Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2017), *31*, 30–54.
© Cambridge University Press 2016 0890-0604/16
doi:10.1017/S0890060416000160

A dynamic semisupervised feedforward neural network clustering

ROYA ASADI,[1] SAMEEM ABDUL KAREEM,[1] SHOKOOFEH ASADI,[2] AND MITRA ASADI[3]

[1]Department of Artificial Intelligence, Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia
[2]Department of Agricultural Management Engineering, Faculty of Ebne-Sina, University of Science and Research Branch, Tehran, Iran
[3]Department of Research, Iranian Blood Transfusion Organization, Tehran, Iran

(RECEIVED January 13, 2015; ACCEPTED January 6, 2016)

Abstract

An efficient single-layer dynamic semisupervised feedforward neural network clustering method with one epoch training, data dimensionality reduction, and controlling noise data abilities is discussed to overcome the problems of high training time, low accuracy, and high memory complexity of clustering. Dynamically after the entrance of each new online input datum, the code book of nonrandom weights and other important information about online data as essentially important information are updated and stored in the memory. Consequently, the exclusive threshold of the data is calculated based on the essentially important information, and the data is clustered. Then, the network of clusters is updated. After learning, the model assigns a class label to the unlabeled data by considering a linear activation function and the exclusive threshold. Finally, the number of clusters and density of each cluster are updated. The accuracy of the proposed model is measured through the number of clusters, the quantity of correctly classified nodes, and *F*-measure. Briefly, in order to predict the survival time, the *F*-measure is 100% of the Iris, Musk2, Arcene, and Yeast data sets and 99.96% of the Spambase data set from the University of California at Irvine Machine Learning Repository; and the superior *F*-measure results in between 98.14% and 100% accuracies for the breast cancer data set from the University of Malaya Medical Center. We show that the proposed method is applicable in different areas, such as the prediction of the hydrate formation temperature with high accuracy.

Keywords: Artificial Neural Network; Feedforward Neural Network; Nonrandom Weight; Online Dynamic Learning; Semisupervised Clustering; Supervised and Unsupervised Learning

1. INTRODUCTION

An artificial neural network (ANN) is inspired by the manner of biological nervous system, such as human brain to process the data, and is one of the numerous algorithms used in machine learning and data mining (Dasarthy, 1990; Kemp et al., 1997; Goebel & Gruenwald, 1999; Hegland, 2003; Kantardzic, 2011). In a feedforward neural network (FFNN), data processing has only one forward direction from the input layer to the output layer without any backward loop or feedback connection (Bose & Liang, 1996; McCloskey, 2000; Andonie & Kovalerchuk, 2007; Kantardzic, 2011). Learning is an imperative property of the neural network. There are many types of learning rules used in the neural networks, which fall under the broad categories of supervised learning, unsupervised learning, and reinforcement learning. Most approaches to unsupervised learning in machine learning are statistical modeling, compression, filtering, blind source separation, and clustering (Hegland, 2003; Han & Kamber, 2006; Andonie & Kovalerchuk, 2007; Kantardzic, 2011). In this study, the clustering aspect of unsupervised neural network learning is considered. Learning from observations with unlabeled data in an unsupervised neural network clustering is more desirable and affordable than learning by examples in supervised neural network classification because preparing the training set is costly, time consuming, and possibly dangerous in some environments. However, to evaluate the performance of unsupervised learning, there is no error or reward indication (Kohonen, 1997; Demuth et al., 2008; Van der Maaten et al., 2009). One of the popular supervised FFNN models is the backpropagation network (BPN; Werbos, 1974). The BPN uses gradient-based optimization methods in two basic steps: to calculate the gradient of the error function and to employ the gradient. The optimization

Reprint requests to: Roya Asadi, Department of Artificial Intelligence, Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, 60503, Selangor, Malaysia. E-mail: royaasadi@siswa.um.edu.my

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

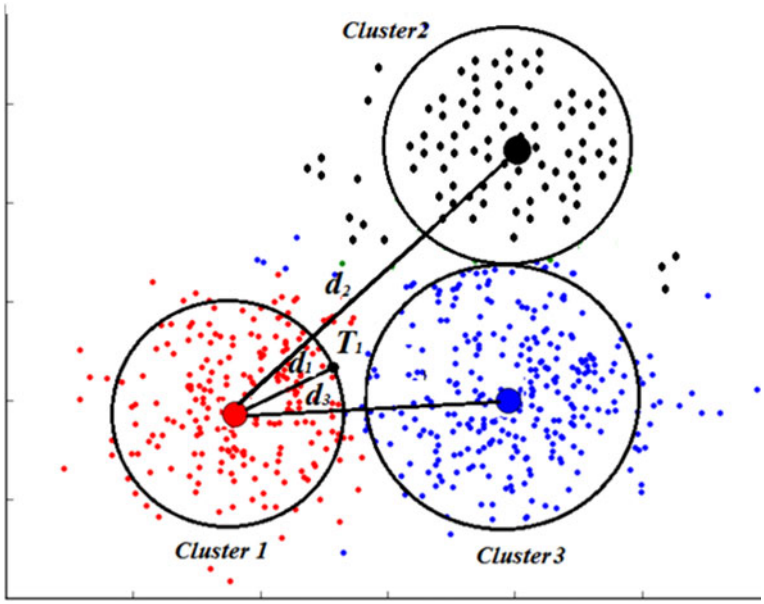


Fig. 1. A sample of the distances between the clusters and within each cluster.

procedure, which includes a high number of small steps, causes the learning process to be considerably slow. An optimization problem in supervised learning can be shown as the sum of squared errors between the output activations and the target activations in the neural network as well as the minimum weights (Bose & Liang, 1996; Craven & Shavlik, 1997; Andonie & Kovalerchuk, 2007). The next sections are the history and overview of the unsupervised FFNN (UFFNN) clustering.

1.1. UFFNN clustering

The UFFNN clustering has great capabilities, such as the inherent distributed parallel processing architectures and the ability to adjust the interconnection weights to learn and divide data into meaningful groups. The UFFNN clustering method classifies related data into similar groups without using any class label, and in addition controls noisy data and learns types of input data values based on their weights and properties (Bengio et al., 2000; Hegland, 2003; Andonie & Kovalerchuk, 2007; Jain, 2010; Rougier & Boniface, 2011). In UFFNN clustering, data is divided into meaningful groups with special goals, with related data classified as higher similarities within groups and unrelated data as dissimilarities between groups without using any class label. For example in Figure 1, T_1 shows the maximum threshold of cluster 1, and d shows the distance between two data nodes.

The UFFNN methods often use Hebbian learning, or competitive learning, or competitive Hebbian learning (Martinetz, 1993; Fritzke, 1997). Hebb (1949) developed the meaning of the first learning rule and proposed Hebbian learning. Figure 2 illustrates the single-layer UFFNN as a simple topology with Hebbian learning (Laskowski & Touretzky, 2006). Hebb described a synaptic flexibility mechanism in which the synaptic connection between two neurons is strengthened, and neuron j becomes more sensitive to the action of neuron i if the latter is close enough to stimulate the former while repeatedly contributing to its activation.

The Hebbian rule is shown in Eq. (1):

$$\Delta W_i = \gamma Y X_i, \tag{1}$$

where X is the input vector; Y is the output vector; and γ is the learning rate, where $\gamma > 0$ is used to control the size of each training iteration. The competitive learning network is a UFFNN clustering based on learning the nearest weight vector to the input vector as the winner node according to the computing distance, such as Euclidean. Figure 3 shows a sample topology of an unsupervised competitive learning neural network (Haykin & Network, 2004; Du, 2010).

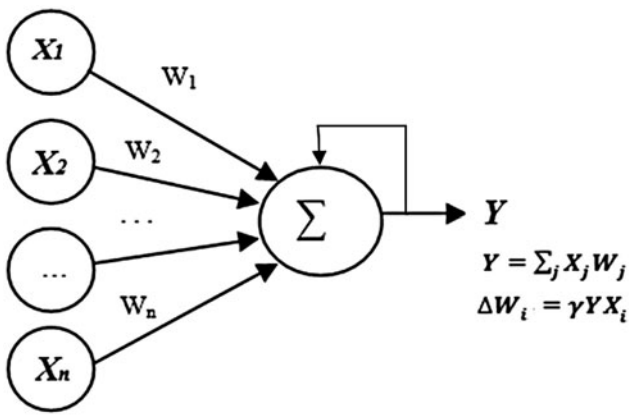


Fig. 2. Single-layer unsupervised feedforward neural network with the Hebbian learning.

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

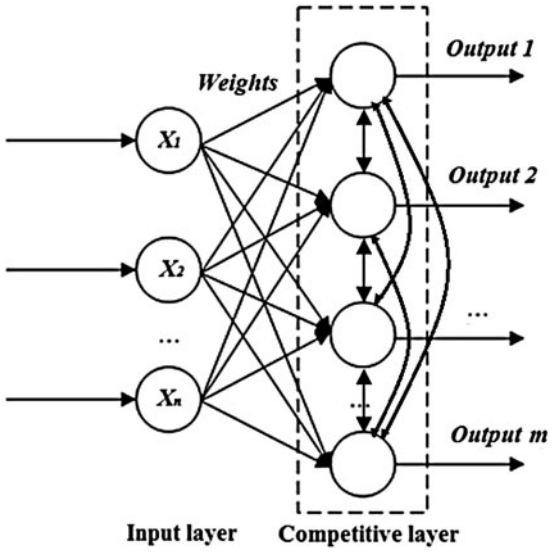


Fig. 3. A sample topology of the competitive clustering.

The similarities between Hebbian learning and competitive learning include unsupervised learning without error signal, and are strongly associated with biological systems. However, in competitive learning, only one output must be active, such that only the weights of the winner are updated in each epoch. By contrast, no constraint is enforced by neighboring nodes in Hebbian learning, and all weights are updated at each epoch. In the case of competitive Hebbian learning, the neural network method shares some properties of both competitive learning and Hebbian learning. Competitive learning can apply vector quantization (VQ; Linde et al., 1980) during clustering. VQ, K-means (Goebel & Gruenwald, 1999), and some UFFNN clustering methods, such as Kohonen's self-organizing map (SOM; Kohonen, 1997) and growing neural gas (GNG; Fritzke, 1995), are generally considered as the fundamental patterns in the current online dynamic UFFNN (ODUFFNN) clustering methods (Asadi et al., 2014*b*). Linde et al. introduced an algorithm for the VQ design to obtain a suitable code book of weights for input data nodes clustering. VQ is based on probability density functions by distribution of the vector of the weights. VQ divides a large set of the data (vectors) into clusters, each of which is represented by its centroid node, as in the K-means, which is a partitioning clustering method and some other clustering algorithms. The GNG method is an example that uses the competitive Hebbian learning, in which the connection between the winner node and the second nearest node is created or updated in each training cycle. The GNG method can follow dynamic distributions by adding nodes and deleting them in the network during clustering by using the utility parameters. The disadvantages of the GNG include the increase in the number of nodes to obtain the input probability density and the requirement for predetermining the maximum number of nodes and thresholds (Germano, 1999; Hamker, 2001; Furao et al., 2007; Hebboul et al., 2011). Kohonen's SOM maps multidimensional data onto lower dimensional subspaces, with the geometric relationships between points indicating their similarity. SOM generates subspaces with unsupervised learning neural network training through a competitive learning algorithm. The weights are adjusted based on their proximity to the "winning" nodes, that is, the nodes that most closely resembles a sample input (Ultsch & Siemon, 1990; Honkela, 1998).

1.2. ODUFFNN clustering

The UFFNN methods with online dynamic learning in realistic environments such as astronomy and satellite communications, e-mail logs, and credit card transactions must be improved and have some necessary properties (Kasabov, 1998; Schaal & Atkeson, 1998; Han & Kamber, 2006; Hebboul et al., 2011). The data in these environments are nonstationary, so ODUFFNN clustering methods should have lifelong (online) and incremental learning. The flexible incremental and dynamic neural network clustering methods are able to do the following (Kasabov, 1998; Schaal & Atkeson, 1998; Bouchachia et al., 2007; Hebboul et al., 2011):

- to learn the patterns of high dimensional and huge continuous data quickly; therefore, training should be in one pass. In these environments, the data distributions are not known and may be changed over time.
- to handle new data immediately and dynamically, without destroying old data. The ODUFFNN clustering method should control data, adapt its algorithm, and adjust itself in a flexible style to new conditions of the environment over time dynamically for processing of both data and knowledge.
- to change and modify its structure, nodes, connection, and and so forth with each online input data.
- to accommodate and prune data and rules incrementally, without destroying old knowledge.
- to be able to control time, memory space, accuracy, and so forth efficiently.
- to learn the number of clusters and density of each cluster without predetermining the parameters and rules.

Incremental learning refers to the ability of training in repetition by adding or deleting the data nodes in lifelong learning without destroying outdated prototype patterns (Schaal & Atkeson, 1998; Furao et al., 2007; Rougier & Boniface, 2011). The ODUFFNN clustering methods should train online data fast without relearning. Relearning during several epochs takes time, and clustering is considerably slow. Relearning affects the clustering accuracy, and time and memory usage for clustering. For the ODUFFNN methods, storing the whole details of the online data and their connection during relearning with the limited memory are impossible. In this case, the topological structure of the incremental online data cannot be represented well, the number of clusters and density of each cluster are not clear, and they cannot be easily learned (Pavel, 2002; Deng & Kasabov, 2003; Hinton & Salakhutdinov,

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

2006; Hebboul et al., 2011). The number of data instances by receiving online data will be grown. This action causes difficulty in clustering, managing the structure of the network and connections between the data, and recognizing the noisy data. Furthermore, clustering of some kinds of data is so difficult because of their character and structure. High feature correlation and the noise in the data cause difficulty in the clustering process, and recognizing the special property of each attribute and finding its related cluster will be difficult. The main disadvantage of dimensionality reduction or feature extraction of data as the data-preprocessing technique is missing values. Missing values are because some important parts of the data are lost, which reflects the accuracy of clustering results (DeMers & Cottrell, 1993; Furao et al., 2007; Van der Maaten et al., 2009). Therefore, recognizing the special property of each attribute and finding its related cluster will be difficult (Kasabov, 1998; Hebboul et al., 2011; Rougier & Boniface, 2011).

