2004 IEEE International Conference on Systems, Man and Cybernetics

Sequential RBF Function Estimator – Memory Regression Network

Chi-kin Chow and Hung-tat Tsui
Visual Signal Processing and Communications Laboratory
Department of Electronic Engineering
The Chinese University of Hong Kong
E-mail: {ckchow1, httsui}@ee.cuhk.edu.hk

Abstract – *The neural-network training algorithm can be divided into 2 categories: (1) Batch mode and (2) Sequential mode. In this paper, a novel online RBF network called "Memory Regression Network (MRN)" is proposed. Different from the previous approaches [2, 11], MRN involves two types of memories: Experience and Neuron, which handle short and long term memories respectively. By simulating human's learning behavior, a given function can be estimated without memorizing the whole training set. Two sets of function estimation experiments are examined in order to illustrate the performance of the proposed algorithm. The results show that MRN can effectively approximate the given function within a reasonable time and acceptable mean square error.*

1 Introduction

Radial Basis Function (RBF) network is a multi-layers feedforward network using radial basis functions in the solution of the real multivariate interpolation problem. It consists of three functionally distinct layers. The input layer is simply a set of sensory units. The second layer is a hidden layer of sufficient dimensions, which applies a non-linear transformation of the input domain to a higher dimension hidden-unit domain such that the training samples can be linearly separated. The third layer performs a linear transformation from the hidden-unit domain to the output domain. It has been applied successfully in a number of applications including image processing [6], speech recognition [1, 8, 12], time series analysis and adaptive equalization [4].

There are, in general, two types of RBF network training algorithms: (1) Batch mode and (2) Sequential mode. For the batch mode approach, a complete sample set is presented at the whole training process. In addition, the parameters of neurons are determined from the correlation among the training samples. Vapnik [10] suggests using the sample set to formulate the energy function of Support Vector Machine. By minimizing this function, the network parameters are determined. In contrast, the sequential approaches [1, 7] only requires a sub-set of the previous training set, or even just the current sample for training. A significant contribution to sequential training algorithm was made by Platt [5] through the development of Resource Allocation Network (RAN), which hidden neurons are added sequentially based on the new data. Kadirkamanathan and Niranjan proposed to use EKF instead of LMS for updating the neurons' center, variance and weight to improve the regression accuracy. It is known as the enhanced RAN and named RANEKF[9]. In this paper, we propose a new type of sequential RBF neural network called "*Memory Regression Network MRN*". Different from the previous network architecture, MRN involves a set of extra elements called "Experience". As a result, only a subset of samples is required for training.

This paper is organized as follows. In section 2, the objective, and hence the advantages of sequential RBF network are discussed. The architecture of MRN and its training algorithm are introduced in Section 3. 2 sets of functions are estimated so as to illustrate the performance of the proposed algorithm in Section 4. And a conclusion is drawn in Section 5.

2 Sequential RBF Network

Sequential learning is an important component of learning in many applications of intelligent systems [3, 13]: adaptive control, time series prediction, financial engineering and DNA sequencing. Moreover, due to the flexibility on the experience learning, this type of neural network is suitable for modeling a time-variant system.

Instead of training a neural network by using the pre-stored data, the sequential learning consists of propagating the experience's input vector forward through the neural network to compute its output. By comparing of this output with a reference (desired output), the regression error is computed. Finally the neuron weights are modified in such way as to reduce the sum of square error. Therefore, instead of training the weights of all neurons, the network needs only concern the current sample.

A sequential network is commonly used on robot applications. A robot collects samples by interacting with the environment, which is relative longer than that of the training process. Therefore, different from the performance measurement of off-line training that

0-7803-8566-7/04/$20.00 © 2004 IEEE.

evaluating the training speed, the performance of a sequential trained network should be measured on the sample size involved for training.

3 Memory Regression Network – MRN

3.1. Architecture of MRN

In general, the performance of a neural network is measured by two qualities: *Accuracy* and *Generalization*. A large-scale network guarantees a high accuracy but low generalization. In contrast, a small-scale network leads to a poor accuracy but preserves a better generalization. It is a trade-off between accuracy and generalization To tackle this problem, Platt [5] proposed a sequential training process called "Growing and Pruning GP" strategy, which is used to classify the training state as one of the 3 possible states listed in below:

1. Inserting a neuron for improving the accuracy.
2. Deleting a neuron for enhancing the generalization.
3. Maintaining the current network size and adjusting the internal parameters to improve the accuracy.

