TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol. 13, No. 2, February 2015, pp. 369 ~ 378
DOI: 10.11591/telkomnika.v13i2.7032

Radial Basis Function Network Learning with Modified Backpropagation Algorithm

Usman Muhammad Tukur, Siti Mariyam Shamsuddin*
Soft Computing Research Group, Faculty of Computing, Universiti Teknologi Malaysia
81310 Johor, Malaysia
*Corresponding author, e-mail: mtusmaan@yahoo.com, mariyam@utm.my

Abstract

Radial Basis Function Network (RBFN) is a class of Artificial Neural Network (ANN) that was used in many classification problems in science and engineering. Backpropagation (BP) algorithm is a learning algorithm that was widely used in ANN. However, BP has major disadvantages of slow error rate convergence and always easily stuck at the local minima. Hence, Modified BP algorithm was proposed in this study to improve the learning speed of RBFN using discretized data. C programming language was used to develop the program for the proposed method. Performance measurement of the method was conducted and the experimental results indicate that our proposed method performs better in error rate convergence and correct classification compared to the result with continuous dataset. T-test statistical analysis was used to check the significance of the results and most were found to be satisfactorily significant.

Keywords*: ,odified backpropagation, cost function, discretization, radial basis function network*

1. Introduction

Artificial Neural Network (ANN) was developed as a parallel distributed system inspired by human brain learning process. ANN learning capacity to solve problems through training makes it popular. There are many types of ANN such as Backpropagation Network, Self-Organizing Map (SOM), Spiking Neural Network, Radial Basis Function Network (RBFN), etc. RBFN is a class of ANN that employs Radial Basis Functions (RBFs) as activation functions. RBFN is a unique type of ANN with only three layers. It has only one hidden layer unlike other types of ANN that have one or more hidden layers. It is a feed forward and fully connected network [1]. RBFN has several benefits above its predecessors. Some of the benefits include having simple network architecture, ability to approximate better and also its algorithms learn faster. RBFNs are used to solve problems such as classification problems, function approximation, system control, and time series prediction. As a result, RBFN it is widely used in science and engineering. In this paper, we are applying the Modified Backpropagation (MBP) algorithm [2] with improved learning parameter value to train RBFN with discretized data. Thus, this study attempts to explore the performance of RBFN by determining values for error rate convergence or learning rate and correct classification accuracy of the network. The rest of the paper is organized into sections. Section 2 is the Proposed Method; section 3, The Research Method. In section 4, Results and Discussion of the findings are given. The Conclusion of the study is given in section 5, and finally the References consulted are listed in section 6.

2. Proposed Method

In RBFN training, clustering algorithms are used to determine the centre and weight of the hidden layer. On the other hand, Least Mean Squares (LMS) algorithms are employed in determining the network weights between hidden and output layer [3]. Weights of network neurons thou can be adjusted have a value of 1 between input layers and hidden layers [1]. The hidden layer nodes determine behaviour and structure of the network. Gaussian function is used to activate the hidden layer [4].

The numerous synaptic associations that link the billions of neurons in animals are modified to understand new things. Likewise, an ANN behaves in the same way by mimicking

Received November 15, 2014; Revised December 28, 2014; Accepted January 14, 2015

the neurons in human brain. It modifies its weights to enable it learn new patterns. Therefore, ANN is a machine learning process inspired, by our brain. Every node of the artificial neuron has an activation function which is in charge of mapping the inputs of the neuron to its output [5]. Backpropagation (BP) algorithm has been widely used as training algorithm for ANN in supervised learning mode [6]. A study by [7] reported that at present ANN have numerous real life applications in machine learning and other fields.

RBFN is trained using unsupervised and supervised learning modes. The unsupervised mode is implemented between input to hidden layers and supervised mode is implemented from hidden to output layer. Functions given by hidden neurons create random starting point for input patterns [8]. Clustering algorithms are capable of finding cluster centres that best represents the distribution of data in the first stage of the training. Supervised learning is guided by the desired target values for the network. Using modified BP to train the network with modified cost function proved by other studies to have enhanced ANN learning is implemented [2] but in our case, discretized datasets was used which also was reported to enhance the speed of learning [9]. In RBFN, each layer of the network performs a different tasks and that is one of the main problems with this network. Therefore, separating the processes of the layers with various techniques is a good idea. In this paper, five standard classification problems dataset was used to test the proposed algorithm. The study compared the result of the proposed algorithm: RBFN learning with Modified BP algorithm (MBP-RBFN) using continuous and discretized datasets.

