Volume 14- Number 2 – 2022 (41 -53)

Clustering Large-Scale Data using an Incremental Heap Self-Organizing Map

Mehdi Fasanghari[*]
Iran Telecommunication Research Center
Tehran, Iran.
fasanghari@itrc.ac.ir

Helena Bahrami
School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, New Zealand.
helena.bahrami@aut.ac.nz

Hamideh Sadat Cheraghchi
Iran Health Insurance Organization
Tehran, Iran.
h.cheraghchi@ihio.gov.ir

Received: 5 April 2022 – Revised: 29 May 2022 - Accepted: 28 June 2022

Abstract— **In machine learning and data analysis, clustering large amounts of data is one of the most challenging tasks. In reality, many fields, including research, health, social life, and commerce, rely on the information generated every second. The significance of this enormous amount of data in all facets of contemporary human existence has prompted numerous attempts to develop new methods for analyzing large amounts of data. In this research, an Incremental Heap Self-Organizing Map (IHSOM) is proposed for clustering a vast amount of data that continues to grow. The gradual nature of IHSOM enables environments to change and evolve. In other words, IHSOM can quickly adapt to the size of a dataset. The heap binary tree structure of the proposed approach offers several advantages over other structures. Initially, the topology or neighborhood relationship between data in the input space is maintained in the output space. The outlier data are then routed to the tree's leaf nodes, where they may be efficiently managed. This capability is supplied by a probability density function as a threshold for allocating more similar data to a cluster and transferring less similar data to the following node. The pruning and expanding nodes process renders the algorithm noise-resistant, more precise in clustering, and memory-efficient. Therefore, heap tree structure accelerates node traversal and reorganization following the addition or deletion of nodes. IHSOM's simple user-defined parameters make it a practical unsupervised clustering approach. On both synthetic and real-world datasets, the performance of the proposed algorithm is evaluated and compared to existing hierarchical self-organizing maps and clustering algorithms. The outcomes of the investigation demonstrated IHSOM's proficiency in clustering tasks.**

Keywords: Self-organizing map (SOM); Binary heap tree; Incremental hierarchical structure; Probability density function.

Article type: Research Article

[DOI: 10.52547/itrc.14.2.41]

I. Introduction

In the era of computer technology and with the advent of the Internet of Things, the amount of data generated by humans is rapidly increasing. Data analysts and computer engineers face a formidable challenge in managing this quantity of data. These massive datasets are meaningless by necessity. In other words, many types of data are received from widely diverse devices ranging from personal computers to sensors found in our surroundings, the majority of which are signals or, at most, figures or strings with no

[*] Corresponding Author

"label". Unsupervised clustering techniques are the best options for extracting usable information from this ocean of vast amounts of data. Unsupervised techniques help to build representations of the input data without human intervention by discovering patterns in the observed data that can be used for decision making, predicting future inputs, and facilitating interpretation, as it is necessary to assess a significant volume of heterogeneous, unstructured, and unlabeled data collected from multiple sources. Numerous efforts have been made to employ clustering techniques for large-scale data analysis, such as K-means [1, 2], Decision Trees [3-5], Support Vector Machines [6-9], Bayesian Classifiers [10-12], and Neural Networks [13-17]. Among these methods, Neural Networks, specifically Self-Organizing Map (SOM) [18-21], is the preferred clustering technique due to its fast-learning process and less time-consuming performance, high accuracy in clustering complex multi-dimensional data, and its great visualization ability makes SOM a potent clustering technique. Although SOMs are popular clustering algorithms, their predefined, fixed lattice makes them inappropriate for contexts that are subject to change. Therefore, numerous researchers have attempted to enhance the standard SOM by incorporating dynamism into its structure in order to make it more adaptable to expanding and changing information. Among these successful endeavors, incremental hierarchical SOMs have garnered considerable interest due to their adaptable and flexible qualities, making them resilient and accurate in clustering tasks. In this research, a novel Incremental Heap Self-Organizing Map (IHSOM) algorithm is presented that inherits the hierarchical adaptive properties of its predecessors while incorporating new techniques for outlier handling, topology preservation, and rapid accurate clustering. The suggested algorithm provides a number of advantages over prior approaches, including: 1) IHSOM maintains the topology or neighborhood relationship between data in output space as well. 2) It can handle outlier data efficiently by sending them to the tree's leaf nodes. A probability density function offers this capability as a threshold for allocating more similar data to a cluster node and sending less similar data to the following node. 3) The proposed algorithm's pruning and expanding nodes processes make it noise-resistant, more accurate in clustering applications, and memory-efficient. 4) The heap tree structure of the technique accelerates node traversal and reorganization following the addition or deletion of nodes. 5) IHSOM's single user-defined parameter makes it a practical unsupervised approach for clustering tasks.

The remaining sections of the paper are organized as follows: Section 2 examines, analyzes, and compares several prior related research publications. The third section presents an overview of the Binary tree time adaptive SOM, a hierarchical tree structure SOM-based approach from which IHSOM inherits some of its characteristics. In Section 4, the proposed IHSOM algorithm is thoroughly explored. Section 5 includes experimental findings and analysis of algorithm performance for both simulated and real-world datasets. The final section concludes the paper.

II. Related Works

This section reviews a brief history of some of the previous research works carried out in large-scale data clustering.

SOTA is an algorithm Javier Herrero and colleagues developed for grouping gene expression. SOTA is a hierarchical method that evolves using the topology of a binary tree. This algorithm employs a top-down approach to clustering to resolve the highest hierarchical levels before proceeding to the lowest level. The tree's growth can be halted at any desired level using a criterion based on approximation probability distribution [22].

S-tree refers to a self-organizing tree suggested by Marcos Campos and Gail Carpenter [23] for data clustering and online vector quantization. This method is a hierarchical unsupervised clustering tree that applies a twofold route search process for clustering and vector quantization tasks, including a Gauss-Markov source benchmark and a picture compression application [23]. Anne Denton et al. developed a solution related to the density-based clustering algorithm DENCLUE, but with alternative reasoning. Their description of a cluster center in their P-trees defines a natural hierarchy that builds clusters at various levels [24]. Shen Furao et al. proposed an incremental self-organizing neural network with enhanced self-organization for unsupervised online learning. Their algorithm, ESOINN, is an enhanced version of their previous SOINN method. This algorithm replaces SOINN's two-layer network structure with a single-layer network, separates clusters with dense overlap, requires fewer parameters, and is more stable than SOINN [25]. Shen Furao and Osamu Hasegawa suggested a quick closest neighbor classifier based on a self-organizing incremental neural network, which is an improved version of SOINN. This Adjusted SOINN Classifier (ASC) automatically learns the number of prototypes necessary to define the decision boundary and learns new information without discarding previously learned information [26-28]. Hamed Shah-Hosseini created BTASOM, a binary tree time adaptive SOM, to establish a hierarchical structure of neurons using TASOM networks.

