Extreme learning machine adapted to noise based on optimization algorithms

Extreme learning machine adapted to noise based on optimization algorithms

A Vásquez[1], M Mora[1], E Salazar[1], and E Gelvez[2]

[1] Laboratorio de Investigaciones Tecnológicas en Reconocimiento de Patrones, Universidad Católica del Maule, Talca, Chile

[2] Facultad de Ciencias Básicas y Biomédicas, Universidad Simón Bolívar, San José de Cúcuta, Colombia

E-mail: marcomoracofre@gmail.com

Abstract. The extreme learning machine for neural networks of feedforward of a single hidden layer randomly assigns the weights of entry and analytically determines the weights the output by means the Moore-Penrose inverse, this algorithm tends to provide an extremely fast learning speed preserving the adjustment levels achieved by classifiers such as multilayer perception and support vector machine. However, the Moore-Penrose inverse loses precision when using data with additive noise in training. That is why in this paper a method to robustness of extreme learning machine to additive noise proposed. The method consists in computing the weights of the output layer using non-linear optimization algorithms without restrictions. Tests are performed with the gradient descent optimization algorithm and with the Levenberg-Marquardt algorithm. From the implementation it is observed that through the use of these algorithms, smaller errors are achieved than those obtained with the Moore-Penrose inverse.

1. Introduction

Artificial neural networks have been widely used to solve problems of medical diagnosis [1], voice recognition [2], image processing [3], facial recognition [4], between many other applications [1, 2, 5–8]. Despite this, they present clear inconveniences as the high time in the training and the convergence to local minimums. Many works have been developed with the finality of solve these problems and one of the most recent is the method known as extreme learning machine (ELM).

In general, the supervised learning algorithms of the neural networks (multilayer perceptron, convolutional neural networks) are based on non-linear optimization algorithms without restrictions, such as gradient escent or higher speed variants. Extreme learning machine was initially proposed for single-hidden layer feedforward neural networks in [9], and in its training algorithm the input weights and biases of the hidden layer can be chosen randomly, and the weights of the output layer is calculated using a least squares method based on the application of the generalized Moore-Penrose inverse. Unlike other learning algorithms based on gradient descent, the ELM algorithm does not require iterative techniques to adjust the weights and biases of the hidden layer during the training process, so it becomes in a simple learning method, and with an extremely fast learning speed.

Recently, it has been shown in [10] that ELM with weights and biases in the hidden layer assigned arbitrarily and with almost any nonzero activation function correspond to a universal

approximator of functions. This result makes that ELM networks compete with the well known multilayer perceptron and suport vector machine.

Noisy data is a common factor in the processing of data that comes of the real-world. Sources of noise include physical measurement limitations, the use of stochastic simulation models, incomplete sampling of large spaces, and human-computer interaction. Many of the search methods in use nowadays for data with noise can be traced back to the least squares methodology approach, which consists in determine a curve that best is fits to the data set [11–13]. A known fact is that in this type of methods the approximation error is degrades as the Noisy data increases. This paper proposes the use of nonlinear optimization algorithms without restrictions to improve the robustness of the estimate that provided the ELM network in presence of highly noisy data. The algorithms considered are the gradient descent and the Levenberg-Marquardt method, which allow to estimate the model parameters optimally in the sense of the mean square error. The results obtained show that the optimization methods improve the estimation of the parameters of the ELM network than the provided by the Pseudo-Inverse in a acceptable search time.

In section 2, presents the Moore-Penrose generalized inverse solution, the least squares solution for a general linear system, two optimization algorithms that allow to determine the minimum value of a function and a brief introduction to the ELM algorithm. Section 3 explains the materials and methods to address the proposed approaches. Performance evaluation is presented in section 4, while the conclusions are given in section 5.

2. Preliminaries

This section gives a brief introduction to the Moore-Penrose generalized inverse, the least squares solution, the gradient descent, the Levenberg-Marquardt method and the extreme learning machine algorithm.

