Philip Reiner, Bogdan M. Wilamowski, "Nelder-Mead Enhanced Extreme Learning Machine", *17-th IEEE Intelligent Engineering Systems Conference, INES 2013,* Costa Rica, June 19-21., 2009, pp. 225-230

Nelder-Mead Enhanced Extreme Learning Machine

Philip Reiner, Bogdan M. Wilamowski
Department of Electrical and Computer Engineering, Auburn University, Auburn, USA
Email: {pdr0001, wilambm}@ auburn.edu

***Abstract* – Many algorithms such as Support Vector Regression (SVR), Incremental Extreme Learning Machine (I-ELM), Convex Incremental Extreme Learning Machine (CI-ELM), and Enhanced random search based Incremental Extreme Learning Machine (EI-ELM) are being used in current research to solve various function approximation problems. This paper presents a modification to the I-ELM family of algorithms targeted specifically at Single Layer Feedforward Networks (SLFN) using Radial Basis Function (RBF) nodes. The modification includes eliminating randomness in both the center positions of the RBF units as well as the widths of the RBF units. This is accomplished by assigning the center of each incrementally added node to the highest point in the residual error surface and using Nelder-Mead's Simplex method to iteratively select an appropriate radius for the added node. Using this technique, the properties of I-ELM that allow for universal approximation and appropriate generalization are preserved, while the sizes of the RBF networks are greatly reduced.**

***Index Terms* – Neural Networks, Machine Learning, RBF Networks, Function Approximation, Radial Basis Function, Support Vector Machines.**

I. Introduction

Artificial neural networks (ANNs) are widely used as universal approximators and have been proven to have good approximation capabilities in the form of feedforward networks. ANNs are popular because they can provide complex nonlinear mappings directly from input data. They are capable of providing models for highly nonlinear and noisy data that is difficult to handle using classical parametric techniques. While there are many different kinds of ANN architectures, SLFNs are considered very simple and have been thoroughly investigated for approximation purposes.

For this paper, only SLFNs with RBF nodes in the hidden layer are investigated. The output function of a SFLN can be described by the following equation:

$$f(x) = \sum_{n=1}^{N} \beta_n g_n(x) \qquad (1)$$

Where n is the number of nodes in the hidden layer of the network, β_i is the output weight for hidden node i, and $g(x)$ is the RBF, or activation function, for node i.

If this activation function, $g(x)$, is continuous, has finite bounds, and is not constant, then a SFLN architecture is capable of producing good approximations of continuous mappings [1]. Furthermore, it was shown by Leshno [2] that SLFNs constructed incrementally can approximate any continuous function with non-polynomial hidden units. It was shown by Park and Sandberg [3] that a network of RBF units can provide good approximations of any continuous function if the RBF units are carefully chosen. Yu *et al*, demonstrated the feasibility of RBF networks for practical purposes in [4].

Several algorithms such as Support Vector Regression (SVR) [5], [6] and the Extreme Learning Machine (ELM) family: I-ELM [7], CI-ELM [8], and EI-ELM [9], can be used to create single hidden layer RBF networks for approximation. These algorithms have all been thoroughly tested on real world datasets, and it has been shown that they are able to achieve good training and validation errors. The algorithm presented in this paper attempts to achieve similar or better training and validation errors while creating a more compact network. It was shown in [10] that more compact networks tend to show better generalization properties.

The rest of this paper is organized in the following way: Section II reviews the fundamentals of RBF networks and the Simplex method for parameter optimization. Section III outlines the details of the I-ELM algorithm and how the proposed NME-ELM algorithm attempts to improve the performance. Section IV presents experimental results done on real world datasets. Finally, Section V summarizes and concludes the results.

II. Computational Fundamentals

Let us establish some indices and terms that will be used throughout the rest of this paper.

- p is the index from 1 to P, where P is the number of input patterns.
- n is the index from 1 to N, where N is the number of RBF nodes in the network.
- d is the index from 1 to D, where D is the number of dimensions in the input set.
- Input patterns, x, of dimension, D, are described as $x_p=[x_{p,1}, x_{p,2}, \ldots, x_{p,D}]$.