BTASOM's hierarchical binary tree structure makes it computationally efficient in static and dynamic contexts. The novelty of the BTASOM model is inspired by real trees in that young branches are thin and flexible, but elderly branches are massive and unyielding [29]. Bingwei Liu et al. propose a scalable sentiment categorization approach utilizing a naive Bayes classifier. Their method provides a

[DOI: 10.52547/itrc.14.2.41]

straightforward and comprehensive solution for sentiment mining on huge datasets utilizing a Naïve Bayes classifier and the Hadoop framework [30]. Self-tuned kernel spectral clustering model (KSC) for large-scale complex networks was created by Raghvendra Mall et al. The KSC approach functions by constructing a model on a subgraph of the complex network. The method automatically determines the number of clusters by employing projections of validation nodes in eigenspace to build an affinity matrix [31, 32]. Amanpreet Kaur Toor and Amarpreet Singh presented an advanced clustering algorithm (ACA) to deal with large data sets and high dimensionality. ACA accelerates the clustering process by calculating the distance between the current data input and the new cluster center; if this distance is less than or equal to the distance to the old center, the data input remains in the cluster to which it was previously assigned. Therefore, calculating the distance between this data input and the remaining k-1 clustering centers is unnecessary. This technique expedites access to K-1 cluster center locations [33]. Raghavi Chouhan and Abhishek Chauhan suggested a solution for the K-medoid algorithm that eliminates the shortcoming of the existing K-medoid for the clustering of large amounts of data. Their improved partitioning clustering technique automatically adjusts the number of clusters by comparing the similarity value to a predetermined similarity threshold, resulting in the establishment of stable clusters each time the procedure is executed [34]. Petra Perner introduced case-based reasoning as an incremental learning and knowledge discovery strategy for mining massive data. The approach finds similarities among a group of instances that have been previously processed and saved in a case database, and the closest (most similar) examples with their corresponding results are picked and displayed in the output [35, 36]. Rui Maximo Esteves et al. suggested a novel competitive K-Means algorithm to overcome the stochastic nature of K-Means++ and its time-consuming serial processing mechanism. A new parallel seeding algorithm, CK-Means, surpasses the K-Means++ serial approach to data analysis by applying clustering to subsets of the dataset in parallel and selecting the winner cluster based on a fitness measure. In addition, they utilized a MapReduce framework that scaled effectively with enormous datasets for their innovative CK-Means [37].

Zhongwei Shi et al. developed a class incremental learning technique for real-time recognizing frequent label combinations in a developing data stream. In their method, the learning process consists of two parts. In the first step of initializing the learning model, several samples with common label combinations are collected. In the second stage, each consecutive sample's label combination is compared to the set of frequent label combinations. If the label combination is not present in the list of label combinations, its occurrence number will be recorded to update the learning model [38]. Hongwei Zhang et al. offer a load-balancing self-organizing incremental neural network (LB-SOINN), which is an additional type of SOINN. LB-SOINN solves the deficiencies of E-SOINN, such as its dependence on input data order, instability, and consequent loss of precision as the data dimension rises. LB-SOINN employs a load-balancing strategy to improve network stability by utilizing the learning time of each node as a representation of its load. Additionally, it applies a smoothing technique based on Voronoi tessellation to reduce the turbulence caused by deleting the overlapping zone between classes. In addition, Hongwei Zhang et al. [39] presented a novel measure of similarity between two vectors that is ideal for online incremental high-dimensional learning tasks.

Although these valuable studies have shown good clustering outcomes, there are downsides to employing the techniques mentioned above. For example, Nearest Neighbor clustering is an expensive technique whose cost climbs exponentially as dataset size and dimension increase.

Widely used K-means-based algorithms are too static, and the cluster numbers should be supplied at the outset. Decision-tree models are subject to instability and overfitting (small variations in the input data might result in a completely different tree). In addition, decision tree approaches generate biased trees when dominating classes already exist. Naïve Bayesian Classifier has strong feature independence assumptions that lead this method to inaccurate classification. In other words, it has trouble understanding the interplay of dataset properties. Despite the rapid and accurate categorization of SOM, the output lattice is fixed and cannot adapt to a changing environment.

Among the clustering approaches mentioned above, the incremental hierarchical neural network is a suitable methodology for clustering expanding large-scale datasets because of its high classification accuracy, low time consumption, and ability to readily adapt to a growing dataset size. ACS, BTASOM, and SOTA are successful instances of this methodology. In the part that follows, various background knowledge regarding the suggested model is examined to lay the foundation for the proposed method.

III. Background Knowledge

As stated in the preceding sections, incremental neural networks are effective methods for unsupervised clustering tasks that are best suited for processing vast amounts of data. This section of the paper introduces a new incremental neural network-based strategy that can address the drawbacks of the previous version. IHSOM is the proposed method based on a SOM and binary heap tree structure. In reality, IHSOM is an adaptive incremental neural network that learns based on SOM rules, and its nodes are organized as a heap binary tree. Before introducing

[DOI: 10.52547/itrc.14.2.41]

IHSOM, a quick review of SOM and binary heap tree will be presented. In addition, the final subsection includes an overview of the BTASOM method, which served as the basis for the proposed IHSOM algorithm.

A. Conventional SOM

Teivo Kohonen [40] introduced the SOM in 1989 as an unsupervised neural network employing a competitive learning algorithm. The original SOM algorithm maps a complex multi-dimensional input space $\{x_i(n): i = 1, \dots, m\}$ into a lattice of output space $\{O_j: j = 1, \dots, k\}$ by associating each input vector to every node in the output lattice via a weighted connection. Adjusting the weight of winning nodes (nodes that most closely resemble the input vectors) and their neighboring (neighbor) nodes in the lattice to make them more similar to the input vectors constitutes the learning phase of the method. Finally, similar input vectors will be mapped to nearby winning nodes on the output map, completing the clustering task.

As a result of reducing a high-dimensional environment to a low-dimensional space, one of the valuable characteristics of the SOM is its capacity to visualize data. It uses a neighborhood function to maintain the same topological relationship between input space data and the mapped output lattice. These two characteristics of the SOM algorithm and its fast unsupervised learning algorithm make its output results more understandable and suitable for classification tasks. However, SOM has a lattice of a fixed number of output nodes that cannot adapt to the growing size dataset. In order to solve this deficiency, the fixed lattice of SOM is substituted with the hierarchical binary heap structure.

B. Binary Heap Tree

The Binary heap tree [41-43] is a complete binary tree data structure that fulfills the heap ordering task. In the heap tree algorithm, each node is ordered concerning its parent node value so that the child node value is either less than (or equal to) its parent in min heap ordering or greater than (or equal to) its parent in max ordering. In order to build a heap tree, the first input data is considered the root node; then, according to the min-heap or max-heap strategy, when the following data is added to the tree, it will be compared with its parent. If the value of the child node is greater than its parent in min-heap or less than its parent in the max-heap, their position in the tree will swap. With the help of the Heap tree technique, SOM fixed size lattice becomes an incremental tree whose nodes are sorted and adapted to the changing size of data inputs. By combining these two methods, the proposed IHSOM algorithm is created.

C. A review of the BTASOM algorithm

BTASOM is a hierarchical binary tree structure SOM based on the time adaptive SOM technique. In other words, each node in BTASOM is an individual TASOM network. Each node in TASOM has a fixed number of neurons that create a one-dimensional lattice. To preserve the algorithm's binary tree structure, each node in the TASOM network of BTASOM has a maximum of two neurons.

