Granular Computing (2023) 8:1051–1065
https://doi.org/10.1007/s41066-023-00370-5

ORIGINAL PAPER

A supervised method to enhance distance-based neural network clustering performance by discovering perfect representative neurons

Qiang Fu[1] · Yuefeng Li[1] · Mubarak Albathan[2]

Received: 27 December 2022 / Accepted: 19 February 2023 / Published online: 5 March 2023

Abstract
Distance-based neural network clustering requires the intrinsic assumption that a particular neuron in the network represents a cluster centroid. However, not all these neurons can perfectly represent the training data; these neurons can only represent part of the training samples. This paper proposes an effective training data splitting method (TDSM) to find perfect representative neurons and improve the clustering results in a distance-based neutral network without changing the original network's internal algorithm or the training data quality. The method allows a network with N neurons to be enlarged to a new network with $m \times N$ neurons. These neurons represent m subnetworks, and each subnetwork perfectly represents a part of the training set, where the clustering qualification indicators (the purity, normalized mutual information, and adjusted rand index measures) all equal 1. The results are statistically validated with a t test, and we demonstrate that the TDSM performs better than the original clustering paradigm on some real datasets.

Keywords Distance-based neural network clustering · Self-organizing map · Iterative splitting training data · Representative neurons

1 Introduction

As one of the most critical tasks of unsupervised learning, clustering divides an entire dataset into several groups based on the similarity of the data, and a measure of the similarity between two patterns is essential to most clustering procedures. It is common to use a distance measure for continuous and categorical feature clustering (Chiu et al. 2001). Several distance metrics, including Euclidean (the most frequently used), correlation, direction cosine, Minkowski (Mailagaha Kumbure and Luukka 2022) and block distance (Miljković 2017; Rawat et al. 2011) metric, can be used. In distance-based neural network (NN) clustering, a cluster center is usually represented by a trained neuron and can be called "center of a class" or an agglomeration of the data point (Litinskii and Romanov 2006), and input data can be clustered according to its nearest neuron. How to improve the clustering quality has always been a hot topic. One weakness of distance-based NN clustering is that the data grouped into the same neuron or cluster do not belong to the same class; additionally, the generated cluster number in a distance-based model is sensitive to the NN's neuron number. It is well known that one of the most challenging problems in data clustering is determining the number of clusters automatically. From the view of representation learning, if network B can represent the data samples better than network A, we say that B has a "better representation ability" than A. Neurons that can adequately represent specific input patterns are occasionally lacking, and this is a recurring problem in distance neural network clustering; we call it the "lack of representation ability problem." Based on Thrun (2021), biased clustering occurs due to the difference between given cluster structures and the reproduced structures. For

✉ Qiang Fu
q20.fu@hdr.qut.edu.au

Yuefeng Li
y2.li@qut.edu.au

Mubarak Albathan
mmalbathan@imamu.edu.sa

[1] School of Computer Science, Queensland University of Technology, Brisbane, Australia

[2] College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University, Riyadh, Saudi Arabia

instance, the self-organizing map (SOM), commonly well known as the Kohonen network (Kohonen 1982), is a powerful unsupervised, competitive learning method for the visualization and analysis of high-dimensional data. It is a single-layer network, and the units are distributed along m-dimensional grids (most applications use two-dimensional rectangular grids). As a hard clustering algorithm (Jain et al. 1999), it allocates each input data point to a single neuron. SOM can preserve the topological structure of data and group training data based on the distance between data and neurons. In technical contexts, SOM is utilized in the fashion of neighborhood preserving vector quantizers and projects data from the input space to a position in some output space (the map's weight matrix).

However, the algorithm is derived from heuristic ideas rather than statistical principles (Bishop et al. 1996), leading to several significant limitations. First, the cluster number is restricted by the number of neurons. Each neuron represents a cluster centroid; frequently, not all data belong to the same class, and there are always some data that have class labels that are different than the labels of the majority of the data in each cluster. SOM lacks extra neurons to represent these data due to a predefined neuron number. Second, due to the unclear clustering boundaries of nodes in SOM, the clustering results are inconsistent each time due to the influence of the initial value of neurons and the data input order (Deng and Mei 2009). After training, the trained weight matrix of neurons only represents part of the input samples; we call these data "represented data." There will always be some part of the sample data that is weakly represented by the node matrix; we denote this part of data as "unrepresented data." We can manually adjust the number of neurons and train the model again. However, because it is a new round of training, the previous "unrepresented data" are changed to new "unrepresented data".

This paper presents a new method to solve the above problem. Without modifying the traditional SOM algorithm to solve the "lack of representation ability problem," we aim to find new neurons that can adequately represent "unrepresented data." Different from the typical definition, where a neuron is just a vector in the data space, we extended the concept of a neuron to be a representative neuron as follows: each neuron unit $N_i = (w_i, M_i)$, has two properties: First, it contains a synaptic weight vector $w = [n_{i_0}, n_{i_1}, \ldots, n_{i_{m-1}}]$, which has the same dimensionality as an input data ($[i_0, i_1, \ldots, i_{m-1}]$) and represents the location of the neuron in the data space. The second property is the training data samples it can represent, called its members M. Each w of a neuron unit, which is in an m-dimensional space, will change during training based on the input dataset. After training, a weight matrix W_0 (the combination of each neuron's w) represents the entire dataset (the combination of each neuron's M) in an m-dimensional data space. However, for traditional SOM, W_0's representation ability is not perfect. We try to find the solution by splitting training data with the help of external validation and finding the "unrepresented data," which will be used for extra training with the SOM algorithm. We obtain a new topology representation by combining the newly generated weight matrices with the initial one. The newly combined weight matrix has better clustering quality. In summary, our method's main task is splitting the training samples to discover "missing" neurons whose members can be perfectly represented without modifying the original network's algorithmic structure and internal operations, which is delineated in the latter part of the paper.

The remainder of the paper is organized as follows: Related studies on SOM are described in Sect. 2. Section 3 describes SOM, the proposed TDSM, and its algorithm. Section 4 presents several main experiments based on real datasets and validates the TDSM. Finally, we conclude the paper in Sect. 5.

2 Related Studies

Because this research is based on SOM, this section first describes the SOM algorithm. There are two core concepts in SOM: competitive learning and adjusting the local synaptic plasticity of the neurons in learning (Kohonen and Honkela 2007). These two facets empower SOM to evolve a robust self-organization map (Bauer and Villmann 1997). When input data go through the network, the output of the clustering function is an index of the trained weight matrix (W_0), which represents the index of the cluster it belongs to; for example, if a function $F(i, W_0)$ takes a data point i as input and the neuron representation W_0 outputs the index of the winner neuron in W_0, the index is denoted by x ($x \in [0, n-1]$, where n is the total number of neurons). When the function is applied to the whole dataset, it is $F(I, W_0)$, and the output is $[x_0, x_1, \ldots x_i]$ (i is the number of neurons minus 1, and I is the input dataset).

SOM defines the similarity based on the distance and usually applies a complete learning algorithm, where "winner-takes-all" circuits play a central role in competitive learning networks (Kaski and Kohonen 1994). The network is initialized with a weight matrix $W_{initial}$ in a simple SOM algorithm. Suppose that the input data is an m-dimensional vector; then, $W_{initial}$ is an $N * m$ matrix, and N is the number of neurons. Each element's initial value in $W_{initial}$ is randomized and drawn from a Gaussian distribution. The winning neuron is the input data's best match unit (BMU), representing the neuron with the minimum

distance from the input data. To show the basic principle, SOM establishes a simple model: by utilizing the discriminant function $F(I, W_0)$ to compare the distances among the input data and each unit in the network with the formula shown in Eq. 1:

$$d_{ij} = \| I_i - N_j \|_2, \tag{1}$$

where d_{ij} is the Euclidean distance between input vector I_i and one neuron vector N_j in a weight matrix W. SOM can find the winning unit, whose position is replaced by the input data so that BMU is the perfect representation of the input data. Other non-winning neurons remain static or move toward the winner neuron. Figure 1 (a) shows two input data points and an initial SOM network. Figure 1 (b) shows that after one comparison, the adjusted topology represents the training data (blue and red points), and the output is $F(I_1, W_{initial}) = 0$ and $F(I_2, W_{initial}) = 8$. In the figure, I_1 is represented by the red point and is assigned to Cluster 0, and I_2 is represented by the blue point and is assigned to Cluster 8.

