Volume 7, Issue 2, February 2017 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

New Pattern Reduction Method for Generalized Regression Neural Network

Serkan Kartal, Mustafa Oral
Department of Computer Engineering & University of Cukurova,
Turkey

DOI: 10.23956/ijarcsse/V7I2/01213

Abstract— Generalized Regression Neural Network (GRNN) is a radial basis function based neural network used for function approximation and prediction. Thanks to its easy modelling structure, and one pass learning, it has been utilized in many applications as an alternative to other prediction methods such as multilayer perceptron (MLP) and support vector machines (SVM). 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 decrease in computational time will emerge for huge datasets. Therefore, k-nearest neighbour (kNN) and clustering methods such as k-means and hierarchical clustering, etc. have been frequently used for pattern layer size reduction. Pattern layer size reduction may provide not only simplification in structure but also increase in prediction accuracy. In this work, a pattern layer size reduction approach utilizing Angle Based Nearest Neighbor (ABNN) algorithm is proposed for three-dimensional datasets. The proposed method divides training space into specific angles and for each test datum, it searches the nearest training datum within each angle. At the end, there exists a few training data that will be used in GRNN's pattern layer and these training data are similar to the test datum. Performance of the proposed method was evaluated by using fifteen benchmark global optimization test functions and compared with that of standard GRNN and a hybrid method using kNN as a pre-processor. Simulation results show that the proposed method provides 99.33% reduction in pattern layer size and accuracy is also increased maximally to 65.61%.

Keywords— Generalized regression neural network, prediction neural networks, nearest neighbor, pattern reduction and reduced dataset.

I. INTRODUCTION

Donald F Specht proposed the model of General Regression Neural Network (GRNN) as an alternative to back propagation training algorithms [1]. The main advantage of the GRNN model over the back propagation model is its one pass learning. However, in the Specht's GRNN, the number of hidden nodes equals to the number of training samples. The more training samples included in dataset; the larger network size is obtained. This drawback causes decrease in efficiency and accuracy of the GRNN. Hence, if some of the representative samples from training data can be detected and used in pattern layer instead of using whole dataset, GRNN structure will be simplified and its drawback will be eliminated. Representative samples can be detected by grouping similar training samples. Clustering or cluster analysis is a method for separating data into groups (clusters) of similar objects. This helps in the abstraction process of large dataset and summarizing its main characteristics [2]-[4]. Hence, clustering has been used for calculating similarities between input elements, compressing training samples, and determining representative data of samples. If such an algorithm is used as pre-processor for GRNN, representative samples can be detected and pattern size weakness may be eliminated.

Clustering has been utilized as a pre-processor in various applications. Due to its wide range of usage, it is difficult to support all requirements of different applications through one clustering method. There exists different clustering algorithms whose methodologies vary in accordance with the properties of applications such as number of samples, error tolerance, and size of each instance [5], [6]. Therefore, examining the characteristics of the given problem is critical step for choosing or designing appropriate clustering algorithm. 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 accordance with the similarity matrix [7]. Partitioning methods are divided into two major subcategories named as the centroid-based and the medoid-based clustering [7]. While clusters are represented by using gravity center of the instances in centroid-based clustering; in medoid clustering, each cluster is represented with mean values of the samples closest to the gravity center of instances. In density-based clustering, clusters are denoted with the density of the regions. The cluster grows in any direction leading by its density and each cluster has to contain at least minimum number of data in a neighborhood of a given radius [3]. In grid based algorithms, initially clustering space is divided into finite number of cells. Statistical features are collected for all data located in each cell. Then, clustering is performed on the grid [6]. Model based approaches begin with random parameter sets and they adjust these parameter sets over the iterations to find maximum likelihood estimator [3]. Similar to other optimization problems, clustering is also a kind of NP-hard problem and one of the biggest problem encountered in clustering approaches is to estimate the number of clusters. There is no convincingly acceptable solution to this problem [8].

