Int. J. Appl. Math. Comput. Sci., 2018, Vol. 28, No. 2, 411–424
DOI: 10.2478/amcs-2018-0031

PATTERN LAYER REDUCTION FOR A GENERALIZED REGRESSION NEURAL NETWORK BY USING A SELF–ORGANIZING MAP

SERKAN KARTAL [a,*], MUSTAFA ORAL [a], BUSE MELIS OZYILDIRIM [a]

[a]Department of Computer Engineering
University of Cukurova, 01330 Balcali, Saricam/Adana, Turkey
e-mail: skartal@cu.edu.tr

In a general regression neural network (GRNN), the number of neurons in the pattern layer is proportional to the number of training samples in the dataset. The use of a GRNN in applications that have relatively large datasets becomes troublesome due to the architecture and speed required. The great number of neurons in the pattern layer requires a substantial increase in memory usage and causes a substantial decrease in calculation speed. Therefore, there is a strong need for pattern layer size reduction. In this study, a self-organizing map (SOM) structure is introduced as a pre-processor for the GRNN. First, an SOM is generated for the training dataset. Second, each training record is labelled with the most similar map unit. Lastly, when a new test record is applied to the network, the most similar map units are detected, and the training data that have the same labels as the detected units are fed into the network instead of the entire training dataset. This scheme enables a considerable reduction in the pattern layer size. The proposed hybrid model was evaluated by using fifteen benchmark test functions and eight different UCI datasets. According to the simulation results, the proposed model significantly simplifies the GRNN's structure without any performance loss.

Keywords: generalized regression neural network, artificial neural network, self-organizing maps, nearest neighbour, reduced dataset.

1. Introduction

The general regression neural network (GRNN) was introduced by Donald F. Specht as an alternative to the well-known back-propagation training algorithms (Specht, 1991). The main advantage of the GRNN model is that unlike the other training algorithms that require large numbers of iterations to be performed in the training phase to converge to a desired solution, the GRNN does not require an iterative training process (Krenker *et al.*, 2011). Due to its simple structure and ease of implementation, the GRNN has been widely employed to solve a variety of problems in pattern classification, clustering, function approximation, optimization and prediction.

Reducing the data complexity and determining the relevant training data are two of the most important steps in the GRNN. Usually, while a portion of the training data are informative, some may be noisy or have no important relationship with the test record being modelled. Thus, without using any pre-processing algorithm for related input determination, shortcomings may emerge, including the following (Maier and Dandy, 1997; Szemenyei and Vajda, 2017):

- increased computational complexity,
- increased memory requirements,
- increased processing time,
- reduced prediction accuracy.

The use of pre-processing algorithms provides obvious advantages in the selection of appropriate training data for GRNNs. Several methods based on various clustering and local learning algorithms have been proposed to overcome these problems. First, Specht presented the clustering version of a GRNN to reduce the number of neurons in the pattern layer (Specht, 1991). In subsequent studies, various clustering techniques have been proposed for compressing all training data into a few clusters. Zheng *et al.* (2008) used a k-means clustering algorithm for separating training data into groups. In that study, an improvement in the prediction

*Corresponding author

accuracy of the GRNN was achieved by grouping data and assigning different smoothing parameters to each group. An estimation-of-distribution algorithm (EDA) was employed to determine the optimum smoothing parameter. When clustering methods are employed as a pre-processor, the number of clusters, another critical parameter affecting the GRNN performance, emerges. The number of cluster is selected by increasing the number k with a predefined step size.

Zhao *et al.* (2007) proposed a method based on fuzzy means clustering (FMC) to reduce the number of pattern neurons. FMC groups similar data into clusters, and cluster centres represent all the samples in the cluster. During the clustering operation, first, similarity values between the training data and cluster centres are calculated according to a given similarity metric. Second, similarity is compared with a given threshold value. If the similarity value is less than the threshold value, the data are placed into the related cluster; otherwise, they are designated as the centre of the newly created cluster. While the algorithm given by Zhao *et al.* (2007) simplified the GRNN structure, the accuracy of the model decreased slightly.

In another study, Bowden *et al.* (2005) reported several disadvantages of the application of all potential training data in a GRNN, including the increased computational complexity and memory requirements. To address these shortcomings, the authors proposed an SOM combined with a hybrid genetic algorithm and a GRNN (SOM-GAGRNN). This algorithm utilizes the SOM to cluster the training data into groups of similar data. Then, from each cluster, the training record with the shortest Euclidean distance to the cluster weights is selected. Finally, the obtained training data are applied to the GRNN.

Another GRNN algorithm based on FMC was suggested by Husain *et al.* (2004). In this algorithm, a similarity index is calculated through a simple similarity measure to indicate the degree of similarity in the data to be clustered. The cluster centre of related data is determined according to an iterative process. In the work of Husain *et al.* (2004), the calculated cluster centres were used to define the parameters of the GRNN.

A hybrid model combining the fuzzy ART (FA) and a GRNN for noisy data was proposed by Yuen *et al.* (2004). FA is a powerful unsupervised classifier based on adaptive resonance theory. When the FA was used as the pre-processor for the GRNN to compress training data, the obtained output values were employed in pattern layer neurons for prediction.

Kokkinos and Margaritis (2015) introduced the local learning model of the GRNN algorithm. The method uses only the k-nearest neighbours of the test record and reduces the operation cost from $\mathcal{O}(N)$ to $\mathcal{O}(k)$, where $\mathcal{O}(N)$ denotes the Big-O notation of the computational cost for the use of all training data and $\mathcal{O}(k)$ is the Big-O notation of the computational cost for the use of k-nearest training data. Since $k \ll N$, the method improves the suitability of the GRNN for large-scale applications. First, a network is constructed with a minimum number of nearest neighbours. Then, new neurons are progressively added until the process reaches the last nearest neighbour. At each step, a new network structure is created and a prediction value is obtained by averaging the results of each network.

Recent studies indicate that clustering versions of the GRNN may provide several advantages such as speed, reduced cost, and more reliable results. With the use of clustering methods, training data are grouped into clusters, and instead of employing all training samples as kernels, only the training samples belonging to these clusters are used. Hence, the redundant samples can be removed from the training set, reducing the required calculations.

In the present study, a hybrid GRNN model, an SOM-GRNN, is proposed to select the relevant training data and reduce the sample complexity of the standard GRNN (SGRNN), which includes sigma tuning and excludes any pre-processing algorithm. SOM-based clustering is an appropriate alternative to traditional clustering algorithm techniques. Hence, the main objectives of the proposed study are to design a specific SOM-based pre-processing algorithm for GRNNs and to provide more reliable results than the traditional pre-processing algorithms.

2. Problem of input data representation

A review the literature reveals that in many cases, the lack of a reliable algorithm for selecting the related training data raises doubt about the representation efficiency of the selected training data. Training data determination methodologies vary depending on the employed algorithm. The main algorithms can be divided into two categories.

In the first category of algorithms, training data are separated into a set of independent groups, called clusters, which are not necessarily representative of test data. Thus, while similar data are grouped within the same cluster, dissimilar data are grouped into distinct clusters. This operation helps users to discover and understand the natural structure of the data and extract the essence of large datasets (Harkanth and Phulpagar, 2013; Berkhin, 2002). When a new test record is applied, the related data cluster is determined and used as the training dataset. Only the relationships between test records and cluster centres are calculated. The proposed algorithms that use the clustering algorithms as a pre-processor are examples of this category (Zheng *et al.*, 2008; Zhao *et al.*, 2007; Husain *et al.*, 2004; Yuen *et al.*, 2004).

