Proceedings of the 2011 International Conference on Machine Learning and Cybernetics, Guilin, 10-13 July, 2011

SENSITIVITY BASED GROWING AND PRUNING METHOD FOR RBF NETWORK IN ONLINE LEARNING ENVIRONMENTS

PATRICK P. K CHAN, XI-RONG WU, WING W. Y. NG, DANIEL S. YEUNG

Machine Learning and Cybernetic Research Center, School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
E-MAIL:patrickchan@ieee.org, leahjob@163.com

Abstract:

How to define the architecture of classifiers dynamically is one of the major research topics in online learning. This paper presents a new online learning algorithm for Radial Basis Function Network named Sensitivity Based Neurons Growing and Pruning Method for RBF network (SBGAP). The performance of SBGAP is evaluated experimentally by comparing accuracy and the number of neurons with the existing methods. The experimental results show that SBGAP achieve litter higher accuracy with fewer hidden units in most situations.

Keywords:

SBGAP; L-GEM; Sensitivity; Decouple Extended Kalman Filter (DEKF)

1. Introduction

Different from offline learning which has a batch of samples in the training process, online learning is a model that learns the sample one by one. [13] An online learning algorithm, described in [3], has the following distinguishing features: 1) samples are presented to the learning system one-by-one 2) training sample is seen and learned at any time 3) training samples is discarded as soon as it has been learned and 4) no prior knowledge of the number of total training samples being presented.

One of the important research issues in online learning is to adjust the architecture of classifiers dynamically to adapt the problem [8, 13]. As RBF Network is well-known classifier with good generalization ability in many applications, growing and pruning rules in RBF network will be focused in our study.

The algorithm Minimal Resource Allocation Network (MRAN) proposed by Lu Yingwei et al. [1]. The growing criteria depend on 1) Root Mean Square Error (RMSE) of the classifier for the latest m samples, 2) the distance between the latest sample and nearest neuron, and 3) the error of the latest sample. The outputs of neurons for the latest m samples are considered as the pruning criterion. The method named improved RBF network [7] is proposed based on MRAN. This method changes some functions or parameters in the pruning and growing criteria from MRAN. The drawbacks of these methods are time consumed and also a neuron with a small output does not mean it has bad performance.

Guang-Bin Huang et al. proposed growing and pruning algorithm for RBF Network (GAP-RBF) and Generalized GAP-RBF in [2] [3] respectively. The pruning and growing criteria in these methods are based on the significance of a neuron. The significance is defined by the output of the neuron to all samples. However, the prior knowledge on the density function of the problem is required. It may not be reasonable for some situation. Some modified GAP-RBF in [5, 6] try to estimate the density functions using different methods, e.g. assume uniform distribution data or use Gaussian mixture models to estimate the density. However, these estimations are only suitable for particular distribution; otherwise, the performance will be degrade significantly. Another drawback of this type of method is time consuming. A fast growing and pruning algorithm for RBF network (FGAP-RBF) [4] reduce the time complexity by only considering the latest M samples. The major problem of these methods is a neuron with high significances may have a bad performance.

In this paper, a new method for growing or pruning based on the sensitivity of a hidden unit in its neighbors has been proposed. The fast adjusting algorithm Decouple Extended Kalman Filter (DEKF) [4, 10] is used to reduce the time complexity.

This paper is organized as follow. Section 2 describes the Localized Generalization Error Model; the proposed method is introduced in Section 3. Experimental Comparison between the performance of the proposed method, MRAN and FGAP-RBF are given in Section 4. Finally, the conclusion is discussed in Section 5.

978-1-4577-0308-9/11/$26.00 ©2011 IEEE

2. Localized Generalization Error Model (L-GEM)

Yeung et al. [11] proposed L-GEM for RBF network. The basic idea is to minimize the sum of local error and sensitivity of the network.

For a given training set $D=\{(x_i,y_i)\}_{i=1}^{N}$, where N is the number of training set, x_i is a n dimension vector denoted the i^{th} training sample, y_i represents the true class id of x_i.