A useful alternative approach to find representative data is the k-Nearest Neighbor (kNN) method. Unlike clustering methods, it does not require estimation of the number of classes, it uses only neighborhoods. Since kNN uses only k-nearest training samples, kNN reduces operation cost from *O(n)* to *O(k)*; where k<n and which is more appropriate for large scale applications. However, there are two important issues that should be concerned: how to determine the best neighbors and the number of neighbors *(k)*. Mostly, Euclidean distance is used for determining the nearest k-patterns. However, determining the number of nearest neighbors *(k)* is more critical. If the small number of neighbor is chosen, network structure may be unreliable. On the other hand, if it is too large, some of the remote data may also be taken into evaluation and efficiency of the method may decrease.

Even if all mentioned pre-processing algorithms reduce the sample complexity of GRNN network, prediction accuracy is not improved adequately. At most, they produce the same results with the standard GRNN by using reduced dataset. In addition to problems encountered in mentioned approaches, there is another weakness of GRNN has not been addressed yet: getting stuck on local data groups. By the nature of the GRNN, the test datum is more sensitive to the nearest training data. Therefore, if there exists a data heap around the test data, estimated value may be close to average value of that data heap and more prediction error may be obtained than expected. In this study, we proposed the "Angle Based Nearest Neighbor" (ABNN)-GRNN that reduces sample complexity and prediction error for three-dimensional problems.

II. GENERALIZED REGRESSION NEURAL NETWORK (GRNN)

GRNN is one pass learning function approximation algorithm utilizing Parzen window estimation [1]. It has four layers; input, pattern, summation and output. The schematic depiction of the four layers are illustrated in Figure 1. In the first layer number of neurons is equal to the total number of parameters. The second is pattern layer where the number of neurons is equal to number of training samples. It computes the Euclidean distance between applied pattern and stored patterns and applies Gaussian function.

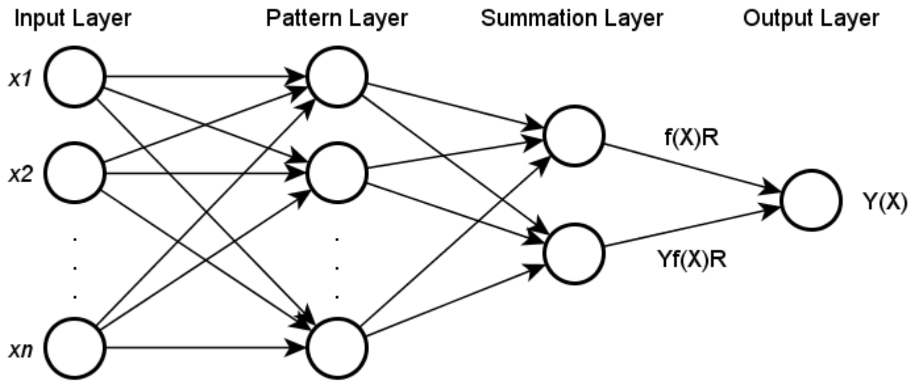

Fig. 1 Schematic diagram of GRNN

The output values of pattern layer is fed to summation layer which consists of two different units; denominator summation unit and numerator summation unit. In summation layer, denominator is calculated by summing up of weight values calculated in each pattern neurons and numerator is computed by summing up the weight values multiplied by actual target value for each pattern neuron. Finally, output unit divides the value accumulated in the numerator summation unit by the value in the denominator summation unit to provide prediction result. A critical consideration in the effectiveness of GRNN is choice of smoothing factor σ. As σ is very large, GRNN approaches the mean of the training set outputs; and as σ is very small only a few samples play role in evaluation. It is the only parameter that needs to be tuned by the user.