Since the network tries to maximize its accuracy, neurons' parameters will be adjusted such that:

$$f(x) = y + \varepsilon \text{ where } \varepsilon \text{ approach to zero} \qquad \text{Eq. (1)}$$

However, as GP considers only the latest sample $S = [x_s \mid y_s]$ with its regression error $\varepsilon = f(x_s) - y_s$, the network has no ability to classify if the regression error ε caused by either itself or S such that $y_s = y_f + \varepsilon_s$ where y_f is the noise-free output and ε_s is the embedded noise. Alternatively, according to the Eq. (1), the network will be trained in which ε_s is embedded to the network, i.e. $f(x_s) = y_s + \varepsilon_s$. The embedded noise doses not only degrade the network performance, but also occupy more neuron, and hence requires longer computation time. To prevent from the effect that sensitive to noise, statistical method is a well-known approach to remove the Gaussian noise embedded in the sample set effectively. Mean Square Error (MSE) is an popular measurement to reflect the goodness of a solution. In order to apply the statistical approach, MRN introduces a new type of memory unit called *experience* in which stores the sample for calculating the corresponding MSE. Therefore, totally 2 types of memory units are involved inside the MRN: the short-term memory unit (STMU) *experience* is used for against the pattern noise while *neuron* is a long-term memory unit (LTMU) for estimating the target function. In often, the size of *experience* is larger than that of *neuron*.

3.2 Training Algorithm of MRN

Memory Regression Network is a type of online RBF network such that its training algorithm is simulating the learning behavior of human. In general, Human Learning Behavior (HLB) can be divided into 3 stages:

Stage 1: Sample Memorizing

Conceptually, knowledge to environment Ψ is achieved by the experiences collected from Ψ. As more experiences are collected, the constructed knowledge is more accurate. Therefore, for a human that survive at an initially unknown Ψ, he prefers to directly memorize the experiences collected by interacting with Ψ. This strategy guarantees that the formulated knowledge is most benefit to his status.

Stage 2: Knowledge Generalization

As the number of collected experience excess from the capacity of memory, a human tries to generalize the experiences by expressing some of them as the correlation of the others.

Stage 3: Knowledge Selection

For the case that a human cannot memorize more distinct experience, he will try to select the contributed experiences to establish the knowledge for surviving. It is found that the experiences and the knowledge are mutually dependent as experiences achieve knowledge and the knowledge reflects the contribution of experiences.

In the view of MRN, the experience in HLB is represented as STMU while the knowledge is expressed as the regressive function of MRN. Similar to the HLB described in previous, MRN training algorithm is divided into 3 phases. For the first phase, the collected samples are simply batched by the STMU in which the centers of neurons are assigned as the same of sample input **x**. Further, the weights of neurons can be determined by using the Matrix Inversion Method (MIM). As the number of *experience* increases, MIM has no longer applicable. The network improves the accuracy and generalization by minimizing the regression error of the *experience* set, which is equivalent to the stage 2 of HLB. For the case that all *experiences* are in used while a new sample is collected, the *experience* with minimum regression error will be replaced by the new sample (stage 3 of HLB). The training process of MRN is mainly divided into 3 phases:

$$\text{Training Procedure} = \begin{cases} \text{Phase 1} & \text{if } N_s \le N_e \text{ and } N_s \le N_n \\ \text{Phase 2} & \text{if } N_s > N_n \\ \text{Phase 3} & \text{if } N_s > N_e \end{cases}$$

where N_e, N_n and N_s are the number of experience, neuron and collected sample respectively. The details of 3 phases are described as follows:

Phase 1: Sample Memorizing

The parameters of i^{th} experience $P_i = [\beta_i, e_i]$ are assigned as $\beta_i \leftarrow x_s$ and $e_i \leftarrow y_s$. In addition, the center and variance of i^{th} neuron $H_i = [\mu_i, \sigma_i, M_i]$ are assigned as:

$$\vec{\mu}_k \leftarrow \vec{x}_s; \quad \sigma_k \leftarrow \sqrt{\frac{\min_{m \neq i} \|\vec{\mu}_k - \vec{\beta}_m\|}{-8 \ln 0.8}}$$

Consequently, the magnitudes of neuron set [M] can be determined by the Matrix Inversion method, i.e. $[M] = [G]^{-1}[Y]$ where $G_{i,j} = \exp(-\|\mu_j - \mu_i\| / 2\sigma_j^2)$ and $Y_i = e_i$.