Current ODUFFNN clustering methods often use competitive learning as used in the dynamic SOM (DSOM; Rougier & Boniface, 2011), or competitive Hebbian learning as used in the evolving SOM (ESOM). Furao and Hasegawa introduced enhanced self-organizing incremental neural network (ESOINN; Furao et al., 2007) based on the GNG method. The ESOINN method has one layer, and in this model, it is necessary that very old learning information is forgotten. The ESOINN model finds the winner and the second winner of the input vector, and then if it is necessary to create a connection between them or to remove the connection. The density, the weight of the winner, and the subclass label of nodes will be updated in each epoch and the noise nodes depending on the input values will be deleted. After learning, all nodes will be classified into different classes. However, it cannot solve the main problems of clustering. Hebboul et al. (2011) proposed incremental growing with neural gas utility parameter as the latest online incremental unsupervised clustering. The structure is based on the GNG and Hebbian (Hebb, 1949) models, but without any restraint and control on the network structure. The structure of the incremental growing with neural gas utility parameter contains two layers of learning. The first layer creates a suitable structure of the clusters of the input data nodes with lower noise data, and computes the threshold. The second layer uses the output of the first layer in parallel and creates the final structure of clusters. ESOM (Deng & Kasabov, 2003) as an ODUFFNN method is based on SOM and GNG methods. ESOM starts without nodes. The network updates itself with online entry, and if necessary, it creates new nodes during one training epoch. Similar to the SOM method, each node has a special weight vector. The strong neighborhood relation is determined by the distance between connected nodes; therefore, it is sensitive to noise nodes, weak connections, and isolated nodes based on Hebbian learning. If the distance is too big, it creates a weak threshold and the connection can be pruned. Figure 4 is an example of this situation (Deng & Kasabov, 2003).

ESOM is a method based on a normal distribution and VQ in its own way, and creates normal subclusters across the data space. DSOM (Rougier & Boniface, 2011) is similar to SOM based on competitive learning. In order to update the weights of the neighboring nodes, time dependency is removed, and the parameter of the elasticity or flexibility is considered, which is learned by using trial and error. If the parameter of the elasticity is too high, DSOM does not converge; if it is too low, it may prevent DSOM from occurring and is not sensitive to the relation between neighbor nodes. If no node is close enough to the input values, other nodes must learn according to their distance to the input value. The main critical issues in the ODUFFNN clustering method are low training speed, accuracy, and high memory complexity of clustering (Kasabov, 1998; Han & Kamber, 2006; Andonie & Kovalerchuk, 2007; Asadi et al., 2014*b*). Some sources of the problems are associated with these methods. High dimensional data and huge data sets cause difficulty in managing new data and noise, while pruning causes data details to be lost (Kohonen, 2000; Deng & Kasabov, 2003; Hinton & Salakhutdinov, 2006; Van der Maaten et al., 2009). Using random weights, thresholds, and parameters for controlling clustering tasks, create the paradox of low accuracy and high training time (Han & Kamber, 2006; Hebboul et al., 2011; Asadi & Kareem, 2014). Moreover, the data details and their connections are lost by relearning that affects the CPU time usage, memory usage, and clustering accuracy (Pavel, 2002; Hebboul et al., 2011). Some literature is devoted to improving the UFFNN and ODUFFNN methods by the technique of using constraints such as class labels. The constraints of class labels are based on the knowledge of experts and the user guide as partial supervision for better controlling the tasks of clustering and desired results (Prudent & Ennaji, 2005; Kamiya et al., 2007; Shen et al., 2011). The semi-SOINN (SSOINN; Shen et al., 2011) and semi-ESOM (Deng & Kasabov, 2003), which were developed based on the SOINN and

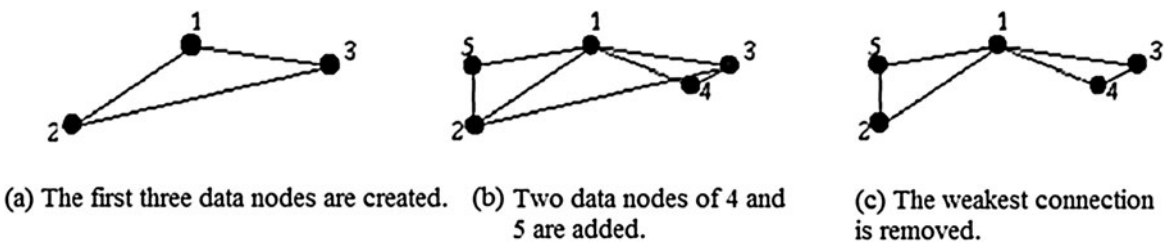


Fig. 4. An example of the evolving self-organizing map clustering.

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

ESOM clustering methods, respectively, are examples in this area. In order to improve the methods, the users manage and correct the number of clusters and density of each cluster by inserting and deleting the data nodes and clusters. After clustering, the models assigned a class label to the winning node and consequently assigned the same class labels to its neighbor nodes in its cluster. Each cluster must have a unique class label; if the data nodes of a cluster have different class labels, the cluster can be divided into different subclusters. However, assigning the class labels to the data nodes between the clusters can be somewhat vague. The judgment of users can be wrong, or they may make mistakes during the insertion, deletion, or finding the link between nodes and assigning the class label to each disjoint subcluster. Asadi et al. (2014*a*) applied this technique and introduced a UFFNN method, the efficient real semisupervised FFNN (RSFFNN) clustering model, with one epoch training and data dimensionality reduction ability to overcome the problems of high CPU time usage during training, low accuracy, and high memory complexity of clustering, suitable for stationary environments (Asadi et al., 2014*a*). Figure 5 shows the design of the RSFFNN clustering method.

The RSFFNN considered a matrix of data set as input data for clustering. During training, a nonrandom weights code book was learned through input data matrix directly, by using normalized input data and standard normal distribution. A standard weight vector was extracted from the code book, and afterward fine-tuning is applied by single-layer FFNN clustering section. The fine-tuning process includes two techniques of smoothing

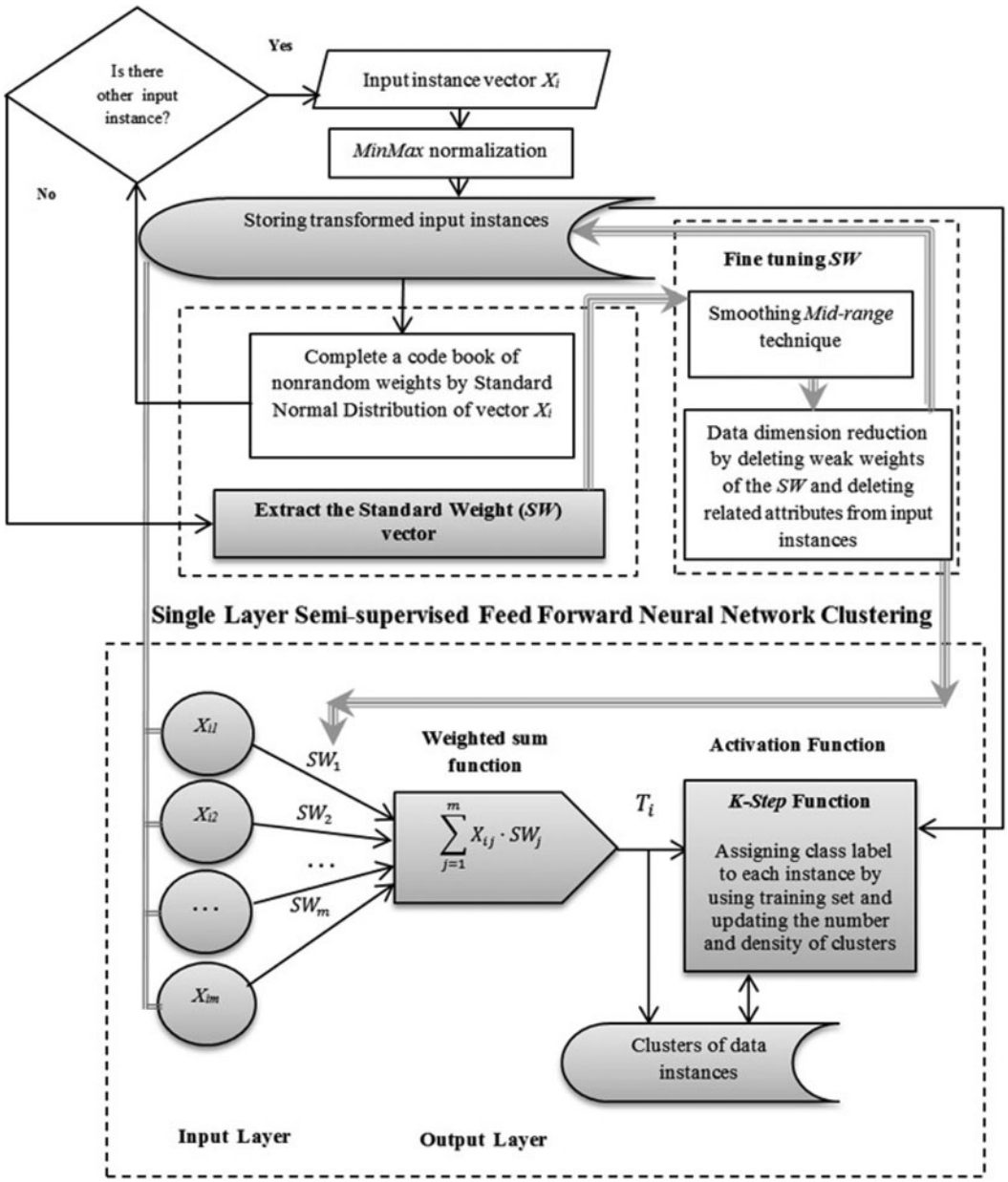


Fig. 5. The design of the real semisupervised feedforward neural network model for clustering (Asadi et al., 2014*a*).

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

the weights and pruning the weak weights: the midrange technique as a popular smoothing technique is used (Jean & Wang, 1994; Gui et al., 2001); then, the model prunes the data node attributes with weak weights in order to reduce the dimension of data. Consequently, the single-layer FFNN clustering section generates the exclusive threshold of each input instance (record of the data matrix) based on a standard weight vector. The input instances were clustered on the basis of their exclusive thresholds. In order to improve the accuracy of the clustering results, the model assigns a class label to each input instance by considering the training set. The class label of each unlabeled input instance was predicted by utilizing a linear activation function and the exclusive threshold. Finally, the RSFFNN model updated the number of clusters and density of each cluster. The RSFFNN model illustrated superior results; however, the model must be developed for using as the ODUFFNN clustering method. We introduced an efficient DSFFNN clustering method by developing and modifying the structure of the RSFFNN to overcome the mentioned problems.

2. METHODOLOGY

In order to overcome the problems of the ODUFFNN clustering methods as discussed in the last section, we developed the DSFFNN clustering method. For the purposes, the RSFFNN (Asadi et al., 2014*a*) method as an efficient UFFNN cluster-

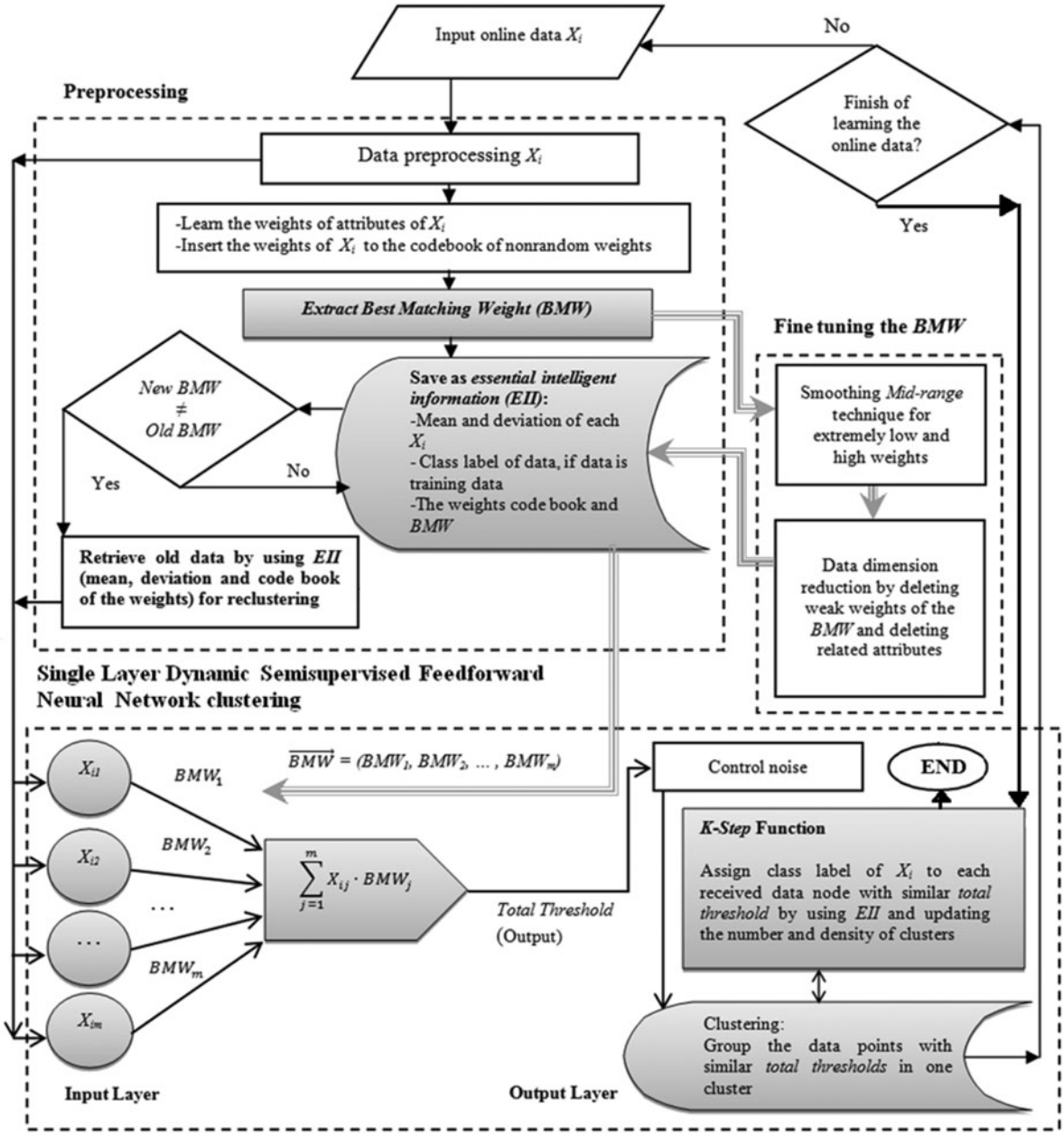


Fig. 6. The design of the dynamic semisupervised feedforward neural network clustering method.

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

ing method is structurally improved. Therefore, the DSFFNN model updates its structure, connections, and knowledge through learning the online input data dynamically. The DSFFNN model starts without any random parameters or coefficient value that needs predefinition. Figure 6 shows the design of the DSFFNN clustering method.

