A Nodes Reduction Procedure for RBFNDDA through Histogram

Pey Yun Goh, Shing Chiang Tan, and Wooi Ping Cheah

Multimedia University, Jln. Ayer Keroh Lama, 75450 Melaka
{pygoh,sctan,wpcheah}@mmu.edu.my

Abstract. This paper presents a two-stage learning algorithm to reduce the hidden nodes of a radial basis function network (RBFN). The first stage involves the construction of an RBFN using the dynamic decay adjustment (DDA) and the second stage involves the use of a modified histogram algorithm (HIST) to reduce hidden neurons. DDA enables the RBFN to perform constructive learning without pre-defining the number of hidden nodes. The learning process of DDA is fast but it tends to generate a large network architecture as a result of its greedy insertion behavior. Therefore, an RBFNDDA-HIST is proposed to reduce the nodes. The proposed RBFNDDA-HIST is tested with three benchmark medical datasets. The experimental results show that the accuracy of the RBFNDDA-HIST is compatible with to that of RBFNDDA but with less number of nodes. This proposed network is favorable in a real environment because the computation cost can be reduced.

Keywords: radial basis network, nodes reduction, histogram, dynamic decay adjustment.

1 Introduction

The Radial Basis Function Network (RBFN) is one of the popular artificial neural networks (ANNs) being used due to its fast learning characteristic through the use of locally-tuned neurons [1, 2]. It learns the non-linear relationship among the input and output data with a simple topological structure [3, 4]. An RBFN is also an universal approximator with the use of continuous functions (usually the Gaussian functions) in the hidden units for information processing [1] [4] [5]. An RBFN can be trained by using a dynamic decay adjustment (DDA) algorithm [24]. The DDA algorithm can speed up the training process of a network to construct an RBFNDDA with a satisfactory performance in prediction. A note is, users should set manually the number of hidden nodes in a conventional RBFN but an RBDNDDA requires no such a parameter setting because the network topology can be built up automatically. However, such constructive-learning approach could result in large network architecture. A huge network architecture is fast in convergence and also favorably used for solving problems [6]. However, the complexity of a huge network is high and over-fitting is likely to occur during its training process. The network may give a poor performance on presence of spurious information such as outliers, noise and overlapping nodes [7, 8].

C.K. Loo et al. (Eds.): ICONIP 2014, Part I, LNCS 8834, pp. 127–134, 2014.

On the other hand, a small network has a better generalization ability but to search for a small yet suitable network architecture could be a time-consuming task [8, 9]. There is also a possibility where the small network may not be able to solve problem and trapped in local minima [8]. Various methods have been proposed to improve generalization capability of an ANN. One of the common methods is node pruning. Sietsma and Dow [13] proposed a two-stage pruning algorithm. In the first stage, a hidden node is removed if it has a constant output over all the training inputs and if the outputs of any two hidden nodes are the same. In the second stage, nodes are removed when they are linearly independent from other nodes in the same layer. As commented by [9], the method [13] will cause the network to take a long training time. Liang [14] proposed a node pruning method with the use of orthogonal projection and weight crosswise propagation (CP) calculation. The pruning method consists of two stages. In Stage 1, the node with shortest distance in its orthogonal projection is removed. In Stage 2, the author used the weight CP to propagate the information of deleted hidden nodes to the other hidden nodes. As such, information loss in stage 1 can be reduced. Zhang and Qiao [15] proposed the pruning algorithm based on the neural complexity (PBNC). The PBNC algorithm calculates the network complexity after deleting a hidden node. All hidden nodes must be selected once. The node with the highest neural complexity is removed. The learning process of the network is terminated when the average error of the adjustment process is less or equal to the objective error. Indeed, these methods [13, 14, 15] involved a complicated calculation process to prune the network.

As simpler and more efficient methods are favorably adopted to formulate solutions to many real-world problems [10]. The research in this study is aimed to reduce the network size of an RBFN with the use of histogram. Histogram is a traditional statistical approach that enables the visualization of statistical approximation [11]. In this work, it is employed to reduce hidden nodes of an RBFN due to its simplicity and computational efficiency [12]. Histogram is commonly used to approximate the distribution of data. From the literature, histogram is vastly applied in the field of image processing and computer vision, especially in the usage of summarizing the characteristics of images [11]. There are researchers who used histogram for pruning in speech recognition. Researchers [22, 23] proposed to use a histogram-based pruning method to further limit the search space in automatic speech recognition system. In this work, we proposed a node reduction method by absorbing a histogram algorithm [16] into the RBFNDDA learning process. Notably, the histogram algorithm was proposed by Shimazaki and Shinomoto [16] to optimize the distribution of neuronal spike signals. To our best knowledge, the use of a histogram approach for reducing the number of hidden nodes of an ANN is new.

