Indonesian Journal of Electrical Engineering and Computer Science
Vol. 18, No. 1, April 2020, pp. 459~469
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v18.i1.pp459-469

Radial basis function neural network for 2 satisfiability programming

Shehab Alzaeemi[1], Mohd. Asyraf Mansor[2], Mohd Shareduwan Mohd Kasihmuddin[3], Saratha Sathasivam[4], Mustafa Mamat[5]
[1,3,4]School of Mathematical Sciences, Universiti Sains Malaysia, Malaysia
[2]School of Distance Education, Universiti Sains Malaysia, Malaysia
[5]Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Malaysia

Article Info

Article history:

Received Jun 23, 2019
Revised Sep 5, 2019
Accepted Sep 20, 2019

Keywords:

2 satisfiability
Genetic algorithm
Half-training technique
No-training technique
Radial basis functions neural network

ABSTRACT

Radial Basis Function Neural Network (RBFNN) is very prominent in data processing. However, improving this technique is vital for the NN training process. This paper presents an integrated 2 Satisfiability in radial basis function neural network (RBFNN-2SAT). There are two different types of training in RBFNN, namely no-training technique and half-training technique. The performance of the solutions via Genetic Algorithm (GA) training was investigated by comparing the Radial Basis Function Neural Network No-Training Technique (RBFNN- 2SATNT) and Radial Basis Function Neural Network Half-Training Technique (RBFNN- 2SATHT). The comparison of both techniques was examined on 2 Satisfiability problem by using a C# software that was developed for this experiment. The performance of the RBFNN-2SATNT and RBFNN-2SATHT in performing 2SAT is discussed in terms of root mean squared error (RMSE), sum squared error (SSE), mean absolute percentage error (MAPE), mean absolute error (MAE), number of the hidden neurons and CPU time. Results obtained from a computer simulation showed that RBFNN-2SATHT outperformed RBFNN-2SATNT.

Corresponding Author:

Name: Mohd. Asyraf Mansor,
School of Distance Education,
Universiti Sains Malaysia, 11800 USM, Penang, Malaysia.
Email: asyrafman@usm.my

1. INTRODUCTION

Neural-symbolic integration combines both the connectionist system and symbolic artificial intelligence (AI) systems. Symbolic AI is inspired by human thinking and formalises it by means of logic languages for knowledge representation and reasoning. Neural-symbolic structures are AI systems that can realise the symbolic processes within artificial neural networks. An artificial neural network (ANN), also known as connectionist systems, has received precise attention because of its capacity to evaluate complex non-linear data set. Neural networks (NN) are AI structures that include some mathematical and graphical models. ANN has been successfully applied to solve various applications, such as classification, function approximation, and optimisation [1]. According to connectivity, the NN has two main topologies. The first topology is recurrent network, such as the Hopfield Neural Network (HNN) where the neurons belong to the same layer and send their output to neurons of the next and previous layers. The second topology is the feedforward network, such as the Radial Basis Function Neural Network (RBFNN) where neurons that belong to the same layer receives inputs from neurons of the previous layer and send their values only to neurons of the next layer. Moody and Darken [2] used RBFNN by viewing the design as an approximation problem in a high dimensional space. Billings and Zheng [3] proposed RBFNN configuration with Genetic Algorithms (GA). It was demonstrated that the GA can automatically determine appropriate RBFNN

structures and parameters according to objective functions. Singh and Singh [4] utilised RBFNN with GA in satellite image classification. Delgado et al. [5] proposed the application of multivariate statistical analysis by combining GA with RBFNN, taking into account the statistical significance between independent and dependent variables and their correlation. Han et al. [6] developed an efficient self-organising recurrent RBFNN for nonlinear system modelling. Mohammadi et al. [7] provided a generic hardware solution for classification by using RBFNN on a reconfigurable data path that overcomes the major drawback of fixed-function hardware data paths. The proposed method offers limited flexibility in terms of application interchangeability and scalability, which puts no limitation on the dimension of the input data.

