ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD THESIS

Serkan KARTAL

ENHANCED GENERALIZED REGRESSION NEURAL NETWORK FOR LARGE DATASETS

DEPARTMENT OF COMPUTER ENGINEERING

ADANA-2017

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

ENHANCED GENERALIZED REGRESSION NEURAL NETWORK FOR LARGE DATASETS

Serkan KARTAL

PhD THESIS

DEPARTMENT OF COMPUTER ENGINEERING

We certify that the thesis titled above was reviewed and approved for the award of degree of the Doctor of Philosophy by the board of jury on 28/09/2017

……...	……	……
Assoc. Prof. Dr. Mustafa ORAL	Assoc. Prof. Dr. Zekeriya TÜFEKÇİ	Assoc. Prof. Dr. H. Oya YÜREĞİR
SUPERVISOR	MEMBER	MEMBER

……	…….
Assoc. Prof. Dr. Serdar YILDIRIM	Asst. Prof. Dr. Yakup KUTLU
MEMBER	MEMBER

This PhD Thesis is written at the Department of Institute of Natural And Applied Sciences of Çukurova University.
Registration Number:

Prof. Dr. Mustafa GÖK
Director
Institute of Natural and Applied Sciences

Note: The usage of the presented specific declerations, tables, figures, and photographs either in this thesis or in any other reference without citiation is subject to "The law of Arts and Intellectual Products" number of 5846 of Turkish Republic

ABSTRACT

PhD THESIS

ENHANCED GENERALIZED REGRESSION NEURAL NETWORK FOR LARGE DATASETS

Serkan KARTAL

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER ENGINEERING

Supervisor : Assoc. Prof. Dr. Mustafa ORAL
Year: 2017, Pages: 113
Jury : Assoc. Prof. Dr. Mustafa ORAL
: Assoc. Prof. Dr. Zekeriya TÜFEKÇİ
: Assoc. Prof. Dr. H. Oya YÜREĞİR
: Assoc. Prof. Dr. Serdar YILDIRIM
: Asst. Prof. Dr. Yakup KUTLU

Generalized Regression Neural Network (GRNN) is a kind of radial basis function neural networks used for function approximation and prediction. It has two important properties that make it alternative to other prediction methods such as MultiLayer Perceptron (MLP) and Support Vector Machines (SVM) in many applications. These are easy modelling structure, and one pass learning. Since the number of neurons at GRNN's pattern layer is proportional to the number of training samples in dataset, increase in memory usage and computational time will emerge for huge datasets. Therefore, there is a strong need for pattern layer size reduction. It may provide not only simplification in structure but also increase in prediction accuracy. In this study, three different methods are proposed to solve pattern layer size problem. Performances of the proposed methods were evaluated by using fifteen benchmark test functions and eight different UCI datasets and compared with that of standard GRNN as well as popular hybrid methods used in literature.

Key Words: Generalized regression neural network, Artificial neural networks, Nearest neighbour, Pattern reduction and Reduced dataset.

ÖZ

DOKTORA TEZİ

BÜYÜK VERİ SETLERİ İÇİN GELİŞTİRİLMİŞ GENELLEŞTİRİLMİŞ REGRESYON YAPAY SİNİR AĞI

Serkan KARTAL

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Danışman : Doç. Dr. Mustafa ORAL
Yıl: 2017, Sayfa: 113
Jüri : Doç. Dr. Mustafa ORAL
: Doç. Dr. Zekeriya TÜFEKÇİ
: Doç. Dr. H. Oya YÜREĞİR
: Doç. Dr. Serdar YILDIRIM
: Yrd. Doç. Dr. Yakup KUTLU

Genelleştirilmiş Regresyon Yapay Sinir Ağı (GRYSA), fonksiyon yaklaştırma ve tahmin işlemleri için kullanılan, radyal tabanlı fonksiyon yapay sinir ağlarından biridir. Birçok uygulamada Çok Katmanlı Algılayıcı (CKA) ve Destek Vektör Makineleri (DVM) gibi tahmin edici metotlara bir alternatif olarak kullanılabilmesini sağlayan iki önemli özelliğe sahiptir. Bu özellikler, kolay modellenebilmesi ve tek geçişte öğrenebilmesidir. GRYSA'nın örüntü katmanında bulunan nöron sayısının veri setindeki eğitim verisi sayısına eşit olmasından dolayı, büyük veri setlerinde bellek kullanımında ve hesaplama zamanlarında artış meydana gelmektedir. Bu yüzden, örüntü katmanı yapısının küçültülmesine önemli bir ihtiyaç vardır. Bu yalnızca yapıyı basitleştirmekle kalmayacak, aynı zamanda tahmin tutarlılığını da arttıracaktır. Bu çalışmada, örüntü katmanı boyutu problemini çözmek için üç farklı metot önerilmiştir. Önerilen metotların performansı 15 test fonksiyonu ve 8 farklı UCI veri seti ile değerlendirilip sonuçlar standart GRYSA ve literatürde kullanılan popüler karma yöntemler ile karşılaştırılmıştır.

Anahtar Kelimeler: Genelleştirilmiş regresyon yapay sinir ağı, Yapay sinir ağları, En yakın komşuluk, Örüntü azaltma, Azaltılmış veri seti

EXPANDED ABSTRACT

Artificial Neural Networks (ANN) are inspired by biological neural networks. They are employed to solve variety of problems in pattern classification, clustering, function approximation, optimization and prediction. However, choosing the right learning model depends on the type of the application and data representation of a given problem. In this study, one of the most popular ANN algorithms GRNN was examined and new algorithms were proposed to improve its efficiency.

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 which need larger number of iterations to be performed in training, GRNN does not require an iterative training process (Cover and Hart, 1967). As a result of the simplicity of network structure and ease of implementation, it has been widely employed to solve variety of problems.

However, in GRNN, the number of neurons in the pattern layer is proportional to the number of training samples in dataset. The use of GRNN in applications that have relatively large data sets becomes troublesome for both architecture and speed-wise. Huge numbers of neurons in pattern layer require substantial increase in memory usage and cause substantial decrease in calculation speed. Therefore, there is a strong need for pattern layer size reduction.

In this study, three different hybrid GRNN models named as k-means GRNN with outlier detection, Angle Based Nearest Neighbour-GRNN (ABNN-GRNN) and Self-Organizing Maps-GRNN (SOM-GRNN) were proposed to select the significant training data and reduce the data complexity of the GRNN.

In order to reduce space and data complexity of GRNN, use of clustering algorithm as a pre-processor has been previously studied in the literature. Thus, firstly an appropriate clustering algorithm was determined with a preliminary

study. For this purpose, a quick study was conducted on 10 test functions with 4 popular clustering algorithms; K-means, Hierarchical clustering, Gaussian mixture model and Fuzzy c-means. All the methods were implemented and analysed as pre-processors of GRNN. According to the obtained results, the most popular clustering algorithm in the literature, k-means, was selected to be used as pre-processing clustering algorithm. Secondly, cluster count estimation algorithms are used to determine the best number of cluster. However, the appropriate cluster counts for the used data sets could not be determined. Therefore, according to selected three different data reduction percentage values (80, 90, 95), three different k-cluster counts (5, 10, 20) were utilized to demonstrate the prediction performances of the clustering-GRNN algorithms. When the k-means GRNN algorithm is executed, it is seen that the sample data that lay in between cluster borders (inner cluster outliers) produce large residuals resulting negative influence on prediction performance. Therefore, they need special attention. In the first part of this study, we detect outlier data by using probabilistic density function after applying k-means algorithm, and to reduce negative effects of the outlier data. As a result, these effects on prediction performance has been reduced significantly and proposed approach achieved almost similar prediction performances to standard GRNN's by using ~95% less training samples.

kNN (k-Nearest Neighbours) algorithm, which is the popular alternative to the clustering algorithms, was examined in the second method. Firstly, the kNN-GRNN algorithm was developed and examined, secondly its weakness was detected, and lastly an alternative method that can be used in three-dimensional data was proposed. The proposed algorithm is based on ABNN algorithm and suitable for three-dimensional datasets. Algorithm divides the training search space into fixed sized angle slices, and searches for the nearest training datum within each angle slice. The most similar datum from each slice is taken and fed into the GRNN model for prediction. For the performance evaluation, the test functions given in the material section were used. Results were compared with that of

SGRNN (Standard GRNN) and the proposed method ABNN-GRNN. Results indicate that the proposed method provides 99.33% reduction in pattern layer size and accuracy is also increased maximally to 68.58% comparing to SGRNN.

In the proposed third method, SOM based hybrid GRNN algorithm which provides further improvement on the results is presented. In this method, SOM structure was introduced as pre-processor for the GRNN. Firstly, an SOM map is generated for the training dataset. Secondly, each training datum is labelled with the most similar map unit. Lastly, when a new test datum is applied to the network, the most similar map units are detected and the training data which have the same labels with these detected units are fed into the network instead of all training dataset. This scheme provides considerable reduction in 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.

In evaluation process, 10-fold cross validation was carried out for both test functions and datasets. Tests were repeated with 10 different random seed values for all benchmark functions to decrease the effect of randomness on the results. Therefore, average error values were used to present the efficiency of predictions. Three standard error measures, MSE, MAE and MAPE, were used to evaluate the performance of the methods.

It has been observed that while all the strategies reduce the data complexity meaningfully, their effects on the performance of the SGRNN may vary. This is the result of encountered problems and varying solution proposals. In the first method, outlier problem encountered by k-means GRNN algorithm, was solved reasonably through outlier detection algorithm based on probability density function. In the second method, when kNN-GRNN algorithm was executed, getting stuck on data heap problem encountered and it was solved for 3-dimensional problems by using angle based nearest neighbourhood, instead of kNN. Moreover, when we calculated the average change in error rates ~22% increase in prediction

performance was provided in terms of MSE by using ~99% (6 in 900) less number of training data. In the third method, a new SOM based GRNN algorithm that selects the significant training data and reduces the data complexity is proposed. Unlike traditional clustering algorithms, SOM clustering forms a semantic map where similar cluster centres are mapped close together and dissimilar ones apart. Depending on this feature of the SOM, the major drawback of pattern layer complexity of SGRNN for huge data sets has been overcome. Proposed method decreased the number of required pattern layer neurons around 89% for SOM size 8 without any loss in performance comparing to SGRNN. Consequently, one of the biggest problem of the SGRNN, data complexity, is solved with proposed algorithms.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Assoc. Prof. Dr. Mustafa ORAL for his supervision guidance, encouragements, patience, motivation, useful suggestions and his valuable time for this work.

I would like to thank members of PhD thesis jury, Assoc. Prof. Dr. Zekeriya TÜFEKÇİ, Assoc. Prof. Dr. H. Oya YÜREĞİR, Assoc. Prof. Dr. Serdar YILDIRIM and Asst. Prof. Dr. Yakup KUTLU for their suggestions and corrections.

I would like to express my sincere appreciation to TUBİTAK for their financial support in the process of my PhD education.

Special thanks to Buse Melis Özyıldırım for her endless support, patience and motivation.

Finally, I would like to thank my family, who have supported me throughout entire process, both by their love and encouragements.

CONTENTS **PAGE**

ABSTRACT I
ÖZ II
EXPANDED abstract III
ACKNOWLEDGEMENTS VII
CONTENTS VIII
LIST OF TABLES XII
LIST OF FIGURES XIV
LIST OF ALGORITHMS XVI
LIST OF ABBREVIATIONS XVIII
1. INTRODUCTION 1
1.1. General Overview to the Artificial Neural Network 1
1.2. Choosing the Right Learning Model 3
1.3. Radial Basis Function (RBF) & General Regression Neural Network (GRNN) 3
1.4. Weaknesses of the GRNN 4
1.5. The Aims and Objectives 5
1.6. Contribution to Existing Knowledge 6
1.7. Main Findings of the Research 7
1.8. Outline of the Thesis 7
2. PREVIOUS WORKS 9
3. MATERIALS 15
3.1. Introduction 15
3.2. Generalized Regression Neural Network (GRNN) 15
3.3. An Overview to Clustering Algorithms 18
3.4. k-Means Clustering 21
3.5. k-Nearest Neighbour (kNN) 23
3.6. Self-Organizing Maps (SOM) 24

3.7. Probability Density Function....26
3.8. Standard Test Functions....27
3.8.1. F_1: Beale....29
3.8.2. F_2: Rastrigin....29
3.8.3. F_3: Schwefel....30
3.8.4. F_4: Ackley....30
3.8.5. F_5: Goldstein-Price....31
3.8.6. F_6: Six- Hump Camel....31
3.8.7. F_7: Bird....32
3.8.8. F_8: Leon....32
3.8.9. F_9: Griewangk....33
3.8.10. F_{10}: Giunta....33
3.8.11. F_{11}: Booth....34
3.8.12. F_{12}: Bukin4....34
3.8.13. F_{13}: Cross-In Tray Function....35
3.8.14. F_{14}: Himmelblau....35
3.8.15. F_{15}: Matyas....36
3.9. Real World Datasets....36
3.9.1. Airfoil Self-Noise Dataset....37
3.9.2. Air Quality Dataset....37
3.9.3. Combined Cycle Power Plant Dataset....38
3.9.4. Concrete Compressive Strength Dataset....39
3.9.5. Energy Efficiency Dataset....39
3.9.6. Wine Quality Datasets....40
3.10. Performance Evaluation Methods....41
3.10.1. N-Fold Cross Validation....41
3.10.2. Evaluation Metrics....41
4. PRELIMINARY STUDY....43
4.1. Cluster Count Estimation....43

5. METHODS 53
5.1. A Cluster Based Approach to Reduce Pattern Layer Size for GRNN 53
5.1.1. Inner-Outlier Detection Algorithm 54
5.1.2. Algorithm 57
5.1.3. Complexity Analysis 58
5.2. New Pattern Reduction Method for Generalized Regression Neural Network 59
5.2.1. Angle Based Nearest Neighbour (ABNN) 60
5.2.2. Algorithm 64
5.2.3. Complexity Analysis 65
5.3. Pattern Layer Reduction for Generalized Regression Neural Networks by Using Self Organizing Map 65
5.3.1. SOM-GRNN 66
5.3.2. Algorithm 69
5.3.3. Complexity Analysis 70
6. RESULTS AND DISCUSSIONS 73
6.1. Method 1 - A Cluster Based Approach to Reduce Pattern Layer Size for GRNN 73
6.2. Method 2 - New Pattern Reduction Method for Generalized Regression Neural Network 83
6.3. Method 3 - Pattern Layer Reduction for Generalized Regression Neural Networks by Using Self Organizing Map 91
6.4. Summary and Comparisons 96
7. CONCLUSION 105
REFERENCES 107
CURRICULUM VITAE 113

LIST OF TABLES **PAGE**

Table 3.1. Test Functions with Their Boundaries and Global Minimums 28
Table 3.2. Description of Datasets 36
Table 4.1. Estimated cluster counts 44
Table 4.2. Clustering - GRNN results with 5 cluster (MSE) 45
Table 4.3. Clustering - GRNN results with 5 cluster (MAE) 45
Table 4.4. Clustering - GRNN results with 5 cluster (MAPE) 46
Table 4.5. Clustering - GRNN results with 10 cluster (MSE) 46
Table 4.6. Clustering - GRNN results with 10 cluster (MAE) 47
Table 4.7. Clustering - GRNN results with 10 cluster (MAPE) 47
Table 4.8. Clustering - GRNN results with 20 cluster (MSE) 48
Table 4.9. Clustering - GRNN results with 20 cluster (MAE) 48
Table 4.10. Clustering - GRNN results with 20 cluster (MAPE) 49
Table 6.1. MSE 10-fold cross validation performances 75
Table 6.2. MAE 10-fold cross validation performances 76
Table 6.3. MAPE 10-fold cross validation performances 77
Table 6.4. MSE 10-fold cross validation performances 79
Table 6.5. MAE 10-fold cross validation performances 80
Table 6.6. MAPE 10-fold cross validation performances 81
Table 6.7. MSE 10-fold cross validation performances 84
Table 6.8. MAE 10-fold cross validation performances 85
Table 6.9. MAPE 10-fold cross validation performances 86
Table 6.10. MSE 10-fold cross validation performances 88
Table 6.11. MAE 10-fold cross validation performances 89
Table 6.12. MAPE 10-fold cross validation performances 90
Table 6.13. MSE 10-fold cross validation performances 92
Table 6.14. MAE 10-fold cross validation performances 93
Table 6.15. Clustering - GRNN results with 20 cluster (MAPE) 94

LIST OF FIGURES **PAGE**

Figure 1.1. Feed-forward (FNN) and recurrent (RNN) structure of an ANN 2
Figure 3.1. Schematic diagram of GRNN 17
Figure 3.2. An example view of k-means clustering algorithm with k=5 22
Figure 3.3. 10 Nearest Neighbours 23
Figure 3.4. General structure of SOM (Carrasco and Brunner, 2014) 24
Figure 3.5. Many samples from a multivariate normal distribution with μ=0, the two marginal distributions (Wikipedia, 2017) 27
Figure 4.1. K-means 50
Figure 4.2. Hierarchical Clustering 50
Figure 4.3. Gaussian Mixture Model 50
Figure 4.4. Fuzzy C-means 50
Figure 5.1. Standard GRNN 55
Figure 5.2. GRNN with k-means clustering algorithm 55
Figure 5.3. Inner-outlier detection 57
Figure 5.4. 10 Nearest Neighbours, over Stacked Data 61
Figure 5.5. Finding of the θ between two points 62
Figure 5.6. ABNN with θ = 40 63
Figure 5.7. General overview of the SOM-GRNN 67
Figure 5.8. Training data of most similar cluster on the Bird function 68
Figure 5.9. Training data of most similar and similar clusters on the Bird function 69
Figure 6.1. The percentages of the amount of data used by the algorithms for test functions and real world datasets 98
Figure 6.2. Percentages of error change of algorithms for test functions compared to the SGRNN 101
Figure 6.3. Percentages of error change of algorithms for real world datasets compared to the SGRNN 102

LIST OF ALGORITHMS **PAGE**

Algorithm 5.1. Algorithm of k-means GRNN with outlier detection..... 58
Algorithm 5.2. ABNN -GRNN Algorithm..... 64
Algorithm 5.3. SOM-GRNN Algorithm..... 70

LIST OF ABBREVIATIONS

ABNN : Angle Based Nearest Neighbour
ANN : Artificial Neural Network
BIRCH : Balanced Iterative Reducing and Clustering
CURE : Clustering Using REpresentatives
CCPP : Combined Cycle Power Plant
CKA : Çok Katmanlı Algılayıcı
DBCLASD : Distribution Based Clustering of LArge Spatial Databases
DBSCAN : Density Based Spatial Clustering of Applications with Noise
DENCLUE : DENsity CLUstEring
DVM : Destek Vektör Makineleri
EDA : Estimation of Distribution Algorithm
FA : Fuzzy ART
FCM : Fuzzy C-Means
FFN : Feed-Forward Networks
FFT : Farthest First Traversal
FMC : Fuzzy Means Clustering
GAGRNN : Genetic Algorithm Generalized Regression Neural Network
GMM : Gaussian Mixture Model
GRNN : Generalized Regression Neural Network
GRYSA : Genelleştirilmiş Regresyon Yapay Sinir Ağı
GT : Gas Turbines
kNN : k-Nearest Neighbours
MAE : Mean Absolute Error
MAPE : Mean Absolute Percentage Error
MLP : Multilayer Perceptron
MND : Multivariate Normal Distribution
MSE : Mean Squared Error

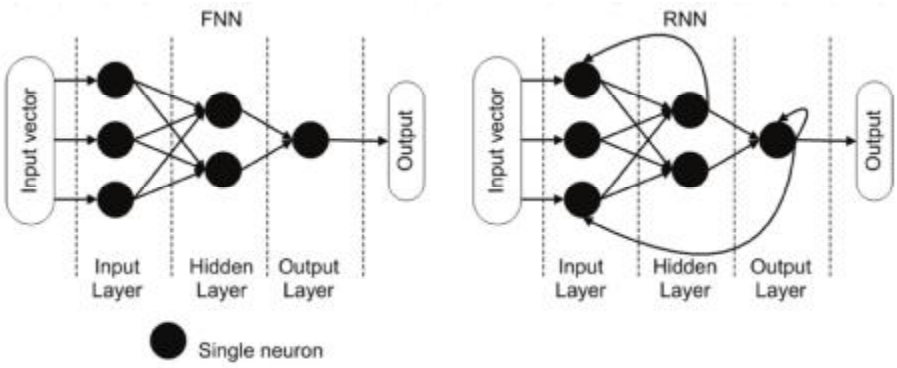
OPTICS : Ordering Points to Identify the Clustering Structure
PDF : Probability Density Function
PDMD : Probability Density of the Multivariate Distribution
PMI : Partial Mutual Information
PNN : Probabilistic Neural Networks
RBF : Radial Basis Function
RBFNN : Radial Basis Function Neural Network
RNN : Recurrent Neural Networks
SGRNN : Standard Generalized Regression Neural Network
SOM : Self-Organizing Maps
STING : Statistical Information Grid-based method
ST : Steam Turbines
SVM : Support Vector Machines

1. INTRODUCTION

1.1. General Overview to the Artificial Neural Network

Artificial neural networks are inspired by biological neural networks. They are employed to solve variety of problems in pattern classification, clustering, function approximation, optimization and prediction. Traditional rule-based approaches have been proposed for solving such problems. However, traditional approaches are only suitable for well-constrained environments, none of them is flexible enough to perform well outside its domain. ANNs surpass the limitations of rule-based approaches and provide exciting alternatives, and many applications can benefit from using them such as; computer vision, speech recognition, diagnosis systems, financial forecasting, weather forecasting etc. (Jain et al., 1996; Ozyildirim and Avci, 2013; Omondi and Rajapakse, 2006; Oral et al., 2012; Specht, 1991).

ANNs can be grouped into two categories in accordance with their structure or topology; Feed-Forward Networks (FFN), in which graphs have no loops, and Recurrent (involving feedback) Neural Networks (RNN), in which loops occur because of feedback connections. Generally, FFNs are static and memory-less in the sense that their response to an input is independent of the previous network state. On the other hand, RNNs are dynamic systems and can use their internal memory to process arbitrary sequences of inputs. Two network structures are illustrated in Figure 1.1.a-b. Figure 1.1.a represents simple feed-forward topology where information moves from inputs to outputs and Figure1.1.b. represents simple recurrent topology where information is no longer transmitted only in one direction but it is also transmitted backwards too (Jain et al., 1996; Ozyildirim and Avci, 2013; Omondi and Rajapakse, 2006; Oral et al., 2012; Specht, 1991; Krenker et al., 2011).



a) b)

Figure 1.1. Feed-forward (FNN) and recurrent (RNN) structure of an ANN

Different network structures require appropriate learning algorithms. There are three major learning models, each one is corresponding to a certain abstract learning task. These are supervised, unsupervised and reinforcement learning (Krenker et al., 2011). Supervised learning model tunes weights of an artificial neural network by using training data to allow the network to produce answers as close as possible to the training data. MLP, SVM, kNN, Gaussian Mixture Model (GMM), decision tree, Radial Basis Function Neural Network (RBFNN) are some of classification approaches (Ozyildirim and Avci, 2013). Reinforcement learning is a variant of supervised learning that tunes parameters of an artificial neural network, where training data are not usually given but generated by interactions with the environment. Reinforcement learning is principally well-matched to problems which include a long-term versus short-term reward trade-off, such as robot control and telecommunications. On the contrary, unsupervised learning does not require a correct answer associated with each input datum in the training dataset. It sets parameters of an ANN depending on given data and a cost function which is to be minimized. The most common form of unsupervised learning is clustering where data are categorized in different clusters depending on their similarity. K-means and Hierarchical clustering are the most popular clustering algorithms.

1.2. Choosing the Right Learning Model

Choosing the right learning model depends on the type of the application and data representation of a given problem. Learning model must address three fundamental and practical issues:

- Capacity
- Computational complexity
- Data complexity

Capacity relates to how many patterns can be stored, and decision boundaries a network can form. Computational complexity concerns with how long it would take for a learning model to estimate a solution from training patterns. Most of the learning models have high computational complexity. Data complexity determines the number of training neurons needed to train the network. These issues are very critical topics and designing an efficient algorithm for neural network is one of the most active research area.

According to previous works, Radial Basis Function (RBF) based approaches like Probabilistic Neural Networks (PNN) and GRNN have proved to be more advantageous than MLP and other statistical methods in terms of prediction accuracy and the modelling ease.

In the next part, one of the most frequently used neural network types; radial basis function neural network and its variants; are discussed.

1.3. Radial Basis Function (RBF) & General Regression Neural Network (GRNN)

GRNN and PNN are RBF based neural networks introduced by Donald F. Specht (Specht, 1991). Although, they are based on the same structure, the GRNN algorithm is used for function approximation, while the PNN algorithm is used for classification. The main advantage of the GRNN and PNN models with respect to

the MLP model is that, unlike MLP which needs more iterations during training step, GRNN and PNN do not require an iterative training process. They need only one pass learning. However, for many problems, the RBF network often contains larger number of hidden units (equal to number of training data). The number of hidden units is a measure of sample complexity and it directly affects the performances of both PNN and GRNN. Thus, after the training operation is completed and new test datum is applied, the response time of the RBF network becomes longer than that of MLP (Jain et al., 1996). In order to overcome this weakness, there are many works proposed for sample complexity and topology of RBF networks (Zhao et al., 2007; Husain et al., 2004; Zheng et al., 2008; Yuen et al., 2004; Mu and Nandi, 2007; Chen et al., 2002).

1.4. Weaknesses of the GRNN

Donald F. Specht introduced GRNN as an alternative to the well-known back propagation training algorithms for prediction problems (Specht and Shapiro, 1991). Due to the simplicity of network structure and ease of implementation, it has been widely utilized to solve variety of function approximation, optimization and prediction problems. However, in SGRNN, the number of pattern units is equal to the number of training data. When the training samples become enormous, the network also becomes huge. This weakens the network's efficiency and restricts its abilities. Hence, in order to simplify the GRNN's structure, network should be designed with some of the representative samples rather than whole dataset. This leads to an important issue in network structure design step: determining the significant training data.