The second important part of SOM is its neighborhood function, derived from the bionics concept of synaptic plasticity (Kohonen and Honkela 2007). Like a chemical effect, the winner neuron also impacts the nodes nearby, which represent the neighborhood. Based on the description of El Atik et al. (2021), neighborhood systems are used to approximate graphs as finite topological structures. With the help of neighborhood function, neurons in SOM can influence or organize other neurons within the same network, giving the network better stability (Miljković 2017). There is a parameter called the BMU distance, which denotes the range affected by the BMU. If the BMU distance = 1, Fig. 1 (b) is similar to Fig. 1 (c); that is, points 1 and 3 are winner 0's neighbors, so they will also move toward the input data, but they are not winners, and cannot "take all"; they can only move a certain distance toward the input data. There are many different neighborhood functions; the exponential decay formula derived from sklearn-som 1.1.0 (Smith 2021) is defined like this:

$$N(u, v) = e^{-\frac{d_{uv}}{\sigma^2}}, \tag{2}$$

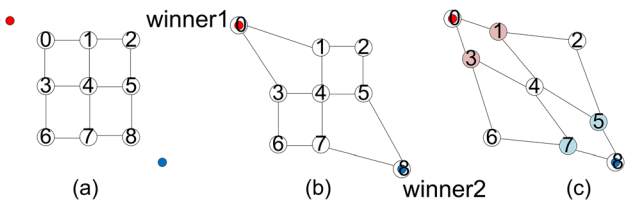

Fig. 1 In the self-organization process of SOM, given input data, the winner directly moves to the position of the input data, (a) is the original map; (b) is the map after training, and the red and blue points represent the training samples; and (c) is the map after using the neighborhood function

where v is the index of neuron node in the network, u is the BMU node, d_{uv} is the distance between u and v, and σ is an optional parameter for the magnitude of change in each weight; higher values mean more aggressive updates to the weights.

Based on the winner-take-all concept and the local neighborhood function, SOM updates $W_{initial}$ in each training iteration with the following formula:

$$W_{i+1} = W_i + N(u, v) \times l \times (i - W_i). \tag{3}$$

In Eq. 3, i is the current iteration index, and $N(u, v)$ is the neighbor function, l means the learning rate which shows the step size for updating the SOM weights, and i is an input data vector. The learning rate and neighborhood radius decrease with each iteration toward zero to guarantee convergence and stability (Lobo 2009). With a certain iteration, after training, $W_{initial}$ is updated to W_0. It is worth noting that spatial locations of the neurons (W_0) in the SOM lattice are topologically related to the features of the input space, which provide a good approximation to the input samples (Miljković 2017). After training, the updated weight matrix W_0 represents the training samples and can be used to predict cluster labels for the test data.

To the best of our knowledge, on the premise of not changing the quality of the training data, e.g., noisy dataset removal, subspace selection, and projected clustering (Bassani and Araujo 2014; Bassani and Araújo 2012; Braga and Bassani 2018), most researchers focus on improving the neural network structure itself to obtain better clustering results. Nevertheless, they seldom start from a data viewpoint and find a way to eliminate the "unrepresented data" in each cluster. Specifically, there are two ways to optimize the network structure: predefining a new structure externally and adaptively adjusting the structure internally during the training phase.

For the external method, as noted in Lampinen and Oja (1992), multiple-layer SOM clusters match the desired classes better than direct SOM clusters. Cascaded SOM (Hua and Mo 2020) utilizes ensemble learning to obtain a more robust network structure by training the same samples multiple times and obtaining multiple weight vector matrices, which are used as the input for further layer SOM training; then a final decision is generated by learning the responses of different clusters.

For internal optimization, unlike the traditional SOM algorithm, a modified SOFM (Ghosh et al. 2009) abandons the competitive learning process to decide which neuron to update. Instead, it calculates the dot product of input and weight vectors of the output neuron; when the dot production reaches a predefined threshold t, the neurons are allowed to be adjusted based on the traditional SOM neighbor function. As a result, different inputs are given to

different output neurons, and weight updating is performed based on the considered threshold. Rating-weighted SOM (Park 2022) updates the formula for choosing the best match unit in the network by adding a bias function of the relevance scores generated from the training set. To address online learning environments, a dynamic self-organizing map (DSOM) (Rougier and Boniface 2011) can be used to remove the time dependency by replacing the initial learning rate and neighbor function, which reflects two main ideas: first, if a neuron is close enough to the data, then it can represent the data, and other neurons do not need to learn. Second, if no neuron is close enough to the data, any neuron learns the data according to its own distance. The growing hierarchical self-organizing map (GHSOM) (Rauber et al. 2002) introduces the concept of quantization error (QE), which represents the sum of the distances between each cluster center and the data. It assumes that a better-trained map has a lower QE value or mean quantization error (MQE). To decrease the value of MQE of the whole network, the GHSOM first finds a neuron called an "error unit" with the maximum QE value. Then, a row or a column of neurons is inserted between the error unit and its most dissimilar neighbor neuron (the neuron that is farthest from the error unit). GHSOM continues the above process until all the data are in one cluster or the whole network MQE, and each neuron's MQE is below a threshold. Furthermore, an adaptive moving self-organizing map (AMSOM) algorithm (Spanakis and Weiss 2016) creates a more flexible structure to dynamically alter the topological position of the neurons by not just adding neurons but removing redundant neurons or adding other neurons; consequently, it can adjust the connections between neurons to enhance the topological properties of the network.

There are also some methods that combine different SOM architectures; for example, a hybrid approach was developed by fusing the concepts of SOM and GSOM for solving the tweet-summarization task. SOM helps in reducing the dataset size in terms of the number of tweets, while, GSOM helps in generating the summary.

Although the data in the networks above become more similar to their cluster centroid or the neuron topological structures become more similar to the patterns, there is no evidence to prove that the "unrepresented data" is sufficiently reduced and misclassification still exists in the clustered training data.

3 Proposed method

In brief, our method extends the concept of neurons in SOM; that is, each neuron has a pair of properties: location w (the internal property) and members M (the external property), which denote the training data the neuron represents. A neuron's extent (the external property) can be changed or evolved if we apply it in a supervised environment. After the change or evolution, each neuron will have a perfect representation ability to its members. We call such neurons perfect representative neurons.

We have to use validation measures to evaluate the performance of the clustering algorithm. There are two types of validation methods: internal and external validation (Sripada and Rao 2011). As external validation measures, the purity, normalized mutual information (NMI), and adjusted rand index (ARI) measures are used extensively to validate the accuracy of a clustering technique. Better clustering models will have a higher score. In a group of clustered data, if we can reduce the incorrectly clustered data ("unrepresented data"), then we can obtain a higher cluster validation score. SOM is an unsupervised clustering method, and we use external validation to distinguish the "represented data" and "unrepresented data" in a generated cluster. Once we can find extra neurons representing the "unrepresented data", the new combined neuron map's clustering quality will be improved significantly.