In the second category, where the aim is to find related data, the relationships between a test record and all other data are determined when the test record is applied. In such algorithms, there is no training phase and operations are performed directly on the test data. Thus, more related training data for the GRNN are obtained. Relations between the test record and other records are calculated one by one, and if the dependence is found to be important, the record is retained in the final training data subset of the GRNN. Otherwise, it is eliminated. Since all these operations are performed after the test data are received, the number of transactions during the prediction process increases greatly.

In the present study, the following input determination methods are implemented to compare their results with the proposed method:

1. k-means, FMC and winner takes all (WTA) are employed as pre-processing algorithms of the first category. The k-means clustering algorithm is one of the most widely used clustering algorithms in the literature (Hartigan and Wong, 1979; Sabo, 2014). The FMC algorithm is frequently used with GRNNs and is included here as a representative of alternative approaches to GRNN modelling (Bezdek *et al.*, 1984). WTA is used as an alternative to the SOM as a competitive learning algorithm. These three clustering algorithms in conjunction with a GRNN enable a detailed and reliable comparison for the study. The most fundamental shortcoming of these algorithms is the difficulty in determining the number of clusters (Kolesnikov *et al.*, 2015).

2. The algorithm chosen for comparison from the second category is k-nearest neighbours (kNN). kNN is an appropriate alternative to the traditional clustering algorithms. The aims of the algorithm are to find the k-nearest neighbours of the record with defined distance functions and to reduce the operation cost from $\mathcal{O}(N)$ to $\mathcal{O}(k)$, where k is the neighbour count, N is the total number of data and $k \ll N$ (Cover and Hart, 1967). As in clustering algorithms, there is a critical parameter named the neighbour count (k) to be determined. In general, a small value of k increases the influence of the noisy data on the result, whereas a larger k increases the computational cost. Detailed information can be found in the work of Cover and Hart (1967).

3. Background

3.1. SOM. The SOM is one of the most widely used neural networks. An SOM net can be considered a two-layered neural network working in two modes: training and mapping. This model projects high-dimensional data onto a two-dimensional map

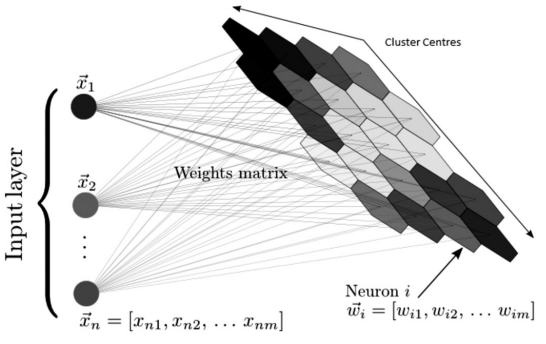

Fig. 1. General structure of the SOM.

(Kohonen, 1982). A graphical representation of the algorithm is given in Fig. 1, where x denotes the m-dimensional N records and w denotes the m-dimensional k weight vectors (Carrasco Kind and Brunner, 2014).

Map units are called nodes or neurons. Each node has its own weight vector, which has the same dimension as the input vector. Nodes are placed on a hexagonal or rectangular grid with certain intervals. In the training mode, competitive learning is employed (Jain *et al.*, 1998). First, all weight vectors are initialized to small random values. Second, the Euclidean distance between the input record $\vec{x}_j$ and weight vector $\vec{w}_i$ is computed as follows:

$$ds_i(\vec{x}_j) = ||\vec{x}_j - \vec{w}_i||, \tag{1}$$

where $ds_i(\vec{x}_j)$ is the Euclidean distance between the j-th input record and the i-th weight vector.

The node whose weight vector is closest to the input data is detected and called the best matching unit (BMU) or winner unit. Lastly, the weight values of the BMU and the neighbouring nodes are adjusted to the input data. For adjusting these weight values, a neighbourhood is calculated. Neighbourhood $H_{(i,\mathrm{BMU}(\vec{x}_j))}(l)$ for the l-th iteration is calculated with the following equation:

$$H_{(i,\mathrm{BMU}(\vec{x}_j))}(l) = \exp\left(\frac{-S^2_{(i,\mathrm{BMU}(\vec{x}_j))}}{2\sigma(l)^2}\right), \tag{2}$$

where $\mathrm{BMU}(\vec{x}_j)$ is the BMU, i is a node in the grid, and $S_{(i,\mathrm{BMU}(\vec{x}_j))}$ denotes the lateral distance between the i-th node and $\mathrm{BMU}(\vec{x}_j)$, and $\sigma(l)$ is a coefficient used for determining the number of affecting neighbours.

The magnitude of the adjustment decreases with respect to the time and distance from the BMU. A popular time dependence is the exponential function

$$\sigma(l) = \sigma(0)\exp\left(\frac{-l}{\lambda}\right), \tag{3}$$

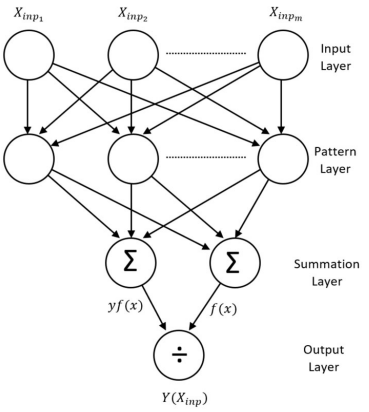

Fig. 2. Schematic diagram of the GRNN.

where $\sigma(0)$ signifies the magnitude at the initial time, λ is a time constant and l is the current time step (iteration number). Due to the change in the magnitude, the neighbourhood will shrink to the size of just one node, the BMU, over time. The weight values for the l-th iteration are updated as follows:

$$\begin{aligned}\vec{w}_i(l) = \vec{w}_i(l-1) \\ + \mathrm{lr}(l) \times H_{(i,\mathrm{BMU}(\vec{x}_j))}(l) \times (\vec{x}_j - \vec{w}_i),\end{aligned} \tag{4}$$

where $\mathrm{lr}(l)$ denotes the learning rate at the l-th iteration.

The nodes of the SOM are themselves cluster centres, and these nodes are utilized to form larger clusters. This approach is a powerful method for detecting outliers and dealing with missing data values. Similar to the clustering methods and kNNs, the critical parameter of an SOM network is its size (k) (Kotsiantis and Pintelas, 2004; Rama *et al.*, 2010).

3.2. GRNN. A GRNN is a four-layer feed-forward neural network based on Parzen's window estimation and often used for function approximation. The GRNN consists of an input layer, a pattern layer, a summation layer and an output layer. Each layer contains a different number of neurons by which the different layers can connect with each other. The general structure of a GRNN model is given in Fig. 2 (Specht, 1991).

Let $\vec{x}_{\text{inp}}$ be the input, $\vec{x}_j$ be the j-th sample stored in the pattern layer and dg_j denote the squared Euclidean distance between $\vec{x}_j$ and $\vec{x}_{\text{inp}}$. Prediction for the input $\vec{x}_{\text{inp}}$ can be calculated as follows:

$$Y(\vec{x}_{\text{inp}}) = \frac{\sum\limits_{j=1}^{N} y_j \exp\left(\frac{-dg_j}{2\sigma^2}\right)}{\sum\limits_{j=1}^{N} \exp\left(\frac{-dg_j}{2\sigma^2}\right)}, \tag{5}$$

where $Y(\vec{x}_{\text{inp}})$ is the predicted value, σ is a smoothing parameter and y_j is the output value for the j-th sample.