Other parameters will be explained as needed.

A. Review of RBF Networks

The standard architecture of a SLFN contains three layers. It consists of P, D, dimensional inputs, N units with activation function $g_n(x)$ and output weight β_n, and a single summing unit at the output layer. Notice that problems with multiple outputs can be processed as a combination of single output problems.

Equation (1) given in the introduction indicates the output of the constructed network. The output of each individual RBF node is given by:

$$g_n(x) = \exp\left(-\frac{\|x_p - c_n\|^2}{\sigma_n}\right) \qquad (2)$$

Where: c_n and σ_n are the center and width of the RBF unit, n, respectively. The Euclidean Norm computation is represented by $\|\quad\|$.

As with most machine learning algorithms, NME-ELM attempts to minimize the error of the problem at each step. In order to do this, the error must be described in a way that can be evaluated and compared to previous errors at each step. The parameters used in this paper to describe the error are:

$$e_p = y_p - f \qquad (3)$$

Where: e_p is the error for pattern, p, and y_p is the desired output for pattern, p. Equation (3) will allow the error for each pattern to be calculated at each training step.

To minimize the error, the magnitude of this value must be minimized. In order to describe and minimize all of the errors at once, the root mean square error (RMSE) is used.

$$RMSE = \sqrt{\frac{\sum_{p=1}^{P} e_p^2}{P}} \qquad (4)$$

B. Nelder-Mead Simplex Algorithm

The main attempt that NME-ELM makes at improving the ELM family of algorithms is to select an appropriate width, σ_n, for each node that minimizes the RMSE. The Nelder-Mead algorithm was originally published in [11]. Since then it has been widely used in a myriad of applications. Its popularity stems from the fact that it is unconstrained and does not require the computation of derivatives of the function to be optimized. However, many studies such as what is presented in [12] show that the Nelder-Mead algorithm has many inefficiencies. Some of these deficiencies were recently corrected in [13].

Keeping this in mind, the Nelder-Mead algorithm was chosen for the task of optimizing the radius of each newly added node for the following reasons: the Nelder-Mead algorithm tends to produce significant improvement over the first few iterations, the Nelder-Mead algorithm does not require many calculations of derivatives only a few function values at each iteration, and finally, it is easy to understand and explain [14]. All of these properties allow the algorithm to be used to very quickly change the radius of each node so that the error is improved.

The Nelder-Mead algorithm was proposed as a method for minimizing a real-valued function $f(x)$ for $x \in R^n$. According to [14], four scalar parameters must be specified to define a complete Nelder-Mead method: coefficients of *reflection* (ρ), *expansion* (χ), *contraction* (γ), and *shrinkage* (α). According to the original publication, these parameters should satisfy:

$$\rho > 0, \chi > 1, \chi > \rho, 0 < \gamma < 1, and\ 0 < \alpha < 1 \qquad (5)$$

In almost all cases (and in this paper) these parameters are chosen to be:

$$\rho = 1, \quad \chi = 2, \quad \gamma = \frac{1}{2}, \ and\ \alpha = \frac{1}{2}. \qquad (6)$$

At the beginning of the k^{th} iteration, $k \geq 0$, a non-degenerate simplex Δ_k is given, as well as its n+1 vertices, each of which is a point in R^n. It is always assumed that iteration k begins by ordering and labeling these vertices as $x_1^k, \ldots, x_{n+1}^k$, such that $f_1^k \leq f_2^k \leq \cdots \leq f_{n+1}^k$. Where f_1^k denotes $f(x_1^k)$. The k^{th} iteration generates a set of n+1 vertices that define a different simplex for the next iteration. In terms of minimizing f, we refer to x_1^k as the best vertex and to x_{n+1}^k as the worst vertex. For explanation purposes in this paper, only one iteration of the Nelder-Mead algorithm will be described and the superscript k will be omitted to avoid confusion. The algorithm explanation shown in this paper was extracted from the explanatory publication [14].