Logic programming began in the early 1970s as a foundation to many applications, such as artificial intelligence. The ability to extract knowledge from the logic programming makes ANN very suitable to develop an artificial intelligence system [8]. The strength of both paradigms makes the integrated system to become more "intelligent" and reduces major drawbacks of both paradigms [9]. Pinkas [10] expanded the idea of logic programming by integrating the competent propositional knowledge or logical mapping system via a symmetric connectionist network. Pursuing that, Abdullah [11] proposed a method to compute the synaptic weight of the network correspond to the propositional logic entrenched to the system. Hölldobler et al. [12] proved that the logic programming can work effectively with recurrent neural network. Sathasivam and Abdullah [13] deployed the Wan Abdullah method in order to compute the synaptic weights for the Horn Logic Programming in Hopfield Neural Network. Kasihmuddin et al. [14] deployed the advantage of the genetic algorithm to deal with learning complexity in the Hopfield neural network in doing logic k-SAT. Pursuing that, Kasihmuddin et al. [15] in their paper provided a solid evidence of the robustness of genetic algorithm to be used in other satisfiability problem by incorporated Hopfield neural network with genetic algorithm in solving MAX-kSAT problem. In their paper, Alzaeemi et al. [16] developed agent based modelling (ABM) using a genetic algorithm in the Hopfield neural network. Observed that genetic algorithm enhance the performance of doing logic programming in Hopfield network. The result has shown the effectiveness and acceleration features of genetic algorithm in doing MAX-2SAT in Hopfield network. Hamadneh et al. [17] presented the logic programming in RBFNN in a single operator logic, Horn satisfiability, and 3SAT in RBFNN. Hamadneh et al. [1] successfully represented 3SAT programming in RBFNN. The problem in RBF neural networks is the number of hidden neuron in the hidden layer and how can compute the parameters of the hidden layer? [24]. The proposed work is able to increase the efficiency of the network and minimise the required number of hidden neurons by using logic programming 2SAT to compute the all parameters in the hidden layer by give the input data in the training set. No pursuit was done to integrate 2SAT in RBFNN. In this paper, the advantages of the RBFNN and logic programming 2SAT in two training techniques, which are no-training and half-training, will be highlighted. Inevitably, RBFNN worked well with 2SAT to accommodate the number of input neurons in the input layer and reduce the number of hidden neurons in the hidden layer of RBFNN. This paper describes the implementation of 2SAT in RBFNN via no-training and half-training techniques.

2. RESEARCH METHOD

2.1. 2 Satisfiability (2SAT) Logic

2 Satisfiability or abbreviated as 2SAT problem is defined as a problem to determine the satisfiability of sets of clauses which are comprised of strictly 2 literals per clause. 2SAT problem can be expressed in the 2CNF form. The three components of 2SAT problem are as follows:

a. Consist of a set of m variables: $v_1, v_2, ..., v_m$.
b. Consist of a set of literals. A literal is a variable or a negation of a variable.
c. A set of n distinct clauses: $l_1, l_2, ..., l_n$. Each clause consists of only literals combined by only $\wedge$ that is logical AND. Each of the variable can only take a bipolar value, which is 1 or 0 that exemplifies the idea of true and false. The goal of 2SAT problem is to decide whether there exists an assignment of truth values to the variables that make the P become satisfiable. The general formula for 2SAT problem is as follows:

$$P = \bigwedge_{i=1}^{n} l_i \quad where \; l_i = \bigvee_{i=1}^{k} A_i \bigvee_{j=1}^{n} B_j, \; k = 2 \tag{1}$$

In this study. 2SAT is the main impetus since the focus of the logic programming is to ensure the programme only considers 2 literals per clause per execution. It was substantiated by various research [18-20] that numerous combinatorial problems can be formulated by using the 2SAT paradigm. The choice

of 2 literals per clause in satisfiability problem reduces the logical complexity in discovering the relation between variables in the neural network or solver. These fundamental reasons make 2SAT paradigm become a relevant approach to represent logical rules in neural network.

2.2. Radial Basis Function Neural Network (Rbfnn)

RBFNN is a neural network that was first used in 1989 [2]. Radial Basis Function Neural Network algorithm has been proven to be useful and beneficial in many industrial applications [21]. RBFNN is different from other networks, possessing several distinctive features due to its universal approximation ability, more compact topology and faster learning speed [22-23]. RBFNN typically has three layers: an input layer, a hidden layer, and an output layer. The input layer is made up of source neurons that connect the network to its environment. The hidden layer receives the input information, followed by specific decomposition, extraction, and transformation steps, to generate the output data [24]. The neurons in the hidden layer are associated with centres and widths that determine the behavioural structure of the network. The output layer in RBFNN provides the responding of the network to the activation pattern of the input layer that works as a summation unit. The parameters of a RBFNN are the centres, widths, constant input weights, and output weights. The hidden layer parameters are the centres and widths. The output weights are linked between the hidden and output layers.

The first phase of the learning procedure is to determine the hidden parameters [25]. The second phase involves output weights. There are different techniques for training RBFNN [26], such as no-training technique and half-training technique. Calculations for the equation of the activation functions for the processing on RBFNN [27] are as follows:

$$\varphi_i(x) = \exp\left(-\left(\frac{\left\|\sum_{j=1}^{N} w_{ji}^{'} x_j - c_i\right\|^2}{2\sigma_i^2}\right)\right) \tag{2}$$

where φ_i is the activation function of RBFNN in hidden neuron I, $w_{ji}^{'}$ is the constant input weight between the input neuron, j and the hidden neuron I, c_i is the centre of the hidden neuron I, σ_i is the width of the hidden neuron I, N is the number of the input neurons, x_j is an input value in φ_i, Structure of RBFNN $\|\ \|$ indicates the Euclidean norm on the input space as follows and Structure of RBFNN as shown in Figure 1.