As shown in Figure 6, the DSFFNN method includes two main sections: the preprocessing section and the single-layer DSFFNN clustering section. In the preprocessing section, the DSFFNN method as an incremental ODUFFNN method considers a part of the memory called essential important information (EII) and initializes the EII by learning the important information about each online input data in order to store and fetch during training, without storing any input data in the memory. The code words of nonrandom weights are generated by training current online input data just in one epoch, and are inserted in the weights code book. Consequently, the unique standard vector is mined from the code book of the weights and stored as the EII in the memory. The single layer of the DSFFNN clustering applies normalized data values, and fetches some information through the EII such as the BMW from the section of the preprocessing and generates thresholds and clusters the data nodes. The topology of the single-layer DSFFNN clustering model is very simple with incremental learning, as shown in Figure 6; it consists of an input layer with m nodes and an output layer with just one node without any hidden layer. The output layer has one unit with a weighted sum function for computing the actual desired output. We generally called the proposed method the DSFFNN clustering method; however, before semisupervised clustering, the model dynamically clusters the data nodes without using any class label. Therefore, we call the clustering phase of the proposed method the dynamic UFFNN (DUFFNN) clustering method. Then in order to improve the accuracy of the result of the DUFFNN clustering, the model applies class labels through using a K-Step activation function. Therefore, we call the proposed model the DSFFNN clustering method.

2.1. Overview of the DSFFNN clustering method

In this section, we illustrate the complete algorithm of the DSFFNN clustering method and explain the details of the proposed clustering method as shown in Figure 6 step by step.

Algorithm: The DSFFNN clustering

Input: Online input data X_i;

Output: Clusters of data;

Initialize the parameters:

Let X: Data node domain;

Let newMin_i: Minimum value of the specific domain of [0, 1] which is zero;

Let newMax_i: Maximum value of the specific domain [0, 1] which is one;

Let i: Current number of the online data node X_i;

Let j: Current number of the attribute;

Let X_i: ith Current online input data node from domain of X;

Let D: Retrieved old data node from the memory;

Let f: Current number of the received data node;

Let n: Number of received data nodes;

Let m: Number of attributes;

Let W_{ij}: Weight of attribute j of ith current online data node X_i;

Let $\overrightarrow{\text{Prod}}$: A vector of the weights product of each attribute of the weights code book;

Let Prod_j: jth component of the $\overrightarrow{\text{Prod}}$ vector which $\overrightarrow{\text{Prod}} = (\text{Prod}_1, \text{Prod}_2, \ldots, \text{Prod}_m)$;

Let SumBMW: a variable for storing the sum of components of the BMW vector;

Let $\overrightarrow{\text{BMW}}$: Best matching weight vector which $\overrightarrow{\text{BMW}} = (\text{BMW}_1, \text{BMW}_2, \ldots, \text{BMW}_m)$;

Let BMW_j: jth component of the $\overrightarrow{\text{BMW}}$ vector;

Let $\text{BMW}_{j\text{Old}}$: jth component of the old $\overrightarrow{\text{BMW}}$ vector;

Let $\text{BMW}_{j\text{New}}$: jth component of the new $\overrightarrow{\text{BMW}}$ vector;

Let T_{ij}: Threshold of attribute_j of ith current online data node X_i;

Let TT_i: Total threshold of ith current online data node X_i;

Let T_{fj}: Threshold of attribute_j of fth received data node;

Let TT_f: Total threshold of fth received data node;

Method:

While the termination condition is not satisfied

{

Input a new online input data X_i;

// 1- Preprocessing

// Data preprocessing of X_i based on MinMax technique

$$\forall\, j = 1 \text{ to } m$$

$$X_{ij} = \left(X_{ij} - \text{Min}(X_{ij})/\text{Max}(X_{ij}) - \text{Min}(X_{ij})\right) \times (\text{newMax} - \text{newMin}) + \text{newMin}$$

// Compute the weight code words of the current online data X_i and update the codebook

// Compute the standard normalize distribution (SND) of the current online input data of X_i based on μ_i and σ_i which are mean and standard deviation of X_i:

$\forall\, j = 1$ to m

{

$\text{SND}(X_{ij}) = (X_{ij} - \mu_i)/\sigma_i$;

// Consider W_{ij} as weight of X_{ij} equal $\text{SND}(X_{ij})$

$W_{ij} = \text{SND}(X_{ij})$;

Insert W_{ij} into the weights codebook;

// Update $\overrightarrow{\text{Prod}}$ by considering W_{ij}

$\text{Prod}_j = \text{Prod}_j \times W_{ij}$;

}

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

// **Extracting the BMW vector** Extract the global geometric mean vector of the code book of nonrandom weights as the BMW

$\forall\, j = 1$ to m

$\text{BMW}_j = \text{Prod}_j^{1/a}$;

$\text{SumBMW} = \sum_{j=1}^{m} \text{BMW}_j$;

$\forall\, j = 1$ to m

$\text{BMW}_j = \text{Round}\big((\text{BMW}_j/\text{SumBMW}),\, 2\big)$;

// **Update EII** Store the weights, mean, and standard deviation of X_i as the EII in the memory

{

Memory(EII) ← Store $(W_{ij},\, \mu_i,\, \sigma_i)$;

Memory(EII) ← Store $(\overrightarrow{\text{BMW}},\, \overrightarrow{\text{Prod}})$;

If X_i is from training data and has class label

{

Memory(EII) ← Store (class label of X_i)

}

}

//2- Fine-tuning through two techniques

// a) Smooth the components of the BMW vector

$\forall\, j = 1$ to m

Midrange (BMW_j);

// b) Data dimension reduction

Delete attributes with weak weights of the BMW_j, that are close to zero;

//3- Single layer DSFFNN clustering

Fetch $(\overrightarrow{\text{BMW}})$ ← Memory(EII);

// Compute the exclusive total threshold of just X_i based on the new BMW

$\forall\, j = 1$ to m

$\text{TT}_i = \text{TT}_i + X_{ij} \times \text{BMW}_j$;

If X_i is from training data and has class label

{

Memory(EII) ← Store (class label of X_i, TT_i);

}

// If new BMW is different with old BMW, the model fetches W_{fj}, σ_f, and μ_f as the EII from memory and updates thresholds of just the changed attributes, and update the exclusive total threshold of related data point

If $(\overrightarrow{\text{BMW}}_{\text{New}} \neq \overrightarrow{\text{BMW}}_{\text{Old}})$

{

$\forall\, j = 1$ to m

If $\text{BMW}_{j\text{New}} \neq \text{BMW}_{j\text{Old}}$

$\forall\, f = 1$ to $n - 1$

{

Fetch $\left(W_{fj},\ \sigma_f,\ \mu_f\right)$ ← Memory(EII);

$D_{fj} = (W_{fj} \times \sigma_f) + \mu_f$;

$T_{fj\text{Old}} = D_{fj} \times \text{BMW}_{j\text{Old}}$;

$TT_f = TT_f - T_{f\text{Old}}$;

$T_{fj\text{New}} = D_{fj} \times \text{BMW}_{j\text{New}}$;

$TT_f = TT_f + T_{f\text{New}}$;

}

Memory(EII) ← Update list of thresholds and related class labels;

}

// Recognize and delete noise

Delete isolated input data with solitary thresholds TT;

// DUFFNN clustering

{

Group the data points with similar thresholds (TT_i) in one cluster;

Learn and generate optimized number of clusters and their densities;

}

If learning is finished

{

//Improving the result of DUFFNN clustering by using K-step activation function

{

Assign class label to each data node with similar total threshold by using EII;

Prediction the class label to unlabeled data nodes;

Updating the number of clusters and density of each cluster;

}

Output results;

End;

}

Else { Continue to train and cluster next online input data node

}

}

The DSFFNN clustering method involves several phases:

- **Preprocessing:** Preprocessing is the factor that contributes to the development of efficient techniques to achieve desirable results of the UFFNN clustering such as low training time and high accuracy (Abe, 2001; Larochelle et al., 2009; Oh & Park, 2011; Asadi & Kareem, 2014).

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

The DSFFNN clustering method, contrary to the RSFFNN, applies the preprocessing method that is suitable for online input data preprocessing.

1. Data preprocessing: As shown in Figure 6, the MinMax normalization technique that is suitable for online input data preprocessing and is independent of the other data points (Han & Kamber, 2006; Asadi & Kareem, 2014) is considered. The MinMax normalization technique is used to transform an input value of each attribute to fit in a specific range such as [0, 1]. Equation (2) shows the formula of MinMax normalization.

$$\text{Normalized}(X_{ij}) = (X_{ij} - \text{Min}(X_{ij})/\text{Max}(X_{ij}) - \text{Min}(X_{ij})) \times (\text{newMax} - \text{newMin}) + \text{newMin}, \quad (2)$$

where X_{ij} is the jth attribute value of online input data X_i and has special range and domain, where $\text{Min}(X_{ij})$ is the minimum value in this domain and $\text{Max}(X_{ij})$ is the maximum value in this domain; newMin_i is the minimum value of specific domain of [0, 1], which is equal to zero; and newMax_i is the maximum value of the domain, which is equal to one. After transformation of current online input data, the model continues to learn current data in the next stages.

2. Compute the weight code words of the current online data X_i and update the codebook: the DSFFNN method creates a code book of nonrandom weights by entering the first online input data and consequently completes the codebook by inserting the code words of each future online input data. In this stage, the proposed model computes the mean μ_i of the normalized current online input data X_i. Then the standard deviation σ_i of the input data of X_i is computed by considering μ_i. Table 1 provides this information.

   On the basis of the definition of the SND (Ziegel, 2002), the SND shows how far each attribute value of the online input data X_i is from the mean μ_i, in the metric standard deviation unit σ_i. In this step, each normalized attribute value of the online input data X_i is considered as the weight W_{ij} for that value. Each element or code word of the weight code book is equal to W_{ij}. Therefore, each weight vector of the codebook is computed based on the SND of each online input data value of X_i as shown in Eq. (3).

$$\text{SND}(X_{ij}) = (X_{ij} - \mu_i)/\sigma_i. \quad (3)$$

The $\text{SND}(X_{ij})$ is a standard normalized value of each attribute value of the online input data. μ_i and σ_i are the mean and standard deviation of X_i. Therefore, each $\text{SND}(X_{ij})$ shows the distance of each input value of each data from the mean of the online input data. Accordingly, each W_{ij} as weight of X_{ij} is equal to $\text{SND}(X_{ij})$ as in Eq. (4), and the initialization of weights is not at random:

$$W_{ij} = \text{SND}(X_{ij}) \quad i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, m. \quad (4)$$

The weights of attributes of current input data X_i are inserted in the weights code book as the code words of X_i. The model considers a vector of the weights as the product of each attribute of the weights code book. The Prod vector consists of the components Prod_j for the attributes, which is computed by the product of the weights of each attribute of the code book. The parameter n is the number of received data nodes, which were trained by the model, and i is current number of the online input data node X_i; m is the number of attributes, and j is the current number of the attribute. Equations (5) and (6) show these relationships.

$$\text{Prod}_j = \text{Prod}_j \times W_{ij}, \quad (5)$$

$$\overrightarrow{\text{Prod}} = (\text{Prod}_1, \text{Prod}_2, \ldots, \text{Prod}_m). \quad (6)$$

3. Extracting the BMW vector: In the SOM, the weight of the code book that is nearest to the input vector is distinguished as the winner node and the *best matching unit*. In the same way, the DSFFNN learns the standard unique weight as BMW. The BMW vector is the global *geometric mean* (Vandesompele et al., 2002; Jacquier et al., 2003) vector of the code book of the nonrandom weights and is computed based on the gravity center of the current and last trained data nodes. In the DSFFNN method, the codebook of real weights is initialized by considering properties of input values directly and without using any random values or random parameters similar to the RSFFNN clustering method. The RSFFNN computes the standard weight one time through processing of all input

Table 1. *The online input data X*

			X			
Input Data Vector X_i	Attribute_1	Attribute_2	. . .	Attribute_m	Mean	Standard Deviation
X_i	X_{i1}	X_{i2}	. . .	X_{1m}	μ_i	σ_i

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

data instances in the input matrix; however, the DSFFNN computes the BMW based on the gravity center of the current and last trained data nodes, and with entrance of each online input data, the BMW is updated. The BMW vector is computed by Eqs. (7)–(9) as follows:

$$\mathrm{BMW}_j = \mathrm{Prod}_j{}^{1/n}, \tag{7}$$

$$\mathrm{BMW}_j = \mathrm{Round}\big(\big(\mathrm{BMW}_j/\mathrm{SumBMW}\big), 2\big), \tag{8}$$

$$\mathrm{SumBMW} = \textstyle\sum_{j=1}^{m} \mathrm{BMW}_j, \tag{9}$$

$$\overrightarrow{\mathrm{BMW}} = (\mathrm{BMW}_1, \mathrm{BMW}_2, \ldots, \mathrm{BMW}_m), \tag{10}$$

$$\text{Equations (8) and (9)} \Rightarrow \textstyle\sum_{j=1}^{m} \mathrm{BMW}_j = 1. \tag{11}$$

The parameter n is the number of received data nodes (contain the old data points and current online input data node X_i), m is the number of attributes, and j is the current number of attributes. Equation (7) shows the BMW_j is the global geometric mean of all W_{ij} of the attributes X_i. In Eq. (8), the parameter SumBMW is equal to the sum of the components of the BMW vector. As shown in Eq. (9), the model considers *Round* function with two digits of each BMW_j ratio to SumBMW, because in this way, the model is able to control the changing $\mathrm{BMW}_{j\mathrm{New}}$ ratio to $\mathrm{BMW}_{j\mathrm{Old}}$. This technique affects the low time and memory complexities of the model. Equation (11) shows the sum of components BMW is equal to one through Eqs. (8) and (9); therefore, we can understand the distribution of amounts of weights between attributes. Table 2 illustrates the code book of the weights and the process of extracting the BMW vector.

The main goal of the DSFFNN model is learning of the BMW vector as the criterion weight vector. The next stages will show how the threshold of current online data is computed, the current online input data is clustered easily based on the computed BMW vector, and consequently the network of clusters is updated.

4. Update EII: In this stage after learning the weights of attributes of online input data X_i, the model stores the mean and standard deviation of X_i, computed new BMW and Prod as the EII in the memory. Some data nodes as training data have class labels; therefore, in this phase, if current online input data has a class label, consequently, the model keeps it with other important information of the data in the memory. After clustering, the model will consider the class label and its related total threshold in order to semisupervised clustering of the data in the future stages.