3.1 TDSM

The proposed method is based on the theory that the weight matrix W represents the data samples in n-dimensional space (n is the number of neurons in the network). Let I_u denote the "unrepresented data," and I_r is the "represented data" (correctly clustered data, where data with the same class label are grouped based on the distance with cluster centroids), $I = I_u \cup I_r$, where I is the training data. $F(I_r, W)$ returns the predicted cluster labels L_p of I_r, ($L_p = [x_1, x_2, \ldots, x_j]$, j denotes the sample number in I_r) and the cluster quality or precision indicators (purity, NMI,ARI) all equal 1. After eliminating the "unrepresented data," W_0, which initially represents the whole data samples but not perfectly, now perfectly represents the I_r in the feature space. Property 1 (location w) of each neuron in W_0 remains the same, but property 2 (the members in each neuron) is changed. If we can find W_1 that represents the "unrepresented data" I_u, then the conjunction of W_0 and W_1 will theoretically perform better in representing I than W_0 does, as shown in Fig. 2. The new W' matrix is denoted as (Equation: 4):

$$W' = \begin{bmatrix} W_0 \\ W_1 \end{bmatrix}. \tag{4}$$

A 3D illustration of the combination is shown in Fig. 3. To obatin W_1, we use the same neural network model to train the "unrepresented data." The generated weight

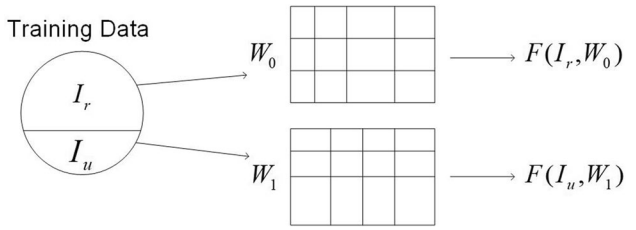

Fig. 2 Ideally, for the training data, BMU of I_r should come from W_0 and BMU of I_u should come from W_1; I_r uses $F(I_r, W_0)$ to obtain the cluster, and I_u uses $F(I_u, W_1)$

matrix W is denoted by W_1. Theoretically, $P(W_1, I_u, L_{t^u}) \geq P(W_0, I_u, L_{t^u})$, ($L_{t^u}$ is the ground-truth class label for I_u); as W_1 is generated by training I_u exclusively, it should have better representation ability than W_0 (W_0 is the perfect representation of I_r and the incorrect representation of I_u). Furthermore, we can apply the same data splitting rules on I_u to find I_u's "represented data" and "unrepresented data." In the extreme case, eventually, there will be no "unrepresented data" in the final splitting stage, e.g., Fig. 4, and the whole training data can be split into a multiple-layer structure. Each layer is a part of the training data, and the training data can be perfectly represented by an exclusive SOM network.

Ideally, after validating the test data, we should obtain the following result: $P(W', I_{train}, L_{train}) \geq P(W_0, I_{train}, L_{train})$ and $P(W', I_{train}, L_{train}) = 1$, (function P denotes the cluster performance indicator (purity, NMI,ARI) score, I_{train} denotes the training data, and L_{train} is the ground-truth class labels in training data); when the training data have sufficient ability to represent the feature space, in test data validation, we should also obtain $P(W', I_{test}, L_{test}) \geq P(W_0, I_{test}, L_{test})$ (I_{test} denotes the test data, and L_{test} is the ground-truth class labels in the test data). Figure 5 illustrates the process of the proposed method, and Fig. 6 is an example of the proposed approach; the external validation operation is the operation that finds "unrepresented data" with the help of a class label in training data.

After each split, the number of neurons in the network doubles. Consequently, the cluster number increases to $(m+1) * N$ (m denotes the split time, and N denotes the number of initial clusters). Namely, after splitting, the

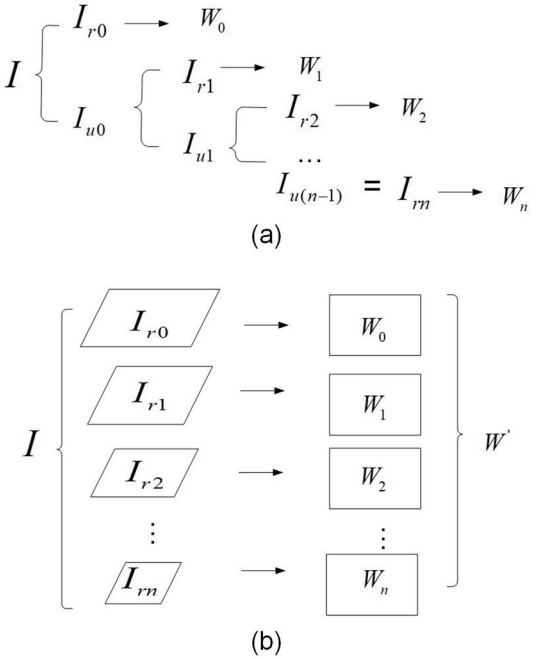

Fig. 4 (**a**) is a process of continuously splitting the "unrepresented data" until there is no "unrepresented data," e.g. $I_{u(n-1)} = I_{rn}$. In Figure (**b**), I is the combination of all the "represented data" and W' is a combination of all the "represented data"'s representation, which is an ideal representation of I; the arrow denotes the "represented by" relationship, and the braces in Figure (**a**) denote the "split" operation

original network is divided into $m+1$ subnetworks; Each subnetwork has N neurons and can perfectly represent part of the training samples; however, not all the neurons have data to represent, and they are called "empty neurons" or "dead neurons" in Ghaseminezhad and Karami (2011). Finally, for given test data, we need to determine which sub-network should be used for its representation. To achieve this goal, we find the test data's nearest neighbor in the training vectors and use the sub-network that the nearest neighbor belonged to as the input weight matrix to predict the test data's cluster in SOM.

3.2 Algorithm

To find the "unrepresented data" in each iteration, one of the most important processes is to obtain the mapping of

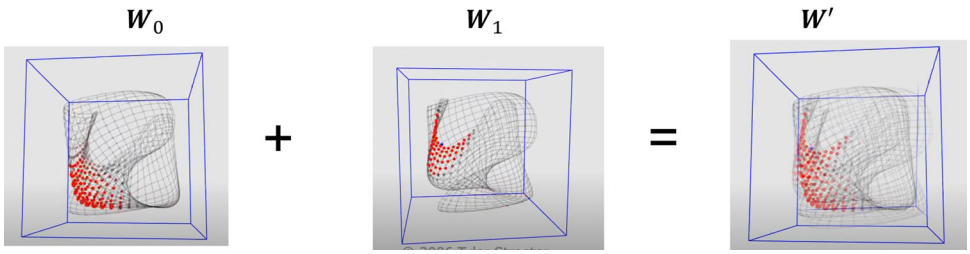

Fig. 3 Combination of W_0 and W_1 in a 3D space. Screenshot images are from (Tyler, 2006)

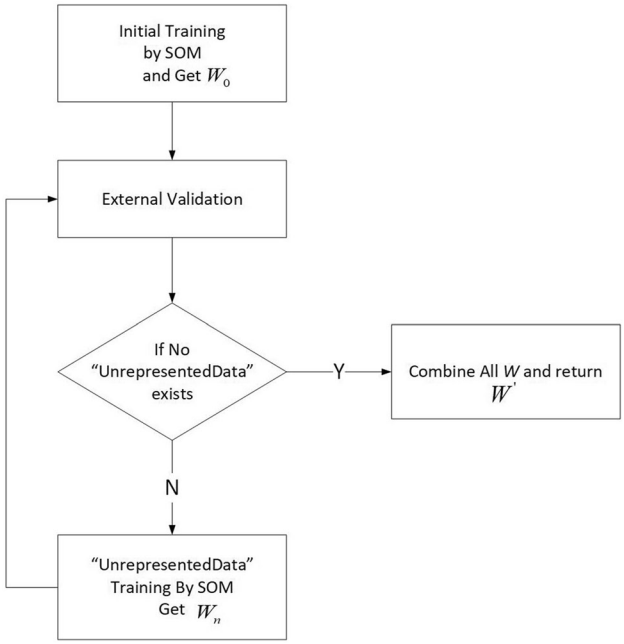

Fig. 5 Flow chart of the TDSM

predicted cluster labels and ground-truth class labels; the operation is represented by $M(L_{t_i}, L_{p_i})$, e.g., Algorithm 2. In the above algorithm, $l_i \in [0, n-1]$, where n denotes the number of clusters that the subnetwork can generate, and l_i begins at 0. As each W is trained independently, the mapping of each W should be calculated separately, as shown in Fig. 7. Algorithm 1 shows the whole process of the TDSM. As long as the combined weight matrix W' is generated, given a test data, the first step is to find its nearest training data, which is a traditional nearest neighbor problem (Andoni 2009). As one of the fastest algorithms for the nearest neighbor query, the K-d tree (Chen et al.2019)algorithm is applied to solve the above problem. Finally, given a sample (i_{test}) from test data, we detect its nearest training data $i_{nearest}$ and the W can represent $i_{nearest}$; then, i_{test} is predicted with the W.