Each layer of the GRNN computes one of the steps of the calculation given in (5). The number of neurons in the first (or input) layer is equal to the dimension of $\vec{x}_j$. The second layer is the pattern layer, where each training record is stored in one neuron. In this layer, the Euclidean distance dg_j between the incoming test record $\vec{x}_{\text{inp}}$ and training record $\vec{x}_j$ is computed as follows:

$$dg_j = (\vec{x}_{\text{inp}} - \vec{x}_j)^T (\vec{x}_{\text{inp}} - \vec{x}_j). \tag{6}$$

Pattern layer neurons output the weight values $f(x)$ by using a Gaussian unit $\exp\left(-dg_j/2\sigma^2\right)$. The third layer has two summation neurons a numerator and a denominator. While the numerator, represented with $yf(x)$, sums the products of weights with the target value of the related training record, the denominator, denoted with $f(x)$, computes the sum of all weights. Finally, the fourth layer divides the numerator by the denominator and produces the estimated output (Specht, 1991).

4. Proposed algorithm

Instead of using all training samples in the pattern layer, various algorithms such as clustering and kNNs have been used in the literature to determine relevant training data.

To select relevant training data, a new SOM-based GRNN algorithm that significantly reduces the sample complexity is proposed in this article. Unlike traditional clustering algorithms, SOM clustering forms a semantic map where similar cluster centres are mapped close together and dissimilar ones further apart. Depending on this feature, while the most similar data are labelled with the same cluster index, less similar data are labelled with the nearest neighbour cluster index and dissimilar ones with the distant cluster index. Based on these advantages of the SOM, not only the best matching cluster data but also less similar nearest clusters' data are taken into account when GRNNs pattern layer neurons are constructed.

In an SOM-GRNN, first an SOM is created with training data to find the clusters. Then, when the test data are applied by using the created map, their clusters and neighbours are detected. All the samples belonging to the chosen clusters are placed into the pattern layer of the GRNN. Finally, the GRNN steps are applied to the test data for prediction.

A general overview of the test step of the SOM-GRNN algorithm by using the most similar and similar clusters is illustrated in Fig. 3. In this figure, $\vec{x}_{\text{inp}}$ is an m-dimensional test vector, its BMU is shown with 5 and the neighbours of the BMU are represented with 1, 2, 3, 4, 6, 7, 8, and 9. Training data x belonging to these clusters are placed into the pattern layer of the GRNN, and $\vec{x}_{\text{inp}}$ is applied to the GRNN.

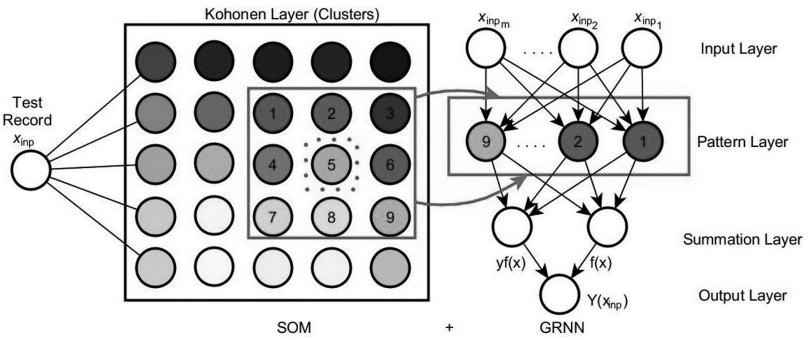

Fig. 3. General overview of the SOM-GRNN.

Let y contain the target values of the x, k be the number of map units (clusters), x^{red} be m-dimensional chosen samples belonging to the BMU and its neighbouring clusters, and R be the number of reduced training data. The algorithm of the SOM-GRNN is summarized as Algorithm 1.

Algorithm 1. Selection of the stationary point.

Require: x, $\vec{x}_{\text{inp}}$, y, k,σ,lr_0,σ_0

1: Set to nom and $denom$ as zeros
2: Assign random values to w
3: Apply steps of the SOM to generate a map with k units using x
4: Apply $\vec{x}_{\text{inp}}$ as test
5: **while** $i \leq k$ **do**
6: $ds_i(\vec{x}_{\text{inp}}) = ||\vec{x}_{\text{inp}} - \vec{w}_i||$;
7: **end while**
8: BMU $= \arg\min_i(ds)$
9: Find nearest neighbours of BMU itself
10: Find training records x^{red} belonging to nearest neighbours of BMU and BMU
11: **while** $r \leq R$ **do**
12: Calculate squared Euclidean distance between $\vec{x}_{\text{inp}}$ and $\vec{x}_r^{\text{red}}$ as $dg_r = \sum_{i=1}^{m}(x_{\text{inp},i} - x_{r,i}^{\text{red}})^2$
13: Compute $nom = nom + y_r \times \exp(\frac{-dg_r}{2\sigma^2})$
14: Compute $denom = denom + \exp(\frac{-dg_r}{2\sigma^2})$
15: **end while**
16: $Y(\vec{x}_{\text{inp}}) = \dfrac{nom}{denom}$
17: **return** $Y(\vec{x}_{\text{inp}})$ {Returns prediction result}

5. Comparison of the SOM-GRNN with popular data selection and clustering algorithms

In this section, one of the employed clustering algorithms, k-means clustering and kNN, are implemented and analysed as a pre-processor of the GRNN. It can be seen from the examples given in Fig. 4 that the major weakness of the clustering algorithm is caused by the occurrence of outliers. Outliers are abnormal test data that are present at the boundaries and are distant from principal units. Their predictions are complicated and inconsistent for the GRNN. The use of clustering to summarize training data for the GRNN causes the emergence of new inner borders that are formed by previously intermediate units.

An example view of the k-means clustering algorithm's result over the Bird function consisting of 400 randomly created samples is given in Fig. 4. The cluster count is chosen as 4, and 4 different marks are used to illustrate the clusters. Lines represent the projection of the Bird function results on two dimensions; the figure and the circled shapes illustrate the regions with peak results of the function.

The proximity of the function lines to each other is proportional to the force of the peak values. In the figure, the $\times$ marks represent test data. The test records lying within the borders of the clusters are called inner outliers. These outliers are illustrated with both $\times$ and circles. Even though these data are not outliers for the SGRNN, they may behave as outliers for clustering-GRNN algorithms. Since some parts of training data similar to the test record (outlier record) are in another cluster, a more erroneous prediction result compared with the standard algorithm is obtained. Assume that one of the inner outlier test records lying between the square and circle clusters is clustered into the circle cluster by the clustering algorithm. In this case, this test record is predicted by using only the training data belonging to the circle cluster, while the square training data closest to this test record cannot be employed. Hence, the prediction error increases.

The SOM has critical importance for the selection of all relevant training data, especially in operations of the test data that are near the cluster borders. Since similar clusters are located close to each other in the output of the SOM, not only the most similar cluster but also its neighbouring clusters with less similar training data are used in the GRNN. Thus, the test record is not allowed to remain in the border area of the training data used, and the emergence of inner outlier data between the clusters is prevented.

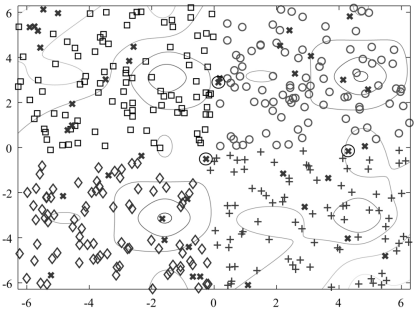

Fig. 4. Clustering-GRNN: inner outliers.