Usually, while a portion of the training data is informative, some may be noisy or has no significant relationship with the test datum being modelled. Thus, without using any pre-processing algorithm for related input determination, some shortcomings may emerge, including (Back and Trappenberg, 1999; Maier and Dandy, 1997; Zheng and Billings, 1996):

- Increased computational complexity.
- Increased memory requirements.
- Increased processing time.
- Reduced prediction accuracy.

Therefore, a reliable pre-processing algorithm is required to overcome these issues and increase efficiency of the SGRNN. In this study, three different hybrid GRNN models are proposed. The use of clustering algorithms as a pre-processor to calculate similarity between input elements and compress training samples or find the best representative dataset for training is one of the research element of the proposed thesis. This strategy may largely simplify the sample complexity and may significantly improve the network's efficiency.

1.5. The Aims and Objectives

It is the aim of this study to propose an effective pre-processing method for selecting significant training data and reducing the sample complexity of the GRNN. This involves the following objectives;

1. To identify the most commonly used pre-processing algorithms for GRNN and to identify how data reduction operation is achieved
2. To classify traditional data reduction methods
3. To carry out the data reduction experiment with the traditional methods
    a. To identify most popular algorithm of each class
    b. To identify shortcomings of the traditional pre-processing algorithms
    c. To derive new methods to propose better pre-processing algorithms

   d. To validate the proposed hybrid GRNN algorithms performances by comparing its prediction results with the SGRNN and traditional hybrid GRNNs
4. To reduce the number of required training data for GRNN through proposed pre-processing algorithms
5. To propose new and efficient GRNN algorithm/s by combining the proposed new pre-processing algorithm/s

Algorithms used for reducing complexity of the GRNN has still weaknesses. For instance, method that clusters the training data, and selects only the training data belonging to the most similar cluster to the test data, has inner outlier problem. Therefore, proposal of efficient data reduction (pre-processing) algorithm is essential for practical use of the SGRNN.

1.6. Contribution to Existing Knowledge

Studies covered in this thesis aim reducing the complexity of GRNN and increasing its efficiency. For this aim, three methods are proposed. Contributions of these methods are summarised below, respectively.

The use of k-means clustering algorithm as a pre-processor has been studied in the literature. However, the sample data that exist between cluster borders (inner cluster outliers) produce large residuals resulting negative influence on prediction performance. With this study, the negative side effects of employing a clustering algorithm as a pre-processor on prediction performance has been reduced significantly and proposed approach achieved almost similar prediction performances to standard GRNN's by using ~95% less training samples.

As an alternative to the clustering GRNN, kNN algorithm which is the most popular algorithm was examined in the second method. kNN algorithm has problem of getting stuck on data heap. In this study, it was solved for 3-dimensional problems by using angle based nearest neighbourhood. Obtained

results indicate that the proposed method provides 99.33% reduction in pattern layer size and accuracy is also increased maximally to 68.58%.

In the third method, in addition to these two algorithms, SOM based hybrid GRNN algorithm which provides further improvement on the results is presented. According to the simulation results, the proposed model significantly simplifies the GRNN's structure without any performance loss and provides superior performance to traditional clustering GRNN approaches.

1.7. Main Findings of the Research

The literature review and evaluation of the traditional clustering algorithms produced the following findings:

1. Increase in the training data count causes decrease in the efficiency of the GRNN. Hence, an efficient data reduction algorithm is needed.
2. The lack of a reliable algorithm for selecting the related training data raises doubt about the optimality of the obtained training data.
3. The sample data that lay in between cluster borders (inner cluster outliers), produce large residuals resulting negative influence on prediction performance.
4. If the test datum is located around of a data heap, the prediction result may be average value of this data heap.

1.8. Outline of the Thesis

This thesis is organized as follows:

In Section 2, the related studies in the literature are reviewed and general information is given about them. Algorithms in the literature are classified according to their properties and the most popular algorithms belonging to each class are presented. Their advantages and disadvantages are discussed.

Section 3, covers materials for this thesis such as algorithms and test functions used at each stage of the work. Firstly, the workflow of the GRNN which is main topic of this thesis, is explained. Then, algorithms that are frequently used with GRNN to simplify its structure are explained. In the last part of this section, the synthetic datasets (benchmark functions) and the actual datasets used for evaluating the proposed studies, and the evaluation functions used for measuring errors are explained.

Section 4 is dedicated to discussions for determining appropriate clustering algorithm that can be used with GRNN.

In Section 5, the proposed three different methods to simplify the structure of GRNN are presented. Encountered problems, and proposed solutions are explained in this section.

The last part of the thesis covers comparison of the proposed methods with classical methods, obtained prediction results, and the discussions on test results, and future works.

2. PREVIOUS WORKS

The use of pre-processing algorithms provides obvious advantages in selection of appropriate training data for GRNN. Several methods based on different clustering and local learning algorithms have been proposed to determine appropriate training data. Training data determination methodologies vary depending on the utilized algorithms. These algorithms can be divided into two main categories.

In the first category of algorithms, training data are separated into a set of independent groups, clusters, which are not necessarily related to test data. Thus, while similar data are placed into the same cluster, dissimilar data are placed into the distinct clusters. This operation helps users to discover and understand the natural structure of the data and extract the essence of large dataset (Harkanth and Phulpagar, 2013; Kotsiantis and Pintelas, 2004; Rama et al., 2010; Kolesnikov et al. 2015; Berkhin, 2002). When a new test datum is applied, the related data cluster is determined and used as training dataset. Only the relationships between test datum and cluster centres are calculated. Proposed approaches which have used the clustering algorithms as a pre-processor can be shown as 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 training data, the relationships between test datum and all training data are determined when test datum is applied. In such algorithms, operations are being performed using directly test data, thus, more reliable training data are obtained. Relations between test datum and each training datum are calculated one by one. If the dependence between test datum and train datum is found to be significant, the train datum is retained in the final training data subset, otherwise it is eliminated. Since all these operations are computed after test datum is received, the number of transactions during the prediction process increases greatly.

Specht (1991) presented the clustering version of GRNN to reduce the number of neurons in the pattern layer. Later, he mentioned about the prediction speed problem of the GRNN and introduced a new algorithm to overcome this drawback (Specht, 2006). The algorithm consists of a combination of three methods, which provides advantages especially for multidimensional and noisy datasets (GRNN with double clustering). This combination includes clustering, kernel regression with adaptive parameters, a second level of clustering, and the construction of a binary decision tree. Algorithm was evaluated on only one problem with its two different dimensions, 5 and 10. While the result in 10 dimensions is clearly superior to other algorithms, in 5 dimensions, differences between the results are decreasing. And there is not any other dataset or result to comment on this algorithm's performance efficiently.

In following studies, various clustering techniques have been suggested to compress all training data into a few clusters. Zheng et al. (2008) used k-means clustering algorithm for separating training data into groups. In their study, improvement on the prediction accuracy of the GRNN was provided by grouping data and assigning different smoothing parameter to each group. Estimation of Distribution Algorithm (EDA) was utilized to determine the optimum smoothing parameter. While using clustering methods as pre-processor, another critical parameter affecting the GRNN performance, clusters number, was emerged. Clusters number was selected by increasing k number with 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 the similar data into clusters and cluster centres represent all the samples grouped in that cluster. During the clustering operation, firstly, similarity values between the training data and cluster centre are calculated according to the given similarity metric. Secondly, similarity is compared with certain threshold value. If it is under the threshold value, the data are placed into the related cluster, otherwise, it is determined as the centre of newly created cluster. For this reason, threshold is the critical parameter

in this algorithm. While the algorithm given in (Zhao et al, 2007) simplified the GRNN structure, a small amount of loss was occurred in accuracy.

Husain et al (2004) suggested another GRNN algorithm based on Fuzzy C-means clustering. They calculate similarity index that indicates the degree of similarity in which data will be clustered, through a simple similarity measurement. The cluster centre of related data is determined by iterative process and calculated cluster centres are used to define the parameters of the GRNN.

A hybrid model combining the Fuzzy ART (FA) and GRNN for noisy data was proposed by Yuen et al (2004). FA is a powerful unsupervised classifier based on the 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. Thus, it was assumed that, noises in the datasets exist in the borders, were eliminated. Performance of the algorithm was evaluated with two benchmark datasets. Results exhibited that the algorithm was statically superior to several learning systems for two datasets. However, the algorithm was not tested on any real dataset and the results were concluded from only two synthetic datasets. That causes the reliability of the algorithm to be questioned.

In Seng et al. (2002), Adaptive GRNN which has some different features compared to the basic GRNN, such as flexible pattern nodes add-in and delete-off mechanism, dynamic initial sigma assignment using non-statistical method, and sigma tuning was presented for dynamic plant modelling. In Adaptive GRNN, the network can be formed dynamically from a null network, i.e. with zero pattern nodes. Number of neurons in pattern layer increases during training process. When a new training datum is applied to the network, a new pattern node may be created to store this new knowledge. A new neuron is formed only if it is necessary, and the maximum number of neuron is also restricted. Necessity is determined by using a threshold level. Moreover, less valuable pattern nodes can be discarded.

Kokkinos and Margarit (2015) introduced Local Learning Model of the GRNN algorithm. Method uses only k-nearest training data to the test datum and reduces the complexity from O(N) to O(k) which makes GRNN more suitable for large scale applications. Firstly, network is constructed with a minimum number of nearest neighbours. Then, new neurons are progressively added until it reaches the last nearest neighbour. At each step, a new network structure is created and prediction value is obtained by averaging the results of each network.

In another study, Bowden et al. (2005a; 2005b) pointed out that use of all training dataset has several disadvantages such as poor performance, more difficult learning with irrelevant data and increases in computational complexity. Then, they proposed new data selection models in two-part series articles. In the first article, two input selection algorithms were introduced. The first one is Partial Mutual Information (PMI) algorithm to define which training data have a significant relationship with the output data. Second one is based on the SOM, combined with genetic algorithm (SOM-GAGRNN) which uses SOM to cluster training data and selects one representative data from each cluster with minimum Euclidean distance to its cluster centre. Then, these training data are utilized by GAGRNN. In the first article, algorithms were tested on several synthetic datasets whose dependence attributes were known in advance. In the second paper, the proposed algorithms were applied to real dataset for forecasting salinity 14-days in advance in the River Murray at Murray Bridge, South Australia. Even though both methods were adequate from the point of view of getting valuable information about the system, they had some limitations. One limitation that the PMI algorithm may encounter, is determining a suitable bandwidth when the data are highly non-Gaussian. In second algorithm, SOM-GAGRNN algorithm does not guarantee that the actual training data will be selected, because training-data highly connected to the actual training data can be selected if they are nearest to the cluster centre. Moreover, GA requires additional parameters such as population size, number of iteration and suited objection function.

Recent studies indicate that clustering versions of GRNN may provide several advantages such as speed and reduced cost. Instead of employing all training samples as kernels, clustering process might be used to compress training data into fewer kernels. Hence, the redundant samples might be removed from training set and calculations might be reduced. However, none of the proposed methods provided the desired level of improvement for the GRNN, and there is still a lack of literature in this field. When we reviewed the literature, it is seen that in many cases, the lack of a reliable algorithm for selection of the related training data raises doubt about the optimality of the obtained training data.

In this thesis, one input determination method belonging to the first category that groups training data into a set of independent clusters, and one input determination method belonging to the second category that identifies training data relative to test datum are implemented to compare their results with that of proposed methods:

1. k-means algorithm, one of the most popular clustering algorithm employed for pre-processing is chosen as an example of first category algorithms (Hartigan and Wong, 1979). Detailed information about the k-means can be accessed from section 3.4.
2. The algorithm chosen from the second category for comparison is 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 datum with defined distance functions (Cover and Hart, 1967). Detailed information about the kNN can be accessed from section 3.5.

3. MATERIALS

3.1. Introduction

This section contains detailed information about GRNN and a brief introduction to clustering algorithms examined in this study. In addition to these, employed pre-processing algorithms such as; k-means, kNN and SOM and utilized mathematical method (probability density function) are explained in detail. Later on in this section, an introduction to a set of famous benchmark functions (Beale, Rastrigin, Schwefel, etc.) as well as real world data sets (Airfoil Self-Noise, Air Quality, Combined Cycle Power Plant, etc.) used to evaluate prediction performance of the proposed algorithms are included. Finally, the metrics that are used to quantify and compare the quality of the prediction results is given.

3.2. Generalized Regression Neural Network (GRNN)

GRNN is based on Parzen window estimation and often used for function approximation (Specht, 1991). GRNN estimates joint density function of parameters as in (3.1.) where $\vec{x}_j$ and y_j are j^{th} samples from x and y; N is the number of samples and m is the number of dimension of $\vec{x}_{inp}$.

$$pdf(\vec{x}_{inp}, y_{inp}) = \frac{1}{(2\pi)^{m+1/2}\sigma^{m+1}}\frac{1}{N}\sum_{j=1}^{N} e^{\left(-\frac{(\vec{x}_{inp}-\vec{x}_j)^T(\vec{x}_{inp}-\vec{x}_j)}{2\sigma^2}\right)} e^{\left(-\frac{(y_{inp}-y_j)^2}{2\sigma^2}\right)} \quad (3.1.)$$

After the joint probability density function is estimated, conditional mean of y $(\hat{y})$ can be calculated as in (3.2.) for x (Zheng et al., 2008).

$$\hat{y}(\vec{x}_{inp}) = \frac{\sum_{j=1}^{N} e^{\left(-\frac{(\vec{x}_{inp}-\vec{x}_j)^T(\vec{x}_{inp}-\vec{x}_j)}{2\sigma^2}\right)} \int_{-\infty}^{+\infty} y e^{\left(-\frac{(y-y_j)^2}{2\sigma^2}\right)}}{\sum_{j=1}^{N} e^{\left(-\frac{(\vec{x}_{inp}-\vec{x}_j)^T(\vec{x}_{inp}-\vec{x}_j)}{2\sigma^2}\right)} \int_{-\infty}^{+\infty} e^{\left(-\frac{(y-y_j)^2}{2\sigma^2}\right)}} \quad (3.2.)$$

Since $(\vec{x}_{inp} - \vec{x}_j)^T(\vec{x}_{inp} - \vec{x}_j)$ is squared Euclidean distance$(dist_j^2)$, (3.2.) can be rewritten as in (3.3.) after integrations are calculated.

$$\hat{y}(\vec{x}_{inp}) = \frac{\sum_{j=1}^{N} y_j \, e^{\left(\frac{-dist_j^2}{2\sigma^2}\right)}}{\sum_{j=1}^{N} e^{\left(\frac{-dist_j^2}{2\sigma^2}\right)}} \quad (3.3.)$$

Let $\vec{x}_{inp}$ be input vector, $\vec{x}_j$ be j^{th} sample stored in pattern layer and $dist_j$ denote squared Euclidean distance between $\vec{x}_{inp}$ and $\vec{x}_j$.$dist_j$ can be calculated as in (3.4.).

$$dist_j = (\vec{x}_{inp} - \vec{x}_j)^T (\vec{x}_{inp} - \vec{x}_j) \quad (3.4.)$$

Then, according to the (3.3.), prediction for the input $\vec{x}_{inp}$ can be calculated from (3.5.) where $Y(\vec{x}_{inp})$ is predicted value, N is the number of samples placed in pattern layer, σ denotes smoothing parameter and y_j is the output value for the j^{th} sample.

$$Y(\vec{x}_{inp}) = \frac{\sum_{j=1}^{N} y_j e^{\left[\frac{-dist_j}{2\sigma^2}\right]}}{\sum_{j=1}^{N} e^{\left[\frac{-dist_j}{2\sigma^2}\right]}} \quad (3.5.)$$

GRNN is a four-layer feed forward neural network, including input layer, pattern layer, summation layer and output layer. Each layer contains different number of neurons by which different layers can connect with each other. The general structure of GRNN model is given in Figure 3.1.

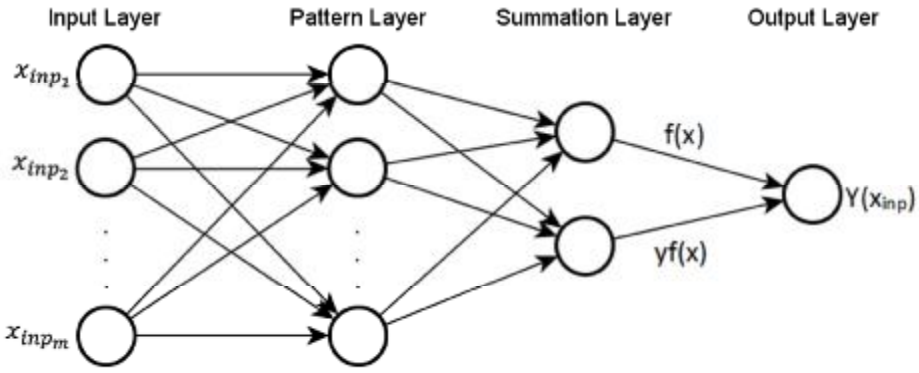


Figure 3.1. Schematic diagram of GRNN

Each layer of the GRNN computes one of the steps of calculations given in Equation (3.5.). The number of neurons in the first (or input) layer is equal to the dimension of input variable $\vec{x}_{inp}$. The second layer is the pattern layer where each training datum is stored in one neuron $\vec{x}_j$. In this layer, the Euclidean distances $dist_j$ between the incoming test datum $\vec{x}_{inp}$ and training data $\vec{x}_j$ are computed and these neurons output the weight values by using Gaussian unit $e^{\left[\frac{-dist_j}{2\sigma^2}\right]}$. Third layer has two summation neurons; nominator and denominator. While numerator, sums the products of weights with target value of related training datum; denominator computes the sum of all weights. Finally, the fourth layer divides the numerator by denominator and produces the estimated output (Specht, 1991).

3.3. An Overview to Clustering Algorithms

Cluster analysis or clustering is a separation of data into groups (clusters) of similar objects. This helps in the abstraction process of large dataset and summarizing its main characteristics (Omondi and Rajapakse, 2006; Oral et al., 2012). It is used in a wide variety of fields, including bioinformatics, image processing, machine learning, pattern recognition, information retrieval, etc. (Jain et al., 1999; Fahad et al., 2014; Berkhin, 2002).

Clustering algorithms differ depending on the requirements of an application. The methodologies of these algorithms vary according to number of sample to be considered, accuracy of the result and size of each instance (Harkanth and Phulpagar, 2013). Various methods and algorithms have been developed as a result of these factors. However, there is no clustering algorithm that can be generally used to solve all problems (Kotsiantis and Pintelas, 2004). In this section, a brief overview of these clustering techniques, their advantages and disadvantages are explained.

Generally, clustering approaches can be categorized into hierarchical clustering, partitioning clustering, density-based clustering, grid-based clustering, and model-based clustering. Hierarchical clustering algorithms organize data into a hierarchical structure in proportion with the similarity matrix. These methods unite data to form clusters according to their distance. There are two basic techniques under this category; agglomerative (bottom-up) and divisive (top-down). In agglomerative approach, firstly each datum is used as one cluster, later, data are combined depending on the distance. In divisive approach, firstly whole dataset is considered as a cluster. Then that cluster is separated into smaller clusters depending on their similarity. These methods are not very robust to outliers which need extra clusters or even cause other clusters to merge. Also, the complexity is $O(N^3)$ which makes them too slow for large datasets. Some of the hierarchical clustering algorithms are: Clustering Using REpresentatives(CURE), Balanced

Iterative Reducing and Clustering using Hierarchies (BIRCH) and CHAMELEON (Rama et al., 2010).

Partitioning methods are divided into two major subcategories, the centroid-based and the medoid-based algorithms. K-Means, Fuzzy K-Means, Fuzzy C-Means (FCM), K-Modes, Farthest First Traversal k-centre (FFT), K-Medoids (PAM), CLARA, Fuzzy K-Modes, CLARANS are some proposed methods in this group (Rama et al., 2010). In centroid-based clustering, clusters are represented by using the gravity centre of the instances. In medoid algorithms, the means of the samples closest to the gravity centre are used to represent related clusters.

One of the most popular partitioning algorithm is the k- means (Rama et al., 2010). It is efficient in large datasets. Firstly, the algorithm requires the cluster number -k- to be determined in advance, which is considered as one of the biggest difficulty of this algorithm. Secondly, each datum is assigned to its nearest cluster centre. Then, centroids of newly assembled clusters are recomputed. This step is repeated until a stopping criterion is achieved (for example, no reassignments). Another difficulty is finding a general distance or dissimilarity function that works well with all types of data (Jain et al., 1999).

In standard partition algorithms, each datum belongs to only one cluster. Conversely, FCM is a “soft” clustering method, a datum may belong to more than one cluster with membership value ranging from 0 to 1 (Sato et al., 1997). Higher membership values indicate higher likelihood between data and cluster. FCM is based on k-means algorithm. It iteratively searches the cluster centres and updates the memberships of data.

In density-based clustering, clusters are defined depending on density of the regions. The cluster grows in any direction that density leads to and each cluster has to contain at least minimum number of data in a neighbourhood of a given radius. This approach is used in the DBSCAN (Density Based Spatial Clustering of Applications with Noise), OPTICS (Ordering Points to Identify the

Clustering Structure), DBCLASD (Distribution Based Clustering of LArge Spatial Databases) and DENCLUE (DENsity CLUstEring) algorithms (Fahad et al., 2014). These algorithms have ability to discover clusters of irregular shapes and are less sensitive to outliers. The most well-known algorithm is DBSCAN. It is based on connecting data within certain radius. To find a cluster, it begins with a random selected data and retrieves all data in the dataset with respect to radius value and minimum number of instance. The key difficulty of DBSCAN is that it expects some kind of density drop to detect cluster borders.

In grid based algorithms, initially clustering space is divided into finite number of cells. Statistical features are collected for all the data objects located in each cell. Then, clustering is performed on the grid. Cells that contain more than certain number of data objects are handled as dense and the dense cells are connected to form the clusters. The performances of these methods depend on the size of the grid, which is generally less than the size of the dataset. STING (Statistical Information Grid-based) and Wave-Cluster are the most common algorithms of this category. STING firstly, separate the clustering area into several levels of rectangular cells in order to form a hierarchical tree. The high-level cells are composed by the lower level cells. Thus, the hierarchical structure of the grid cell is generated and used to represent clustering information at different levels. It generates good clustering results in a short running time but, its performance relies on the granularity of the lowest level of the hierarchical structure and resulting clusters are never diagonal (Kotsiantis and Pintelas, 2004).

Model based approaches begin with random parameter set and they adjust them over the iterations to find maximum likelihood estimator. Also, it leads to a way of automatically deciding cluster number based on standard statistics. There are two major approaches under this category; statistical approaches and neural network approaches. Statistical approaches determine the clusters or concepts by using probability measures. The neural network approach has different features. It uses a set of connected input/output units, where each connection has a weight

associated with it. Then, they learn by incrementally adjusting their interconnection weights (Hruschka et al, 2009). Self-organizing feature maps (SOM) is the typical example of this category. The SOM net can be thought of as two-layer neural network and works in two modes: training and mapping. It projects high dimensional data onto a two-dimensional map. Map units are called as nodes or neurons. Each node has its own weight vector which has the same dimension as the input data vector. Nodes are placed with certain intervals on a hexagonal or rectangular grid. Then the input data is placed onto the map by finding the closest weight vector to the input vector. The nodes of the SOM are themselves cluster centres and these nodes are utilized to form bigger clusters. It is powerful approach to detect outliers and deal with missing data values (Rama et al, 2010; Kotsiantisand and Pintelas, 2004).

In this study, in addition to the information obtained from the literature, some preliminary studies have been carried out to select most suitable clustering algorithm to be used in this study.

3.4. k-Means Clustering

One of the most popular clustering algorithm used in the literature is the k-means (Rama et al., 2010). In k-means, before starting to the clustering process number of clusters k must be determined. It is considered as one of the most fundamental shortcoming of this algorithm (Kolesnikov et al., 2015). The k-means algorithm is briefly outlined below. Further information can be found in (Hartigan and Wong, 1979).

Algorithm starts with selecting k data randomly from the dataset as cluster centres. Then, the following steps are executed for all iterations. In each iteration, the distances between each datum and cluster centres are measured and each datum is assigned to its nearest cluster centre. Then, centroids of newly assembled clusters are recomputed as in (3.6.).

$$C_i = \left(\frac{1}{N_i}\right)\sum_{j=1}^{N_i} x_i \tag{3.6.}$$

This step is repeated until a stopping criterion is achieved (for example, no reassignments). The aim of the algorithm is to minimize the total distance (Jain et al., 1999). The objective function, which minimizes the within-cluster sum of squares is given in (3.7.).

$$d = \sum_{i=1}^{k}\sum_{j=1}^{N_i}\left\|x_j - c_i\right\|^2 \tag{3.7.}$$

where, $\left\|x_j - c_i\right\|^2$ is distance measure between a data point x_j and the cluster centre c_i. k is the cluster count and the N_i is the number of data in i^{th} cluster.

An example view of k-means clustering algorithm's result over a dataset consisting of 1000 sample is given in Figure 3.2. Cluster count is chosen as 5 and X labels illustrate the centre of clusters.

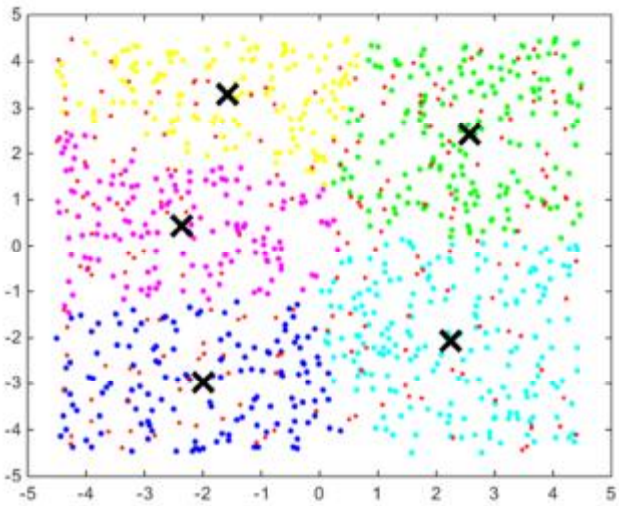


Figure 3.2. An example view of k-means clustering algorithm with k=5

3.5. k-Nearest Neighbour (kNN)

kNN is one of the most popular algorithm in the literature used for regression and classification. The main goal of the algorithm is to find the k nearest neighbours to the related test datum with defined distance functions and reduce the operation cost from O(N) to O(k) (Cover and Hart, 1967).