The notations used in the algorithm are listed as follows:

(a) L_{p_i} is the set of predicted cluster labels for CurrentTrainData
(b) L_{t_i} is the set of ground-truth class labels for each element in CurrentTrainData, Mapping(W_i) is the class label and cluster label in W_i
(c) G_i is a predicted cluster (the data that a neuron can represent)
(d) $L_{t_{G_i}}$ is the set of ground-truth class labels for each element in G_i
(e) I_{e_i} is the unrepresented data for each split
(f) I_{r_i} is the represented data for each split
(g) i is the predicted cluster index in each W

Algorithm 1 TDSM

Input: training data, test data and initial SOM parameters
Output: predicted class labels of all test data

NoErrorDataExit = False
Training data with SOM and obatining W_0
$W' = W_0$
CurrentTrainData = Training data
while NoErrorDataExit!= True **do**
    Train CurrentTrainData with SOM and obtain W_i and L_{p_i} (a)
    Use $M(L_{t_i}, L_{p_i})$ to obtain I_{e_i} and Mapping(W_i) (b)
    Allocate the split index I_s to each data sample in I_{e_i}
    $W' = Combine(W', W_i)$
    CurrentTrainData = I_{e_i}
end while
for each data I_t in test data **do**
    Find the nearest neighbor in the training data and obtain the neighbor's split index m
    Predict I_t with SOM using W_m and Mapping(W_m)
end for

Algorithm 2 $M(L_t, L_p)$

Input: training data's true class labels, predicted class labels
Output: a list of class labels to show the mapping of L_t and L_p

Group each value l_i in L_p and obtain a ClusterGroup
for each group G_i in ClusterGroup **do** (c)
    Get the ground-truth class label values $L_{t_{G_i}}$ in G_i
    Obtain the most frequent value f_{t_i} in $L_{t_{G_i}}$ (d)
    Obtain the data whose class label $l \neq f_{p_i}$ in each G_i and save them in a set I_{e_i} (e)
    Obtain the data whose cluster label $l = f_{p_i}$ in each G_i and save them in a set I_{r_i} (f)
    Map f_{p_i} with i (g)
end for
return Mapping(W_i)

4 Experiment

To validate our theory, the competitor model we use is the traditional SOM algorithm. The TDSM trains the training data by the original SOM algorithm; we suppose that other

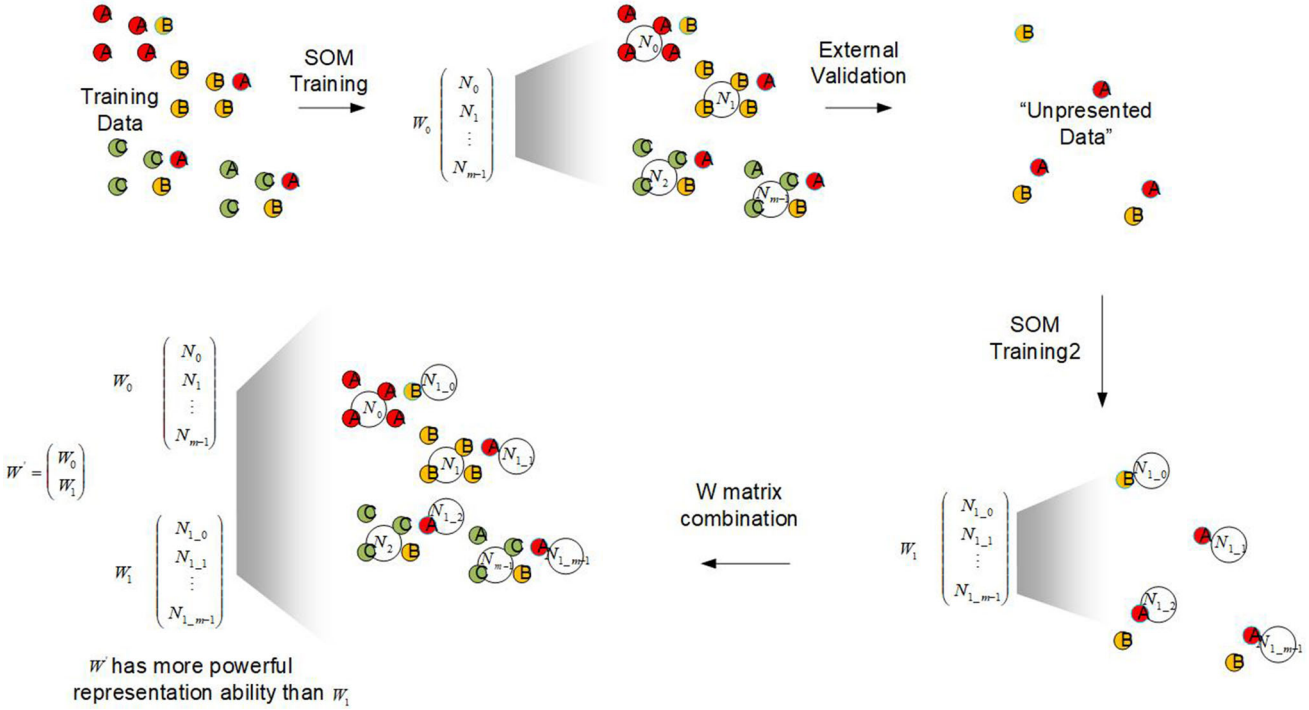

Fig. 6 Example of the TDSM with one split. *A*, *B*, and *C*, denote the ground-truth class labels, and *N* denotes the cluster centroids

conditions or parameters, such as the initial neuron number and domain knowledge utilization, are all the same. The number of features that are used in the calculation remains the same. Nine datasets obtained from the UCI machine learning repository (Dua and Graff, 2017) are used in our experiments, as described in Table 1. Data preprocessing includes dropping useless or meaningless attributes, transferring features from string to float form, and choosing the class label among the features. The last three columns in Table 1 show the preprocessing for each dataset. For the datasets that do not provide test data, we resample 30% of the total training data as test data.

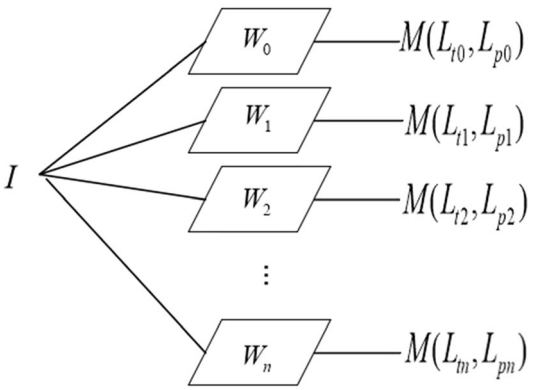

Fig. 7 When input data pass through different sub-networks, the output should use diffident clustering mapping functions

4.1 Clustering performance external validation

We evaluate the clustering results with three commonly used evaluation metrics: purity, NMI, and ARI. Purity is used extensively to test the accuracy of a clustering technique; see Eq. 5 (Jiang and Chung 2012). Better clustering models will have higher purity scores.

$$Purity(W, C, D) = \frac{1}{N}\sum_{k} \max_{j}(n_k \cap c_j), \tag{5}$$

where D is the training data samples, N is the sum of the samples in D, $W = N_1, N_2, \ldots, N_m$ denotes the set of clusters, $C = c_1, c_2, \ldots, c_j$ is the set of classes (obtained from the ground-truth class labels in the training data), n_k is cluster k, and c_j is class j.

