A MACHINE LEARNING APPROACH FOR MODELING AND ITS APPLICATIONS

Dr Shuxiang Xu[(a)], Dr Yunling Liu[(b)], Dr Byeong-Ho Kang[(c)], Dr Wanlin Gao[(d)]

[(a)]School of Computing and Information Systems, University of Tasmania, Launceston, Tasmania 7250, Australia
[(b)](Corresponding Author) College of Information and Electrical Engineering, China Agricultural University, Beijing, China
[(c)]School of Computing and Information Systems, University of Tasmania, Hobart, Tasmania 7001, Australia
[(d)]College of Information and Electrical Engineering, China Agricultural University, Beijing, China

[(a)]Shuxiang.Xu@utas.edu.au, [(b)]lyunling@163.com, [(c)]Byeong.Kang@utas.edu.au, [(d)]gaowlin@cau.edu.cn

ABSTRACT
This paper proposes a new learning algorithm for Higher Order Neural Networks for the purpose of modelling and applies it in three benchmark problems. Higher Order Neural Networks (HONNs) are Artificial Neural Networks (ANNs) in which the net input to a computational neuron is a weighted sum of its inputs and products of its inputs (rather than just a weighted sum of its inputs as in traditional ANNs). It was well known that HONNs can implement invariant pattern recognition. The new learning algorithm proposed is an Extreme Learning Machine (ELM) algorithm. ELM randomly chooses hidden neurons and analytically determines the output weights. With ELM algorithm only the connection weights between hidden layer and output layer are adjusted. This paper proposes an ELM algorithm for HONN models and applies it in an image processing problem, a medical problem, and an energy efficiency problem. The experimental results demonstrate the advantages of HONN models with the ELM algorithm in such aspects as significantly faster learning and improved generalization abilities (in comparison with standard HONN and traditional ANN models).

KEYWORDS: Artificial Neural Network, Extreme Learning Machine, Feedforward Neural Network, Higher Order Neural Network, Machine Learning.

1. INTRODUCTION

An actively researched machine learning algorithm, Artificial Neural Networks (ANNs) have been widely used as powerful information processing tools for modeling a diverse range of applications. HONNs (Higher Order Neural Networks) (Lee et al, 1986) are networks in which the net input to a computational neuron is a weighted sum of its inputs and products of its inputs (see Figure 1.1 for an example of a second order HONN). Such neuron is called a Higher-order Processing Unit (HPU) (Lippman, 1989). It was known that HONN's can implement invariant pattern recognition (Psaltis et al, 1988 ; Reid et al, 1989 ; Wood et al, 1996). In (Giles et al, 1987) it was shown that HONN's have impressive computational, storage and learning capabilities. In (Redding et al, 1993), HONN's were proved to be at least as powerful as any other FNN (Feedforward Neural Network) architecture when the orders of the networks are the same. Kosmatopoulos et al (1995) studied the approximation and learning properties of one class of recurrent HONNs and applied these architectures to the identification of dynamical systems. Thimm et al (1997) proposed a suitable initialization method for HONN's and compared this method to weight initialization techniques for FNNs. More recently, In Alanis et al (2007) an application of HONN was proposed to successfully solve the tracking problem for a class of nonlinear systems in discrete time using backstepping technique. HONNs were employed in Xu (2010a, 2010b) for several data mining tasks and achieved significant results which outperformed conventional ANNs. A HONN was investigated in Dunis (2011) for forecasting and trading EUR/USA exchange rates, with outstanding results when compared against other ANN architectures as well as traditional statistical approaches. In Fallahnezhad et al (2011) a hybrid HONN model was developed for handling several benchmark classification problems, resulted in significant improvements of accuracy (in comparison with the best accuracy obtained from other methods).

Unlike traditional ANN learning algorithms (such as back-propagation), Extreme Learning Machine (ELM) algorithm randomly chooses hidden neurons and analytically determines the output weights (Huang et al 2005, 2006, 2008). With ELM algorithm, only the connection weights between hidden layer and output layer are adjusted. Many types of hidden nodes including additive nodes, RBF (radial basis function) nodes, multiplicative nodes, and other non-neural alike nodes can be used as long as they are piecewise nonlinear. ELM algorithm tends to generalize better at very fast learning

speed: it can learn thousands of times faster than conventional popular learning algorithms (Huang et al 2006).