The organization of this paper is as follows. In Section 2, the details of the RBFNDDA, the histogram algorithm and the proposed RBFN are described. In Section 3, the results from an experimental study using several benchmark data sets from the UCI machine-learning repository [17] are presented, analyzed and compared. Lastly, the conclusion of the paper and future work are described in Section 4.

2 The Methods

2.1 Overview of RBFNDDA

The DDA algorithm is applied to build an RBFNDDA. During training, new nodes are inserted into the hidden layer of RBFNDDA to encode new information from the data samples. The dynamics of RBFNDDA are governed by two user-defined parameters, i.e., the positive threshold θ^+ and the negative threshold θ^-. They regulate the width of a node (prototype) and correspond to distinguish a prototype from its neighbors (prototypes) of other classes. In this regard, θ^+ represents the lowest correct-classification probability for the correct class, and θ^- represents the highest probability tolerable to an incorrect class. In [24], the default settings of θ^+ and θ^- are 0.4 and 0.2 respectively. The training algorithm of RBFNDDA in a single epoch is as follows.

Step 1: Initialize the weight $W_i = 0$

Step 2: Consider a training input x of a class k, assume that $\boldsymbol{P}_i^k$ denote an RBF hidden node of class k has been inserted in the network. Increase the weight $W_i^k = W_i^k + 1$ if the $\boldsymbol{P}_i^k$ has the Gaussian activation $R_i^k \geq \theta^+$ (in other words, the input is correctly classified)

Otherwise, insert a new hidden node $\boldsymbol{P}_{m_k}^k$ (m_k denote number of nodes for class k) and perform the following actions.

Set $m_k = m_k + 1$

$W_{m_k}^k = 1$

Center of neuron $\boldsymbol{z}_{m_k}^k = \boldsymbol{x}$

Width $\sigma_{m_k}^k = \min_{\substack{j \neq k \\ 1 \leq b \leq m_j}} \left\{ \sqrt{-\frac{||\boldsymbol{z}_b^j - \boldsymbol{z}_{m_k}^k||^2}{ln\theta^-}} \right\}$

Step 3: Shrink the width of all the conflicting nodes where $j \neq k, 1 \leq b \leq m_j$

$$\sigma_b^j = \min \left\{ \sigma_b^j, \sqrt{-\frac{||x - z_b^j||^2}{ln\theta^-}} \right\}$$

Step 4: Repeat steps 2 and 3 for the next training inputs.

2.2 Overview of Histogram

Finding a suitable bin size is important to construct a histogram for representing the actual data distribution as close as possible [16]. As such, Shimazaki and Shinomoto [16] proposed a histogram algorithm to determine a suitable bin size with equal width and number of sequences (trials) required when representing the time-dependent spike rate. The optimal bin size can be obtained by minimizing $C_n(\Delta)$. Throughout the process, the number of bin N, the width Δ and the n sequences will change accordingly to generate the cost function. In [16], N is a setting from a range between 2 and 50

whereas *n* is set as 30. The optimum bin size *N* with the most minimum cost function will be selected. The algorithm is as below:

Step 1: Observation period *T* is divided into *N* bins of width Δ. The frequency of spikes k_i from all *n* sequences that enter the *i*-th bin is computed.
Step 2: Compute the mean, $\bar{k}$ and variance, *v* of the number of spikes, as follows.

$$\bar{k} \equiv \frac{1}{N}\sum_{i=1}^{N} k_i \tag{1}$$

$$v \equiv \frac{1}{N}\sum_{i=1}^{N}(k_i - \bar{k})^2 \tag{2}$$

Step 3: Calculate the cost function (C_n)

$$C_n(\Delta) = \frac{2\bar{k}-v}{(n\Delta)^2} \tag{3}$$

Steps 4: Repeat steps 1 to 3 by varying different numbers of bin to find the corresponding bin width Δ that minimizes $C_n(\Delta)$.