Based on the equation above, if we can reduce the incorrectly clustered data (unrepresented data) (∘ in Cluster1, □ and x in Cluster2, and ∘ and x in Cluster3), then we can obtain a higher purity value. Nevertheless, one problem is that when the number of clusters is large, it is easy to obtain a high purity score; an extreme case is that purity is one if each data point is assigned a cluster. Thus, purity cannot be used to evaluate the clustering quality of two algorithms if the number of clusters in each algorithm is different (Forest et al. 2021). In the TDSM, the original network with n clusters will be enlarged to an $(m+1) \times n$ cluster network after m splits. To use purity as an indicator,

Table 1 Dataset description (Class label is not included in the Attribute no.) and data preprocessing

Dataset	Classes no.	Attribute no.	Samples no.	Classes label	Dropped attributes	Numberized attributes
IRIS	3	4	150	Species	Id	Species
User knowledge modeling	5	4	403	UNS	NULL	UNS
HCV data	5	12	615	Category	ID	Category,Sex
Absenteeism	28	20	740	Reason for absence	NULL	NULL
Mice protein expression	8	80	1080	class	MouseID	Genotype,Treatment class,Behavior
Estimation of obesity levels based on eating habits and physical condition	7	6	2111	NObeyesdad	NULL	CAEC,Gender,FAVC family_history_with_overweight SMOKE,CALC,MTRANS, SCC,NObeyesdad
Anuran calls (MFCCs)	10	22	7197	Species	Family,Genus RecordID	NULL
Turkiye Student Evaluation	13	32	5820	class	NULL	NULL
Crowdsourced Mapping	6	28	10546	class	NULL	NULL

we compare the TDSM with another SOM that has $(m+1)\times n$ neurons.

NMI is a variant of a common measure in information theory called mutual information (MI), which means the "amount of information" one can extract from a random variable regarding a second one, denoted by $I(A, B)$ and its formulation is in Eq. 7 (Amelio and Pizzuti 2015). NMI depends on the mutual information $I(A, B)$ and the entropy of the ground-truth label set and the predicted clustered set; see Eq. 6, ($NMI(A,B) \in [0,1]$, 0 denotes no mutual information and 1 denotes a perfect correlation). A higher NMI score indicates better clustering results.

$$NMI(A,B) = \frac{2I(A,B)}{(H(A)+H(B))}, \tag{6}$$

where $I(A, B)$ is the mutual information of two jointly random variables, A and B, (A is the ground-truth class labels and B is the predicted labels), and $H(A)$ and $H(B)$ are the entropy values of A and B, respectively, which can be regarded as the probabilities that certain information exits; see Eq. 8 (Kvålseth 2017).

$$I(A,B) = \sum_k \sum_j P(A_k \cap B_j) \log \frac{P(A_k \cap B_j)}{P(A_k)P(B_j)}, \tag{7}$$

where $P(A_k)$ and $P(B_j)$ are the marginal distributions of A_k and B_j, respectively, and $P(A_k \cap B_j)$ is the joint distribution of A_k and B_j, where $A_k \in A$ and $B_j \in B$.

$$H(A) = -\sum_k P(A_k) \log P(A_k). \tag{8}$$

ARI is another standard clustering external validation indicator. By calculating the number of sample pairs assigned to the same or different clusters in the ground-truth labels and predicted cluster labels, ARI can evaluate the validity of the clustering algorithm. It ranges from -1.0 to 1.0. Random labels have an ARI close to 0.0. 1.0 stands for a perfect match. According to Park and Jun (2009), ARI is defined in Eq. 9:

$$ARI = \frac{2(ad-bc)}{(a+b)(b+d)+(a+c)(c+d))}, \tag{9}$$

where a is the number of pairs of objects that are placed in the same class in A and in the same cluster in B, b is the number of pairs in the same class in A but not in the same cluster in B, c is the number of pairs in the same cluster in A but not in the same class in B, and d is the number of pairs in different classes in A and different clusters in B.

Different from purity, both NMI and ARI can be used to validate the clustering performance among models with varying numbers of clusters.

4.2 Elbow method

To validate the strengths of the TDSM compared to SOM, we first need to determine the best neuron number K for SOM. The elbow method in the K-means clustering algorithm is used as a reference; inertia is the sum of the squared distance of samples to their closest cluster center. The inertia formula is shown in Eq. 10, and this number should be as small as possible:

$$Inertia = \sum_{k=1}^{n} \sum_{x_i \in S_k} \| x_i - C_k \|, \tag{10}$$

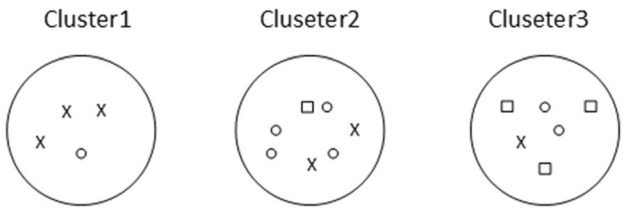

Fig. 8 Majority class in three clusters: $x:3, \circ:4, \square:3$. The purity value is $(3+4+3)/(4+7+6) \approx 0.588$

where k is the number of clusters, $\| x_i - C_k \|$ is a Euclidean norm between each data point x_i and C_k (the centroid of each cluster), and S_k denotes data in the kth cluster (Worasutr et al. 2022).

However, similar to purity, there is an extreme example: when K equals the number of samples, the inertia value is 0. Our goal is to cluster the data into the optimal number of clusters, and the inertia value decreases as the number of clusters increases, so we need to manually select K while considering the trade-off between the inertia value and the number of clusters. People usually use the elbow method and choose the elbow point in the inertia graph.

The elbow method does not always yield the "obvious" K value. When the inertia line fluctuates dramatically, it is difficult to find the elbow point. In this case, a smooth line function with a parameter called "SmoothWeight" is used to determine the elbow point; see Fig. 9. Equation 11 shows the formula for calculating the smooth value:

$$S_m = D_0 * S_w + (1 - S_w) * D_i, \tag{11}$$

where S_m is the smoothed value, D_0 is initialized as the first data sample in the inertia plot, S_w is the "SmoothWeight," and D_i is newly given data sample in the plot. After each iteration, D_0 is replaced by S_m.

Furthermore, the inertia plot can sometimes be very smooth and shows no distinct K value. As an alternative, the silhouette coefficient (Aranganayagi and Thangavel 2007) is used to measure how similar an object is to its cluster compared to other clusters. The mathematical description of the silhouette coefficient is shown in Eq. 12 (Iqbal et al. 2021):

$$S_i = \frac{b_i - a_i}{max(b_i, a_i)}, \tag{12}$$

where S_i is the silhouette score of data i, a_i is the mean distance of i to all other data points in the next-nearest cluster data, and a_i denotes the mean distance of i to all other data points in the cluster i.

$S_i \in [-1, 1]$, where a high value indicates that the data is well matched to their own cluster and poorly matched to neighboring clusters. Given a range of K values, we choose a number that generates a higher silhouette score as the optimal cluster number.

4.3 Overfitting problem

The objective of clustering is not to find the best partition of the given sample but the actual partition in the underlying space (Bubeck and Von Luxburg 2007). A machine learning model overfits the training data if the trained model is more accurate on the training data; but less accurate on the test data (Prieditis and Sapp 2013). In our hypothesis, W' can perfectly represent given samples, as presented in Fig. 11 and Fig. 12 with the purity, NMI, and ARI scores all equal to 1 on the training data. This result can be achieved on all the experimental datasets. We cannot ignore the overfitting problem in our model. Combined with Fig. 10, we can see that as the split number increases, the "unrepresented data" proportion decreases to zero; simultaneously, the purity score of the training data inversely increases to 1, which means that the