BTASOM's binary tree topology facilitates dynamic data input adaptation and accelerates grouping. The level and nodes of BTASOM are constructed at the time of data entry. According to their quantization inaccuracy, the nodes may be added or removed. The BTASOM algorithm begins with a root node that generates the tree's initial and highest level. The root node's TASOM weights are then initialized with random values. The TASOM learning algorithm adapts the root node's neuron (or neurons) to the input data distribution by the time data is fed to the tree. If the root node contains any child nodes, the input data will be sent to the relevant child at the subsequent stage. Until the input vector reaches a leaf node, the TASOM learning algorithm trains each visited node as the input vector is propagated through the tree. After training the last leaf node in the tree with the current input vector, the next input data are introduced, and the training procedure is repeated from the root node to a leaf node until the BTASOM has received the predetermined number of input data. During the training phase, the quantization error of each node is computed; based on its value, an undesired node is removed from the tree or an intransitive node is inserted. If a node's quantization error is smaller than a predefined minimal error, then the node and its subtree are removed from the BTASOM tree entirely. A new child node or nodes are added to the BTASOM tree if the quantization error of a node is greater than a preset maximum error. The maximum and minimum quantization error bounds are defined by the intervals [Minqe × Kmul, Maxqe × Kmul]. Trial-and-error selection of a positive integer value for the constant parameter Kmul enables the TASOM network to accurately approximate the input distribution. The fixed parameters Maxqe and Minqe provide the maximum and minimum quantization error values so that the inequality $0 \leq Minqe \leq Maxqe/2$ is satisfied. In Shah-Hosseini's paper, Maxqe and Minqe are set to 0.5 and 0.1, respectively. By selecting smaller Maxqe and Minqe values, more nodes will grow on the BTASOM tree, resulting in a more accurate approximation. Although additional nodes in a tree result in a more precise categorization, it is memory-intensive and slows down clustering. Therefore, the trade-off between clustering precision, memory usage, and algorithm performance must be considered.

In BTASOM, the number of child nodes that a node's subtree has determines the flexibility of that node. Similar to a natural tree, as the number of nodes in a subtree increases, the root node becomes less flexible. Otherwise, the flexibility of a root node will be increased in a subtree with fewer child nodes. By

[DOI: 10.52547/itrc.14.2.41]

this mechanism, the behavior of a real tree branch in nature is simulated; the heavy old branches of a tree may shrink, and new branches may grow based on environmental conditions. BTASOM controls the flexibility (shrinking-growing) of each node by the parameter s_f. Each neuron has its own s_f. Neurons with a low s_f learn more quickly and become more adaptable to environmental changes, whereas neurons with a high s_f become less adaptable.

Although BTASOM is an exciting approach to clustering with some remarkable properties, there are some disadvantages to using this algorithm. BTASOM cannot preserve the topology of input data in output space due to a lack of specifying the topological neighborhoods for its output nodes. Changing the order of sequence of serial input data produces different results, meaning the resulting tree will be altered with the different sequence input data in each run. When a node is deleted, that node's child nodes will be removed too. The deleted node is an outlier located in a sparse area of data distribution, and deleting the whole subtree causes losing potential clusters. Finally, there are too many parameters in the BTASOM algorithm to adjust by the user.

In the following sections, a new Incremental clustering method will be proposed to solve these defects using a binary heap tree structure for a self-organizing map called IHSOM.

IV. Proposed Model: Incremental Heap Self-Organizing Map (IHSOM)

IHSOM is a hierarchical adaptive SOM method with a binary heap tree topology. IHSOM begins with a root node so that a random weight vector according to the size of the training dataset is assigned to it. Since the serial data input in some previous incremental methods like BTASOM, SOINN, and its modified versions causes instability in the output results, IHSOM uses batch data input to overcome this difficulty. Thus, each node is trained with the whole input dataset. Every node in IHSOM has a cost used as a critical value to build the heap tree. The cost of each node is determined by the mean value of the weights designated to it. It is determined for which data inputs the present node is the winning node by calculating the Euclidean distance between the input data and the node's weight vector. Afterward, the most similar data inputs are selected to assign to the node using the probability density function. To this end, a threshold is defined as the mean of probability density function values, and those data inputs whose Euclidean distance probability density functions are above the threshold are chosen to earmark for the current node. The majority of these selected data (about 70 percent) are removed from the original dataset so that less similar data to the current node are passed to the next node for clustering. By keeping 30 percent of similar data to the current node in the original data set, there is a chance for them to assign to another node and the previous winning node to find the best matching unit. The strategy of eliminating similar data to the current node from the training dataset determines the direct outliers (or data with less resemblance to the others) down to the tree's leaf nodes. The next node will be treated like the previous one. A random weight vector according to the size of the remaining input data is dedicated to the node, its cost value is determined, the Euclidean distance of the data presented to the node is calculated, and similar data to the cluster node are selected based on their probability density functions. This process is repeated until the size of the dataset is zero. After creating all the cluster nodes, the binary heap tree is built out of them. The next node is added to the tree as the left child for the root node, considering the first node as the root node for the binary heap tree. If the cost of the child node is greater than its parent, their place is swapped. Otherwise, they will maintain their positions. The next node is added to the bottom level of the tree (as the right child), and this routine is repeated so that the max binary heap tree is built. Afterward, the weights of each node are updated with the batch learning of the SOM weight adjustment rule, and the neighboring function for each node in the heap tree is calculated. The neighboring radius decreases through the iterations (like the original SOM algorithm). Then Heap tree of SOM nodes is re-ordered according to the new weights. In order to give more dynamism to the IHSOM tree, new nodes will be added in dense areas, and nodes in a sparse area of data distribution removed based on their quantization error.

According to the above explanation, the IHSOM algorithm is an adaptive hierarchical network that can easily adapt to the size of the input training data. The tree structure of IHSOM speeds up nodes traversing and restructuring after adding or deleting nodes. Adding or deleting nodes mechanism in the proposed algorithm is attained in place. Thus, unlike BTASOM, just one node is deleted, and the tree can restructure quickly. Due to its heap ordering property, the IHSOM can also preserve the topological relationship between data input space and the output space. IHSOM has a few parameters that need to be adjusted by the user, making the algorithm a perfect choice for unsupervised clustering tasks. The graphical explanation of the IHSOM algorithm is illustrated in Figure 1.

[DOI: 10.52547/itrc.14.2.41]

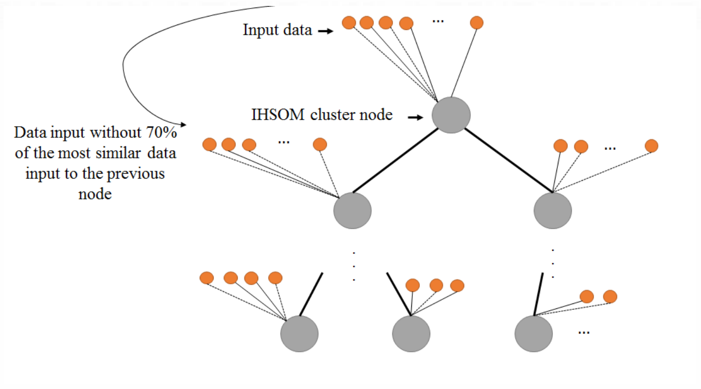

Input data
IHSOM cluster node
Data input without 70% of the most similar data input to the previous node
...

Fig. 1. Graphical illustration of proposed IHSOM

Quantization error and topography error are employed to determine the IHSOM's reliability. These two measures are outlined in detail below.

A. Quantization Error

The Quantization Error (qe) [44] calculates the average distance of the data inputs to the cluster centroids (winner node) by which they are represented (see Equation 1) in order to evaluate the fitting of IHSOM to the dataset.

$$qe = \frac{1}{N}\sum \left\| \bar{x}_i - BMU_{\bar{x}_i} \right\|, \qquad (1)$$

where N represents the number of data inputs and $BMU_{\bar{x}_i}$ represents the best matching unit of the corresponding $\bar{x}_i$ data inputs.