2.1. Moore-Penrose generalized inverse

The solution of a linear system $\boldsymbol{Ax} = \boldsymbol{y}$ can be calculated very simply by using the generalized inverse of Moore-Penrose [14], where the matrix $\boldsymbol{A}$ can be singular and even rectangular.

A matrix G of order $n \times m$ is the Moore-Penrose generalized inverse of matrix $\boldsymbol{A}$ of order $m \times n$, if it satisfies the conditions of Equation (1).

$$\boldsymbol{AGA} = \boldsymbol{A}, \quad \boldsymbol{GAG} = \boldsymbol{G}, \quad (\boldsymbol{AG})^T = \boldsymbol{AG} \quad \text{y} \quad (\boldsymbol{GA})^T = \boldsymbol{GA}. \tag{1}$$

The Moore-Penrose generalized inverse of matrix $\boldsymbol{A}$ will be denoted by $\boldsymbol{G} = \boldsymbol{A}^{\dagger}$.

2.2. Least squares solution

We say that $\boldsymbol{x}_0$ is a least squares solution for a linear system $\boldsymbol{Ax} = \boldsymbol{y}$, if for any $\boldsymbol{y} \in \mathbb{R}^m$ satisfies the Equation (2).

$$\|\boldsymbol{x}_0\| \leq \|\boldsymbol{x}\|, \quad \forall\, \boldsymbol{x} \in \{\boldsymbol{x} : \|\boldsymbol{Ax} - \boldsymbol{y}\| \leq \|\boldsymbol{Az} - \boldsymbol{y}\|, \quad \forall\, \boldsymbol{z} \in \mathbb{R}^n\}, \tag{2}$$

that is, a solution $\boldsymbol{x}_0$ is said to be a minimum norm least squares solution of a linear system $\boldsymbol{Ax} = \boldsymbol{y}$ if it has the smallest norm among all the least squares solutions.

2.3. Optimization algorithms gradient descent and Levenberg-Marquardt

The gradient descent algorithm is described by the Equation (3), this is most intuitive technique to find minim of a function [15, 16]. The search process of the minimum is carried out through an iterative process following the opposite direction to the gradient of a function E with a convergence rate α.

$$\boldsymbol{W}_{k+1} = \boldsymbol{W}_k - \alpha \nabla E. \tag{3}$$

It can be observe of [15–17] that the gradient descent method and the of Gauss-Newton are two complementary methods in the advantages they provide. In order to make sure that the approximated Hessian matrix $\boldsymbol{H} \approx \boldsymbol{J}^T\boldsymbol{J}$ is invertible [16], Levenberg-Marquardt algorithm introduces another approximation to Hessian matrix by the Equation (4).

$$\boldsymbol{H} \approx \boldsymbol{J}^T\boldsymbol{J} + \mu\boldsymbol{I}, \tag{4}$$

where μ is the combination coefficient, $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{J}$ is the Jacobian matrix. Levenberg and Marquardt proposed a new algorithm based on these observations, whose update rule is a combination of the algorithms mentioned above, such as be shown in Equation (5).

$$\boldsymbol{W}_{k+1} = \boldsymbol{W}_k - \left(\boldsymbol{J}_k^T\boldsymbol{J}_k + \mu\boldsymbol{I}\right)^{-1}\boldsymbol{J}_k. \tag{5}$$

The combination coefficient μ in the previous equation is a parameter that be adjusted in each cycle of according a the evolution of the error. If μ is very small, the $\boldsymbol{J}_k^T\boldsymbol{J}_k$ matrix becomes an approximation to the Hessian and Gauss-Newton algorithm is used. If $\mu \gg 1$, the method becomes analogous to gradient descent.

2.4. Extreme learning machine

For N arbitrary training samples $\{(\boldsymbol{x}_i, \boldsymbol{t}_i)\}_{i=1}^N$, where $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in \mathbb{R}^n$ and $\boldsymbol{t}_i = [t_{i1}, t_{i2}, \ldots, \boldsymbol{t}_{im}]^T \in \mathbb{R}^m$. The ELM learning algorithm for single hidden layer neural networks with L hidden neurons and activation function $g(x)$ are mathematically modeled by Equation (6).