Phase 2: Knowledge Generalization

The parameters of i^{th} experience P_i are assigned as $\beta_i \leftarrow x$ and $e_i \leftarrow y$ so as to memorize the incoming sample. If there exists the neuron $H_k = [\mu_k, \sigma_k, M_k]$ such that:

1. $|M_k| \leq |M_i|$ for all i
2. and $|M_k| \leq E(x_s, y_s)$

where $E(\vec{x}, y) = y - \sum_{j=1}^{N_n} M_j \exp\left(-\frac{\|\vec{\mu}_j - \vec{x}\|}{2\sigma_j^2}\right)$

the parameters of H_k will be assigned as:

$$\vec{\mu}_k \leftarrow \vec{x}; \quad \sigma_k \leftarrow \sqrt{\frac{\min_{m \neq i} \|\vec{\mu}_k - \vec{\beta}_m\|}{-8 \ln 0.8}}; \quad M_k \leftarrow E(\vec{x}, y)$$

Due to the neuron adjustment process listed in the above, a Gaussian regression error region is occurred. Instead of adjusting all neurons to minimize the error, a subset of them (Excited neuron set $\{O_i\} = \{\mu_{o,i}, \sigma_{o,i}, M_{o,i}\}$) is extracted for error minimization. The procedures of extracting the excited neuron set are listed in below:

Step 1: Determine the Excitivity of i^{th} neuron (C_i):

$$C_i = \frac{V_i}{U_i}; \quad V_i = \int e^{-\frac{\|\vec{\mu}_i - \vec{x}\|}{2\sigma_i^2}} e^{-\frac{\|\vec{\mu}_e - \vec{x}\|}{2\sigma_e^2}} d\vec{x}; \quad U_i = \int e^{-\frac{\|\vec{\mu}_i - \vec{x}\|}{2\sigma_i^2}} d\vec{x}$$

where μ_e and σ_e is the mean and variance of H_i before the adjustment.

Step 2: Sort $\{C_i\}$ in descending order.

Step 3: Assume that $R(i)$ is the rank of i^{th} neuron H_i.

Step 4: By giving the Correlation constant η, H_i is defined as the excited neuron if:

$$\sum C_h \leq \eta \sum_{j=1}^{p} C_j \text{ for } R(i) \geq R(h) \text{ where } p \text{ is the network size}$$

In addition, the experience $P_k = [\beta_i, e_i]$ is regarded as the element of excited experience $\{F_i\} = \{\beta_{e,i}, e_{e,i}\}$ if $\|\beta_i - \mu_{o,j}\| \leq a\sigma_{o,j}$ for some j. By minimizing R defined at Eq. (1), the network with minimum error can be achieved.

$$R = \sum_{i=1}^{N_p} \left(e_{p,i} - \sum_{j=1}^{N_n} M_{o,j} \exp\left(-\frac{\|\vec{\mu}_{o,j} - \vec{\beta}_i\|}{2\sigma_{o,j}^2}\right)\right)^2 \qquad \text{Eq. (1)}$$

Phase 3: Knowledge Selection

If there exists an experience $P_k = [\beta_k, e_k]$ such that

1. $E(\beta_k, e_k) \leq E(\beta_i, e_i)$ for all i
2. $E(\beta_k, e_k) \leq E(x, y)$

β_k and e_k will be adjusted as x and y respectively.

4 Experimental Results

4.1 Regression Ability of MRN

In this section, the regression performance of MRN is examined by three different problems in the function approximation area:

(a) $y = x \quad \text{for } x \in [0,1]$

(b) $y = 0.8\sin(50x)e^{-x} \quad \text{for } x \in [0,1]$

(c) $y = \frac{\sin(6\pi\sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2})}{6\pi\sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2}} \quad \text{for } x_1, x_2 \in [0,1]$