The least average quantization error means that the map properly fits the data. A smaller amount of quantization error indicates that input data are closer to their cluster centroids or the winner node. By increasing the number of IHSOM tree nodes, quantization error can be reduced due to the proper coverage over data input space on the IHSOM map. However, the lower quantization error may lead to distortion of the IHSOM map's topology. Therefore, a trade-off between vector quantization and projection properties of the IHSOM should be considered.

B. Topographic Error

Topographic Error (te) [45] evaluates the accuracy with which IHSOM maintains relative distances between locations in the input data space. This mistake takes into account the distance between the IHSOM map's top and second best-matching units. If the first and second best-matching units (BMU) are adjacent vectors, then the topology of input data is preserved by the IHSOM map. Thus, the lower the topographic error, the better topology preservation. The topographic error is calculated as follows:

$$te = \frac{1}{N}\sum_{i=1}^{N} u(\bar{x}_i), \qquad (2)$$

where the function $u(\bar{x}_i)$ returns 1 if the first and second BMUs of the $\bar{x}_i$ data input are not contiguous and 0 otherwise.

C. IHSOM algorithm in details

In this part of the paper, the IHSOM algorithm is specified by step:

1. Set input dataset as the Training Data.
2. Create a node as the root node for the IHSOM.
3. Set the root node as the Current Node.
4. Set a random weight vector for the neurons of the Current Node. (According to the number of Training Data, the weight vector grows larger).
5. Calculate the mean of the weight vector as M_w for Current Node (M_w is the cost of the current node).

$$M_w = \frac{\sum_{i=1}^{n} w_i}{n}, \qquad (3)$$

where w_i is the weight vector, n is the weight vector's size, and i denotes the index of a particular weight in the vector.

Determine the Euclidean distance (ED-CurrentNode) between each input data vector and the weight vector of the node (the node's neurons) as $ED_{CurrentNode}$.

$$ED_{CurrentNode} = \sqrt{\sum_{i=0}^{i=n} (TrainingData_i - W_i)^2}\,. \qquad (4)$$

6. Calculate for the current node, the probability density function of $ED_{CurrentNode}$ as ED_{pdf}.

$$ED_{pdf} = \frac{1}{\sqrt{2\pi\sigma^2}} \times e^{-\frac{(ED_{CurrentNode_i}-\mu)^2}{2\sigma^2}}, \qquad (5)$$

where μ and σ are the mean value and standard deviation of $ED_{CurrentNode}$, respectively.

7. Find the mean of the ED-pdf and set it as the threshold.

[DOI: 10.52547/itrc.14.2.41]

$$Threshold = \frac{\sum_{i=1}^{n}(ED_{pdf})_i}{n}. \tag{6}$$

In this step, the current node is set as a winning node to those input vectors ($TrainingData_{winner}$) which their ED_{pdf} are more than $Threshold$.

8. Omit 70% of the $TrainingData_{winner}$ from the original dataset and set it as Training Data. In this way, 30% of assigning data to the Current winning node have a chance to assign to another node and the previous winning node to find Best Matching Unit.
9. Create the Next node, set it as the current node, and go to step 4. Repeat until the size of the Training Data is zero.
10. Call BuildHeap () Function to build a binary heap tree from the nodes created in the earlier stages.
11. Update weights using the batch learning SOM weight adjusting rule [35].

$$w_{updated} = \frac{\sum_{i=1}^{n} Nb_{heapNode_{i},j} \times TrainingData_i}{\sum_{i=1}^{n} Nb_{heapNode_{i},j}}, \tag{7}$$

where, $Nb_{heapNode_{i},j}$, is the neighborhood function around each heap tree's node and is calculated as follows:

$$Nb_{heapNode_{i},j} = \exp\left(-\frac{\left\|r_j - r_{heapNode_i}\right\|^2}{2\delta^2(t)}\right), \tag{8}$$

where $\left\|r_j - r_{heapNode_i}\right\|$ is the distance between $heapNode_i$ and the jth node in the binary heap tree and $\delta(t)$ is the neighborhood width at time t.

12. Decrease the neighborhood width according to the below equation:

$$\delta(t) = \delta_0\left(1 - \frac{t}{t_{max}}\right), \tag{9}$$

where δ_0 is the initial value of the neighborhood width and will be set as the total number of IHSOM tree's nodes at the beginning of the algorithm and decreases by the time. Moreover, the current iteration is *t*, and the maximum iteration for IHSOM's learning phase is t_{max}.

13. Call MaxHeapify() Function to heap sorting the tree according to the new weight vectors.

Calculate quantization error for each IHSOM node:

$$qe = \frac{1}{n}\sum_{i=1}^{n}\|TrainingData_i - w_i\|. \tag{10}$$

The average Euclidean distance between the input vector $TrainingData$ and its matching winning node's weight vector w_i is the quantization error qe.

If $qe < 10^{-6}$, Call HeapDeleteNode (), function.

Else if $qe > 0.5$, create a new node with a node cost M_w which is a random number between the parent and its left child' M_w of the place that the new node intended to insert.

14. Call HeapInsertNode() function.
15. Choose 30% of the new inserted node parent's training dataset to assign to it (new node).

Set a random weight vector such that the upper bound and lower bound are between the M_w of the inserted node parent and left child, respectively. Then, go to the step 13, Re-compute M_w for the inserted node.

Calculate Topological Error te for the IHSOM tree using the following equation:

$$te = \frac{1}{N}\sum_{i=1}^{N} u(\overline{TrainingData_i}), \tag{11}$$

where $u(\overline{TrainingData_i})$ is 1 if best and second-best matching units are not adjacent and is 0, otherwise. If maximum iteration is reached, stop else, go to step 13. The BuildHeap, MaxHeapify, HeapDeleteNode, and HeapInsertNode algorithms are defined in the followings.

1) BuildHeap algorithm

The binary heap tree, as previously stated, is a complete binary tree that may be built using an array (list) structure. A heap tree is represented in a level order from left to right. Thus, it is necessary to know the parent and its left and right child's location in the array. In order to build a heap tree from an input array by assuming the root index at 1, the parent of node *i* is located at the index floor (i/2). The left and right child of node *i* are respectively at indices (2i) and (2i+1). According to the above explanation, the following steps are used to build a max heap tree.

```
Put all IHSOM nodes in an array; $H[i]$ = IHSOM nodes$_i$.
Set $heap_{size} = length(H)$.
Set $j = \left\lfloor \frac{heap_{size}}{2} \right\rfloor$; the last parent in the array.
3-1. Repeat step 4 until $j = 1$.
Call MaxHeapify(H); this function heapifies the sub tree i and its children in a way that the largest node is stored at the root.
Decrease j by 1 if the stop condition meets exit else, go back to step 4.
(12)
$j = \left\lfloor \frac{heap_{size}}{2} \right\rfloor - 1$
```

2) MaxHeapify algorithm

The aforementioned MaxHeapify function, which maintains the heap property of the IHSOM heap tree is specified as below:

[DOI: 10.52547/itrc.14.2.41]

```
    Set $LeftChildIndex = 2i$.
    Set $RightChildIndex = 2i + 1$.
    If $LeftChildIndex < heap_{size}$ and $H[LeftChildIndex] > H[i]$.
    Then set $Largest = LeftChildIndex$.
    Else set $Largest = i$.
    If $RightChildIndex < heap_{size}$ and $H[i] > H[Largest]$.
    Then set $Largest = RightChildIndex$.
    If $Largest \neq i$.
    Then exchange $H[i] \leftrightarrow H[Largest]$.
    Call MaxHeapify().
```