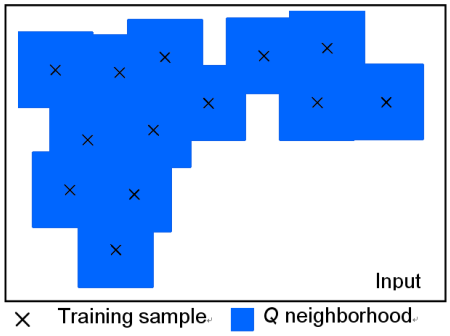

Figure 1: Q neighborhood of an artificial training set in the input space

The Q neighborhood is simply defined as the union of all Q_i neighborhoods of the training sample x_i.

$$Q=\bigcup_{i=1}^{N} Q_i$$

where $Q_i=\{x\,|\,x=x_i+\Delta x;|\Delta x|\le q\}$, i=1..N, N is the number of training samples, $x_i \in D$ and q is the parameter to control the size of the region being considered.

Figure 1 illustrates the Q neighborhood of a two-dimensional training set. The localized generalization error in the Q neighborhood is defined as:

$$R_Q=\int_Q (f(x)-F(x))^2 p_Q(x)dx$$

where $P_Q(x)$ denotes the probability density function of x in Q.

The L-GEM provides the upper bound of generalization error R_Q^* for unseen samples located in the Q-Union.

$$R_Q=\int_Q (f(x)-F(x))^2 P_Q(x)dx \le R_Q^*$$

Assume that x is uniformly distributed in the Q neighborhood, then

$$R_Q^*=(\sqrt{R_{emp}}+\sqrt{E_D E_{Q_i}((\Delta Y_i)^2)}+A)^2$$

where $\Delta Y_i=f(x)-f(x_i)$, $R_{emp}=1/N\sum_{i=1}^{N}(f(x_i)-F(x_i))^2$, and A is the square root value of the difference between the maximum and minimum value of the target outputs, which is fixed for a given dataset.

In the L-GEM framework, a classifier with a smaller R_Q^* is preferred. It means that a classifier with a small combined value of training error and sensitivity value is expected to have better generalization performance. L-GEM has been used in model selection, feature selection, and sample selection [REF].

3. The proposed methods

L-GEM evaluates the performance of a classifier in offline learning. In this paper, the idea of Q-neighborhood is applied to calculate the sensitivity of a neuron. That is sensitivity measure is defined as the root value of the squared output difference between neurons and unseen samples within its N neighborhoods. The sensitive neuron is removed since usually a stable is more preferable. The sensitivity of a neuron k is defined as:

$$S_k=\sqrt{(\sum_{j}^{N}(f(x_j)-f(\mu_k))^2)/N} \qquad (1)$$

where μ_k is the center of neuron k, x_j is j^{th} neighborhood of neurons k, $f(x)$ is the output of the network. Its neighborhoods are in

$$Q_k=\{x_j\,|\,x_j=\mu_k+\Delta x;|\Delta x|\le q\} \qquad (2)$$

Section 3.1 introduces the parameters. Growing strategy and pruning strategy are discussed in section 3.2 and 3.3. Finally, section 3.4 describes the proposed method.

3.1. Parameters

The following paramters will be used in the proposed model:

- $\varepsilon_{\min}$: the minimal distance threshold between input samples and nearest hidden units.
- $\varepsilon_{\max}$: the maximal distance threshold between input samples and nearest hidden units.
- γ: a decay constant, $1<\gamma<1$
- S: a sensitivity threshold
- Out: the maximum output of the RBF network for a input sample
- κ: an overlap factor that determines the overlap of the responses of the hidden neurons in the input space
- N: the number of neighborhoods of a hidden unit
- q: the distance limited between a hidden unit and

its neighborhoods
K : the number of hidden units
nr : the nearest neuron of a input sample
$actThr$: a parameter to measure a neuron active or not

3.2. Growing Strategy

Given a sample x_n, the growing strategy in the proposed method is based on the following criteria:

1) the sensitivity of a neuron

Sensitivity measure if x_n is sensitive to the hidden unit. Small sensitivity value is preferred, based on the L-GEM. Sensitivity measurement is defined as

$$S_{K+1} < S \tag{3}$$

2) the distance between x_n and the nearest hidden unit nr

The new added hidden unit should be located far way from the nearest unit. Distance is calculated by formula (4) and distance in growing criteria is (5):

$$\varepsilon_n = \max\{\varepsilon_{\min}, \varepsilon_{\max}\gamma^n\} \tag{4}$$

$$\| x_n - \mu_{nr} \| > \varepsilon_n \tag{5}$$

3)the max output of these hidden units for x_n

The response of xn from an unit with large width may still be large although xn is far away from that unit [9]. Hence, only measuring the distance is not enough. However, as mentioned in [1], the unit may give a wrong response. Therefore, a new unit is allocated only when the recent network gives a low or wrong response to x_n. Maximum output of the network and classification precise in growing criteria are:

$$Out = \max_k \{|\phi_k(x_n)|\} < actThr \tag{6}$$

$$f(x_n) \neq y_n \tag{7}$$

where k is between 1 and K.

4) whether the classifier classifies correctly to decide allocating a hidden unit

3.3. Pruning Strategy

Pruning criterion is based on the sensitivity of a hidden unit and its N neighborhoods satisfied (2). Similar to the first criterion in growing, the senstivitiy is defined as :

$$S_{nr} > S \tag{8}$$

where S_{nr} is calculated by formula (1).

3.4. SBGAP algorithm:

Table 1. SBGAP ALGORITHM

For each observation (x_n, y_n), where n=1, 2, 3... , $\boldsymbol{x}_n \in R^p$, do

Step1: Compute the network output:

$$f(x) = \sum_{k=1}^{K} \alpha_k \exp(-\frac{1}{\sigma_k^2} \| x_n - \mu_n \|^2)$$

Step2: Calculate the parameters for the potential new $(K+1)^{th}$ hidden unit.

$$\alpha_{K+1_n} = e_n, \mu_{K+1} = x_n, \sigma_{K+1} = \kappa \| x_n - \mu_{nr} \|$$

Step3: random generate N neighbors of $(K+1)^{th}$ hidden unit by the rule of (2) and then calculate the parameters S_{K+1}, ε_n, *Out*

Step4: If $\| x_n - \mu_{nr} \| > \varepsilon_n$ and $S_{K+1} < S$ and
(Out < *actThr* or $f(x_n) \neq y_n$) Then
allocate a new neuron K+1, K=K+1;
Else
Adjust the network parameters of the nearest neuron, $\alpha_{nr}, \mu_{nr}, \sigma_{nr}$ using DEKF.
Check the criterion for pruning the hidden neurons:
Random generate N neighborhoods of the nr^{th} hidden unit by the rule of (2), and calculate S_{nr} by (1)
If $S_{nr} > S$
remove the nr^{th} hidden units.
Endif
Endif

4. Performance evaluation of SBGAP for classification problems

Five datasets, such as Segment, Satimage, Letter, Heart, and Shuttle, from UCI [12], are used in the experiments. Table 2 shows the number of samples, classes, and features for each dataset. When a sample comes, the current classifier firstly classifies and then checks growing rules or adjusts the parameters and check pruning rule for each problem. The number of output neurons for the RBF is the same as the number of classes of the problem. The classes of letter data set are figured from 1 to 26 responding A to Z. For the reason that Letter and Shuttle have many samples and MRAN is time consumed with large storage, the experiments are only done by SBGAP and FGAP-RBF. The common parameters for SBGAP, MRAN, FGAP-RBF are given as $\gamma = 0.99, \kappa = 0.8, \varepsilon_{\min} = 0.5, \varepsilon_{\max} = 100$, for five classification problems. Parameters of SBGAP are given as S=0.005, N=20, q=0.05, $actThr$=0.6, parameters of FGAP-RBF are $e_{\min}$ =0.01, M=40 for each dataset, which are determined by experiments. Parameters in MRAN are $e_{\min}$ =0.2, $e'_{\min}$ =0.2, NM=60 for Segment and Heart, and $e_{\min}$ =0.3, $e'_{\min}$ =0.2, NM=70 for Satimage.