$$\left\|\sum_{j=1}^{N} w_{ji}^{'} x_j - c_i\right\| = \sqrt{\sum_{i=1}^{m}\left(\sum_{j=1}^{N} w_{ji}^{'} x_j - c_i\right)^2} \tag{3}$$

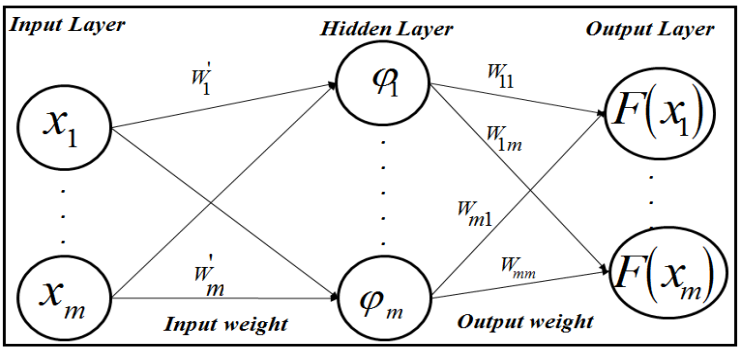

Figure 1. Structure of RBFNN

Accordingly, the output of RBFNN technique has the following form [31]:

$$F(x)=\sum_{i=1}^{j} w_i \varphi_i(x) \tag{4}$$

where w_i is the output weight between the hidden neuron and the output neuron, $F(x)$ is the actual output value of an output neuron.

2.2.1. Satisfiability Radial Basis Function Neural Network

The objective of doing logic programming 2SAT in RBFNN is to find the truth value of the input values in RBFNN for the clauses. The general form of the logic programming 2SAT is [17]:

$$P_{2SAT} = \bigvee_{i=1}^{k} A_i \bigvee_{j=1}^{n} B_j \tag{5}$$

where, $k, n \in \mathbb{N}$. A_i and B_j are atoms. To embed the logic programming 2SAT in RBFNN, the training data set for each clause must be determined. The general form to calculate the training data for each clause in 2SAT is given by the following equation:

$$x=\sum_{i=1} I(A_i)+\sum_{j} I(B_j);\ where\ A_i, B_j \in \{0,1\} \tag{6}$$

The logic programming of 2SAT in RBFNN, is by using binary input neurons $\{0,1\}$ in Table 1 to exemplify the possible input data values and output target data values for 2SAT in RBFNN. By using the embedding method, input data value for (7) is shown in Table 1.

For example, the following 2SAT that has three clauses in the RBFNN training:

$$\begin{aligned} P &= A, B \leftarrow \\ &\quad D \leftarrow C \\ &\quad E \leftarrow F \end{aligned} \tag{7}$$

Table 1. The Input Data Form and the Training Data of 2SAT in (7)

Clause	$A, B \leftarrow$			$D \leftarrow C$			$E \leftarrow F$		
DNF	$A \vee B$			$\neg C \vee D$			$E \vee \neg F$		
Input the value of data form x	$x = A + B$			$x = D - C$			$x = E - F$		
Input data in the training set x_i	0	1	2	-1	0	1	-1	0	1
Output target data y_i	0	1	1	0	1	1	0	1	1

2.4. Radial Basis Function Neural Network 2 Satisfiability via No-Training Technique (RBFNN-2SATNT)

For no-training techniques in RNFNN-2SAT, all parameters, such as the centres, the widths, and output weights are calculated and fixed. The RBFNN-2SAT training in this technique is by metaheuristic algorithms to minimise the error in output weight between hidden neurons and output neurons. The algorithm for RBFNN-2SAT by using the no-training technique is as follows:

Step 1: Calculate the training data for each clause in 2SAT by taking the centres as training data of input values.

Step 2: Initialise the input weight $w_{ji}^{'} \in \{-1,1\}$ between input neuron and hidden neuron beside 1 for atom and -1 for the negative atom.

Step 3: Calculate the widths by using the following equation according to the paper [28].

$$\sigma = \frac{d_{\max}}{\sqrt{2}\, m_i} \tag{8}$$

where, m_i is the number of the literals per clauses and $d_{\max}$ is the maximum distance between the centres of the hidden layer.

Step 4: Calculate the RBFNN-2SAT Gaussian activation function values by using (2).

Step 5: Obtain the output weight by using the output linear (4) of RBFNN-2SAT.

