

Table of Contents

Preface	xvii
Acknowledgment.....	xxv

Section 1

Artificial Higher Order Neural Networks for Computer Science

Chapter 1

Higher Order Neural Network Group-Based Adaptive Tolerance Trees	1
<i>Ming Zhang, Christopher Newport University, USA</i>	

Chapter 2

Higher Order Neural Networks for Symbolic, Sub-Symbolic and Chaotic Computations	37
<i>João Pedro Neto, Faculdade de Ciências, Portugal</i>	

Chapter 3

Evolutionary Algorithm Training of Higher Order Neural Networks	57
<i>M. G. Epitropakis, University of Patras, Greece</i>	
<i>V. P. Plagianakos, University of Patras, Greece</i>	
<i>M. N. Vrahatis, University of Patras, Greece</i>	

Chapter 4

Adaptive Higher Order Neural Network Models for Data Mining.....	86
<i>Shuxiang Xu, University of Tasmania, Australia</i>	

Chapter 5

Robust Adaptive Control Using Higher Order Neural Networks and Projection.....	99
<i>Wen Yu, Cinvestav-IPN, México</i>	

Chapter 6

On the Equivalence Between Ordinary Neural Networks and Higher Order Neural Networks	138
<i>Mohammed Sadiq Al-Rawi, University of Aveiro, Portugal</i>	
<i>Kamal R. Al-Rawi, Petra University, Jordan</i>	

Chapter 6

On the Equivalence Between Ordinary Neural Networks and Higher Order Neural Networks

Mohammed Sadiq Al-Rawi
University of Aveiro, Portugal

Kamal R. Al-Rawi
Petra University, Jordan

ABSTRACT

In this chapter, we study the equivalence between multilayer feedforward neural networks referred as Ordinary Neural Networks (ONNs) that contain only summation (Sigma) as activation units, and multilayer feedforward Higher order Neural Networks (HONNs) that contains Sigma and product (PI) activation units. Since the time they were introduced by Giles and Maxwell (1987), HONNs have been used in many supervised classification and function approximation. Up to the date of writing this chapter, the most cited HONN article by ISI Thomson Web of Knowledge is the work of Kosmatopoulos et al., (1995) by which they introduced a recurrent HONN modeling. A simple comparison with ONNs is usually performed in order to demonstrate the performance of some newly introduced HONN architecture. Is it true that HONNs outperform ONNs, how much do they differ? And how much do they commute? Does equivalence exists between a HONN and an ONN? Is it possible to convert a HONN to an equivalent ONN? And how neural network equivalence is defined? This chapter tries to answer most of these questions. Due to the existence of huge neural networks architectures in the literature, the authors of this work are concerned and think that equivalence studies are necessary to give abstract definitions and unified approaches which might help in better understanding of HONNs performance and their

DOI: 10.4018/978-1-61520-711-4.ch006

respective design. On contrary to most of the previous works where HONN weights are non-negative integers, HONNs are given in this chapter in a form such that weights are adjustable real-valued numbers. In doing that, HONNs might have more expressive power and there is an increase probability of having complex valued neuron outputs. To enable the use of the real-valued weights that may result in a complex valued neuron output we introduce normalization to the input data as well as a modification to neuron activation functions. Using simple mathematics and the proposed normalization to input data, we showed that HONNs are equivalent to ONNs. The converted equivalent ONN posses the features of HONN and they have exactly the same functionality and output. The proposed conversion of HONN to ONN would permit using the huge amount of optimization algorithms to speed up the convergence of HONN and/or finding better topology. Recurrent HONNs, cascaded correlation HONNs, or any other complicated HONN can be simply defined via their equivalent ONNs and then trained with backpropagation, scaled conjugate gradient, Lavenberg-Marquadt algorithm, brain damage algorithms (Duda et al., 2000), etc. Using the developed equivalency model, this chapter also gives an easy bottom-up approach to convert a HONN to its equivalent ONN. Results on XOR and function approximation problems showed that ONNs obtained from their corresponding HONNs converged well to a solution. Different optimization training algorithms have been tested equivalent ONNs having feedforward structure and/or cascade correlation where the later have shown outstanding function approximation results.