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i g(\boldsymbol{w}_i \cdot \boldsymbol{x}_j + b_i) = \boldsymbol{o}_j, \quad j = 1, \ldots, N, \tag{6}$$

where $\boldsymbol{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ is the weight vector connecting the ith hidden neuron and the input neurons, $\boldsymbol{\beta}_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ is the weight vector connecting the ith hidden neuron and the output neurons, b_i is the threshold of the ith hidden neuron and $\boldsymbol{w}_i \cdot \boldsymbol{x}_j$ denotes the inner product of $\boldsymbol{w}_i$ and $\boldsymbol{x}_j$ [9].

The standard ELM algorithm with L hidden neurons and activation function $g(x)$ can approximate these N samples with zero error means that $\sum_{j=1}^{N} \|\boldsymbol{o}_j - \boldsymbol{t}_j\| = 0$, that is, there exist $\boldsymbol{\beta}_i$, $\boldsymbol{w}_i$ and b_i such as Equation (7).

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i g(\boldsymbol{w}_i \cdot \boldsymbol{x}_j + b_i) = \boldsymbol{t}_j, \quad j = 1, \ldots, N. \tag{7}$$

The Equation (7) can be written in matrix form of Equation (8).

$$\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{T}, \tag{8}$$

where the elements of Equation (8) are given by Equation (9).

$$\boldsymbol{H} = \begin{bmatrix} g(\boldsymbol{w}_1 \cdot \boldsymbol{x}_1 + b_1) & \cdots & g(\boldsymbol{w}_L \cdot \boldsymbol{x}_1 + b_L) \\ \vdots & \ddots & \vdots \\ g(\boldsymbol{w}_1 \cdot \boldsymbol{x}_N + b_1) & \cdots & g(\boldsymbol{w}_L \cdot \boldsymbol{x}_N + b_L) \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_L^T \end{bmatrix} \quad \text{and} \quad \boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_N^T \end{bmatrix}. \tag{9}$$

As mentioned in [18, 19], the matrix $\boldsymbol{H}$ defined in Equation (9) is called the hidden layer output matrix of the neural network; the ith column of $\boldsymbol{H}$ is the ith hidden neuron's output vector with respect to inputs $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$.

If the number of hidden neurons is equal to the number of training samples $L = N$, matrix $\boldsymbol{H}$ is square and invertible, and SLFNs can approximate these training samples with zero error [18–20]. However, in most cases the number of hidden neurons is much less than the number of training samples $L \ll N$, $\boldsymbol{H}$ is a nonsquare matrix and there may not exist $\boldsymbol{w}_i, b_i, \beta_i$ $(i = 1, \ldots, L)$ such that $\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{T}$. Thus, instead one may need to find specific $\hat{\boldsymbol{w}}_i, \hat{b}_i, \hat{\boldsymbol{\beta}}$ $(i = 1, \ldots, L)$ such that Equation (10).

$$\|\boldsymbol{H}(\hat{\boldsymbol{w}}_1, \ldots, \hat{\boldsymbol{w}}_L, \hat{b}_1, \ldots, \hat{b}_L)\hat{\boldsymbol{\beta}} - \boldsymbol{T}\| = \min_{\boldsymbol{w}_i, b_i, \boldsymbol{\beta}} \|\boldsymbol{H}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_L, b_1, \ldots, b_L)\boldsymbol{\beta} - \boldsymbol{T}\|. \tag{10}$$

The Equation (10) is equivalent to minimizing the cost function.