2.3 The Proposed Method

The proposed method is a two-stage learning process. Inputs are trained through RBFNDDA in the first stage. The generated RBFNDDA hidden nodes (or RBF centers) are sent to the proposed method, which we called HIST algorithm. This HIST algorithm is an extension of Shimazaki and Shinomoto [16]'s histogram algorithm. Our experiment does not need the number of sequences, as the purpose of HIST is not to train the nodes but to identify unneeded nodes. Therefore, *n* sequence is set as 1. The minimum bin number *N* is set as minimum 3 instead of 2. The reason is too less number of bins will cause difficulty in nodes reduction and degrade the performance of classification. In order to decide which bins contain unneeded nodes, steps 7 – 11 are proposed. The hidden nodes have multiple dimensions but HIST accepts one dimension inputs. Therefore, each hidden node is transformed to one dimension by aggregating sum of all its attributes and enter the HIST according to class label. The details of the HIST algorithm are as below:

Step 1: All hidden nodes *H* are categorized according to class *c* and transformed to one dimension (eq. 4, *D* = number of dimension),

$$Input\ X_i = \sum_{j=1}^{D} x_j \qquad i=1,\ldots,H \tag{4}$$

Step 2: All Input *X*s of class *c* are divided into *N* bins of width Δ (eq. 5). Count the frequency of Input *X*, k_i which enters the *i*-th bin.

$$\Delta = \frac{Input\ X_{max} - Input\ X_{min}}{N} \tag{5}$$

Step 3: Generate the mean, $\bar{k}$ (eq. (1)) and variance, *v* (eq. 2) of k_i
Step 4: Calculate the C_n (eq. 3).

Step 5: Step 1 through 4 are repeated by varying the setting of N and Δ to search for an optimum bin size and width that minimize $C(\Delta)$.
Step 6: Histogram is constructed using the optimum bin size and width from Step 5.
Step 7: Compute the magnitude $p(a)$ (eq. 6), which is the probability of the RBFNDDA weight for each bin, where a = value of RBFNDDA weightage. Assume that a bin contains 7 Input X, with 3 of them are having the weightage a, which is 1, another 3 Input X with weightage which is 3 and 1 Input X with weightage which is 5. Therefore, the probability $p(a)$ are $\frac{3}{7}, \frac{3}{7}$, and $\frac{1}{7}$ respectively. The number of weight category m for this bin is 3.

$$p(a) = \frac{number\ of\ Input\ X\ of\ a\ RBFNDDA\ weight\ category}{total\ number\ of\ Input\ X\ per\ bin} \tag{6}$$

Step 8: Compute the expected value $E(a)$ for each bin, where $E(a) = \sum_1^m a\, p(a)$. ($E(a)$ reflects the score of weightage for each bin. Continue with the example in Step 7, $E(a) = (1)\left(\frac{3}{7}\right) + (3)\left(\frac{3}{7}\right) + (5)\left(\frac{1}{7}\right) = 2.43$. The highest the score, the more important is the bin.)
Step 9: Normalize $E(a)$ between 0 and 1. Assume that the histogram has 3 bins, each bin with the $E(a)$ score is 2.43, 0.38 and 4.50. After normalization, the obtained scores are 0.50, 0 and 1.00 respectively.
Step 10: Bins with normalized expected value lowered than a threshold (we set it as 0.2) are deleted. As such, based on the assumption in Step 9, bin 2 is deleted.
Step 11: Repeat Steps 2 to 10 for all the classes.
Step 12: Send retains nodes back to RBFNDDA and proceed to testing phase.

Normalization of $E(a)$ is to generalize the proposed method to all data sets as each data set will generate different number of DDA weights with different aggregate sum of input X. Without generalization, the threshold used to delete the bin need to change accordingly. With normalization, one threshold value can be used for different data sets.

3 Experiment and Discussion

The results of RBFNDDA-HIST is benchmarked with three medical data sets from the UCI Machine Learning Repository [17]: Diabetes, Cancer and Heart. All of the data sets consist of two classes with the number of samples 768, 699, 270, respectively, and number of attributes 8, 9 and 13 respectively. In this experiment, we used a two-fold cross validation where 50% of the data was used for training and the remaining data was used for testing. The involved parameter settings were $\theta^+ = 0.4$, $\theta^- = 0.2$, $n = 1$, N = minimum 3, maximum 50, and threshold for $E(a) = 0.2$. RBFNDDA was trained in multi epochs before the execution of the HIST algorithm. The maximum training epoch of RBFNDDA was set as 6. The proposed RBFNDDA-HIST was run on a computer having the following specifications: operating system Windows 7, Intel Core (TM) CPU i5-2410M and 4.0 GB RAM. The performance of RBFNDDA-HIST is then compared with the original RBFNDDA and RBFNDDA-T [18].