2.1. Backpropagation Algorithm

Backpropagation Algorithm (BP) takes the lead in training ANN. It is one of the supervised learning algorithms that use gradient descent technique to reduce the cost function. Many factors affect the performance of BP algorithm, such as initial parameters, initial weight, rate of learning, momentum term, size of network, number of epoch, connection between the units. The learning performance of BP depends on network parameters. A good choice of these parameters may greatly improve performance. It generally takes BP a long time to train [10]. Other BP learning weaknesses are local minima trapping and slow rate of convergence [11], [12]. As a result of these limitations, researchers have been working hard to improve BP performance.

The BP training process starts at the input layer (*i*) through the hidden layer (*j*) to output layer (*k*) as explained in equations below. The network activation function applied is sigmoid function.

Between layer *(i)* and layer *(j)*

$$O_j = f(net_j) = \frac{1}{1+e^{-net_j}} \quad (1)$$

$$net_j = \sum_j W_{ij}\, O_i + \theta_j \quad (2)$$

Where "O_j is output of layer *j,* O_i is output of layer *i,* W_{ij} is weight connecting layer *i* to *j* and θ_j is the bias at layer *j"*

Between layer (*j*) and layer (*k*)

$$O_k = f(net_k) = \frac{1}{1+e^{-net_k}} \quad (3)$$

$$net_k = \sum_k W_{jk}\, O_j + \theta_k \quad (4)$$

Where "O_k is output of layer *k,* O_j is output of layer *j,* W_{jk} is weight connecting layer *j to k, and* θ_k is the bias at layer *k"*

Between layer (*j*) and (*k*)

The error is computed using (5) to determine the differences of desired output and actual output. Error is then propagated backward. The propagation starts at layer *(k)* to *(j)* then finally to layer *(i).* The weights are adjusted to decrease the error as it is backward propagated through the layers.

$$MSE = \frac{1}{N}\sum_{l=1}^{N}(Target\ Output\ -\ Actual\ Output)^2 \quad (5)$$

From the error computed, BP is used from (*k*) to (*j*) as in Equation (6) and (7).

$$w_{kj}\,(t+1) = w_{kj}\,(t) + \Delta w_{kj}\,(t+1) \quad (6)$$

$$\Delta w_{kj}\,(t+1) = \eta\delta_k O_j\,(t) + \alpha\Delta w_{kj}\,(t) \quad (7)$$

With,

$$\delta_k = O_k(1 - O_k)\,(t_k - O_k) \quad (8)$$

Where "$w_{kj}\,(t)$ represent the weight from layer *k* to layer *j* at time *t,* $\Delta w_{kj}\,(t)$ represent the weight change, η represent the learning rate, α represent momentum rate, δ_k represent error at node *k,* O_j represent the actual network output at layer *j,* O_k represent the actual network output at layer *k,* t_k represent the target output value at layer *k."*

Equations (9) and (10) are for backward calculations from layer (*j*) to (*i*).

$$w_{ji}\,(t+1) = w_{ji}\,(t) + \Delta w_{ji}\,(t+1) \quad (9)$$

$$\Delta w_{ji}\,(t+1) = \eta\delta_j O_i\,(t) + \alpha\Delta w_{ji}\,(t) \quad (10)$$

With,

$$\delta_j = O_j(1 - O_j)\,\textstyle\sum_k \delta_k\, w_{kj}) \quad (11)$$

$$O_k = f(net_k) = \frac{1}{1+e^{-net_k}} \quad (12)$$

$$net_k = \textstyle\sum_k W_{jk}\, O_j + \theta_k \quad (13)$$

Where "$w_{ji}\,(t)$ is the weight from *j* to *i* at time *t,* $\Delta w_{ji}\,(t)$ is the weight adjustment, η is the learning rate, α is the momentum rate, δ_j is error at node *j,* δ_k is error at node *k,* O_i is the actual network output at node *i,* O_j is the actual network output at node *j,* O_k is the actual network output at node *k,* w_{kj} is the weight connected between node *j* and *k,* θ_k is the bias of node *k."*

This procedure is reiterated till convergence is attained. During training, standard BP normally uses learning rate which is the amount of correction applied to adjust weights at the time of training and momentum term. It is used to set the rate of adjustment and is always within the range of [0, 1]. The small value of learning rate will make the changes of weight very small. Momentum term is the amount of previous corrective term that should be remembered. It is used to speed up the propagation, and is within [0.1, 1.0] range. Hence, these two parameters are used generally to monitor the adjustment of the weight [6] and [13].