III. RELATED WORK ON GRNN

The various GRNN algorithms in [9]-[16] perform clustering to simplify network structure. Firstly, Specht [1] proposed a clustering version of GRNN to reduce the number of neurons in pattern layer. Later, Shi-jun et al. [9] introduced a method based on fuzzy means clustering to reduce the number of pattern neurons. Fuzzy means clustering groups the input-output data pairs into clusters. There are two steps to divide clusters. Firstly, similarities are calculated among input data to find the best clustering centers. Secondly, these values are compared with a defined threshold value. If similarity is smaller than threshold value, corresponding datum is placed into the related cluster. Otherwise, it is taken as the center of a newly created cluster. According to the given results, proposed method simplified GRNN's structure with a tiny of loss in accuracy.

GRNN and fuzzy c-means clustering were also used by Husain et al. [10] Fuzzy c-means uses one-pass simple similarity measure on data to calculate the similarity index, which indicates the degree of similarity. Calculated values are compared with a threshold level to determine whether the data are in the vicinity of each other. Indices higher than the threshold level are considered as similar and related data are placed into the appropriate cluster, otherwise they are considered as 'dissimilar' and data are determined as the representation of new cluster. Lastly, the number of clusters and the calculated centers are used to define the parameters of GRNN. Zheng et al., proposed GRNN with k-means clustering and estimation of distribution algorithm (EDA) to reduce the number of smoothing parameters. Firstly, the training data are partitioned into groups using k-means clustering. A smoothing parameter is assigned to each group. Then, each parameter is optimized with EDAs. Lastly, the k optimized smoothing parameters are utilized to form GRNN model for the prediction. According to the [11], modelling errors were reduced by the proposed GRNN method.

A new regression model for noisy data was proposed by Yuen et al [12]. It is a hybrid model and formed by the combination of Fuzzy ART (FA) and GRNN. FA is one of the powerful unsupervised classifier and was developed based on Adaptive Resonance Theory (ART). It is employed as the pre-processor of the GRNN to compress the training data into a few kernels. The approach utilizes the k-nearest-neighbors (kNN) to estimate the width of each kernel. Center of a kernel is determined by centroid of training data being clustered into that kernel.

Seng et al. [14] presented Adaptive GRNN for dynamic plant modelling 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. In Adaptive GRNN, the network can evolve 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 presented into the network, a new pattern node may be created to store this new knowledge. A new neuron is created only if it is necessary, and the maximum number of neuron is also limited. A threshold level is used to determine the necessity. Moreover, less valuable pattern nodes can be discarded.

Kokkinos and Margaritis [15] proposed Progressive Local Learning GRNN ensemble. The algorithm utilizes the order of neighbors and reduces amount of training samples by optimizing the number of k nearest neighbor of sample data. The ensemble members are constructed progressively by adding nearest neighbors to each local GRNN member. While the first member is constructed with a minimum number of nearest neighbors of the testing point, the last member uses the full list of nearest neighbors.

IV. ANGLE BASED NEAREST NEIGHBORS (ABNN)

k-Nearest Neighbors is a simple algorithm that finds the k closest training samples to the test datum. Nearest neighbors are commonly measured by Euclidean distance for continuous variables. Generally, it is used for both classification and regression. An example of kNN algorithm applied to generated sample dataset is illustrated in Fig. 2 where k is chosen as ten. While blue-circles point out train data group, red- triangles point out k nearest data to the green-square test datum.