Table 2 Detail of classification data sets for online classify and learning

Data sets	No. of samples	No. of classes	No. of attributes
Segment	2310	7	19
Satimage	6435	7	36
Letter	10000	26	16
Heart	270	2	13
Shuttle	43500	7	9

In this paper, we use the accuracy of the whole samples having been received, and per-class accuracy to evaluate the performance of SBGAP.

Table 3, No. in second column is the neurons number difference and Acc in third column is the accuracy error between SBGAP and FGAP-RBF. So do SBGAP VS MRAN. Table 3 shows that SBGAP allocates fewer neurons and achieve a little higher accuracy than FGAP-RBF in most situations. Although SBGAP generates more neurons than MRAN, but its accuracy is higher.

Figure 2, 4, 6 are about numbers of neurons the three methods allocate dynamically for Segment, Satimage and Heart respectively. Figure 8 and 10 are about neurons numbers generated by SBGAP and FGAP-RBF for Letter and Shuttle. X-coordinate is the number of samples received so far and y-coordinate is the numbers of neurons generated. These figures obviously show that SBGAP allocates fewer neurons than FGAP-RBF, and more than MRAN in most cases.

Figure 3, 5, 7 are about accuracies of three methods allocate for Segment, Satimage and Heart respectively. Figure 9 and 11 are accuracies of SBGAP and FGAP-RBF for Letter and Shuttle. X-coordinate is the number of samples received so far and y-coordinate is the accuracies of the methods. These figures show that SBGAP achieves litter higher accuracy than FGAP-RBF and MRAN in most situations.

Per-class accuracy of each dataset is given in Table 4. The values from left to right represent the accuracies of classes of each data set respectively. For example, Segment, the seven values per line are the accuracies from class 1 to class 7. From Table 4, we can see that SBGAP almost reaches higher accuracy of each class not just focuses on some classes.

Table 3. Difference of Classification Accuracies and the numbers of neurons for SBGAP comparing with MRAN, and FGAP-RBF

	SBGAP VS FGAP-RBF		SBGAP VS MRAN	
Date sets	Neuron #	Accuracy	Neuron #	Accuracy
Segment	-45	0.26%	196	7.4%
Satimage	-61	0.3%	1190	7.54%
Heart	-31	3.33%	- 21	1.85%
Letter	-409	0.40%		
Shuttle	26	0.97%		

Table 4. Per-class classification accuracies for SBGAP, MRAN, and FGAP-RBF

Data sets	Algorithms	Accuracy per class (%)
Segment		**from 1 to 7**
	SBGAP	88.49, 99.70, 84.55, 88.49, 82.12, 95.15, 97.58
	MRAN	73.33, 99.70, 73.64, 80.61, 64.24, 94.89, 97.88
	FGAP-RBF	87.58, 99.70, 83.94, 86.97, 80.91, 97.58, 97.58
Satimage		**from 1 to 5 and 7**
	SBGAP	97.33, 97.30, 90.13, 62.94, 88.40, 89.39
	MRAN	94.91, 88.05, 92.56, 51.76, 65.21, 77.32
	FGAP-RBF	96.22, 97.30, 89.54, 61.34, 88.12, 90.58
Heart		**from 0 to 1**
	SBGAP	68.67, 50.83
	MRAN	70.67, 44.17
	FGAP-RBF	60.00, 54.16
Letter		**from A to Z**
	SBGAP	94.91, 84.52, 88.62, 83.98, 80.91, 84.17, 84.20, 76.66, 90.39, 88.27, 82.43, 89.07, 88.78, 86.36, 82.90, 85.09, 87.57, 79.67, 83.66, 88.99, 91.626, 85.86, 91.81, 86.47, 86.52, 89.11
	FGAP-RBF	94.40, 83.25, 88.36, 83.98, 80.15, 82.32, 84.20, 75.92, 91.21, 86.99, 81.62, 89.88, 89.02, 86.36, 83.95, 85.33, 86.76, 79.95, 82.54, 89.23, 90.89, 86.39, 90.96, 85.46, 84.07, 89.11
Shuttle		**from 1 to 7**
	SBGAP	99.79, 78.38, 83.33, 99.05, 98.90, 33.33, 63.64
	FGAP-RBF	99.63, 83.78, 84.09, 97.64, 98.78, 33.33, 63.64