Algorithm has only one critical parameter k, to be determined. It points the number of the nearest data. The selection of the best k depends on the feature of the data (Everitt et al., 2011). Another issue is how the distances among the data can be measured. Commonly, Euclidean distance function is employed for continuous variables. For discrete variables, such as for text classification, Hamming distance is preferred (Hamming, 1980).

As a result, when a new test datum is applied, firstly, the distance between test datum and each of the training data is measured. Secondly, the k training data having minimum distance values are labelled as k nearest data. An example outputs of kNN algorithm for ten nearest neighbours on sample dataset is illustrated in Figure 3.3. 10 Nearest Neighbours While the blue dots point out the train data group, the greens stars point out the k nearest data to the test datum shown with red circle.

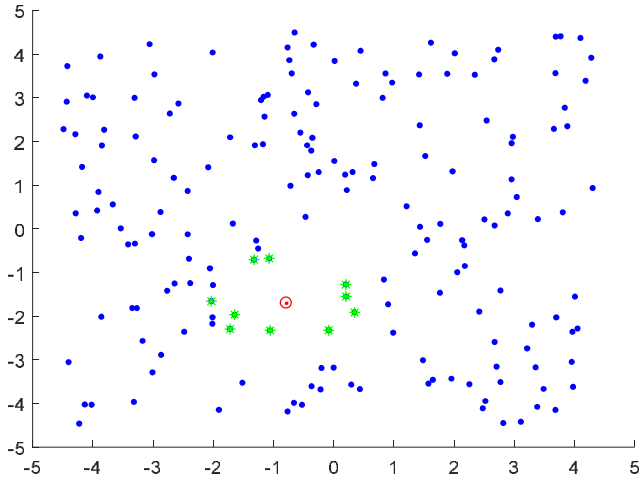


Figure 3.3. 10 Nearest Neighbours

3.6. Self-Organizing Maps (SOM)

SOM is a clustering algorithm which has mapping feature. SOM net can be thought of as two layered neural network working in two modes: training and mapping. It represents high dimensional data with a two-dimensional map. General structure of the algorithm is given in Figure 3.4.

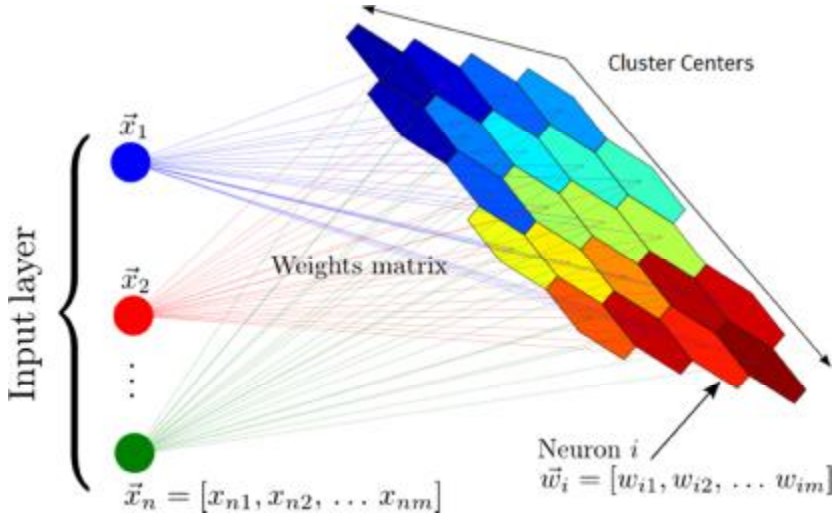


Figure 3.4. General structure of SOM (Carrasco and Brunner, 2014)

The map consists of units called nodes or neurons. These nodes have their own weight vectors which have the same dimension as the input vector. Nodes are located on a hexagonal or rectangular grid with certain intervals. During the training of these nodes, competitive learning is employed (Jain et al., 1999). First, random values are assigned to all weight vectors. Secondly, Euclidean distance between the input datum $\vec{x}_j$ and each weight vector $\vec{w}_i$ is measured as in (3.8.) where m points the dimension of the data and $distsom_i(\vec{x}_{inp})$ is squared Euclidean distance between input datum and i^{th} weight vector.

$$distsom_i(\vec{x}_j) = \left\|\vec{x}_j - \vec{w}_i\right\| \quad (3.8.)$$

The node having the smallest distance value to the input data is determined and called as the best matching unit (BMU) or winner unit. Finally, the weight values of the BMU and the neighbouring nodes are adjusted towards the input data.

Before adjusting the weight values, neighbourhoods to the BMU are calculated. Assume that $BMU(\vec{x}_j)$ represents the BMU, i is the i^{th} node, $S_{\left(i,BMU(\vec{x}_j)\right)}$ denotes the lateral distance between i^{th} node and $BMU\left(\vec{x}_j\right)$, and $\sigma(l)$ is a coefficient used for determining the number of affecting neighbour, neighbourhood $H_{\left(i,BMU(\vec{x}_j)\right)}(l)$ is calculated as in (3.9.) for l^{th} iteration.

$$H_{\left(i,BMU(\vec{x}_j)\right)}(l) = exp\left(\frac{-S^2_{\left(i,BMU(\vec{x}_j)\right)}}{2\sigma(l)^2}\right) \tag{3.9.}$$

The magnitude of the adjustment changes depending on the time and distance from the BMU. A popular time dependence is an exponential function and given in (3.10.) where σ_0 points the magnitude at the initial time, the Greek letter lambda, λ, points a time constant and l is the current time-step (iteration number).

$$\sigma(l) = \sigma_0 \exp\left(\frac{-l}{\lambda}\right) \tag{3.10.}$$

Depending on the change of the magnitude over the time, the neighbourhood will shrink to the size of just one node, BMU. Weight values are updated as in (3.11.) for l^{th} iteration where $lr(l)$ denotes the learning rate at the l^{th} iteration.

$$\vec{w}_i(l) = \vec{w}_i(l-1) + lr(l) * H_{\left(i,BMU(\vec{x}_j)\right)}(l) * (\vec{x}_j - \vec{w}_i) \tag{3.11.}$$

Learning rate is also updated during iterations as in (3.12.) where l denotes l^{th} iteration, λ_{lr} is a time constant and lr_0 is the initial learning rate.

$$lr(l) = lr_0 \exp\left(\frac{-l}{\lambda_{lr}}\right) \quad (3.12.)$$

Each node in the SOM represents a cluster centre and these nodes are utilized to form bigger clusters. It is a useful algorithm for detecting outliers and dealing with the missing data values. Same as in the clustering and kNN algorithms, SOM network has one critical parameter be tuned to achieve the best prediction performance; its size (cluster count) (Kotsiantis and Pintelas, 2004; Rama et al., 2010).

3.7. Probability Density Function

Probability Density Function (PDF) is a function which gives the likelihood of observing the related datum in the data space. In other words, PDF is used to specify the probability that the random variable falls into a specific range of values. The results are always positive and the total integral over the entire space is equal to one.

In our thesis, all data have more than one variable. Hence, we need a formulation to calculate "probability density of the multivariate datum", which is related with Multivariate Normal Distribution (MND). The MND is generally used to define, at least approximately, any set of correlated real-valued random variables each of which clusters around a mean value. Many samples from a multivariate normal distribution with μ=0, the two marginal distributions, are illustrated in Figure 3.5.

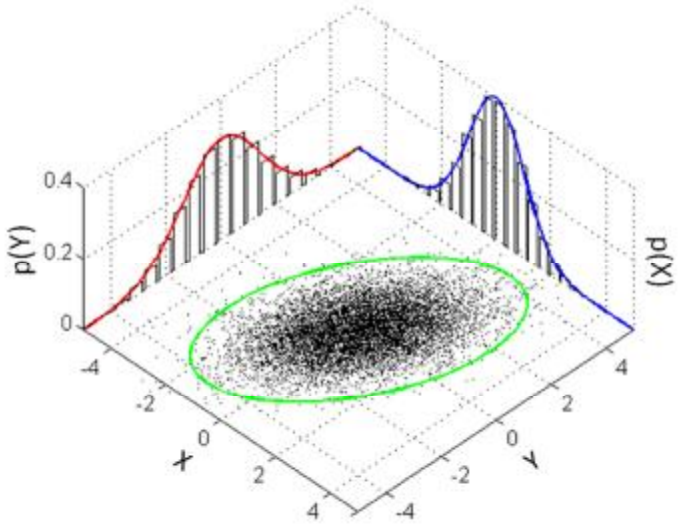


Figure 3.5. Many samples from a multivariate normal distribution with μ=0, the two marginal distributions (Wikipedia, 2017)

Probability Density of the Multivariate Distribution (PDMD) of a test datum can be calculated as in (3.13),

$$\text{PDMD}(\vec{x}_{inp}) = \frac{exp\left(-\frac{1}{2}(x-\mu)^T \sum^{-1}(x-\mu)\right)}{\sqrt{|2\pi\sum|}} \qquad (3.13.)$$

where, x is m-dimensional training data and $\vec{x}_{inp}$ is the test datum, μ is the m-dimensional mean vector, $\sum$ is the mxm covariance matrix and $|\sum|$ is the determinant of $\sum$.

3.8. Standard Test Functions

It was considered necessary to compare the input determination methods on synthetic data, before applying these methods to the real-world case study. Fifteen well known benchmark test functions were employed for analysing and comparing the performances of the proposed algorithms and the others (SGRNN, k-means-GRNN and kNN-GRNN) (Tang et al., 2008). Test functions contain three

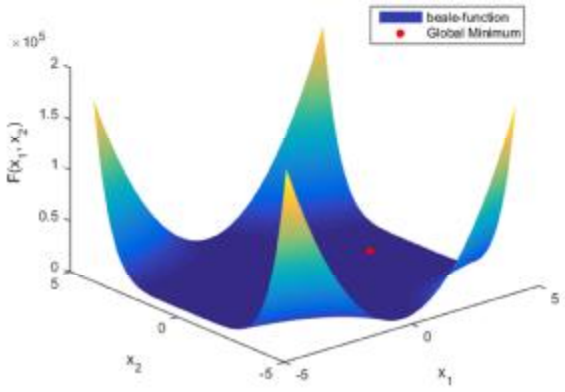
parameters (x1, x2 and y) and, have different sized search spaces. The corresponding test function, their boundaries and global minimums are listed in Table 3.1.

In the following sections, the detailed view of each test function is given. The section is divided into sub sections that include a plot of each function in the range of the problem space. Furthermore, optimum point(s) is/are also pointed out.

Table 3.1: Test Functions with Their Boundaries and Global Minimums

Function	Definition	Boundaries	Min
f_1	Beale	[-4.5, 4.5]	0
f_2	Rastrigin	[-5.12, 5.12]	0
f_3	Schwefel	[-500, 500]	-837.9658
f_4	Ackley	[-32, 32]	0
f_5	Goldstein-Price	[-2, 2]	3
f_6	Six- Hump Camel	[-5, 5]	-1.031628...
f_7	Bird	[-2*π, 2*π]	-106.764537
f_8	Leon	[-1.2, 1.2]	0
f_9	Griewangk	[-100, 100]	0
f_{10}	Giunta	[-1, 1]	0.060247
f_{11}	Booth	[-10, 10]	0
f_{12}	Bukin4	[-15, -5], [-3, +3]	0
f_{13}	Cross-In Tray	[-10, 10]	0
f_{14}	Himmelblau	[-5, 5]	0
f_{15}	Matyas	[-10, 10]	0

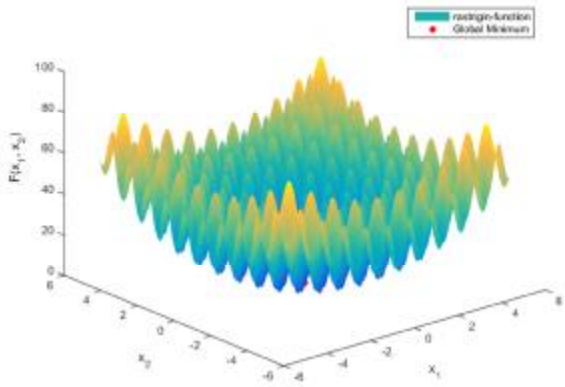
3.8.1. F_1: Beale



$$f_1 = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1{x_2}^2)^2 + (2.625 - x_1 + x_1{x_2}^3)^2$$

Test area is usually restricted to -4.5≤ x_i≤4.5, for all i = 1, 2. Its global minimum equal f(x) = 0 is obtainable at (x^*) = (3, 0.5).

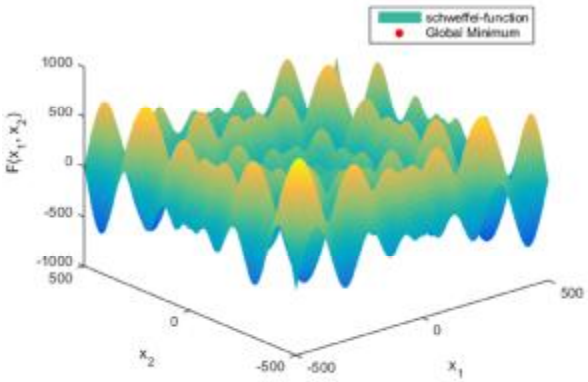
3.8.2. F_2: Rastrigin



$$f_2 = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right]$$

Test area is usually restricted to -5.12≤ x_i≤ 5.12, i = 1...., n. Its global minimum equal f(x) = 0 is obtainable for (x^*) = (0, 0).

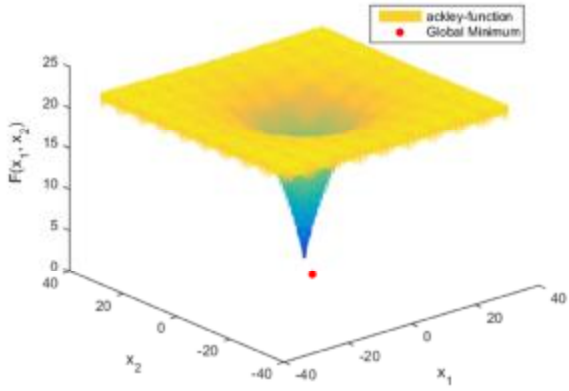
3.8.3. F_3: Schwefel



$$f_3 = 418.9829\text{n} - \sum_{i=1}^{n}(x_i sin(\sqrt{|x_i|}))$$

Test area is usually restricted to -500≤ x_i≤ 500, i = 1,..., n. Its global minimum equal f(x) = -418.9829n is obtainable for (x^*) = (420.9687, 420.9687).

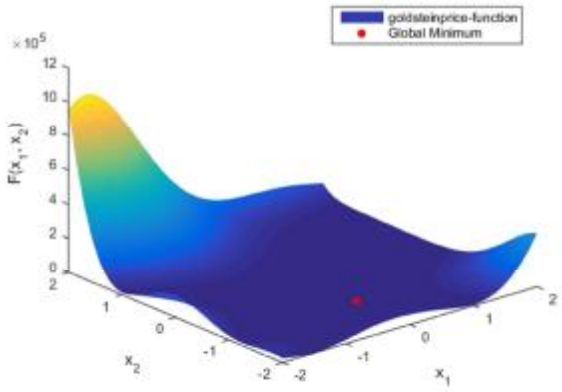
3.8.4. F_4: Ackley



$$f_4 = -20 * \exp\left(-0.2 * \sqrt{\frac{1}{n}\sum_{1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + \exp(1)$$

Test area is usually restricted to -32≤ x_i≤ *32*, i = 1,..., *n*. Its global minimum equal $f(x)$ = 0 is obtainable for (x^*) = (0, 0).

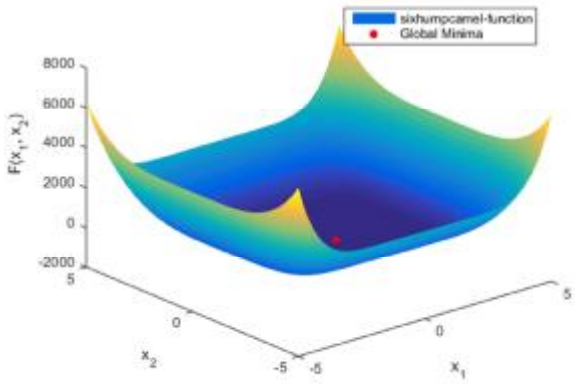
3.8.5. F_5: Goldstein-Price



$$f_5 = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3{x_1}^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] * [30 + (2x_1 + 3x_2)^2(18 - 32x_1 + 12{x_1}^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$$

Test area is usually restricted to -2≤ x_i≤ 2, i = 1, 2. Its global minimum equal $f(x)$ = 3 is obtainable for (x^*) = (0, -1).

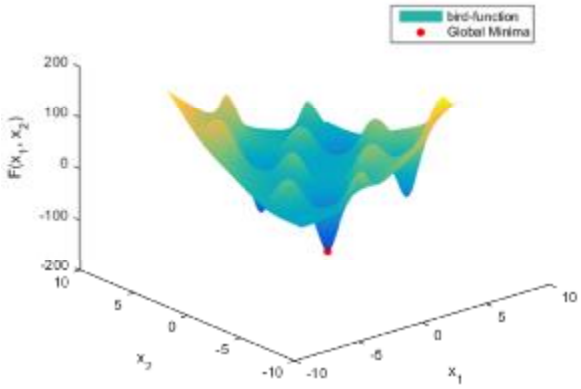
3.8.6. F_6: Six- Hump Camel



$$f_6 = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$$

Test area is restricted to -5≤ x_1≤ 5, i = 1....n. Its global minimum equal f(x) = -1.0316 is obtainable for (x^*) = (0898, -0.7126) and (-0898, 0.7126).

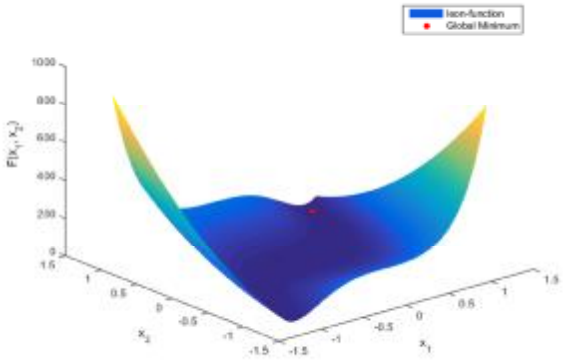
3.8.7. F_7: Bird



$$f_7 = (x_1 - x_2)^2 + e^{[1-sin(x_1)]^2} cos(x_2) + e^{[1-cos(x_2)]^2} sin(x_1)$$

Test area is restricted to $-2\pi \leq x_i \leq 2\pi$, i = 1....n. Its global minimum equal f(x) = -106.7645 is obtainable for (x^*) = (-3.13, -1.58).

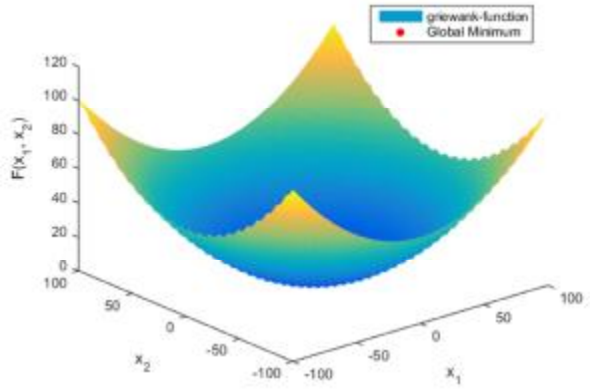
3.8.8. F_8: Leon



$$f_8 = (1 - x_1)^2 + 100(x_2 - x_1^3)^2$$

Test area is restricted to $-1.2 \leq x_i \leq 1.2$, i = 1, 2. Its global minimum equal f(x) = 0 is obtainable for $(x^*) = (1,1)$.

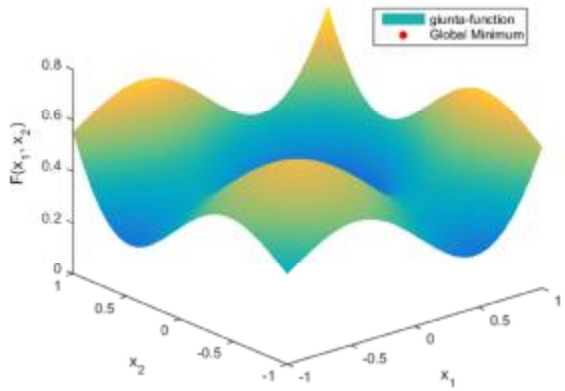
3.8.9. F_9: Griewangk



$$f_{9=}\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

Test area is usually restricted to -100≤ x_i≤ 100, i = 1,..., n. Its global minimum equal f(x) = 0 is obtainable for (x^*) = (0, 0).

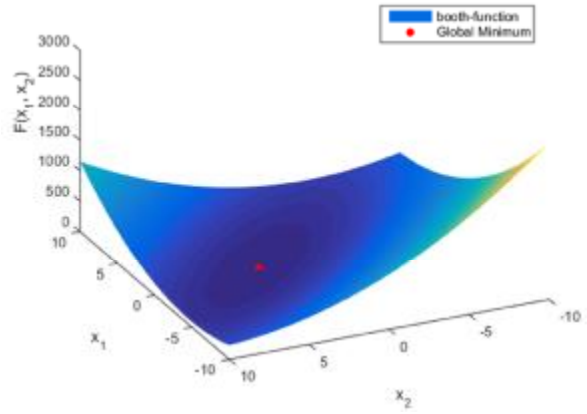
3.8.10. F_{10}: Giunta



$$f_{10} = 0.6 + \sum_{i=1}^{n}\left[sin^2\left(1-\frac{16}{15}x_i\right) - \frac{1}{50}sin\left(4-\frac{64}{15}x_i\right) - sin\left(1-\frac{16}{15}x_i\right)\right]$$

Test area is usually restricted to -1≤ x_i≤ 1, i = 1,..., n. Its global minimum equal f(x) = 0.0644 is obtainable for (x^*) = (0.4673, 0.4673).

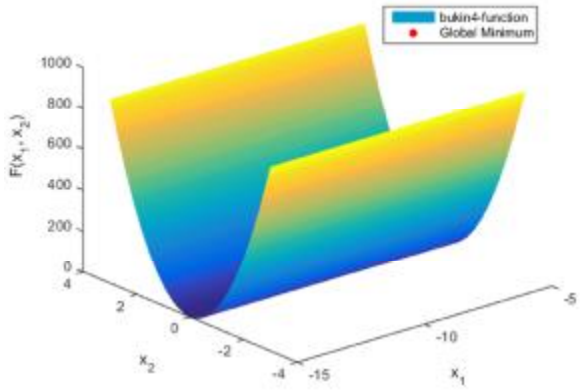
3.8.11. F_{11}: Booth



$$f_{11} = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

Test area is usually restricted to -10≤ x_i≤10, i = 1, 2. Its global minimum equal f(x) = 0 is obtainable for $(x^*) = (1, 3)$.

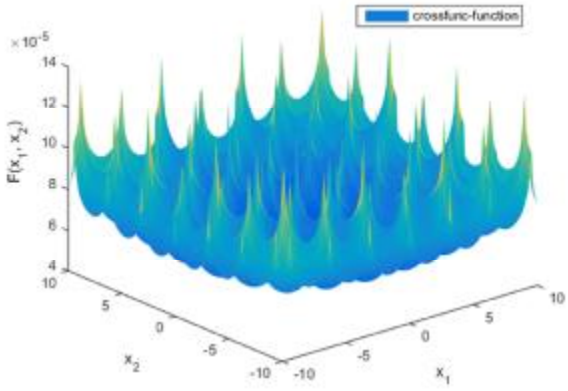
3.8.12. F_{12}: Bukin4



$$f_{12} = 100x_2^2 + 0.01|x_1 + 10|$$

Test area is usually restricted to -15≤ x_1≤-5 and -3≤ x_2≤3, i = 1, 2. Its global minimum equal f(x) = 0 is obtainable for $(x^*) = (-10, 0)$.

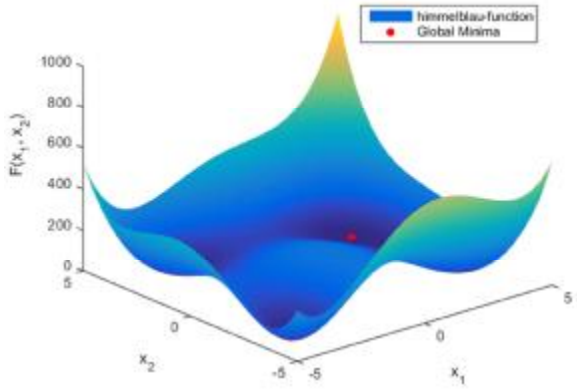
3.8.13. F_{13}: Cross-In Tray Function



$$f_{13} = -0.0001\left(\left|sin(x_1)sin(x_2)exp\left(\left|100 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\right|\right)\right| + 1\right)^{0.1}$$

Test area is usually restricted to -10≤ x_i≤10, i = 1, 2. Its global minimum equal f(x) = -2.06261 is obtainable for $(x^*) = \pm 1.3491$.