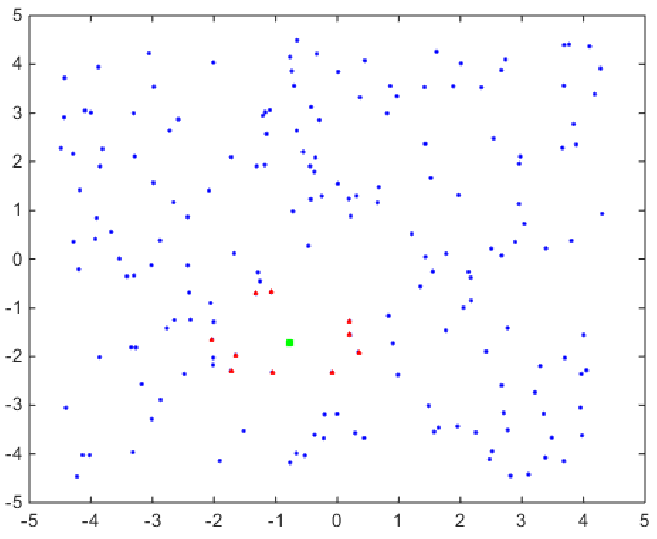

Fig. 2 Nearest neighbors over normal scattered data

kNN is also used as an alternative pre-processing method for GRNN [15]. In standard GRNN, distance value between each train datum and test datum is used as a measure of how well training datum can represent result. However, if k-nearest train data are located in a narrow space, then the prediction value will be around this collection. Therefore, the prediction error may be more than expected and prediction accuracy may decrease. This problem is one of the less discussed weakness of traditional GRNN and has become more prominent by use of the local learning algorithms.

As mentioned before, one of the biggest problem encountered in kNN is to determine the best estimate of the value of k. In addition to this problem, kNN algorithm is also sensitive to the local structure of the data. All the nearest neighbors may be selected from data which get stuck in a confined space. Then, the classification or regression algorithm may produce inaccurate results. In contrast to the results in Fig. 2, the kNN algorithm may produce a result as in Fig. 3. It can be observed that, most of the train data are grouped in a certain region and this may result in a prediction value that is around the average point of these data.

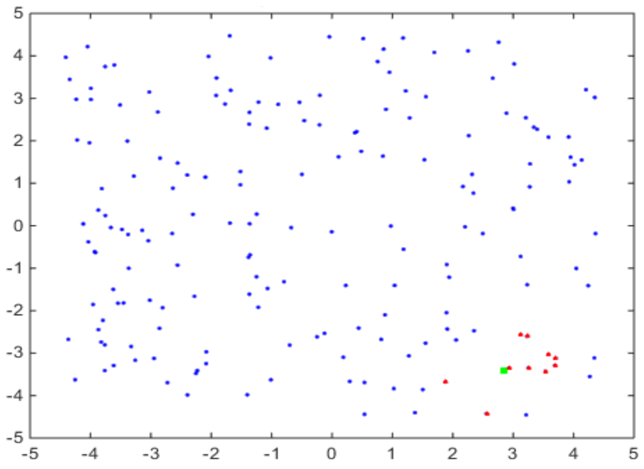

Fig.3. 10 Nearest Neighbors, over Stacked Data

To overcome these weaknesses and prevent GRNN to get stuck on local data groups, in this study, an Angle Based Nearest Neighbor (ABNN) algorithm is proposed as a pre-processor for traditional GRNN. ABNN promotes finding nearest training data from all around the test datum step by step. In each step, just one nearest training datum is searched in given angle. After all steps are completed, all obtained closest data are combined and applied to the GRNN as training data. In ABNN, angle θ between two points is measured by using arctangent function. Calculation of θ between two points, A and B, is illustrated in Fig. 4. Firstly, distance values for each dimensions are calculated to find θ. Then, values are given to the arctangent function as parameters, in (6). Lastly, obtained radian value is converted to the degree with Eq. (7).

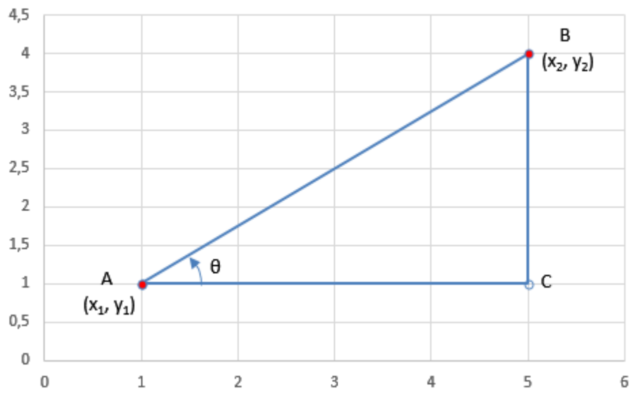

Fig.4. θ calculation of two points, A and B.