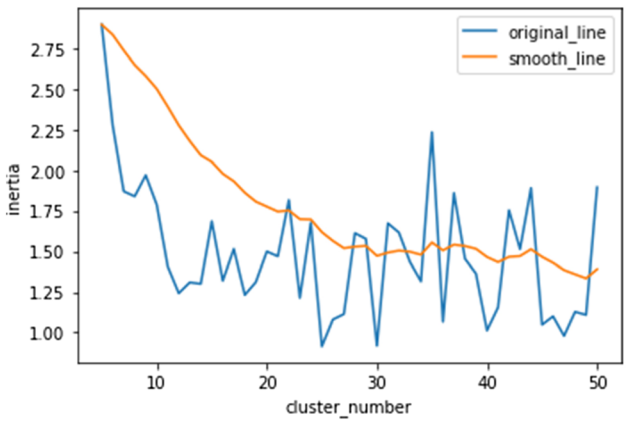

Fig. 9 Inertia graphic with a smooth function

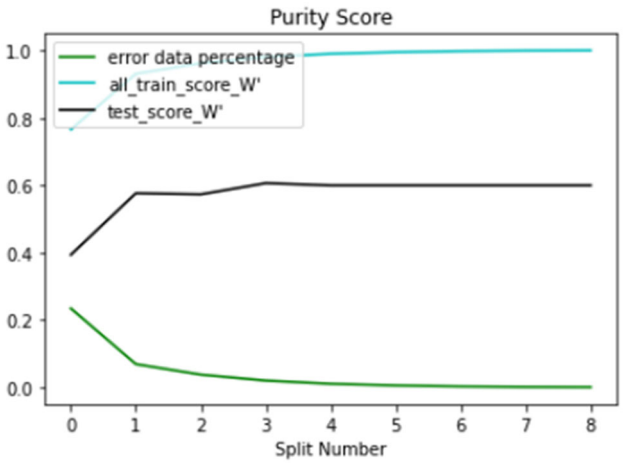

Fig. 10 The trend of the purity score and unrepresented data percentage based on an increase in the training data split number; the plot is generated from the Crowdsourced Mapping dataset with a neuron number of 12

Table 2 *T* test results for all datasets, with the assumption that test_score_W0 < test_score_W′

Dataset	Unstable repeateNum	ScopeNum	Scoretype	T test *P*-value
IRIS	30	60	Purity	**0.0003**
			NMI	**0.0000**
			ARI	**0.0000**
User knowledge modeling	10	30	Purity	**0.0002**
			NMI	**0.0000**
			ARI	**0.0000**
HCV data	30	20	Purity	**0.0034**
			NMI	**0.0000**
			ARI	**0.0000**
Absenteeism	10	20	Purity	**0.0008**
			NMI	**0.0000**
			ARI	**0.0000**
Mice protein expression	10	20	Purity	**0.0000**
			NMI	**0.0000**
			ARI	**0.0000**
Estimation of obesity levels based on eating habits and physical condition	10	20	Purity	**0.0000**
			NMI	**0.0000**
			ARI	**0.0000**
Anuran calls (MFCCs)	30	20	Purity	**0.0000**
			NMI	**0.0000**
			ARI	**0.0000**
Turkiye student evaluation	10	30	Purity	**0.0005**
			NMI	**0.0000**
			ARI	**0.0000**
Crowdsourced mapping	1	30	Purity	**0.0002**
			NMI	0.3136
			ARI	**0.0074**

Bold value indicates *P*-value and significance level

representation ability with W' increases. Nonetheless, a nontrivial outcome is that the purity score on test data reaches its maxima when the split number is not at its peak. In theory, with a decrease in the unrepresented data percentage, the quantity of data represented by the newly generated weight matrix will be smaller, which explains why the line gradually becomes horizontal at the end. Additionally, the chance that these data are outliers or noise increases as the spit number increases; for instance, when splitting the training data 1 time, we can find the majority of unrepresented data (the green line named "error data percentage" in Fig. 10). However, we can split the training data more times until there are no unrepresented data; apparently, with an increase in the split number, we can only find a very small part of unrepresented data in each split, and these data can be regarded as noise or outliers, which affect the performance of the algorithm and results in misplaced cluster centers (Askari 2021). From this point of view, similar to the idea of "neighborhood rough sets in outlier removal" (NeROR) method (Sewwandi et al. n.d.), which uses granule mining techniques (Liu et al. 2012; Sewwandi et al. 2021) to identify 100% pure clusters of data to detect outliers, the TDSM can also be used to detect outliers. Consequently, we could resort to properly reducing the split number to obtain a better result on the test data and to prevent overfitting caused by excessively splitting the training data.

4.4 Experimental results

We use the optimal cluster number K to train the model on the experimental datasets and compare the external validation results with those of the TDSM algorithm. Table 3 shows that the TDSM outperforms SOM on the listed datasets, where SOM uses the optimal cluster number. However, the TDSM uses an excessively large split number, not the optimal one. Nonetheless, the clustering quality

Table 3 SOM and the TDSM external validation result comparison, where "SmoothWeight" denotes the parameter in inertia graph smooth function, "ClusterNumRange" indicates the range of K to determine the optimal cluster number, "BestClusterNum" is the K value generated by the elbow method, "Percentage" means the increased level from SOM to the TDSM, "TDSMSplitNum" means the final split number of the TDSM, and "TDSMClusterNum" is the cluster number generated by the TDSM

Dataset	SmoothWeight	ClusterNumRange	SOMBestClusterNum	ScoreType	SOM	TDSM	Percentage	TDSMSplitNum	TDSMClusterNum
IRIS	0.5	3–30	12	NMI	0.714	**0.769**	7.70%	1	6
				ARI	0.583	**0.677**	16.12%	1	6
User knowledge modeling	0.6	4–40	18	NMI	0.185	**0.381**	105.95%	2	12
				ARI	0.206	**0.371**	80.10%	1	8
HCV data	0.8	5–50	20	NMI	0.517	**0.647**	25.15%	2	15
				ARI	0.501	**0.905**	80.64%	2	15
Absenteeism	0.4	28–84	43	NMI	0.180	**0.359**	99.44%	10	308
				ARI	0.009	**0.100**	1011.11%	12	364
Mice protein expression	0.5	8–40	24	NMI	0.384	**0.849**	121.09%	3	32
				ARI	0.293	**0.884**	201.71%	3	32
Estimation of obesity levels based on eating habits and physical condition	0.4	7–35	15	NMI	0.513	**0.765**	49.12%	5	42
				ARI	0.354	**0.750**	111.86%	4	35
Anuran calls (MFCCs)	0.5	10–50	30	NMI	0.633	**0.870**	37.44%	4	50
				ARI	0.873	**0.957**	9.62%	4	50
Turkiye student evaluation	0.6	13–65	48	NMI	0.081	**0.189**	133.33%	13	182
				ARI	0.017	**0.100**	488.24%	13	182
Crowdsourced mapping	0.3	6–60	21	NMI	0.293	**0.321**	9.56%	7	48
				ARI	0.167	**0.246**	47.31%	6	42

Bold numbers in the TDSM column mean that they have a better result than numbers in the SOM column

