Extreme Learning Machine with Randomly Assigned RBF Kernels*

Guang-Bin Huang and Chee-Kheong Siew
School of Electrical and Electronic Engineering
Nanyang Technological University
Nanyang Avenue
Singapore 639798
Email: {egbhuang, ecksiew}@ntu.edu.sg.

Abstract

A new learning algorithm called **e**xtreme **l**earning **m**achine (ELM) has recently been proposed for **s**ingle-hidden **l**ayer **f**eedforward neural **n**etworks (SLFNs) with additive neurons to easily achieve good generalization performance at extremely fast learning speed. ELM randomly chooses the input weights and analytically determines the output weights of SLFNs. It is proved in theory that ELM can be extended to **s**ingle-hidden **l**ayer **f**eedforward neural **n**etworks (SLFNs) with **r**adial **b**asis **f**unction (RBF) kernels - RBF networks, which allows the centers and impact widths of RBF kernels to be randomly generated and the output weights to be simply analytically calculated instead of iteratively tuned. The kernel function of ELM can be any nonlinear bounded integrable function which is almost continuous anywhere. Interestingly, the experimental results show that the ELM algorithm for RBF networks can complete learning at extremely fast speed and produce generalization performance very close to that of SVM in some benchmarking function approximation and classification problems.

Index terms - Radial basis function network, feedforward neural networks, real time learning, extreme learning machine, ELM, arbitrary kernels.

*The preliminary idea of the ELM algorithm for RBF networks has been shown in the Proceedings of the Eighth International Conference on Control, Automation, Robotics and Vision (ICARCV 2004), Dec 6-9, Kunming, China.

1 Introduction

It is clear that gradient descent based learning methods generally run very slowly due to improper learning steps or may easily converge to local minimums. Many iterative learning steps are required by such learning algorithms in order to obtain better learning performance and on the other hand cross-validation and/or early stopping need to be used in order to prevent over-fitting.

It is not surprising to see that it may take several minutes, several hours and several days to train neural networks in most applications. Furthermore, it should not be neglected that more time need to be spent on choosing appropriate learning parameters (i.e, learning rates) by trial-and-error in order to train neural networks properly using traditional methods. Unlike traditional popular implementations, for **s**ingle-hidden **l**ayer **f**eedforward neural **n**etworks (SLFNs) with additive neurons we have recently proposed a new learning algorithm called **e**xtreme **l**earning **m**achine (ELM)[1] which randomly chooses the input weights and the hidden neurons' biases and *analytically determines* the output weights of SLFNs. Input weights are the weights of the connections between input neurons and hidden neurons and output weights are the weights of the connections between hidden neurons and output neurons. In theory, it has been shown [2, 3] that SLFNs' input weights and hidden neurons' biases need not be adjusted during training and one may simply randomly assign values to them. The experimental results based on a few artificial and real benchmark function regression and classification problems have shown that compared with other gradient-descent based learning algorithms (such as backpropagation algorithms (BP)) for feedforward networks this ELM algorithm tends to provide better generalization performance at extremely fast learning speed and the learning phase of many applications can now be completed within seconds[1]. Baum[4] has also claimed that (seen from simulations) one may fix the connections on one level and simply *adjust* the connections on the other level and *no gain is possible* by using an algorithm able to adjust the weights on both levels simultaneously. It should be noted that tuning methods may neither be suitable for non-differential activation functions nor prevent the troubling issues such as stopping criteria, learning rate, learning epoches, and local minima. However, ELM algorithm may avoid these difficulties very well.

The main target of this paper is to extend ELM from SLFNs with additive neurons case to SLFNs with **r**adial **b**asis **f**unction (RBF) kernels case - RBF networks. It will simply be proven in theory in this paper that instead of tuning the centers and impact widths of RBF kernels, we may just simply randomly choose values for these parameters and analytically calculate the output weights of RBF networks. Very interestingly, testing results on a few benchmark artificial and real function regression and classification problems ELM for RBF networks can reach the generalization performance very close to those obtained by the SVMs but complete learning phase at extremely fast speed.