$$\text{Radian(A,B)= arctangent } \left(\frac{|BC|}{|AC|}\right) \quad (6)$$

$$\theta(\text{A,B})= \left(\frac{\pi}{180}\right) * \text{Radian (A,B)} \quad (7)$$

Process of ABNN algorithm with θ =40 is illustrated in Fig. 5. While green-square points out the test datum, red-triangles point out found nearest data in given angle.

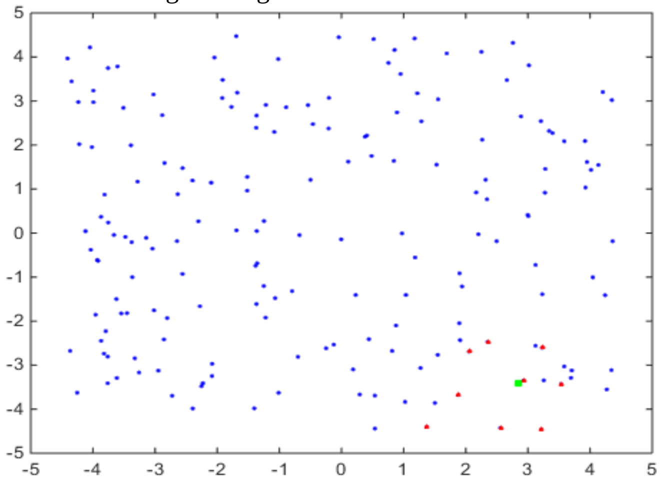

Fig.5. ABNN with θ = 40

If we compare suitability of ABNN result with that of standard kNN, it can be seen that ABNN provides better representation for all training samples than the standard kNN provides. Moreover, while training samples obtained from kNN lead to the same prediction result with that of traditional GRNN, training samples obtained from ABNN do not get stuck on local data group and produce more accurate predictions.

Computational complexity of kNN for one test datum is *Q(n x cost of distance calculation between test datum and a training datum)*, where n is the number of training data. In addition to this, ABNN requires one extra operation for each test datum to get angle between test datum and each training datum. Thus, computational cost of the ABNN method for one test datum is *Q(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, *Q(n),* asymptotically.

V. TESTS AND RESULTS

15 real valued, well known benchmark test functions were employed to compare performances of the proposed method with that of standard GRNN and kNN-GRNN [17]. Since the proposed algorithm is applicable for three-dimensional data, 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 1.

Table I: 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	Goldstein-Price	[-2, 2]	3
f_5	Six- Hump Camel	[-5, 5]	-1.031628...
f_6	Bird	[-2*π, 2*π]	-106.764537
f_7	Leon	[-1.2, 1.2]	0
f_8	Griewangk	[-100, 100]	0
f_9	Giunta	[-1, 1]	0.060247
f_{10}	Booth	[-10, 10]	0
f_{11}	Bukin4	[-15, -5], [-5, +3]	0
f_{12}	Carromtable	[-10, 10]	24.156815
f_{13}	Crossfunc	[-10, 10]	0
f_{14}	Himmelblau	[-5, 5]	0
f_{15}	Matyas	[-10, 10]	0

Standard GRNN, kNN, and ABNN have to be configured and tuned with different network structures and parameters such as best sigma, k number and angle size to achieve the best prediction performances. Optimization of sigma was provided by 10-fold cross-validation technique. It was started at 0.1 and increased gradually up to 3 with 0.1 steps to find the most suitable value. The same sigma value is used in all other methods. However, if we intended to carry out similar optimization procedure for k and angle parameter, calculations would need huge number of tests and time. Preliminary works showed that three different k (5, 10, and 20) and three different angle (60, 40, and 20) parameters are sufficient to demonstrate the prediction performances of the algorithms. The angle values were selected relatively to the chosen k values to make more reliable comparison between kNN and ABNN. Angles 60, 40 and 20 are reciprocal to 6, 9 and 18 nearest neighbors, respectively. Those values are approximated to 5, 10, and 20 for kNN. Unlike standard GRNN that uses whole training set in the pattern layer, only the k numbers of pattern units are used for kNN and 360/ θ number of pattern units used for ABNN.

