B. Krawczyk, Y. Djenouri, and A. Cano, "Self-adjusting k nearest neighbors for continual learning from multi-label drifting data streams," *Neurocomputing*, vol. 442, pp. 10–25, 2021.
- [42] G. G. Colombini, I. B. M. de Abreu, and R. Cerri, "A self-organizing map-based method for multi-label classification," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 4291–4298.
- [43] A. Benyettou, Y. Bennani, A. Bendahmane, and G. Cabanes, "Semisupervised multi-label classification through topological active learning," *International Journal on Communications Antenna and Propagation (I. Re. CAP)*, vol. 7, no. 3, pp. 222–232, 2017.
- [44] S. Boulbazine, G. Cabanes, B. Matei, and Y. Bennani, "Online semi-supervised growing neural gas for multi-label data classification," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8, doi: 10.1109/IJCNN.2018.8489776.
- [45] T. T. Nguyen, M. T. Dang, A. V. Luong, A. W.-C. Liew, T. Liang, and J. McCall, "Multi-label classification via incremental clustering on an evolving data stream," *Pattern Recognition*, vol. 95, pp. 96–113, 2019.
- [46] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 698–713, 1992.
- [47] F. Benites, F. Brucker, and E. Sapozhnikova, "Multi-label classification by ART-based neural networks and hierarchy extraction," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, 2010, pp. 1–9.
- [48] F. Benites and E. Sapozhnikova, "Improving scalability of ART neural networks," *Neurocomputing*, vol. 230, pp. 219–229, 2017.
- [49] L. X. Yuan, S. C. Tan, P. Y. Goh, C. P. Lim, and J. Watada, "Fuzzy ARTMAP with binary relevance for multi-label classification," in *International Conference on Intelligent Decision Technologies*, 2017, pp. 127–135.
- [50] J.-Y. Park and J.-H. Kim, "Incremental class learning for hierarchical classification," *IEEE Transactions on Cybernetics*, vol. 50, no. 1, pp. 178–189, 2018.
- [51] S. Marriott and R. F. Harrison, "A modified fuzzy ARTMAP architecture for the approximation of noisy mappings," *Neural Networks*, vol. 8, no. 4, pp. 619–641, 1995.
- [52] D. J. Henderson and C. F. Parmeter, "Normal reference bandwidths for the general order, multivariate kernel density derivative estimator," *Statistics & Probability Letters*, vol. 82, no. 12, pp. 2198–2205, 2012.
- [53] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "Mulan: A java library for multi-label learning," *The Journal of Machine Learning Research*, vol. 12, pp. 2411–2414, 2011.
- [54] K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma, "The extreme classification repository: Multi-label datasets and code," <http://manikvarma.org/downloads/XC/XMLRepository.html>, [Online; accessed 20-August-2021].
- [55] N. Masuyama, Y. Nojima, C. K. Loo, and H. Ishibuchi, "Multi-label classification based on adaptive resonance theory," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020, pp. 1913–1920.
- [56] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, no. 1, pp. 1–30, 2006.
- [57] C. Zhu, D. Miao, Z. Wang, R. Zhou, L. Wei, and X. Zhang, "Global and local multi-view multi-label learning," *Neurocomputing*, vol. 371, pp. 67–77, 2020.
- [58] P. Skryjomski, B. Krawczyk, and A. Cano, "Speeding up k -nearest neighbors classifier for large-scale multi-label learning on GPUs," *Neurocomputing*, vol. 354, pp. 10–19, 2019.
- [59] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.
- [60] A. Ahmad and S. S. Khan, "Survey of state-of-the-art mixed data clustering algorithms," *IEEE Access*, vol. 7, pp. 31 883–31 902, 2019.



Naoki Masuyama (S'12–M'16) received the B.Eng. degree from Nihon University, Funabashi, Japan, in 2010, the M.E. degree from Tokyo Metropolitan University, Hino, Japan in 2012, and the Ph.D. degree from the Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia, in 2016.

He is currently an Assistant Professor with the Department of Computer Science and Intelligent Systems, Osaka Prefecture University, Sakai, Japan.

His current research interests include clustering, data mining, and continual learning.



Yusuke Nojima received the B.S. and M.S. Degrees in mechanical engineering from Osaka Institute of Technology, Osaka, Japan, in 1999 and 2001, respectively, and the Ph.D. degree in system function science from Kobe University, Hyogo, Japan, in 2004.

Since 2004, he has been with Osaka Prefecture University, Osaka, Japan, where he is currently a Professor in Department of Computer Science and Intelligent Systems.

His research interests include evolutionary fuzzy systems, evolutionary multiobjective optimization, and parallel distributed data mining. He was a guest editor for several special issues in international journals. He was a task force chair on Evolutionary Fuzzy Systems in Fuzzy Systems Technical Committee of IEEE Computational Intelligence Society. He was an associate editor of IEEE Computational Intelligence Magazine (2014–2019).



Chu Kiong Loo (SM'14) holds a Ph.D. (University Sains Malaysia) and B.Eng. (First Class Hons in Mechanical Engineering from the University of Malaya).

He was a Design Engineer in various industrial firms and is the founder of the Advanced Robotics Lab. at the University of Malaya. He has been involved in the application of research into Perus's Quantum Associative Model and Pribram's Holonomic Brain Model in humanoid vision projects. Currently he is Professor of Computer Science and Information Technology at the University of Malaya, Malaysia. He has led many projects funded by the Ministry of Science in Malaysia and the High Impact Research Grant from the Ministry of Higher Education, Malaysia. Loo's research experience includes brain-inspired quantum neural networks, constructivism-inspired neural networks, synergetic neural networks and humanoid research.



Hisao Ishibuchi (M'93–SM'10–F'14) received the B.S. and M.S. degrees in precision mechanics from Kyoto University, Kyoto, Japan, in 1985 and 1987, respectively, and the Ph.D. degree in computer science from Osaka Prefecture University, Sakai, Osaka, Japan, in 1992.

Since 1987, he has been with Osaka Prefecture University for 30 years. He is currently a Chair Professor with the Department of Computer Science and Engineering, Southern University of Science Technology, Shenzhen, China.

His current research interests include fuzzy rule-based classifiers, evolutionary multiobjective optimization, many-objective optimization, and memetic algorithms.

Dr. Ishibuchi was the IEEE Computational Intelligence Society (CIS) VicePresident for Technical Activities from 2010 to 2013. He was an IEEE CIS AdCom Member from 2014 to 2019, an IEEE CIS Distinguished Lecturer from 2015 to 2017, and an Editor-in-Chief of the IEEE Computational Intelligence Magazine from 2014 to 2019. He is also an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON CYBERNETICS, and the IEEE ACCESS.