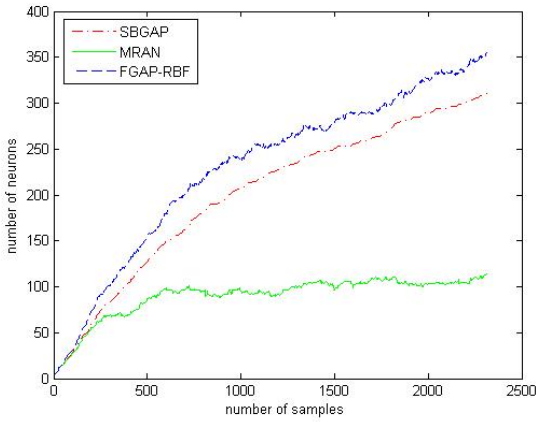

Figure 2. Number of neurons allocated for Segment

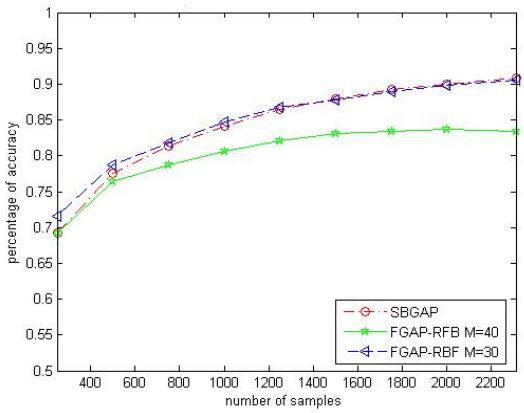

Figure 3. Accuracies about Segment

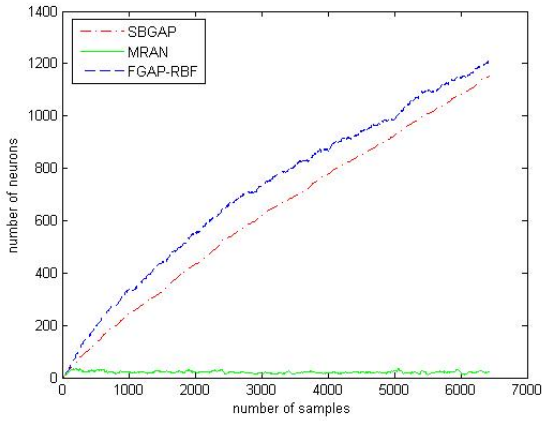

Figure 4. Number of neurons allocated for Satimage

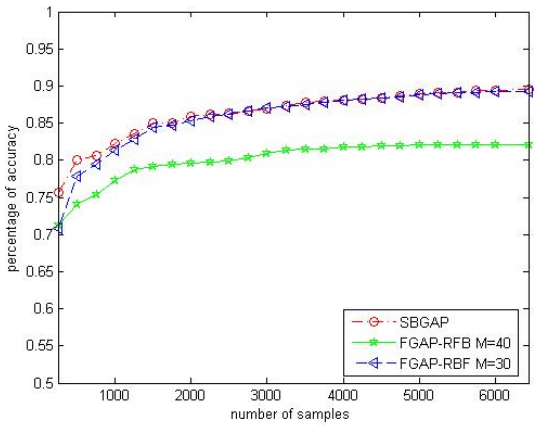

Figure 5. Accuracies about Satimage

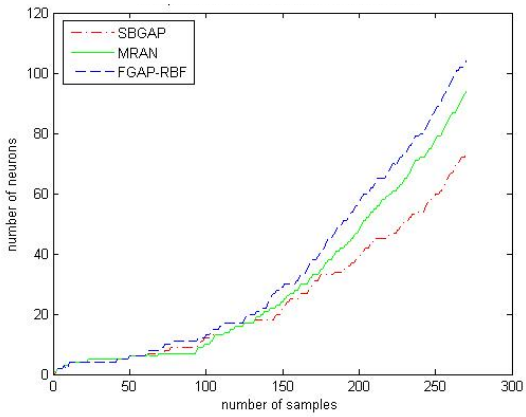

Figure 6. Number of neurons allocated for Heart

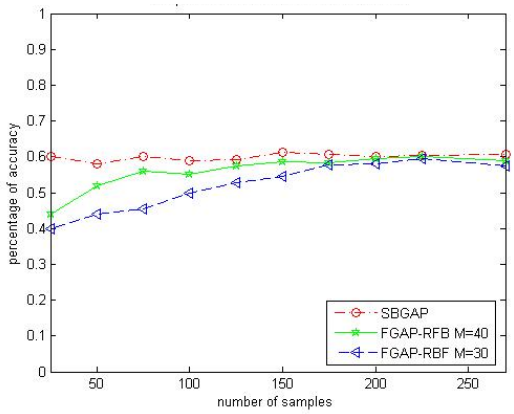

Figure 7. Accuracies about Heart

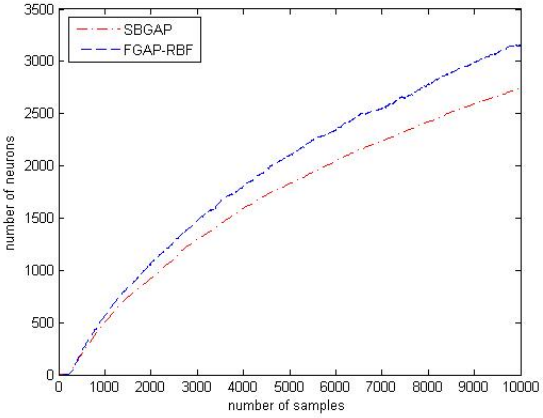

Figure 8. Number of neurons allocated for Letter

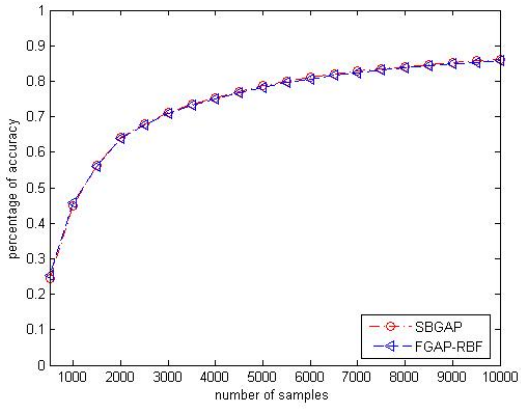

Figure 9. Accuracies about letter

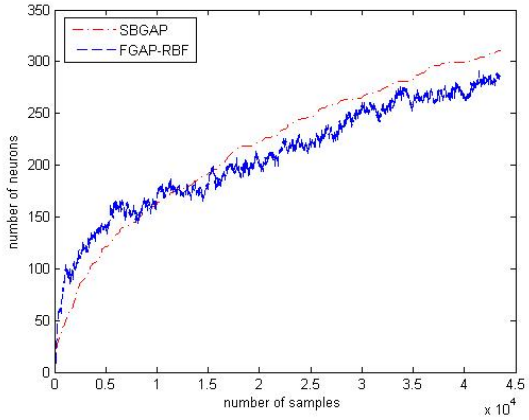

Figure 10. Number of neurons allocated for shuttle

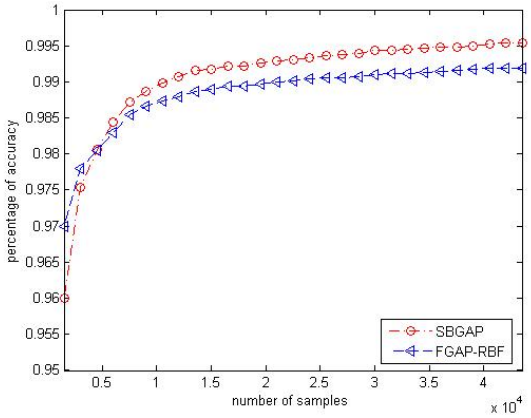

Figure 11. Accuracies about shuttle

5. Conclusion

In this article, a new online algorithm SBGAP has proposed. SBGAP can adjust the architecture of classifiers dynamically and achieve higher accuracy with fewer neurons by its growing and pruning rules through five classification problems. A problem of SBGAP is that, for testing growing rules and the pruning rule, it has to generate *N* neighborhoods of the nearest hidden unit every time. This may need some extra time and storages, but since it is not sensitive to *N* and only generates small number of neighborhoods, it doesn't matter much.