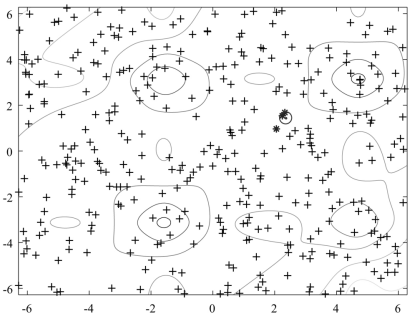

Fig. 5. Training data of the most similar cluster on the Bird function.

In Fig. 5, the use of training samples from only the most similar cluster of the test record is illustrated with the Bird function. In Fig. 6, the use of training samples from both the most similar and similar clusters (its neighbouring clusters) is illustrated with the Bird function for the same test record. While the + marks indicate all training data, the stars indicate selected training data and the dot with a circle indicates the test record. Figure 5 illustrates that the use of the training data belonging only to the BMU (cluster) may produce inner outlier data as in the k-means clustering algorithm.

A second popular pre-processing algorithm, kNN, requires excessive calculations for the prediction of each test record separately. When a new test record is applied, first, distances are measured between the test record and all training data. Then, one more calculation step is required to determine the k-nearest training data. Furthermore, if the selected nearest data are located near the training data heap, the prediction result will be derived from unevenly distributed neighbours, rather than from the surrounding examples reflecting the true nature of test record, and it may be close to the mean value of selected records.

5.1. Complexity analysis. The SOM-GRNN algorithm consists of two separate steps, the SOM and

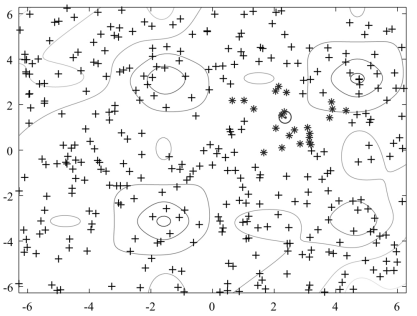

Fig. 6. Training data of the most similar and similar clusters on the Bird function.

GRNN. If we analyze the complexity of the algorithm, both the algorithms must be examined separately and summed at the final step.

Suppose that N is the number of training data, k is the number of map units that is small enough to be ignored when compared with N, and L is the number of iterations.

The computational complexity of the traditional SOM algorithm for each iteration will be $\mathcal{O}(kN)$ and the total complexity of the SOM can be considered as $\mathcal{O}(LkN)$. The second step of the algorithm consists of the GRNN, which is the one pass learning algorithm and has $\mathcal{O}(N)$ computational complexity for the prediction of one test record. However, in the SOM-GRNN, the number of training data is reduced from N to R, where $R \ll N$. Consequently, the SOM-GRNN algorithm has $\mathcal{O}(LkN) + \mathcal{O}(R)$ complexity, which equals $\mathcal{O}(LkN)$. The WTA algorithm, which is another "competitive learning" algorithm, has the same calculation complexity, as it has the same processing procedures as the SOM.

The other two popular clustering-GRNN algorithms (FMC and k-means) have structures similar to the SOM-GRNN. In the clustering step, both the algorithms require $\mathcal{O}(kN)$ calculations for each iteration, where k is the cluster count. Since the FMC algorithm requires a membership matrix, once the distance from the cluster centre is calculated, the membership matrix is updated in the same loop. Although this computation slightly increases the transaction count, the computational complexity does not change. In both algorithms, with the addition of the SGRNN, the total complexity can be considered as $\mathcal{O}(LkN) + \mathcal{O}(N)$ for L iterations.

In contrast, in the kNN-GRNN, both algorithms are in nested structures. In the basic kNN, the k nearest training data have to be detected for each test record, which requires N distance calculation and sorting operations. These operations require N and $N \log N$ operations, respectively. For one test record, the total computational complexity for kNN and the GRNN is $\mathcal{O}(N + N \log N + k)$.

According to the complexity analysis, SOM-GRNN,

WTA-GRNN, k-means-GRNN, FMC-GRNN algorithms have $\mathcal{O}(LkN)$, the kNN-GRNN has $\mathcal{O}(N \log N)$, and the SGRNN has $O(N)$ computational complexities. When the results are superficially examined, the complexity of the standard algorithm, $\mathcal{O}(N)$, seems less than that of the others. However, SOMs and other clustering methods (k-means, FCM and WTA) are pre-processing algorithms and have to be performed before the testing step. Thus, when a new test record is applied, there remain two operations: the detection of nearest clusters and the GRNN operation with reduced training data. This feature ensures faster prediction of these four algorithms for the instantaneous test record and a decrease in computational complexity from $\mathcal{O}(N)$ to $\mathcal{O}(R)$, where $R \ll N$.

6. Tests and results

6.1. Real-world datasets and test functions. To test the practicality of the SOM-GRNN, the method was tested on eight datasets with known dependence attributes. The benchmark datasets we use in the test process are listed in Table **??**. Datasets are downloaded from the UCI repository (Bache and Lichman, 2013). All the input features and targets have been normalized into the range [0, 1].

It was considered necessary to compare the input determination method on synthetic data, before applying this method to the real-world case study. Fifteen well-known benchmark test functions were employed for analyzing and comparing the performances of the proposed SOM-GRNN algorithm and the others (SGRNN, k-means-GRNN, FMC-GRNN, WTA-GRNN and kNN-GRNN) (Tang *et al.*, 2009). The functional form, boundaries and global minimum points of the test functions are given in Table 2.

6.2. Cluster count estimation. Clustering algorithms have been fundamental pre-processors for GRNNs in the literature. However, determining the number of clusters (k) is the main problem in data clustering, and selection of the appropriate k is often ambiguous. In this study, four k-estimation algorithms proposed in the literature (Calinski–Harabasz, Davies–Bouldin, Gap and Silhouette) are employed to determine the best number of clusters for each real-world dataset and benchmark function (Caliński and Harabasz, 1974; Davies and Bouldin, 1979; Rousseeuw, 1987; Tibshirani *et al.*, 2001). Since the generated random data are the same for all the test functions, only one test is performed for randomly created data. Cluster numbers from 2 to 100 were evaluated by the algorithms, and the most suitable numbers of clusters in this range are given in Table 3.

There is no common cluster count for any dataset. Therefore, according to three different data reduction percentage values selected (80, 90, 95), three different k-cluster counts (5, 10, 20) were utilized to demonstrate the prediction performances of the clustering-GRNN algorithms.

6.3. Results and discussion. In this study, in addition to the SOM-GRNN, SGRNN and integration of the GRNN with four other pre-processing algorithms, k-means clustering, FMC, WTA and kNN were developed and implemented for more reliable discussions. All the algorithms were implemented using MATLAB.

Before starting the test procedures in addition to the cluster count, there are three more parameters: best sigma, nearest-neighbour count and SOM size. These parameters have to be tuned for the GRNN, clustering algorithms, kNN and the SOM, respectively. Sigma optimization is crucial and was accomplished by ten-fold cross validation. For benchmark test functions, the sigma value was initiated at 0.1 and increased gradually up to 3 by 0.1 steps to find an optimum value. However, for real datasets, since all the input features and targets were normalized into the range [0, 1], sigma was initiated at 0.01 and increased gradually up to 1 by 0.01 steps. While there is only one sigma value for the SGRNN, in the clustering-GRNN there are different sigma values for each cluster. Thus, on completion of the clustering process, a sigma optimization procedure was applied for each cluster separately. Since in kNN and the SOM-GRNN training data are determined in accordance with the test record during the test process, their optimal sigma values cannot be obtained until the test operation starts. The sigma searching process at the test step causes extra operation costs and decreases in speed. Moreover, in kNN, due to the small number of training data, the process of searching for the optimal sigma value may not provide reliable results. Hence, the optimal sigma value was tuned for the SGRNN and all clustering-GRNNs. The optimal sigma value calculated for the SGRNN is also used for the kNN-GRNN and the SOM-GRNN. Additionally, if we intended to carry out a similar optimization procedure for the SOM size and the nearest-neighbour count, a huge number of tests would be required and it would take quite a long time. Therefore, three different nearest-neighbour counts (5, 10, 20) and three different SOM sizes (8, 10, 16) were utilized to demonstrate the prediction performances of the algorithms.