Sensitivity of BP to the learning rate value makes it runs very slow. That is why BP is always locked in local minima. But, optimum convergence speed of BP can be obtained by tuning the learning parameter. A study by [14] proposed an adaptive learning rate based on Barzilai and Borwein learning rate update for steepest descent method, and the result showed that the proposed method improves the convergence rates of BP algorithm. Therefore, from the previous studies, it shows that in BP learning, it is essential to pick the learning constant correctly.

2.2. Cost Function

The error calculations employed to train an ANN are significant [15]. Also, [16] reported that output error measurement gives us a way to update the weights iteratively to reduce error. The cost function is interchangeably called objective or error function. Mean Square Error (MSE) is widely used in weight adjustment in BP training.

$$MSE = \frac{1}{N}\textstyle\sum_{i=1}^{N}(Desired - Actual)^2 \quad (14)$$

Where, *Desired r*epresent target value and *Actual* represent real value generated by network. The "error signal" δ_k is defined as:

$$\delta_k = -(T_k - O_k)O_k(1 - O_k) \quad (15)$$

Where, T_k represent target output and O_k represent the actual output of the network.

However, in this study, modified cost function (MM) in [2] defined in (16) is used with MBP-RBFN in place of the commonly used MSE.

$$mm = \sum_k \rho_k \quad (16)$$

With,

$$\rho_k = \frac{E_k^2}{2a_k(1-a_k^2)} \quad (17)$$

Where "$E_k = t_k - a_k$, and, E_k is the error at *(k)*,t_k is the target value at *(k),* a_k is the activation of *(k)"*

Applying partial derivatives using chain rule, for weight updating, an error signal is generated for MBP for the output layer as,

$$\delta_k = \frac{2(E+\rho(1-3a_k^2))}{1+a_k} \quad (18)$$

While MBP error signal for hidden layer is same as SBP,

$$\delta_k = \sum \delta_k w_j \sigma'(a_j) \quad (19)$$

Where "w_j is the weight of connection, $\sigma(a_j)$ a sigmoid function of (1/1+e $^{-2x}$)".

2.3. Radial Basis Function Network

RBFs are used as activation functions in the hidden layer of RBFN. The link between inputs to hidden layer is not weighted while the link between hidden to output layer is weighted. The hidden layer neurons provide *functions* set to makeup random *basis* for input patterns [17].

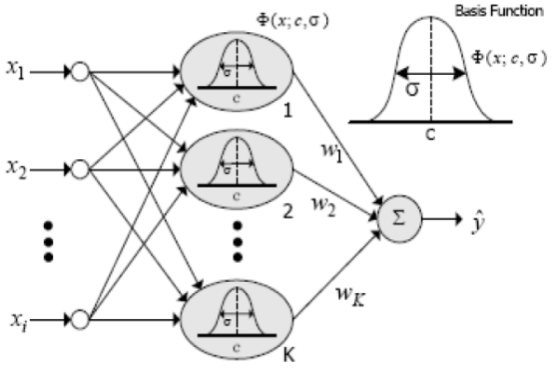

Figure 1. The RBF Network Structure [18]

The output $y_j(i)$ of the j^{th} hidden layer in a RBFN model, is as [4]:

$$y_j(i) = \sum_{k=1}^{n} w_{jk} \Phi_k[x(i), c_k, \sigma_k] \quad (20)$$

$$j = 1,2, \ldots\ldots, n \qquad i = 1, 2, \ldots, N$$

Here *k* represents the RBFs applied, while $C_k \in R^m$, $\sigma^k \in R^m$ are centre and width value vectors of RBF, given as:

$$\boldsymbol{C_k} = [\boldsymbol{C_{k1}}\ \boldsymbol{C_{k2}}\ \ldots\ \boldsymbol{C_{km}}]^{\boldsymbol{T}} \in \boldsymbol{R^m}, \boldsymbol{k} = \boldsymbol{1}, \ldots . \boldsymbol{k} \quad (21)$$

$$\boldsymbol{\sigma_k} = \boldsymbol{\sigma_{k1}}\ \boldsymbol{\sigma_{k2}} \ldots\ \boldsymbol{\sigma_{km}}]^{\boldsymbol{T}} \in \boldsymbol{R^m}, \boldsymbol{k} = \boldsymbol{1}, \ldots, \boldsymbol{k}, \quad (22)$$

Which are weights of RBFs linked to *jth* output.