3) *HeapDeleteNode algorithm*

The HeapDeleteNode algorithm for removing a node from the IHSOM heap tree is as follows:

```
    $Index = FindIndex(H, value)$ . // Find the index of the value to delete;
    Set $Count = heap_{size}$ .
    Replace the deleted node with the right most node on the lowest level of the binary heap tree; $H[Index] = H[Count]$.
    Set $heap_{size} = Count - 1$.
    Call MaxHeapify().
```

4) *HeapInsertNode algorithm*

To add a node to the IHSOM tree, the following algorithm is used:

```
    Set $Count = heap_{size}$ .
    Add the new node to the right most empty location at the bottom level of the heap tree:
     $H[Count + 1] = Node_{new}$.
    Set $heap_{size} = Count + 1$.
     Call MaxHeapify().
```

V. Experimental Results and Analysis

In this section, the empirical results achieved by applying IHSOM to both synthetic and real-world datasets are demonstrated. IHSOM reliability and mapping quality are analyzed based on two criteria; quantization error and topographic error. In order to compare the proposed algorithm with other methods, BTASOM, LB-SOINN, and GHSOM [46] algorithms are selected, which are incremental hierarchical SOM-based methods.

A. *Artificial Datasets*

In order to evaluate IHSOM performance, an artificial dataset with 5000, 25000, and 200000 input data instances are generated. Figure 2 illustrates data distribution in the artificial dataset divided into three parts; rectangular class, oval class, and ring class.

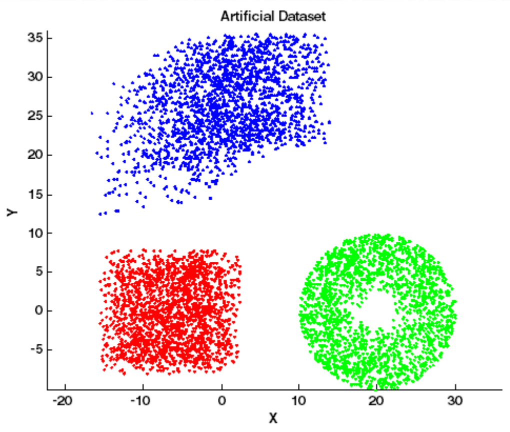


Fig. 2. The Artificial Dataset with three classes

Different size of datasets is chosen to demonstrate IHSOM ability to adapt to the growing size of input datasets appropriately. The proportion of the most similar data allocated to a node is the single user-defined parameter in the IHSOM algorithm. This parameter is empirically set as 70% to gain better results. Experiment results on the artificial dataset confirm the efficient performance of IHSOM. As shown in Table 1, the value of Quantization Error is very low, and with increasing the number of data instances decreases. In fact, increasing the volume of training data improves clustering accuracy and reduces quantization error.

The number of network nodes grows moderately as the dimension of training data augments. This memory-efficient characteristic of IHSOM empowers it to cluster large-scale data. Another remarkable property of IHSOM is its ability to preserve neighborhood topology. As the Topographic Error column is shown in Table 1, the IHSOM perfectly preserves data neighborhood topology, and the average value of topographic error is 0. The execution Time column also confirms the acceptable run time of IHSOM. Despite the fact that the suggested algorithm's execution time grows as the data dimension increases, IHSOM gives a quick clustering when the size of the training data is considered. The hierarchical tree structure of the suggested algorithm allows fast traversing of the nodes so that data can easily direct to the next node in the heap tree.

In order to evaluate IHSOM clustering reliability in the presence of noisy data, different amounts of noise samples are added to the artificial dataset (i.e., 500, 2000, and 5000 noisy data samples are added). Figure 3 depicts the fabricated dataset with noisy data (outliers). From the results presented in Table 2, it can be determined that IHSOM performs well with noisy data and that its technique for controlling outliers is effective.

[DOI: 10.52547/itrc.14.2.41]

TABLE I. EXPERIMENTAL RESULTS WERE OBTAINED BY APPLYING IHSOM TO THE ARTIFICIAL DATASET

	Artificial Dataset	Number of Instances	Number of Classes	Maximum Iteration	Number of Nodes	Quantization Error	Topographic Error	Execution Time (s)
IHSOM	1	5000	3	1000	10	0.03787	0	3.182×10
	2	25000	3	1000	13	0.02009	0	3.979×10^{2}
	3	200000	3	1000	14	0.00333	0	1.144×10^{3}

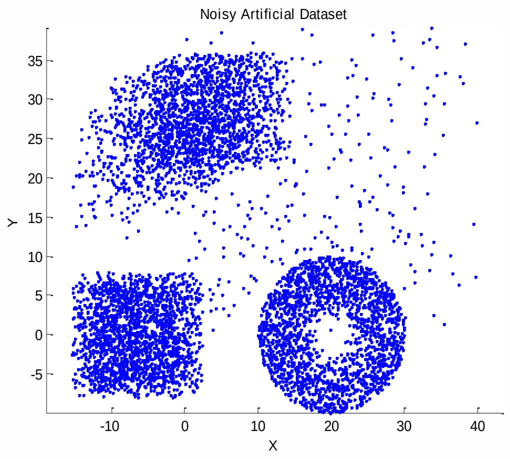
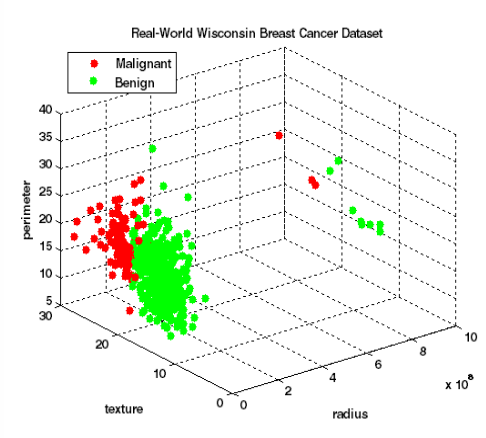


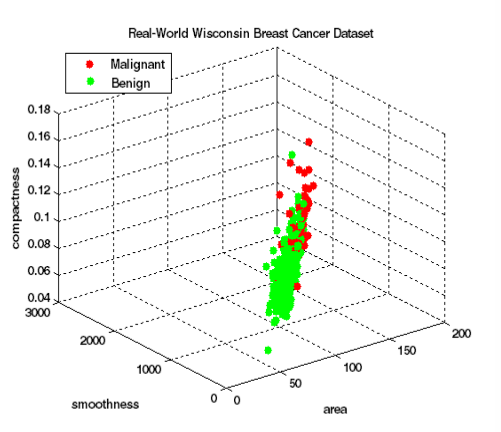
Fig. 3. The Noisy Artificial Dataset with three classes

TABLE II. EXPERIMENTAL RESULTS OBTAINED BY APPLYING IHSOM ON THE NOISY ARTIFICIAL DATASET

	Noisy Artificial Dataset	Number of Instances	Noise Percentage	Number of Classes	Maximum Iteration	Number of Nodes	Quantization Error	Topographic Error	Execution Time (s)
IHSOM	0	10000	0	3	1000	11	0.0814	0	7.156×10
	1	10000+500	5%	3	1000	11	0.12124	0	8.216×10
	2	10000+2000	20%	3	1000	14	0.20361	0	8.731×10^{2}
	3	10000+5000	50%	3	1000	16	0.6734	0	1.012×10^{3}

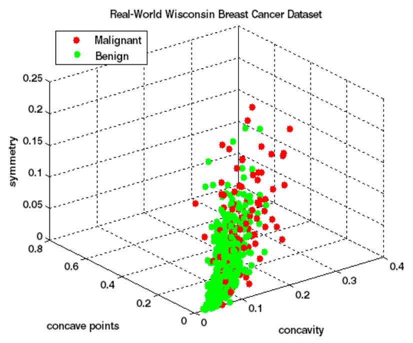


a. Wisconsin Breast Cancer Database containing two classes (Malignant-Benign) according to radius, texture, and perimeter features.



b. Wisconsin Breast Cancer Database containing two classes (Malignant-Benign) according to area, smoothness, and compactness features.