The result of each iteration is one of two cases:

(1) A single new vertex – the accepted point – replaces the vertex, x_{n+1} in the set of vertices for the next iteration.

(2) A shrink is performed and a set of n new points is generated that, together with x_1, form the simplex at the next iteration.

The steps of a single iteration of the Nelder-Mead Algorithm are:

1. Order the n+1 vertices so that $f(x_1) \leq f(x_2) \leq \cdots \leq f(x_{n+1})$.
2. **Reflection.** Compute the reflection point x_r from
$$x_r = (1+\rho)\hat{x} - \rho x_{n+1}, \qquad (7)$$
where $\hat{x} = \sum_{i=1}^{n} x_i/n$ is the centroid of the n best points. Then evaluate $f_r=f(x_r)$. If $f_1 \leq f_r < f_n$, accept the point and terminate the iteration.
3. **Expansion.** If $f_r < f_n$, calculate the expansion point x_e,
$$x_e = \hat{x} + \chi(x_r - \hat{x}) \qquad (8)$$
And evaluate $f(x_e)$. If $f_e < f_r$, accept x_e and terminate the iteration. Otherwise, accept x_r and terminate the iteration. But if $f_r \geq f_n$, move to step 4.
4. **Contraction.** Perform either and outside or inside contraction.
If $f_n \leq f_r < f_{n+1}$, perform an outside contraction. Calculate:
$$x_c = \gamma(x_r - \hat{x}) + \hat{x}. \qquad (9)$$
Evaluate $f(x_c)$. If $f(x_c) < f_r$, accept x_c and terminate the iteration. Otherwise, go to step 5.
If $f_r \geq f_{n+1}$, perform and inside contraction. Calculate:
$$x_{cc} = \gamma(x_{n+1} - \hat{x}) + \hat{x} \qquad (10)$$
Evaluate $f(x_{cc})$. If $f(x_{cc}) < f_{n+1}$, accept x_{cc} and terminate the iteration. Otherwise, go to step five.
5. **Shrink.** Evaluate f at the n points $v_i = x_1 + \alpha(x_i - x_1)$, $i = 2,\ldots, n+1$. The vertices of the simplex at the next iteration will consist of $x_1, v_2, \ldots, v_{n+1}$. Terminate the iteration.

Typically this process only needs to be repeated 5-10 times for it to provide a very large improvement over the starting point. Figures 1 and 2 depict the effects of each step in two dimensions, where the simplex is a triangle. Both figures assume the values for the simplex parameters to be equal to those given in equation (6).

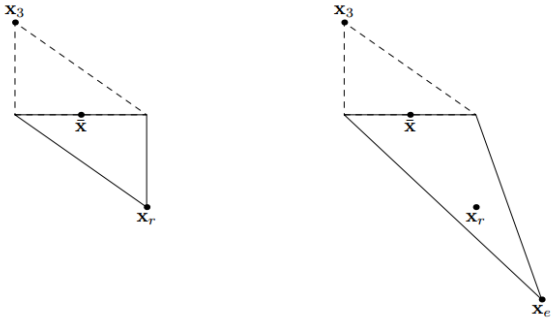

Fig. 1. Nelder-Mead simplices after a reflection (left) and an expansion (right). The original simplex is shown with a dashed line.

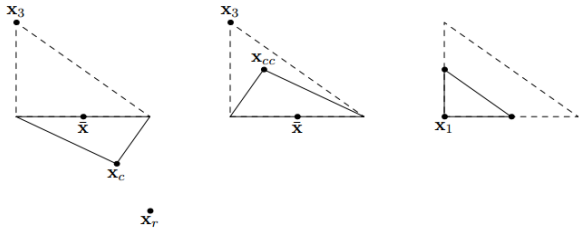

Fig. 2. Nelder-Mead simplices after an outside contraction (left), an inside contraction (middle), and a shrink (right). The original simplex is shown with a dashed line.