The proposed SOM-GRNN, the SGRNN and the other four hybrid GRNN algorithms were tested on eight different UCI datasets and 1000 randomly created data for each test function. Ten-fold cross validation was carried out for each configuration to compare the prediction results obtained from these datasets. The UCI datasets and randomly created data were separated to ten sub-groups. While nine sub-groups were used for training the models, the remaining sub-group was used as test data. Until all ten sub-groups were applied as

Table 2. Test functions with their boundaries and global minimums.

Function	Definition	Functional form	Boundaries	Min
f_1	Beale	$(1.5 - x_1 + x_1x_2)^2 + \left(2.25 - x_1 + x_1x_2^2\right)^2 + \left(2.625 - x_1 + x_1x_2^3\right)^2$	$[-4.5, 4.5]$	0
f_2	Rastrigin	$f(\mathbf{x}) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right]$	$[-5.12, 5.12]$	0
f_3	Schwefel	$\sum_{i=1}^{n}(-x_i \sin(\sqrt{\lvert x_i\rvert})) + 418.9829n$	$[-500, 500]$	-837.96
f_4	Ackley	$f(x_0 \cdots x_n) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})$ $-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	$[-32, 32]$	0
f_5	Goldstein Price	$\left(1 + (x_1 + x_2 + 1)^2\left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right)$ $\left(30 + (2x_1 - 3x_2)^2\left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right)$	$[-2, 2]$	3
f_6	Six Hump Camel	$4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$[-5, 5]$	-1.03
f_7	Bird	$(x_1 - x_2)^2 + \sin(x_1) \cdot e^{[1 - \cos(x_2)]^2} + \cos(x_2) \cdot e^{[1 - \sin(x_1)]^2}$	$[-2\pi, 2\pi]$	-106.76
f_8	Leon	$100\left(x_2 - x_1^3\right)^2 + (1 - x_1)^2$	$[-1.2, 1.2]$	0
f_9	Griewangk	$1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(x_i/\sqrt{i})$	$[-100, 100]$	0
f_{10}	Booth	$(x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	$[-10, 10]$	0
f_{11}	Bukin4	$100x_2^2 + 0.01\lvert x_1 + 10\rvert$	$[-15, -5]$ $[-5, 3]$	0
f_{12}	Carrom Table	$-\frac{1}{30}e^{2\lvert 1 - \sqrt{x_1^2 + x_2^2}/\pi\rvert}\cos^2(x_1)\cos^2(x_2)$	$[-10, 10]$	24.15
f_{13}	Crossfunc	$-0.0001\left(\lvert \sin(x_1)\sin(x_2)\exp\left(\left\lvert 100 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert\right)\rvert + 1\right)^{0.1}$	$[-10, 10]$	0
f_{14}	Himmelblau	$(x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$.	$[-5, 5]$	0
f_{15}	Matyas	$0.26\left(x_1^2 + x_2^2\right) - 0.48x_1x_2$	$[-10, 10]$	0

Table 3. Estimated cluster counts.

Dataset	Cal. Har.	Sil.	Gap	Dav. Boul.
B_1	100	2	97	2
B_2	98	2	97	2
B_3	98	2	97	2
B_4	97	2	100	2
B_5	99	2	98	2
B_6	98	2	2	2
B_7	100	2	2	2
B_8	100	2	2	2
f	99	2	2	2

test data, this process was repeated. For each benchmark function, tests were repeated with 10 different random seed values to decrease the effect of randomness on the results (Hamzacebi, 2008). For these data, the results were averaged over 10.

The performances of the algorithms were measured on the basis of two criteria: mean squared error (MSE) and mean absolute error (MAE),

$$\text{MSE} = \frac{1}{N_{\text{test}}} \sum_{q=1}^{N_{\text{test}}} (y_{\text{test}_q} - Y_q(\vec{x}_{\text{inp}}))^2, \tag{7}$$

$$\text{MAE} = \frac{1}{N_{\text{test}}} \sum_{q=1}^{N_{\text{test}}} |y_{\text{test}_q} - Y_q(\vec{x}_{\text{inp}})|, \tag{8}$$

where N_{test} is the number of test data, y_{test_q} is the actual value of the q-th test data and $Y_q(\vec{x}_{\text{inp}})$ is the predicted value of the q-th test data.

Average results of the SGRNN and the hybrid GRNN with clustering algorithms in terms of MSE and MAE are given in Tables 4–6. The tables illustrate the prediction error values of the SGRNN and percentages of the error changes with respect to the SGRNN. While '0' means the same result as the one obtained from the SGRNN, '–' indicates a worse result. The others denote positive improvements on the SGRNN. The results show that although the clustering algorithms greatly reduce the complexity of the pattern layer, the negative impact over all the test functions can be seen clearly. In all clustering algorithms, the increase in the error rate is proportional to the increase in the number of clusters.

In a detailed examination of the k-means GRNN, when k is increased from 5 to 10 and 20, the differences between the SGRNN and k-means-GRNN increase. If we calculate the average change in MSE rates for synthetic and real datasets, there are -7.88% and -6.38% decreases in prediction accuracy for 5 clustered GRNNs, respectively. When we increase the cluster count to 10, the average error percentage increases to -14.12% -11.92% and, when the cluster count is 20, the average error percentage increases to -28.16% and -27.87% for synthetic and real datasets, respectively. A similar relation between the increase in error rate and cluster count can also be observed in the MAE measurements.

According to Table 5, the results obtained for the FMC algorithm are similar to those for the k-means GRNN. According to both results, it is clear that the clustering process decreases the performance of the SGRNN, and this ratio increases in proportion to the number of clusters, as in the k-means-GRNN. However, when the effects of the WTA algorithm on the GRNN are examined, the results seem to be worse than for the other clustering algorithms. This finding shows that the competitive learning algorithm based on the principle of the "winner takes all" does not work well with the GRNN.

It is assumed that if the most similar training data to the test record are acquired by clustering approaches and a different tuned sigma value is used for each cluster, the prediction error rate will decrease with respect to the SGRNN. However, the results given in Tables 4, 5 and 6 indicate that using clustering methods as a pre-processor made conventional GRNN algorithms performance sensitive to the newly emerged inner outliers. For example, while the SGRNN is steadier than the 5-means GRNN, the 5-means GRNN is steadier than the 10-means GRNN and the 20-means GRNN. This finding clearly shows the effects of emerging cluster-outliers on the prediction accuracy.

According to Table 7, the kNN-GRNN for $k = 5$ provides better or equal prediction performances for four test functions (f_2, f_3, f_8, f_{13}) and three UCI datasets in terms of MSE by using merely five training data. When k is increased to 10 and 20, the results indicate that the differences between the performances of the SGRNN and the kNN-GRNN decrease. Although there are significant reductions in the numbers of training data, there is no consistency and reliability in the achieved results. Additionally, kNN requires more calculation costs for sorting and selecting the nearest k training data for each test record. By taking these into consideration, we can conclude that performance of the kNN-GRNN is not sufficient to improve the overall prediction accuracy and speed.