Training samples with its desired output are provided for the functions (a), (b) and (c) respectively. All training inputs are selected randomly under the uniform distribution. After applying the proposed training algorithm, the parameters of the neurons are determined. In order to analyse the effect of network size on the MRN's generalization, the experiments are repeated with various combinations of network size N_n and the number of experience N_e. The network size is varied from 10 to 100 and the number of experience is assigned as 1, 2, 3 and 4 times of the network size. Furthermore, the generalization L of the resultant MRN is defined as:

$$L = \begin{cases} \frac{1}{101} \sum_{i=0}^{100} |E(0.001i, y)| & D = 1 \\ \frac{1}{101^2} \sum_{i=0}^{100} \sum_{j=0}^{100} |E([0.001i, 0.001j], y)| & D = 2 \end{cases}$$

where D is the function dimension. For each combination of N_n and N_e, 10 trials of experiments are undergone in order to obtain the average generalization. Table 1, 3 and 5 list the average generalizations of MRN on the regression of function (a), (b) and (c) respectively. Moreover, the computational time of MRN on function (a), (b) and (c) are listed at the Table 2, 4 and 6 respectively. The results show that MRN requires smaller training set to construct the output with generalization similar to the one trained by the RAN.

In all simulations, the Correlation constant is chosen as 0.85. Moreover, they are processed by a PC with 1.7GHz CPU and 256MB memory.

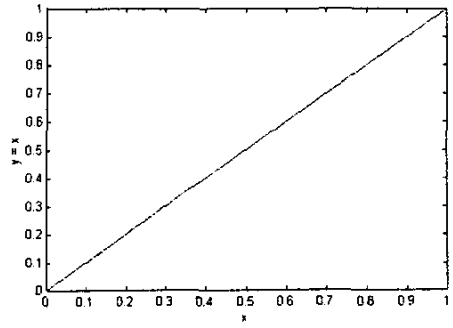


Figure 1a Desired Output of Function (a)

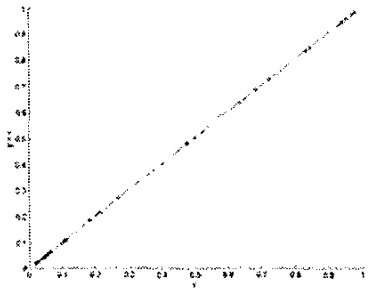


Figure 1b Regression output of Function (a) by the trained MRN with 10 neurons and 30 experiences

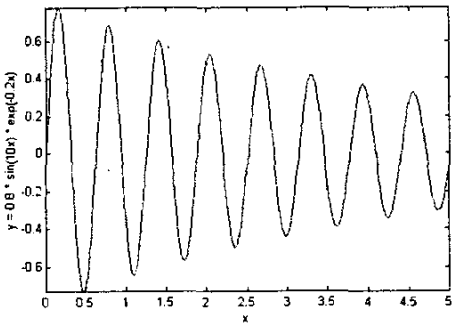


Figure 2a Desired Output of Function (b)

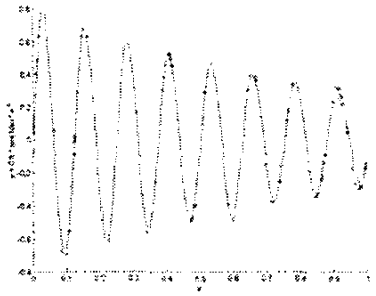


Figure 2b Regression output of Function (b) by the trained MRN with 40 neurons and 80 experiences

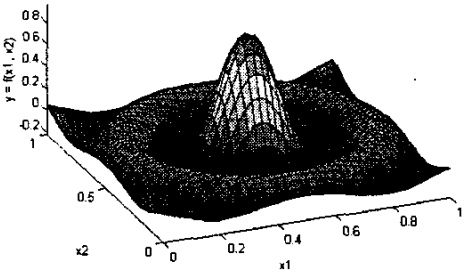


Figure 3a Desired Output of Function (c)

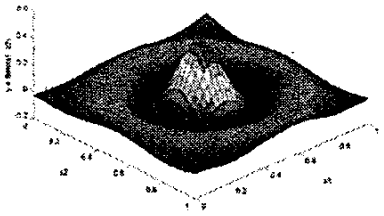
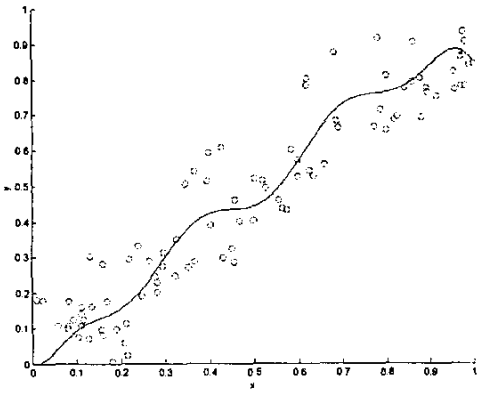