1. INTRODUCTION

In our daily life we face several classification problems that are considered nonlinear, i.e., one cannot separate two categories using simply a line for two dimensional patterns, a plane for three dimensional patterns, or a hyper plane as in multi-dimensional patterns. Inspired by the biological neuronal system, computational intelligent based classification systems were developed in the past few decades and are widely known as computational neural networks. These computational networks possess powerful nonlinear classification ability and they are also known in the literature with other proximate names, such as artificial neural networks, and statistical neural networks. In this work, we choose to call multi-layer feedforward neural networks as Ordinary Neural Networks (ONNs) in order to distinguish them from Higher Order Neural Networks (HONNs). The reason is that both HONNs and ONNs are artificial, computational, multi-layer feedforward neural networks. Nonetheless, other terminologies of ONNs might exist such as first order neural networks (Giles, et al., 1988), or multi-layer perceptrons (Minsky and Papert 1969),

In order to solve nonlinear classification problems, an ONN with one or more hidden layers can be employed. Determining the proper number of hidden layers and the number of units in each hidden layer is accomplished by trial and error, dynamic adaptive algorithms e.g. surgeon brain damage algorithms (Duda et al., 2000). Several studies have used HONNs rather than ONNs in order to obtain better performance (Thimm, 1998; Thimm & Fiesler, 1997; Spirkovska & Reid, 1993; Rovithakis et al., 2004). To what degree we can rely on these outperformance results? When we investigate the literature we see that ONNs have only Sigma (summation) activation units, for example, the output of a $d - 2 - 1$ ONN is given by:

$$z_k = f \left(w_{20} + w_{21} f \left(\sum_{i=0}^d w_i x_i \right) + w_{22} f \left(\sum_{i=0}^d w_i x_i \right) \right) \quad (1)$$

where w 's are the weights that connect different layers of ONN, x_i is the value of the i_{th} input taken from the input pattern $\mathbf{x} = [x_1, x_2, \dots, x_d]$. In contrast, a HONN must have at least one PI (product unit) as was shown by Giles & Maxwell, (1987), hence, the output of an up to the second order HONN is given by:

$$z_k = f \left(w_0 + \sum_{i=0}^d w_i x_i + \sum_{i_1=0}^d \sum_{i_2=0}^d w_{i_1, i_2} x_{i_1} x_{i_2} \right) \quad (2)$$

Thus, the major difference between HONNs and ONNs is the way the activation is calculated, i.e., only sigma units are used to construct ONNs, while Sigma and PI units or just PI units are used to construct HONNs. Does this matter? In computer architecture, a multiplication operation can be implemented via an algorithm implementing several addition operations (Knuth, 1997; Kulisch, 2002). In fact, multiplications are defined for the whole numbers in terms of repeated addition and even multiplications of real numbers could be defined by a systematic generalization of this basic idea. With this in mind, HONNs could be converted to a very complex, large size, constrained ONNs. The hypothetical large sized ONN that equiv a HONN might justify the power of a moderate size HONN. Nonetheless, it is unfair to compare the computational cost of some HONN to another ONN that has the same number of units and synaptic connections. More than that, it is also unfair to compare the expressive power of a HONN to an ONN when they have the same number of units and synaptic connections. The reason is that the computational architecture and computational complexity of a HONN is much higher than that of an ONN. To overcome this dilemma it is necessary to develop a mathematical model for converting a HONN to its equivalent ONN and further studies can be performed later to answer questions about the expressive power and the computational complexity of both architectures.

In this chapter, neural networks equivalence is defined as follows; "Two neural networks are equivalent if they satisfy the following; randomly initialized to the same initial weights, pass through the same error values per epoch, their corresponding weights are exactly the same at each corresponding iteration and at the end of training, give exactly the same output for the same input, but their synaptic connections, synaptic activation, and their topology might differ". Lower terms or more constrained terms for equivalence definition might be defined without any problem. To explore this neural equivalence we will show using simple mathematical analyses that it is possible to convert a HONN to a nearly similar size equivalent ONN. Experimental tests will be conducted on the converted ONNs to show their ability to converge and classify typical classification problems.