[DOI: 10.52547/itrc.14.2.41]



c. Wisconsin Breast Cancer Database containing two classes (Malignant-Benign) according to concavity, concave points, and symmetry features.

Fig. 4. Wisconsin Breast Cancer Dataset with two classes

Although increasing the number of noisy samples decreases, the quantization error is acceptable. However, due to the increase in the learning rate of the noisy data, the execution time of the IHSOM grows rapidly with the size of outliers. This phenomenon can affect the clustering task time when the outlier's sample is very high, but the execution time for a normal noise distribution is tolerable For a more comprehensive evaluation of IHSOM's performance, two real-world datasets are employed, and the proposed algorithm's efficacy is compared to that of hierarchical incremental SOM techniques such as BTASOM, LB-SOINN, and GHSOM.

A. Real-world datasets

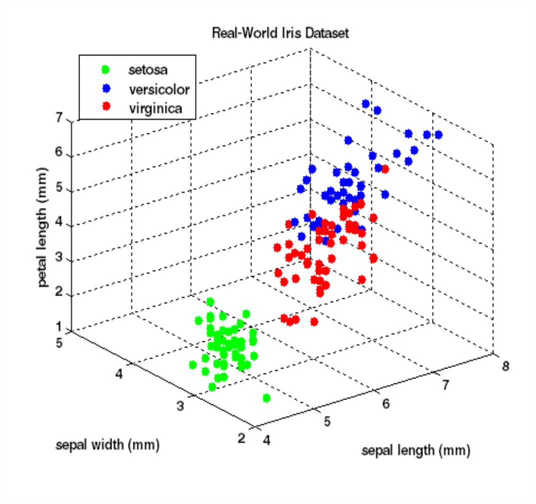
In real-world examinations, two Iris and Wisconsin breast cancer datasets are used to analyze and compare IHSOM performance versus BTASOM, LB-SOINN, and GHSOM algorithms. Madison and the Wisconsin breast cancer dataset (WDBC) were acquired from the University of Wisconsin Hospitals and Dr. William H. Wolberg, respectively [47]. WDBC has 569 instances with 32 attributes distributed in two classes: malignant and benign. Each sample in the WDBC characteristics includes an ID number, a diagnosis (M = malignant, B = benign), and ten real-valued features computed for each cell nucleus, including Radius, Texture, Perimeter, Area, Smoothness, Compactness, Concavity, Concave points, Symmetry, and Fractal dimension. These characteristics are extracted from the digital photographs of a fine needle aspirate of a breast lump. Figure 3 provides a visual demonstration of two types of breast cells, malignant and benign, according to the real-valued features. The Iris dataset is another notable multivariate data set given as an example of discriminant analysis by R. A. Fisher [48]. The Iris dataset included four iris flower characteristics: sepal length, sepal width, petal length, and petal width. This data set contains 150 samples, or 50 samples for each of the three species (classes) Setosa, Virginica, and Versicolor. Figure 4 illustrates the classes of Iris flowering trees. Table 3 shows WDBC and Iris datasets' characteristic.

TABLE III. REAL-WORLD DATASETS

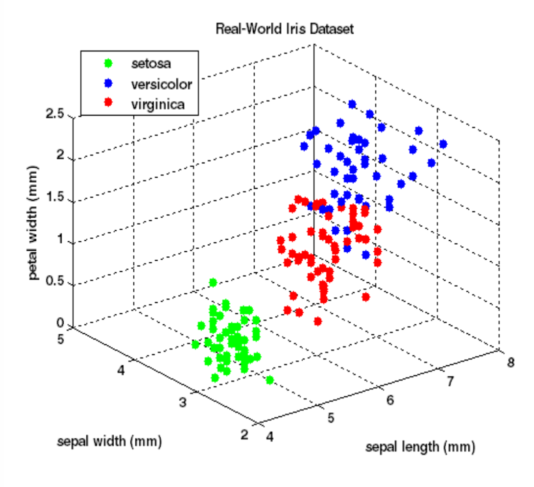
Dataset	Number of Instances	Number of Attributes	Number of Classes
Iris	150	4	3 (Setosa, Virginica, Versicolor)
Wisconsin Breast Cancer	569	32	2 (Benign, Malignant)

For IHSOM performance assessment, four hierarchical and incremental algorithms are used to compare the results obtained from deploying real-world datasets; BTASOM, LB-SOINN, and GHSOM algorithms. Each of these algorithms has some parameters that are set according to their reference papers' values in the following implementations. Table 4, illustrates the results of applying these algorithms to the Iris dataset and Wisconsin Breast Cancer dataset. From the obtained results, IHSOM proposed algorithm outreaches almost the other three clustering algorithms both in accuracy (quantization error) and execution time. The great disadvantage of BTASOM is that it cannot preserve the topology neighborhoods, and the quantization error grows drastically with the increasing dataset dimension, which can be checked with the Wisconsin breast cancer dataset with 569 samples and 32 features. However, the BTASOM execution time in the Iris dataset is better than the proposed IHSOM and other compared algorithms. In fact, the BTASOM shows faster clustering performance in lower dimensions due to its binary hierarchical structure, but its performance is not necessarily accurate. LB-SOINN and GHSOM algorithms both show better results than BTASOM in the quantization error, but both are defeated in competition with IHSOM clustering speed and accuracy. Due to the k-means approach that LB-SOINN uses to perform a local search in the input space and the hierarchical multi-layered structure of GHSOM, in which each layer is composed of several independent SOMs, they both have lengthy execution times despite their excellent quantization error performance.

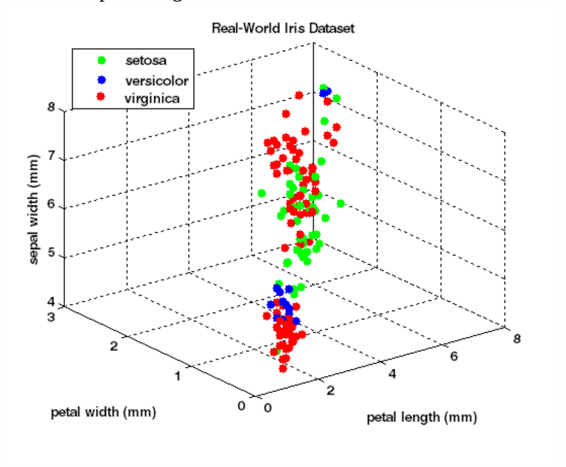
[DOI: 10.52547/itrc.14.2.41]



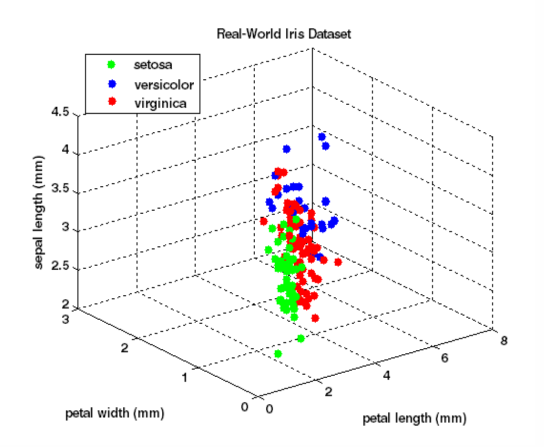
a. Iris Dataset including three classes (Setosa, Versicolor, and Virginica) according to sepal length, sepal width, and petal length.



b. Iris Dataset including three classes (Setosa, Versicolor, and Virginica) according to sepal length, sepal width, and petal width.



c. Iris Dataset including three classes (Setosa, Versicolor, and Virginica) according to petal length, petal width, and sepal width.



d. Iris Dataset including three classes (Setosa, Versicolor, and Virginica) according to petal length, petal width, and sepal length.