This paper is organized as follows. Section 2 simply proves that from the function approximation point of view the RBF kernels can be randomly chosen. Section

3 extends the ELM learning algorithm from SLFN case to RBF networks case. Performance evaluation is presented in Section 4. Conclusions are given in Section 5.

2 Universal Approximation of SLFNs with Arbitrary RBF Kernels

The output of a RBF network with $\tilde{N}$ kernels for an input vector $\mathbf{x} \in \mathbf{R}^d$ is given by

$$f_{\tilde{N}}(\mathbf{x}) = \sum_{i=1}^{\tilde{N}} \beta_i \phi_i(\mathbf{x}) = \phi(\mu_i, \sigma_i, \mathbf{x}) \tag{1}$$

$\beta_i = [\beta_{i1}, \beta_{i2}, \cdots, \beta_{im}]^T$ is the weight vector connecting the ith kernel and the output neurons and $\phi_i(\mathbf{x})$ is the output of the ith kernel. $\mu_i = [\mu_{i1}, \mu_{i2}, \cdots, \mu_{in}]^T$ is the ith kernel's center and σ_i is its impact width. ϕ is a radially symmetric kernel function and it is assumed that $\phi(x)$ is nonlinear bounded integrable and almost continuous anywhere. Such kernels include the popularly used Gaussian function:

$$\phi(\mu_i, \sigma_i, \mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mu_i\|^2}{\sigma_i^2}\right) \tag{2}$$

and some other functions which may not be continuous or differential functions.

Let $L^2(X)$ be a space of functions f on a measurable compact subset X in the d-dimensional Euclidean space $\mathbf{R}^d$ such that $|f|^2$ are integrable. The norm in L^2 space will be denoted as $\|\cdot\|$, and the closeness between the output f_n of a RBF network and the target function f is measured by the L^2-norm distance:

$$\|f_n - f\| = \left[\int_X |f_n(\mathbf{x}) - f(\mathbf{x})|^2 d\mathbf{x}\right]^{1/2} \tag{3}$$

In this paper the sample input space X is always considered as a bounded measurable compact subset of the Euclidean space $\mathbf{R}^d$. It is very unlikely that one will deal with nonmeasurable sample input sets in the applications of neural networks. As rigorously proved in Huang et al.[5], we have the following universal approximation theorem for RBF networks:

Theorem 2.1. *For any continuous target function f, given any RBF network function sequence f_n with their kernel centers μ_i and impact factors σ_i randomly generated based on any continuous sampling distribution probability, we have $\|\lim_{n\to+\infty} f_n - f\| = 0$ with properly chosen output weights β_i.*

According to Theorem 2.1, obviously furthermore we can have

Theorem 2.2. *For any continuous target function f and any small positive constant $\delta > 0$, given any RBF network function sequence f_n with their kernel centers μ_i and impact factors σ_i randomly generated based on any continuous sampling distribution probability, there exists a positive integer $\tilde{N}$ such that the probability we have $\|f_{\tilde{N}} - f\| < \delta$ with properly chosen output weights β_i is one.*

Thus, for any continuous target function f and any small positive constant $\delta > 0$, given any RBF network function sequence f_n with their kernel centers μ_i and impact factors σ_i randomly generated based on any continuous sampling distribution probability, given $\tilde{N}$ RBF kernels, we may only need to find the output weights β_i such that

$$\min_{\beta} \|f_{\tilde{N}} - f\| = \left[\int_X |f_{\tilde{N}}(\mathbf{x}) - f(\mathbf{x})|^2 \, d\mathbf{x} \right]^{1/2} \tag{4}$$

3 Extension of Extreme Learning Machine to RBF Case

In this section, we will show that the ELM previously proposed for the case of SLFNs with additive neurons [1] can *linearly* be extended to the case of SLFNs with RBF kernels.