Table 2. The Values of the Activation Function in RBFNN-2SAT which Represents the Clause $A, B \leftarrow$

Input values (x_i)	$\varphi_i(x,0,0.074)$	$\varphi_i(x,1,0.074)$	$\varphi_i(x,2,0.074)$	Output target value y_i
0	1	1.3525e-6	3.3675e-24	0
1	1.3525e-6	1	1.3525e-6	1
2	3.3675e-24	1.3525e-6	1	1

Table 3. The Values of the Activation Function in RBFNN-2SAT which Represents the Clause $D \leftarrow C$

Input values (x_i)	$\varphi_i(x,-1,0.074)$	$\varphi_i(x,0,0.074)$	$\varphi_i(x,1,0.074)$	Output target value y_i
-1	1	1.3525e-6	3.3675e-24	0
0	1.3525e-6	1	1.3525e-6	1
1	3.3675e-24	1.3525e-6	1	1

Table 4. The Values of the Activation Function in RBFNN-2SAT which Represents the Clause $E \leftarrow F$

Input values (x_i)	$\varphi_i(x,-1,0.074)$	$\varphi_i(x,0,0.074)$	$\varphi_i(x,1,0.074)$	Output target value y_i
-1	1	1.3525e-6	3.3675e-24	0
0	1.3525e-6	1	1.3525e-6	1
1	3.3675e-24	1.3525e-6	1	1

Table 5. The Output Weights of RBFNN-2SATNT that Represents 2SAT using the No-Training Technique

W_{11}	W_{21}	W_{31}	W_{12}	W_{22}	W_{32}	W_{13}	W_{23}	W_{33}
-1.3525e-06	1	1	1	-1.3525e-06	1	1	-1.3525e-06	1

In the no-training technique, the hidden neuron centre values in hidden layers should be equal to the training data value that was obtained from 2SAT. Structure of the RBFNN-2SAT with the hidden parameters, by using no-training technique as shown in Figure 2.

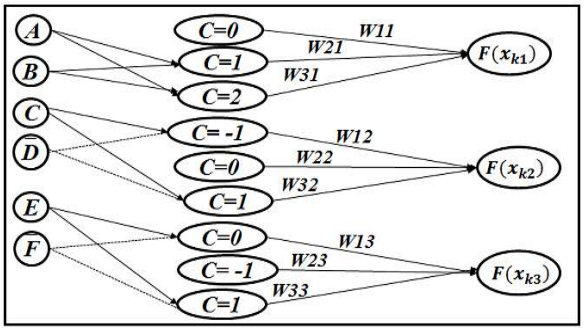

Figure 2. Structure of the RBFNN-2SAT with the hidden parameters, by using no-training technique

2.5. Radial Basis Function Neural Network 2 Satisfiability via Half Training Technique (RBFNN-2SATHT)

Half-training is a hybrid learning that involves two phases. The first phase is called unsupervised learning, and the second phase is called supervised learning. By determining the parameters (the centres, widths, and output weights) in RBFNN-2SAT by using k-means clustering algorithm.

The first phase of unsupervised training from k-means clustering was used to determine the centre and the width [17]. Then the second phase is called supervised learning algorithm to adjust the output weights by using Genetic Algorithm (GA).

2.5.1. *k*-means Clustering Algorithm

The k-means algorithm partitions a collection of N vectors into clusters. The k-means clustering algorithm is to determine the hidden parameters (the centres and the widths) in half-training technique by Hamadneh et al. [17]. Equation (10) finds the cluster centres by minimising the distance function:

$$d\left(x_j, c_i\right) = \left\| x_j - c_i \right\| \tag{9}$$

where, $d\left(x_j, c_i\right)$ is the distance between the centre c_i and data point x_j. The new optimal centres will be obtained by (10):

$$c_i = \frac{1}{m_i} \sum x_i \tag{10}$$

where, c_i is the centre, m_i is the number of the literals per clauses. After calculating the centres by using the k-means clustering algorithm, the width for each hidden neuron is calculated by (12) in the paper [17]:

$$\sigma_i^2 = \frac{1}{m_i} \sum \left[d\left(x_j, c_i\right) \right]^2 \tag{11}$$

where, σ_i is the width of hidden neuron, m_i is the number of the literals per clauses. The Gaussian activation function values by using Equation are as shown in Table 7 and Structure of the RBFNN-2SAT with the hidden parameters, by using half-training technique as shown in Figure 3.