RBFN implementation is represented in two layered network form. Non-linear transform is implanted by the input layer. While in the hidden layer, each term Φ_k (.) is an activation function. The output layer on the other hand applies linear transformation.

$$\Phi_{\mathrm{k}}(\mathrm{x}, \mathrm{c_k}, \sigma_{\mathrm{k}}) = \prod_{\mathrm{i}=1}^{\mathrm{m}} \phi_{\mathrm{ki}}\ (\mathrm{x_i}, \mathrm{c_{ki}}, \sigma_{\mathrm{ki}}) \quad (23)$$

Also, the most common option for $\boldsymbol{\phi}$ (.) is the Gaussian form below in Equation (19).

$$\phi_{\mathrm{ki}}(\mathrm{x_i}, \mathrm{c_{ki}}, \sigma_{\mathrm{ki}}) = \exp[-(\mathrm{x_i} - \mathrm{c_{ki}})^2 / 2\sigma_{\mathrm{ki}}^2] \quad (24)$$

Equation (20) output becomes (25)

$$yj(i) = \sum_{k=1}^{K} w_{jk}\ \exp\{-\sum_{i=1}^{m}[(\ (x_i - c_{ki})^2 / 2\sigma_{ki}^2)]\} \quad (25)$$

A study by [19] proposed a new tree based RBFN model combining logistic regression, aimed to enhance its classification performance by input data preprocessing using RBFN frame. The proposed method was binary classification and was easy to generalize for many problems. The model was tested with data obtained from "hydraulic fracturing in Oil and Gas wells", and the results are superior to logistic regression.

Data from 1200 individuals was collected by [20], for empirical comparison of the network models. Fasting plasma glucose (FPG) as criteria for classifying a patient as a diabetic was used in selecting diabetic patients. Risk parameters employed are gender, age and family history of diabetes; and many other factors are also considered. The study used RBFN in diagnosing of diabetes mellitus and the results compared with MLP Network and logistic regression. The results obtained proved that RBFN has a better performance compared to other models.

Also, [21] applied breast cancer and gene to RBFN with Growing and Pruning (GAP) algorithm. [22] used RBFN for trees species recognition and classification for forest inventories to estimate the wood quantity in a given forest area. Trees diameter, thickness bark, growth of diameter and height are the parameters used. Tree clustering with diverse input was taken by the network. The results improved forecasting techniques in forest inventories. [23] applied RBF Network for time series forecasting in a hybrid system comprising of many sub-RBFNs which produced a faster training speed with slightly better generalization capability.

2.4. Discretization

Discretization according to [24] can be defined as one way of reducing data or changing original continuous attribute into discrete attribute. Discretization process helps in data reduction and simplification, as a result, the data becomes easier to understand, used and explained. In addition achieving, faster and accurate training process [25, 26]. Many studies on discretization algorithms were tested in Data mining and knowledge discovery domain, to show that it has the potential to reduce data and improving predictive accuracy. The algorithms are "supervised versus unsupervised, global versus local and dynamic versus static [27]". Also, [28], explained that based on different theoretical origins, various methods for discretization in literature are identified such as statistics, Boolean reasoning, clustering techniques and entropy calculations. Equal Frequency Binning (EFB) Equal Width Binning (EWB) and Entropy-based Discretization (EBD) are the most general ones. In this paper, MDL discretization algorithm in Rough Set Toolkit for Data Analysis is used to discretize all input datasets.

2.5. Classification Problem

RBFN have been applied effectively in many fields for diverse purposes such as in classification. Classification is applied in virtually all circumstances where correlation between inputs and outputs exist. It can compute several possible results across various parameters, if

we have enough information from the past [16]. Also, [29] in a study proposed Enhancement of RBFN with Fuzzy-OSD algorithm for online system that classify radar pulse. It needs quick network retraining when new unseen emitters are detected. The network performance was superior to Three-Phase OSD method when it was evaluated and compared. Also, adjusting RBF units with great care in the network will improve performance in less training cycles, which is vital for real-time systems.