3.1 Approximation Problem of RBFs

For N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \cdots, x_{in}]^T \in \mathbf{R}^d$ and $\mathbf{t}_i = [t_{i1}, t_{i2}, \cdots, t_{im}]^T \in \mathbf{R}^m$, RBFs with $\tilde{N}$ kernels can be mathematically modeled as

$$\sum_{i=1}^{\tilde{N}} \beta_i \phi_i(\mathbf{x}_j) = \mathbf{o}_j, \;\; j = 1, \cdots, N \tag{5}$$

Similar to SLFN case, that standard RBFs with $\tilde{N}$ kernels can approximate these N samples with zero error means that $\sum_{j=1}^{\tilde{N}} \|\mathbf{o}_j - \mathbf{t}_j\| = 0$, i.e., there exist β_i, μ_i and σ_i such that

$$\sum_{i=1}^{\tilde{N}} \beta_i \phi(\mu_i, \sigma_i, \mathbf{x}_j) = \mathbf{t}_j, \;\; j = 1, \cdots, N. \tag{6}$$

The above N equations can be written compactly as:

$$\mathbf{H}\beta = \mathbf{T} \tag{7}$$

where

$$\mathbf{H}(\mu_1, \cdots, \mu_{\tilde{N}}, \sigma_1, \cdots, \sigma_{\tilde{N}}, \mathbf{x}_1, \cdots, \mathbf{x}_N) = \begin{bmatrix} \phi(\mu_1, \sigma_1, \mathbf{x}_1) & \cdots & \phi(\mu_{\tilde{N}}, \sigma_{\tilde{N}}, \mathbf{x}_1) \\ \vdots & \cdots & \vdots \\ \phi(\mu_1, \sigma_1, \mathbf{x}_N) & \cdots & \phi(\mu_{\tilde{N}}, \sigma_{\tilde{N}}, \mathbf{x}_N) \end{bmatrix}_{N \times \tilde{N}} \tag{8}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m} \tag{9}$$

Similar to SLFNs [6, 2], $\mathbf{H}$ is called the hidden layer output matrix of the RBF network; the ith column of $\mathbf{H}$ is the output of the ith kernel with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$.

3.2 Minimum Norm Least-Squares Solution of RBF

Since in practice the network is trained using finite training samples $(\mathbf{x}_i, \mathbf{t}_i)$, where $\mathbf{x}_i \in X$, finding the $\min_\beta \|f_{\tilde{N}} - f\|$ can be equivelant to $\min_\beta \|\mathbf{H}(\mu_1, \cdots, \mu_{\tilde{N}}, \sigma_1, \cdots, \sigma_{\tilde{N}})\beta - \mathbf{T}\|$. According to Theorem 2.2 RBF kernels can be randomly generated instead of being tuned. For fixed kernel centers μ_i and impact widths σ_i, to train an RBF is simply equivalent to finding a least-squares solution $\hat{\beta}$ of the linear system $\mathbf{H}\beta = \mathbf{T}$:

$$\|\mathbf{H}(\mu_1, \cdots, \mu_{\tilde{N}}, \sigma_1, \cdots, \sigma_{\tilde{N}})\hat{\beta} - \mathbf{T}\| = \min_{\beta} \|\mathbf{H}(\mu_1, \cdots, \mu_{\tilde{N}}, \sigma_1, \cdots, \sigma_{\tilde{N}})\beta - \mathbf{T}\| \quad (10)$$

However, in most practical applications $\tilde{N} \neq N$ (the number of kernels may be much less than the number of distinct training samples, $\tilde{N} \ll N$,) $\mathbf{H}$ is a nonsquare matrix and there may not exist β_i $(i = 1, \cdots, \tilde{N})$ such that $\mathbf{H}\beta = \mathbf{T}$. According to our previous analysis[1], the unique smallest norm least-squares solution $\hat{\beta}$ of the above linear system is:

$$\hat{\beta} = \mathbf{H}^\dagger \mathbf{T} \quad (11)$$

where $\mathbf{H}^\dagger$ is the Moore-Penrose generalized inverse[7].

Relationship between weight norm and generalization performance

Bartlett[8] pointed out that for feedforward networks with many small weights but small squared error on the training examples, the *Vapnik-Chervonenkis* (VC) dimension (and hence number of parameters) is irrelevant to the generalization performance. Instead, the magnitude of the weights in the network is more important. The smaller the weights are, the better generalization performance the network tends to have. Since RBF networks look like the standard feedforward neural networks except that different hidden neurons are used, it is reasonable to conjecture that Bartlett's conclusion for feedforward neural networks may be valid in RBF network case as well. Our approach of finding the kernels and output weights not only reaches the smallest squared error on the training examples but also obtains the smallest output weights. Thus, reasonably speaking, this approach may tend to have good generalization performance, which is consistent with our simulation results in a few benchmark problems.