Fig. 5. Iris Dataset with three classes

TABLE IV. COMPARED RESULTS OBTAINED BY IHSOM

Datasets	Clustering Methods	Quantization Error	Topographic Error	Execution Time (s)
Iris	IHSOM	0.04093	0	1.5228
	BTASOM	0.5877	------	1.0661
	LB-SOINN	0.1018	0.0538	24.7617
	GHSOM	0.8092	0.0007	17.0159
Wisconsin Breast Cancer	IHSOM	1.2380	0	2.6349
	BTASOM	20.8091	-----	26.3878
	LB-SOINN	5.1641	0.1733	121.7058
	GHSOM	7.9160	0.0013	53.4190

As previously mentioned, both BTASOM and LB-SOINN algorithm structure depends on the sequence of input data meaning that if the sequence of the same training data changes during one run, the entire tree structure will differ. Since the IHSOM algorithm is a batch learning algorithm, it resolves the problem mentioned earlier and demonstrates a robust performance on various datasets with different sizes.

Furthermore, the binary heap tree structure can also preserve the input space topology ideally in output space. The Topographic Error results shown in Table 4 verify this allegation. Another remarkable property of the IHSOM algorithm is that, unlike BTASOM, LB-SOINN, and GHSOM, it has a few (just one) parameters to adjust by a user. This makes IHSOM more desirable for unsupervised clustering tasks. In

[DOI: 10.52547/itrc.14.2.41]

other words, to gain a better result from BTASOM, LB-SOINN, and GHSOM algorithm, there are too many parameters that need to be adjusted empirically.

According to the results provided in Table 1, Table 2, and Table 4, the IHSOM algorithm has an excellent capability for unsupervised clustering of a large amount of data. The proposed algorithm demonstrates its efficiency concerning BTASOM, LB-SOINN, and GHSOM, which are incremental SOM-based algorithms like IHSOM.

VI. Conclusion

This paper proposes an Incremental Heap Self-Organizing Algorithm (IHSOM) for large-scale data clustering. IHSOM is an adaptive hierarchical SOM algorithm with a binary heap tree structure that overcomes the disadvantages of the previous methods like BTASOM, SOINN, and its modified versions such as the LB-SOINN algorithm. IHSOM proposed algorithm preserves the topology of input data by its heap structure and efficiently handles outlier data by forwarding them down to the tree’s leaf nodes using a probability density function as a threshold for assigning more similar data to a cluster node. The proposed algorithm pruning and growing nodes mechanisms make it robust to noises and more accurate in clustering tasks as well as memory efficient. The heap tree structure of the algorithm speeds up nodes traversing and restructuring after adding or deleting nodes. Furthermore, IHSOM has one user-defined parameter, making it a powerful unsupervised method for clustering tasks, and due to its properties, it can be used for clustering “Big Data”. For future work, the capability of IHSOM for clustering big data will be tested.

VII. References

[1] X. Cui, P. Zhu, X. Yang, K. Li, and C. Ji, "Optimized big data K-means clustering using MapReduce," *The Journal of Supercomputing,* vol. 70, no. 3, pp. 1249-1259, 2014.

[2] R. M. Alguliyev, R. M. Aliguliyev, and L. V. Sukhostat, "Parallel batch k-means for Big data clustering," *Computers & Industrial Engineering,* vol. 152, p. 107023, 2021.

[3] Y. Zhang and Y.-M. Cheung, "Discretizing numerical attributes in decision tree for big data analysis," in *2014 IEEE International Conference on Data Mining Workshop*, 2014: IEEE, pp. 1150-1157.

[4] S. K. Punia, M. Kumar, T. Stephan, G. G. Deverajan, and R. Patan, "Performance analysis of machine learning algorithms for big data classification: Ml and ai-based algorithms for big data analysis," *International Journal of E-Health and Medical Communications (IJEHMC),* vol. 12, no. 4, pp. 60-75, 2021.

[5] G. Li, Z. Liu, J. Lu, H. Zhou, and L. Sun, "Big data-oriented wheel position and geometry calculation for cutting tool groove manufacturing based on AI algorithms," *The International Journal of Advanced Manufacturing Technology,* pp. 1-12, 2022.

[6] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," *Physical review letters,* vol. 113, no. 13, p. 130503, 2014.

[7] [7] M. Tanveer, T. Rajani, R. Rastogi, Y. Shao, and M. Ganaie, "Comprehensive review on twin support vector machines," *Annals of Operations Research,* pp. 1-46, 2022.

[8] S. Lu, Y. Chen, X. Zhu, Z. Wang, Y. Ou, and Y. Xie, "Exploring Support Vector Machines for Big Data Analyses," in *2021 4th International Conference on Computer Science and Software Engineering (CSSE 2021)*, 2021, pp. 31-37.

[9] G. Teles, J. J. Rodrigues, R. A. Rabêlo, and S. A. Kozlov, "Comparative study of support vector machines and random forests machine learning algorithms on credit operation," *Software: Practice and Experience,* vol. 51, no. 12, pp. 2492-2500, 2021.

[10] V. D. Katkar and S. V. Kulkarni, "A novel parallel implementation of Naive Bayesian classifier for Big Data," in *2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE)*, 2013: IEEE, pp. 847-852.

[11] L. Wang, X. Zhang, K. Li, and S. Zhang, "Semi-supervised learning for k-dependence Bayesian classifiers," *Applied Intelligence,* vol. 52, no. 4, pp. 3604-3622, 2022.

[12] R. Rahmadi and R. A. Rajagede, "Analisis Sentimen Politik Berdasarkan Big Data dari Media Sosial Youtube: Sebuah Tinjauan Literatur," *AUTOMATA,* vol. 2, no. 1, 2021.

[13] B. Liang and J. Austin, "A neural network for mining large volumes of time series data," in *2005 IEEE International Conference on Industrial Technology*, 2005: IEEE, pp. 688-693.

[14] D. Aberdeen, J. Baxter, and R. Edwards, "92¢/mflops/s, ultra-large-scale neural-network training on a piii cluster," in *SC'00: Proceedings of the 2000 ACM/IEEE Conference on Supercomputing*, 2000: IEEE, pp. 44-44.

[15] W. Höpken, T. Eberle, M. Fuchs, and M. Lexhagen, "Improving tourist arrival prediction: a big data and artificial neural network approach," *Journal of Travel Research,* vol. 60, no. 5, pp. 998-1017, 2021.