The unknown $\boldsymbol{W}$ in Equations (3) and Equation (5) corresponds to the set of weights and biases $\{\boldsymbol{w}_i, \boldsymbol{\beta}_i, b_i\}$ of the ELM algorithm. In addition, the comparison of results will be carried out minimizing the objective function E defined in Equation (11) using the Levenberg-Marquardt algorithm and the gradient descent.

$$E = \frac{1}{N}\sum_{j=1}^{N}\left(\sum_{i=1}^{L}\boldsymbol{\beta}_i g(\boldsymbol{w}_i \cdot \boldsymbol{x}_j + b_i) - \boldsymbol{t}_j\right)^2. \tag{11}$$

3. Materials and methods

The training and testing data for the ELM algorithm have been generated around of a n grade polynomial function through additive noise, that is Equation (12).

$$\text{data} = f(\boldsymbol{x}) + r \times \text{rand}(N, 1), \tag{12}$$

where $f(\boldsymbol{x}) = a_0 \times \text{ones}(N, 1) + a_1\boldsymbol{x} + a_2\boldsymbol{x}^2 + \cdots + a_n\boldsymbol{x}^n$, N is the set of training samples, $\boldsymbol{x}$ is a vector of dimension N and r is the amplitude of the noise. The analysis was done taking different scenarios with respect to the degree of the polynomial and the number of hidden neurons in the neural network.

In this paper be proposes the gradient descent method (GD-ELM) and the of Levenberg-Marquardt (LM-ELM) for determine the output weights of the ELM algorithm through iterative processes defined in the Equation (3) and Equation (5). The procedure consists in estimate the model parameters optimally in the sense of the mean square error adjusting the noise r of the data obtained from Equation (12). In addition, we choose the sigmoid function (*i.e.* $g(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{b}) = 1/(1 + \exp(-(\boldsymbol{w}\boldsymbol{x} + \boldsymbol{b})))$) as the activation function of GD-ELM, LM-ELM and ELM.

The input weights $\boldsymbol{w}$ are randomly generated from the range of $(-1, 1)$ and the hidden biases $\boldsymbol{b}$ are generated randomly from the range of $(0, 1)$ using a uniform distribution. For each regression problem, the average results over 50 trials are obtained for each algorithm. In this study, the algorithms have been implemented through the MATLAB R2014a environment.

4. Experimental results and analysis

In this section, a series of experiments are carried out to demonstrate the effectiveness of the proposed GD-ELM and LM-ELM approaches. To illustrate the good behavior of GD-ELM and LM-ELM, are compared with the basic ELM. The training problem that serves as example is to reconstruct a polynomial through the network. The results obtained are compared via the root mean square error (RMSE), such as is shown in the following Equation (13).

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)}, \tag{13}$$

where $\hat{y}_i$ is the estimated value and y_i is the true value.

The experiments are conducted on three sets (S_1, S_2 and S_3) specified in Table 1. The coefficients of the polynomials are random constants between -1 and 1. Training and testing samples were taken, where the input are distributed evenly in $[0, 1]$. Furthermore, the number of maximum iterations for gradient descent and the Levenberg-Marquardt algorithm are set in 15000 and 70, respectively.

Table 1. Specification of three datasets.

Polynomial degree (PD)		Training data	Testing data	Noise amplitude
Set 1 (S_1)	90	322	178	r =0.5
Set 2 (S_2)	50	254	146	r =0.4
Set 3 (S_3)	20	189	111	r =0.3

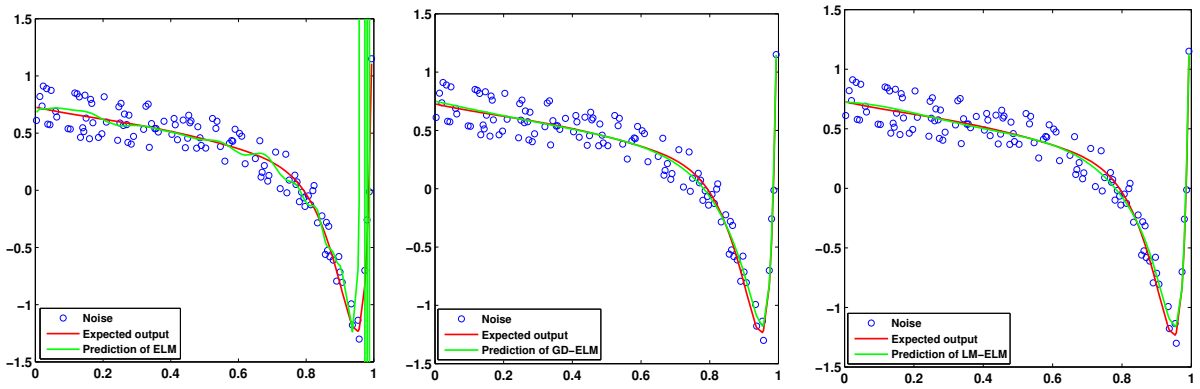
The testing error, the training error, the standard deviation (SD) and learning time are selected as the indicators for performance testing. The smaller testing RMSE denotes the better generalization performance of the algorithm and the smaller SD indicates the better stability of the algorithm. The comparison of the three algorithms is shown in Table 2.