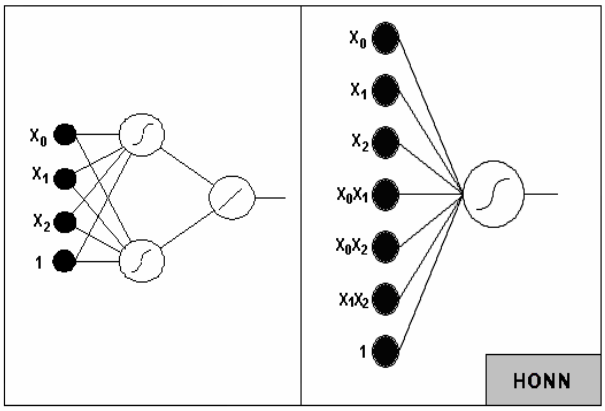

Figure 1.1, Left, FNN with three inputs and two hidden nodes; Right, second order HONN with three inputs

This paper develops an ELM algorithm for HONN models and applies it in an image processing problem, a medical problem, and an energy efficiency problem. Following the introduction, Section 2 introduces the ELM algorithm for HONN. Section 3 describes three HONN modelling experiments to demonstrate the advantages of ELM HONN (against standard HONN and traditional ANNs such as MLP and RBF networks). Results are given and discussed. Section 4 summarises this report.

2. HONN MODELS WITH ELM ALGORITHM

Based on a modified version of a two-dimensional (second order) HONN defined in Zhang et al (2002), this paper proposes the following ELM algorithm for HONN. The main idea of ELM lies in the random selection of hidden neuron activation functions (must be infinitely differentiable) with random initialization of the SFNN (single-hidden-layer feedforward neural network) weights and biases. Then, the input weights and biases do not need to be adjusted during training, only the output weights are learned. In this work, an adaptive neuron activation function (infinitely differentiable) has been used. The free parameters in the adaptive activation function are adjusted in a way similar to how the output weights are tuned.

Consider a set of s distinct training samples (x_i, y_i) with $x_i \in R^n$ and $y_i \in R^m$, where $i = 1, 2, \ldots, s$, n and m are positive integers, where n represents the dimension of an input space, and m the dimension of an output space. Then an SFNN with N hidden neurons can be mathematically represented by

$$\sum_{i=1}^{N} o_i f(W_i \cdot X + b_i) \qquad (2.1)$$

with f being the randomly selected neuron activation function, W_i the input to hidden layer weight vector, b_i the biases, o_i the output weights, and X the input vector: $X = [x_1 \ x_2 \ \ldots \ x_s]^T$.

In this work, the following adaptive neuron activation function is used:

$$f(x) = A1 \cdot e^{-B1 \cdot x^2} + \frac{A2}{1 + e^{-B2 \cdot x}} \qquad (2.2)$$

where $A1$, $A2$, $B1$, and $B2$ are real variables which will be adjusted during training (in the same way as connection weights, see the end of the current section)). A justification of the use of free parameters in a neuron activation function can be found in Zhang et al (2002).

In case of two-dimensional (second order) HONN with a single hidden layer, equation (2.1) becomes

$$\sum_{i=1}^{N} o_i f(W_i \begin{bmatrix} X \\ H(X) \end{bmatrix} + b_i) \qquad (2.3)$$

where

$$H(X) = [x_1 x_2 \ x_1 x_3 \ \cdots \ x_{s-1} x_s]^T \qquad (2.4)$$

Equations (2.3) and (2.4) show that for a two-dimensional HONN, the number of input neurons is defined by

$$n + C_n^2 = n + \frac{n(n-1)}{2} \qquad (2.5)$$

Assume that the single layer HONN approximates the training samples perfectly, then the errors between the estimated outputs and the actual outputs are zero, which means

$$\sum_{i=1}^{N} o_i f(W_i \begin{bmatrix} X \\ H(X) \end{bmatrix} + b_i) = Y \qquad (2.6)$$

where Y is the output vector: $Y = [y_1 \ y_2 \ \ldots \ y_s]^T$.

Equation (2.6) can be rewritten as $H \cdot O = Y$, with

$$H = \begin{bmatrix} f(w_1 x_1 + b_1) & \cdots & f(w_N x_1 + b_N) \\ \vdots & \vdots & \vdots \\ f(w_1 x_s + b_1) & \ddots & f(w_N x_s + b_N) \\ f(w_1 x_1 x_2 + b_1) & & f(w_N x_1 x_2 + b_N) \\ \vdots & \vdots & \vdots \\ f(w_1 x_{s-1} x_s + b_1) & \ldots & f(w_N x_{s-1} x_s + b_N) \end{bmatrix} \qquad (2.7)$$

and O as the hidden layer to output layer weight vector: $O = [o_1 \ o_2 \ \ldots \ o_N]^T$.