[16] I. K. Nti, J. A. Quarcoo, J. Aning, and G. K. Fosu, "A mini-review of machine learning in big data analytics: Applications, challenges, and prospects," *Big Data Mining and Analytics,* vol. 5, no. 2, pp. 81-97, 2022.

[17] M. Dabbu, L. Karuppusamy, D. Pulugu, S. R. Vootla, and V. R. Reddyvari, "Water atom search algorithm-based deep recurrent neural network for the big data classification based on spark architecture," *International Journal of Machine Learning and Cybernetics,* pp. 1-16, 2022.

[18] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics,* vol. 43, no. 1, pp. 59-69, 1982.

[19] B. Sara and A. Otman, "New Learning Approach for Unsupervised Neural Networks Model with Application to Agriculture Field," *International Journal of Advanced Computer Science and Applications,* vol. 11, no. 5, 2020.

[20] S. Jovanović and H. Hikawa, "A Survey of Hardware Self-Organizing Maps," *IEEE Transactions on Neural Networks and Learning Systems,* 2022.

[21] G. A. Angulo-Saucedo, J. X. Leon-Medina, W. A. Pineda-Muñoz, M. A. Torres-Arredondo, and D. A. Tibaduiza, "Damage Classification Using Supervised Self-Organizing Maps in Structural Health Monitoring," *Sensors,* vol. 22, no. 4, p. 1484, 2022.

[22] J. Herrero, A. Valencia, and J. Dopazo, "A hierarchical unsupervised growing neural network for clustering gene expression patterns," *Bioinformatics,* vol. 17, no. 2, pp. 126-136, 2001.

[23] M. M. Campos and G. A. Carpenter, "S-TREE: self-organizing trees for data clustering and online vector quantization," *Neural Networks,* vol. 14, no. 4-5, pp. 505-525, 2001.

[24] A. Denton, Q. Ding, W. Perrizo, and Q. Ding, "Efficient Hierarchical Clustering of Large Data Sets Using P-trees," in *CAINE*, 2002, pp. 138-141.

[25] S. Furao, T. Ogura, and O. Hasegawa, "An enhanced self-organizing incremental neural network for online unsupervised learning," *Neural Networks,* vol. 20, no. 8, pp. 893-903, 2007.

[26] 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.

[27] F. Carpine, C. Mazzariello, and C. Sansone, "Online IRC botnet detection using a SOINN classifier," in *2013 IEEE International Conference on Communications Workshops (ICC)*, 2013: IEEE, pp. 1351-1356.

[28] N. Masuyama, I. Tsubota, Y. Nojima, and H. Ishibuchi, "Class-wise Classifier Design Capable of Continual Learning using Adaptive Resonance Theory-based Topological Clustering," *arXiv preprint arXiv:2203.09879,* 2022.

[29] H. Shah-Hosseini, "Binary tree time adaptive self-organizing map," *Neurocomputing,* vol. 74, no. 11, pp. 1823-1839, 2011.

[30] B. Liu, E. Blasch, Y. Chen, D. Shen, and G. Chen, "Scalable sentiment classification for big data analysis using naive bayes

classifier," in *2013 IEEE international conference on big data*, 2013: IEEE, pp. 99-104.
[31] R. Mall, R. Langone, and J. A. Suykens, "Self-tuned kernel spectral clustering for large scale networks," in *2013 IEEE International Conference on Big Data*, 2013: IEEE, pp. 385-393.
[32] R. Mall, R. Langone, and J. A. Suykens, "Ranking Overlap and Outlier Points in Data using Soft Kernel Spectral Clustering."
[33] A. Toor, "An Advanced Clustering Algorithm (ACA) for Clustering Large Data Set to Achieve High Dimensionality," *Global Journal of Computer Science and Technology*, 2014-05-15 2014. [Online]. Available: https://computerresearch.org/index.php/computer/article/view/77.
[34] R. Chouhan and A. Chauhan, "An Ameliorated Partitioning Clustering Algorithm for Large Data Sets," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 3, no. 7, 2014.
[35] P. Perner, "Mining sparse and big data by case-based reasoning," *Procedia Computer Science*, vol. 35, pp. 19-33, 2014.
[36] Z. Nenova and J. Shang, "Chronic disease progression prediction: Leveraging case - based reasoning and big data analytics," *Production and Operations Management*, vol. 31, no. 1, pp. 259-280, 2022.
[37] [37] R. M. Esteves, T. Hacker, and C. Rong, "A new approach for accurate distributed cluster analysis for Big Data: competitive K-Means," *International Journal of Big Data Intelligence 5*, vol. 1, no. 1-2, pp. 50-64, 2014.
[38] Z. Shi, Y. Xue, Y. Wen, and G. Cai, "Efficient class incremental learning for multi-label classification of evolving data streams," in *2014 international joint conference on neural networks (IJCNN)*, 2014: IEEE, pp. 2093-2099.
[39] H. Zhang, X. Xiao, and O. Hasegawa, "A load-balancing self-organizing incremental neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 6, pp. 1096-1105, 2013.
[40] T. Kohonen, *Self-organization and associative memory*. Springer Science & Business Media, 2012.
[41] M. A. Weiss, "Data Structures and Algorithm Analysis in C++, 2007," *Data Structures and Algorithm Analysis in Java*, 2007.
[42] R. L. Villacarlos, J. M. Samaniego, A. J. Jacildo, and M. A. A. D. Clariño, "A Tale of Two Trees: New Analysis for AVL Tree and Binary Heap," *arXiv preprint arXiv:2010.04752*, 2020.
[43] S. Bae, *JavaScript Data Structures and Algorithms: An Introduction to Understanding and Implementing Core Data Structure and Algorithm Fundamentals*. Apress, 2019.
[44] E. A. Uriarte and F. D. Martín, "Topology preservation in SOM," *International journal of applied mathematics and computer sciences*, vol. 1, no. 1, pp. 19-22, 2005.
[45] H. Matsushita and Y. Nishio, "Batch-learning self-organizing map with false-neighbor degree between neurons," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008: IEEE, pp. 2259-2266.
[46] A. Rauber, D. Merkl, and M. Dittenbach, "The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data," *IEEE Transactions on Neural Networks*, vol. 13, no. 6, pp. 1331-1341, 2002.
[47] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology," *Proceedings of the national academy of sciences*, vol. 87, no. 23, pp. 9193-9196, 1990.
[48] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179-188, 1936.

Mehdi Fasanghari received his Ph.D. in Industrial Engineering at the University of Tehran. He received his B.Sc. degree from Tarbiat Modarres University. National Broadband Network, 5th generation of Mobile Network, Large Scale System Development, Soft Computing, and Advanced Multi-Criteria Decision Analysis are his current research interests.

Hamideh Sadat Cheraghchi received her Ph.D. in the Department of Computer Engineering at Shahid Beheshti University (2018). She received the B.Sc degree from Azad University, South Tehran branch (2006); M.Sc. from Azad University, Qazvin branch (2009). Her current research interests focuses on Data Mining in Social Network, and Soft Methods and Health Care Systems.

Helena Bahrami is a Ph.D. graduate from AUT Knowledge Engineering & Discovery Research Institute (KEDRI). She Obtained an M.Sc. degree in Artificial Intelligence from Qazvin Azad University, Iran in 2011 and B.Sc. degree in Computer Software Engineering from Iran in 2006. Her current research interests are Artificial Intelligence, Machine Learning, Deep Learning, Quantum Computing.

[DOI: 10.52547/itrc.14.2.41]