Table 2. *The code book of the weight vectors and the BMW vector*

Code Book of Weight Vectors				
Weight Vector of X_i	$\mathrm{Attribute}_1$	$\mathrm{Attribute}_2$	...	$\mathrm{Attribute}_m$
Weight vector of X_1	W_{11}	W_{12}	...	W_{1m}
Weight vector of X_2	W_{21}	W_{22}	...	W_{2m}
⋮	⋮	⋮	⋮	⋮
Weight vector of X_n	W_{n1}	W_{n2}	...	W_{nm}
$\overrightarrow{\mathrm{Prod}}$	Prod_1	Prod_2	...	Prod_m
$\overrightarrow{\mathrm{BMW}}$	BMW_1	BMW_2	...	BMW_m

- **Fine-tuning:** The DSFFNN similar the RSFFNN clustering method applies the techniques of fine-tuning, but after each updating of the BMW process. As Asadi et al. (2014*a*) explained, in order to adapt the weights accurately to achieve better results of clustering the data points, two techniques of smoothing the weights and pruning the weak weights can be considered:
  1. Smoothing the weights: The speed, accuracy, and capability of the training of the FFNN clustering can be improved, by application of some techniques such as smoothing parameters and weights interconnection (Jean & Wang, 1994; Peng & Lin, 1999; Gui et al., 2001; Tong et al., 2010). In the Midrange technique as an accepted smoothing technique (Jean & Wang, 1994; Gui et al., 2001), the average of high weight components of the BMW vector is computed and considered as middle range (Midrange). The input data attributes with too high weight amounts may cause them to dominate the high thresholds and highly effect the clustering results. When some BMW_j are considerably higher than other components, the BMW_j can be smoothed based on the Midrange smooth technique. If the weights of some components of the BMW are higher than the Midrange, the model will reset their weights to equal the Midrange value.
  2. Data dimensionality reduction: The DSFFNN model can reduce the dimension of data by recognizing the weak weights BMW_j and deleting the related attributes. The weak weights that are close to zero are less effective on thresholds and the desired output. Hence, the weights can be controlled and pruned in advance.
- **Single-layer DSFFNN clustering:** The main section of the DSFFNN clustering model is a single-layer FFNN topology to cluster the online data by using normalized values and the components of the BMW vector. The proposed model is able to recluster all old data points by retrieving information from the EII. Two major tasks of the DSFFNN model in this section are to cluster the data points dynamically and after learning to assign class labels and semicluster the data.
- **DUFFNN clustering:** The DSFFNN clustering carried out during one training iteration is based on nonrandom weights, without any updating weight, activation function,

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

and error function such as mean square error. The threshold or output is computed by using normalized values of the input data node and the fetched BMW vector from EII. When the DSFFNN model learns a huge amount of data for clustering, the new BMW is changed slowly and is close to the last computed BMW. IF components of the new BMW_j are equal to the components of the old BMW_j, the model just clusters X_i. In this case, computing the threshold of each attribute of the current online input data X_{ij} and *total threshold* of the online input data vector X_i are computed by Eqs. (12) and (13).

$$T_{ij} = X_{ij} \times \mathrm{BMW}_j, \tag{12}$$

$$\mathrm{TT}_i = \sum_{j=1}^{m} X_{ij} \times \mathrm{BMW}_j \quad \text{or} \quad \mathrm{TT}_i = \sum_{j=1}^{m} T_i, \tag{13}$$

$$\mathrm{Memory(EII)} \leftarrow \mathrm{Store(Class\ label\ of}\ X_i, \mathrm{TT}_i). \tag{14}$$

As Eq. (14) shows, if the current online input data X_i has a class label, the DSFFNN model stores the computed total threshold as related TT_i to this class label in the memory. During the semisupervised feedforward clustering stage, the model needs this class label. If the new BMW vector is different from the last BMW, the model considers the BMW_{New} vector based on the feature of Hebbian learning and reclusters all old data points by retrieving information from the EII, based on their related total threshold during one iteration. In this case, the model considers Eqs. (15)–(20), respectively, and checks which components of BMW_{New} are changed. Consequently, the model fetches the related W_{fj}, σ_f, μ_f of each component BMW_{jOld} and retrieves related data node D_{fj} from EII based on Eqs. (3) and (4), and computed related threshold of the data node D_{fj}. By considering the total threshold of the D_{fj} as the EII from the memory and replacing the amounts of old threshold of attribute j of D_{fj} as $T_{f\mathrm{Old}}$ and new threshold of attribute j of D_{fj} as $T_{f\mathrm{New}}$, the data node D_{fj} lies in the special place of axis of the total threshold and suitable cluster.

$$\mathrm{Fetch}(W_{fj}, \sigma_f, \mu_f) \leftarrow \mathrm{Memory(EII)}, \tag{15}$$

$$\mathrm{D}_{fj} = (\mathrm{W}_{fj} \times \sigma_f) + \mu_f, \tag{16}$$

$$T_{f\,\mathrm{Old}} = D_{fj} \times \mathrm{BMW}_{j\mathrm{Old}}, \tag{17}$$

$$\mathrm{TT}_f = \mathrm{TT}_f - \mathrm{T}_f\mathrm{Old}, \tag{18}$$

$$T_{f\,\mathrm{New}} = D_{fj} \times \mathrm{BMW}_{j\mathrm{New}}, \tag{19}$$

$$\mathrm{TT}_f = \mathrm{TT}_f + \mathrm{T}_{f\mathrm{New}}. \tag{20}$$

Therefore, it is not necessary to compute and update all thresholds of all old data node attributes. The single-layer DSFFNN computes the total threshold of current input data node X_i, updates the total thresholds of old data nodes, and lists thresholds and related class labels. Consequently, the model reclusters all received data nodes. Figure 7 illustrates how the data nodes are clustered by the DSFFNN model.

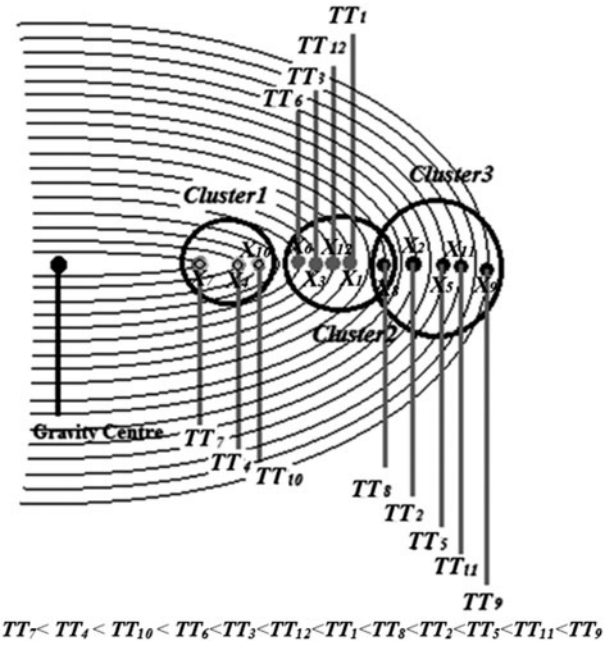


Fig. 7. An example for clustering the data nodes to three clusters by the dynamic semisupervised feedforward neural network.

As shown in Figure 7, each TT_i is the total threshold vector of each data point ratio to the gravity center of the data points. Therefore, each data vector takes its own position on the thresholds axis. The online input data X_i, based on its exclusive TT_i, lies on the axis, respectively. Each data point has an exclusive and individual threshold. If two data points possess an equal total threshold, but are in different clusters, clustering accuracy is decreased because of the error of the clustering method. The DSFFNN considers the data points with close total thresholds into one cluster. Figure 7 is an example of clustering data points into three clusters. Figure 8 is an example of clustering the Iris data points by the DSFFNN clustering. In Figure 8a and b, the Iris data from the UCI repository is clustered to three clusters based on a unique total threshold of each data point by the DSFFNN method. Data point 22 has $TT_{22} = 0.626317059$ and lies inside of cluster 2 or the cluster of the Iris Versicolour.

- **Pruning the noise:** The DSFFNN distinguishes isolated data points through the solitary total thresholds. The total threshold of an isolated data point is not close to the total thresholds of other clustered data points. Therefore, the isolated data point lies out of the locations of other clusters. The proposed DSFFNN method sets apart these data points as noise and removes them. The action of removing the noise causes high speed and clustering accuracy with low memory usage of the network in big data sets.
- **Improving the result of the DUFFNN clustering by using K-step activation function:** As mentioned in Section 1, there is a technique of converting a clustering method to semisupervised clustering by considering some

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

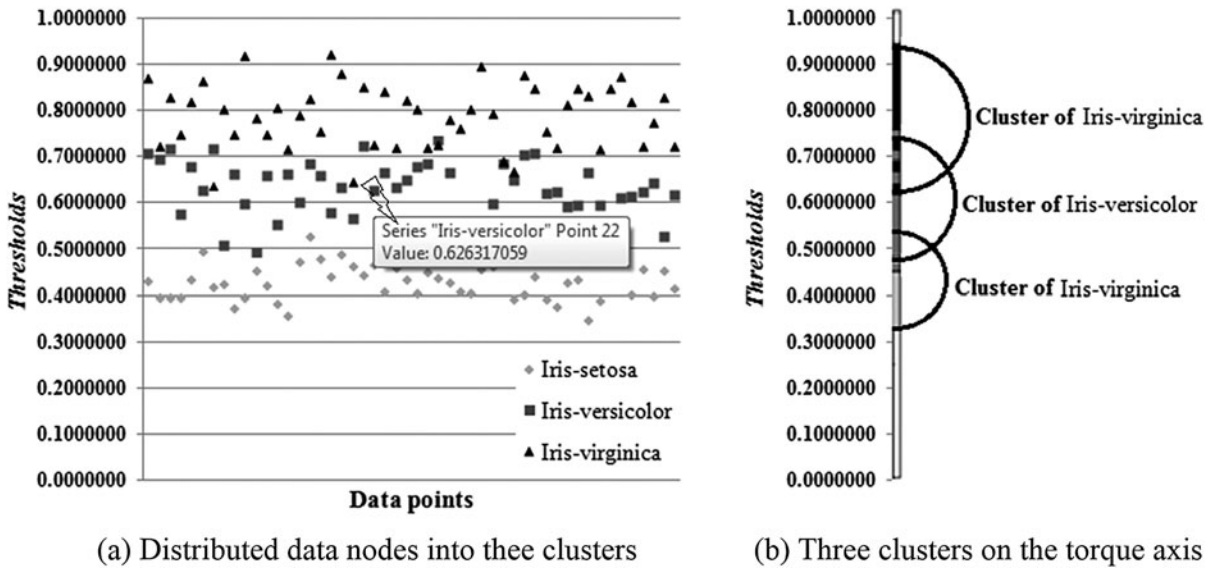


Fig. 8. The outlook of clustering the online Iris data to three clusters based on a unique total threshold of each data point by the dynamic semisupervised feedforward neural network.

constraint or user guides as feedback from users. K-step activation function (Alippi et al., 1995) or threshold function is a linear activation function for transformation of input values. This kind of function is limited with K values based on the number of classes of the data nodes, and each limited domain of thresholds refers to the special output value of the K-step function. The *binary-step* function is a branch of the K-step function for two data classes 0 and 1. It is often used in single-layer networks. The function $g(TT_i)$ is the K-step activation function for the transformation of TT_i and output will be 0 or 1 based on the threshold TT_i. After clustering the data node by the DUFFNN, the proposed method considers the class label as a constraint in order to improve the accuracy of the result of clustering. If the current online input data X_i is a training data and has a class label, the model fetches the class label and related total threshold of X_i from memory as EII and assigns this class label to the data nodes with similar threshold and updates the data nodes by using the K-step function. Consequently, based on K class labels and related exclusive thresholds in the EII, the proposed method expects K clusters and for each cluster considers a domain of thresholds. By considering the cluster results of the last phase, if there are some data points with a related threshold in each cluster but without a class label (unknown or unobserved data), the model supposes that these data nodes have the same class label as their clusters. During the process of future online data nodes, when the model updates data nodes, the class labels of these unknown data nodes will be recognized and adjusted for the suitable cluster if necessary. Hence, the class label of unknown data is predicted in two ways: during clustering by considering the related cluster, which the data lies in there, and by using the K-step function based on relationships between thresholds and class labels in the EII. Therefore, the DSFFNN clustering method similar to the RSFFNN applies the "trial and error" method, in order to predict the class label for the unobserved data. The class label of each unknown observation is signed and predicted based on the K-step function and the related cluster and thresholds domain of the cluster where the input instance is. The accuracy of the results of the DSFFNN clustering is measured by F-measure function with 10-fold cross-validation, and the accuracy will show the validation of the prediction. Furthermore, the method updates the number of clusters and the density of each cluster by using class labels. Figure 9 shows an example for clustering a data set to two clusters (Part A), and improving the result by DSFFNN clustering (Part B).

3. EXPERIMENTAL RESULTS AND COMPARISON

All of the experiments were implemented in Visual C#.Net in Microsoft Windows 7 Professional operating system with a 2 GHz Pentium processor. To evaluate the performance of the DSFFNN clustering model, a series of experiments on several related methods and data sets were used.

3.1. Data sets from UCI Repository

The Iris, Spambase, Musk2, Arcene, and the Yeast data sets from the University of California at Irvine (UCI) Machine Learning Repository (Asuncion & Newman, 2007) are selected for evaluation of the proposed model as shown in Table 3. As mentioned in the UCI Repository, the data sets are remarkable because most conventional methods do not process well on these data sets. The type of the data set is

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

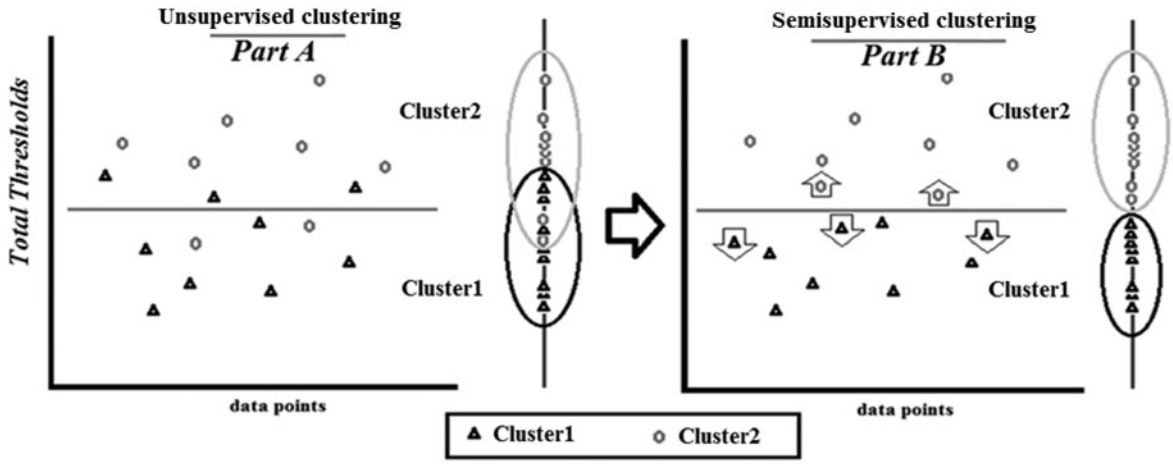


Fig. 9. The outlook of the dynamic semisupervised feedforward neural network clustering method.