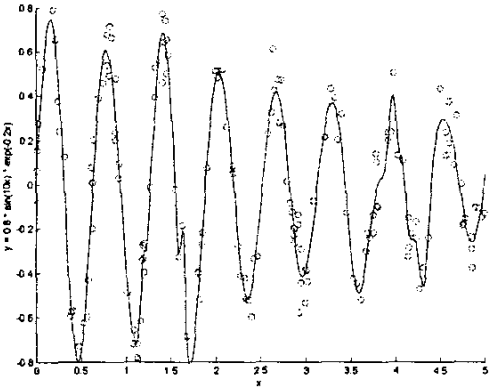
Figure 3b Regression output of Function (c) by the trained MRN with 40 neurons and 100 experiences

4.2 Noise Sensitivity of MRN

In this experiment set, noise sensitivity of MRN was evaluated. The sample selection strategy in this experiment is the same of the previous experiment except that a Gaussian noise with mean = 0 and variance = 0.1 is added to the sample output. In order to handle the noise pattern, more training samples are required for noise reduction in which the sample size is double of previous section. Similar to the previous result illustration, the solid-lines at fig. 2 show the regression result while the circles represent the training samples presented. The results at table 2 show that the proposed training algorithm is robust on handling noisy pattern.



a. Regression Output of Function (a) in Section 4.2



b. Regression Output of Function (b) in Section 4.2

Figure 4 Resultant Approximated functions at Section 4B

5 Conclusion

In this paper, we proposed a novel RBF network called "Memory Regression Network MRN". Its training algorithm is somewhere between the batch mode and the sequential mode. Different from the existing RBF network [2, 11] that comprise of neuron only, MRN involves an extra short-term memory called "Experience". By simulating human's learning behavior with the concept of excited elements, the MRN is well trained within a reasonable time while the accuracy and generalization of MRN is acceptable. The results on function approximation show that the proposed network can handle both static and highly oscillated patterns. In addition, we have tested the network with noisy pattern with a noise level up to 20% of the desired value. Based on the approximated results, it is concluded that the proposed network is quite insensitive to noise.

Table 1 Average generalization error of function (a) in Section 4.1

Number of Neuron	Number of Experience / Number of Neuron			
	1	2	3	4
10	0.011628	0.002360	0.002695	0.000749
20	0.004863	0.003467	0.001163	0.000795
30	0.002702	0.000627	0.000618	0.000738
40	0.000546	0.001424	0.000507	0.000304
50	0.002740	0.000604	0.000445	0.000404
60	0.000644	0.000363	0.000405	0.000272
70	0.000993	0.000343	0.000425	0.000273
80	0.001226	0.000348	0.000296	0.000253
90	0.000534	0.000292	0.000279	0.000218
100	0.000592	0.000301	0.000271	0.000262

Table 2 Average computational time (sec.) of function (a) in section 4.1

Number of Neuron	Number of Experience / Number of Neuron			
	1	2	3	4
10	0.3	0.4	0.6	0.6
20	0.7	0.7	1.4	0.8
30	0.8	1.1	1.7	1.2
40	1.1	1.2	1.7	1.8
50	1.3	1.7	1.9	2.8
60	1.3	2.0	2.8	4.1
70	1.4	2.5	3.7	4.7
80	1.6	3.0	4.4	5.9
90	2.4	3.6	6.0	6.8
100	2.3	4.2	6.3	7.9

Table 3 Average generalization error of function (b) in Section 4.1

Number of Neuron	Number of Experience / Number of Neuron			
	1	2	3	4
10	0.195087	0.156842	0.156720	0.124429
20	0.135123	0.049631	0.053141	0.032568
30	0.050133	0.016640	0.009887	0.008143
40	0.043754	0.011013	0.004903	0.008709
50	0.038812	0.009085	0.004739	0.002468
60	0.017335	0.005304	0.004224	0.002344

Table 4 Average computational time (sec.) of function (b) in section 4.1

Number of Neuron	Number of Experience / Number of Neuron			
	1	2	3	4
10	0.4	0.7	0.8	1.0
20	0.8	1.1	1.2	1.1
30	1.1	1.5	1.6	2.3
40	1.3	1.8	2.2	2.7
50	1.7	2.0	2.6	3.8
60	2.0	2.1	2.9	3.8