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. The created data were separated to ten sub-groups containing 100 samples in each. 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 15 different random seed values for each benchmark function to decrease the effect of randomness on the results. Therefore, average error values were used to present the efficiency of predictions [16]. Three standard error measures were implemented to evaluate the performance; Mean Squared Error (MSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE). Their equations were given in (8), (9) and (10), respectively.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(A_i - P_i)^2 \quad (8)$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|A_i - P_i| \quad (9)$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{A_i - P_i}{A_i}\right| * 100\% \quad (10)$$

where;

- n number of data used for prediction.
- A_i actual value of the ith element of the dataset.
- P_i predicted value of the ith element of the dataset.

Average prediction errors for MSE, MAE and MAPE obtained from the tests are presented in Table 2, 3 and 4, respectively. According to the Table-2 5-NN provides better or equal prediction performances for 4 test functions with only 5 training data. However, negative effect of the 5-NN is clearly seen over the remaining eleven test functions. The results in Table 3 and Table 4 also show that the 5-NN produces better result for up to 5 functions. Prediction performances for other functions are not enough to improve overall prediction accuracy. When the k is increased to 10 and 20, the results indicate that the differences between the traditional GRNN and kNN-GRNN predictions become smaller. Although there are not significant improvements on the results, it can be seen that almost the same prediction performances are obtained by using 20 train data which is the 2.2\% of all training data.

Table II MSE 10-Fold Cross Validation Performances

	STD GRNN	KNN-GRNN (Neighbour Count-MSE)			ANN-GRNN (Angle Range)		
		k=5	k=10	k=20	Θ=60	Θ=40	Θ=20
1	14025559,35	14209767,71	14048023,37	14027500,63	**12394325,96**	12518984,47	12900134,69
2	47,58952942	45,54387801	47,21337228	47,58256864	**42,39800063**	43,53031924	45,2765183
3	12509,46084	12509,46084	12509,46084	12509,46084	12512,76277	12513,70392	**12507,61403**
4	165672055,9	174729140	168125334,6	165730633,4	**124665717,8**	135546394,8	146686555,5
5	34861,28947	35188,79633	34923,86503	34862,27008	**30158,23163**	31134,72031	32567,02383
6	30,8160371	30,97257242	30,76606748	30,80588242	**23,22244507**	24,85835944	27,3016535
7	390,7235041	309,691218	350,930012	381,8578107	**197,5403207**	224,6700315	277,0296139
8	3,890109742	3,898443638	3,890085379	3,890109742	3,845262026	3,843757287	**3,838110134**
9	0,00047961	0,00028717	0,00034042	0,000422102	**0,000164916**	0,000194218	0,000272015
10	430,9307794	492,2161002	447,3167172	431,7534103	**303,8746563**	314,1970735	360,5567281
11	779,5879277	865,200676	789,2845927	780,4757305	**558,0603911**	613,5600618	686,0850957
12	1,320906971	1,338665558	1,320555993	1,320899565	**1,262471605**	1,301255181	1,303006821
13	7,34195E-11	7,34682E-11	7,32126E-11	7,34064E-11	7,23524E-11	**7,12141E-11**	7,21179E-11
14	335,3318832	341,0959918	336,1316676	335,3595003	**294,2597854**	299,6367843	314,6151636
15	1,177352546	1,317406557	1,223816988	1,179276357	**0,829923189**	0,893975012	1,00781014