2.6. Network Architecture for each Dataset

In this study, Modified BP algorithm was applied to enhance RBFN learning with discretized dataset. The results of MBP-RBFN with continuous and discretized datasets were compared to ascertain the performance of our proposed method. Defining network architecture means choosing suitable number of layers and nodes for each dataset. Table 1 depicts the network architecture for this study. Optimal network design comes from careful selection of the number of hidden neurons. Generalization capacity, training time and complexity is affected by the size of the hidden layer. Though, there is no standard rule for determining optimal number of hidden nodes [30]. Many techniques have been put forward by researchers on how best, hidden nodes can be determined.

According to [31] the performance of an ANN is strongly influenced by the networks' structure and its parameters. If the network structure is too simple, it will lower the classification capacity of the network, hence affecting the final result. However, if the network structure is too complex it will lower the learning speed and ultimately the final error may increase. Also the momentum, learning rate parameters will also affect network learning speed and accuracy. If suitable network structure and parameters are used it will get the best result. The breakdown of the architectural configuration for each dataset is given in Table 1, with three learning rates of 0.5, 0.1 and 0.7 respectively. A constant momentum of 0.9 is maintained. The minimum error rate is set to 0.005. In the experiments k-fold data partition method was used. For each k experiments, k-1 fold was employed for network learning while what is left is for testing. The value of k=5. The stopping conditions for each run for all the datasets are reaching a minimum error of 0.005 or maximum of 10000 iterations reached.

Table 1. Dataset summary and architectural configuration

	XOR	Balloon	Iris	Cancer	Ionosphere
Input	3	4	4	9	34
Hidden	2	2	3	3	2
Output	1	1	3	1	1
Instances	8	16	150	699	351

2.7. Modified BP-based RBF Network Training Algorithm

The algorithm is hereby presented in detail, and part of it is adopted from [32].

Algorithm

1. Initialize network
2. Forward pass by inserting input and desired output, and compute network outputs layer by layer.
3. Backward pass

$$\frac{\partial E}{\partial w}, \frac{\partial E}{\partial c}, \frac{\partial E}{\partial \sigma^2}$$

4. Parameters update

$$w_{kj}(t+1) = w_{kj}(t) + \Delta w_{kj}(t+1) \quad (21)$$

$$\Delta w_{kj}(t+1) = \eta_3 \delta_k O_j \quad (22)$$

With,

$$\delta_k = O_k(1 - O_k)(t_k - O_k) \quad (23)$$

$$O_j = \exp\left(-\frac{(x-c_j)^2}{2\sigma_j^2}\right) \quad (24)$$

Where "$w_{kj}(t)$ is the weight from *k* to *j* at time *t*, Δw_{kj} is the weight change, η_3 is the learning rate, δ_k is error the at *k*, O_j is the actual network output at *j*, O_k is the actual network output at *k*, t_k is the target output value at *k*".

$$c_{ji}(t+1) = c_{ji}(t) + \Delta c_{ji}(t+1) \quad (25)$$

$$\Delta c_{ji}(t+1) = \frac{\eta_2 \delta_k w_{kj} O_j}{\sigma_j^2 (x_{ji} - c_{ji})} \quad (26)$$

Where "$c_{ji}(t)$ is the centre from *j* to *i* at time *t*, Δc_{ji} is the centre change, η_2 is the learning rate, δ_k is the error at *k*, O_j is the actual network output at *j*, σ_j is the width at *j*, $w_{kj}$is the weight connected between *j* and *k* and x_{kj} is the input *j* to *i*."

$$\sigma_j(t+1) = \sigma_j(t) + \Delta\sigma_j(t+1) \quad (27)$$

$$\Delta\sigma_j(t+1) = \eta_1 \delta_k w_{kj} O_j \left(\frac{(x-c_j)^2}{2\sigma_j^4}\right) \quad (28)$$

Where "$\sigma_j(t)$ is the width of *j* at time *t*, $\Delta\sigma_j$ is the width change, η_1 is the learning rate δ_k is error at *k*, O_j is the actual network output at *j*, σ_j is the width at *j*, w_{kj} is the weight connected between *j* and *k*, x_{ji} is the input at *i* and where η_3, η_2, η_1 are learning rate factors in the range [0; 1]."