Acknowledgement

This work is supported by National Natural Science Foundation of China (61003171 and 61003172) and the Fundamental Research Funds for the Central Universities 2009ZZ0050 and 2011ZM0066.

References

[1] Lu Yingwei, N. Sundararajan, P. Saratchandran "A Sequential Learning Scheme for Function Approximation Using Minimal Radial Basis Function Neural Networks", Neural Computation 9, 461–478 (1997), Massachusetts Institute of Technology

[2] Guang-Bin Huang, P. Saratchandran, Narasimhan Sundararajan "An Efficient Sequential Learning Algorithm for Growing and Pruning RBF (GAP-RBF) Networks" IEEE Trans. Systerm, Man, and Cybernetics—PART B: CYBERNETICS, VOL. 34, NO. 6, DECEMBER 2004

[3] Guang-Bin Huang, P. Saratchandran and Narasimhan Sundararajan "A Generalized Growing and Pruning RBF(GGAP-RBF) Neural Network for Function Approximation", IEEE Trans. Neural Networks, VOL. 16, NO. 1, JANUARY 2005

[4] Runxuan Zhang, Guang-Bin Huang, N. Sundararajan, P. Saratchandran "Improved GAP-RBF network for classification problems", 2006 Elsevier

[5] Tohid Alizadeh1, Karim Salahshoor, Mohammad Reza Jafari , Abdollah Alizadeh, Mehdi Gholami "On-line Identification of Hybrid Systems Using an Adaptive Growing and Pruning RBF Neural Network" 2007 IEEE