The comparative results for the SOM-GRNN and the SGRNN are listed in Table 8. It can be clearly seen that SOM-GRNNs with sizes of 8×8 and 10×10 produce either better or the same results as the SGRNN for all given test functions and very small differences for real-world datasets in both the error measurements, MSE and MAE. Moreover, the proposed pre-processing algorithm produces more consistent results than the clustering-GRNN and kNN-GRNN algorithms.

When the performance of pre-processing algorithms is compared, clustering-GRNN algorithms require 80%, 90% and 95% less training data for 5, 10 and 20 cluster counts, respectively. However, none of the clustering

Table 4. Error values of the SGRNN and the deviation percentages of the k-means-GRNN algorithm from the SGRNN.

	MSE				MAE			
	SGRNN	$k=5$	$k=10$	$k=20$	SGRNN	$k=5$	$k=10$	$k=20$
f_1	1E+07	-5.94	-6.46	-8.96	984.34	-5.55	-6.47	-12.03
f_2	49.893	-4.96	-7.49	-14.88	5.51	-1.37	-2.28	-4.95
f_3	11269	-3.37	-6.37	-12.47	77.69	-1.57	-2.68	-4.93
f_4	0.4254	-4.2	-5.78	-9.28	0.5	-1.55	-1.83	-3.9
f_5	1E+08	-4.09	-7.9	-22.07	4252.4	-3.44	-7.17	-15.58
f_6	32376	-6.78	-12.71	-23.52	86.5	-3.71	-7.27	-13.76
f_7	26.703	-13.63	-27.78	-52.55	3.21	-6.12	-10.95	-18.91
f_8	349.75	-4.28	-7.28	-24.17	8.37	-4.48	-10.51	-25.75
f_9	3.1504	-6.64	-14.01	-26.47	1.27	-2.89	-6.49	-11.37
f_{10}	434.14	-16.43	-37.56	-66.28	12.78	-7.99	-17.18	-27.03
f_{11}	668.95	-11.39	-23.29	-37.9	16.29	-5.47	-12.68	-20.67
f_{12}	0.6085	-7.83	-6.89	-13.76	0.25	-4.14	-4.76	-8.66
f_{13}	0	-3.96	-5.7	-28.2	0	-1.15	-2.25	-14.05
f_{14}	290.83	-7.76	-14.86	-25.53	9.28	-4.21	-9.45	-14.59
f_{15}	1.1253	-17.05	-27.75	-56.45	0.64	-9.03	-14.27	-25.74
B_1	0.0171	-7.9	-27.97	-25.59	0.1	-3.56	-8.81	-9.33
B_2	0	-1.34	-1.88	-3.08	0	-1.39	-1.21	-1
B_3	0.0023	-2.92	-2.97	-3.31	0.03	-0.14	-1.06	-0.59
B_4	0.0276	6.44	7.77	-0.02	0.12	2.91	2.74	1.98
B_5	0.0036	-26.84	-33.59	-51.85	0.03	-23.48	-31.03	-43.95
B_6	0.0043	-14.54	-27.26	-45.5	0.04	-11.75	-20.22	-32.06
B_7	0.0181	-1.64	-6.38	-10.31	0.11	0.39	-2.08	-2.12
B_8	0.0151	-2.31	-3.14	-4.12	0.1	0.18	-1.48	-1.66

Table 5. Error values of the SGRNN and the deviation percentages of the FMC-GRNN algorithm from the SGRNN.

	MSE				MAE			
	SGRNN	$k=5$	$k=10$	$k=20$	SGRNN	$k=5$	$k=10$	$k=20$
f_1	1E+07	-7.11	-8.14	-23.25	984.34	-5.66	-8.18	-12.97
f_2	49.893	-5.44	-8.66	-16.08	5.51	-1.51	-2.63	-5.61
f_3	11269	-3.73	-7.69	-12.59	77.69	-1.74	-3.22	-4.97
f_4	0.4254	-3.58	-3.11	-10.43	0.5	-1.63	-1.53	-4.06
f_5	1E+08	-4.53	-11.15	-21.55	4252.4	-3.09	-8.12	-15.62
f_6	32376	-8.99	-15.67	-21.16	86.5	-4.22	-8.37	-13.18
f_7	26.703	-13.47	-29.42	-51.42	3.21	-5.69	-11	-19.13
f_8	349.75	-5.41	-10.16	-25.97	8.37	-4.5	-10.81	-26.39
f_9	3.1504	-7.91	-14.35	-28.11	1.27	-3.51	-6.21	-11.49
f_{10}	434.14	-16.3	-39.46	-61.84	12.78	-8.24	-15.65	-24.67
f_{11}	668.95	-10.53	-22.84	-47.03	16.29	-5.32	-11.84	-22.66
f_{12}	0.6085	-3.85	-7.64	-20.58	0.25	-3.75	-5.14	-10.02
f_{13}	0	-2.5	-5.29	-10.51	0	-1.18	-1.97	-6
f_{14}	290.83	-8.46	-14.71	-30.01	9.28	-4.76	-8.88	-15.1
f_{15}	1.1253	-14.57	-27.86	-54.36	0.64	-8	-13.7	-24.71
B_1	0.0171	-9.24	-12.77	-23.53	0.1	-3	-5.52	-10.48
B_2	0	-2.61	-2.45	-5.66	0	-1.67	-1.52	-2.68
B_3	0.0023	-2.37	-3.96	-4.29	0.03	-0.73	-1.3	-1.19
B_4	0.0276	5.34	8.87	-0.1	0.12	2.01	2.71	-0.29
B_5	0.0036	-11.08	-33.26	-91.05	0.03	-14.48	-34.9	-55.58
B_6	0.0043	-10.24	-28.16	-59.85	0.04	-11.45	-23.85	-37.71
B_7	0.0181	-2.2	-7.72	-9.83	0.11	-0.13	-2.49	-3.2
B_8	0.0151	-3.6	-4.34	-6.94	0.1	-2.03	-2.24	-3.14

Table 6. Error values of the SGRNN and the deviation percentages of the WTA-GRNN algorithm from the SGRNN.