Then the idea of ELM algorithm, when applies to a HONN, states that with randomly initialized input weights and biases, and with the condition that the randomly selected neuron activation function is infinitely

differentiable, the output weights can be determined so that the single layer HONN provides an approximation of the sample values to any degree of accuracy. The way to calculate the output weights from the hidden layer output matrix and the target values is proposed with the use of a Moore-Penrose generalized inverse (Rao et al 1972) of the matrix (2.7), denoted as H^{-1}.

Overall, the improved ELM algorithm for HONNs is proposed as follows.

Given a set of *s* distinct training samples (x_i, y_i) with $x_i \in R^n$ and $y_i \in R^m$, an neuron activation function $f: R \to R$ which is infinitely differentiable, and the number of hidden layer neurons *N*:

Step 1. Randomly assign hidden layer parameters (input weights and biases);

Step 2. Calculate the hidden layer to output layer weight matrix *H*;

Step 3. Calculate the output weights matrix $O = H^{-1}Y$.

3. HONN MODELLING EXAMPLES

In this section, the ELM HONN modelling has been used for an image processing problem, a medical problem, and an energy efficiency problem. The algorithm has been implemented based on a HONN implementation in Matlab version R2011a, run on a standard Windows 7 operating system with a 4-core CPU speed of 2.70GHz and a RAM of 8GB.

To discover the advantages and disadvantages of the HONN model (with the Sigmoid activation function and one hidden layer), the following ANNs have also been applied onto the datasets of the problems for comparison studies: a conventional standard HONN model (with the Sigmoid activation function and one hidden layer); a Multi-Layer Perceptron (MLP) (with the Sigmoid activation function and one hidden layer); An RBF Neural Network (with the Gaussian activation function and one hidden layer). The standard MLP and RBF algorithms offered within the Neural Network Toolbox of Matlab version R2011b are used to train these traditional ANNs. The learning algorithm for the conventional HONN model is from Zhang et al (2002). For all of these ANNs the number of hidden layer neurons has been determined using an approach from Xu et al (2008).

3.1 Skin Segmentation

One of the significant topics in human face image recognition is to automatically determine skin and non-skin areas of a face image. A skin and non-skin dataset has been generated using skin textures which come from face images of people of different ages, genders, and races (Bache et al 2013). The dataset has been collected by sampling RGB (Red, Green, Blue) values of the face images. There are 4 attributes in the dataset with three of them representing the input attributes (the RGB values) and the last one representing the class attribute (skin or non-skin). There is a total of 245057 instances, however, for the purpose of this modelling only half of them have been used for faster learning and testing.

For this experiment, the dataset is divided into a training/learning set made of 80% of the original set and a test set made of 10% of the original set. The final 10% is used for evaluating the model's generalisation abilities.

For ELM HONN and Standard HONN (second order), the number of input neurons is calculated as follows:

$$3 + C_3^2 = 3 + \frac{3(3-1)}{2} = 6$$

For MLP and RBF networks, the number of input neurons is 3 (each for an input attribute).

The experimental results are displayed in Table 3.1. It can be seen that the ELM HONN is considerably faster than standard HONN, and in this case it produces an accuracy which is higher by 7.1%. For standard HONN model, training usually takes longer time because of the increased number of input neurons (compared with conventional MLP and RBF neural networks): in this experiment, the number of input layer neurons is 6 for the HONNs while 3 for the MLP and RBF network. However, ELM HONN is significantly faster. We can also see that the correctness rates (accuracy) produced by the conventional ANNs (MLP and RBF) are lower.

Table 3.1. Comparing ELM HONN against standard HONN, MLP, RBF neural networks. HL: Hidden Layer, TT: Training Time

ANN	Dataset	HL nodes #	TT (secs)	Correctness or accuracy
ELM HONN	Skin	9	10.8	84.2%
Standard HONN	Skin	9	38.7	77.1%
MLP	Skin	22	19.3	71.4%
RBF	Skin	22	17.4	73.5%

3.2 Freezing of Gait Problem

Freezing of Gait (FoG, inability to step) is a typical problem suffered by people with Parkinson's disease. A Daphnet Freezing of Gait dataset has been recorded to recognize gait freeze from wearable acceleration sensors placed on legs and hip of the patients who have volunteered to participate in the study (Bache et al 2013). There is a total of 237 instances, with 9 input attributes and 3 class attributes in the dataset. The input attributes are