III. I-ELM AND NME-ELM

A. I-ELM algorithm

The I-ELM algorithm was originally published in [7], and has spawned a family of algorithms known as Extreme Learning Machines that includes I-ELM, CI-ELM, and EI-ELM. Both CI-ELM and EI-ELM are algorithms that offer improvements to the I-ELM algorithm. For this reason, the I-ELM algorithm will be covered, and then the proposed improvements will be presented. For this section, we will go back to the indices and variables presented in Section II, A. The algorithm shown below was extracted from [7].

Given a training set $\{(x_p, y_p)|x_p \in R^D, y_p \in R, p = [1 \ldots P]\}$, an activation function $g(x)$, a maximum node number N, and an expected learning accuracy ε:

As described before the activation function in equation (2) will be used.

1. **Initialize:** Let the number of nodes, $n = 0$ and residual error, $E = y$.
2. **Learning:**
While ($n < N$) and ($RMSE > \varepsilon$)
   a. Increase the number of hidden nodes by 1.
   b. Assign random center c_n and a width σ_n within an acceptable range for the new hidden node.
   c. Based on the random activation function and the error, calculate the output weight β_n for the node
   $$\beta_n = \frac{\sum_{p=1}^{P} e_p g_n(x_p)}{\sum_{p=1}^{P} g_n(x_p)^2} \qquad (11)$$
   d. Calculate the residual error after adding the new hidden node:
   $$E = E - \beta_n * g_n(x) \qquad (12)$$
   Where x and E are the vectors containing all of the input patterns and errors for each pattern respectively.
End while loop
3. Output is calculated using equation (1).

This algorithm will give a network consisting of a single hidden layer of RBF nodes connected with weights to a

summing output node. It was proven in [7] that this network will work as a universal approximator. This algorithm allows for a very fast training time as there is only one calculation to make per iteration. Most environments allow the calculation of β in matrix form to be very fast.

B. The NME-ELM algorithm

The goal of the NME-ELM algorithm is to provide similarly fast training times and errors, but to also provide a more compact network with better generalization properties. This is done by adjusting the I-ELM algorithm so that it now runs like the following:

Given the same input data and activation functions that were used before:

1. **Initialize:** Let the number of nodes, $n = 0$ and the error, $E = y$.
2. **Learning:**
   While ($n < N$) and ($RMSE > \varepsilon$)
   a. Increase the number of hidden nodes by 1.
   b. Find the index, j, of the maximum error, e_{pmax}.
   c. Assign the center c_n, of the new node to be the input pattern x_j.
   d. Assume the initial value of β_n to be equal to e_{pmax}.
   e. Initialize the Nelder-Mead Simplex [11] algorithm:
      (1) Set the Simplex parameters according to Equation (6).
      (2) Choose some initial values for σ_n.
      (3) Calculate:
      $$SSE(\sigma_{i,n}) = \sum_{p=1}^{P}(e_p - \beta_n * gn(x_p))^2$$
      For each $\sigma_{i,n}$. These *SSE* and $\sigma_{i,n}$ values will be the initial f values and x values of the Simplex respectively.
   f. Perform k iterations of the Simplex algorithm (k is typically 5-10).
   g. Re-calculate β_n using equation (11).
   h. Calculate the residual error, E, using equation (12).
   End While Loop.
3. Output is calculated using equation (1).

The proposed algorithm still generates a SLFN that acts as a universal approximator. Instead of a random center and width however, the algorithm chooses a center designed to eliminate the largest point in the error surface and a radius optimized based on this center selection. In this way, it is expected that a more compact network can be generated.

Let us examine the step by step construction process of the NME-ELM algorithm on a simple problem. The problem chosen will be one period of a simple sine wave. The desired function is shown in Figure 3.

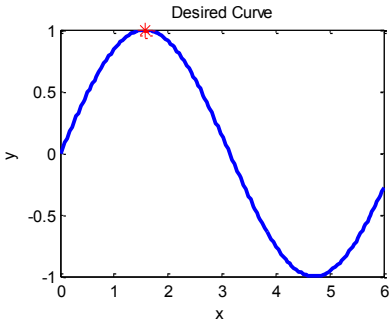

Fig. 3 Desired sinusoidal function with the first center c_1 shown by the red asterisk.