Table 3. *The information of selected data sets in this study from the UCI Repository*

	Characteristics		Number of		
Data Set	Data Set	Attribute	Instances	Attributes	Classes
Iris	Multivariable	Real	150	4	Three classes: Iris Setosa, Iris Versicolour, and Iris Virginica
Spambase	Multivariable	Integer–real	4601	57	Two classes: spam and nonspam
Musk2	Multivariable	Integer	6598	168	Two classes: musk or nonmusk molecules
Arcene	Multivariable	Real	900	10,000	Two classes: cancer patients and healthy patients
Yeast	Multivariable	Real	1484	8	Ten classes

the source of clustering problems, such as estimation of the number of clusters and the density of each cluster; or in other words, recognizing similarities of the objects and relationships between attributes of the data set. Large and high-dimensional data creates some difficulties of clustering, especially in real dynamic environments, as mentioned in the Section 1. For experimentation, the speed of processing was measured by the number of epochs. The accuracy of the methods is measured through the number of clusters and the quantity of *correctly classified nodes* (CCNs), which shows the total nodes and density with the correct class in the correct related cluster in all clusters created by the model. The CCNs are the same as the true positive and true negative nodes. Furthermore, the accuracy of the proposed method is measured by *F*-measure function for 10-folds of the test set. The precision of computing was considered with 15 decimal places for more dissimilar threshold values. For simulation of the online real environment, every time just one instance of the training or test data is selected randomly and is processed by the models.

3.1.1. Iris data set

The Iris plants data set was created by Fisher (1950; Asuncion & Newman, 2007). The Iris can be classified into Iris Setosa, Iris Versicolour, and Iris Virginica. Figure 10 shows the final computed BMW vector of the received Iris data points by the DSFFNN clustering method.

The total thresholds of the received Iris data were computed on the basis of the final BMW vector. As shown in Figure 11, three clusters can be recognized.

Table 4 shows the comparison of the results of the proposed DSFFNN method with the results of some related methods for the Iris data (Asadi et al., 2014*a*, 2014*b*). As shown in Table 4, the ESOM clustered the Iris data points with 144 CCNs, 96.00% density of the CCNs and 96.00% accuracy by the *F*-measure after 1 epoch during 36 ms. The DSOM clustered the data points with 135 CCNs and 90.00% density of the CCNs and 90.00% accuracy by the *F*-measure after 700 epochs during 39 s and 576 ms. The semi-ESOM clustered the Iris data points with 150 CCNs, 100% density of the CCNs and 100% accuracy by the *F*-measure. The DUFFNN clustering method clustered this data set with 146 CCNs, 97.33% density of the CCNs and 97.33% accuracy by *F*-measure. The BPN, as a supervised FFNN classification model, learned this data set after 140 epochs with the accuracy of 94.00% by using *F*-measure. As Table 4 shows, the DUFFNN clustering method has superior results. All clustering methods show three clusters for this data set. In order to get a better result of the DUFFNN clustering method,

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

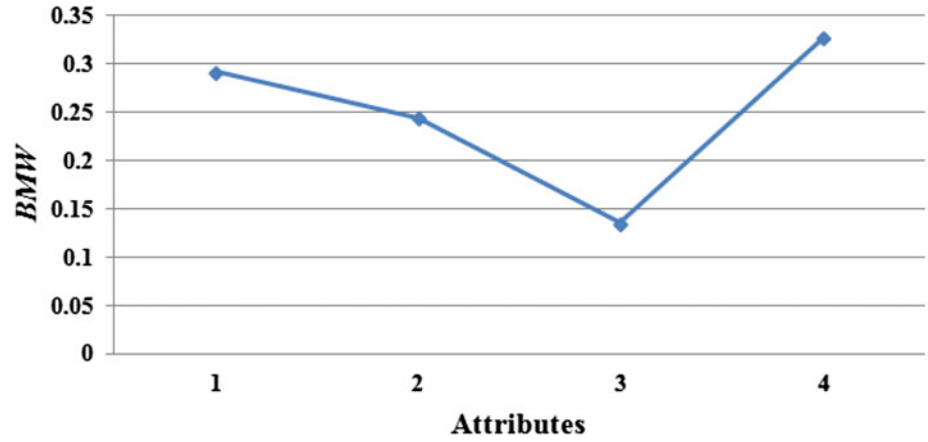


Fig. 10. The final computed best matching weight vector components of the received Iris data by using the dynamic unsupervised feedforward neural network method.

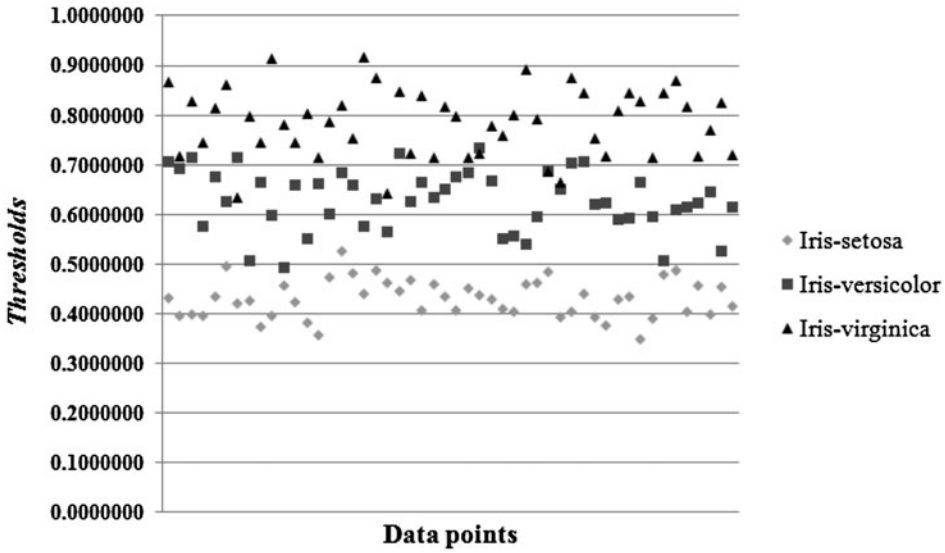


Fig. 11. The clusters of the received data points from the Iris data by using the dynamic unsupervised feedforward neural network method.

Table 4. *Comparison of the clustering results on the Iris data points by the DSFFNN and some related methods*

Methods	CCN	Density of CCN (%)	Accuracy by *F*-Measure (%)	Epoch	CPU Time (ms)
ESOM	144	96.00	96.00	1	36
DSOM	135	90.00	90.00	700	39 s (576)
Semi-ESOM	150	100	100	1	36
DUFFNN	146	97.33	97.33	1	26.68
DSFFNN	150	100	100	1	26.68

we implemented the DSFFNN by using the Iris data. The results of the DSFFNN show 150 CCNs, 100% density of the CCNs and 100% accuracy by *F*-measure with 1 epoch for training just during 26.68 ms.

3.1.2. *Spambase data set*

The Spambase E-mail data set is created by Mark Hpkins, Erik Reeber, George Forman, and Jaap Suermondt (Asuncion & Newman, 2007). The Spambase data set can be classified

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

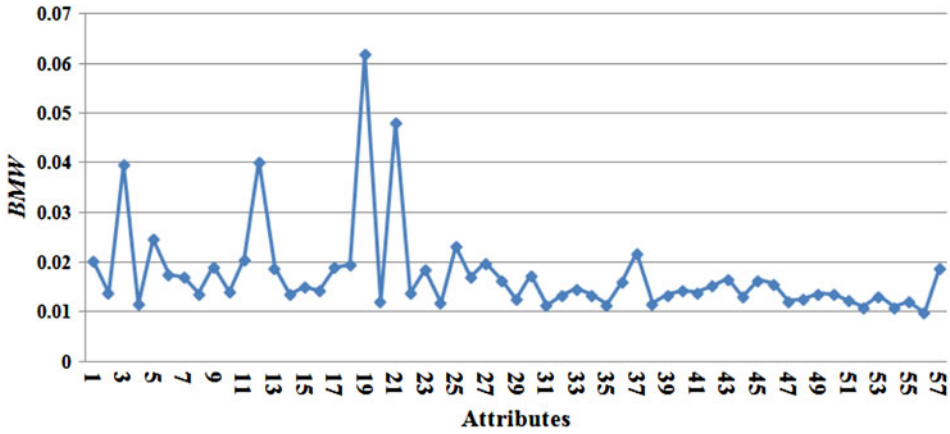


Fig. 12. The final computed best matching weight vector components of the received Spambase data by using the dynamic supervised feedforward neural network method.

into spam and nonspam. Figure 12 shows the final computed BMW vector of the received Spambase data by the DSFFNN clustering method.

The total thresholds of input data were computed based on the BMW vector. Consequently, the input data were clustered. As shown in Figure 13, two clusters are recognized.

Table 5 shows the comparison of the results of the proposed DSFFNN method with the results of some related methods for the Spambase data (Asadi et al., 2014*a*, 2014*b*). As shown in Table 5, the ESOM clustered the Spambase data points with 2264 CCNs, 49.21% density of the CCNs and 57.85% accuracy by the *F*-measure after 1 epoch during 14 min, 39 s, and 773 ms. The DSOM clustered the data points with 2568 CCNs and 55.83% density of the CCNs and 62.78% accuracy by the *F*-measure after 700 epochs during 33 min,, 27 s, and 90 ms. The semi-ESOM clustered the Spambase data points with 2682 CCNs, 58.29% density of the CCNs and 65.03% accuracy by the *F*-measure. The DUFFNN clustering method clustered this data set with 3149 CCNs, 68.44% density of the CCNs and 73.96% accuracy by *F*-measure after 1 epoch during 35 s and 339 ms. The BPN learned this data set after 2000 epochs with the accuracy of 79.50% by using *F*-measure. As Table 5 shows, the DUFFNN clustering method has superior results. All clustering methods show two clusters for this data set. The results of the DSFFNN show 4600 CCNs, 99.97% density of the CCNs and 99.96% accuracy by *F*-measure with 1 epoch for training.

3.1.3. *Musk2 data set*

The Musk2 data set (version 2 of the musk data set) is selected from the UCI Repository. The data set was created by the Artificial Intelligence group at the Arris Pharmaceutical Corporation, and describes a set of musk or nonmusk molecules (Asuncion & Newman, 2007). The goal is to train to

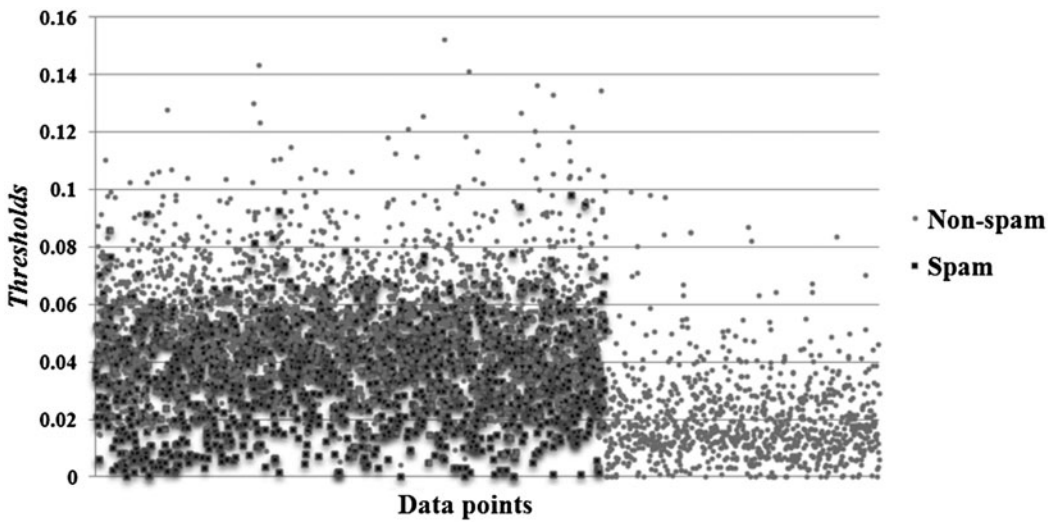


Fig. 13. The clusters of the received data points of the Spambase data by using the dynamic supervised feedforward neural network method.

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

Table 5. *Comparison of the clustering results on the Spambase data points by the DSFFNN and related methods*

Methods	CCN	Density of CCN (%)	Accuracy by *F*-Measure (%)	Epoch	CPU Time (ms)
ESOM	2264	49.21	57.85	1	14 min, 39 s (773)
DSOM	2568	55.83	62.78	700	33 min, 27 s (90)
Semi-ESOM	2682	58.29	65.03	1	14 min, 39 s (773)
DUFFNN	3149	68.44	73.96	1	35 s (339)
DSFFNN	4600	99.97	99.96	1	35 s (339)

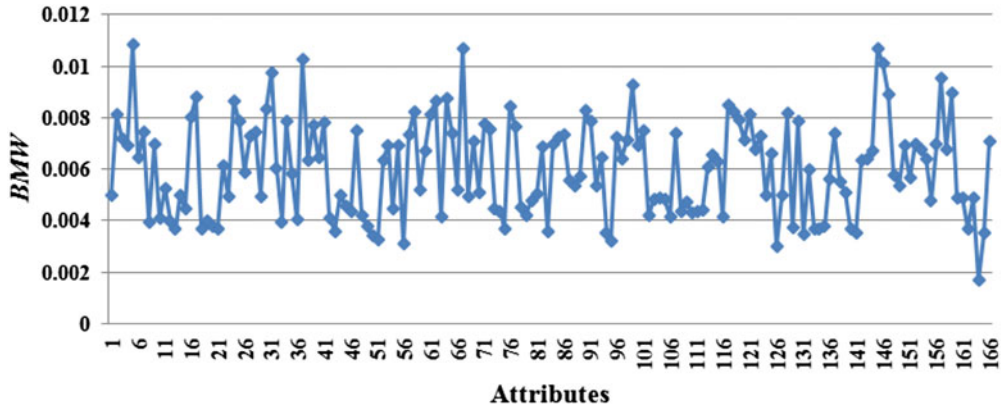


Fig. 14. The final computed best matching weight vector components of the received Musk2 data by using the dynamic supervised feedforward neural network method.

predict whether new molecules will be musk or nonmusk based on their features. Figure 14 shows the final computed BMW vector of the received Musk2 data by the DSFFNN clustering method.

The total thresholds of input data were computed based on the BMW vector. Consequently, the input data were clustered. As shown in Figure 15, two clusters are recognized.

Table 6 shows the comparison results of the proposed DSFFNN method with the results of some related methods for the Musk2 data (Asadi et al., 2014*a*, 2014*b*). As Table 6 shows, the ESOM clustered the Musk2 data set with 4657 CCNs, 70.58% density of the CCNs and 56.40% accuracy by *F*-measure after 1 epoch during 28 min and 1 ms. The DSOM clustered the Musk2 data set with 3977 CCNs, 60.28% density of the CCNs and 41.40% accuracy by *F*-measure after 700 epoch during 41 min, s, and 633 ms. The semi-ESOM clustered the Musk2 data set with 5169 CCNs, 78.34% density of the CCNs and 87.19% accuracy by *F*-measure. The DUFFNN clustering method clustered this data set with 4909 CCNs, 74.40% density of the CCNs and 84.86% accuracy by *F*-measure after 1 epoch during 27 s and 752 ms. The BPN learned this data set after 100 epochs with 67.00% accuracy by *F*-measure. All clustering methods show two clusters for this data set. The results of the DSFFNN show 6598 CCNs, 100% density of the CCNs and 100% accuracy by *F*-measure.