	MSE				MAE			
	SGRNN	$k=5$	$k=10$	$k=20$	SGRNN	$k=5$	$k=10$	$k=20$
f_1	1E+07	-7	-18.4	-260.5	984.34	-6.36	-17.49	-89.55
f_2	49.893	-17.23	-37.18	-106.7	5.51	-6.01	-13.21	-38.13
f_3	11269	-6.26	-7.04	-14.2	77.69	-3.54	-2.89	-6.35
f_4	0.4254	-7.68	-35.85	-48.11	0.5	-2.45	-15.56	-19.06
f_5	1E+08	-21.66	-167.6	-917.1	4252.4	-12.22	-57.39	-162.3
f_6	32376	-13.21	-32.12	-250.8	86.5	-7.26	-16.77	-81.87
f_7	26.703	-36.85	-127.1	-517.4	3.21	-13.5	-35.72	-113.2
f_8	349.75	-15.17	-144.7	-275.6	8.37	-13.34	-65.81	-94.32
f_9	3.1504	-58.84	-63.45	-81.72	1.27	-30.58	-31.59	-37.41
f_{10}	434.14	-31.24	-88.93	-509.3	12.78	-13.55	-29.62	-101.5
f_{11}	668.95	-27.4	-121.3	-822.2	16.29	-12.36	-40.43	-149.8
f_{12}	0.6085	-8.09	-12	-61.69	0.25	-4.91	-9.02	-32.05
f_{13}	0	-5.07	-10.61	-35.11	0	-2.77	-5.58	-18.65
f_{14}	290.83	-17.82	-45.4	-276.1	9.28	-9.9	-22.39	-90.64
f_{15}	1.1253	-25.24	-68.64	-349.8	0.64	-13.6	-26.2	-87.66
B_1	0.0171	-10.62	-39.57	-21.52	0.1	-3.55	-14.85	-9.21
B_2	0	-1E+06	-67.96	-48.2	0	-1329	-24.14	-15.99
B_3	0.0023	-40.61	-100.8	-159.1	0.03	-16.18	-37.56	-51.41
B_4	0.0276	-3.71	-22.48	-9.67	0.12	-5.34	-14.77	-9.96
B_5	0.0036	-45.9	-88.04	-96.77	0.03	-29.77	-51.86	-56
B_6	0.0043	-31.23	-79.11	-77.51	0.04	-19.54	-41.95	-44.17
B_7	0.0181	-4.07	-23.01	-22.82	0.11	-0.67	-6.59	-9.11
B_8	0.0151	-7.51	-19.03	-23.36	0.1	-3.25	-6.81	-10.32

Table 7. Error values of the SGRNN and the deviation percentages of the kNN-GRNN algorithm from the SGRNN.

	MSE				MAE			
	SGRNN	$k=5$	$k=10$	$k=20$	SGRNN	$k=5$	$k=10$	$k=20$
f_1	1E+07	-1.94	-0.31	-0.01	984.34	-2.97	-0.41	-0.01
f_2	49.893	4.69	0.9	0.01	5.51	2.78	0.51	0.01
f_3	11269	0.04	0	0	77.69	0.02	0	0
f_4	0.4254	-6.33	-0.57	0.3	0.5	-3.83	-0.92	-0.03
f_5	1E+08	-2.82	-1.14	-0.06	4252.4	-4.02	-0.83	-0.02
f_6	32376	-1.87	-0.16	0	86.5	-2.07	-0.2	0
f_7	26.703	-2.11	0.93	0.13	3.21	-1.64	0.28	0.05
f_8	349.75	24.65	11.34	2.42	8.37	11.75	7.53	2.9
f_9	3.1504	-2.17	-0.07	0	1.27	-1.25	-0.04	0
f_{10}	434.14	-17.64	-4.33	-0.4	12.78	-10.7	-2.87	-0.24
f_{11}	668.95	-8.53	-1.82	-0.08	16.29	-6.81	-1.43	-0.03
f_{12}	0.6085	-0.44	0.18	0	0.25	1.22	0.34	0.01
f_{13}	0	1.23	0.6	0.04	0	2.62	0.95	0.07
f_{14}	290.83	-3.1	-0.34	0	9.28	-2.49	-0.28	0
f_{15}	1.1253	-12.69	-3.33	-0.2	0.64	-7.52	-1.89	-0.09
B_1	0.0171	-10.5	-4.75	0.38	0.1	-2.85	-1.64	0.98
B_2	0	-0.8	1.88	2.13	0	0.15	1.4	1.4
B_3	0.0023	1.37	2.27	1.54	0.03	4.02	3.35	1.95
B_4	0.0276	6.19	4.73	3.18	0.12	6.9	4.71	2.57
B_5	0.0036	2.5	1.42	1.05	0.03	3.67	1.12	1.05
B_6	0.0043	1.78	1.35	1.04	0.04	2.57	1.39	1.04
B_7	0.0181	-15.08	-6.63	-1.92	0.11	-1.53	0.64	1.81
B_8	0.0151	-12.18	-5.06	-1.15	0.1	-2.96	-0.85	0.4

Table 8. Error values of the SGRNN and the deviation percentages of the SOM-GRNN algorithm from the SGRNN.

	MSE				MAE			
	SGRNN	8×8	10×10	16×16	SGRNN	8×8	10×10	16×16
f_1	1E+07	0	0	-0.01	984.34	0	0	-0.02
f_2	49.893	0	0	0.02	5.51	0	0	0.01
f_3	11269	0	0	0	77.69	0	0	0
f_4	0.4254	0	0.02	0.1	0.5	0	0	-0.09
f_5	1E+08	0	0	-0.07	4252.4	0	0	-0.07
f_6	32376	0	0	-0.01	86.5	0	0	-0.01
f_7	26.703	0	0	0.06	3.21	0	0	0.01
f_8	349.75	0	0.02	0.57	8.37	0.01	0.1	1.07
f_9	3.1504	0	0	0.02	1.27	0	0	0.01
f_{10}	434.14	0	0	-0.38	12.78	0	0	-0.21
f_{11}	668.95	0	0	-0.17	16.29	0	0	-0.11
f_{12}	0.6085	0	0	0	0.25	0	0	0
f_{13}	0	0	0	0.01	0	0	0	0.04
f_{14}	290.83	0	0	-0.01	9.28	0	0	-0.02
f_{15}	1.1253	0	0	-0.38	0.64	0	0	-0.23
B_1	0.0171	-0.45	-0.11	-6.36	0.1	-0.13	0.02	-1.23
B_2	0	0	-0.02	-0.38	0	0.01	0	-0.18
B_3	0.0023	0	-0.01	0	0.03	0	0	0.03
B_4	0.0276	-0.07	0.3	1.15	0.12	0.06	0.27	0.91
B_5	0.0036	-0.01	-0.03	-0.39	0.03	-0.02	-0.06	-0.27
B_6	0.0043	0	-0.02	-0.01	0.04	-0.03	-0.04	-0.09
B_7	0.0181	-0.55	0.08	-1.09	0.11	0.33	0.72	1.02
B_8	0.0151	-0.16	-0.36	-0.82	0.1	0.08	-0.09	-0.03

algorithms produce results as reliable as the SGRNN. The kNN-GRNN, in contrast, employed 5, 10 and 20 nearest neighbours instead of all training data. Although, the results were obtained by using much fewer training data, there was still no important improvement in the performance results for $k = 5$ and $k = 10$. However, almost the same prediction performances are obtained by using 20 training data. Nevertheless, the prediction time calculation is still one of the greatest drawbacks of this algorithm.

When the SOM size is selected as 8 and 10, the algorithm produces almost the same results as the SGRNN, and when the SOM size is increased to 16, a slight difference in the performance starts to be observed depending on the reduction of training data. In addition, the SOM-GRNN reduces the number of training data from 900 to 99, 64 and 25 for 8, 10 and 16 SOM sizes (64, 100 and 256 cluster counts), respectively. Due to the non-homogeneous dispersion of training data over the clusters, the number of utilized training data could not be calculated directly from the cluster counts. Hence, the total number of utilized training data were counted while the prediction operation was executed, and then an average value was calculated. If it is assumed that the training data were dispersed homogeneously over the clusters and the map structure is hexagonal, then the following formula can be used to calculate the number of utilized training data:

$$R = 7\frac{N}{k}, \tag{9}$$

where 7 represents the cluster count (BMU and its 6 neighbours in the hexagon map). If this equation is used, almost the same result with the given counts will be obtained.

7. Conclusion

In this study, a new method to solve the sample complexity problem of the GRNN has been proposed. The performance of the proposed method has been evaluated on fifteen benchmark test functions and eight different UCI datasets and compared with that of the SGRNN as well as popular hybrid methods used in the literature. Although all the pre-processing algorithms given in the literature reduce the data complexity of the GRNN, prediction accuracy could not be enhanced adequately.