[6] M. Bortman, M. Aladjem "A Growing and Pruning Method for Radial Basis Function Networks", IEEE Trans. Neural Networks, VOL. 20, NO. 6, JUNE 2009

[7] Zhang Xiao Ming, Ning Guang Liang "An improved RBF network on-line learning algorithm" Second International Symposium on Information Science and Engineering, 2009 IEEE

[8] Alan Fern, Robert Givan, "Online Ensemble Learning: An Empirical Study", 2003 Kluwer Academic Publishers. Manufactured in The Netherlands.

[9] F.J. Pelayo, A. Prieto, "Time series analysis using normalized PG-RBF network with regression weights", 2002 Elsevier Science B.V.I. Rojas, H. Pomares, J.L. Bernier, J. Ortega, B. Pino,

[10] G.V. Puskorius, L.A. Feldkamp, "Neurocontrol of nonlinear dynamics systems with kalman filter trained recurrent networks", IEEE Trans. Neural Networks 5 (2) (1994) 279 – 297.

[11] W.W.Y. Ng, D.S. Yeung, D. Wang, E.C.C. Tsang, X.Z. Wang, "Localized generalization error of Gaussian-based classifiers and visualization of decisionboundaries", Soft Computating 11 (4) (2007) 375 – 381.

[12] http://www.ics.uci.edu/~mlearn/MLRepository.html

[13] Nikunj C. Oza, "Online Bagging and Boosting" Systems, Man and Cybernetics, 2005 IEEE International Conference