Table 6. Training Set of the 2SAT, and the Radial Basis Function Values by using Gaussian Function

i	Input value x_i	Output target value y_i	$\varphi_i(x, 0.5, 1.125)$	$\varphi_i(x, 1, 1)$	$\varphi_i(x, 1.5, 1.375)$	$\varphi_i(x, 0, 0.074)$	$\varphi_i(x, 0.5, 1.375)$
1	0	0	0.8948	0.6065	0.4412	0.84208	0.9131
2	1	1	0.8948	1	0.9131	0.9131	0.9131
3	2	1	0.367879	0.6065	0.9131	0.4412	0.4412
4	1	1	0.8948	1	0.9131	0.9131	0.9131
5	1	1	0.8948	1	0.9131	0.9131	0.9131

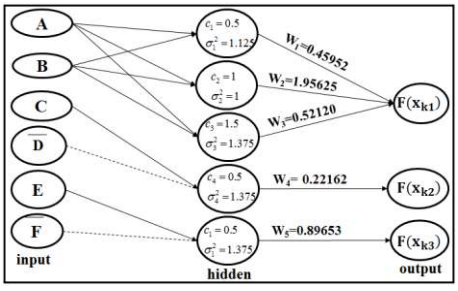

Figure 3. Structure of the RBFNN-2SAT with the hidden parameters, by using half-training technique

2.5.2. Genetic Algorithm in RBFNN-2SAT

Genetic Algorithm (GA) is one of the well-known and widely used metaheuristic solution algorithms for optimisation problems. GA has the ability to minimise multi-objective optimisation problems [29-30]which is very vital in this study. GA investigates the entire search space by locating the global solution and locally reaching the optimal solution.

In this section, the RBFNN-2SAT in both techniques. In RBFNN-2SAT by using no-training technique will be optimised by GA through minimising the error produced from the training. In RBFNN-2SAT by using half-training technique, the network is required to find the output weight by solving the linear equation that consists of RBFNN-2SAT output. In this case, GA can be used in RBFNN-2SAT to improve the solution of the RBFNN-2SAT half-training technique output weight and minimising the error. The solution (output weight) in RBFNN-2SAT generated by GA is called a chromosome.

Step 1. Initialisation. The population of chromosomes will be initialised.

Step 2. Evaluation. Computes the objective function value for each chromosome produced in initialisation by the following equation:

$$f(x) = \sum w_i \varphi_i(x) - y_i \tag{12}$$

where, w_i is the output weight between the hidden neuron and the output neuron , $\varphi_i(x)$ is the Gaussian activation function in RBFNN. y_i is the output target value.

Step 3. Selection: The fittest chromosomes have higher probability to be selected for the next generation. To compute the fitness probability, the fitness of each chromosome must be computed to avoid divide by zero problem, and the value of objective function is added by 1.

$$fit_i = \frac{1}{(1 + f(x))} \tag{13}$$

fit_i is the fitness and $f(x)$ is the objective function. The probability (p_i) for each chromosomes is formulated by:

$$p_i = \frac{fit_i}{\sum_{i=0}^{n} fit_i} \tag{14}$$

Step 4: Crossover: The number of cross-populations is determined according to the crossover rate. During crossover, information of the parent's chromosome will be exchanged to produce new chromosomes.

Step 5. Mutation: Number of chromosomes that have mutations in a population is determined by the mutation rate parameter. In mutation phase, the information of the chromosomes will randomly change one value of the chromosomes (the output weight) to improve the solution (output weight) of RBFNN.

Step 6: Termination: If the given evolution termination criteria are met, the calculation is stopped, otherwise go to Step 2 with i=i+1 (GA will undergo up to 10000 iterations (Generations) to improve the chromosomes (output weights) until the output of RBFNN becomes less or equals to target output.).

3. EXPERIMENTAL SETUP

In this study, the proposed GA with 2SAT in the Radial Basis Function Neural Network-2SAT No-Training (RBFNN-2SATNT) and the Radial Basis Function Neural Network-2SAT Half Training (RBFNN-2SATHT). The experiment was run for various numbers of neurons (NN) from 6 up to 96, depending on the number of literals per clause (NC1, NC2, NC3). The network complexity increases as the number of neuron (NN) increases. The population size was 50, the crossover rate was 1, the mutation rate was 1, and the maximum number of GA generation was 10000. In addition, simulations were performed on Microsoft Visual C# 2010 Express software. The experiments were done by using Microsoft Window 7 Professional 64-bit, with the specification of 500GB hard disk, 4096MB RAM, and processor 3.40GHz.

4. RESULTS AND DISCUSSION:

The result of the RBFNN-2SATNT and RBFNN-2SATHT are summarised in Figure 4 to Figure 7. Based on these Figures, RBFNN-2SATHT outperforms RBFNN-2SATNT in terms of errors (RMSE, SSE, MAPE, and MAE). When the number of neurons increased, RBFNN-2SATHT was able to sustain more neurons. Theoretically, the increase in the number of hidden neurons leads to an increase in the rate of errors due to the centres as a subset of the input values of the training with RBFNN-2SATNT. It is also can be deduced that, most solutions in RBFNN-2SATNT have lower fitness and require more iterations compared to RBFNN-2SATHT. RBFNN-2SATHT is the best model because it has lower values of RMSE, SSE, MAPE and MAE. Figure 8 illustrates the number of hidden neuron result for RBFNN-2SATNT and RBFNN-2SATHT.