The proposed method employs an SOM as a pre-processing algorithm for the GRNN. Instead of using all training data in GRNNs pattern layer, using a small group of data that are the neighbours of the test record significantly reduces memory complexity. As a

pre-processor, an SOM has been employed to cluster the training data off-line. Whenever a test record needs to be predicted, it is subjected to the SOM to identify the similar cases in the training data. The data in the best matching cluster and its neighbouring clusters are utilized as the pattern layer of the SOM-GRNN.

The proposed method provides three advantages over compared methods. The first advantage is a reduction in the pattern layer size without an inner outlier problem. The second is data reduction with an acceptable computational complexity. The last advantage is the production of almost the same prediction results as the SGRNN with fewer training data.

Consequently, the major drawback of the pattern layer complexity of the SGRNN for huge datasets is overcome. The proposed method decreases the number of required pattern layer neurons by approximately 89% for an SOM of size 8 without any loss in performance compared with the SGRNN. Furthermore, the numerical results denoting accuracy and calculation complexity show the superiority of the proposed method over popular approaches such as the clustering-GRNN and the kNN-GRNN.

References

Bache, K. and Lichman, M. (2013). *UCI Machine Learning Repository*, University of California, Irvine, CA.

Berkhin, P. (2002). Survey of clustering data mining techniques, *Technical report*, Accrue Software, `https://www.cc.gatech.edu/~isbell/reading/papers/berkhin02survey.pdf`.

Bezdek, J.C., Ehrlich, R. and Full, W. (1984). FCM: The fuzzy c-means clustering algorithm, *Computers & Geosciences* **10**(2–3): 191–203.

Bowden, G.J., Dandy, G.C. and Maier, H.R. (2005). Input determination for neural network models in water resources applications. Part 1—Background and methodology, *Journal of Hydrology* **301**(1): 75–92.

Caliński, T. and Harabasz, J. (1974). A dendrite method for cluster analysis, *Communications in Statistics—Theory and Methods* **3**(1): 1–27.

Carrasco Kind, M. and Brunner, R.J. (2014). SOMs: Photometric redshift PDFs with self-organizing maps and random atlas, *Monthly Notices of the Royal Astronomical Society* **438**(4): 3409–3421.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification, *IEEE Transactions on Information Theory* **13**(1): 21–27.

Davies, D.L. and Bouldin, D.W. (1979). A cluster separation measure, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **1**(2): 224–227.

Hamzacebi, C. (2008). Improving genetic algorithms performance by local search for continuous function optimization, *Journal of Applied Mathematics and Computation* **196**(1): 309–317.

Harkanth, S. and Phulpagar, B.D. (2013). A survey on clustering methods and algorithms, *International Journal of Computer Science and Information Technologies* **4**(5): 687–691.

Hartigan, J.A. and Wong, M.A. (1979). Algorithm AS 136: A k-means clustering algorithm, *Journal of the Royal Statistical Society C: Applied Statistics* **28**(1): 100–108.

Husain, H., Khalid, M. and R., Y. (2004). Automatic clustering of generalized regression neural network by similarity index based fuzzy c-means clustering, *IEEE Region 10 Conference, Chiang Mai, Thailand*, pp. 302–305.

Jain, A.K., Mao, J. and Mohiuddin, K.M. (1998). Artificial neural networks: A tutorial, *IEEE Computer* **29**(3): 31–44.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps, *Biological Cybernetics* **43**(1): 59–69.

Kokkinos, Y. and Margaritis, K.G. (2015). A fast progressive local learning regression ensemble of generalized regression neural networks, *Proceedings of the 19th Panhellenic Conference on Informatics, Athens, Greece*, pp. 109–114 .

Kolesnikov, A., Trichina, E. and Kauranne, T. (2015). Estimating the number of clusters in a numerical data set via quantization error modeling, *Pattern Recognition* **48**(3): 941–952.

Kotsiantis, S.B. and Pintelas, P.E. (2004). Recent advances in clustering: A brief survey, *WSEAS Transactions on Information Science and Applications* **1**(1): 73–81.

Krenker, A., Bester, J. and Kos, A. (2011). Introduction to the artificial neural networks, *in* K. Suzuki (Ed.), *Artificial Neural Networks—Methodological Advances and Biomedical Applications*, Intech, Rijeka, pp. 3–18.

Maier, H. and Dandy, G. (1997). Determining inputs for neural network models of multivariate time series, *Microcomputers in Civil Engineering* **12**(5): 353368.

Rama, B., Jayashree, P. and Jiwani, S. (2010). A survey on clustering, current status and challenging issues, *International Journal on Computer Science and Engineering* **2**(9): 2976–2980.

Rousseeuw, P.J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, *Journal of Computational and Applied Mathematics* **20**: 53–65.

Sabo, K. (2014). Center-based l_1-clustering method, *International Journal of Applied Mathematics and Computer Science* **24**(1): 151–163, DOI: 10.2478/amcs-2014-0012.

Specht, D.F. (1991). A general regression neural network, *IEEE Transactions on Neural Networks* **2**(6): 568–576.

Szemenyei, M. and Vajda, F. (2017). Dimension reduction for objects composed of vector sets, *International Journal of Applied Mathematics and Computer Science* **27**(1): 169–180, DOI: 10.1515/amcs-2017-0012.

Tang, K., Li, X., Suganthan, P.N., Yang, Z. and Weise, T. (2009). Benchmark functions for the CEC'2010 special session and competition on large scale global optimization, *Technical report*, Nature Inspired Computation and Applications Laboratory, USTC, Hefei.

Tibshirani, R., Walther, G. and Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic, *Journal of the Royal Statistical Society B: Statistical Methodology* **63**(2): 411–423.

Yuen, R.K.K., Lee, E.W.M., Lim, C.P. and Cheng, G.W.Y. (2004). Fusion of GRNN and FA for online noisy data regression, *Neural Processing Letters* (19): 227–241.

Zhao, S.J., Zhang, J.L., Li, X. and Song, W. (2007). Generalized regression neural network based on fuzzy means clustering and its application in system identification, *Proceedings of the International Symposium on Information Technology Convergence, Joenju, South Korea*, pp. 13–16.

Zheng, L.G., Yu, M.G., Yu, S.J. and Wang, W. (2008). Improved prediction of nitrogen oxides using GRNN with k-means clustering and EDA, *Proceedings of the 4th International Conference on Natural Computation, Jinan, China*, pp. 91–95.

Serkan Kartal received his BE, MSc and PhD degrees from the Department of Computer Engineering, Cukurova University, in 2010, 2013 and 2017, respectively. He has been working there since 2011. His research interests include artificial neural networks, artificial intelligence and software engineering.

Mustafa Oral received his BE degree from the Department of Electronics and Communication Engineering, Yildiz Technical University, in 1990. He then received his PhD degree from the Department of Computer Science, University of South Wales, in 1998. He works as an associate professor in the Department of Computer Engineering, Cukurova University. His research interests include swarm intelligence, image processing and data compression.

Buse Melis Özyildirim received her BE and MSc degrees from the Department of Computer Engineering, Cukurova University, in 2010 and 2012, respectively. She then received her PhD degree from the Department of Electrical and Electronics Engineering of the same university in 2015. She now works as an assistant professor in the Department of Computer Engineering there. Her research interests include artificial neural networks.

Received: 27 April 2017
Revised: 11 August 2017
Re-revised: 26 September 2017
Accepted: 25 October 2017