Once the inputs and desired output are fed into the algorithm, a center and an initial weight for the first RBF node are chosen. In this case, the center $c_1 = 1.5678$, and is depicted in Figure 3 as the red asterisk. The initial weight $\beta_1 = 1.000$. This is from the output of the sine function at input equal to c_1. Then the width σ_n is optimized using the Nelder-Mead algorithm. This eventually gives $\sigma_1 = 1.1429$. Now β_1 is re-calculated using equation (11). This gives $\beta_1 = 1.0714$. The output of the first node is shown in Figure 4.

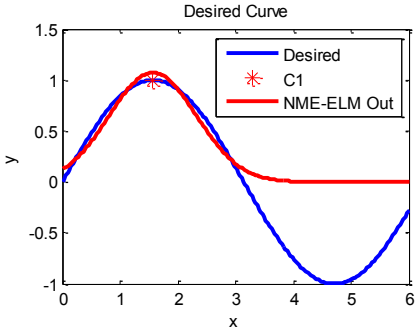

Fig. 4. The output of one node using NME-ELM.

Now the new error is calculated using equation (12) and this error is used as the desired surface for the addition of a second RBF node. Figure 5 shows the error surface (new desired curve) and the initially chosen center c_2.

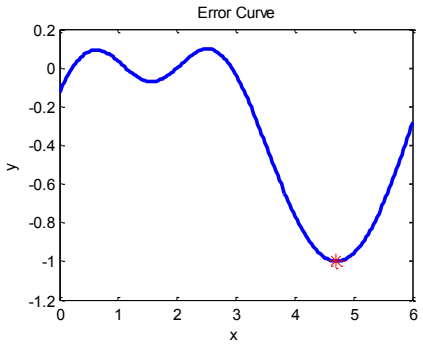

Fig. 5. The desired curve for the second node and the center c_2.

Following the same process as before, the parameters for the second RBF unit are selected as:
$c_2 = 4.7035$, $\sigma_2 = 0.6797$, and $\beta_2 = 1.0260$. Figure 6 shows the resulting output for two RBF units and the original desired surface. With only two RBF units, the NME-ELM algorithm reaches a *RMSE* value of 0.0627. If the algorithm is allowed to continue to a total of five RBF units, the *RMSE* value is as low as 0.0266.

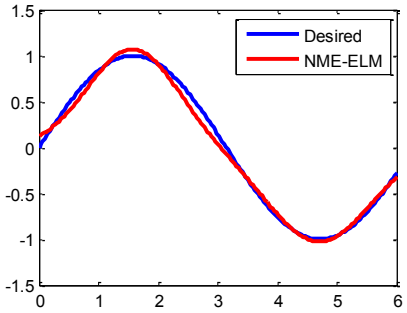

Fig. 6. Desired surface and NME-ELM output after 2 RBF units have been added to the network. RMSE = 0.0627.

IV. Experimental Results

A. *Highly Nonlinear Benchmark: Peaks Problem*

The peaks problem is a benchmark problem used to test how well the algorithm handles nonlinear data sets. This problem can be described by the equation:

$$z(x,y) = -\frac{1}{30}\exp\left(-1-6x-9x^2-9y^2\right)+ \\ -\left(0.6x-27x^3-243y^5\right)\exp\left(-9x^2-9y^2\right)+ \\ +\left(0.3-1.8x+2.7x^2\right)\exp\left(-1-6y-9x^2-9y^2\right) \quad (13)$$

It is used because it is easy to visualize the algorithm's progress as it trains to the peaks problem. This way it is possible to make sure that the algorithm is behaving as expected before attempting to solve some real world problems. Figures 7 and 8 show the desired surface of the peaks problem and the training results of several algorithms on the peaks problem respectively.