Table III MAE 10-Fold Cross Validation Performances

	STD GRNN	KNN-GRNN (Neighbour Count)			ANN-GRNN (Angle Range)		
		k=5	k=10	k=20	Θ=60	Θ=40	Θ=20
1	1129,879141	1142,706349	1134,185875	1130,143728	**984,2764968**	1002,432233	1043,862938
2	5,437273387	5,294099524	5,41122676	5,436793467	**5,02437949**	5,148978304	5,274353519
3	80,95301035	80,95301035	80,95301035	80,95301035	80,95202959	80,96178013	**80,94187845**
4	4669,868505	4868,806564	4726,061709	4671,122698	**3867,466172**	4110,028724	4322,432768
5	87,05244161	88,48390817	87,15906418	87,05360381	**77,04019271**	78,88883913	82,67533149
6	3,394541535	3,413426872	3,390673197	3,394701653	**2,843984125**	2,996732753	3,180044273
7	8,628839485	7,611063794	8,06316191	8,441954145	**5,324537896**	5,708476277	6,79227476
8	1,429962136	1,432784803	1,429948528	1,429962136	1,418882971	1,420621506	**1,418578406**
9	0,013474613	0,010551349	0,010986123	0,012137525	**0,00660498**	0,007179222	0,009133591
10	12,6747812	13,7948604	12,90654137	12,68704571	**9,847178462**	10,33086364	11,33584626
11	16,55481857	17,73919009	16,77242888	16,56246712	**12,97223399**	13,75960169	15,07906642
12	0,290108051	0,28911966	0,289639155	0,290089485	**0,274527617**	0,280358259	0,284860479
13	5,04614E-06	5,00544E-06	5,01553E-06	5,0445E-06	4,88922E-06	**4,87288E-06**	4,95595E-06
14	9,504374526	9,697829611	9,523362571	9,504795428	**8,395219216**	8,677973765	9,040765084
15	0,634541884	0,685326828	0,65066892	0,635354851	**0,515888467**	0,54167026	0,581295297

Table IV MAPE 10-FOLD Cross Validation Performances

	STD GRNN	KNN-GRNN (Neighbour Count)			ANN-GRNN (Angle Range)		
		k=5	k=10	k=20	Θ=60	Θ=40	Θ=20
1	18,6132454	19,93753957	18,61867472	18,60296575	17,27671768	**16,91961781**	17,59922202
2	16,89167182	16,63865874	16,82347388	16,89023541	**15,6818479**	16,0250286	16,47190417
3	185,494409	185,494409	185,494409	185,494409	185,4384765	**181,1953355**	185,4935678
4	15,69918603	220,313404	15,50031582	15,65253256	**12,84928179**	13,39330652	14,44846881
5	86,57950036	18,96100317	35,16659636	85,37044881	25,86743604	31,2425199	**17,10998533**
6	102,2730094	30,77679794	29,86060975	238,2061842	26,94952211	**24,90164329**	96,42061408
7	15,62483286	13,34754704	13,65714443	14,83403744	**10,12164709**	10,96514631	12,61638615
8	6,784584795	6,796867876	6,78455716	6,784584795	6,780227299	6,773059818	**6,750482268**
9	4,109271845	3,09686616	3,228137651	3,620249956	**1,937566036**	2,14899134	2,755631976
10	5,774022821	5,957766546	5,757938241	5,761968756	**4,59537389**	4,837365971	5,25629586
11	12,39616663	13,0935088	12,40937631	12,39089827	**10,64511113**	10,84294793	11,70192162
12	71,8079749	78,75804764	72,11042688	71,81101133	71,55207651	74,10388915	**71,25119134**
13	7,016617892	6,887563029	6,958278923	7,013511969	**6,715007402**	6,723045373	6,860506629
14	9,408404686	9,656971555	9,416836301	9,407939141	**8,325836411**	8,61192464	8,953522657
15	6,712230298	7,137820042	6,797071038	6,713696975	**5,444905837**	5,695562545	6,148495265