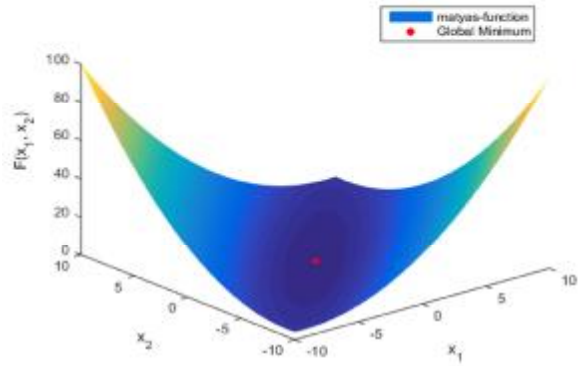
3.8.14. F_{14}: Himmelblau



$$f_{14} = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

Test area is usually restricted to -5≤ x_i≤5, i = 1, 2. Its global minimum equal f(x)=0 is obtainable for $(x^*) = (3,2), (3.5, -1.8), (-3.7, -3.2), (-2.8, 3.1)$.

3.8.15. F_{15}: Matyas



$$f_{15} = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$$

Test area is usually restricted to -10≤ x_i≤10, i = 1, 2. Its global minimum equal f(x) = 0 is obtainable for $(x^*) = (0, 0)$.

3.9. Real World Datasets

Eight real world datasets with known dependence attributes were used to test the practicality of the proposed algorithms. The benchmark datasets we use in the test process are listed in Table 3.2. Datasets are downloaded from the UCI repository (Lichman, 2013). All the input features and targets have been normalized into the range [0, 1].

Table 3.2. Description of Datasets

Dataset	Instances	Features
Airfoil Self-Noise	1503	6
Air Quality	9358	15
Combined Cycle Power Plant	9568	5
Concrete Comp. Strength	1030	9
Energy efficiency (Output 1)	768	10
Energy efficiency (Output 2)	768	10
Wine Quality-Red	1599	12
Wine Quality-White	4989	12

3.9.1. Airfoil Self-Noise Dataset

Airfoil Self-Noise is a NASA dataset which is obtained from a series of aerodynamic and acoustic tests of two and three-dimensional airfoil blade sections conducted in an anechoic wind tunnel (Brooks et al., 1989). Each datum has the following features:

1. Frequency, in Hertzs.
2. Angle of attack, in degrees.
3. Chord length, in meters.
4. Free-stream velocity, in meters per second.
5. Suction side displacement thickness, in meters.

The output is:

6. Scaled sound pressure level, in decibels.

3.9.2. Air Quality Dataset

Air Quality includes the responses of a gas multisensor device positioned in an Italian city (Vito et al., 2008). Hourly average responses are recorded along with gas concentrations references obtained from a certified analyzer. Each datum has the following features, however in this study first two features are ignored due to their data format:

1. Date (DD/MM/YYYY)
2. Time (HH.MM.SS)
3. Concentration CO in mg/m3
4. PT08.S1 (tin oxide) sensor response
5. Overall Non Metanic HydroCarbons concentration in microg/ m3

6. Benzene concentration in microg/ m3
7. PT08.S2 (titania)
8. NOx concentration in ppb
9. PT08.S3 (tungsten oxide)
10. NO2 concentration in microg/m3
11. PT08.S4 (tungsten oxide)
12. PT08.S5 (indium oxide)
13. Temperature in Â°C
14. Relative Humidity (%)

The output is:

15. AH Absolute Humidity

3.9.3. Combined Cycle Power Plant Dataset

Combined Cycle Power Plant Dataset obtained from a Combined Cycle Power Plant over 6 years (2006-2011) (Pınar, 2014). CCPP is consist of Steam Turbines (ST), Gas Turbines (GT) and heat recovery steam generators. In a CCPP, the electricity is produced by gas and steam turbines, which are combined in one cycle, and is transmitted from one turbine to another. While the Vacuum is collected from and has effect on the Steam Turbine, other three of the ambient variables affect the GT performance. Features consist of hourly average values. Each datum has the following features:

1. Temperature (T) in the range 1.81°C and 37.11°C,
2. Ambient Pressure (AP) in the range 992.89-1033.30 milibar,
3. Relative Humidity (RH) in the range 25.56% to 100.16%
4. Exhaust Vacuum (V) in teh range 25.36-81.56 cm Hg

The output is:

5. Net hourly electrical energy output (EP) 420.26-495.76 MW

3.9.4. Concrete Compressive Strength Dataset

The concrete compressive strength is the prediction problem. Predictions can be made from the available data (Yeh, 1998). Each datum in the dataset has the following features:

1. Cement, kg in an m3
2. Blast Furnace Slag, kg in an m3
3. Fly Ash, kg in an m3
4. Water , kg in an m3
5. Superplasticizer, kg in an m3
6. Coarse Aggregate, kg in an m3
7. Fine Aggregate, kg in an m3
8. Age, day (1~365)

The output is:

9. Concrete compressive strength

3.9.5. Energy Efficiency Dataset

In this dataset energy analysis is performed by using 12 different building shapes simulated in Ecotect (Tsanas and Xifara, 2012). The buildings differ with respect to the glazing area, the glazing area distribution, and the orientation, amongst other parameters. The dataset aiming to predict two real valued results. Each datum in the dataset has the following features:

1. Relative Compactness
2. Surface Area
3. Wall Area
4. Roof Area
5. Overall Height
6. Orientation
7. Glazing Area
8. Glazing Area Distribution

The outputs are:

9. Heating Load
10. Cooling Load

Since this dataset includes two outputs, in this study this dataset is considered as two different datasets by treating each output separately.

3.9.6. Wine Quality Datasets

Wine quality includes two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine. The goal is to predict wine quality based on physicochemical tests (Cortez et al., 2009). Each datum in the dataset has the following features:

1. fixed acidity
2. volatile acidity
3. citric acid
4. residual sugar

5. chlorides
6. free sulfur dioxide
7. total sulfur dioxide
8. density
9. pH
10. sulphates
11. alcohol

Output variable (based on sensory data):

12. quality (score between 0 and 10)

3.10. Performance Evaluation Methods

3.10.1. N-Fold Cross Validation

Proposed algorithms, SGRNN and other two hybrid GRNN were tested on eight UCI datasets and 1000 randomly created data for each test functions. 10-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 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 and average error values were used to present the efficiency of predictions (Hamzacebi, 2008).

3.10.2. Evaluation Metrics

Three standard error measures were implemented to evaluate the prediction performance; Mean Squared Error (MSE), Mean Absolute Error (MAE) and Mean

Absolute Percentage Error (MAPE). MSE is a commonly used measure of the quality of an estimator. It is the second moment (about the origin) of the distance; thus, it includes both the variance of the estimator and its bias. Outliers have greater effect on MSE than on MAE and MAPE. As the name suggests, MAE is average of the absolute errors, and uses the same scale as the data being measured. MAPE is another measure of prediction accuracy of a forecasting method in statistics and expresses accuracy as a percentage of the error. Hence, it can be easier to understand than the other measurements. For all three measures given in (3.14.), (3.15.) and (3.16.), respectively; smaller values indicate better fitting model.

$$\text{MSE} = \frac{1}{N_{test}} \sum_{q=1}^{N_{test}} \left(y_{test_q} - Y_q(\vec{x}_{inp}) \right)^2 \tag{3.14.}$$

$$\text{MAE} = \frac{1}{N_{test}} \sum_{q=1}^{N_{test}} \left| y_{test_q} - Y_q(\vec{x}_{inp}) \right| \tag{3.15.}$$

$$\text{MAE} = \frac{1}{N_{test}} \sum_{q=1}^{N_{test}} \left| \frac{y_{test_q} - Y_q(\vec{x}_{inp})}{y_{test_q}} \right| * 100\% \tag{3.16.}$$

where;

N_{test} number of test data

y_{tes_q} actual value of the q^{th} element of the dataset

$Y_q(\vec{x}_{inp})$ predicted value of the q^{th} test data

4. PRELIMINARY STUDY

Clustering algorithms have been fundamental pre-processors for GRNN in the literature. Hence, in our study before starting to suggest new methodology, we determine an appropriate clustering algorithm to use it as pre-processor for GRNN. When determining this algorithm, there exists some issues to be considered: chosen algorithm should both represent traditional approaches and be used as a basis for our proposed methods. Performance of this conventional method will also be used for comparison with proposed methods.

For this purpose, a quick study was conducted on 10 test functions with 4 popular clustering algorithms; K-means, Hierarchical clustering, Gaussian mixture model and Fuzzy c-means. All the methods were implemented and analysed as pre-processors of GRNN. Unlike standard GRNN that uses whole training set in the pattern layer, in clustering GRNN only related data groups were used for current test datum.

4.1. Cluster Count Estimation

However, like all other optimization problems, clustering is also a kind of NP-hard problem. One of the biggest problem is to determine the best estimate for the number of clusters (k). 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 cluster for each real world datasets and benchmark functions (Calinski 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 cluster numbers in this range were given in the Table 4.1

Table 4.1. Estimated cluster counts

Dataset	Cal. Har.	Sil.	Gap	Dav. Boul.
R_1	100	2	97	2
R_2	98	2	97	2
R_3	98	2	97	2
R_4	97	2	100	2
R_5	99	2	98	2
R_6	98	2	2	2
R_7	100	2	2	2
R_8	100	2	2	2
f	99	2	2	2

It is seen that, there is not any 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. In detail, for 5 cluster GRNN all the training data are separated into the 5 clusters. Then, when a new test datum is applied, data belonging to the best matching cluster are used as training data and remaining training data that belongs to the other 4 clusters, are ignored. In other words, selected training data correspond to the 1/5 (20%) of the all training data.

After the clustering process is completed, each data group has to be configured with different sigma values. Therefore, sigma optimization procedure was applied for groups one by one. Prediction results of the GRNN with four different clustering algorithms for three chosen cluster counts are given in Table 4.2. - Table 4.10.

Table 4.2. Clustering - GRNN results with 5 cluster (MSE)

Func.	K-means	Hier.	FCM	Gauss.
f_1	**1.15E+07**	1.33E+07	1.17E+07	4.07E+07
f_2	**5.24E+01**	6.94E+01	5.26E+01	7.79E+01
f_3	**1.16E+04**	1.72E+04	1.17E+04	2.55E+04
f_4	4.43E-01	6.32E-01	**4.41E-01**	6.08E-01
f_5	**1.50E+08**	2.59E+08	1.51E+08	6.42E+08
f_6	**3.46E+04**	4.78E+04	3.53E+04	7.16E+04
f_7	**3.03E+01**	7.26E+01	**3.03E+01**	1.36E+02
f_8	**3.65E+02**	4.63E+02	3.69E+02	7.28E+02
f_9	**3.36E+00**	6.82E+00	3.40E+00	1.44E+01
f_{10}	4.83E-04	6.79E-04	**4.76E-04**	8.96E-04

Table 4.3. Clustering - GRNN results with 5 cluster (MAE)

Func.	K-means	Hier.	FCM	Gauss.
f_1	**1.04E+03**	1.17E+03	**1.04E+03**	1.69E+03
f_2	**5.59E+00**	6.27E+00	**5.59E+00**	6.63E+00
f_3	**7.89E+01**	9.19E+01	7.90E+01	1.08E+02
f_4	**5.12E-01**	5.58E-01	**5.12E-01**	5.59E-01
f_5	4.40E+03	5.42E+03	**4.38E+03**	7.67E+03
f_6	**8.97E+01**	1.06E+02	9.01E+01	1.31E+02
f_7	3.41E+00	4.59E+00	**3.39E+00**	5.93E+00
f_8	**8.74E+00**	1.03E+01	8.75E+00	1.32E+01
f_9	**1.31E+00**	1.68E+00	1.32E+00	2.30E+00
f_{10}	1.47E-02	1.70E-02	**1.46E-02**	1.92E-02

Table 4.4. Clustering - GRNN results with 5 cluster (MAPE)

Func.	K-means	Hier.	FCM	Gauss.
f_1	**4.24E+01**	1.05E+02	4.69E+01	1.08E+02
f_2	**2.42E+01**	2.72E+01	2.43E+01	3.31E+01
f_3	3.00E+02	3.06E+02	**2.93E+02**	4.04E+02
f_4	**2.87E+00**	3.23E+00	2.88E+00	3.20E+00
f_5	**2.23E+01**	3.62E+01	**2.23E+01**	6.58E+01
f_6	3.34E+01	5.42E+01	**3.20E+01**	1.05E+02
f_7	4.51E+01	6.64E+01	**4.39E+01**	8.55E+01
f_8	**2.05E+02**	2.09E+02	**2.05E+02**	3.64E+02
f_9	**6.32E+00**	8.58E+00	6.36E+00	1.23E+01
f_{10}	4.65E+00	5.41E+00	**4.64E+00**	6.81E+00

Table 4.5. Clustering - GRNN results with 10 cluster (MSE)

Func.	K-means	Hier.	FCM
f_1	**1.16E+07**	1.44E+07	1.18E+07
f_2	**5.36E+01**	7.45E+01	5.42E+01
f_3	**1.20E+04**	1.80E+04	1.21E+04
f_4	4.50E-01	5.98E-01	**4.39E-01**
f_5	**1.55E+08**	2.74E+08	1.60E+08
f_6	**3.65E+04**	4.97E+04	3.74E+04
f_7	**3.41E+01**	6.76E+01	3.46E+01
f_8	**3.75E+02**	5.10E+02	3.85E+02
f_9	**3.59E+00**	6.82E+00	3.60E+00
f_{10}	**4.89E-04**	7.78E-04	4.92E-04

Table 4.6. Clustering - GRNN results with 10 cluster (MAE)

Func.	K-means	Hier.	FCM
f_1	**1.05E+03**	1.21E+03	1.06E+03
f_2	**5.64E+00**	6.42E+00	5.66E+00
f_3	**7.98E+01**	9.27E+01	8.02E+01
f_4	5.13E-01	5.49E-01	**5.12E-01**
f_5	**4.56E+03**	5.80E+03	4.60E+03
f_6	**9.28E+01**	1.11E+02	9.37E+01
f_7	**3.56E+00**	4.59E+00	**3.56E+00**
f_8	**9.25E+00**	1.11E+01	9.27E+00
f_9	1.36E+00	1.71E+00	**1.35E+00**
f_{10}	**1.51E-02**	1.85E-02	**1.51E-02**

Table 4.7. Clustering - GRNN results with 10 cluster (MAPE)

Func.	K-means	Hier.	FCM
f_1	**4.22E+01**	5.53E+01	4.44E+01
f_2	**2.52E+01**	2.77E+01	**2.52E+01**
f_3	**3.04E+02**	3.02E+02	3.06E+02
f_4	2.88E+00	3.13E+00	**2.86E+00**
f_5	2.32E+01	3.21E+01	**2.31E+01**
f_6	**3.60E+01**	4.68E+01	3.66E+01
f_7	4.77E+01	5.81E+01	**4.55E+01**
f_8	2.06E+02	2.13E+02	**2.05E+02**
f_9	6.52E+00	8.23E+00	**6.48E+00**
f_{10}	4.86E+00	6.12E+00	**4.85E+00**

Table 4.8. Clustering - GRNN results with 20 cluster (MSE)

Func.	K-means	Hier.	FCM
f_1	**1.19E+07**	1.92E+07	1.34E+07
f_2	**5.73E+01**	7.66E+01	5.79E+01
f_3	**1.27E+04**	1.74E+04	**1.27E+04**
f_4	**4.65E-01**	5.47E-01	4.70E-01
f_5	1.76E+08	3.22E+08	**1.75E+08**
f_6	4.00E+04	5.95E+04	**3.92E+04**
f_7	4.07E+01	6.80E+01	**4.04E+01**
f_8	**4.34E+02**	6.22E+02	4.41E+02
f_9	**3.98E+00**	7.06E+00	4.04E+00
f_{10}	**6.23E-04**	8.43E-04	6.24E-04

Table 4.9. Clustering - GRNN results with 20 cluster (MAE)

Func.	K-means	Hier.	FCM
f_1	**1.10E+03**	1.35E+03	1.11E+03
f_2	**5.78E+00**	6.51E+00	5.82E+00
f_3	**8.15E+01**	9.26E+01	8.16E+01
f_4	**5.24E-01**	5.47E-01	5.25E-01
f_5	**4.92E+03**	6.17E+03	**4.92E+03**
f_6	9.84E+01	1.19E+02	**9.79E+01**
f_7	**3.82E+00**	4.69E+00	**3.82E+00**
f_8	1.05E+01	1.23E+01	1.06E+01
f_9	**1.42E+00**	1.76E+00	**1.42E+00**
f_{10}	**1.76E-02**	2.02E-02	1.77E-02

Table 4.10. Clustering - GRNN results with 20 cluster (MAPE)

Func.	K-means	Hier.	FCM
f_1	**4.43E+01**	5.53E+01	4.88E+01
f_2	**2.46E+01**	2.80E+01	**2.46E+01**
f_3	**2.99E+02**	3.30E+02	3.04E+02
f_4	**2.90E+00**	3.08E+00	2.91E+00
f_5	**2.46E+01**	3.21E+01	**2.46E+01**
f_6	**3.59E+01**	5.94E+01	4.05E+01
f_7	5.43E+01	5.85E+01	**5.**24E+01
f_8	2.06E+02	2.04E+02	**2.03E+02**
f_9	6.74E+00	8.03E+00	6.78E+00
f_{10}	**5.64E+00**	6.47E+00	**5.64E+00**

Results in Table 4.2.- Table 4.10. show that the prediction accuracies of GRNN with k-means and Fuzzy c-means are almost the same and performances of both algorithms are superior to that of Hierarchical and Gaussian clustering with respect to MSE, MAE and MAPE values for all given cluster counts. In Table 4.2. MSE values of four clustering algorithms with cluster count 5 are given. The results given in bold illustrate the best results of related test functions. The other two tables following the Table 4.2. illustrate the MAE and MAPE values for the same cluster count. Similar notation was used for cluster count 10 and 20 in Tables 4.5.- 4.10. In addition, when the cluster count raised to 10, in Gaussian mixture model, some cluster size became less than kfold value and this caused to an exception. Accordingly, this algorithm could not be evaluated. In addition to the obtained results, an example view of clustering algorithms' results for Beale function is given in Figure 4.1.–Figure 4.4., respectively. X labels illustrate the centre of clusters.

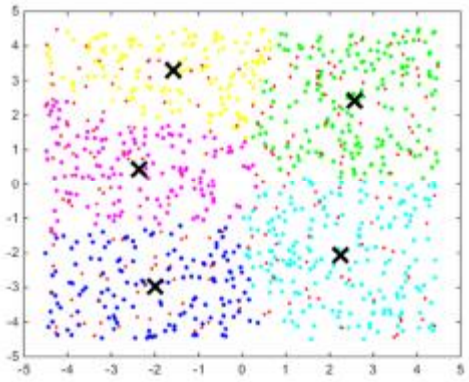


Figure 4.1. K-means

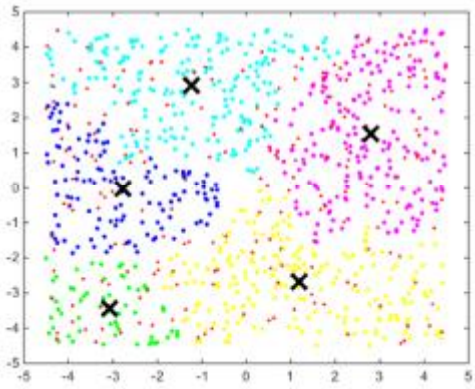


Figure 4.2. Hierarchical Clustering

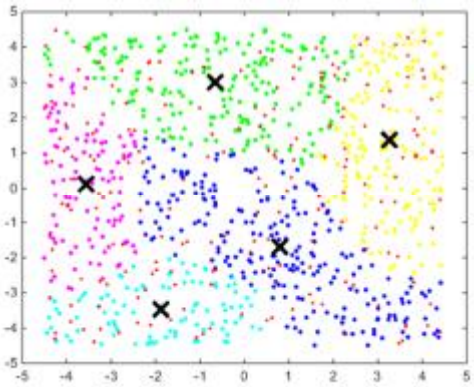


Figure 4.3. Gaussian Mixture Model

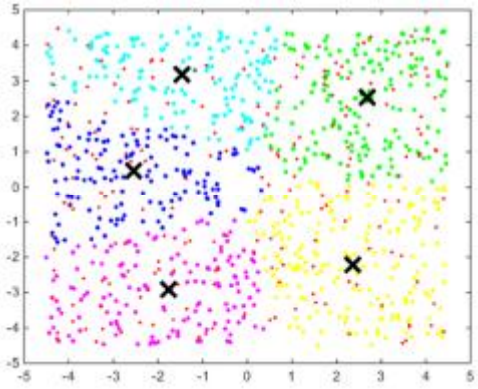


Figure 4.4. Fuzzy C-means

In given figures, each colour represents a different cluster. In other words, while the training data illustrated with the same colour are members of the same cluster, the training data illustrated with the different colours are members of the different clusters. According to this illustration, it is expected that the training data having the same cluster colour should be close to each other at maximum level, as a result of a well-designed clustering algorithm.

Consequently, k-means and fuzzy c-means gave the best results among the other clustering algorithms. Also, it can be clearly seen from the figure k-means and fuzzy c-means offered the best performances. According to the experimental

results, the most popular clustering algorithm in the literature, k-means, was selected to be used as pre-processing algorithm.

5. METHODS

5.1. A Cluster Based Approach to Reduce Pattern Layer Size for GRNN

In order to reduce space and time complexity of GRNN, use of k-means clustering algorithm as a pre-processor has been studied in the literature. However, the sample data that lay between cluster borders (inner cluster outliers), produce large residuals resulting negative influence on prediction performance. Therefore, they need special attention. In this study, we detect such data by using probabilistic density function after applying k-means algorithm, and try to eliminate them.

As mentioned before, the main purpose of the clustering algorithms is to group similar data given in the dataset (Jain et al., 1999; Fahad et al., 2014; Berkhin, 2002). Thus, when it is necessary to predict the result of a test datum with the GRNN algorithm, only the training data belong to the same cluster with the test datum can be used instead of all dataset. In addition, it is aimed to make the estimation process with fewer errors by recalibrating the sigma (σ) parameter, which is the only parameter affecting the GRNN for each cluster separately.

While pre-processor algorithms are being utilized to reduce the processing complexity of the GRNN, in some applications it has been observed that the amount of error in prediction results increased. The reason for this problem is that GRNN makes highly erroneous predictions if it encounters with outliers. When clustering algorithms are used, the test datum existing in the borders of the clusters behaves as an outlier for that cluster (Yamanishi et al, 2000). In this case, these data are estimated only by the training data of one cluster and cause a high amount of errors.

In this study, unlike the pre-processor algorithms in the literature, not only clustering pre-processing was performed, but also test data falling between the clusters were detected. Thus, outlier data formation was prevented and the resulting errors were significantly decreased. The proposed algorithm was tested on fifteen different test functions and eight real datasets, which are widely used in the

literature. The obtained results were compared with the results of the SGRNN and the k-means GRNN.

5.1.1. Inner-Outlier Detection Algorithm

In clustering-GRNN, the training data which are different from test datum are removed from system, in other words, the test datum is estimated by using only the training data which are the most similar to it. Hence, it is thought that there should not be a major change in the quantity of error. However, when the algorithms are tested, it is clearly seen that the results are different from the expected. This leads to an idea that there should be a new situation arising from clustering operations. During the evaluation process, when the test data causing suddenly increase in error value were examined, it was determined that these data existed in the boundary regions of the clusters. Even though these data are not outliers for the standard GRNN, they may behave as outliers for clustering-GRNN algorithms. Owing to some part of training data which are similar to the test datum (outlier datum) are in another cluster, more erroneous prediction result comparing to the standard algorithm is obtained. In this case, a test datum which is not an outlier for the SGRNN, turns into an outlier datum in clustering-GRNN, it is illustrated in Figure 5.1. and Figure 5.2., respectively. While in Figure 5.1., the circled test data is estimated by using all the training data shown by green, in Figure 5.2. it is estimated by using only the training data shown in blue.

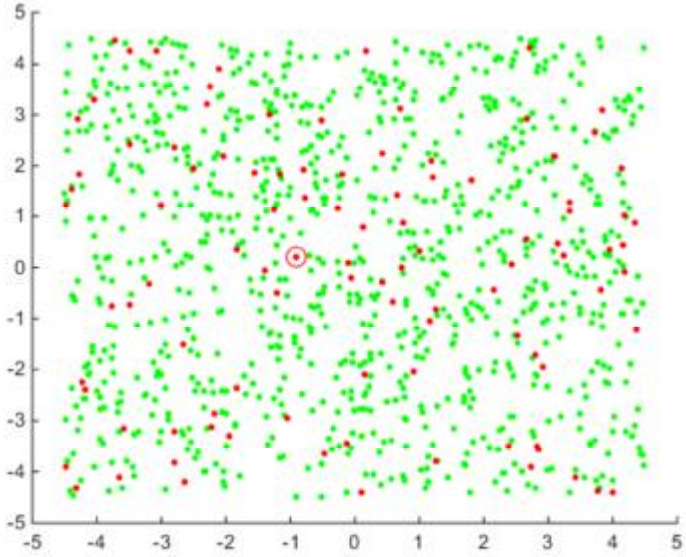


Figure 5.1. Standard GRNN

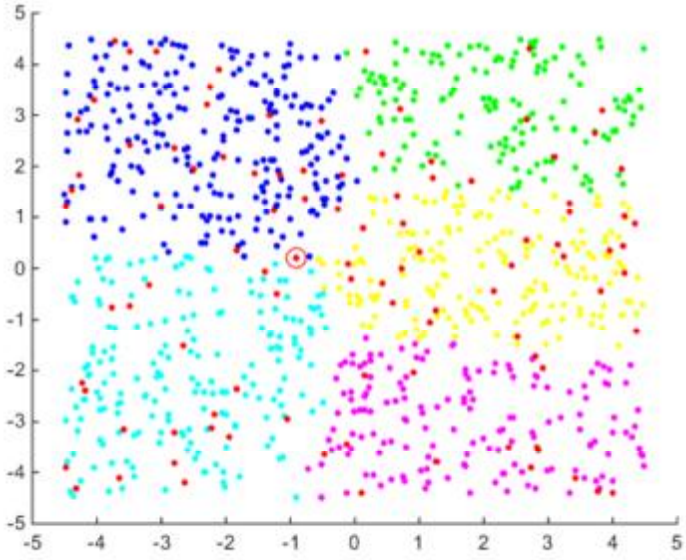


Figure 5.2. GRNN with k-means clustering algorithm

In this study, unlike other studies in the literature, the emerging inner-outlier data are detected and other cluster sets adjacent to them are also used during the estimation of these data. A multivariable normal probability density function is used during the detection of outliers. Initially, membership degree of each test datum to the clusters are calculated by this function. The sum of the calculated membership values for a test datum is 1. A threshold value is set to decide which clusters the test datum belongs to. A test datum whose membership values were calculated, is considered as member to all clusters for which its membership value is bigger than threshold value. Test data that belong to the more than one cluster are marked as inner-outlier data. If a test datum does not member to any cluster, threshold value only for related test datum is reduced, and the membership operations are repeated until the clusters are determined which the test datum belongs to. Since the different test functions are used, they have different search space boundaries, a threshold function has been formulated to be used in all test functions. The formulated test function is given in (5.1.).

$$TH = \left(\frac{1}{|ll-ul|*2}\right)^{m-1} \tag{5.1.}$$

where, ll and ul point lower and upper limit values of each dimension, respectively, m points the dimensions of the data and TH points the calculated threshold value. An example outcome of outlier detection algorithm for cluster count 5 is shown in Figure 5.3., where red-circles point out the outlier data.