- Ankle (shank) acceleration - horizontal forward acceleration
- Ankle (shank) acceleration - vertical
- Ankle (shank) acceleration - horizontal lateral
- Upper leg (thigh) acceleration - horizontal forward acceleration

- Upper leg (thigh) acceleration - vertical
- Upper leg (thigh) acceleration - horizontal lateral
- Trunk acceleration - horizontal forward acceleration
- Trunk acceleration - vertical
- Trunk acceleration - horizontal lateral

The 3 class attributes are:

- 0 (irrelevant movement)
- 1: no freeze (can be any of stand, walk, turn)
- 2: freeze

The challenge is to model this problem to determine what movements would cause a gait freeze (for patients with Parkinson's disease).

For this experiment, the dataset is divided into a training/learning set made of 75% of the original set and a test set made of 15% of the original set. The final 10% is used for evaluating the model's generalisation abilities.

For ELM HONN and Standard HONN (second order), the number of input neurons is calculated as follows:

$$9 + C_9^2 = 9 + \frac{9(9-1)}{2} = 45$$

For MLP and RBF networks, the number of input neurons is 9 (each for an input attribute).

The experimental results are displayed in Table 3.2. It appears that for this experiment the ELM HONN produces similar accuracy as standard HONN, however, ELM HONN is significantly faster than standard HONN (as well as MLP and RBF networks). Additionally, both HONN modes produce higher accuracy than MLP and RBF neural networks.

Table 3.2. Comparing ELM HONN against standard HONN, MLP, RBF neural networks. HL: Hidden Layer, TT: Training Time

ANN	Dataset	HL nodes #	TT (secs)	Correctness or accuracy
ELM HONN	FoG	11	4.9	81.5%
Standard HONN	FoG	11	21.4	80.3%
MLP	FoG	25	11.4	70.2%
RBF	FoG	25	10.4	68.1%

3.3 Modelling Energy Efficiency

The third experiment deals with modelling energy efficiency of buildings: assessing the heating load and cooling load requirements of buildings based on 8 building parameters (factors) (Bache et al 2013). The buildings differ from each other with respect to the glazing area, the glazing area distribution, and the orientation, and others. There is a total of 768 instances in the dataset, with 8 input attributes (each representing a building feature). The 8 features are

- Relative Compactness
- Surface Area
- Wall Area
- Roof Area
- Overall Height
- Orientation
- Glazing Area
- Glazing Area Distribution

There are 2 class attributes:

- Heating Load
- Cooling Load

The challenge is to learn the relationships between the 8 features and its heating load and cooling load.

For this experiment, the dataset is divided into a training/learning set made of 70% of the original set and a test set made of 15% of the original set. The final 15% is used for evaluating the model's generalisation abilities.

For ELM HONN and Standard HONN (second order), the number of input neurons is calculated as follows:

$$8 + C_8^2 = 8 + \frac{8(8-1)}{2} = 36$$

For MLP and RBF networks, the number of input neurons is 8 (each for an input attribute).

The experimental results are displayed in Table 3.3. Unsurprisingly the ELM HONN is considerably faster than standard HONN (as well as MLP and RBF), and in this experiment it produces an accuracy which is higher by 10.2% than standard HONN. We can also see that the simulation accuracies produced by the HONN models are significantly higher than the traditional ANN models (MLP and RBF).

Table 3.3. Comparing ELM HONN against standard HONN, MLP, RBF neural networks. HL: Hidden Layer, TT: Training Time

ANN	Dataset	HL nodes #	TT (secs)	Correctness or accuracy
ELM HONN	Energy	13	7.9	87.6%
Standard HONN	Energy	13	25.5	77.4%
MLP	Energy	33	15.3	68.3%
RBF	Energy	33	16.4	70.1%

4. CONCLUSIONS

This paper proposes an ELM algorithm for HONN models and applies it in an image processing problem, a medical problem, and an energy efficiency problem. An obvious outcome is that HONN model with ELM algorithm is significantly faster than standard HONN model as well as traditional ANNs such as MLP and RBF, due to the nature of ELM. It appears that, generally speaking, ELM HONN produces higher accuracy than standard HONN, as demonstrated in the first and third experiments, although occasionally (as in the second experiment) both models produce similar correctness rates. It can be seen that HONN models produce higher accuracies than MLP and

RBF neural networks. This paper is a report of work in progress. In the future, more experiments involving larger datasets will be conducted to further test the new ELM algorithm for HONN. Another direction for future research would be the use of an ensemble of HONN models for modelling and simulation.