Table 2. Performance comparison of the GD-ELM, LM-ELM and ELM.

Datasets	Algorithms	Testing RMSE	Testing SD	Time	Hidden nodes
S_1	ELM	0.4857	0.2334	0.0092	30
	GD-ELM	0.0886	0.0192	3.2729	30
	LM-ELM	0.2096	0.0897	0.9012	30
S_3	ELM	0.2803	0.2803	0.0018	15
	GD-ELM	0.0843	0.0251	2.4390	15
	LM-ELM	0.1907	0.1128	0.0874	15
S_3	ELM	0.1096	0.0396	0.0030	20
	GD-ELM	0.0331	0.0023	2.6522	20
	LM-ELM	0.0366	0.0019	0.0585	20

From Table 2, we observe that the proposed GD-ELM and LM-ELM algorithms has the RMSE and the SD of testing smallest for the three data sets, which means that GD-ELM and LM-ELM can achieve better generalization performance and stability than ELM.

Figure 1 shows the behavior of the exact and approximate curves obtained by the optimization algorithms for the set S_1. Then, testing samples were used to see the stability of the proposed algorithms compared with the basic ELM. Figure 2 shows the test results obtained to the executing the code 50 times maintaining the same number of hidden neurons. It is observed that GD-ELM and LM-ELM always behave better compared with ELM.



(a) Pseudoinverse (b) Gradient descent (c) Levenberg-Marquardt

Figure 1. Comparison of the actual curve and the output predicted by ELM, GD-ELM and LM-ELM for a noise of radius 0.2. The GD-ELM and LM-ELM algorithms present good approximations to the real model as shown in (b) and (c), however the ELM algorithm begins to oscillate as shown in (a) obtaining bad models.

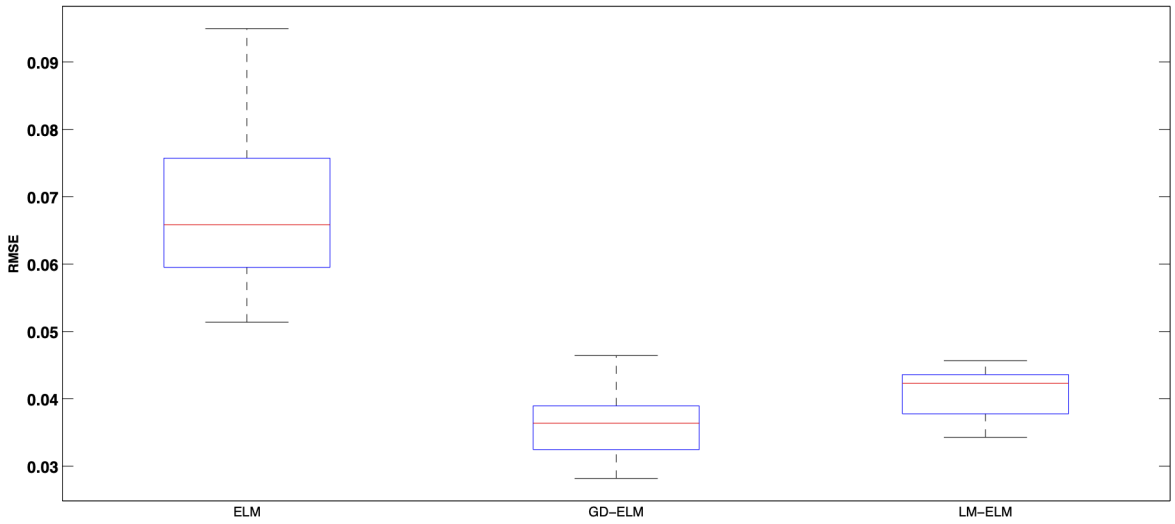


Figure 2. Comparison of the RMSE of ELM, GD-ELM and LM-ELM considering 50 initializations of the weights.