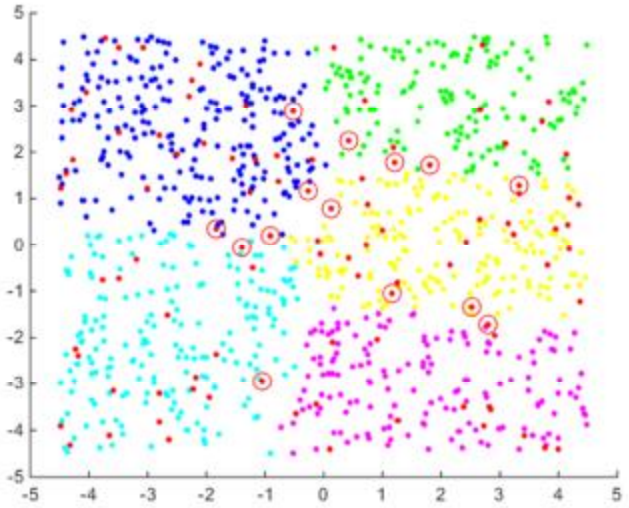


Figure 5.3. Inner-outlier detection

5.1.2. Algorithm

Let x denote m-dimensional training data, N be the number of training data, y represent target values of the training samples, x^{red} be m-dimensional chosen samples belonging to selected clusters, R be the number of reduced training data, k be the number of cluster, c be the cluster centres, S_{CLS} be the selected clusters and $\vec{x}_{inp}$ m-dimensional test datum. Algorithm of k-means GRNN with outlier detection is given below:

Algorithm 5.1. Algorithm of k-means GRNN with outlier detection

Data: $x,\ \vec{x}_{inp} y, c$

Result: $Y(\vec{x}_{inp})$

Assign random values to c;

Apply steps of k-means to cluster training data using x and c ;

Apply $\vec{x}_{inp}$ as test ;

While $j <= k$ do

$dist_j(\vec{x}_{inp}) = \|\vec{x}_{inp} - \vec{c}_j\|$;

end

Calculate threshold as

$TH = \left(\frac{1}{|ll-ul|*2}\right)^{m-1}$;

Find clusters S_{CLS} whose $dist$ value less than TH;

Find training data x^{red} belonging to S_{CLS};

While $r <= R$ do

Calculate squared Euclidean distance between $\vec{x}_{inp}$ and $\vec{x}_r^{red}$ as

$$dist_r = \sum_{i=1}^{m} \left(x_{inp,i} - x_{r,i}^{red}\right)^2 ;$$

Compute $nom = nom + y_r * e^{\left[\frac{-dist_r}{2\sigma^2}\right]}$;

Compute $denom = denom + e^{\left[\frac{-dist_r}{2\sigma^2}\right]}$;

end

$Y(\vec{x}_{inp}) = \frac{nom}{denom}$;

5.1.3. Complexity Analysis

K-means GRNN with outlier detection consists of two separate steps, k-means and GRNN. Outlier detection is the single step operation which is small enough to be ignored and performed before GRNN process started. If we analyse complexity of the k-means GRNN, both algorithms must be examined separately and summed at final step.

Suppose that N is the number of m-dimensional training data, k is the number of cluster which is small enough to be ignored when compared to number of training data and I is number of iterations.

Computational complexity of the traditional k-means algorithm for each iteration will be $O(Nkm)$ and total complexity of the k-means can be considered as $O(INkm)$. The second step of the algorithm consists of GRNN, which is the one pass learning algorithm and has $O(Nm)$ computational complexity for prediction of one test datum. However, in k-means GRNN from N to R where $R << N$. Consequently, k-means GRNN algorithm has $O(INkm) + O(Rm)$ complexity, which equals to $O(INkm)$.

However, k-means is a pre-processing algorithm and has to be performed before testing step. Thus, when a new test datum applied, there remains two operations that are detection of nearest clusters and GRNN operation with reduced training data. This feature ensures faster prediction for the instantaneous test datum, and decrease in computational complexity from $O(Nm)$ to $O(Rm)$, where $R << N$.

5.2. New Pattern Reduction Method for Generalized Regression Neural Network

As noted in Section 2., data reduction algorithms are examined under two categories. As an alternative to the first category algorithms, in other words to the clustering algorithms, kNN algorithm which is the most popular algorithm of the second category was used in this part of the thesis. Firstly, the kNN-GRNN algorithm was developed and examined, secondly its weakness was detected, and lastly an alternative method that can be used for three-dimensional data was proposed.

kNN method is a useful alternative to the clustering algorithms for pattern layer size reduction. Contrary to the clustering methods, it does not require estimation of the cluster number, it uses only neighbourhoods. Since, kNN uses

only k-nearest training data, it reduces operation cost from O(N) to O(k); where k<N and it is more applicable for large scale applications. However, there are still two critical questions have to be answered, how to decide the best neighbours and the number of neighbours (k). Commonly, Euclidean distance is used for detecting the nearest k-data. However, defining the number of nearest neighbours (k) is more critical issue. While the selection of small k makes the network unreliable, the selection of large k may cause some remote data be taken into evaluation, which decreases prediction efficiency.

Even though all the pre-processing algorithms given in the literature reduce the data complexity of GRNN network, prediction accuracy could not be enhanced adequately. Furthermore, there is one more shortcoming of the GRNN algorithm that has not been addressed yet; being sensitive to the local data groups. By the nature of the GRNN algorithm, the test datum is more sensitive to the training data closest to itself. So, if there exists a data heap around the test datum, prediction may be close to average value of that data heap and more prediction error may occur than expected. Taking into account these troubles, in this study, we proposed the "Angle Based Nearest Neighbour" (ABNN)-GRNN that reduces sample complexity and prediction error for three-dimensional datasets.

5.2.1. Angle Based Nearest Neighbour (ABNN)

kNN is also used as an alternative pre-processing method for GRNN (Kokkinos and Margaritis, 2015). In standard GRNN, distance value between each training datum and test datum is utilized as a measure of how well training datum can represent test datum. But, if the test datum is located around of a data heap, the prediction result may be average value of this data heap. This situation may cause increase in prediction error and decrease in prediction accuracy. Even though this problem has not been discussed too much, becoming more apparent with the use of local learning algorithms.

As mentioned earlier, the most challenging part of the kNN algorithm is determination of the suitable k value. Also, when the kNN algorithm cooperates with the GRNN algorithm, the sensitivity to the data heaps problem may occur. Selected k nearest training data may have been taken from the same data heap. In this case, the classification or regression algorithm may produce inaccurate results. Unlike the results in Figure 3.3, the kNN algorithm may produce a result as in Figure 5.4. It is clearly seen that most of the training data are grouped in a certain region and this may result in a prediction value that is around the average point of these data.

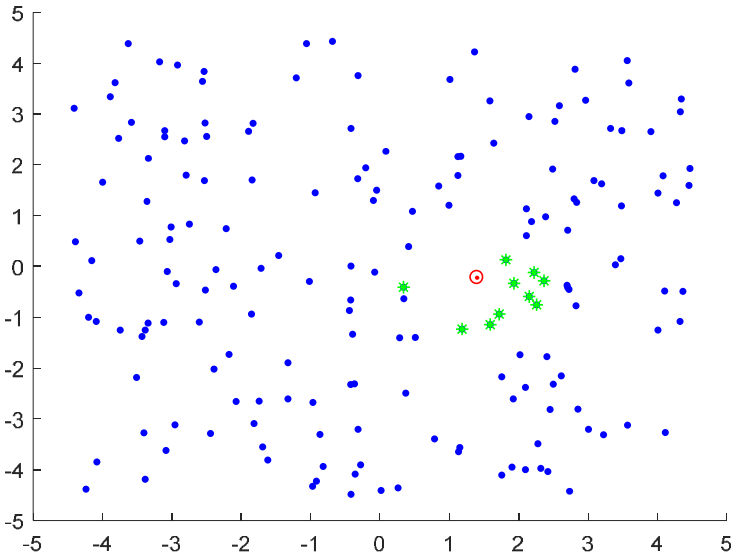


Figure 5.4. 10 Nearest Neighbours, over Stacked Data

To overcome this problem, an Angle Based Nearest Neighbour (ABNN) algorithm was employed instead of kNN (Kartal and Oral, 2017). ABNN ensures finding the nearest training data from all around the test datum step by step. In each step, only one training datum is selected within given angle. Once the all training data are selected around the test datum, they are fed into the GRNN to be used in the forecasting process.

While choosing training data, it is necessary to find the angle θ between the test datum and each training datum. In ABNN, we utilized arctangent function to find angle θ. Calculation of the θ between two points, A and B, is illustrated in Figure 5.5.

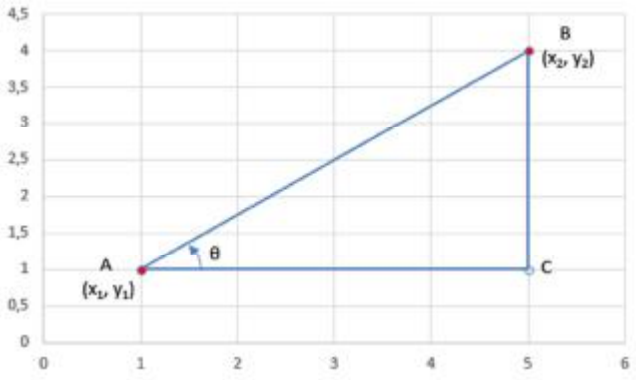


Figure 5.5. Finding of the θ between two points

Initially, distance values between related points for each dimension are measured. Then, measurements are given to the arctangent function as parameters, as in (5.2.). Lastly, calculated radian value is converted to the degree with Equation (5.3.).

$$\text{Radian}(A, B) = \text{arctangent}\left(\frac{|BC|}{|AC|}\right) \tag{5.2.}$$

$$\theta(A, B) = \left(\frac{\pi}{180}\right) * \text{Radian}\,(A, B) \tag{5.3.}$$

Outcomes of ABNN algorithm with θ =40 are shown in Figure 5.6. While green-square points out the test datum, red-triangles point out found the nearest data in given angle.

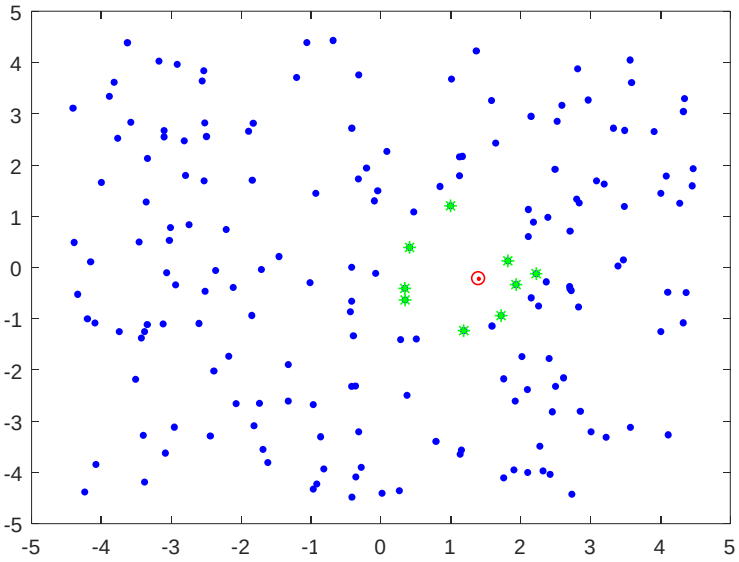


Figure 5.6. ABNN with θ = 40

When compatibilities of the kNN and ABNN algorithms are compared, it can be seen that ABNN is more successful than kNN in selecting of representative training data. Furthermore, while training data obtained from kNN lead to the same prediction result with that of SGRNN, training data obtained from ABNN do not get stuck on data heaps and produce more accurate results.

If we consider complexity analysis, computational complexity of kNN for one test datum is O(N x cost of distance calculation between test datum and a training datum), where N is the number of training data. Additionally, ABNN needs one extra calculation for each test datum to find out angle between test datum and each training datum. Thus, computational cost of the ABNN algorithm for one test datum is O(N x costs of distance and angle calculations between test datum and a training datum). Nevertheless, calculations indicate that both kNN and ABNN have the same computational complexity, O(N), asymptotically.

5.2.2. Algorithm

Let x denote 3-dimensional training data, $dist_j(\vec{x}_{inp})$ be distance between test datum and j^{th} training data, $\theta_j(\vec{x}_{inp})$ be angle between test datum and j^{th} training data, N be the number of training data, R be the number of reduced training data, y contain target values of the training samples, x^{red} be 3-dimensional chosen nearest neighbours, θ be the angle value and $\vec{x}_{inp}$ denote 3-dimensional test datum. Algorithm of ABNN-GRNN is given below:

Algorithm 5.2. ABNN -GRNN Algorithm

Data:) $x, \vec{x}_{inp} y, \theta$
Result: $Y(\vec{x}_{inp})$
Apply $\vec{x}_{inp}$ as test ;
While $j <= N$ do
Calculate angle and Euclidean distance between $\vec{x}_{inp}$ and $\vec{x}_j$ as
$Radian_j(\vec{x}_{inp}) = arctangent\left(\frac{|x_{inp,2}-x_{j,2}|}{|x_{inp,1}-x_{j,1}|}\right)$;
$\theta_j(\vec{x}_{inp}) = \left(\frac{\pi}{180}\right) * Radian_j(\vec{x}_{inp})$;
$dist_j(\vec{x}_{inp}) = \sqrt{\sum_{i=1}^{m}\left(x_{inp,i} - x_{j,i}\right)^2}$;
end
Sort All the training data according to $\theta(\vec{x}_{inp})$; $R = \frac{360}{\theta}$;
While $a <= R$ do
Calculate upper and lower limits for a^{th} angle
$\theta_{lower} = \frac{360}{\theta} * (a-1)$;
$\theta_{upper} = \frac{360}{\theta} * (a)$;
Find training data, t_{angle} , having angle value between θ_{lower} and θ_{upper};
Sort t_{angle} according to $dist(\vec{x}_{inp})$;
Take nearest training data and add into the x^{red}
end
While $r <= R$ do
Calculate squared Euclidean distance between $\vec{x}_{inp}$ and $\vec{x}_r^{red}$ as
$dist_r = \sum_{i=1}^{m}\left(x_{inp,i} - x_{r,i}^{red}\right)^2$;
Compute $nom = nom + y_r * e^{\left[\frac{-dit_r}{2\sigma^2}\right]}$;
Compute $denom = denom + e^{\left[\frac{-dist_r}{2\sigma^2}\right]}$;
end
$Y(\vec{x}_{inp}) = \frac{nom}{denom}$;

5.2.3. Complexity Analysis

In ABNN-GRNN, both algorithms are in nested structure. Relationships between the test datum and all training data are determined when test datum is applied. If we analyse complexity of the algorithm.

Suppose that N is the number of m-dimensional training data, θ is the angle value, R is the number of selected training data which is small enough to be ignored when compared to number of training data and I is number of iterations.

In basic ABNN for each test datum, R training data have to be detected, which requires $2N$ angle calculation, N distance calculation, 1 sorting operation for angle values and R sorting operations for distance values. These operations have $O(3Nm)$, $O(NlogNm)$, and $O(RNlogNm)$ complexity, respectively. Then, GRNN algorithm starts, which is the one pass learning algorithm and has $O(Nm)$ computational complexity for prediction of one test datum. However, in ABNN-GRNN number of training data is reduced from N to R where $R << N$. Consequently, ABNN -GRNN algorithm has $O(3Nm+ NlogNm+ RNlogNm+ Rm)$ complexity, which equals to $O(RNlogNm)$. However, ABNN is not a pre-processing algorithm. Therefore, all the operations have to be done after test datum is applied.

5.3. Pattern Layer Reduction for Generalized Regression Neural Networks by Using Self Organizing Map

Two algorithms we developed previously have provided improvements in the performance of conventional algorithms. In this part of the thesis, in addition to these two algorithms, SOM based hybrid GRNN algorithm which provides further improvement on the results, was presented.

In the first method, the disadvantages of the clustering algorithms were significantly removed and the traditional k-means GRNN algorithm was converted into a more effective algorithm which produces tolerable amount of error with much less training data. In the proposed second algorithm, the problem of the

kNN-GRNN, getting stuck on local data groups, was solved by using an ABNN algorithm instead of kNN.

In this part of the study, a new SOM based GRNN algorithm that selects the significant training data and reduces the data complexity, is proposed. SOM based clustering is an appropriate alternative to traditional clustering algorithms with different characteristic feature. Unlike traditional clustering algorithms, SOM clustering forms a semantic map where similar cluster centres are mapped close together and dissimilar ones apart. Depending on this feature of the SOM, both the outlier data problem encountered in the traditional clustering algorithms were eliminated, and the same results as SGRNN were achieved with much less training data.

5.3.1. SOM-GRNN

Several methods such as clustering and kNN have been used as a pre-processor for GRNN to reduce data complexity and determine significant training data. In addition to these methods, a hybrid SOM based GRNN algorithm that reduces the sample complexity considerably, is proposed in this study. Unlike traditional clustering algorithms, SOM clustering generates a semantic map where similar cluster centres are mapped close together and dissimilar ones apart. According to this feature, while much similar data are labelled with the same cluster index, less similar are labelled with the nearest neighbours' cluster indexes and dissimilar ones with distant clusters' indexes. Based on these advantages of SOM, not only the best matching cluster data but also less similar nearest clusters' data are taken into consideration while selecting the significant training data.

In SOM-GRNN, firstly SOM map is produced with training data to identify the clusters. Secondly, when the result of a new test datum is required to be predicted by using created map, its cluster and neighbours are detected. Thirdly, all

the training data belonging to the detected clusters are placed into the pattern layer of GRNN. At last, GRNN process is started for prediction.

Figure 5.7. illustrates how the SOM-GRNN works by using the most similar and less similar clusters for an applied test datum. In the given figure, $\vec{x}_{inp}$ is m-dimension test vector and its BMU is labelled with 5 and neighbours of BMU are labelled with 1,2,3,4,6,7,8, and 9. Training data x^{red} belonging to these clusters are placed into pattern layer of GRNN and $\vec{x}_{inp}$is applied to GRNN.

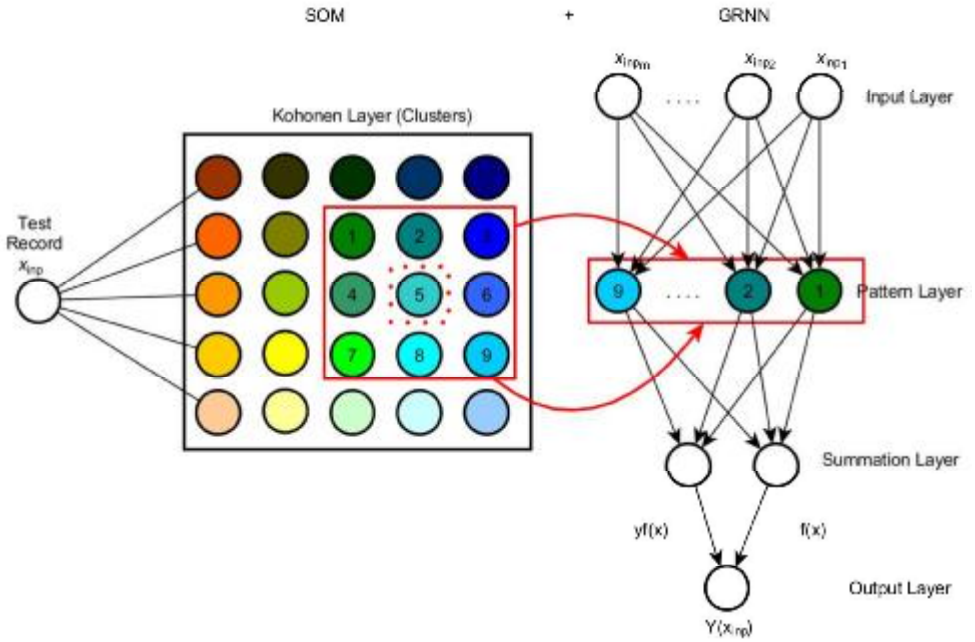


Figure 5.7. General overview of the SOM-GRNN

SOM map is an important feature for determining all the significant training data, especially for the test data which are near the cluster borders. Differences between selecting training data of only the most similar cluster and the most similar and similar clusters were illustrated over the Bird function by using an example test datum on Figure 5.8. and Figure 5.9., respectively.

In Figure 5.8., the use of training samples from only the most similar cluster of the test record is illustrated over Bird Function. On the other hand, in Figure 5.9., the use of training samples from both the most similar and similar clusters (its neighbouring clusters) is illustrated over the Bird Function for the same test record. While blue dots point out all training data, the green stars point out selected training data and red dot with a circle points out the test record.

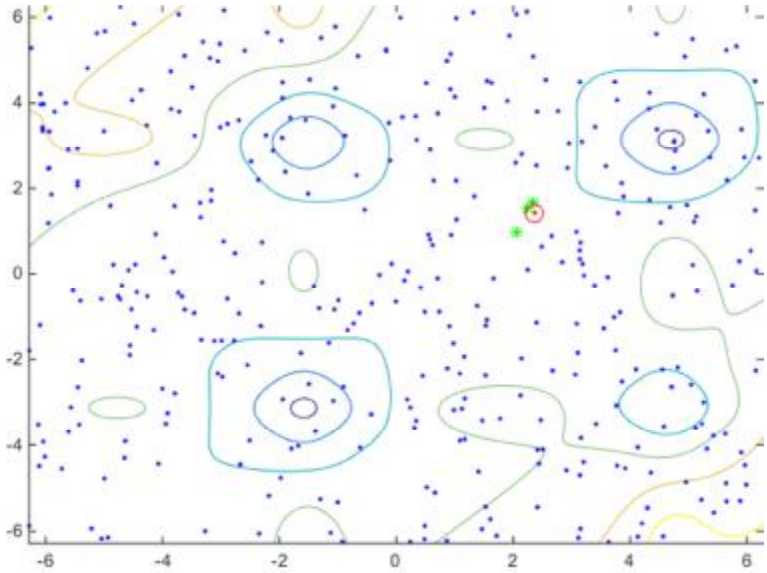


Figure 5.8. Training data of most similar cluster on the Bird function

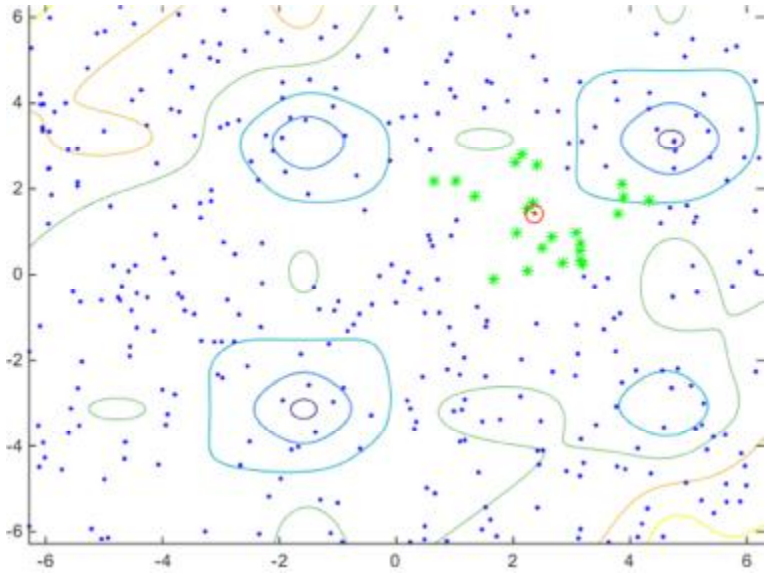


Figure 5.9. Training data of most similar and similar clusters on the Bird function

5.3.2. Algorithm

Let *x* denote *m*-dimensional training data, *N* be the number of training data, *y* contain target values of the training samples, *k* be the number of map units (clusters), x^{red}be *m*-dimensional chosen samples belonging to BMU and its neighbouring clusters, *R* be the number of reduced training data, *w* be the weight vectors, and $\vec{x}_{inp}$ *m*-dimensional test datum. Algorithm of SOM-GRNN is given below:

Algorithm 5.3. SOM-GRNN Algorithm

Data: x, $\vec{x}_{inp}$, y, k, σ, σ_0
Result: $Y(\vec{x}_{inp})$
Assign random values to w;
Apply steps of SOM to generate map with k units using x
Apply $\vec{x}_{inp}$ as test ;
While $i <= k$ do
    $distsom_i(\vec{x}_{inp}) = \left\|\vec{x}_{inp} - \vec{w}_i\right\|$;
end
$BMU = argmin_i(distsom)$;
Find the nearest neighbours of BMU;
Find training data x^{red} belonging to the nearest neighbours of BMU and BMU;
While $r <= R$ do
    Calculate squared Euclidean distance between $\vec{x}_{inp}$ and $\vec{x}_r^{red}$ as
    $dist_r = \sum_{i=1}^{m}\left(x_{inp,i} - x_{r,i}^{red}\right)^2$;
    Compute $nom = nom + y_r * e^{\left[\frac{-dist_r}{2\sigma^2}\right]}$;
    Compute $denom = denom + e^{\left[\frac{-dist_r}{2\sigma^2}\right]}$;
end
$Y(\vec{x}_{inp}) = \frac{nom}{denom}$;

5.3.3. Complexity Analysis

SOM-GRNN algorithm consists of two separate steps, SOM and GRNN. If we analyse complexity of the algorithm, both algorithms must be examined separately and summed at final step.