ACKNOWLEDGMENTS

This project is supported by Chinese Universities Scientific Fund, Project number: 2013QJ051, and National Key Technology R&D Program of China during the 12th Five-Year Plan Period (Project numbers: 2012BAJ18B07, 2012BAD35B02).

REFERENCES

Alanis AY, Sanchez EN, Loukianov AG, Discrete-Time Adaptive Backstepping Nonlinear Control via High-Order Neural Networks, IEEE TRANSACTIONS ON NEURAL NETWORKS, 18 (4) (2007) 1185-1195.

Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Dunis, CL, Laws J, and Sermpinis G, Higher order and recurrent neural architectures for trading the EUR/USD exchange rate, Quantitative Finance, 11 (4) (2011) 615–629.

Fallahnezhad M, Moradi, MH, Zaferanlouei S, A Hybrid Higher Order Neural Classifier for handling classification problems, Expert Systems with Applications, 38 (2011) 386–393.

Giles, C.L., Maxwell, T. (1987) Learning, invariance, and generalization in higher order neural networks, *Applied Optics*, 26(23), 4972-4978.

Huang, GB, Siew, CK, "Extreme Learning Machine with Randomly Assigned RBF Kernels," International Journal of Information Technology, vol. 11, no. 1, pp. 16—24, 2005.

Huang, GB, Zhu, QY, Siew, CK, "Extreme Learning Machine: Theory and Applications", Neurocomputing, vol. 70, pp. 489-501, 2006.

Huang, GB, Li, MB, Chen, L, and Siew, CK, "Incremental Extreme Learning Machine With Fully Complex Hidden Nodes," Neurocomputing, vol. 71, pp. 576-583, 2008.

Kosmatopoulos, E.B., Polycarpou, M.M., Christodoulou, M.A., Ioannou, P.A. (1995). High-order neural network structures for identification of dynamical systems, *IEEE Transactions on Neural Networks*, 6(2), 422-431.

Lee, Y.C., Doolen, G., Chen, H., Sun, G., Maxwell, T., Lee, H., Giles, C.L. (1986) Machine learning using a higher order correlation network, *Physica D: Nonlinear Phenomena*, 22, 276-306.

Lippman, R.P. (1989) Pattern classification using neural networks, *IEEE Commun. Mag*., 27, 47-64.

Psaltis, D., Park, C.H., Hong, J. (1988) Higher order associative memories and their optical implementations, *Neural Networks*, 1, 149-163.

Rao C R, Mitra S K, (1972). Generalized Inverse of Matrices and Its Applications, New York: Wiley

Redding, N., Kowalczyk A. and Downs, T., (1993). "Constructive high-order network algorithm that is polynomial time", *Neural Networks*, Vol.6, pp.997-1010.

Reid, M.B., Spirkovska, L., Ochoa, E. (1989). Simultaneous position, scale, rotation invariant pattern classification using third-order neural networks, *Int. J. Neural Networks*, 1, 154-159.

Thimm, G., Fiesler, E. (1997). High-order and multilayer perceptron initialization, *IEEE Transactions on Neural Networks*, 8(2), 349-359.

Wood, J., Shawe-Taylor, J. (1996). A unifying framework for invariant pattern recognition, *Pattern Recognition Letters*, 17, 1415-1422.

Xu S (2010a), Adaptive Higher Order Neural Network Models for Data Mining, Artificial Higher Order Neural Networks for Computer Science and Engineering: Trends for Emerging Applications, Information Science Reference, Ming Zhang (ed), Hershey, United States, 86-98. ISBN 978-1-61520-711-4.

Xu S (2010b), Data Mining Using Higher Order Neural Network Models With Adaptive Neuron Activation Functions, International Journal of Advancements in Computing Technology, 2 (4) 168 - 177.

Xu, S and Chen, L (2008), A novel approach for determining the optimal number of hidden layer neurons for FNN's and its application in data mining, Proceedings The 5th International Conference on Information Technology and Applications, 23-26 June 2008, Carins, Qld, pp. 683-686.

Zhang, M., Xu, S., Fulcher, J., (2002). Neuron-Adaptive Higher Order Neural-Network Models for Automated Financial Data Modeling, *IEEE Transactions on Neural Networks*, Vol 13, No. 1.