5. Conclusion

In this study, two optimization algorithms were presented for the calculation of ELM parameters for regression problems.Experimental results show that the GD-ELM and LM-ELM algorithms can achieve better test results in compared with the pseudoinverse to determine the output weights of the neural network when working with for highly noisy data. Such as literature indicates, the LM-ELM algorithm is faster than the GD-ELM algorithm, which allows the solution of the problem in a reasonable time.

References

[1] Al-Shayea Q K 2011 Artificial neural networks in medical diagnosis *International Journal of Computer Science Issues* **8(2)** 150

[2] Melin P, Urias J, Solano D, Soto M, Lopez M and Castillo O 2006 Voice Recognition with Neural Networks, Type-2 Fuzzy Logic and Genetic Algorithms *Engineering Letters* **13(2)** 108

[3] Suresh S, Babu R V and Kim H J 2009 No-reference image quality assessment using modified extreme learning machine classifier *Applied Soft Computing* **9(2)** 541
[4] Mohammed A A, Minhas R, Wu Q J and Sid-Ahmed M A 2011 Human face recognition based on multidimensional PCA and extreme learning machine *Pattern Recognition* **44(10-11)** 2588
[5] Bai Z, Huang G B, Wang D, Wang H and Westover M B 2014 Sparse extreme learning machine for classification *IEEE Transactions on Cybernetics* **44(10)** 1858
[6] Huang G, Song S, Gupta J N and Wu C 2014 Semi-supervised and unsupervised extreme learning machines *IEEE Transactions on Cybernetics* **44(12)** 2405
[7] Huang G B, Zhou H, Ding X and Zhang R 2011 Extreme learning machine for regression and multiclass classification *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **42(2)** 513
[8] Huang Z, Yu Y, Gu J and Liu H 2016 An efficient method for traffic sign recognition based on extreme learning machine *IEEE Transactions on Cybernetics* **47(4)** 920
[9] Huang G B, Zhu Q Y and Siew C K 2004 Extreme learning machine: a new learning scheme of feedforward neural networks *Neural Networks* **2** 985
[10] Huang G B, Chen L and Siew C K 2006 Universal approximation using incremental constructive feedforward networks with random hidden nodes *IEEE Trans. Neural Networks* **17(4)** 879
[11] Arnold D V and Beyer H G 2003 A comparison of evolution strategies with other direct search methods in the presence of noise *Computational Optimization and Applications* **24(1)** 135
[12] Cantú-Paz E 2004 Adaptive sampling for noisy problems *Genetic and Evolutionary Computation-GECCO 2004* **3102** 947
[13] Ridout D and Judd K 2002 Convergence properties of gradient descent noise reduction *Physica D: Nonlinear Phenomena* **165(1-2)** 26
[14] Courrieu P 2008 Fast computation of Moore-Penrose inverse matrices *Neural Information Processing - Letters and Reviews* **8(2)** 25
[15] Ranganathan A 2004 The levenberg-marquardt algorithm *Tutoral on LM algorithm* **11(1)** 101
[16] Yu H and Wilamowski B M 2011 Levenberg-marquardt training *Industrial electronics handbook* **5(12)** 1
[17] Gavin H 2013 *The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems* (Durham: Duke University) p 1
[18] Huang G B and Babri H A 1998 Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions *IEEE Transactions on Neural Networks* **9(1)** 224
[19] Huang G B 2003 Learning capability and storage capacity of two-hidden-layer feedforward networks *IEEE Transactions on Neural Networks* **14(2)** 274
[20] Tamura S I and Tateishi M 1997 Capabilities of a four-layered feedforward neural network: four layers versus three *IEEE Transactions on Neural Networks* **8(2)** 251