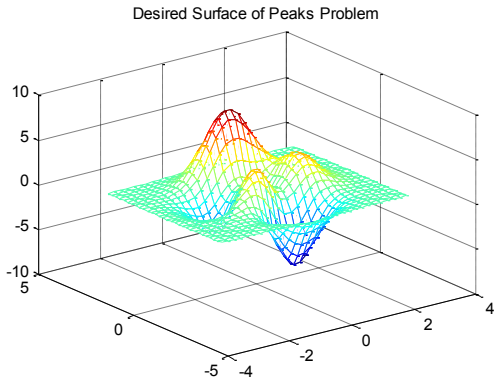

Fig 7. Desired surface for the peaks problem.

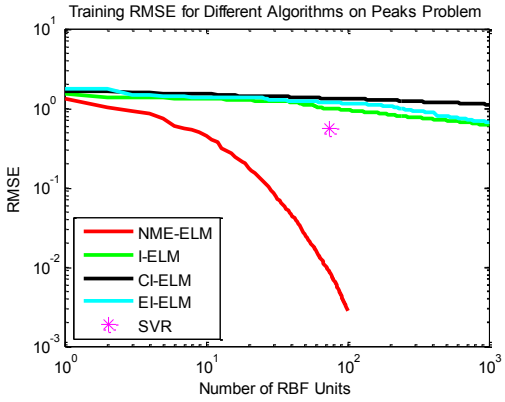

Fig 8. Training results for different algorithms on the peaks problem.

From Figure 8 it can be seen that the NME-ELM algorithm converges very quickly to nearly zero error for this problem.

B. *Real World Data*

In order to prove the robustness of the proposed algorithm, it is tested on real world data from the widely used UCI Repository of Machine Learning Databases [15]. The results obtained are then compared to the results from the ELM family of algorithms and SVR. The testing environment used consists of Windows 7 Professional 64-bit operating system, an Intel Core i7-2600 CPU @ 3.4 GHz processor, and 8GB RAM. MATLAB was used for the ELM algorithms and LIBSVM for SVR [16].

For each real world data set that was tested, 20 runs were performed with a training set consisting of 70% of the data set and a testing set consisting of 30% of the data set. For each run, the training and test sets were randomly generated out of the original data set. Then the runs were averaged and the standard deviation was calculated. All input sets were normalized to [-1, 1] and all output sets were normalized to [0, 1].

Figure 9 shows the testing results for the NME-ELM algorithm on several real world datasets. It can be seen that

the testing *RMSE* converges within 50 RBF units for each problem. This *RMSE* is comparable to the *RMSE* obtained by the ELM algorithms, but in this case only 50 RBF nodes are used.

The data presented in Table I shows that the NME-ELM algorithm performs very well in terms of testing *RMSE* and network size. The major shortcoming that becomes apparent by examining Table II is that the training time is much longer for NME-ELM than the ELM family and SVR. This is probably due to the fact that the Nelder-Mead algorithm requires the function output to be calculated many times per iteration.

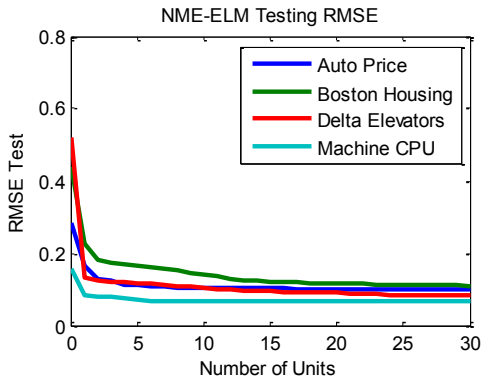

Fig. 9. Testing *RMSE* of NME-ELM on several real world datasets.

V. Conclusion

This paper presents a new algorithm that is designed to expand on the work done by the ELM family of algorithms. These algorithms generate a SLFN architecture using RBF activation functions for universal approximation. Previous research [1]-[4] and [7] proves that RBF networks in a SLFN architecture can be used as universal approximators. Furthermore, [7] proves that these networks can be constructed using random hidden nodes. This research is what created the ELM algorithms which show good generalization properties and fast training times.