Thus, similar to SLFNs, the extreme learning machine (ELM) for RBF networks can now be summarized as follows:

ELM Algorithm for RBFs: Given a training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^d, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \cdots, N\}$, kernel function ϕ, kernel number $\tilde{N}$,

step 1 Assign arbitrarily kernel centers μ_i and impact widths σ_i, $i = 1, \cdots, \tilde{N}$.

step 2 Calculate the hidden (kernel) layer output matrix $\mathbf{H}$.

step 3 Calculate the output weight β

$$\beta = \mathbf{H}^{\dagger}\mathbf{T} \tag{12}$$

where $\mathbf{T} = [\mathbf{t}_1 \cdots \mathbf{t}_N]^T$

4 Performance Evaluation

In this section, the performance of the proposed ELM learning algorithm for RBF networks is compared with the popular Support Vector Machines (SVMs) [9, 10, 11, 12] on several benchmark problems in function regression and classification areas. It shows that the ELM-RBF could reach good generalization performance which is very close to SVMs, however, our ELM-RBF can be simply conducted and runs much faster, especially for function regression applications. 50 repeated trials have been conducted for both ELM-RBF and SVM for each benchmark problem and the training and testing data are randomly generated at each trial. The average training and testing performance of both ELM-RBF and SVM are given in this section. 8,000 training data and 12,640 testing data randomly generated from the California Housing database[1]. for each trial of simulation. 3000 training data and 1177 testing data are randomly generated from the Abalone database[13] for each trial of simulation as usually done in literature. For Diabetes problem[2], 75% and 25% samples are randomly chosen for training and testing at each trial.

All the simulations for the ELM-RBF algorithm[3] are carried out in MATLAB 6.5 environment running in a Pentium 4, 1.9 GHZ CPU. The simulations for SVM are carried out using popular compiled C-coded SVM packages: LIBSVM[4] running in the same PC. Both ELM-RBF and SVM use the same Gaussian kernel function: $\phi(\mathbf{x}, \mu, \sigma) = \exp\left(-\frac{\|\mathbf{x}-\mu\|^2}{\sigma^2}\right)$. The inputs of all cases are normalized into the range $[-1, 1]$ for both the ELM-RBF and SVM algorithms while The output value is normalized into $[0, 1]$.

In order to get good generalization performance, the cost parameter C and kernel parameter γ of SVM need to be chosen appropriately. Similar to Hsu and Lin[14] we estimate the generalized accuracy using different combinations of cost parameters C and kernel parameters γ: $C = [2^{12}, 2^{11}, \cdots, 2^{-1}, 2^{-2}]$ and $\gamma = [2^{4}, 2^{3}, \cdots, 2^{-9}, 2^{-10}]$. Therefore, for each problem we try $15 \times 15 = 225$ combinations of parameters (C, γ) for SVM.

Seen from Tables 1-2, ELM-RBF can get generalization performance close to SVR

[1]http://www.niaad.liacc.up.pt/~ltorgo/Regression/cal_housing.html
[2]ftp://ftp.ira.uka.de/pub/neuron/proben1.tar.gz
[3]Refer to http://www.ntu.edu.sg/eee/icis/cv/egbhuang.htm for ELM source codes.
[4]http://www.csie.ntu.edu.tw/~cjlin/libsvm/

(Support Vector Machine for Regression) with much less kernels and faster learning speed.

Problems	Algorithms	Training Time (seconds)	Testing		No of SVs/ Neurons
			RMS	Dev	
Housing	ELM-RBF	7.1282	0.1265	0.0043	100
	SVR ($C = 2^2, \gamma = 2^1$)	56.6582	0.1181	0.0011	2193.2
Abalone	ELM-RBF	0.0325	0.0779	0.0022	15
	SVR ($C = 2^{10}, \gamma = 2^{-6}$)	44.0474	0.0785	0.0023	457.8300

Table 1: Performance comparison of ELM and SVR in function regression applications: California Housing Prediction and Abalone Age Prediction.