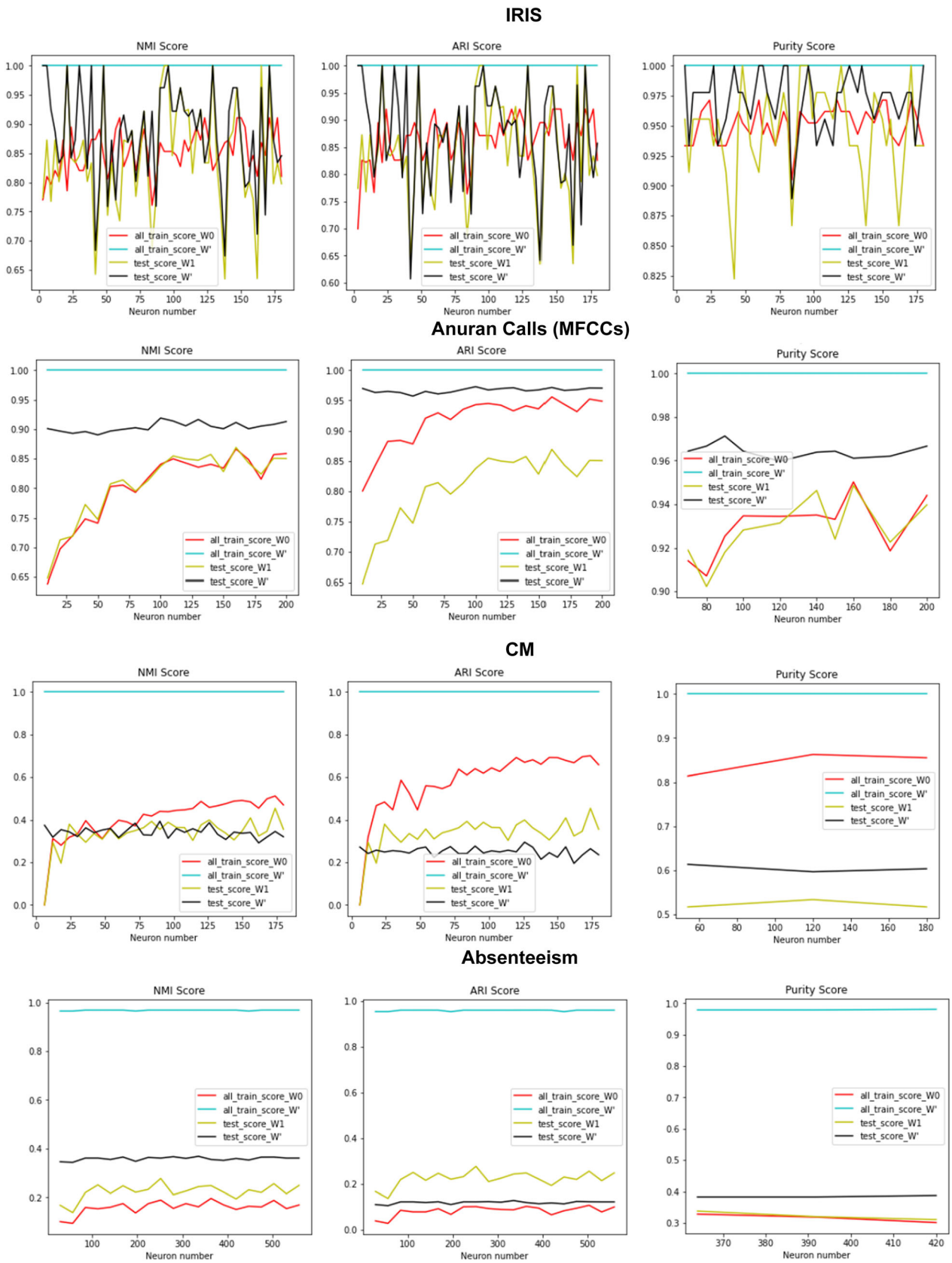

Fig. 11 The experimental results for each dataset (part 1); the purity graph have a lower density than the NMI and ARI graphics because we compute the purity score with the same cluster number in SOM and the TDSM, and there are few data samples

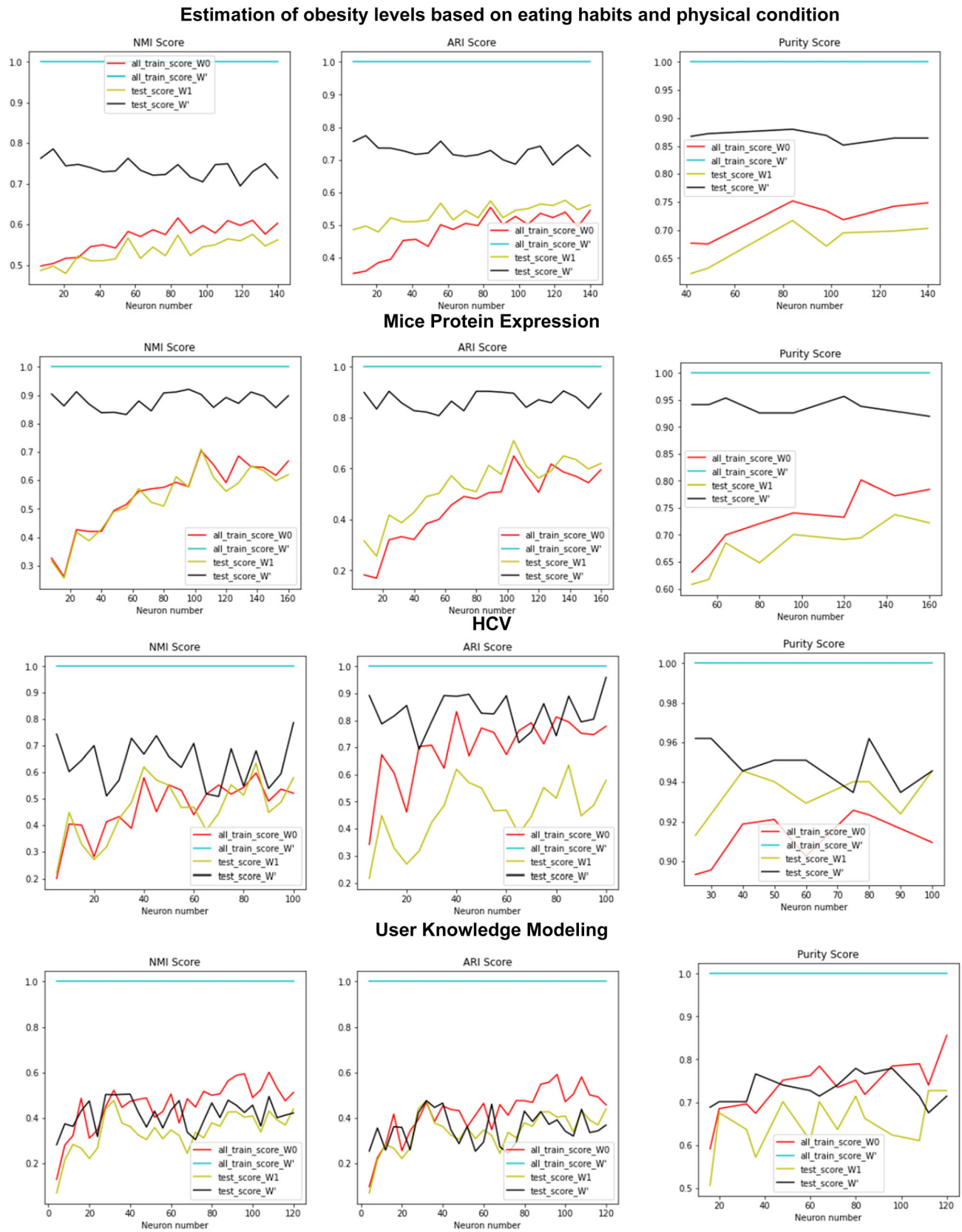

Fig. 12 The experimental results for each dataset (part 2)

is improved in terms of NMI (7.70%−133.3%) and ARI (9.62%−1011.11%). The TDSM significantly increases the clustering quality from that of the original SOM algorithm and can be applied to most datasets.

A T test is used to check whether the TDSM clustering quality is better than that of the original SOM algorithm. To obtain sufficient samples for the T test, a parameter called "ScopeNum" is applied as the upper limit of the neurons used in each test. For instance, if ScopeNum is 30, then the maximum number of neurons is the class_number $\times$ 30 (the class_number denotes the class number in the training data). The initial neuron number (minimum cluster number) is the same as the chosen ground-truth class number in each dataset. For some training sets, the test result is not stable; to make it more durable, a parameter called "Unstable RepeateNum" is added, and it represents how many times the test should be repeated. The T test result is shown in Table 2; apart from the NMI score on the Crowdsourced Mapping dataset, whose P-value (describes the probability of obtaining a sample statistic) is not less than 5% (significance level), all other results demonstrate that the TDSM outweighs SOM, the bold number shows that the assumption of the T test is true and not likely to happen strictly by chance.

5 Conclusion

This paper introduced a data-driven multiple-layer supervised clustering method called the training data splitting method (TDSM), to reduce the data samples that cannot be correctly clustered in each training step and to discover a new network structure (composed of perfect representative neurons) to represent those data. The critical point of the proposed method is to distinguish "unrepresented data" (data that are grouped into error clusters with given neurons) and "represented data" (data for which the current trained network can perfectly represent the data) in the training samples using external validation. When "unrepresented data" are removed from the clustered group data, the rest of the data can be perfectly represented by the initial neurons. By iteratively training the "unrepresented data" using the same neural network model, for every iteration, a weight matrix that represents part of the "unrepresented data" is generated until no "unrepresented data" exists. The experimental results show that the proposed model can significantly improve the clustering performance of the original SOM network in terms of purity, NMI, and ARI.