The intent of the algorithm presented in this paper (NME-ELM) is to achieve similar properties while creating a more compact network.

These goals were realized by modifying the I-ELM algorithm in two ways:

(1) Fix the center of each newly added node at the largest magnitude points of error so that this error is eliminated quickly.

(2) Optimize the width of added RBF units to further minimize error. For NME-ELM, this is done by using Nelder-Mead's Simplex algorithm to choose an appropriate width for each newly added RBF unit.

Experimental results, shown in section IV, show that the proposed algorithm achieves similar testing errors with fewer nodes. The training time of the NME-ELM algorithm is considerably higher than the algorithms to which it was compared. This is undoubtedly because of the way the Nelder-Mead Simplex algorithm must calculate the error values many times during each iteration.

TABLE I

NME-ELM, EI-ELM, CI-ELM, AND SVR TRAINING TIME, TESTING RMSE, AND NETWORK SIZES ON UCI DATA SETS

	NME-ELM			EI-ELM (RBF, k=10)			CI-ELM (RBF)			SVR (RBF Kernel)		
Problems	Mean Time(s)	Mean RMSE	# Nodes	Mean Time(s)	Mean RMSE	# Nodes	Mean Time(s)	Mean RMSE	# Nodes	Mean Time(s)	Mean RMSE	# SVs
Abalone	15.378	0.1208	21	5.9603	0.0927	200	0.6098	0.0845	200	2.9060	0.0771	1121
Auto Price	0.4331	0.0871	44	0.2571	0.1393	200	0.0299	0.1197	200	0.0912	0.0930	62
Boston Housing	1.3915	0.1120	50	0.7742	0.1622	200	0.0831	0.1682	200	0.0975	0.0925	145
California Housing	190.00	0.2141	18	29.920	0.1743	200	3.6035	0.1756	200	109.16	0.1127	7038
Delta Ailerons	33.686	0.0519	25	9.9601	0.0610	200	1.0091	0.0416	200	6.1117	0.0418	807
Delta Elevators	56.4552	0.0783	50	13.407	0.0814	200	1.5035	0.0566	200	3.3442	0.0530	2016
Machine CPU	0.5403	0.0745	24	0.3078	0.0777	200	0.0353	0.0675	200	0.0907	0.0539	31

References

[1] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[2] M. Leshno and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Networks*, vol. 6, pp. 861–867, 1993.

[3] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Comput.*, vol. 3, no. 2, pp. 246–257, Jun. 1991.

[4] H. Yu, T. Xie, S. Paszczynski, and B. M. Wilamowski, "Advantages of Radial Basis Function Networks for Dynamic System Design," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 12, pp. 5438–5450, Dec. 2011.

[5] V. N. Vapnik, *Statistical Learning Theory*, 1st ed. Wiley-Interscience, 1998.

[6] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[7] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879 – 892, Jul. 2006.

[8] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16–18, pp. 3056–3062, Oct. 2007.

[9] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, no. 16–18, pp. 3460–3468, Oct. 2008.

[10] B. M. Wilamowski, "Neural network architectures and learning algorithms," *IEEE Industrial Electronics Magazine*, vol. 3, no. 4, pp. 56 –63, Dec. 2009.

[11] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, Jan. 1965.

[12] E. Boyd, K. W. Kennedy, R. A. Tapia, V. J. Torczon, and V. J. Torczon, "Multi-Directional Search: A Direct Search Algorithm for Parallel Machines," Rice University, 1989.

[13] N. Pham and B. M. Wilamowski, "Improved Nedler Mead's Simplex Method and Applications," *Journal of Computing*, vol. 3, no. 3, pp. 55–63, Mar. 2011.

[14] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions," *SIAM Journal of Optimization*, vol. 9, pp. 112–147, 1998.

[15] A. Asuncion and A. Frank, "UCI Machine Learning Repository." University of California, Irvine, School of Information and Computer Sciences, 2010.

[16] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines." ACM Transactions on Intelligent Systems and Technology, 2011.