70	0.009669	0.002824	0.002368	0.001608
80	0.010254	0.004480	0.002851	0.003198
90	0.005769	0.002367	0.001938	0.002824
100	0.004833	0.003620	0.002696	0.001915

Table 5 Average generalization error of function (c) in Section 4.1

Number of Neuron	Number of Experience / Number of Neuron			
	1	2	3	4
10	0.118347	0.062102	0.055085	0.049227
20	0.080891	0.022624	0.019445	0.015522
30	0.051761	0.014739	0.010607	0.007447
40	0.042516	0.012375	0.007334	0.005654
50	0.033631	0.009839	0.005203	0.004182
60	0.031125	0.008011	0.004657	0.002753
70	0.023137	0.008185	0.003896	0.003141
80	0.028118	0.004951	0.004472	0.002617
90	0.029392	0.006776	0.003908	0.002321
100	0.020538	0.005429	0.003790	0.002426

70	2.3	2.8	3.9	4.4
80	3.3	3.3	4.5	6.3
90	3.4	3.8	4.9	6.1
100	3.6	4.5	6.1	7.6

Table 6 Average computational time (sec.) of function (c) in Section 4.1

Number of Neuron	Number of Experience / Number of Neuron			
	1	2	3	4
10	< 1	1	1	1
20	< 1	2	5	5
30	1	5	8	13
40	2	7	12	26
50	2	11	26	29
60	3	17	26	43
70	5	26	48	75
80	7	30	53	81
90	7	34	60	117
100	9	39	80	119

References

[1] Changxue Ma, Randolph M.A., Drish J., "*A support vector machines-based rejection technique for speech recognition*", Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001 (ICASSP '01). 2001, Vol. 1, Page(s): 381 –384, vol.1

[2] de Freitas N., Milo M., Clarkson P., Niranjan M., Gee A., "Sequential support vector machines", *Neural Networks for Signal Processing IX*, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop, 23-25 Aug. 1999, Page(s): 31 –40

[3] Fravolini, M.L., Campa G., Napolitano K., Yongkyu Song, "Minimal resource allocating networks for aircraft SFDIA", *Proceedings. 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2001, Page(s): 1251 –1256, Vol. 2

[4] Jianping D., Sundararajan N., Saratchandran P., "*Nonlinear magnetic storage channel equalization using minimal resource allocation network (MRAN)*", IEEE Transactions on Neural Networks, Vol. 12, Issue: 1, 2001, Page(s): 171 –174

[5] J. Platt, "A resource-allocating network for function interpolation", *Neural Computation*, Vol. 3, Page(s). 213-225, 1991.

[6] Kai Tik Chow, Tong Lee, "*Image approximation and smoothing by support vector regression*", Proceedings of International Joint Conference on Neural Networks 2001, Volume: 4, Page(s): 2427 - 2432, vol. 4.

[7] Ruping S., "Incremental learning with support vector machines", *Proceedings of IEEE International Conference on Data Mining 2001*, ICDM 2001, Page(s): 641 – 642, 2001

[8] Sankar R., Sethi N.S., "*Robust speech recognition techniques using a radial basis function neural network for mobile applications*", Proceedings of IEEE Southeastcon '97. 'Engineering New Century', 1997, Page(s): 87 –91.

[9] V. Kadirkamanathan, M. Niranjan, "A function estimation approach to sequential learning with neural networks", *Neural Computation*, Vol. 5, Page(s) 954-975, 1993

[10] V. N. Vapnik, C. Cortes, "*Support Vector Networks*", Machine Learning, 20(3), 273-297, 1995

[11] W. Schiffmann, M. Joost, and R. Werner, "*Optimization of the backpropagation algorithm for training multilayer perceptrons*", *Technical Report*, Univerity of Koblnez, Institute pf Physics, 1993.

[12] Yonghong S., Saratchandran P., Sundararajan N., "Minimal resource allocation network for adaptive noise cancellation", *Electronics Letters*, Volume: 35, Issue: 9, Page(s): 726 –728

[13] Yue Xicai, Ye Datian, Liu Ming, "*Text-independent speaker identification by genetic clustering radial basis function neural network*", Proceedings of the 23rd Annual International Conference of the Engineering in Medicine and Biology Society IEEE, 2001, Vol.: ,2, Page(s): 1777 –1780, Vol. 2.