5. Repeat the algorithm for all training inputs.

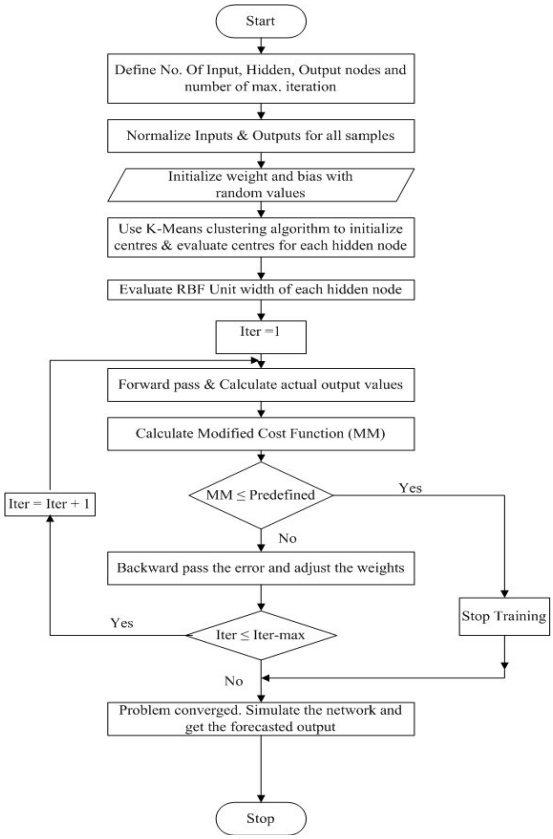

Figure 2. Modified BP-based RBF Network Training Flowchart

3. Results and Discussion

The experiments are in two parts. Part 1, is experiment for RBFN training and testing with MBP algorithm using continuous dataset. Part 2, is an experiment for RBFN training and testing with MBP algorithm using discretized dataset. The classification accuracy and error rate on test results are compared. Result summary for MBP-RBFN with both continuous and discretized datasets are given in the table and figures below.

Table 2. Result summary for MBP-RBFN with different types of dataset

Dataset	MBP-RBFN with continuous dataset		MBP-RBFN with discretized dataset	
	Classification Accuracy (%)	Error rate	Classification Accuracy (%)	Error rate
XOR	56.25	0.0876	59.90	0.0145
Balloon	54.55	0.0419	51.31	0.0374
Iris	82.95	0.4411	83.78	0.5276
Cancer	88.93	0.1081	86.86	0.0343
Ionosphere	59.17	0.1339	78.27	0.1046

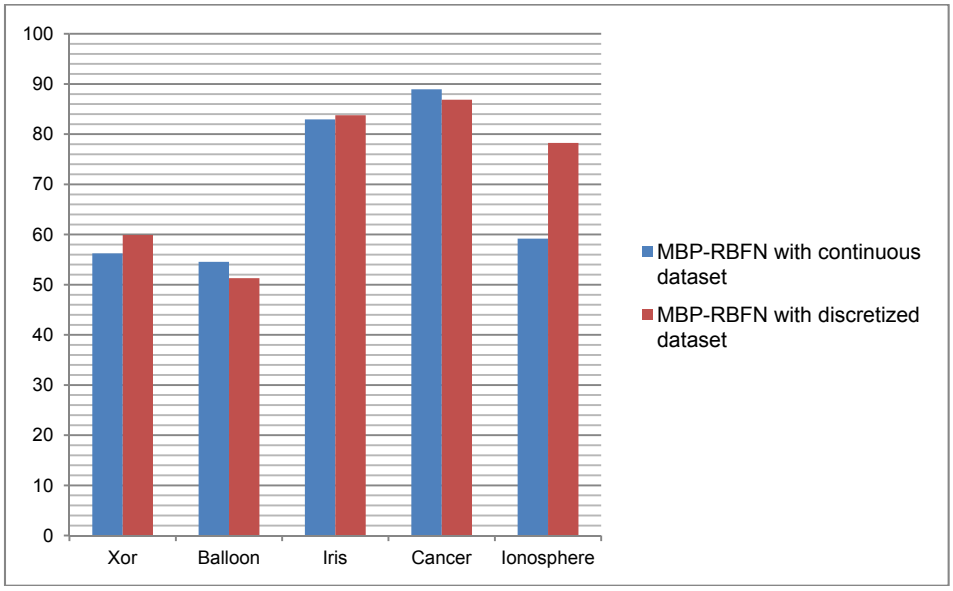

Figure 3. Classifcation Result for MBP-RBFN

From Table 2 and Figure 3, the results comparisons between continuous and discretized datasets for MBP-RBFN indicate that MBP-RBFN with discretized datasets has better classification results in XOR, Iris and Inonsphere while on the other hand; MBP-RBFN with continuous datasets has better results in Ballon and Cancer.