3.1.4. Arcene data set

The Arcene data set was collected from two different sources: the national cancer institute and the eastern Virginia medical school (Asuncion & Newman, 2007). All data were earned by merging three mass-spectrometry data sets to create training and test data as a benchmark. The training and validation instances include patients with cancer (ovarian or prostate cancer) and healthy patients. Each data set of training and validation contains 44 positive samples and 56 negative instances with 10,000 attributes. We considered the training data set and validation data set with 200 total instances together as one set. The Arcene data set can be classified into cancer patients and healthy patients. Arcene's task is to distinguish cancer versus normal patterns from mass-spectrometric data (Asuncion & Newman, 2007). This data set is one of five data sets of the Neural Information Processing Systems 2003 feature selection challenge (Guyon, 2003; Guyon & Elisseeff, 2003). Therefore, most current existing papers are subject to the best selection of attributes in order to reduce the dimension of the arcane data set with better accuracy, CPU time usage, and memory usage. In this research, we cluster the Arcene data by using the DUFFNN and the DSFFNN clustering methods with just fine-tuning without selection of special attributes. After final updating of the codebook of nonrandom weights and extracting final BMW, Figure 16 shows two clus-

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

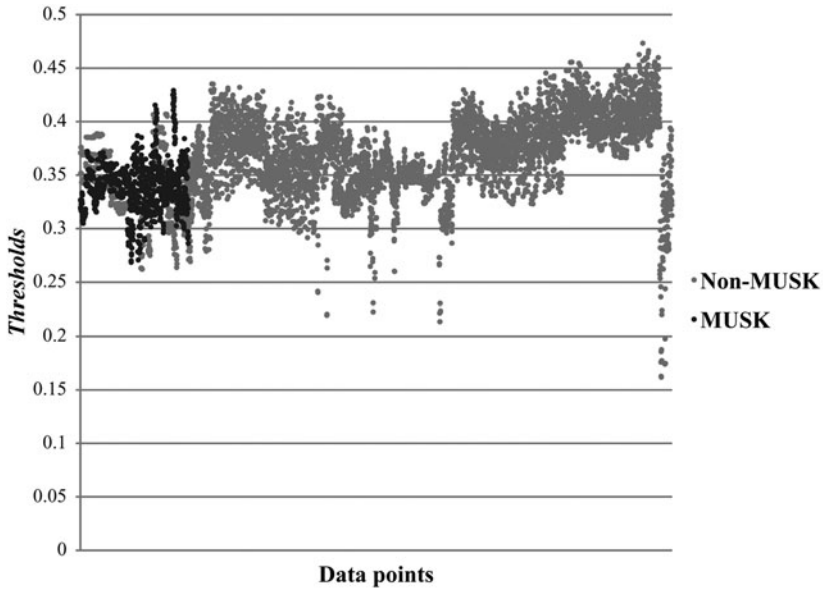


Fig. 15. The clusters of the received data points from the Musk2 data by the dynamic supervised feedforward neural network method.

Table 6. *Comparison of the clustering results on the Musk2 data points by the DSFFNN and some related methods*

Methods	CCN	Density of CCN (%)	Accuracy by *F*-Measure (%)	Epoch	CPU Time (ms)
ESOM	4657	70.58	56.40	1	28 min (1)
DSOM	3977	60.28	41.40	700	41 min, 1 s (633)
Semi-ESOM	5169	78.34	87.19	1	28 min (1)
DUFFNN	4909	74.40	84.86	1	27 s (752)
DSFFNN	6598	100	100	1	27 s (752)

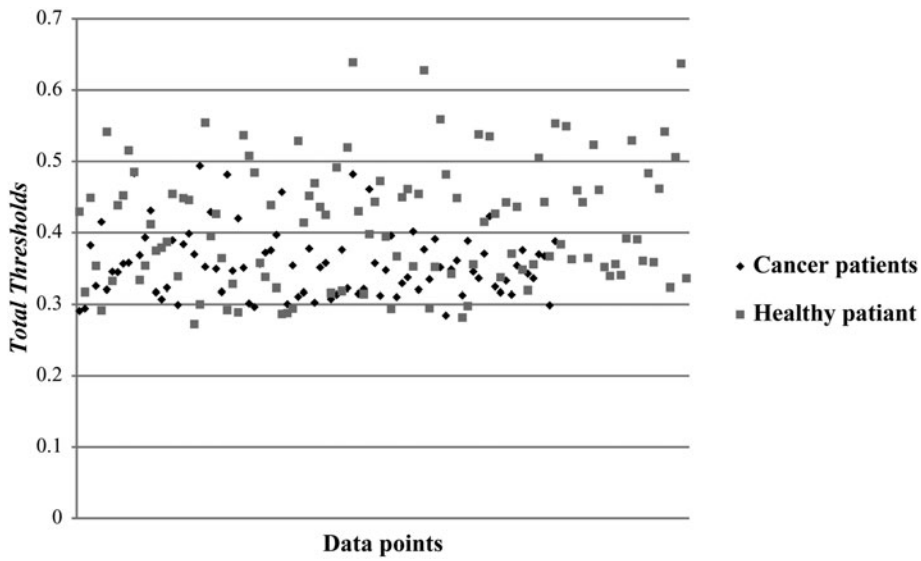


Fig. 16. The clusters of the received data points of the Arcene data by using the dynamic supervised feedforward neural network method.

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

Table 7. *Comparison of the clustering results on the Arcene data points by the DSFFNN and some related methods*

Methods	CCN	Density of CCN (%)	Accuracy by *F*-Measure (%)	Epoch	CPU Time (ms)
ESOM	96	48.00	53.57	1	56 s (998)
DSOM	94	47.00	52.68	20	43 min, 12 s (943)
Semi-ESOM	121	60.50	63.93	1	56 s (998)
DUFFNN	124	62.00	66.07	1	13 s (447)
DSFFNN	200	100	100	1	13 s (447)

ters of the received data points from the Arcene data was recognized by the DUFFNN clustering method.

Table 7 shows the comparison the proposed DSFFNN method with some related methods for the Arcene data. The ESOM clustered the Arcene data set with 96 CCNs, 48.00% density of the CCNs and 53.57% accuracy by *F*-measure after 1 epoch during 56 s and 998 ms. The DSOM clustered the Arcene data set with 94 CCNs, 47.00% density of the CCNs and 52.68% accuracy by *F*-measure after 20 epochs during 43 min, 12 s, and 943 ms. The semi-ESOM clustered the Arcene data set with 121 CCNs, 60.50% density of the CCNs and 63.93% accuracy by *F*-measure. The DUFFNN clustering method clustered this data set with 124 CCNs, 62.00% density of the CCNs and 66.07% accuracy by *F*-measure after 1 epoch just during 13 s and 447 ms. All clustering methods show two clusters for this data set. The results of the DSFFNN show 200 CCNs, 100% density of the CCNs and 100% accuracy by *F*-measure. The DSOM is not a suitable method to implement for clustering of this data set through its time and memory complexities. Recently, Mangat and Vig (2014) reported classification of the Arcene data set by several classification methods such as *K*-nearest neighbor (K-NN). K-NN is a supervised classifier that is able to learn by analogy and performs on *n*-dimensional numeric attributes (Dasarthy, 1990). Given an unknown instance, K-NN finds K instances in the training set that are closest to the given instance pattern and predicts one or an average of class labels or credit rates. Unlike BPN, K-NN assigns equal weights to the attributes. The K-NN (K = 10) was able to classify the Arcene data set with 77.00% accuracy by *F*-measure after several epochs and 10 times running the method. Comparison of the results of the DSFFNN clustering method with the results of other related methods shows the superior results of the DSFFNN clustering method.

3.1.5. *Yeast data set*

The Yeast data set is obtained from the UCI Repository. The collected data set is reported by Kentai Nakai from the Institute of Molecular and Cellular Biology, University of Osaka (Asuncion & Newman, 2007). The aim is to predict the cellular localization sites of proteins. The Yeast data set contains 1484 samples with eight attributes. The classes are cytosolic, nuclear, mitochondrial, and membrane protein: no N-terminal signal, membrane protein: uncleaved signal and membrane protein: cleaved signal. Extracellular, vacuolar, peroxisomal, and endoplasmic reticulum lumen (Asuncion & Newman, 2007). In this research, we cluster the Yeast data by using the DSFFNN clustering method taking 1 s and 373 ms training time. Table 8 shows the speed of processing based on the number of epochs and the accuracy based on the density of the CCNs in the Yeast data set by the DSFFNN method.

In Table 8, based on the results of the experiment, the ESOM clustered the Yeast data with 435 CCNs, 29.31% density of the CCNs and 17.63% accuracy by *F*-measure after 1 epoch during 37 s and 681 ms. The DSOM clustered the Yeast data with 405 CCNs, 27.29% density of the CCNs and 24.53% accuracy by *F*-measure after 20 epoch during 11 s and 387 ms. The semi-ESOM clustered this data with 546 CCNs, 36.79% density of the CCNs and 20.72% accuracy by *F*-measure. The DUFFNN clustering method clustered

Table 8. *Comparison of the clustering results on the Yeast data points by the DSFFNN and some related methods*

Methods	CCN	Density of CCN (%)	Accuracy by *F*-Measure (%)	Epoch	CPU Time (ms)
ESOM	435	29.31	17.63	1	37 s (681)
DSOM	405	27.29	24.53	20	11 s (387)
Semi-ESOM	546	36.79	20.72	1	37 s (681)
DUFFNN	426	28.71	27.25	1	1 s (373)
DSFFNN	1484	100	100	1	1 s (373)

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

Type	SG	Pressure	Temprature
Exp	0.6	6.83E+01	6.58E+03
Exp	0.6	6.57E+01	5.97E+03
Exp	0.6	5.15E+01	3.83E+03
Exp	0.6	4.77E+01	3.09E+03
Exp	0.6	3.35E+01	1.14E+03
ANN	0.6	6.87E+01	6.55E+03
ANN	0.6	6.05E+01	5.07E+03
ANN	0.6	5.79E+01	
ANN	0.6		
ANN	0.6		

Fig. 17. A sample of the clathrate hydrate formation data set.

this data set with 426 CCNs, 28.71% density of the CCNs and 27.25% accuracy by *F*-measure after 1 epoch just during 1 s and 373 ms. The DSFFNN clustering method clustered this data set with 1484 CCNs, 100% density of the CCNs and also by *F*-measure. Several studies reported the difficulty of clustering or classification of the Yeast data set. As Longadge et al. (2013) reported, classification of the Yeast data set was done by several classification methods such as K-NN. The K-NN ($K = 3$) was able to classify the Yeast data set with 0.11% accuracy by *F*-measure after several epochs and times running the method. In addition, Ahirwar (2014) reported the *K*-means was able to classify the Yeast data set with 65.00% accuracy by *F*-measure after several epochs.

3.2. Prediction of clathrate hydrate temperature

Clathrate hydrates or gas hydrates are crystalline water-based solids that physically appear like ice. In clathrate hydrates, molecules such as gases form a framework that traps other molecules. Otherwise, the grille structure of the clathrate hydrate breaks into formal ice crystal structure or liquid water. For example, the low molecular weight gases, such as H_2, O_2, N_2, CO_2, and CH_2, often form hydrates at suitable temperatures and pressures. Cathrate hydrates are not formally chemical compounds, and their formation and decomposition are first-order phase transitions. The details of the formation and decomposition mechanisms at a molecular level are still critical issues to research. In 1810, Humphry Davy (1811) investigated and introduced the clathrate hydrate. In hydrate studies, two main areas have attracted the attention: prevention or elimination of hydrate formation in pipelines, and feasibility examination of the gas hydrate technological applications. In studying any of these areas, the issue that should be addressed first is hydrate thermodynamic behavior. For example, thermodynamic conditions of the hydrate formation are often detected in pipelines, because the clathrate crystals can be accumulated and plug the line. Many researchers have tried to prognosticate this phenomena by using various predictive methods and understanding the conditions of hydrate formation (Eslamimanesh, 2012; Ghavipour, 2013; Moradi, 2013). One of these methods is a hydrate formation temperature estimation by applying neural network methods (Kobayashi et al., 1987; Shahnazar & Hasan, 2014; Zahedi et al., 2009).

Kobayashi et al. (1987) selected six different specific gravities and experimentally considered the relationship between the variables of pressure and temperature of gas when the clathrate hydrate is formed. They collected 203 data points, with 136 data points as a training set and 67 data points as a test set. Zahedi et al. (2009) applied a multilayer perceptron neural network classification model with seven hidden layers, in the Matlab neural network toolbox (Mathworks, 2008), to predict hydrate formation temperature for each operating pressure, and compared their results with the results of statistical methods by Kobayashi et al. (1987). Their results showed that their neural network method had better results.

In order to evaluate the DUFFNN model performance, a data set with 299 experimental data points in a temperature range of 33.7–75.7°F and a pressure of 200–2680.44 psi are considered (Shahnazar & Hasan, 2014). Figure 17 shows a sample of this data set.

The DUFFNN clustering results are compared with the ANN classification result of Zahedi et al. (2009), and laboratory experimental results of Kobayashi et al. (1987), which is available in the literature, as shown in Figure 18.

Figure 18 shows that the result of the DUFFNN clustering is closer than the Matlab-ANN classification to the laboratory experience. The DUFFNN clustering model proved to predict the hydrate formation temperature with more than 98% accuracy. Therefore, the results of Figure 18 show that the proposed clustering is able to respond to the unobserved samples that lie in the curve of the DUFFNN clustering with more than 98% accuracy.

3.3. Breast cancer data set from the University of Malaya Medical Center (UMMC)

The data set was collected by the UMMC, Kuala Lumpur, from 1992 until 2002 (Hazlina et al., 2004; Asadi et al., 2014*a*). As shown in Table 9, the data set was divided into nine subsets based on the interval of survival time: first year, second year, . . . , ninth year.

The DSFFNN model was implemented on each data set by considering the class labels. Figure 19 shows the sample of breast cancer data set from the UMMC.

As Table 10 shows, the breast cancer data set contains 13 attributes. The number of input data in the data set is 827, the number of attributes is 13 continuous and 1 attribute for showing the binary class in two cases of alive or dead. The used breast cancer data set from the UMMC has class labels of "0" for alive and "1" for dead as constraints.