RBFNDDA-T is an online pruning technique where node is marked as temporary once it is inserted into the network. The status of the node will change to permanent if more than two samples are covered by the node. After each epoch training, nodes with temporary status are deleted and are not used in the next training epoch. Paetz [18] conducted an experiment in 8 runs for both RBFNDDA and RBFNDDA-T. In our work, the experiment was repeated for 30 runs, and the average results in terms of the number of hidden nodes and the accuracy rates of RBFNDDA-HIST were computed. The results are listed in Table 1.

Table 1. Experiment results (Acc. = average test accuracy; #nodes = average number of hidden nodes; standard deviation in round brackets)

Data Set	Paetz [18]				RBFNDDA-HIST	
	RBFNDDA		RBFNDDA-T			
	Acc.	# nodes	Acc.	# nodes	Acc.	# nodes
Diabetes	74.35	288.5	73.5	65.6	72.85	202.9
	(0.97)	(6.1)	(2.38)	(4.2)	(1.24)	(26.2)
Cancer	96.86	70.6	96.9	38.3	96.80	19.1
	(0.99)	(13.4)	(0.39)	(4.6)	(0.97)	(10.3)
Heart	79.26	83.6	79.82	32.6	80.10	36.95
	(2.83)	(3.3)	(3.54)	(2.1)	(2.55)	(1.9)

As compared to RBFNDDA, RBFNDDA-HIST manages to reduce the hidden nodes significantly with the percentages of reduction are 29.68%, 73.00% and 55.80% in Diabetes, Cancer and Heart respectively. As compared to RBFNDDA-T, the accuracy of RBFNDDA-HIST in Diabetes is lower with a higher number of hidden nodes; whereas in Cancer and Heart, RBFNDDA-HIST is competitive to give high accuracy rates and small number of hidden nodes. The Diabetes data set consists of data samples that are highly overlapped [18]. In Diabetes, RBFNDDA-T achieves better result in node reduction, i.e. 77.26% because RBFNDDA-T implements node pruning for unwanted information from its network whereas RBFNDDA-HIST re-organizes all existing nodes of RBFNDDA into a more compact structure without removing such overlapping information intensively. On the other hand, for Cancer data sets, RBFNDDA-HIST achieves better result where the percentage of node reduction for RBFNDDA-HIST is 73.00% and RBFNDDA-T is 45.75% from RBFNDDA. The percentages of node reduction for both RBFNDDA-T and RBFNDDA-HIST are almost similar when dealing with Heart data set. In other words, the performances of RBFNDDA-HIST and RBFNDDA-T in node reduction are problem-dependent. But RBFNDDA-HIST shows a significant advantage in generating a more compact network structure when compared to RBFNDDA.

Further comparison was performed with different experiment setup to see how the reduction of nodes will affect the accuracy and computational cost of RBFNDDA-HIST. We run this experiment by using the Cancer data set and compare with the reported methods in [19]. The experiment excluded all the missing values. The remaining 400 instances were used in training and 283 instances were used in testing. Table 2 presents the comparison of the results. The numbers of hidden nodes for MPANN are lower than those of RBFNDDA-HIST, which are 4.125 ± 1.360 and

21.567 ± 7.655 respectively. This is expected as we aim to have a simple and yet a fast learning neural network. While MPANN is having a complicated construction of neural network that match this reason: a successive smaller networks can be time consuming [8]. The computational cost of RBFNDDA-HIST is much lower than the methods in [19, 20, 21]. The accuracy is about the same when compared with these methods.

Table 2. Performance comparison (accuracy and number of epoch)

Methods	Accuracy	Number of epoch
RBFNDDA-HIST	0.976±0.313	3
MPANN [19]	0.981±0.005	5100
EAA [20]	0.981±0.464	200000
C-net [21]	0.975±1.800	10000

4 Conclusion

In this study, a RBFNDDA-HIST network is proposed to reduce hidden nodes to constitute a more compact network structure than the original RBFNDDA. The process is simple and fast in handling superfluous nodes. This is further shown when the proposed method is compared with MPANN, EAA and C-net. In the future, additional experiments will be carried out by using other benchmark and real data sets to examine the effectiveness of RBFNDDA-HIST.