Also, MBP-RBFN with discretized datasets has the least convergence error rates in all the five datasets except in Iris, where MBP-RBFN with continuous dataset has the least convergence error rate as shown in Table 2 and Figue 4. This proves the claims of [9] and other researchers who said discretization improves the classification performance and faster convergence of error rates. The results above are backed by the statistical test conducted which showed that the t-test for all the results was statistically significant except for Bolloon which was statistically not significant. We can summarise here that MBP-RBFN with discretized dataset has better results in most of the cases.

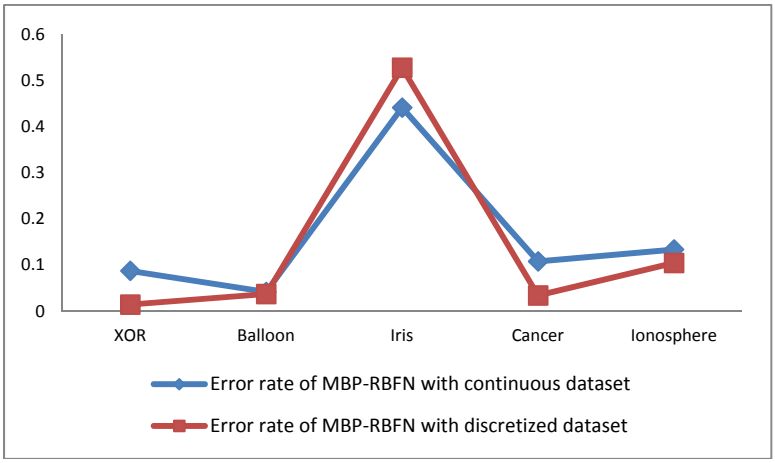

Figure 4. Error rate for MBP-RBFN with continuous and discretized dataset

The objective of this study was to test if training of RBFN with modified BP using discretized dataset would enhance the classification accuracy and small error rates of convergence.

4. Conclusion

In this study, MBP algorithm was used to train RBFN with both continuous and discretized datasets. For uniform comparison in this study, same parameters were used in all experiments for the five datasets. Finally, investigation is done by comparing results of MBP-RBFN using both continuous and discretized dataset and for each dataset and the outcome is an improved result for MBP-RBFN with discretized dataset which is better than MBP-RBFN with continuous dataset which also justified the study carried out.

Acknowledgements

Authors would like to thank Soft Computing Research Group (SCRG) Universiti Teknologi Malaysia and UTM Big Data Centre for supporting this study from inception to the end of the study.

References

[1] Chen Y. Application of a Non-liner Classifier Model based on SVR-RBF Algorithm. in *Power and Energy Engineering Conference (APPEEC), 2010 Asia-Pacific*. IEEE. 2010.
[2] Shamsuddin SM, MN Sulaiman, M Darus. *An improved error signal for the backpropagation model for classification problems.* International Journal of Computer Mathematics. 2001; 76(3): 297-305.
[3] Yu D, J Gomm, D Williams, *A recursive orthogonal least squares algorithm for training RBF networks.* Neural Processing Letters. 1997; **5**(3): 167-176.
[4] Qasem SN, SM Shamsuddin. *Hybrid Learning Enhancement of RBF Network Based on Particle Swarm Optimization*. *Advances in Neural Networks–ISNN 2009*. Springer. 2009; 19-29.
[5] Chang WY. *Estimation of the state of charge for a LFP battery using a hybrid method that combines a RBF neural network, an OLS algorithm and AGA.* International Journal of Electrical Power & Energy Systems. 2013. **53**: 603-611.
[6] Zweiri YH, et al. *A new three-term backpropagation algorithm with convergence analysis. Robotics and Automation. Proceedings. ICRA'02. IEEE International Conference on*. 2002. IEEE.
[7] Shahpazov VL, VB Velev, LA Doukovska. *Design and application of Artificial Neural Networks for predicting the values of indexes on the Bulgarian Stock market*. *Signal Processing Symposium (SPS), 2013*. IEEE. 2013.
[8] Qu M, et al. *Automatic solar flare detection using MLP, RBF, and SVM.* Solar Physics. 2003; **217**(1): 157-172.
[9] Leng WY, SM Shamsuddin. *Writer identification for Chinese handwriting.* Int. J. Advance. Soft Comput. Appl. 2010; 2(2): 142-173.