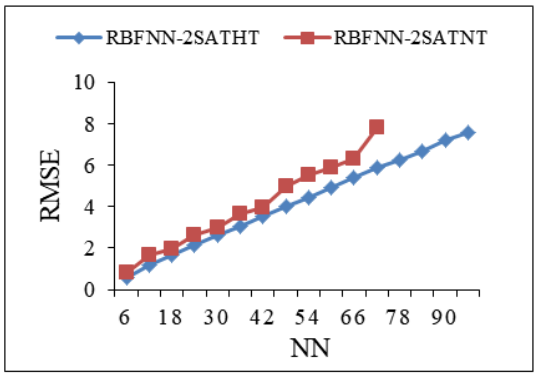

Figure 4. RMSE value for all RBFNN-2SAT models

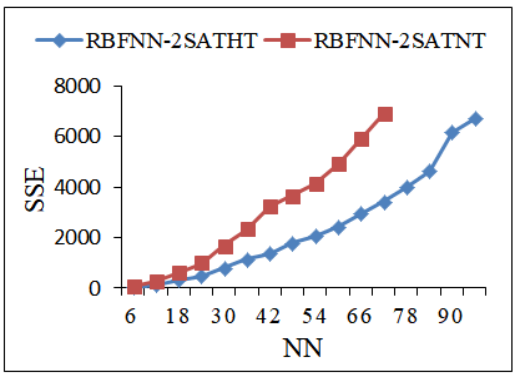

Figure 5. SSE value for RBFNN-2SAT models

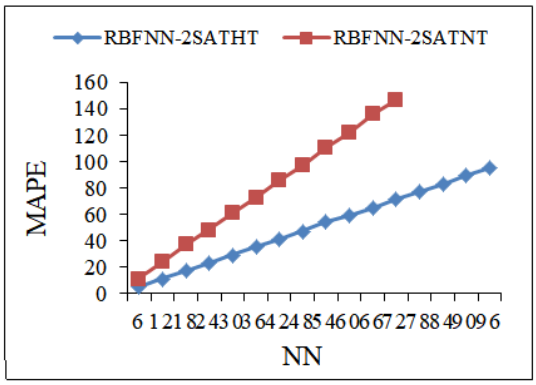

Figure 6. MAPE value for RBFNN-2SAT models

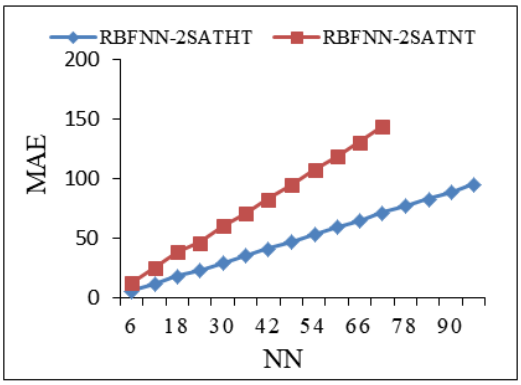

Figure 7. MAE value for RBFNN-2SAT models

Based on Figure 8, magnitude of the number of the hidden neuron for RBFNN-2SATNT dramatically increased as the number of 2SAT clauses increased. RBFNN model that acquired the least value of the hidden neurons was considered as the best model among all neural network models. According to Panchal et al., [24], the lowest value of the hidden neuron indicated the best model for a given number of neurons. When the network has many neurons in the hidden layer, over fitting may result (the weights became extremely large because the hidden neurons in the hidden layer were highly correlated with each other). This means the capacity of the processing system was not enough to train all the neurons in the hidden layer. Based on the results, RBFNN-2SATHT has the least value of the hidden neuron compared to RBFNN-2SATNT. In this case, RBFNN-2SATHT in comparison with RBFNN-2SATNT is considered the best technique in the RBFNN-2SAT model.

Figure 9 illustrates the computation time or CPU time result for RBFNN-2SATNT and RBFNN-2SATHT. The computation time or CPU time is defined as the time taken for a RBFNN-2SAT to generate the best solutions, including the training process. According to Figure 9, the computation time for the RBFNN-2SATHT was faster than RBFNN-2SATNT. RBFNN-2SATNT took a longer time for the network

to specify the number of hidden neurons and the output between the hidden layer and output layer, as the network spends more time in the training state. Hamadneh et al. [17] mention RBFNN by no training technique took longer time due to the value of the centres are equal to the values of the training data. Also, many neurons in the hidden layer cause the appearance of fake output weight as shown in RBFNN-2SAT by no-training technique that will increase the time it takes to train the network. However, when RBFNN-2SAT by half-training technique was implemented, the computation time was faster. In Figure 9, RBFNN-2SATHT is faster because the centres were not restricted to be the input of data points and width of the hidden layer that was calculated by using k-means clustering.