Algorithms	Training Time (seconds)	Success Rate		No of SVs/ Neurons
		Rate	Dev	
ELM-RBF	0.0408	76.48%	2.81%	30
SVM ($C = 2^{11}, \gamma = 2^{-7}$)	0.9436	77.70%	2.94%	294.07

Table 2: Performance (successful testing rate) comparison in real medical diagnosis Application: Diabetes.

5 Conclusions

This paper has extended the **e**xtreme **l**earning **m**achine (ELM) from **s**ingle-hidden **l**ayer **f**eedforward neural **n**etworks (SLFNs) with additive neurons to SLFNs with RBF kernels - RBF networks. The main feature of the proposed ELM for RBF networks is that ELM arbitrarily assigns the kernels instead of tuning them. Compared with the popular SVM, the proposed ELM can be used easily and the ELM can complete learning phase at very fast speed and provide more compact network. As demonstrated in a few simulations on real and artifical benchmark problems, the proposed ELM for RBF networks can achieve comparable generalization performance with SVM. It is worth further systematically investigating the arbitrariness of the RBF kernels.

References

[1] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: A new learning scheme of feedforward neural networks,” in *Proceedings of International Joint Conference on Neural Networks (IJCNN2004) (also in http://www.ntu.edu.sg/eee/icis/cv/egbhuang.htm)*, (Budapest, Hungary), 25-29 July, 2004.

[2] G.-B. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 274–281, 2003.

[3] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental feedforward networks with arbitrary input weights," in *Technical Report ICIS/46/2003*, (School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore), Oct. 2003.

[4] E. Baum, "On the capabilities of multilayer perceptrons," *Journal of Complexity*, vol. 4, pp. 193–215, 1988.

[5] G.-B. Huang, L. Chen, K. Z. Mao, C.-K. Siew, P. Saratchandran, and N. Sundararajan, "Universal approximation using incremental RBF networks with arbitrary kernels," in *Technical Report: ICIS/16/2004*, (School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore), 2004.

[6] G.-B. Huang and H. A. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," *IEEE Transactions on Neural Networks*, vol. 9, no. 1, pp. 224–229, 1998.

[7] D. Serre, "Matrices: Theory and applications," Springer-Verlag New York, Inc, 2002.

[8] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 525–536, 1998.

[9] V. N. Vapnik, "The nature of statistical learning theory," New York: Springer-Verlag, 1995.

[10] V. N. Vapnik, "Statistical learning theory," New York: Wiley, 1998.

[11] C. Cortes and V. N. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[12] A. Smola and B. Schölkopf, "A tutorial on support vector regression," *NeuroCOLT2 Technical Report NC2-TR-1998-030*, 1998.

[13] C. Blake and C. Merz, "UCI repository of machine learning databases," in *http://www.ics.uci.edu/∼mlearn/MLRepository.html*, Department of Information and Computer Sciences, University of California, Irvine, USA, 1998.

[14] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

Guang-Bin Huang received the B.Sc degree in applied mathematics and the M.Eng degree in computer engineering from Northeastern University, China, in 1991 and

1994, respectively, and the Ph.D degree in electrical engineering from Nanyang Technological University, Singapore, in 1999. From June 1998 to May 2001, he was a Research Fellow with Singapore Institute of Manufacturing Technology (formerly known as Gintic Institute of Manufacturing Technology), where he has led/implemented several key industrial projects. Since May 2001, he has been an Assistant Professor in the Information Communication Institute of Singapore (ICIS), School of Electrical and Electronic Engineering, Nanyang Technological University. His current research interests include soft computing and networking. He is a Senior Member of IEEE.

Chee-Kheong Siew is currently the Head of Information Communication Institute of Singapore (ICIS), School of EEE, Nanyang Technological University. He obtained his B. Eng. in Electrical Engineering from University of Singapore in 1979 and MSc. in Communication Engineering, Imperial College in 1987. After six years in the industry, he joined NTU in 1986 and was appointed as the Head of the Institute in 1996. His current research interests include neural networks, packet scheduling, traffic shaping, admission control, service curves and admission control for QoS provisioning, congestion control and multipath routing.