BACKGROUND

In their article, Giles and Maxwell (1987) stated that higher-order weights capture higher-order correlations in data. Their first HONN work was introduced to solve a challenging computer vision problem known as invariance. After that, several other HONN works emerged: Huguen and Hollon (1991) stated

that higher order networks have the advantage of ease training over multilayer perceptrons and showed better classification for Radar data than a Gaussian classifier. A HONN with a prior knowledge of the binary training patterns reduces computation time and memory when applied to 100x100 Pixels images (Artyomov & Yadid-Pecht, 2005). Rovithakis et al., (2004) presented an algorithm to determine the structure of HONNs applied to function approximation. Clark, (1995) employed HONNs for tracking, code recognition and memory management. Rovithakis, (1999) employed HONNs in control. Foltyniewicz, (1995) developed a PI-Sigma-PI network structure for effective recognition of human faces in gray scale irrespective to their position, orientation and scale, and he claimed that it had small number of adjustable weights, rapid learning convergence, and excellent generalization properties. Abdelbar, (1998) showed that Sigma-PI HONNs perform better than ONNs in classification age-groups of abalone shellfish bench-mark. Kosmatopoulos et al., (1995) showed that if enough higher order connections are allowed to recurrent HONN then it is capable of approximating arbitrary dynamical systems. Their explanation is that the dynamic components are distributed throughout the network in the form of dynamic neurons. A HONN that learns the recognition of affine transformed images without any prior knowledge of the imaging geometry was proposed in (Al-Rawi, 2006). As most neural networks software is devoted for ONNs, Al-Rawi & Bisher, (2005) developed a software tool that can be used interactively to build HONNs. Still, the software cannot cope with the many optimization algorithms available for ONNs and only gradient descent with momentum had been considered. Therefore, if any HONN is equivalent to an ONN, then lots of HONN complicated optimization modeling efforts can be saved.

Despite the fact that many ideas have been proposed to apply HONNs for different problems and compare their classification accuracies to ONNs, there were no theoretical studies showing how they are related to each other. Different modifications to HONNs are heuristic, and/or model based inspired. As in any scientific research strategy, a proposed HONN is supported by experimental results to show its superiority upon other HONNs and ONNs. Therefore, we think that HONN importance is vague and its discriminatory power has not been fully exploited. Anyone can notice from the large literature and other scientific web resources that HONNs were not considered as one of the major artificial neural networks categories. Major artificial neural networks categories only include feedforward (ONNs), self organizing maps (Kohonen, 2000), recurrent neural networks (Elman, 1990), Hopfield networks, radial bases functions networks, stochastic neural networks, Boltzmann machine, Neuro-fuzzy networks, pprobabilistic neural networks, see (Duda et al., 2000) for details. Using methodologies similar to what we are going to discuss throughout this chapter, equivalence studies are very important in order to associate and equiv different artificial neural networks categories, this however is out of the scope of this chapter.

2. CONVERSION OF HIGHER ORDER NEURAL NETWORKS TO ITS EQUIVALENT ORDINARY NEURAL NETWORKS

We shall use simple mathematical analysis to show that HONNs are equivalent to ONNs. Without loosing generality, the analysis shown below does not assume a mix of PI and Sigma units at the same layer. Nonetheless, this mix of unites will be resolved later in this chapter. Let us start by defining the activation of neural units and then discuss and state their equivalence.

2.1 The Activation of a Summation Unit (Sigma Unit)

Suppose that $\mathbf{x} = [x_1, x_2, \dots, x_d]$ is the input vector to a neuron unit, in the case we are discussing here, each Sigma unit has the following activation:

$$net_j = \sum_{i=1}^d w_{ji} x_i, \quad (3)$$

where $j = 1, 2, \dots, n$, for a total of n Sigma units at this layer, $w_{ji} \in R$ is the synaptic weight at the connection between the i_{th} input (or the previous) unit and the j_{th} Sigma unit, and R is the set of real numbers.