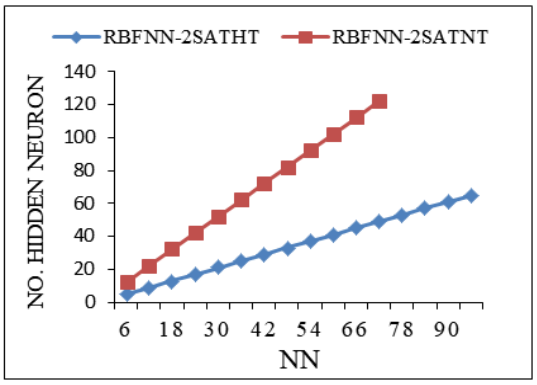

Figure 8. No. Hidden neuron for RBFNN-2SAT models

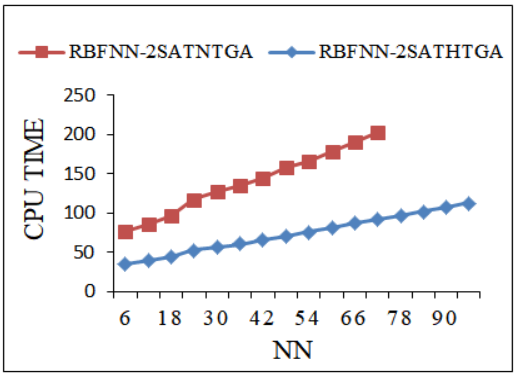

Figure 9. CPU Time for RBFNN-2SAT models

5. CONCLUSION

The paper presents 2SAT logic programming in RBFNN. The power of this technique can be seen clearly when computing the stability by using the RBFNN. In this study, the principal components were RBFNN, GA, and 2SAT. To determine the RBFNN parameters and calculate the errors represented, the two techniques were proposed namely, RBFNN-2SATNT and RBFNN-2SATHT. Results showed that the half-training technique (RBFNN-2SATHT) was more powerful and more effective than the no-training technique (RBFNN-2SATNT), due to the number of hidden neurons was typically much less than the number of data points, and the centres were not restricted to be data points.

ACKNOWLEDGMENT

This research is supported by Universiti Sains Malaysia and Fundamental Research Scheme (FRGS) (6711689) by Ministry of Higher Education Malaysia.

REFERENCES

[1] N. Hamadneh, *et al.,* "Optimization of microchannel heat sinks using prey-predator algorithm and artificial neural networks," *Machines*, vol. 6(2), pp. 26-36, 2018.
[2] J. Moody, and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural computation*, vol. 1(2), pp. 281-294, 1989.
[3] S. A. Billings, and G. L. Zheng, "Radial basis function network configuration using genetic algorithms," *Neural Networks,* vol. 8(6), pp. 877-890, 1995.
[4] A. Singh, and K. K. Singh, "Satellite image classification using genetic algorithm trained radial basis function neural network, application to the detection of flooded areas". *Journal of Visual Communication and Image Representation*, vol. 42, pp. 173-182, 2017.
[5] De Leon-Delgado, *et al,* "Multivariate statistical inference in a radial basis function neural network," *Expert Systems with Applications*, vol. *93*, pp. 313-321, 2018.
[6] H. G. Han, *et al*, "Nonlinear system modeling using a self-organizing recurrent radial basis function neural network," *Applied Soft Computing*, vol. *71*, pp. 1105-1116, 2018.
[7] M. Mohammadi, *et al,* "A hardware architecture for radial basis function neural network classifier," *IEEE Transactions on Parallel and Distributed Systems*, vol. *29*(3), pp. 481-495, 2018.
[8] J. W Lloyd," Foundations of logic programming," *Springer Science & Business Media*, 6 Dec 2012.
[9] W.A.T.W. Abdullah, "The logic of neural networks," *Physics Letters A*, vol. *176*(3), pp. 202-206, 1993.