Suppose that N is the number of m-dimensional training data, k is the number of map units which is small enough to be ignored when compared to number of training data and I is number of iterations.

Computational complexity of the traditional SOM algorithm for each iteration will be $O(kNm)$ and total complexity of the SOM can be considered as $O(IkNm)$. The second step of the algorithm consists of GRNN, which is the one

pass learning algorithm and has *O(Nm)* computational complexity for prediction of one test datum. However, in SOM-GRNN number of training data is reduced from *N* to *R* where *R* << *N*. Consequently, SOM-GRNN algorithm has *O(IkNm)* + *O(Rm)* complexity, which equals to *O(IkNm)*.

However, SOM is a pre-processing algorithm and has to be performed before testing step. Hence, when a new test datum is applied, there remains two operations that are detection of nearest clusters and GRNN operation with reduced training data. This feature ensures faster prediction for the instantaneous test datum, and decrease in computational complexity from *O(Nm)* to *O(Rm)*, where *R* << *N*.

6. RESULTS AND DISCUSSIONS

In order to make more reliable analysis and increase the legibility, the results and discussions of each method are given separately. All methods are analysed in detail and compared with SGRNN and traditional hybrid GRNN algorithms. After detailed performance analysis are completed, contributions and advantages of proposed methods and comparisons of all the methods for both performance and complexity are summarized in the last title of this section.

6.1. Method 1 - A Cluster Based Approach to Reduce Pattern Layer Size for GRNN

Fifteen different benchmark test functions (Tang et al., 2008) and eight real datasets, which are given in section 3.8. and 3.9., were employed for analysing and comparing the performance of the SGRNN and the proposed k-means GRNN.

Before starting to a test, best sigma values were tuned for both SGRNN and k-means GRNN by utilizing 10-fold cross-validation technique. For benchmark test functions, sigma value was started at 0.1 and increased gradually up to 3 with 0.1 steps to find the most suitable value. However, for real datasets, since all the input features and targets were normalized into the range [0, 1], sigma was started at 0.01 and increased gradually up to 1 with 0.01 steps. During this process, errors for all σ values in each fold were calculated, then the total errors produced for the same σ values in 10 folds were compared. The value of σ, which provided the smallest total error, was accepted as the most appropriate σ value for that dataset. While there is only one sigma value for SGRNN, in k-means GRNN there are different sigma values for each cluster. Thus, after the clustering process was completed, sigma optimization procedure was applied for each cluster separately.

1000 random data were created and strewn over the search space for each test function. 10-fold cross validation was carried out for each configuration to

compare the prediction results obtained from all datasets. The UCI datasets and randomly created data for benchmark functions were separated to ten sub-groups. Nine sub-groups were used to train the models and the remaining sub-group was used as test data. This process was repeated ten times until all ten sub-groups were applied as test data. Tests were repeated with ten different random seed values for each benchmark function to decrease the effect of randomness on the results (Hamzacebi, 2008). Therefore, average error values were used to present the results of the benchmark functions. MSE, MAE and MAPE results of the SGRNN and hybrid GRNN with k-means clustering algorithm are given in Table 6.1., Table 6.2. and Table 6.3., respectively. Tables illustrate the obtained prediction error values and percentage of the error changes with respect to the SGRNN where, f_i points the i^{th} benchmark function and R_i points the i^{th} real dataset.

Table 6.1. MSE 10-fold cross validation performances

		Cluster Numbers			Deviation from SGRNN(%)		
Func.	**SGRNN**	**5**	**10**	**20**	**5**	**10**	**20**
f_1	1.09E+07	1.15E+07	1.16E+07	1.19E+07	-5.94	-6.46	-8.96
f_2	4.99E+01	5.24E+01	5.36E+01	5.73E+01	-4.96	-7.49	-14.88
f_3	1.13E+04	1.16E+04	1.20E+04	1.27E+04	-3.37	-6.37	-12.47
f_4	4.25E-01	4.43E-01	4.50E-01	4.65E-01	-4.20	-5.78	-9.28
f_5	1.44E+08	1.50E+08	1.55E+08	1.76E+08	-4.09	-7.9	-22.07
f_6	3.24E+04	3.46E+04	3.65E+04	4.00E+04	-6.78	-12.71	-23.52
f_7	2.67E+01	3.03E+01	3.41E+01	4.07E+01	-13.63	-27.78	-52.55
f_8	3.50E+02	3.65E+02	3.75E+02	4.34E+02	-4.28	-7.28	-24.17
f_9	3.15E+00	3.36E+00	3.59E+00	3.98E+00	-6.64	-14.01	-26.47
f_{10}	4.02E-04	4.83E-04	4.89E-04	6.23E-04	-19.94	-21.42	-54.75
f_{11}	4.34E+02	5.05E+02	5.97E+02	7.22E+02	-16.43	-37.56	-66.28
f_{12}	6.69E+02	7.45E+02	8.25E+02	9.22E+02	-11.39	-23.29	-37.90
f_{13}	6.98E-11	7.26E-11	7.38E-11	8.95E-11	-3.96	-5.70	-28.20
f_{14}	2.91E+02	3.13E+02	3.34E+02	3.65E+02	-7.76	-14.86	-25.53
f_{15}	1.13E+00	1.32E+00	1.44E+00	1.76E+00	-17.05	-27.75	-56.45
R_1	1.71E-02	1.84E-02	2.24E-02	2.15E-02	-7.90	-31.33	-26.05
R_2	9.05E-07	9.17E-07	9.22E-07	9.33E-07	-1.34	-1.88	-3.08
R_3	2.28E-03	2.35E-03	2.35E-03	2.36E-03	-2.92	-2.97	-3.31
R_4	2.76E-02	2.58E-02	2.53E-02	2.76E-02	6.44	8.45	0.11
R_5	3.65E-03	4.62E-03	5.09E-03	5.85E-03	-26.84	-39.63	-60.42
R_6	4.27E-03	4.89E-03	5.52E-03	6.56E-03	-14.54	-29.33	-53.83
R_7	1.81E-02	1.84E-02	1.94E-02	2.01E-02	-1.64	-6.83	-10.70
R_8	1.51E-02	1.54E-02	1.56E-02	1.57E-02	-2.31	-3.14	-4.17

Table 6.2. MAE 10-fold cross validation performances

		Cluster Numbers			Deviation from SGRNN(%)		
Func.	**SGRNN**	**5**	**10**	**20**	**5**	**10**	**20**
f_1	9.84E+02	1.04E+03	1.05E+03	1.10E+03	-5.55	-6.47	-12.03
f_2	5.51E+00	5.59E+00	5.64E+00	5.78E+00	-1.37	-2.28	-4.95
f_3	7.77E+01	7.89E+01	7.98E+01	8.15E+01	-1.57	-2.68	-4.93
f_4	5.04E-01	5.12E-01	5.13E-01	5.24E-01	-1.55	-1.83	-3.90
f_5	4.25E+03	4.40E+03	4.56E+03	4.92E+03	-3.44	-7.17	-15.58
f_6	8.65E+01	8.97E+01	9.28E+01	9.84E+01	-3.71	-7.27	-13.76
f_7	3.21E+00	3.41E+00	3.56E+00	3.82E+00	-6.12	-10.95	-18.91
f_8	8.37E+00	8.74E+00	9.25E+00	1.05E+01	-4.48	-10.51	-25.75
f_9	1.27E+00	1.31E+00	1.36E+00	1.42E+00	-2.89	-6.49	-11.37
f_{10}	1.28E-02	1.47E-02	1.51E-02	1.76E-02	-14.73	-17.64	-37.63
f_{11}	1.28E+01	1.38E+01	1.50E+01	1.62E+01	-7.99	-17.18	-27.03
f_{12}	1.63E+01	1.72E+01	1.84E+01	1.97E+01	-5.47	-12.68	-20.67
f_{13}	5.00E-06	5.05E-06	5.11E-06	5.70E-06	-1.15	-2.25	-14.05
f_{14}	9.28E+00	9.67E+00	1.02E+01	1.06E+01	-4.21	-9.45	-14.59
f_{15}	6.36E-01	6.94E-01	7.27E-01	8.00E-01	-9.03	-14.27	-25.74
R_1	1.03E-01	1.06E-01	1.13E-01	1.12E-01	-3.56	-10.07	-9.42
R_2	7.02E-04	7.12E-04	7.11E-04	7.09E-04	-1.39	-1.21	-1.00
R_3	3.45E-02	3.45E-02	3.49E-02	3.47E-02	-0.14	-1.06	-0.59
R_4	1.16E-01	1.12E-01	1.13E-01	1.13E-01	2.91	2.76	1.99
R_5	3.28E-02	4.05E-02	4.37E-02	4.83E-02	-23.48	-33.16	-47.35
R_6	3.98E-02	4.44E-02	4.83E-02	5.39E-02	-11.75	-21.47	-35.57
R_7	1.09E-01	1.08E-01	1.11E-01	1.11E-01	0.39	-2.25	-2.45
R_8	9.64E-02	9.62E-02	9.78E-02	9.80E-02	0.18	-1.48	-1.71

Table 6.3. MAPE 10-fold cross validation performances

		Cluster Numbers			Deviation from SGRNN(%)		
Func.	**SGRNN**	**5**	**10**	**20**	**5**	**10**	**20**
f_1	4.21E+01	4.24E+01	4.22E+01	4.43E+01	-0.76	-0.37	-5.32
f_2	2.50E+01	2.42E+01	2.52E+01	2.46E+01	3.23	-0.75	1.64
f_3	2.89E+02	3.00E+02	3.04E+02	2.99E+02	-3.79	-4.96	-3.32
f_4	2.86E+00	2.87E+00	2.88E+00	2.90E+00	-0.24	-0.57	-1.46
f_5	2.13E+01	2.23E+01	2.32E+01	2.46E+01	-4.88	-9.31	-15.93
f_6	3.26E+01	3.34E+01	3.60E+01	3.59E+01	-2.42	-10.58	-10.24
f_7	4.03E+01	4.51E+01	4.77E+01	5.43E+01	-11.94	-18.33	-34.56
f_8	2.04E+02	2.05E+02	2.06E+02	2.06E+02	-0.25	-0.76	-0.72
f_9	6.10E+00	6.32E+00	6.52E+00	6.74E+00	-3.59	-6.82	-10.55
f_{10}	4.10E+00	4.65E+00	4.86E+00	5.64E+00	-13.29	-18.49	-37.59
f_{11}	1.06E+01	9.36E+00	9.38E+00	9.58E+00	11.44	11.31	9.37
f_{12}	4.99E+02	4.92E+02	3.02E+02	4.54E+02	1.26	39.37	8.91
f_{13}	6.64E+00	6.71E+00	6.79E+00	7.60E+00	-1.07	-2.26	-14.47
f_{14}	1.48E+01	1.57E+01	1.71E+01	1.73E+01	-5.94	-15.47	-16.51
f_{15}	2.27E+01	2.11E+01	2.04E+01	1.92E+01	7.11	10.20	15.27
R_1	8.70E+02	9.85E+02	1.07E+03	1.08E+03	-13.29	-22.62	-24.48
R_2	3.99E+00	3.99E+00	3.99E+00	3.99E+00	-0.02	-0.02	-0.02
R_3	3.45E+02	3.45E+02	3.26E+02	3.45E+02	0.02	5.34	0.05
R_4	2.80E+03	2.67E+03	2.71E+03	3.29E+03	4.53	3.28	-17.58
R_5	1.73E+03	1.73E+03	1.73E+03	1.73E+03	-0.12	-0.17	-0.24
R_6	1.06E+03	1.06E+03	1.06E+03	1.11E+03	-0.18	-0.33	-4.77
R_7	2.84E+04	2.81E+04	2.83E+04	2.77E+04	1.23	0.41	2.71
R_8	1.68E+04	1.73E+04	1.68E+04	1.64E+04	-3.39	-0.21	2.18

The results show that, although the clustering algorithm greatly reduces the complexity of the pattern layer, the negative impact over all the test functions can be seen clearly. Increase in the error rate is proportional to the increase in the number of cluster count. When the k is increased from 5 to 10 and 20, the results

shows that the differences between the SGRNN and k-means-GRNN increase. If we calculate the average change in MSE rates for synthetic and real data sets, it can be seen that there are -8.69% and -6.38% decreases in prediction accuracy for 5 clustered GRNN. When we increase the cluster count to the 10, the average error percentage increase to -15.09% – -13.33% and, when the cluster count is 20, the average error percent reach to the -30.89% – -20.18 for synthetic and real datasets, respectively. Similar relation between the increase in error rate and cluster count can be observed in MAE and MAPE measurements.

In the second phase of the tests, the increasing errors are avoided by detecting the inner-outliers. The results of the new model, in which the k-means clustering algorithm is developed with the inner-outlier data detection algorithm and used as the pre-processor for the standard GRNN model are given in Table 6.4., Table 6.5. and Table 6.6.

Table 6.4. MSE 10-fold cross validation performances

Func.	SGRNN	Cluster Numbers and MAE			Deviation from SGRNN(%)		
		5	10	20	5	10	20
f_1	1.09E+07	1.15E+07	1.15E+07	1.14E+07	-5.86	-5.7	-4.87
f_2	4.99E+01	5.04E+01	4.98E+01	5.05E+01	-0.93	0.26	-1.13
f_3	1.13E+04	1.13E+04	1.14E+04	1.14E+04	-0.65	-1.01	-0.99
f_4	4.25E-01	4.19E-01	4.11E-01	4.14E-01	1.41	3.35	2.62
f_5	1.44E+08	1.47E+08	1.45E+08	1.48E+08	-1.78	-0.53	-2.92
f_6	3.24E+04	3.37E+04	3.45E+04	3.44E+04	-4.23	-6.49	-6.3
f_7	2.67E+01	2.80E+01	2.70E+01	2.80E+01	-4.81	-1.05	-4.77
f_8	3.50E+02	3.58E+02	3.55E+02	3.59E+02	-2.45	-1.46	-2.72
f_9	3.15E+00	3.21E+00	3.21E+00	3.20E+00	-2.00	-1.79	-1.48
f_{10}	4.02E-04	4.30E-04	4.17E-04	4.28E-04	-6.90	-3.58	-6.27
f_{11}	4.34E+02	4.51E+02	4.63E+02	4.65E+02	-3.97	-6.66	-7.02
f_{12}	6.69E+02	6.82E+02	6.93E+02	6.88E+02	-1.94	-3.60	-2.88
f_{13}	6.98E-11	7.11E-11	7.21E-11	7.18E-11	-1.75	-3.26	-2.74
f_{14}	2.91E+02	3.01E+02	3.07E+02	3.04E+02	-3.46	-5.61	-4.42
f_{15}	1.13E+00	1.19E+00	1.17E+00	1.20E+00	-6.08	-4.33	-6.96
R_1	1.71E-02	1.75E-02	2.13E-02	2.13E-02	-2.69	-24.63	-24.65
R_2	9.05E-07	9.15E-07	9.14E-07	9.28E-07	-1.10	-0.94	-2.48
R_3	2.28E-03	2.29E-03	2.28E-03	2.28E-03	-0.29	0.10	0.37
R_4	2.76E-02	2.55E-02	2.59E-02	2.76E-02	7.55	6.11	0.12
R_5	3.65E-03	4.02E-03	4.04E-03	4.60E-03	-10.21	-10.85	-26.09
R_6	4.27E-03	4.60E-03	4.88E-03	5.64E-03	-7.78	-14.33	-32.18
R_7	1.81E-02	1.81E-02	1.85E-02	1.91E-02	0.35	-2.09	-5.03
R_8	1.51E-02	1.52E-02	1.52E-02	1.54E-02	-0.44	-0.92	-2.22

Table 6.5. MAE 10-fold cross validation performances

Func.	SGRNN	Cluster Numbers and MAE			Deviation from SGRNN(%)		
		5	10	20	5	10	20
f_1	9.84E+02	1.03E+03	1.02E+03	1.03E+03	-5.12	-3.92	-4.59
f_2	5.51E+00	5.50E+00	5.44E+00	5.46E+00	0.28	1.28	0.83
f_3	7.77E+01	7.79E+01	7.80E+01	7.81E+01	-0.31	-0.41	-0.47
f_4	5.04E-01	5.03E-01	4.99E-01	5.03E-01	0.23	0.94	0.20
f_5	4.25E+03	4.30E+03	4.27E+03	4.30E+03	-1.05	-0.46	-1.04
f_6	8.65E+01	8.83E+01	8.92E+01	8.91E+01	-2.08	-3.14	-3.03
f_7	3.21E+00	3.27E+00	3.21E+00	3.24E+00	-1.73	0.07	-1.06
f_8	8.37E+00	8.49E+00	8.49E+00	8.58E+00	-1.42	-1.43	-2.54
f_9	1.27E+00	1.28E+00	1.28E+00	1.28E+00	-0.68	-0.65	-0.61
f_{10}	1.28E-02	1.34E-02	1.32E-02	1.34E-02	-4.48	-2.83	-4.54
f_{11}	1.28E+01	1.29E+01	1.31E+01	1.31E+01	-1.27	-2.12	-2.31
f_{12}	1.63E+01	1.64E+01	1.66E+01	1.66E+01	-0.71	-1.77	-1.79
f_{13}	5.00E-06	5.03E-06	5.03E-06	5.04E-06	-0.60	-0.80	-0.93
f_{14}	9.28E+00	9.43E+00	9.48E+00	9.45E+00	-1.67	-2.19	-1.87
f_{15}	6.36E-01	6.53E-01	6.47E-01	6.57E-01	-2.56	-1.73	-3.18
R_1	1.03E-01	1.03E-01	1.10E-01	1.11E-01	-0.93	-6.80	-8.60
R_2	7.02E-04	7.10E-04	7.07E-04	7.12E-04	-1.12	-0.72	-1.37
R_3	3.45E-02	3.42E-02	3.44E-02	3.42E-02	0.74	0.30	0.79
R_4	1.16E-01	1.13E-01	1.15E-01	1.16E-01	2.74	0.61	0.12
R_5	3.28E-02	3.71E-02	3.81E-02	4.08E-02	-13.03	-16.14	-24.37
R_6	3.98E-02	4.27E-02	4.57E-02	5.00E-02	-7.44	-14.84	-25.76
R_7	1.09E-01	1.08E-01	1.10E-01	1.12E-01	0.25	-1.15	-2.92
R_8	9.64E-02	9.67E-02	9.71E-02	9.76E-02	-0.33	-0.73	-1.27

Table 6.6. MAPE 10-fold cross validation performances

Func.	SGRNN	Cluster Numbers and MAPE			Deviation from SGRNN(%)		
		5	10	20	5	10	20
f_1	4.21E+01	4.14E+01	4.05E+01	4.09E+01	1.62	3.82	2.93
f_2	2.50E+01	2.39E+01	2.43E+01	2.35E+01	4.66	2.85	6.14
f_3	2.89E+02	2.91E+02	2.90E+02	2.90E+02	-0.65	-0.18	-0.26
f_4	2.86E+00	2.81E+00	2.78E+00	2.78E+00	1.88	2.88	2.81
f_5	2.13E+01	2.15E+01	2.14E+01	2.14E+01	-0.97	-0.54	-0.52
f_6	3.26E+01	3.30E+01	3.31E+01	3.15E+01	-1.30	-1.70	3.37
f_7	4.03E+01	4.27E+01	4.13E+01	4.18E+01	-5.96	-2.47	-3.75
f_8	2.04E+02	2.04E+02	2.04E+02	2.03E+02	0.00	0.11	0.51
f_9	6.10E+00	6.16E+00	6.14E+00	6.13E+00	-0.89	-0.62	-0.52
f_{10}	4.10E+00	4.26E+00	4.23E+00	4.23E+00	-3.92	-3.19	-3.17
f_{11}	1.06E+01	8.91E+00	8.41E+00	8.20E+00	15.74	20.49	22.42
f_{12}	4.99E+02	5.00E+02	3.94E+02	4.19E+02	-0.29	20.98	15.99
f_{13}	6.64E+00	6.68E+00	6.68E+00	6.69E+00	-0.59	-0.67	-0.78
f_{14}	1.48E+01	1.55E+01	1.57E+01	1.53E+01	-4.37	-5.60	-2.95
f_{15}	2.27E+01	2.04E+01	1.88E+01	1.73E+01	9.87	16.91	23.88
R_1	8.70E+02	9.65E+02	9.66E+02	9.83E+02	-10.93	-11.04	-12.99
R_2	3.99E+00	3.99E+00	3.99E+00	3.99E+00	-0.02	-0.01	-0.02
R_3	3.45E+02	3.45E+02	3.38E+02	3.45E+02	0.03	1.99	0.04
R_4	2.80E+03	2.67E+03	2.92E+03	2.71E+03	4.52	-4.38	3.09
R_5	1.73E+03	1.73E+03	1.78E+03	1.73E+03	-0.07	-2.60	-0.08
R_6	1.06E+03	1.06E+03	1.06E+03	1.07E+03	-0.10	-0.52	-0.68
R_7	2.84E+04	2.85E+04	2.83E+04	2.87E+04	-0.35	0.43	-1.03
R_8	1.68E+04	1.71E+04	1.69E+04	1.74E+04	-2.06	-0.91	-3.62

As in the previous result tables, the results of the proposed approach and percentages of change in error rates are given along with the results of the SGRNN algorithm. The results show that the proposed approach provides much better results than the pure k-means GRNN for three different evaluation criteria in all

test functions and real datasets. When the results are examined in detail, it is seen that the maximum error deviation for cluster count 5, 10 and 20 reduced from -26.84%, -39.6%, and -66.28% to -10.21%, -24.63%, and -32.18%, respectively. It is also seen that the algorithm produces very close results to that of standard GRNN for tested three different cluster counts and for some functions improved prediction results are obtained in terms of MAPE with much less training data.

As a result, as in the literature studies, when all the results are examined, it is seen that the k-means clustering algorithm considerably reduces the number of training data when it is used as a pre-processor of GRNN algorithm. However, accuracy of the results significantly reduced depending on this process.

On the other hand, when proposed inner-outlier detection algorithm is applied to the clustering algorithm, errors arisen from the use of the k-means clustering are reduced ~65%, ~82% and ~89% for benchmark test functions, and ~71%, ~55% and ~43% for real datasets in terms of MSE, when cluster counts are chosen as 5, 10, and 20 respectively. Error rate changes are calculated according to the (6.1.).

$$ERC = \frac{E_{k-means\ GRNN} - E_{outlier\ GRNN}}{E_{k-means\ GRNN}} * 100 \qquad (6.1.)$$

where, ERC points Error rate changes, $E_{k-means\ GRNN}$ and $E_{outlier\ GRNN}$ point the average error values for k-means GRNN and k-means GRNN with outlier detection algorithm, respectively.

In addition, ~95% data reduction is achieved in the datasets. Consequently, for both synthetic and real datasets, the major drawback of clustering algorithms on prediction performance was decreased significantly and almost the same prediction performances were achieved by using ~95% less training samples by means of the proposed algorithm.

6.2. Method 2 - New Pattern Reduction Method for Generalized Regression Neural Network

While kNN-GRNN was tested on both synthetic and real datasets, since the proposed algorithm is applicable for three-dimensional data, fifteen benchmark test functions containing three parameters (x1, x2 and y) were employed for ABNN-GRNN to compare performances of the proposed method with that of standard GRNN and kNN-GRNN.

Before starting to test, some parameters such as best sigma, k number and angle size have to be tuned to achieve the best prediction performances. Optimization of sigma was done as in the first algorithm, by 10-fold cross-validation technique. Unlike standard GRNN that whole training data are used in the pattern layer, in kNN and ABNN only k and 360/ θ number of training data are used in the pattern layer. However, as noted in the clustering algorithm, there is no convincingly acceptable solution to the problem of estimating the best k number. Therefore, three different k (5, 10, and 20) and three different angle (60, 40, and 20) parameters, reciprocal to 6, 9 and 18 nearest neighbours, were selected to demonstrate the prediction performances of the algorithms. Selected angle values were approximated to 5, 10, and 20 for kNN.

As in first approach, 1000 random data were created and strewn over the search space for each test function. 10-fold cross validation was carried out for each configuration to compare the prediction results obtained from these methods. Each test was repeated with 10 different random seeds to eliminate the effect of randomness and average results were presented. MSE, MAE and MAPE measurements are used to compare algorithm performance.

Average prediction errors for kNN-GRNN algorithm in terms of MSE, MAE and MAPE are presented in Table 6.7., Table 6.8. and Table 6.9, respectively.