Results in Table 2, 3 and 4 show that the prediction accuracy of ABNN-GRNN is superior to kNN-GRNN and traditional GRNN with respect to MSE, MAE and MAPE values for all given angles. Proposed pre-processing algorithm provides better prediction performance for 14 benchmark functions and equal performance for only 1 function by using less training data. Error tables displaying the best and the worst prediction performances of the compared algorithms with their different parameters show that ABNN provides the best performance for angle 60 (6 neighbors). While in function 9, 65.61% increase in prediction performance is provided in terms of MSE, it requires ~99% (6 in 900) less number of training data. Furthermore, 56.01% and 54.07% performance increases are obtained for function 7 and 9 with respect to MAPE, respectively. Moreover, by using ABNN, performances are improved in the range of 11.63-65.65%, 10.64-59.51% and 6.18%-43.28% for 10 different benchmark functions for 60, 40 and 20 angles, respectively. Almost the same prediction performance is obtained with the compared methods for only function 3, by using 6 train data

Considering all test functions, kNN and ABNN decreased the number of required hidden neurons nearly 99%. While prediction accuracies of traditional GRNN and kNN-GRNN are almost the same with respect to MAPE, MSE and MAE, the prediction accuracy of the ABNN -GRNN is superior to both of them in all error measures. Consequently, test results show that the proposed method provides two major improvements to standard GRNN. While it reduces the number of required train data; it also improves the prediction accuracy. By using appropriate angle, smaller train datasets and better prediction performances may be achieved.

VI. CONCLUSIONS

GRNN is an efficient prediction algorithm, however, when the training samples become enormous the network becomes huge. Therefore, an appropriate pre-processing method is required to simplify the structure of the GRNN. In this study, an approach selecting the best representative data samples from training dataset, is proposed to avoid huge structure for three dimensional problems. Three different error measurements; MAPE, MSE, MAE; are used to evaluate the performance of proposed algorithm. By using ABNN, the nearest training data from all around the test datum are searched step by step in a certain angle. The only parameter affecting the performance of ABNN is the angle range. According to the test results, using ABNN as a pre-processor provides two main improvements on GRNN. While ABNN decreases the number of required training data, it also improves the prediction accuracy for all test functions. In addition to these, the proposed method provides a solution to convergence problem of the GRNN especially in some special datasets which have local data heaps

REFERENCES

[1] D. F., Specht, "A general regression neural network", IEEE Transactions on Neural Networks, 2, 6,568-576, 1991.

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

[3] Fahad, A., Alshatri, N., Tari, Z., "A survey of clustering algorithms for big data: Taxonomy and empirical analysis", IEEE Transactions on Emerging Topics in Computing, 2, 3, 267-279, 2014.

[4] Berkhin, P., "Survey of clustering data mining techniques", Technical report, Accrue Software, 2002.

[5] Harkanth, S. and Phulpagar, B. D., "A Survey on clustering methods and algorithms", International Journal of Computer Science and Information Technologies, 4, 5, 687-691, 2013.

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

[7] Rama, B., Jayashree, P., Jiwani, S., "A survey on clustering, current status and challenging issues. International Journal on Computer Science and Engineering", 02, 09, 2976-2980, 2010.

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

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

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

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

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

[13] Mu, T., Nandi, A. K., "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 (2007).

[14] Seng, T. L., Khalid, M., Tusof, R., "Adaptive GRNN for the modeling of dynamic plants", Proceedings of the 2002 IEEE International Symposium on Intelligent Control Vancouver, 21, 3, 217-222 (2002).

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

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

[17] Tang, K., Li, X., Suganthan, P. N., Yang, Z., Weise, T., "Benchmark Functions for the CEC'2010 Special Session and Competition on Large Scale Global Optimization", Technical Report, Nature Inspired Computation and Applications Laboratory (2009).