2.2 The Activation of a Product Unit (PI Unit)

Many forms of HONNs are proposed in the literature, they are all based on PI and Sigma units' structure, and the way their synaptic weights are represented as fixed or adjustable, real-valued or positive integer valued. To discuss the general case, PI units with adaptive real-valued exponent weights is considered in this work. For input \mathbf{x} , each PI unit has the following activation:

$$net_j = \prod_{i=1}^d x_i^{w_{ji}}, \quad (4)$$

where $j = 1, 2, \dots, n$, for a total of n PI units at this layer, $w_{ji} \in R$ is the synaptic weight at the connection between the i_{th} input (or previous) unit and the j_{th} PI unit and R is the set of real numbers. Theoretically, a PI unit may be placed at any layer and its related weights can be calculated in the training phase using backpropagation of error and/or other optimization algorithms.

2.3 Equivalence of PI units with Sigma Units.

The purpose of this section is to find the relation between PI units and Sigma units. By using $\ln(\exp())$, equation (4) can be rewritten as:

$$net_j = \exp\left(\ln\left(\prod_{i=1}^d x_i^{w_{ji}}\right)\right), \quad (5)$$

which can be rewritten as:

$$net_j = \exp\left(\sum_{i=1}^d \ln x_i^{w_{ji}}\right), \quad (6)$$

and using the notation $X_i = \ln x_i$, we may write (6) as

$$net_j = \exp\left(\sum_{i=1}^d w_{ji} X_i\right). \quad (7)$$

Equation (7) states that the activation of a PI unit is the same as the activation of a Sigma unit with two major differences; taking $\ln(\cdot)$ function to the inputs, and taking $\exp(\cdot)$ function to the result of the summation. Thus, converting a PI unit to a Sigma unit is possible using the form shown in (7) after taking care of $\exp(\cdot)$ and $\ln(\cdot)$ functions. How does this affects a HONN when a mix of Sigma and PI layers is used? i.e., does a HONN equivalent to an ONN in this case? This we shall discuss later.

2.4 Conversion of HONNs to ONNs

In this section, the equivalence of PI interconnections to Sigma interconnections is given. This concept can be used later to convert any HONN to its equivalent ONN. Let us consider a HONN with various PI and Sigma units, in this case, four different interconnections are possible between each pair of units: either Sigma-Sigma, PI-Sigma, Sigma-PI, or PI-PI (and the later three types are called PI interconnections). According to this vision, ONNs have only Sigma-Sigma interconnections having the following output at the current Sigma unit:

$$y_j = f_j \left(\sum_{i=1}^d w_{ji} x_i \right), \quad (8)$$

and the next Sigma unit fires the following output:

$$z_k = f_k \left(\sum_{j=1}^n w_{kj} y_j \right). \quad (9)$$

Thus, any HONN can be converted to an ONN if and only if we can reduce its possible PI interconnections to Sigma-Sigma interconnections with outputs as those shown in (8) and (9) and this constitutes the core idea of this work. Since Sigma-Sigma interconnections need no conversion, only the other three interconnections need conversion methodology. Without losing generality, let each layer either contains PI units or Sigma units, nonetheless, this restriction will be resolved later. In practice, the following interconnections might occur:

1. PI-Sigma Interconnection

The first case we shall discuss is a connection from a PI unit to a Sigma unit. In this case, the input vector \mathbf{x} that feeds to the current PI unit comes from the previous layer, and the output of the current PI unit is given by:

$$y_j = f_j(\text{net}_j), \quad (10)$$

where $f_j(\cdot)$ is the activation function acting at the current PI unit, and net_j is PI's activation which has the form shown in (4) or the more progressive one shown in (7). Now, substituting (7) into (10) we get:

$$y_j = f_j \left(\exp \left(\sum_{i=1}^d w_{ji} X_i \right) \right), \quad (11)$$

where $X_i = \ln x_i$. By defining the composite function $\psi_j = f_j(\exp(\cdot))$, it is possible to write (11) as follows:

$$y_j = \psi_j \left(\sum_{i=1}^d w_{ji} X_i \right). \quad (12)$$

The PI output which is y_j that is shown in (12) is then feedforward to the next Sigma unit that has the following output:

$$z_k = f_k \left(\sum_{j=1}^n w_{kj} y_j \right) \quad (13)$$

where $f_k(\cdot)$ is the activation function acting at the Sigma unit. Equation (12) shows that the activation of each PI unit can be expressed as Sigma activation $\sum_{i=1}^d w_{ji} X_i$ similar to (3), but with an acting activation function given by $\psi_j(\cdot)$. It is clear from (12) and (13) that a PI-Sigma interconnection is equivalent to a Sigma-Sigma interconnection after taking $\ln(\cdot)$ to the inputs of the PI unit and using $\psi_j(\cdot)$ as activation function at the PI unit, see Figure1-a for illustration.

2. Sigma-PI Interconnection

Here we are discussing a connection from a Sigma unit to a PI unit. The output of the current Sigma unit is given by:

$$y_j = f_j \left(\sum_{i=1}^d w_{ji} x_i \right), \quad (14)$$

and similar to the treatment shown in (4) to (7), the output of the next PI unit is given by:

$$z_k = f_k \left(\exp \left(\sum_{j=1}^n w_{kj} \ln(y_j) \right) \right). \quad (15)$$

Substituting (14) into (15) yields:

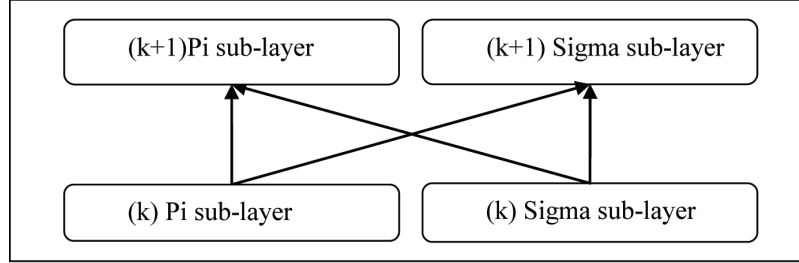
$$z_k = \psi_k \left(\sum_{j=1}^n w_{kj} Y_j \right), \quad (16)$$

with

$$Y_j = \varphi_j \left(\sum_{i=1}^d w_{ji} x_i \right), \quad (17)$$

where $\varphi_j = \ln(f_j(\cdot))$ is the activation function resulted from the composition of $\ln(\cdot)$ and $f_j(\cdot)$, and the other composite activation function acting at the next unit is given by $\psi_k = f_k(\exp(\cdot))$. It is obvious from (16) and (17) that a Sigma-PI interconnection is reducible and equivalent to a Sigma-Sigma interconnection, see Figure1-b for illustration.

Figure 1. Conversion of possible HONN interconnections, a) PI-Sigma to a Sigma-Sigma interconnection, b) Sigma-PI to a Sigma-Sigma interconnection and, c) PI-PI to a Sigma-Sigma interconnection



3. PI-PI Interconnection

Herewith, we have a connection from a PI unit to another PI unit. Analog to PI-Sigma and Sigma-PI interconnections, a PI-PI interconnection can be given as follows:

$$Y_j = \zeta_j \left(\sum_{i=1}^d w_{ji} X_i \right), \quad (18)$$

$$z_k = \psi_k \left(\sum_{j=1}^n w_{kj} Y_j \right), \quad (19)$$

where $\zeta_j = \ln(f_j(\exp(\cdot)))$, and $\psi_k = f_k(\exp(\cdot))$. Y_j is the output of the current PI unit, and z_k is the output of the next PI unit. Again, a PI-PI interconnection is reducible a Sigma-Sigma interconnection, see Figure 1-c for illustration.

As shown above, all PI interconnections can be converted to Sigma-Sigma interconnections and to perform the conversion one only needs to deal with (or find) the activation functions of the equivalent ONN.