Table 6.7. MSE 10-fold cross validation performances

Func.	SGRNN	KNN-GRNN (Neighbour Count)			Deviation from SGRNN(%)		
		k=5	k=10	k=20	k=5	k=10	k=20
f_1	1.09E+07	1.11E+07	1.09E+07	1.09E+07	-1.94	-0.31	-0.01
f_2	4.99E+01	4.76E+01	4.94E+01	4.99E+01	4.69	0.90	0.01
f_3	1.13E+04	1.13E+04	1.13E+04	1.13E+04	0.04	0	0
f_4	4.25E-01	4.52E-01	4.28E-01	4.24E-01	-6.33	-0.57	0.30
f_5	1.44E+08	1.48E+08	1.46E+08	1.44E+08	-2.82	-1.14	-0.06
f_6	3.24E+04	3.30E+04	3.24E+04	3.24E+04	-1.87	-0.16	0
f_7	2.67E+01	2.73E+01	2.65E+01	2.67E+01	-2.11	0.93	0.13
f_8	3.50E+02	2.64E+02	3.10E+02	3.41E+02	24.65	11.34	2.42
f_9	3.15E+00	3.22E+00	3.15E+00	3.15E+00	-2.17	-0.07	0.00
f_{10}	4.02E-04	2.40E-04	2.86E-04	3.51E-04	40.34	29.04	12.74
f_{11}	4.34E+02	5.11E+02	4.53E+02	4.36E+02	-17.64	-4.33	-0.40
f_{12}	6.69E+02	7.26E+02	6.81E+02	6.70E+02	-8.53	-1.82	-0.08
f_{13}	6.98E-11	6.90E-11	6.94E-11	6.98E-11	1.23	0.60	0.04
f_{14}	2.91E+02	3.00E+02	2.92E+02	2.91E+02	-3.10	-0.34	0.00
f_{15}	1.13E+00	1.27E+00	1.16E+00	1.13E+00	-12.69	-3.33	-0.20
R_1	1.71E-02	1.89E-02	1.79E-02	1.70E-02	-10.50	-4.75	0.38
R_2	9.05E-07	9.13E-07	8.88E-07	8.86E-07	-0.80	1.88	2.13
R_3	2.28E-03	2.25E-03	2.23E-03	2.25E-03	1.37	2.27	1.54
R_4	2.76E-02	2.59E-02	2.63E-02	2.67E-02	6.19	4.73	3.18
R_5	3.65E-03	3.55E-03	3.59E-03	3.61E-03	2.50	1.42	1.05
R_6	4.27E-03	4.19E-03	4.21E-03	4.22E-03	1.78	1.35	1.04
R_7	1.81E-02	2.09E-02	1.93E-02	1.85E-02	-15.08	-6.63	-1.92
R_8	1.51E-02	1.69E-02	1.58E-02	1.53E-02	-12.18	-5.06	-1.15

Table 6.8. MAE 10-fold cross validation performances

Func.	SGRNN	KNN-GRNN (Neighbour Count)			Deviation from SGRNN(%)		
		k=5	k=10	k=20	k=5	k=10	k=20
f_1	9.84E+02	1.01E+03	9.88E+02	9.84E+02	-2.97	-0.41	-0.01
f_2	5.51E+00	5.36E+00	5.48E+00	5.51E+00	2.78	0.51	0.01
f_3	7.77E+01	7.77E+01	7.77E+01	7.77E+01	0.02	0.00	0.00
f_4	5.04E-01	5.23E-01	5.09E-01	5.04E-01	-3.83	-0.92	-0.03
f_5	4.25E+03	4.42E+03	4.29E+03	4.25E+03	-4.02	-0.83	-0.02
f_6	8.65E+01	8.83E+01	8.67E+01	8.65E+01	-2.07	-0.20	0.00
f_7	3.21E+00	3.26E+00	3.20E+00	3.21E+00	-1.64	0.28	0.05
f_8	8.37E+00	7.39E+00	7.74E+00	8.13E+00	11.75	7.53	2.90
f_9	1.27E+00	1.29E+00	1.27E+00	1.27E+00	-1.25	-0.04	0.00
f_{10}	1.28E-02	1.00E-02	1.05E-02	1.15E-02	21.60	17.83	9.76
f_{11}	1.28E+01	1.41E+01	1.31E+01	1.28E+01	-10.70	-2.87	-0.24
f_{12}	1.63E+01	1.74E+01	1.65E+01	1.63E+01	-6.81	-1.43	-0.03
f_{13}	5.00E-06	4.86E-06	4.95E-06	4.99E-06	2.62	0.95	0.07
f_{14}	9.28E+00	9.51E+00	9.30E+00	9.28E+00	-2.49	-0.28	0.00
f_{15}	6.36E-01	6.84E-01	6.48E-01	6.37E-01	-7.52	-1.89	-0.09
R_1	1.03E-01	1.05E-01	1.04E-01	1.02E-01	-2.85	-1.64	0.98
R_2	7.02E-04	7.01E-04	6.93E-04	6.93E-04	0.15	1.40	1.40
R_3	3.45E-02	3.31E-02	3.33E-02	3.38E-02	4.02	3.35	1.95
R_4	1.16E-01	1.08E-01	1.10E-01	1.13E-01	6.90	4.71	2.57
R_5	3.28E-02	3.16E-02	3.24E-02	3.25E-02	3.67	1.12	1.05
R_6	3.98E-02	3.87E-02	3.92E-02	3.94E-02	2.57	1.39	1.04
R_7	1.09E-01	1.10E-01	1.08E-01	1.07E-01	-1.53	0.64	1.81
R_8	9.64E-02	9.92E-02	9.72E-02	9.60E-02	-2.96	-0.85	0.40

Table 6.9. MAPE 10-fold cross validation performances

Func.	SGRNN	KNN-GRNN (Neighbour Count)			Deviation from SGRNN(%)		
		k=5	k=10	k=20	k=5	k=10	k=20
f_1	4.21E+01	3.64E+01	4.04E+01	4.20E+01	13.55	3.93	0.21
f_2	2.50E+01	2.36E+01	2.44E+01	2.50E+01	5.66	2.42	0.08
f_3	2.89E+02	2.89E+02	2.89E+02	2.89E+02	0.10	0.00	0.00
f_4	2.86E+00	2.89E+00	2.85E+00	2.85E+00	-0.80	0.55	0.39
f_5	2.13E+01	1.91E+01	2.03E+01	2.11E+01	9.95	4.47	0.56
f_6	3.26E+01	3.22E+01	3.28E+01	3.26E+01	1.13	-0.80	0.04
f_7	4.03E+01	4.18E+01	4.11E+01	4.03E+01	-3.63	-2.06	0.04
f_8	2.04E+02	7.46E+01	1.34E+02	1.78E+02	63.49	34.36	12.86
f_9	6.10E+00	6.16E+00	6.10E+00	6.10E+00	-0.98	0.03	0.00
f_{10}	4.10E+00	3.01E+00	3.15E+00	3.54E+00	26.63	23.17	13.68
f_{11}	1.06E+01	8.36E+00	9.54E+00	1.04E+01	20.98	9.77	1.61
f_{12}	4.99E+02	3.91E+02	4.75E+02	4.98E+02	21.60	4.73	0.24
f_{13}	6.64E+00	6.43E+00	6.56E+00	6.63E+00	3.10	1.13	0.09
f_{14}	1.48E+01	1.42E+01	1.47E+01	1.48E+01	4.13	0.82	0.03
f_{15}	2.27E+01	1.71E+01	1.96E+01	2.25E+01	24.68	13.33	0.86
R_1	8.70E+02	4.20E+02	6.37E+02	8.07E+02	51.76	26.76	7.20
R_2	3.99E+00	3.98E+00	3.98E+00	3.98E+00	0.08	0.10	0.10
R_3	3.45E+02	3.35E+02	3.42E+02	3.43E+02	2.94	0.91	0.53
R_4	2.80E+03	2.83E+03	2.79E+03	2.80E+03	-1.01	0.37	0.07
R_5	1.73E+03	1.71E+03	1.71E+03	1.71E+03	1.06	1.04	1.04
R_6	1.06E+03	1.05E+03	1.05E+03	1.05E+03	1.08	1.04	1.04
R_7	2.84E+04	2.75E+04	2.81E+04	2.80E+04	3.29	1.15	1.51
R_8	1.68E+04	1.69E+04	1.68E+04	1.66E+04	-0.7	-0.26	0.83

According to the given tables for kNN-GRNN, 5-NN provided better or equal prediction performances than that of SGRNN for 5 test functions with only 5 training data. On the other hand, negative impact of the 5-NN is obviously seen

over the remaining ten test functions. Thus, general performances of the 5-NN were not enough to improve overall prediction accuracy. When the k was increased to 10 and 20, the results shows that the differences between the SGRNN and kNN-GRNN decreased. Even though there are not substantial improvements on the results, it can be seen that almost the same prediction performances were obtained by using 20 training data which is the 2.2% of all training data.

The comparative results for ABNN -GRNN and SGRNN are listed in Table 6.10., Table 6.11. and Table 6.12. Results indicate that the prediction results of ABNN-GRNN is better than kNN-GRNN and SGRNN with respect to MSE, MAE and MAPE measurements for all given angles. Proposed pre-processing algorithm produced better prediction performance for 13 test functions and almost equal prediction performance for only 2 function by using less training data. When we calculated the average change in error rates, it is seen that ABNN-GRNN provided ~22%, ~18.4% and ~11.5% improvements in prediction performance in terms of MSE for angle 60, 40 and 20, respectively. Additionally, according to the test results, the ABNN provides the best performance for angle 60 (6 neighbours). If the tables are examined in detail, in function 10, 68.58% increase in prediction performance was provided in terms of MSE by using ~99% (6 in 900) less number of training data. Also, 53.37% and 52.79% performance increment were obtained for function 8 and 10 with respect to MAPE, respectively. Moreover, by using ABNN, performances were improved in the range of 14.46-68.58%, 10.73-62.43% and 5.67%-45.44% for 10 different benchmark functions for 60, 40 and 20 angles, respectively.

Table 6.10. MSE 10-fold cross validation performances

Func.	SGRNN	ABNN-GRNN (Angle Range)			Deviation from SGRNN(%)		
		Θ=60	Θ=40	Θ=20	Θ=60	Θ=40	Θ=20
f_1	1.09E+07	9.02E+06	9.62E+06	1.01E+07	17.20	11.69	7.22
f_2	4.99E+01	4.39E+01	4.54E+01	4.75E+01	12.00	9.07	4.76
f_3	1.13E+04	1.13E+04	1.13E+04	1.13E+04	-0.09	-0.04	-0.03
f_4	4.25E-01	4.36E-01	4.29E-01	4.23E-01	-2.47	-0.74	0.57
f_5	1.44E+08	1.07E+08	1.14E+08	1.25E+08	25.82	20.6	13.41
f_6	3.24E+04	2.78E+04	2.88E+04	3.04E+04	14.03	10.96	6.08
f_7	2.67E+01	1.95E+01	2.11E+01	2.35E+01	26.89	21.05	11.84
f_8	3.50E+02	1.64E+02	1.87E+02	2.41E+02	53.03	46.63	31.03
f_9	3.15E+00	2.98E+00	3.03E+00	3.09E+00	5.48	3.76	1.81
f_{10}	4.02E-04	1.26E-04	1.51E-04	2.20E-04	68.58	62.43	45.44
f_{11}	4.34E+02	2.87E+02	3.13E+02	3.62E+02	33.81	27.96	16.53
f_{12}	6.69E+02	4.79E+02	5.14E+02	5.79E+02	28.36	23.20	13.42
f_{13}	6.98E-11	6.77E-11	6.76E-11	6.86E-11	3.11	3.27	1.79
f_{14}	2.91E+02	2.49E+02	2.60E+02	2.74E+02	14.46	10.73	5.67
f_{15}	1.13E+00	7.74E-01	8.40E-01	9.64E-01	31.22	25.40	14.38

Table 6.11. MAE 10-fold cross validation performances

Func.	SGRNN	ABNN-GRNN (Angle Range)			Deviation from SGRNN(%)		
		Θ=60	Θ=40	Θ=20	Θ=60	Θ=40	Θ=20
f_1	9.84E+02	8.48E+02	8.87E+02	9.30E+02	13.83	9.88	5.50
f_2	5.51E+00	5.06E+00	5.18E+00	5.34E+00	8.24	6.05	3.16
f_3	7.77E+01	7.77E+01	7.77E+01	7.77E+01	-0.05	-0.02	-0.01
f_4	5.04E-01	5.16E-01	5.11E-01	5.06E-01	-2.46	-1.38	-0.30
f_5	4.25E+03	3.45E+03	3.61E+03	3.89E+03	18.87	15.14	8.57
f_6	8.65E+01	7.67E+01	7.89E+01	8.24E+01	11.34	8.82	4.68
f_7	3.21E+00	2.64E+00	2.77E+00	2.98E+00	17.91	13.67	7.32
f_8	8.37E+00	5.12E+00	5.54E+00	6.55E+00	38.82	33.83	21.76
f_9	1.27E+00	1.23E+00	1.24E+00	1.26E+00	3.66	2.43	1.11
f_{10}	1.28E-02	6.33E-03	6.93E-03	8.74E-03	50.53	45.86	31.66
f_{11}	1.28E+01	9.70E+00	1.04E+01	1.14E+01	24.09	18.97	10.49
f_{12}	1.63E+01	1.28E+01	1.36E+01	1.48E+01	21.12	16.41	8.96
f_{13}	5.00E-06	4.74E-06	4.78E-06	4.88E-06	5.02	4.24	2.37
f_{14}	9.28E+00	8.21E+00	8.47E+00	8.85E+00	11.53	8.70	4.55
f_{15}	6.36E-01	4.98E-01	5.25E-01	5.75E-01	21.79	17.50	9.66

Table 6.12. MAPE 10-fold cross validation performances

Func.	SGRNN	ABNN-GRNN (Angle Range)			Deviation from SGRNN(%)		
		Θ=60	Θ=40	Θ=20	Θ=60	Θ=40	Θ=20
f_1	4.21E+01	3.30E+01	3.58E+01	3.85E+01	21.63	15.03	8.42
f_2	2.50E+01	2.29E+01	2.35E+01	2.43E+01	8.55	6.19	2.79
f_3	2.89E+02	2.89E+02	2.89E+02	2.89E+02	0.02	0.00	-0.02
f_4	2.86E+00	2.84E+00	2.83E+00	2.83E+00	0.80	1.30	1.23
f_5	2.13E+01	1.63E+01	1.75E+01	1.92E+01	23.36	17.66	9.51
f_6	3.26E+01	2.82E+01	3.04E+01	3.21E+01	13.46	6.64	1.37
f_7	4.03E+01	3.19E+01	3.38E+01	3.65E+01	20.94	16.15	9.39
f_8	2.04E+02	9.53E+01	1.17E+02	1.48E+02	53.37	42.80	27.71
f_9	6.10E+00	5.97E+00	6.01E+00	6.05E+00	2.15	1.45	0.77
f_{10}	4.10E+00	1.94E+00	2.15E+00	2.74E+00	52.79	47.66	33.13
f_{11}	1.06E+01	7.53E+00	8.30E+00	9.29E+00	28.76	21.50	12.11
f_{12}	4.99E+02	3.34E+02	3.70E+02	4.33E+02	33.03	25.76	13.19
f_{13}	6.64E+00	6.25E+00	6.32E+00	6.46E+00	5.80	4.82	2.67
f_{14}	1.48E+01	1.27E+01	1.33E+01	1.41E+01	14.75	10.06	5.28
f_{15}	2.27E+01	1.68E+01	1.82E+01	2.07E+01	25.91	19.75	8.47

As a result, both pre-processing algorithms decreased the number of training data nearly 99%. While prediction accuracies of SGRNN and kNN-GRNN were almost the same with respect to MAPE, MSE and MAE, the prediction accuracy of the ABNN -GRNN was superior to both in all error measures. Consequently, test results indicate that the proposed method provides two major improvements to standard GRNN. While it improves the prediction accuracy; it also reduces the number of required train data. In addition to this result, by selection of the suitable angle, smaller train datasets and better prediction performances may be achieved.

6.3. Method 3 - Pattern Layer Reduction for Generalized Regression Neural Networks by Using Self Organizing Map

As in evaluation of k-means GRNN, fifteen different benchmark test functions and eight real datasets given in section 3.7. and 3.8., were employed for analysing and comparing the performances of the proposed SOM-GRNN algorithm and SGRNN.

Before starting the tests, there exist two parameters; best sigma, and SOM size; have to be tuned for SOM-GRNN. Sigma optimization was accomplished by ten-fold cross validation, as in the previous tests. Since SOM GRNN's training data are determined in accordance with the test datum during the test process, their optimal sigma values cannot be obtained until test operation starts. Sigma searching process at the test step causes extra operation costs and decreases in speed. Hence, optimal sigma value was tuned for only SGRNN. The optimal sigma value that was calculated for SGRNN is, also, used for SOM-GRNN. Additionally, if we intended to carry out similar optimization procedure for SOM size, huge number of tests would be required and it would take quite a long time. Therefore, depending on the preliminary works, different number of parameters were selected and employed for each pre-processing algorithms. According to the preliminary works three different SOM sizes (8, 10, 16) were utilized to demonstrate the prediction performances of the algorithms.

The comparative results for SOM-GRNN and SGRNN are listed in Table 6.13., Table 6.14 and Table 6.15. It can be clearly seen that SOM-GRNNs with size of 8x8 and 10x10 produce either better or the same results as SGRNN for all given test functions and very low differences for real-world datasets in three error measurements, MSE, MAE and MAPE.

Table 6.13. MSE 10-fold cross validation performances

		SOM size			Deviation from SGRNN(%)		
Func.	**SGRNN**	**8x8**	**10x10**	**16x16**	**8x8**	**10x10**	**16x16**
f_1	1.09E+07	1.09E+07	1.09E+07	1.09E+07	0.00	0.00	-0.01
f_2	4.99E+01	4.99E+01	4.99E+01	4.99E+01	0.00	0.00	0.02
f_3	1.13E+04	1.13E+04	1.13E+04	1.13E+04	0	0	0
f_4	4.25E-01	4.25E-01	4.25E-01	4.25E-01	0.00	0.02	0.10
f_5	1.44E+08	1.44E+08	1.44E+08	1.44E+08	0	0	-0.07
f_6	3.24E+04	3.24E+04	3.24E+04	3.24E+04	0	0	-0.01
f_7	2.67E+01	2.67E+01	2.67E+01	2.67E+01	0.00	0.00	0.06
f_8	3.50E+02	3.50E+02	3.50E+02	3.48E+02	0.00	0.02	0.57
f_9	3.15E+00	3.15E+00	3.15E+00	3.15E+00	0.00	0.00	0.02
f_{10}	4.02E-04	4.02E-04	4.01E-04	3.82E-04	0.07	0.37	5.18
f_{11}	4.34E+02	4.34E+02	4.34E+02	4.36E+02	0.00	0.00	-0.38
f_{12}	6.69E+02	6.69E+02	6.69E+02	6.70E+02	0.00	0.00	-0.17
f_{13}	6.98E-11	6.98E-11	6.98E-11	6.98E-11	0.00	0.00	0.01
f_{14}	2.91E+02	2.91E+02	2.91E+02	2.91E+02	0.00	0.00	-0.01
f_{15}	1.13E+00	1.13E+00	1.13E+00	1.13E+00	0.00	0.00	-0.38
R_1	1.71E-02	1.72E-02	1.71E-02	1.82E-02	-0.45	-0.11	-6.36
R_2	9.05E-07	9.05E-07	9.05E-07	9.09E-07	0.00	-0.02	-0.38
R_3	2.28E-03	2.29E-03	2.29E-03	2.29E-03	0.00	-0.01	0.00
R_4	2.76E-02	2.76E-02	2.75E-02	2.73E-02	-0.07	0.30	1.15
R_5	3.65E-03	3.65E-03	3.65E-03	3.66E-03	-0.01	-0.03	-0.39
R_6	4.27E-03	4.27E-03	4.27E-03	4.27E-03	0.00	-0.02	-0.01
R_7	1.81E-02	1.82E-02	1.81E-02	1.83E-02	-0.55	0.08	-1.09
R_8	1.51E-02	1.51E-02	1.51E-02	1.52E-02	-0.16	-0.36	-0.82

Table 6.14. MAE 10-fold cross validation performances

		SOM size			Deviation from SGRNN(%)		
Func.	**SGRNN**	**8x8**	**10x10**	**16x16**	**8x8**	**10x10**	**16x16**
f_1	9.84E+02	9.84E+02	9.84E+02	9.85E+02	0.00	0.00	-0.02
f_2	5.51E+00	5.51E+00	5.51E+00	5.51E+00	0.00	0.00	0.01
f_3	7.77E+01	7.77E+01	7.77E+01	7.77E+01	0.00	0.00	0.00
f_4	5.04E-01	5.04E-01	5.04E-01	5.04E-01	0.00	0.00	-0.09
f_5	4.25E+03	4.25E+03	4.25E+03	4.26E+03	0.00	0.00	-0.07
f_6	8.65E+01	8.65E+01	8.65E+01	8.65E+01	0.00	0.00	-0.01
f_7	3.21E+00	3.21E+00	3.21E+00	3.21E+00	0.00	0.00	0.01
f_8	8.37E+00	8.37E+00	8.36E+00	8.28E+00	0.01	0.10	1.07
f_9	1.27E+00	1.27E+00	1.27E+00	1.27E+00	0.00	0.00	0.01
f_{10}	1.28E-02	1.28E-02	1.27E-02	1.22E-02	0.11	0.53	4.37
f_{11}	1.28E+01	1.28E+01	1.28E+01	1.28E+01	0.00	0.00	-0.21
f_{12}	1.63E+01	1.63E+01	1.63E+01	1.63E+01	0.00	0.00	-0.11
f_{13}	5.00E-06	5.00E-06	5.00E-06	4.99E-06	0.00	0.00	0.04
f_{14}	9.28E+00	9.28E+00	9.28E+00	9.28E+00	0.00	0.00	-0.02
f_{15}	6.36E-01	6.36E-01	6.36E-01	6.38E-01	0.00	0.00	-0.23
R_1	1.03E-01	1.03E-01	1.03E-01	1.04E-01	-0.13	0.02	-1.23
R_2	7.02E-04	7.02E-04	7.02E-04	7.04E-04	0.01	0.00	-0.18
R_3	3.45E-02	3.45E-02	3.45E-02	3.45E-02	0.00	0.00	0.03
R_4	1.16E-01	1.16E-01	1.15E-01	1.15E-01	0.06	0.27	0.91
R_5	3.28E-02	3.28E-02	3.28E-02	3.29E-02	-0.02	-0.06	-0.27
R_6	3.98E-02	3.98E-02	3.98E-02	3.98E-02	-0.03	-0.04	-0.09
R_7	1.09E-01	1.08E-01	1.08E-01	1.07E-01	0.33	0.72	1.02
R_8	9.64E-02	9.63E-02	9.65E-02	9.64E-02	0.08	-0.09	-0.03

Table 6.15. Clustering - GRNN results with 20 cluster (MAPE)

		SOM size			Deviation from SGRNN(%)		
Func.	**SGRNN**	**8x8**	**10x10**	**16x16**	**8x8**	**10x10**	**16x16**
f_1	4.21E+01	4.21E+01	4.21E+01	4.20E+01	0.00	0.00	0.25
f_2	2.50E+01	2.50E+01	2.50E+01	2.50E+01	0.00	0.00	0.23
f_3	2.89E+02	2.89E+02	2.89E+02	2.89E+02	0.00	0.00	0.00
f_4	2.86E+00	2.86E+00	2.86E+00	2.86E+00	0.00	0.03	0.21
f_5	2.13E+01	2.13E+01	2.13E+01	2.12E+01	0.00	0.00	0.39
f_6	3.26E+01	3.26E+01	3.26E+01	3.26E+01	0.00	0.00	-0.13
f_7	4.03E+01	4.03E+01	4.03E+01	4.04E+01	0.00	0.00	-0.09
f_8	2.04E+02	2.04E+02	2.04E+02	1.93E+02	0.01	0.22	5.38
f_9	6.10E+00	6.10E+00	6.10E+00	6.10E+00	0.00	0.00	0.02
f_{10}	4.10E+00	4.09E+00	4.06E+00	3.78E+00	0.21	1.06	7.83
f_{11}	1.06E+01	1.06E+01	1.06E+01	1.04E+01	0.00	0.05	1.31
f_{12}	4.99E+02	4.99E+02	4.99E+02	4.97E+02	0.00	0.00	0.33
f_{13}	6.64E+00	6.64E+00	6.64E+00	6.64E+00	0.00	0.00	0.05
f_{14}	1.48E+01	1.48E+01	1.48E+01	1.48E+01	0.00	0.00	0.04
f_{15}	2.27E+01	2.27E+01	2.27E+01	2.15E+01	0.00	0.01	5.32
R_1	8.70E+02	8.70E+02	8.70E+02	8.70E+02	0.00	0.00	0.00
R_2	3.99E+00	3.99E+00	3.99E+00	3.99E+00	0.00	0.00	0.00
R_3	3.45E+02	3.45E+02	3.45E+02	3.45E+02	0.00	0.00	0.00
R_4	2.80E+03	2.80E+03	2.80E+03	2.80E+03	0.01	0.00	0.01
R_5	1.73E+03	1.73E+03	1.73E+03	1.73E+03	0.00	0.00	0.00
R_6	1.06E+03	1.06E+03	1.06E+03	1.06E+03	0.00	0.00	0.00
R_7	2.84E+04	2.83E+04	2.82E+04	2.79E+04	0.53	0.91	1.89
R_8	1.68E+04	1.67E+04	1.67E+04	1.67E+04	0.36	0.44	0.6

According to the test results, when the SOM size is selected as 8 and 10, algorithm produces almost the same results as SGRNN and, when the SOM size increased to 16, slight difference on the performance starts to be observed depending on the reduction of training data. In addition to this, SOM-GRNN

reduces the number of training data for test functions 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 cluster counts. Hence, total number of utilized training data were counted while prediction operation was executed, then average value was calculated. If it is assumed that the training data were dispersed homogenous over the clusters, then the Equation (6.2.) can be used to calculate the number of utilized training data. If the given equation is used, it may be seen that almost the same result with the given counts will be obtained.

$$R = \frac{N}{k} * 7 \tag{6.2.}$$

where N and k are total training data count and total cluster count, respectively. R is the number of utilized training data count and 7 is used as cluster count (1 most similar and 6 similar clusters).