[10] G. Pinkas, "Symmetric neural networks and propositional logic satisfiability". *Neural Computation*, vol. 3(2), pp. 282-291, 1991.
[11] W. A. T. W. Abdullah, "Logic programming on a neural network," *International journal of intelligent systems*, vol. *7*(6), pp. 513-519, 1992.
[12] S. Hölldobler, *et al,* "Approximating the semantics of logic programs by recurrent neural networks," *Applied Intelligence*, vol. *11*(1), pp. 45-58, 1999.
[13] S. Sathasivam, and W. A. T. W. Abdullah, "Logic mining in neural network: reverse analysis method," *Computing*, vol. 91(2), pp. 119-133, 2011.
[14] M. S. M. Kasihmuddin, et al., "Genetic Algorithm for Restricted Maximum k-Satisfiability in the Hopfield Network," *International Journal of Interactive Multimedia & Artificial Intelligence*, vol. 4(2), 2016.
[15] M. S. M. Kasihmuddin, et al., "Robust Artificial Bee Colony in the Hopfield Network for 2-Satisfiability Problem," *Pertanika Journal of Science & Technology*, vol. *25*(2), pp. 453-468, 2017.
[16] S. A. Alzaeemi, *et al,* "Use of Genetic Algorithm for Hopfield Neural Network to do Logic Programming," *In Bio-Genetics Journal,* vol. 5(1), pp. 101-113, 2017.
[17] N. Hamadneh, *et al*, "Learning logic programming in radial basis function network via genetic algorithm," *Journal of Applied Sciences (Faisalabad)*, vol. *12*(9), pp. 840-847, 2012.
[18] S. Mukherjee, and S. Roy, "*Multi terminal net routing for island style FPGAs using nearly-2-SAT computation*," In VLSI Design and Test (VDAT), 2015 19th International Symposium on, (pp. 1-6). IEEE, 2015.
[19] S. Even, *et al*, "*On the complexity of time table and multi-commodity flow problems*," In 16th Annual Symposium on Foundations of Computer Science, (pp. 184-193). IEEE, October, 1975.
[20] R. Miyashiro, and T. Matsui, "A polynomial-time algorithm to find an equitable home–away assignment," *Operations Research Letters*, vol. 33(3), pp. 235-241, 2005.
[21] M., Tayyab, J., Zhou, R., Adnan, C., Meng, & A. Zahra, (2016). Streamflow prediction by applying generalized regression network with time series decomposition method. *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, 4(3), 611-616.
[22] P. Kang, and Z. Jin, "Neural network sliding mode based current decoupled control for induction motor drive". *Information Technology Journal*, vol. *9*(7), pp. 1440-1448, 2010.
[23] H. Zayandehroodi, *et al*, "Automated fault location in a power system with distributed generations using radial basis function neural networks," *Journal of Applied Sciences (Faisalabad)*, vol. 10(23), pp. 3032-3041, 2010.
[24] G. Panchal, Ganatra, *et al*, "Behaviour analysis of multilayer perceptronswith multiple hidden neurons and hidden layers," *International Journal of Computer Theory and Engineering*, vol. 3(2), pp. 332, 2011.
[25] M. T. Vakil-Baghmisheh, and N. Pavešić, "Training RBF networks with selective backpropagation," *Neurocomputing*, vol. 62, pp. 39-64, 2004.
[26] D. C. Dhubkarya, and D. Nagaria, "Implementation of a radial basis function using VHDL" *Global Journal of Computer Science and Technology*, vol. 10(10), 2010.
[27] N. Mahmod, N. A. Wahab (2017). Fouling Prediction Using Neural Network Model for Membrane Bioreactor System. *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, 6(1), 200-206.
[28] Y. S. Hwang, and S. Y. Bang, "An efficient method to construct a radial basis function neural network classifier," *Neural networks*, vol. 10(8), pp. 1495-1503, 1997.
[29] K. A. De Jong, and W. M. Spears, "Using genetic algorithms to solve NP-complete problems," In *ICGA Journal* (pp. 124-132), June 1989.
[30] W. Jia, *et al,* "A new optimized GA-RBF neural network algorithm," *Computational intelligence and neuroscience*, 2014, vol. (44), pp. 1-6, 2014.
[31] Y. U. Zhijun, (2013). RBF neural networks optimization algorithm and application on tax forecasting. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 11(7), 3491-3497.

BIOGRAPHIES OF AUTHORS

Shehab Abdulhabib Alzaeemi received a Bachelor Degree of Education (Science) from Taiz Universiti in 2004 and Master of Science (Mathematics) from Universiti Sains Malaysia in 2016 and now student PhD in Universiti Sains Malaysia. He was fellow under the Academic Staff Training System of Sana'a Community College from 2005-2014. His research interests mainly focus on neural network, logic programming, and data mining.

Mohd Shareduwan Mohd Kasihmuddin is a lecturer in School of Mathematical Sciences, Universiti Sains Malaysia. He received his Ph. D from Universiti Sains Malaysia. His current research interests include Metaheuristics method, neural network development, artificial inteligence and logic programming.

Mohd. Asyraf Mansor is a lecturer in the School of Mathematical Sciences, Universiti Sains Malaysia. He received his Ph.D from Universiti Sains Malaysia. His current research interests include evolutionary algorithm, satisfiability problem, neural networks, logic programming and heuristic method.

Saratha Sathasivam is an Associate Professor in the School of Mathematical Sciences, Universiti Sains Malaysia. She received her MSc and BSc(Ed) from Universiti Sains Malaysia. She received her Ph.D at Universiti Malaya, Malaysia. Her current research interest are neural networks, agent based modeling and constrained optimization problem..

Mustafa bin Mamat is a Professor in the Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin. His current research interests include computational mathematics, convergence analysis and evolutionary algorithm.