[10] Cho TH, RW Conners, PA Araman. *Fast backpropagation learning using steep activation functions and automatic weight reinitialization*. *Systems, Man, and Cybernetics. Decision Aiding for Complex Systems, Conference Proceedings., IEEE International Conference on* IEEE. 1991.
[11] Cui L, C Wang, B Yang, *Application of RBF neural network improved by PSO algorithm in fault diagnosis.* Journal of Theoretical and Applied Information Technology. 2012; 46(1): p. 268-273.
[12] Yufeng L, X Chao, F Yaozu. *The Comparison of RBF and BP Neural Network in Decoupling of DTG*. 2010.
[13] Sirat M, C Talbot. *Application of artificial neural networks to fracture analysis at the Äspö HRL, Sweden: fracture sets classification.* International Journal of Rock Mechanics and Mining Sciences, 2001; 38(5): 621-639.
[14] Plagianakos V, D Sotiropoulos, M Vrahatis. *A nonmonotone backpropagation training method for neural networks.* Dept. of Mathematics, Univ. of Patras, Technical Report, 1998; 98-04.
[15] Edward R. *An Introduction to Neural Networks.* A White paper. Visual Numerics Inc., United States of America, 2004.
[16] Rimer M, T Martinez, *CB3: an adaptive error function for backpropagation training.* Neural processing letters. 2006; 24(1): 81-92.
[17] Qasem S, S Shamsuddin, *Generalization improvement of radial basis function network based on multi-objective particle swarm optimization.* J. Artif. Intell. 2010; **3**(1).
[18] Arisariyawong S, S Charoenseang. *Dynamic self-organized learning for optimizing the complexity growth of radial basis function neural networks*. *Industrial Technology. IEEE ICIT'02. IEEE International Conference on*. 2002.
[19] Akbilgic O. *Binary Classification for Hydraulic Fracturing Operations in Oil & Gas Wells via Tree Based Logistic RBF Networks.* European Journal of Pure & Applied Mathematics. 2013; 6(4).
[20] Venkatesan P, S Anitha. *Application of a radial basis function neural network for diagnosis of diabetes mellitus.* CURRENT SCIENCE-BANGALORE. 2006; 91(9): 1195.
[21] Zhang R, et al. *An efficient sequential RBF network for bio-medical classification problems*. in *Neural Networks.* Proceedings. IEEE International Joint Conference on. 2004.
[22] Castellanos A, A Martinez Blanco, V Palencia. *Applications of radial basis neural networks for area forest.* 2007.
[23] Huang Rb, Ym Cheung. *A fast implementation of radial basis function networks with application to time series forecasting*. *Intelligent Data Engineering and Automated Learning*. Springer. 2003; 143-150.
[24] Goharian N, D Grossman, and N. Raju. *Extending the undergraduate computer science curriculum to include data mining*. *Information Technology: Coding and Computing.* Proceedings IEEE. ITCC 2004. International Conference on. 2004.
[25] Liu H, et al. *Discretization: An enabling technique.* Data mining and knowledge discovery. 2002; 6(4): 393-423.
[26] Dougherty J, R Kohavi, M Sahami. *Supervised and unsupervised discretization of continuous features*. in *ICML*. 1995.
[27] Agre G, S Peev. *On supervised and unsupervised discretization.* Cybernetics And Information Technologies. 2002; 2(2): 43-57.
[28] Saad P, et al. *Rough set on trademark images for neural network classifier.* International journal of computer mathematics. 2002; 79(7): 789-796.
[29] Montazer GA, H Khoshniat, V Fathi. *Improvement of RBF Neural Networks Using Fuzzy-OSD Algorithm in an Online Radar Pulse Classification System.* Applied Soft Computing. 2013.
[30] Kim GH, et al. *Neural network model incorporating a genetic algorithm in estimating construction costs.* Building and Environment. 2004; 39(11): 1333-1340.
[31] Yi H, G Ji, H Zheng. *Optimal Parameters of Bp Network for Character Recognition*. in *Industrial Control and Electronics Engineering (ICICEE)*. IEEE. International Conference on. 2012.
[32] Vakil-Baghmisheh MT, N Pavešić. *Training RBF networks with selective backpropagation.* Neurocomputing. 2004; 62: 39-64.