Table 11 shows the results of the implementation of the DSFFNN clustering model on the UMMC breast cancer data set. Table 10 contains of the number of data nodes of

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

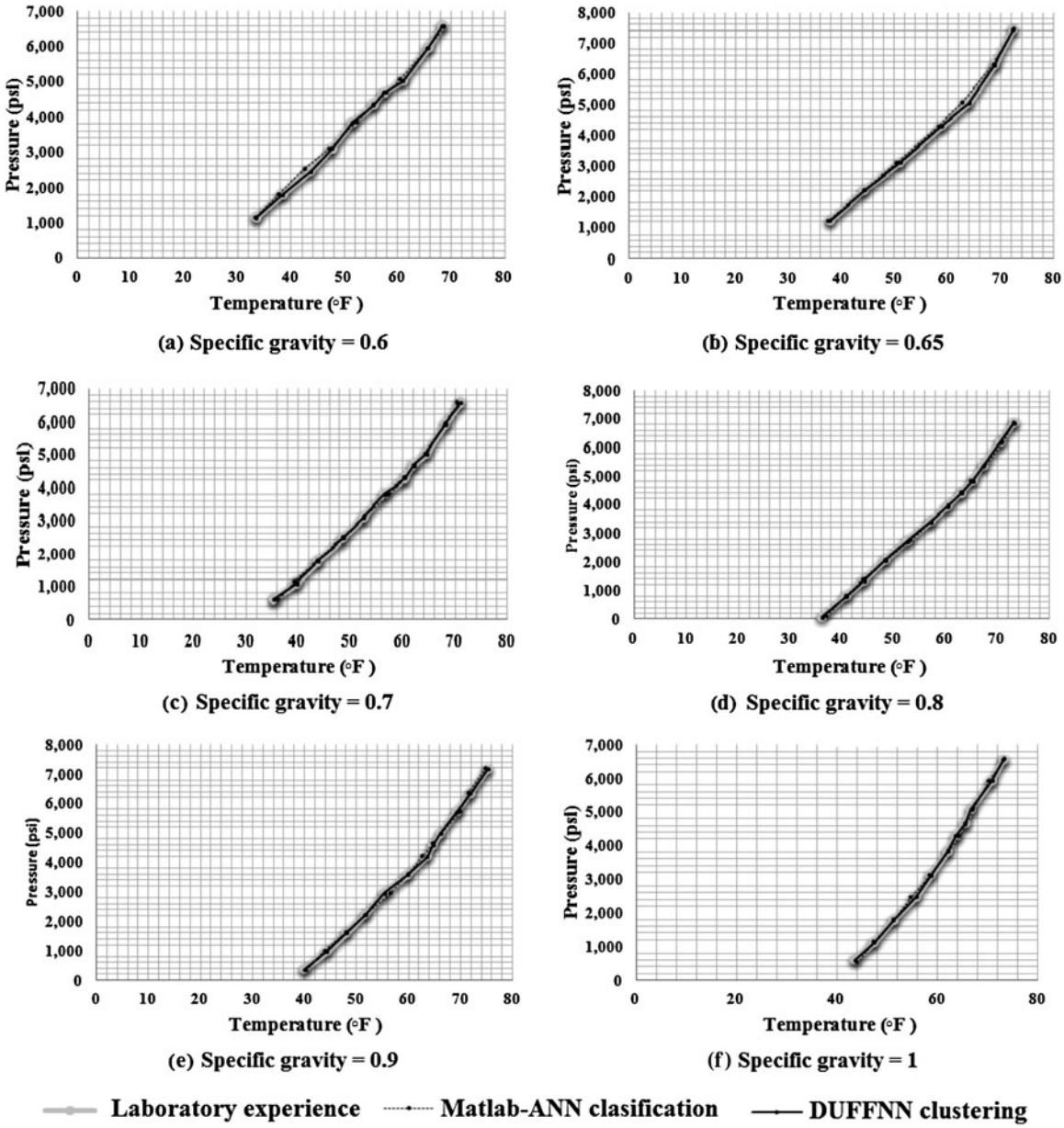


Fig. 18. Comparison of the laboratory experience with the results of the Matlab–artificial neural network classification and the dynamic unsupervised feedforward neural network clustering methods.

each subset; training time per millisecond for each subset during one epoch; and the accuracy of the DSFFNN clustering of each subset of the UMMC breast cancer data set based on the *F*-measure with 10-folds of the test set.

Performances of the ESOM and DSOM on the UMMC breast cancer data set are not suitable and efficient; therefore, we compared our results with some methods as reported by Asadi, Asadi, et al. (2014). Table 11 shows that the training process of the proposed DSFFNN method for each subset of the UMCC breast cancer data set took one epoch between 22.95 and 1 min, 05 s, and 2 ms of CPU time; the accuracies of the DSFFNN for the breast cancer subdata sets were between 98.14 and 100%. We considered the SOM using BPN as a hybrid method for supervised clustering of each subset (Asadi et al., 2014*a*). The SOM clustered each subset of the UMMC breast cancer data set after 20 epochs. The BPN model fine-tuned the codebook of weights of unfolding the SOM method instead of random weights. The training process in the BPN was 25 epochs. The results of the hybrid method of the SOM-BPN are shown in Table 12 for every subset. In addition, the principal component analysis (PCA; Jolliffe, 1986) was considered as a preprocessing technique for dimension reduction and used by the BPN model (Asadi et al., 2014*a*). The PCA is a classical multivariate data analysis method that is useful in linear feature extraction and data compression. Table 12 shows the result of the PCA-BPN hy-

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

Table 9. *The nine subsets of observed data of breast cancer from UMMC based on the interval of survival time*

Treatment Year	Year 1	Year 2	Year 3	. . .	Year 8	Year 9
1993	Data from 1993 to 1994	Data from 1993 to 1995	Data from 1993 to 1996	. . .	Data from 1993 to 2001	Data from 1993 to 2002
1994	Data from 1994 to 1995	Data from 1994 to 1996	Data from 1994 to 1997	. . .	Data from 1994 to 2002	
1995	Data from 1995 to 1996	Data from 1995 to 1997	Data from 1995 to 1998	. . .		
. . .	. . .	. . .	. . .			
2000	Data from 2000 to 2001	Data from 2000 to 2002				
2001	Data from 2001 to 2002					

AGE	RACE	STG	T	N	M	LN	ER	GD	PT	AC	AR	AT
66	I	1	1	0	0	0	X	X	M	N	Y	Y
66	I	2b	2	1	0	1	N	X	M	N	N	Y
43	I	2a	2	0	0	0	X	X	L	N	Y	Y
30	M	2b	2	1	0	2	X	X	L	Y	Y	N
65	C	4	2	x	1	X						
41	M	3a	3									
55	C	2b										
36	C											

Fig. 19. The sample of breast cancer from the University of Malaya Medical Center data set.

brid model for every subset of the UMMC breast cancer data set. The PCA spent the time of the CPU for dimension reduction of the data, and the BPN used the output of the PCA for classification after several epochs.

The results of Table 12 show the accuracies of implementation of the PCA-BPN model for the breast cancer data set were between 62% and 99%, and the accuracies of implementation of the SOM-BPN model for each subset of the breast cancer data set were between 71% and 99%. Furthermore, the training process for each subset of the UMMC breast cancer data set by the RSFFNN clustering took one epoch between 13.7 and 43 s of CPU time, and the accuracies of the RSFFNN for these subdata sets were between 98.29% and 100%. Table 12 shows the DSFFNN has desirable results.

Table 10. *The information of the UMMC Breast Cancer data set attributes*

Attributes	Attribute Information
AGE	Patient's age in year at time first diagnosis
RACE	Ethnicity (Chinese, Malay, Indian, and others)
STG	Stage (how far the cancer has spread anatomically)
T	Tumor type (the extent of the primary tumor)
N	Lymph node type (amount of regional lymph node involvement)
M	Metastatic (presence or absence)
LN	Number of nodes involved
ER	Estrogen receptor (negative or positive)
GD	Tumor grade
PT	Primary treatment (type of surgery performed)
AC	Adjuvant chemotherapy
AR	Adjuvant radiotherapy
AT	Adjuvant Tamoxifen

4. DISCUSSION AND CONCLUSION

In the online nonstationary data environment such as credit card transactions, the data are often high massive continuous, the distributions of data are not known, and the data distribution may be changed over time. In the real online area, the static UFFNN are not suitable to use; however, they are generally considered as the fundamental clustering methods and are adapted/modified to be used in nonstationary environments, and form the current ODUFFNN clustering methods such as the ESOM and DSOM (Kasabov, 1998; Schaal & Atkeson, 1998; Bouchachia et al., 2007; Hebboul et al., 2011). However, the current ODUFFNN clustering methods generally suffer from high training time and low accuracy of clustering, as well as high time and high memory complexities of

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

Table 11. *The results of implementation of the DSFFNN for each subset of the UMMC Breast Cancer data set*

Year	CCN	Density (%)	Data Instances Per Subset	Epoch	CPU Time (ms)	Accuracy of DSFFNN (%)
1	819	99.03	827	1	1 min, 05 s (2)	99.43
2	666	98.96	673	1	882	98.69
3	552	98.44	561	1	501	98.93
4	429	97.5	440	1	252	98.14
5	355	100	355	1	137.39	99.99
6	270	100	270	1	40.72	100
7	200	100	200	1	28.93	100
8	124	100	124	1	25.49	100
9	56	100	56	1	22.95	100

Table 12. *The accuracies of clustering methods on the UMMC Breast Cancer data set*

Year	PCA-BPN (%)	SOM-BPN (%)	RSFFNN (%)	DSFFNN (%)
1	76	82	99.55	99.43
2	63	72	98.85	98.69
3	62	71	99.04	98.93
4	77	78	98.29	98.14
5	83	86	100	99.99
6	93	93	100	100
7	98	98	100	100
8	99	99	100	100
9	99	99	100	100

clustering as scalability of their algorithms (Kasabov, 1998; Andonie & Kovalerchuk, 2007; Bouchachia et al., 2007; Rougier & Boniface, 2011). Essentially, we recognized the reasons of the problems of the current ODUFFNN clustering methods are the structure and features of the data, such as size and dimensions of data, growing of the number of clusters, and size of the network during clustering; and the topology and algorithm of the current ODUFFNN clustering method, such as using random weights, distance thresholds and parameters for controlling tasks during clustering, and relearning over several epochs, which takes time and clustering is considerably slow. In order to overcome the problems, we developed the DSFFNN clustering model with one epoch training of each online input data. Dynamically, after each entrance of the online input data, the DSFFNN learns and stores important information about the current online data, such as the weights, and completes a code book of the nonrandom weights. Then, a unique and standard weight vector such as the BMW is extracted and updated from the codebook. Consequently, a single-layer DSFFNN calculates the exclusive distance threshold of each online data based on the BMW vector. The online input data are clustered based on the exclusive distance threshold. In order to improve the resulting quality of the clustering, the model assigns a class label to the input data through the training data. The class label of each unlabeled input data is predicted by considering a linear activation function and the exclusive distance threshold. Finally, the number of clusters and the density of each cluster are updated.

To evaluate the performance of the DSFFNN clustering method, we compared the results of the proposed method with the results of other related methods for several data sets from the UCI Repository such as the Arcene data set. In addition, we showed that the DSFFNN method has the capability to be used in different environments, such as the prediction of the hydrate formation temperature, with high accuracy. In this section, we consider a real and original medical data set from the UMMC on the subject of breast cancer. Clustering the medical data sets is difficult because of limited observation, information, diagnosis, and prognosis of the specialist; incomplete medical knowledge; and lack of enough time for diagnosis (Melek & Sadeghian, 2009). However, the proposed DSFFNN method has the capability to overcome some of the problems associated with clustering in the prediction of survival time of the breast cancer patients from the UMMC. Table 13 shows the time and memory complexity of the proposed method and some related clustering methods.

The DSFFNN model has time complexity and memory complexity of $O(n.m)$ and $O(n.m.s_m)$, respectively. Parameters n, m, and s_m are the number of nodes, attributes, and size of each attribute, respectively; in addition, f_h is the number of hidden

Table 13. *The time complexities and memory complexities of the DSFFNN method and some related methods*

Method	Time Complexity	Memory Complexity
GNG	$O(c.n^2.m)$	$O(c.n^2.m.s_m)$
SOM	$O(c.n.m^2)$	$O(c.n.m^2.s_m)$
BPN	$O(c.f_h)$	$O(c.f_h.s_m)$
PCA	$O(m^2 \,.\, n) + O(m^3)$	$O((m^2 \,.\, n).s_m) + O((m^3).s_m)$
RSFFNN	$O(n.m)$	$O(n.m.s_m)$
ESOINN	$O(c.n^2.m)$	$O(c.n^2.m.s_m)$
IGNGU	$O(c.n^2.m)$	$O(c.n^2.m.s_m)$
ESOM	$O(n^2.m)$	$O(n^2.m.s_m)$
DSOM	$O(c.n.m^2)$	$O(c.n.m^2.s_m)$
DUFFNN	$O(n.m)$	$O(n.m.s_m)$
DSFFNN	$O(n.m)$	$O(n.m.s_m)$

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

Table 14. *Comparison of the DUFFNN clustering method with some current online dynamic unsupervised feedforward neural network clustering methods*

	ESOM	ESOINN	DSOM	IGNGU	DUFFNN
Base patterns	SOM and GNG, Hebbian	GNG	SOM	GNG and Hebbian	Hebbian and the features of the ESOM
Some bold features (advantages)	Begin without any node	Control the number and density of each cluster	Improve the formula of updating weights	Train by two layers in parallel	Begin without any node
					Input vectors are not stored during learning
	Update itself with online input data	Initialize codebook	Elasticity or flexibility property	Control density of each cluster and size of the network	Update itself with online input data
	The nodes with weak thresholds can be pruned	Prune for controlling noise and weak thresholds		Control noise	The nodes with weak thresholds can be pruned
		Input vector is not stored during learning		Fast training by pruning	Initialize nonrandom weights
					No sensitive to the order of the data entry
		New input does not destroy last learned knowledge			Mining the BMW
					Clustering each online input data during one epoch without updating weights
	Clustering during one epoch		Learning best match unit		Ability to retrieve old data
					Ability to learn the number of clusters

layers and c is the number of iterations. The experimental results showed that the time usage, accuracy, and memory complexity of the proposed DSFFNN clustering method were superior results by comparison with related methods. As shown in Table 14, we compare the DSFFNN clustering method with some strong current ODUFFNN clustering methods.

Table 14 shows some FFNN clustering methods such as the SOM and the GNG, which are used as base patterns and improved by the authors for proposing current ODUFFNN clustering methods (Asadi et al., 2014*b*). As we explained in Section 1, the methods inherited the properties of the base patterns, but by improving their structures; consequently, the ODUFFNN clustering methods obtained new properties. The DSFFNN clustering method inherits the structure, features, and capabilities of the RSFFNN clustering. The DSFFNN clustering with incremental lifelong or online learning property is developed for real nonstationary environments, and it is a flexible method, and with each online continuous data, immediately updates all nodes, weights, and distance thresholds. The proposed DSFFNN method is able to learn the number of clusters, without having any constraint and parameter for controlling the clustering tasks, based on the total thresholds; and it generates the clusters during just one epoch. The DSFFNN is a flexible model, and by changing the BMW, it immediately reclusters the current online data node and old nodes dynamically, and clusters all data nodes based on the new structure of the network without suffering or destroying old data. The DSFFNN clustering method is able to control or delete attributes with weak weights to reduce the data dimensions, and data with solitary thresholds in order to reduce noise.