References

1. Meng Joo, E., Shiqian, W., Juwei, L.: Face recognition using radial basis function (RBF) neural networks. In: Proceedings of the 38th IEEE Conference on Decision and Control, vol. 3, pp. 2162–2167 (1999)
2. Moody, J., Darken, C.J.: Fast learning in networks of locally-tuned processing units. Neural Computation 1(2), 281–294 (1989)
3. Guang-Bin, H., Saratchandran, P., Sundararajan, N.: A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. IEEE Transactions on Neural Networks 16(1), 57–67 (2005)
4. Kang, L., Jian-Xun, P., Er-Wei, B.: Two-stage mixed discrete-continuous identification of Radial Basis Function (RBF) Neural models for nonlinear systems. IEEE Transactions on Circuits and Systems I: Regular Papers 56(3), 630–643 (2009)
5. Freeman, J.A.S., Saad, D.: Learning and generalization in radial basis function networks. Neural Computation 7(5), 1000–1020 (1995)
6. Yu, H.: Network complexity analysis of multilayer feedforward artificial neural networks. In: Schumann, J., Liu, Y. (eds.) Appl. of Neural Networks in High Assur. Sys. SCI, vol. 268, pp. 41–55. Springer, Heidelberg (2010)
7. Zhang, G.P.: Avoiding pitfalls in neural network research. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 37(1), 3–16 (2007)
8. Reed, R.: Pruning algorithms-a survey. IEEE Transactions on Neural Networks 4(5), 740–747 (1993)

9. Karnin, E.D.: A simple procedure for pruning back-propagation trained neural networks. IEEE Transactions on Neural Networks 1(2), 239–242 (1990)
10. Augasta, M.G., Kathirvalavakumar, T.: A novel pruning algorithm for optimizing feedforward neural network of classification problems. Neural Processing Letters 34(3), 241–258 (2011)
11. Ioannidis, Y.: The histogry of histogram. In: Proceedings of the 29th International Conference on Very Large Data Bases, vol. 29, pp. 19–30 (2003)
12. Legg, P.A., Rosin, P.L., Marshall, D., Morgan, J.E.: Improving accuracy and efficiency of registration by mutual information using Sturges' histogram rule. In: Medical Image Understanding and Analysis, pp. 26–30 (2007)
13. Sietsma, J., Dow, R.J.F.: Neural net pruning-why and how. In: IEEE International Conference on Neural Networks, vol. 1, pp. 325–333 (1988)
14. Liang, X.: Removal of hidden neurons in multilayer perceptrons by orthogonal projection and weight crosswise propagation. Neural Computing and Applications 16(1), 57–68 (2007)
15. Zhang, Z., Qiao, J.: A node pruning algorithm for feedforward neural network based on neural complexity. In: 2010 International Conference on Intelligent Control and Information Processing (ICICIP), pp. 406–410 (2010)
16. Shimazaki, H., Shinomoto, S.: A method for selecting the bin size of a time histogram. Journal of Neural Computation 19(6), 1503–1527 (2007)
17. Asunction, A., Newman, D.J.: No Title. University of California, School of Information and Computer Science, Irvine, CA (2007)
18. Paetz, J.: Reducing the number of neurons in radial basis function networks with dynamic decay adjustment. Neurocomputing 62, 79–91 (2004)
19. Abbass, H.A.: An evolutionary artificial neural networks approach for breast cancer diagnosis. Artificial Intelligence in Medicine 25(3), 265–281 (2002)
20. Fogel, D.B., Wasson III, E.C., Boughton, E.M.: Evolving neural networks for detecting breast cancer. Cancer Letters 96(1), 49–53 (1995)
21. Abbass, H.A., Towsey, M., Finn, G.: C-Net: A method for generating non-deterministic and dynamic multivariate decision trees. Knowledge and Information Systems 3(2), 184–197 (2001)
22. Pylkkonen, J.: New pruning criteria for efficient decoding. In: Proceedings of the 9th European Conference on Speech Communication and Technology, pp. 581–584 (2005)
23. Steinbiss, V., Tran, B.H., Ney, H.: Improvement in beam search. In: International Conference on Spoken Language Processing, pp. 2143–2146 (1994)
24. Berthold, M.R., Diamond, J.: Constructive training of probabilistic neural networks. Neurocomputing 19, 167–183 (1998)