When the performances of pre-processing algorithms mentioned in this thesis are compared, kmeans-GRNN requires 80%, 90% and 95% less number of training data for 5, 10 and 20 cluster counts, respectively. It was thought that when a specific sigma value for each cluster and only the most similar training data to the test datum were used together, the prediction error rate would decrease compared to SGRNN. However, the results given in the Table 6.1., Table 6.2. and Table 6.3. indicate that using k-means clustering method as pre-processor made conventional GRNN algorithm's performance sensitive to the newly emerged inner outliers. Even though this problem has been significantly reduced with the proposed outlier detection algorithm, it could not be eliminated completely.

On the other hand, kNN-GRNN, which is used as an alternative to clustering method, employed 5, 10 and 20 nearest neighbours instead of all training data. According to the result of the kNN-GRNN, for k=5 algorithm provides better or equal prediction performances for four test functions (2,3, 8, 13) and three UCI

datasets in terms of MSE by using only five training data. When k is increased to 10 and 20, the results indicate that the differences between the performances of SGRNN and kNN-GRNN decrease. Although, there is significant reduction on the number of training data, there is no consistency and reliability in the achieved results. Additionally, kNN requires O(N) calculations for distance measurements and O(NlogN) additional calculations for sorting and selection of nearest k training data for each test datum separately. By taking into these considerations, we can conclude that kNN-GRNN's performance is not good enough to improve overall prediction accuracy and speed.

In accordance with the comparison it can be said that proposed pre-processing algorithm, SOM-GRNN, produces more consistent and reliable results than clustering-GRNN and kNN-GRNN algorithms. Moreover, if SOM-GRNN is examined in terms of transaction cost, as in k-means clustering algorithm, most of the required calculations for SOM algorithm have done only once and before the prediction operations. It only needs determination of the nearest clusters among cluster centres and requires much less calculation than that of kNN method.

6.4. Summary and Comparisons

To investigate the performance of three proposed methods (k-means GRNN with outlier detection, ABNN-GRNN and SOM-GRNN) and compare with SGRNN, kNN-GRNN and pure k-means GRNN, 15 real valued well known benchmark test functions and 8 UCI datasets were employed. All the methods were coded in MATLAB 2014b Environment.1000 random data were created and strewn over the search space for test functions. 10-fold cross validation was carried out for both test functions and datasets. Tests were repeated with 10 different random seed values for all benchmark functions to decrease the effect of randomness on the results. Therefore, average error values were used to present the efficiency of predictions. Three standard error measures, MSE, MAE and MAPE, were used to evaluate the performance of the methods.

Figure 6.1. illustrates the percentages of the amount of data used by the algorithms for test functions and real world datasets. While the SGRNN shown in blue use entire of the dataset (100%), GRNN algorithms cooperated with pre-processing algorithms (shown with the remaining colour) use a very small part of the datasets. In clustering algorithms, because of the data are always divided into the same number of pieces, ratios of the used data become same for all datasets. On the other hand, since the k-NN algorithm and the ABNN algorithms use the same number of data in all datasets for certain k and θ values, the rate of data used change depending on the size of the dataset.

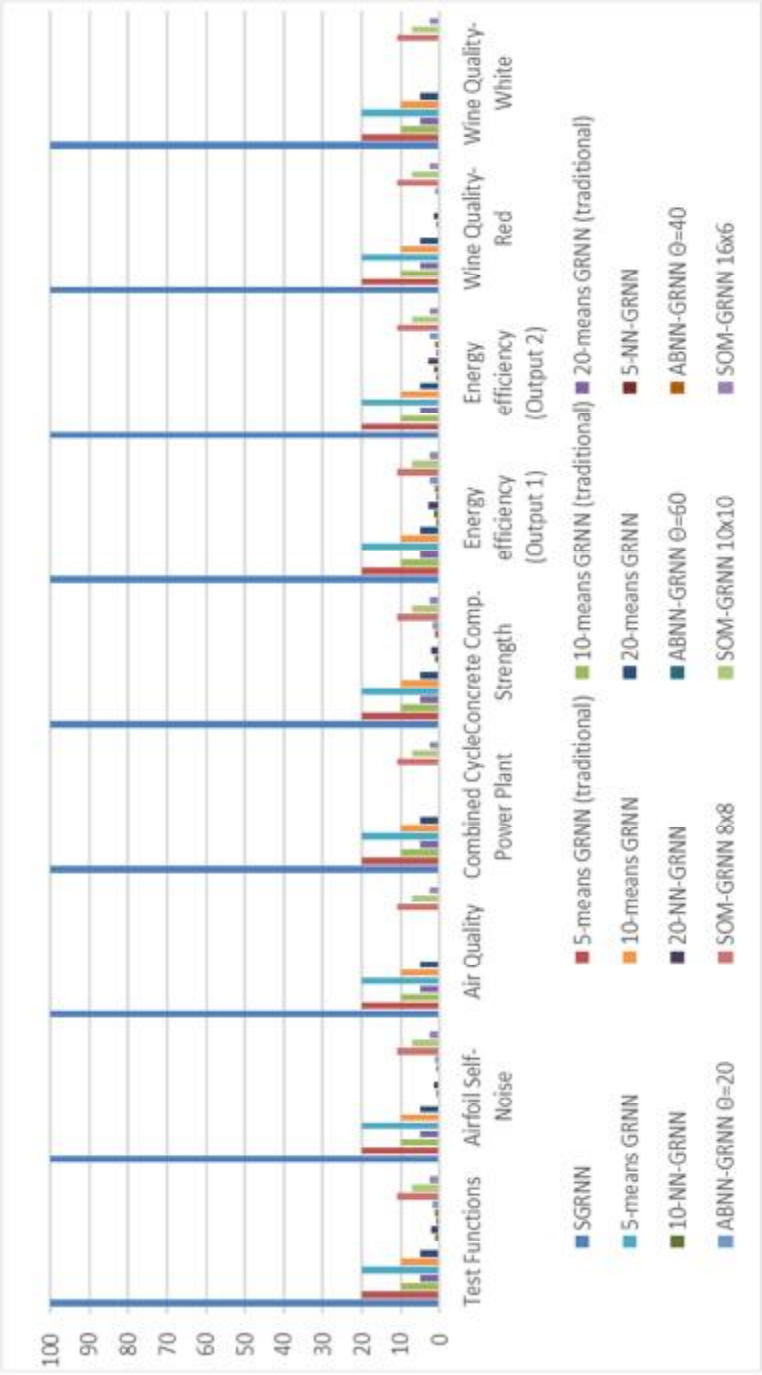


Figure 6.1. The percentages of the amount of data used by the algorithms for test functions and real world datasets

Before starting the tests, sigma value was tuned by using 10-fold cross validation. While there is only one sigma value for SGRNN, in k-means GRNN there are different sigma values for each cluster. Therefore, after the clustering process was completed, sigma optimization procedure was applied for each cluster separately. Since in kNN-GRNN, ABNN-GRNN and SOM-GRNN training data are determined in accordance with the test datum during the test process, their optimal sigma values cannot be obtained until test operation starts. Sigma searching process at the test step causes extra operation costs and decreases in speed. Moreover, in kNN and ABNN due to the small number of training data, the process of searching the optimal sigma value may not provide reliable results. Hence, the optimal sigma value that was calculated for SGRNN was also used for kNN-GRNN, ABNN-GRNN and SOM-GRNN. Additionally, if we intended to carry out similar optimization procedure for cluster count, nearest neighbour count, angle size and SOM size, huge number of tests would be required and it would take quite a long time. Therefore, depending on the preliminary works, three different values were selected and employed for each pre-processing algorithms.

After adjusting the parameters, three different pre-processing methods were examined to observe their effects on SGRNN. It has been observed that while all the strategies reduce the data complexity meaningfully, their effects on the performance of the SGRNN may vary. This is the result of encountered problems and varying solution proposals.

In the first method, outlier problem encountered by k-means GRNN algorithm, was solved reasonably through outlier detection algorithm based on probability density function. Therefore, the negative side effects of employing a clustering algorithm as a pre-processor on prediction performance has been reduced significantly and proposed approach achieved almost similar prediction performances to standard GRNN's by using ~95% less training samples.

In the second method, when kNN-GRNN algorithm was executed, getting stuck on data heap problem encountered and it was solved for 3-dimensional

problems by using angle based nearest neighbourhood, instead of kNN. Moreover, when we calculated the average change in error rates, it is seen that ~22% increase in prediction performance was provided in terms of MSE by using ~99% (6 in 900) less number of training data.

In the third method, a new SOM based GRNN algorithm that selects the significant training data and reduces the data complexity is proposed. Unlike traditional clustering algorithms, SOM clustering forms a semantic map where similar cluster centres are mapped close together and dissimilar ones apart. Depending on this feature of the SOM, the major drawback of pattern layer complexity of SGRNN for huge data sets has been overcome. Proposed method decreased the number of required pattern layer neurons around 89% for SOM size 8 without any loss in performance comparing to SGRNN.

Percentages of error change of algorithms for test functions and real world datasets compared to the SGRNN are summarized in Figure 6.2 and Figure 6.3 respectively. The changes in the positive direction show the percentage of improvement in performance, and the changes in the negative side show the percentage of getting worse in performance. As in SOM-GRNN results, the smaller the amount of change, the more similar the results are to the SGRNN.

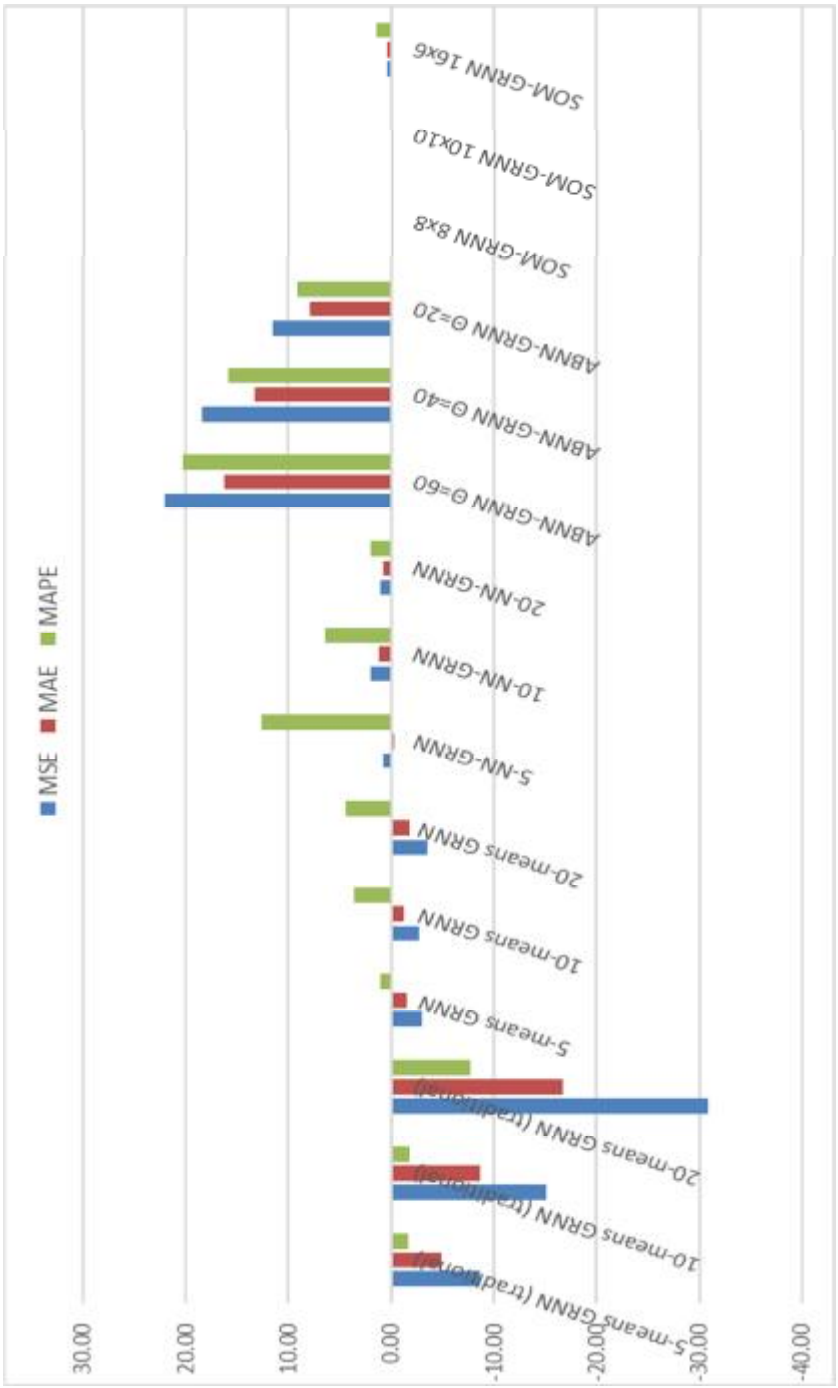


Figure 6.2. Percentages of error change of algorithms for test functions compared to the SGRNN

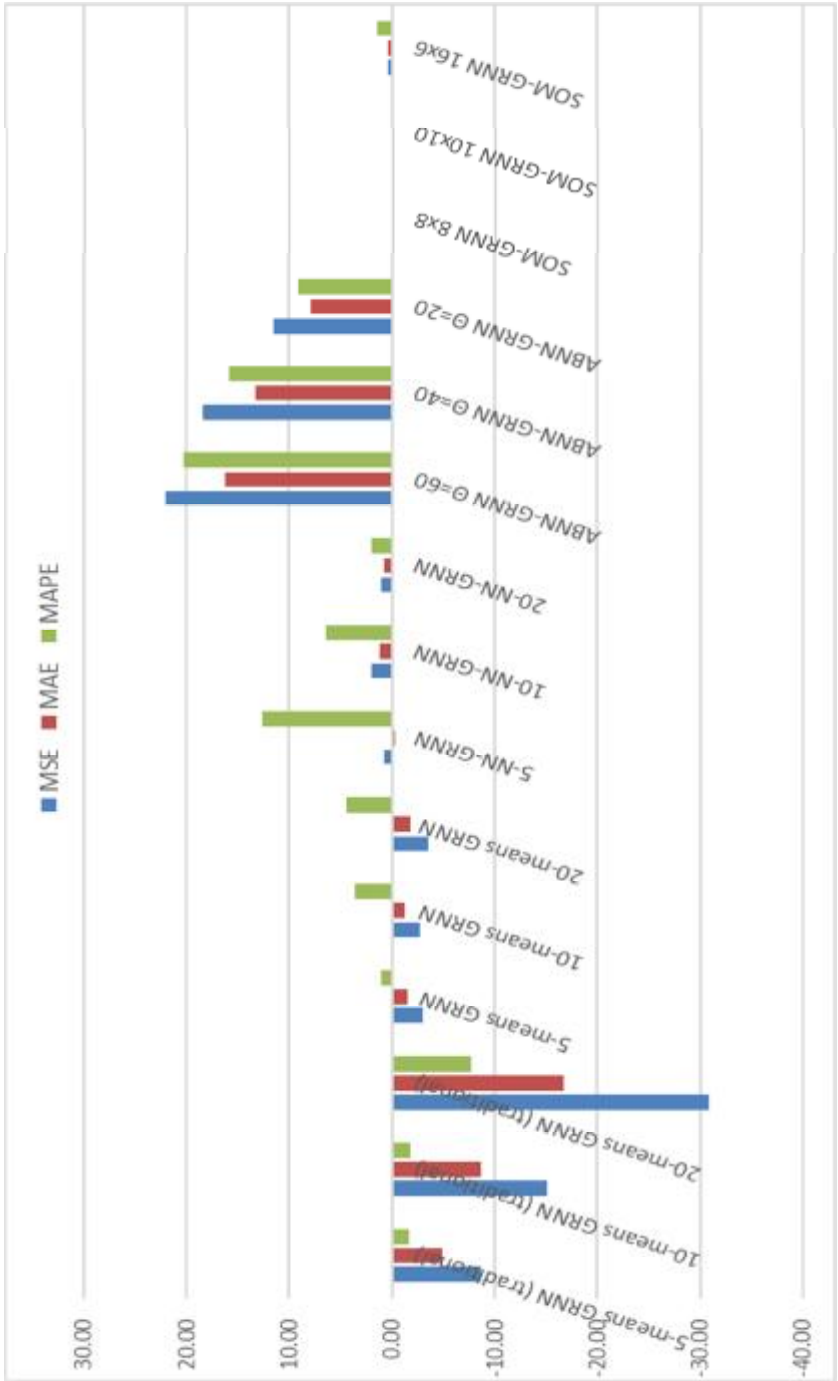


Figure 6.3. Percentages of error change of algorithms for real world datasets compared to the SGRNN

According to the complexity analysis k-means GRNN, kNN-GRNN and SOM-GRNN have *O(INkm)*, *O(RNlogNm)* and *O(IkNm)* computational complexities, respectively, where *I* denotes the iteration count, *N* is the number of training data, *k* is the cluster count, *m* is the dimension of the training data, *R* is the number of reduced dataset. When superficially examined, the complexity of the standard algorithm *O(Nm)* seems less than that of others. However, SOM and k-means are pre-processing algorithms and have to be performed before testing step. Thus, when a new test datum is applied, there remains two operations that are detection of nearest clusters and GRNN operation with reduced training data. This feature ensures faster prediction for the instantaneous test datum, and decrease in computational complexity from *O(Nm)* to *O(Rm)*, where $R << N$.

7. CONCLUSION

In this thesis, designing a new pre-processing algorithm that selects the significant training data and reduces the sample complexity of the SGRNN was aimed. These aims have been achieved by developing three hybrid GRNN algorithms called as k-means GRNN with outlier detection, ABNN-GRNN and SOM-GRNN.

GRNN is an efficient prediction algorithm, having easy modelling structure, and one pass learning. However, it employs a neuron in pattern layer for each data sample in training dataset. Since pattern layer size is proportional to the number of data in training dataset, GRNN requires huge amount of memory space and calculations for very large data sets. Therefore, an appropriate pre-processing method is required to simplify the structure of the GRNN. In this study, three approaches selecting the best representative data samples from training dataset are proposed to reduce required memory space and calculation, or in other words to reduce data complexity.

The first method is based on the idea of finding out and overcoming the inner outliers emerging by using the clustering algorithms. Thanks to the probability density function, outlier data that exist in the borders of the clusters are detected and their negative effect is minimized. In the second method, by using ABNN, the nearest training data from all around the test datum are searched step by step in a certain angle. The proposed method provides a solution to convergence problem of the GRNN especially in some special datasets which have local data heaps. The third method employs SOM as a pre-processing algorithm for GRNN. Using its neighbourhood map, SOM-GRNN eliminates the outlier problems of the clustering algorithms. Moreover, the major drawback of pattern layer complexity of SGRNN for huge data sets is overcome.

Consequently, one of the biggest problem of the SGRNN, data complexity, is solved with proposed algorithms. As our future perspective, we plan to further

improve the GRNN's performance by integrating the best sigma selection algorithms with the proposed pre-processing methods.

REFERENCES

Back, A.D., Trappenberg, T.P., 1999. Input variable selection using independent component analysis. In Proceedings of International Joint Conference on Neural Networks 2:989-992.

Berkhin, P., 2002. Survey of clustering data mining techniques. Technical report, Accrue Software.

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

Bowden, G.J., Maier, H.R., Dandy, G.C., 2005b. Input determination for neural network models in water resources applications. Part 2. Case study: forecasting salinity in a river. Journal of Hydrology, 301(1):93-107.

Brooks, T.F., Pope, D.S., Marcolini, A.M., 1989. Airfoil self-noise and prediction. Technical report, NASA, RP-1218.

Calinski, T., Harabasz, J., 1974. A dendrite method for cluster analysis, Communications in Statistics-theory and Methods, 3(1), 1–27.

Carrasco K.M., Brunner, R. J., 2014. SOM z: photometric redshift PDFs with self-organizing maps and random atlas. Monthly Notices of the Royal Astronomical Society, 438(4), 3409-3421.

Chen, J., Li, H., Tang, S., Sun, J., 2002. A SOM-based probabilistic neural network for classification of ship noises. IEEE Int. Conf. On Communication, Circuits and Systems and West Sino Expositions, 2:1209-1212.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., Reis, J., 2009. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, 47(4):547-553.

Cover, T., Hart, P. 1967. Nearest neighbor pattern classification. IEEE transactions on information theory, 1:21-27.

Davies, D.L., Bouldin, D.W., 1979. A cluster separation measure, IEEE transactions on pattern analysis and machine intelligence (2): 224–227.

Everitt, B.S., Landau, S., Leese, M., Stahl, D., 2011. Miscellaneous clustering methods. Cluster Analysis, 5th Edition, 215-255.

Fahad, A., Alshatri, N. ve Tari, Z., 2014. A Survey of Clustering Algorithms for Big Data: Taxonomy & Empirical Analysis. IEEE Transactions on Emerging Topics in Computing, 2(3):267-279.

Hamzacebi, C., 2008. Improving genetic algorithms' performance by local search for continuous function optimization. Journal of Applied Mathematics and Computation 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., Wong, M.A., 1979. Algorithm AS 136: A k-means clustering algorithm. Journal of the Royal Statistical Society, Series C Applied Statistics. 1:100-108.

Hamming, R.W. 1980. Coding and information theory. Prentice-Hall. 259.

Husain, H., Khalid, M., Yusof R., 2004. Automatic clustering of generalized regression neural network by similarity index based fuzzy C-means clustering. IEEE Region 10 Conference, 2:302–305 .

Jain, A.K., Mao, J., Mohiuddin, K. M., 1996, Artificial Neural Networks: A Tutorial, Computer, IEEE Computer Society Magazine, 29(3):31-44.

Jain, A.K., Murty, M.N., and Flynn, P.J., 1999. Data Clustering: A Review. ACM Computing Surveys, 31(3):264-323.

Kartal, S., Oral M., 2017. New Pattern Reduction Method for Generalized Regression Neural Network. International Journal of Advanced Research in Computer Science and Software Engineering, 7(2):122-129

Kokkinos, Y., Margaritis, K.G., 2015. A fast progressive local learning regression ensemble of generalized regression neural networks. Proceedings of the 19th Panhellenic Conference on Informatics, 109-114.

Kolesnikov, A., Trichina, E., Kauranne, T., 2015. Estimating the number of clusters in a numerical data set via quantization error modelling. Pattern Recognition 3:941-952.

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

Krenker, A., Bester, J., Kos, A. 2011. Introduction to the artificial neural networks. Artificial Neural Networks-Methodological Advances and Biomedical Applications, 1-18.

Lichman, M. 2013. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Maier, H.R., Dandy, G.C., 1997. Determining inputs for neural network models of multivariate time series. Microcomputers in Civil Engineering, 12(5):353–368.

Mu, T., Nandi, A. K., 2007. Breast cancer detection from FNA using SVM with different parameter tuning systems and SOM-RBF classifier. Journal of the Franklin Institute-Engineering and Applied Mathematics, 344.285 - 311.

Wikipedia Contributors, 2017. Multivariate normal distribution. Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Multivariate_normal_distribution&oldid=776291509.

Omondi, A.R., Rajapakse, J.C., 2006. FPGA Implementation of Neural Networks. Springer, Netherlands.

Oral, M., Oral, E., Aydin, A., 2012. Supervised vs. unsupervised learning for construction crew productivity prediction. Automation in Construction, 22:271-276.

Ozyildirim, B. M., Avci, M., 2013. Generalized Classifier Neural Network. Neural Networks, 39:18 – 26.

Rama, B., Jayashree, P., Jiwani, S., 2010. A survey on clustering, current status and challenging issues. International Journal on Computer Science and Engineering, 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.

Sato, M., Sato, Y., Jain, L., 1997. Fuzzy Clustering Models and Applications, Physica-Verlag.

SOM Self Organizing Maps in NetLogo, 2017. http://www.cs.us.es/~fsancho/?e=136.

Seng, T. L., Khalid, M., Tusof, R., 2002. Adaptive GRNN for the Modeling of Dynamic Plants. Proc. of the 2002 IEEE International Symposium on Intelligent Control Vancouver, 21(3):217-222.

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

Specht, D.F., Shapiro, P.D., 1991. Generalization accuracy of probabilistic neural networks compared with backpropagation networks. Neural Networks, IJCNN-91-Seattle International Joint Conference,1:887 –892.

Specht, D.F., 2006. GRNN with double clustering. In Neural Networks. IJCNN'06, International Joint Conference on, 5074-5079.

Tang, K., Li, X., Suganthan, P.N., Yang, Z. ve Weise, T., 2008. Benchmark Functions for the CEC'2010 Special Session and Competition on Large Scale Global Optimization. Technical Report, Nature Inspired Computation and Applications Laboratory.

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

Tsanas, A., Xifara, A., 2012. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. Energy and Buildings, 49:560-567.

Tüfekci, P., 2014. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. International Journal of Electrical Power & Energy Systems, 60:126-140.

Vito, S.D., Massera, E., Piga, M., Martinotto, L., Francia, G.D., 2008. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. Sensors and Actuators B: Chemical, 129(2):750-757.

Yamanishi, K., Takeuchi, J.I., Williams, G., Milne, P., 2000. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, 320-324.

Yeh, I.C., 1998. Modeling of strength of high-performance concrete using artificial neural networks. Cement and Concrete research, 28(12): 1797-1808.

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

Zhao, S., Zhang J., LI, X., Song, W.A., 2007. A Generalized regression neural network based on fuzzy means clustering and its application in system identification. Information Technology Convergence, 13-16.

Zheng, G.L., Billings, S.A., 1996. Radial Basis Function Network Configuration Using Mutual Information and the Orthogonal Least Squares Algorithm. Neural Networks, 9:1619–1637.

Zheng, L.G., Yu, M.G., Yu, S.J., Wang, W., 2008. Improved prediction of nitrogen oxides using GRNN with k-means clustering and EDA. Fourth International Conference on Natural Computation 2: 91-95.

CURRICULUM VITAE

Serkan KARTAL was born in Adana, in 1986. He completed his elementary education at Meryem Abdurrahim Gizer Primary Education School. He graduated from ÇEAŞ Anatolian High School, in 2004. He received the B.E and MSc degrees from Department of Computer Engineering, Çukurova University in 2010 and 2013, respectively. Since 2011, he has been working as a research assistant at Computer Engineering Department of Çukurova University in Adana.