Future research works may focus to apply the DUFFNN and the DSFFNN clustering method in order to cluster higher dimensional data with the higher number of classes in the big data environment.

REFERENCES

Abe, S. (2001). *Pattern Classification: Neuro-Fuzzy Methods and Their Comparison*. London: Springer–Verlag.

Ahirwar, G. (2014). A novel K means clustering algorithm for large datasets based on divide and conquer technique. *International Journal of Computer Science and Information Technologies 5(1)*, 301–305.

Alippi, C., Piuri, V., & Sami, M. (1995). Sensitivity to errors in artificial neural networks: a behavioral approach. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications 42(6)*, 358–361.

Andonie, R., & Kovalerchuk, B. (2007). *Neural Networks for Data Mining: Constrains and Open Problems*. Ellensburg, WA: Central Washington University, Computer Science Department.

Asadi, R., Asadi, M., & Sameem, A.K. (2014). An efficient semisupervised feed forward neural network clustering. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*. Advance online publication. doi:10.1017/S0890060414000675

Asadi, R., & Kareem, S.A. (2014). Review of feed forward neural network classification preprocessing techniques. *Proc. 3rd Int. Conf. Mathematical Sciences (ICMS3)*, pp. 567–573, Kuala Lumpur, Malaysia.

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

Asadi, R., Sabah Hasan, H., & Abdul Kareem, S. (2014*a*). Review of current online dynamic unsupervised feed forward neural network classification. *International Journal of Artificial Intelligence and Neural Networks 4(2)*, 12.

Asadi, R., Sabah Hasan, H., & Abdul Kareem, S. (2014*b*). Review of current online dynamic unsupervised feed forward neural network classification. *Proc. Computer Science and Electronics Engineering (CSEE)*. Kuala Lumpur, Malaysia.

Asuncion, A., & Newman, D. (2007). *UCI Machine Learning Repository*. Irvine, CA: University of California, School of Information and Computer Science. Accessed at http://www.ics.uci.edu/~mlearn/MLRepository

Bengio, Y., Buhmann, J.M., Embrechts, M., & Zurada, M. (2000). Introduction to the special issue on neural networks for data mining and knowledge discovery. *IEEE Transactions on Neural Networks 11(3)*, 545–549.

Bose, N.K., & Liang, P. (1996). *Neural Network Fundamentals With Graphs, Algorithms, and Applications*. New York: McGraw–Hill.

Bouchachia, A.B., Gabrys, B., & Sahel, Z. (2007). Overview of some incremental learning algorithms. *Proc. Fuzzy Systems Conf. Fuzz-IEEE*.

Craven, M.W., & Shavlik, J.W. (1997). Using neural networks for data mining. *Future Generation Computer Systems 13(2)*, 211–229.

Dasarthy, B.V. (1990). *Nearest Neighbor Pattern Classification Techniques*. Los Alamitos, CA: IEEE Computer Society Press.

Davy, H. (1811). The Bakerian Lecture: on some of the combinations of oxymuriatic gas and oxygene, and on the chemical relations of these principles, to inflammable bodies. *Philosophical Transactions of the Royal Society of London 101*, 1–35.

DeMers, D., & Cottrell, G. (1993). Non-linear dimensionality reduction. *Advances in Neural Information Processing Systems 36(1)*, 580.

Demuth, H., Beale, M., & Hagan, M. (2008). *Neural Network Toolbox TM 6: User's Guide*. Natick, MA: Math Works.

Deng, D., & Kasabov, N. (2003). On-line pattern analysis by evolving self-organizing maps. *Neurocomputing 51*, 87–103.

Du, K.L. (2010). Clustering: A neural network approach. *Neural Networks 23(1)*, 89–107.

Eslamimanesh, A., Mohammadi, A.H., & Richon, D. (2012). Thermodynamic modeling of phase equilibria of semi-clathrate hydrates of CO2, CH4, or N2+ tetra-*n*-butylammonium bromide aqueous solution. *Chemical Engineering Science 81*, 319–328.

Fisher, R. (1950). *The Use of Multiple Measurements in Taxonomic Problems: Contributions to Mathematical Statistics* (Vol. 2). New York: Wiley. (Original work published 1936)

Fritzke, B. (1995). A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems 7*, 625–632.

Fritzke, B. (1997). *Some Competitive Learning Methods*. Dresden: Dresden University of Technology, Artificial Intelligence Institute.

Furao, S., Ogura, T., & Hasegawa, O. (2007). An enhanced self-organizing incremental neural network for online unsupervised learning. *Neural Networks 20(8)*, 893–903.

Germano, T. (1999). *Self-organizing maps*. Accessed at http://davis.wpi.edu/~matt/courses/soms

Ghavipour, M., Ghavipour, M., Chitsazan, M., Najibi, S.H., & Ghidary, S.S. (2013). Experimental study of natural gas hydrates and a novel use of neural network to predict hydrate formation conditions. *Chemical Engineering Research and Design 91(2)*, 264–273.

Goebel, M., & Gruenwald, L. (1999). A survey of data mining and knowledge discovery software tools. *ACM SIGKDD Explorations Newsletter 1(1)*, 20–33.

Gui, V., Vasiu, R., & Bojković, Z. (2001). A new operator for image enhancement. *Facta universitatis-series: Electronics and Energetics 14(1)*, 109–117.

Guyon, I. (2003). Design of experiments of the NIPS 2003 variable selection benchmark. *Proc. NIPS 2003 Workshop on Feature Extraction and Feature Selection*. Whistler, BC, Canada, December 11–13.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research 3*, 1157–1182.

Hamker, F.H. (2001). Life-long learning cell structures—continuously learning without catastrophic interference. *Neural Networks 14(4–5)*, 551–573.

Han, J., & Kamber, M. (2006). *Data Mining, Southeast Asia Edition: Concepts and Techniques*. San Francisco, CA: Morgan Kaufmann.

Haykin, S. (2004). *Neural Networks: A Comprehensive Foundation*, Vol. 2. Upper Saddle River, NJ: Prentice Hall.

Hazlina, H., Sameem, A., NurAishah, M., & Yip, C. (2004). Back propagation neural network for the prognosis of breast cancer: comparison on different training algorithms. *Proc. 2nd. Int. Conf. Artificial Intelligence in Engineering & Technology*, pp. 445–449, Sabah, Malyasia, August 3–4.

Hebb, D.O. (1949). *The Organization of Behavior: A Neuropsychological Approach*, Vol. 1., pp. 143–150. New York: Wiley.

Hebboul, A., Hacini, M., & Hachouf, F. (2011). An incremental parallel neural network for unsupervised classification. *Proc. 7th Int. Workshop on Systems, Signal Processing Systems and Their Applications (WOSSPA)*, Tipaza, Algeria, May 9–11, 2011.

Hegland, M. (2003). *Data Mining—Challenges, Models, Methods and Algorithms*. Canberra, Australia: Australia National University, ANU Data Mining Group.

Hinton, G.E., & Salakhutdinov, R.R. (2006). Reducing the dimensionality of data with neural networks. *Science 313(5786)*, 504.

Honkela, T. (1998). *Description of Kohonen's self-organizing map*. Accessed at http://www.cis.hut.fi/~tho/thesis

Jacquier, E., Kane, A., & Marcus, A.J. (2003). Geometric or arithmetic mean: a reconsideration. *Financial Analysts Journal 59(6)*, 46–53.

Jain, A.K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters 31(8)*, 651–666.

Jean, J.S., & Wang, J. (1994). Weight smoothing to improve network generalization. *IEEE Transactions on Neural Networks 5(5)*, 752–763.

Jolliffe, I.T. (1986). *Principal Component Analysis*. Springer Series in Statistics, pp. 1–7. New York: Springer.

Kamiya, Y., Ishii, T., Furao, S., & Hasegawa, O. (2007). An online semi-supervised clustering algorithm based on a self-organizing incremental neural network. *Proc. Int. Joint Conf. Neural Networks (IJCNN)*. Piscataway, NJ: IEEE.

Kantardzic, M. (2011). *Data Mining: Concepts, Models, Methods, and Algorithms*. Hoboken, NJ: Wiley–Interscience.

Kasabov, N.K. (1998). ECOS: evolving connectionist systems and the ECO learning paradigm. *Proc. 5th Int. Conf. Neural Information Processing, ICONIP'98*, Kitakyushu, Japan.

Kemp, R.A., MacAulay, C., Garner, D., & Palcic, B. (1997). Detection of malignancy associated changes in cervical cell nuclei using feed-forward neural networks. *Journal of the European Society for Analytical Cellular Pathology 14(1)*, 31–40.

Kobayashi, R., Song, K.Y., & Sloan, E.D. (1987). Phase behavior of water/hydrocarbon systems. In *Petroleum Engineering Handbook* (Bradley, H.B., Ed.), chap. 25. Richardson, TX: Society of Petroleum Engineers.

Kohonen, T. (1997). *Self-Organizing Maps*, Springer Series in Information Sciences Vol. 30, pp. 22–25. Berlin: Springer–Verlag.

Kohonen, T. (2000). *Self-Organization Maps*, 3rd ed. Berlin: Springer–Verlag.

Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of Machine Learning Research 10*, 1–40.

Laskowski, K., & Touretzky, D. (2006). *Hebbian learning, principal component analysis, and independent component analysis. Artificial neural networks*. Accessed at http://www.cs.cmu.edu/afs/cs/academic/class/15782-f06/slides/hebbpca.pdf

Linde, Y., Buzo, A., & Gray, R. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communications 28(1)*, 84–95.

Longadge, M.R., Dongre, M.S.S., & Malik, L. (2013). Multi-cluster based approach for skewed data in data mining. *Journal of Computer Engineering 12(6)*, 66–73.

Mangat, V., & Vig, R. (2014). Novel associative classifier based on dynamic adaptive PSO: application to determining candidates for thoracic surgery. *Expert Systems With Applications 41(18)*, 8234–8244.

Martinetz, T.M. (1993). Competitive Hebbian learning rule forms perfectly topology preserving maps. *Proc. ICANN'93*, pp. 427–434. London: Springer.

Mathworks. (2008). *Matlab Neural Network Toolbox*. Accessed at http://www.mathworks.com

McCloskey, S. (2000). *Neural networks and machine learning*. Accessed at http://www.cim.mcgill.ca/~scott/RIT/research_project.html

Melek, W.W., & Sadeghian, A. (2009). A theoretic framework for intelligent expert systems in medical encounter evaluation. *Expert Systems 26(1)*, 82–99.

Moradi, M.R., Nazari, K., Alavi, S., & Mohaddesi, M. (2013). Prediction of equilibrium conditions for hydrate formation in binary gaseous systems using artificial neural networks. *Energy Technology 1(2–3)*, 171–176.

Oh, M., & Park, H.M. (2011). Preprocessing of independent vector analysis using feed-forward network for robust speech recognition. *Proc. Neural Information Processing Conf.*, Granada, Spain, December 12–17.

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press

Pavel, B. (2002). *Survey of Clustering Data Mining Techniques*. San Jose, CA: Accrue Software.
Peng, J.M., & Lin, Z. (1999). A non-interior continuation method for generalized linear complementarity problems. *Mathematical Programming 86(3)*, 533–563.
Prudent, Y., & Ennaji, A. (2005). An incremental growing neural gas learns topologies. *Proc. IEEE Int. Joint Conf. Neural Networks, IJCNN'05*, San Jose, CA, July 31–August 5.
Rougier, N., & Boniface, Y. (2011). Dynamic self-organising map. *Neurocomputing 74(11)*, 1840–1847.
Schaal, S., & Atkeson, C.G. (1998). Constructive incremental learning from only local information. *Neural Computation 10(8)*, 2047–2084.
Shahnazar, S., & Hasan, N. (2014). Gas hydrate formation condition: review on experimental and modeling approaches. *Fluid Phase Equilibria 379*, 72–85.
Shen, F., Yu, H., Sakurai, K., & Hasegawa, O. (2011). An incremental online semi-supervised active learning algorithm based on self-organizing incremental neural network. *Neural Computing and Applications 20(7)*, 1061–1074.
Tong, X., Qi, L., Wu, F., & Zhou, H. (2010). A smoothing method for solving portfolio optimization with CVaR and applications in allocation of generation asset. *Applied Mathematics and Computation 216(6)*, 1723–1740.
Ultsch, A., & Siemon, H.P. (1990). Kohonen's self organizing feature maps for exploratory data analysis. *Proc. Int. Neural Networks Conf.*, pp. 305–308.
Van der Maaten, L.J., Postma, E.O., & Van den Herik, H.J. (2009). Dimensionality reduction: A comparative review. *Journal of Machine Learning Research 10(1–41)*, 66–71.
Vandesompele, J., De Preter, K., Pattyn, F., Poppe, B., Van Roy, N., De Paepe, A., & Speleman, F. (2002). Accurate normalization of real-time quantitative RT-PCR data by geometric averaging of multiple internal control genes. *Genome Biology 3(7)*, research0034.
Werbos, P. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. PhD Thesis. Harvard University.
Zahedi, G., Karami, Z., & Yaghoobi, H. (2009). Prediction of hydrate formation temperature by both statistical models and artificial neural network approaches. *Energy Conversion and Management 50(8)*, 2052–2059.
Ziegel, E.R. (2002). Statistical inference. *Technometrics 44(4)*, 407–408.

Roya Asadi is a PhD candidate researcher in computer science with artificial intelligence (neural networks) at the University of Malaya. She received a bachelor's degree in computer software engineering from Hahid Beheshti University and the Computer Faculty of Data Processing Iran Co. (IBM). Roya obtained a master's of computer science in database systems from UPM University. Her professional working experience includes 12 years of service as a Senior Planning Expert 1. Roya's interests are in data mining, artificial intelligence, neural network modeling, intelligent multiagent systems, system designing and analyzing, medical informatics, and medical image processing.

Sameem Abdul Kareem is an Associate Professor in the Department of Artificial Intelligence at the University of Malaya. She received a BS in mathematics from the University of Malaya, an MS in computing from the University of Wales, and a PhD in computer science from University of Malaya. Dr. Kareem's interests include medical informatics, information retrieval, data mining, and intelligent techniques. She has published over 80 journal and conference papers.

Shokoofeh Asadi received a bachelor's degree in English language translation (international communications engineering) from Islamic Azad University and a master's degree in agricultural management engineering from the University of Science and Research. Her interests are English language translation, biological and agrcultural engineering, management and leadership, strategic management, and operations management.

Mitra Asadi is a Senior Expert Researcher at the Blood Transfusion Research Center in the High Institute for Research and Education in Transfusion Medicine. She received her bachelor's degree in laboratory sciences from Tabriz University and her English language translation degree and master's of English language teaching from Islamic Azad University. She is pursuing her PhD in entrepreneurship technology at Islamic Azad University of Ghazvi.

https://doi.org/10.1017/S0890060416000160 Published online by Cambridge University Press