Moreover, future research could include exploring the two limitations of this supervised clustering method: 1) the training data must have a class label used for external validation to find the "unrepresented data.", and 2) there is room to optimize the algorithm's efficiency.

Acknowledgements The authors extend their appreciation to the Deanship of Scientific Research at Imam Mohammad Ibn Saud Islamic University (IMSIU) for funding and supporting this work through the Research Partnership Program no.RP-21-07-11.

Author contributions QF: Writing- Original draft preparation, Methodology, Software, Visualization,Investigation. YL: Supervision, Conceptualization, Methodology, Writing- Reviewing and Editing. MA: Reviewing and Funding acquisition.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Data availability All data generated oranalyzed during this study are included in this published article (and its supplementary information files).

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

Amelio A, Pizzuti C (2015) Is normalized mutual information a fair measure for comparing community detection methods?, In: 'Proceedings of the 2015 IEEE/ACM international conference on advances in social networks analysis and mining 2015', pp. 1584–1585

Andoni A (2009) Nearest neighbor search: the old, the new, and the impossible, PhD thesis, Massachusetts Institute of Technology

Aranganayagi S, Thangavel K (2007) Clustering categorical data using silhouette coefficient as a relocating measure, In: 'International conference on computational intelligence and multimedia applications (ICCIMA 2007)', Vol. 2, IEEE, pp. 13–17

Askari S (2021) Noise-resistant fuzzy clustering algorithm. Granul Comput 6(4):815–828

Bassani HF, Araújo AF (2012) Dimension selective self-organizing maps for clustering high dimensional data, In: 'The 2012 International Joint Conference on Neural Networks (IJCNN)', IEEE, pp. 1–8

Bassani HF, Araujo AF (2014) Dimension selective self-organizing maps with time-varying structure for subspace and projected clustering. IEEE Trans Neural Netw learn Syst 26(3):458–471

Bauer H-U, Villmann T (1997) Growing a hypercubical output space in a self-organizing feature map. IEEE Trans Neural Netw 8(2):218–226
Bishop C, Svensén M, Williams C (1996) Gtm: A principled alternative to the self-organizing map. Adv Neural Inform processing Syst. https://doi.org/10.1007/3-540-61510-5_31
Braga PH, Bassani HF (2018) A semi-supervised self-organizing map for clustering and classification, In: '2018 International Joint Conference on Neural Networks (IJCNN)', IEEE, pp. 1–8
Bubeck S, Von Luxburg U (2007) 'Overfitting of clustering and how to avoid it', Preprint pp. 1–39
Chen Y, Zhou L, Tang Y, Singh JP, Bouguila N, Wang C, Wang H, Du J (2019) Fast neighbor search by using revised kd tree. Inf Sci 472:145–162
Chiu T, Fang D, Chen J, Wang Y, Jeris C (2001) A robust and scalable clustering algorithm for mixed type attributes in large database environment, In: 'Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining', pp. 263–268
Deng Q, Mei G (2009) Combining self-organizing map and k-means clustering for detecting fraudulent financial statements, In: '2009 IEEE International Conference on Granular Computing', IEEE, pp. 126–131
Dua D, Graff C (2017) University of California. UCI machine learning repository, UCI machine learning repository. http://archive.ics.uci.edu/ml.
El Atik AEF, Nawar A, Atef M (2021) Rough approximation models via graphs based on neighborhood systems. Granul Comput 6:1025–1035
Forest F, Lebbah M, Azzag H, Lacaille J (2021) Deep embedded self-organizing maps for joint representation learning and topology-preserving clustering. Neural Comput Appl 33(24):17439–17469
Ghaseminezhad M, Karami A (2011) A novel self-organizing map (som) neural network for discrete groups of data clustering. Appl Soft Comput 11(4):3771–3778
Ghosh S, Patra S, Ghosh A (2009) An unsupervised context-sensitive change detection technique based on modified self-organizing feature map neural network. Int J Approx Reason 50(1):37–50
Hua W, Mo L (2020) Clustering ensemble model based on self-organizing map network. Comput Intell Neurosci. https://doi.org/10.1155/2020/2971565
Iqbal F, Batool R, Fung BC, Aleem S, Abbasi A, Javed AR (2021) Toward tweet-mining framework for extracting terrorist attack-related information and reporting. IEEE Access 9:115535–115547
Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv (CSUR) 31(3):264–323
Jiang W, Chung F-l (2012) Transfer spectral clustering, In: 'Joint European Conference on Machine Learning and Knowledge Discovery in Databases', Springer, pp. 789–803
Kaski S, Kohonen T (1994) Winner-take-all networks for physiological models of competitive learning. Neural Netw 7(6–7):973–984
Kohonen T (1982) Self-organized formation of topologically correct feature maps. Biol Cybern 43(1):59–69
Kohonen T, Honkela T (2007) Kohonen network. Scholarpedia 2(1):1568
Kvålseth TO (2017) On normalized mutual information: measure derivations and properties. Entropy 19(11):631
Lampinen J, Oja E (1992) Clustering properties of hierarchical self-organizing maps. J Math Imaging Vis 2(2):261–272
Litinskii LB, Romanov DE (2006) Neural network clustering based on distances between objects, In: 'International Conference on Artificial Neural Networks', Springer, pp. 437–443
Liu B, Li Y, Wang K (2012) Granule mining and its application for network traffic characterization, In: 'Intelligent Decision Technologies', Springer, pp. 333–343
Lobo VJ (2009) Application of self-organizing maps to the maritime environment, In: 'Information Fusion and Geographic Information Systems', Springer, pp. 19–36
Mailagaha Kumbure M, Luukka P (2022) A generalized fuzzy k-nearest neighbor regression model based on minkowski distance. Granul Comput 7(3):657–671
Miljković D (2017) Brief review of self-organizing maps, In: '2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)', IEEE, pp. 1061–1066
Park KW (2022) 'Extending self-organizing maps with ranking awareness'
Park H-S, Jun C-H (2009) A simple and fast algorithm for k-medoids clustering. Expert Syst Appl 36(2):3336–3341
Prieditis A, Sapp S (2013) Lazy overfitting control, In: 'International Workshop on Machine Learning and Data Mining in Pattern Recognition', Springer, pp. 481–491
Rauber A, Merkl D, Dittenbach M (2002) The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data. IEEE Trans Neural Netw 13(6):1331–1341
Rawat R, Nayak R, Li Y, Alsaleh S (2011) Aggregate distance based clustering using fibonacci series-fibclus, In: 'Asia-Pacific Web Conference', Springer, pp. 29–40
Rougier N, Boniface Y (2011) Dynamic self-organising map. Neurocomputing 74(11):1840–1847
Sewwandi NDS, Li Y, Zhang J (n.d.) 'K-outlier removal based on contextual label information and cluster purity for continuous data classification', Available at SSRN 4214220
Sewwandi M, Li Y, Zhang J (2021) Automated granule discovery in continuous data for feature selection. Inf Sci 578:323–343
Smith R (2021) 'Sklearn-som'. Accessed: 2022-07-10. https://pypi.org/project/sklearn-som/
Spanakis G, Weiss G (2016) 'Amsom: Adaptive moving self-organizing map for clustering and visualization', arXiv preprint arXiv:1605.06047
Sripada SC, Rao MS (2011) Comparison of purity and entropy of k-means clustering and fuzzy c means clustering, Indian journal of computer science and engineering 2(03)
Tyler S (2006) Self-organizing maps'. URL: http://tyl.st/projects/self-organizing-maps/. Accessed 19 July 2022
Thrun MC (2021) Distance-based clustering challenges for unbiased benchmarking studies. Sci Rep 11(1):1–12
Worasutr A, Worasawate D, et al (2022) Improved Target Detection Accuracy of W-BAND FMCW RADAR Using K-Means Clustering and Elbow Technique, PhD thesis, Kasetsart University