3. A SIMPLE METHOD TO CONVERT A HONN TO ITS EQUIVALENT ONN

The previous section shows that any PI interconnection can be directly converted to a Sigma-Sigma interconnection by putting the activation function of each Sigma unit as a composition of $\ln(\cdot)$ and $\exp(\cdot)$, and the original activation function form of the unit. To make life easier in converting a HONN to an equivalent ONN, the following method is proposed:

Define an ONN with the same number of layers and units as that of the HONN you need to convert. Let γ_k be the activation function acting at each unit at the k_{th} layer of the newly defined ONN, and $f_k(\cdot)$ be the original activation function acting at each unit of the k_{th} layer of HONN that we are converting to ONN. Our major target is finding γ_k as a function of $f_k(\cdot)$. Using the equations presented in the previous section, the activation function at each Sigma unit in the newly defined equivalent ONN can be chosen according to the following criteria:

Case 1: Current unit at the k_{th} layer is Sigma and the next unit at the $(k+1)_{th}$ layer is PI yields:

$$\gamma_k = \ln(f_k(\cdot)) \quad (20)$$

Case 2: Current unit at the k_{th} layer is PI, and the next unit at the $(k+1)_{th}$ layer is Sigma yields:

$$\gamma_k = f_k(\exp(\cdot)) \cdot \quad (21)$$

Case 3: Current unit at the k_{th} layer is PI, and the next unit at the $(k+1)_{th}$ layer is PI yields:

$$\gamma_k = \ln(f_k(\exp(\cdot))) \quad (22)$$

Note: $\ln(\cdot)$ should be taken to the inputs whenever they are connected to a PI unit that lies at the first hidden layer.

Using the above conversion criteria, any HONN no matter how complicated can be converted to its equivalent ONN. Since the exponents and multiplications of HONN have been converted to additions in the equivalent ONN that might yield high numerical ranges which needs truncation and rounding, the equivalent ONN have more numerical stability than the original HONN. Using this simple conversion method, we may design very complicated HONNs like recurrent HONNs, cascade correlation HONNs, etc., see (Elman, 1990; Duda et al., 2000; Fahlman & Lebiere C, 1990) then we deal with the equivalent ONNs. Moreover, a bias might be treated as a unit with a fixed input and the generalization of the method to HONNs with bias units is straightforward.

In backpropagation of error one needs the first derivatives of the above activation functions, they are as shown below:

Case 1:

$$\gamma_k = \ln(f_k(\cdot)) \Rightarrow \gamma_k = \ln(f_k(\cdot)) \quad \gamma'_k = [1 / f_k(\cdot)]f'_k(\cdot) \Rightarrow \gamma'_k = [1 / f_k(\cdot)]f'_k(\cdot) \quad (23)$$

Case 2:

$$\gamma_k = f_k(\exp(\cdot)) \Rightarrow \gamma_k = f_k(\exp(\cdot)) \quad \gamma'_k = [f'_k(\exp(\cdot))]\exp(\cdot) \Rightarrow \gamma'_k = [f'_k(\exp(\cdot))]\exp(\cdot) \quad (24)$$

Case 3:

$$\begin{aligned} \gamma_k = \ln(f_k(\exp(\cdot))) & \Rightarrow \gamma_k = \ln(f_k(\exp(\cdot))) \quad \gamma'_k = [1 / f_k(\exp(\cdot))][f'_k(\exp(\cdot))]\exp(\cdot) \Rightarrow \\ \gamma'_k = [1 / f_k(\exp(\cdot))][f'_k(\exp(\cdot))]\exp(\cdot) & \quad (25) \end{aligned}$$

A final important word in this section is that the conversion should be performed consecutively starting from the input layer, the first hidden, and so on until the output layer (current to next) and not the converse, i.e., using a bottom-up approach. Furthermore, having few Sigma to Sigma interconnections at any HONN requires no change in the activation functions at those connections.

4. HAVING SIGMA AND PI UNITS AT THE SAME LAYER

Generally speaking, any HONN may have both PI and Sigma units at the same layer. It is also possible to convert such kind of Mixed-Sigma-PI-Layer HONNs (MSPL-HONN) topology to equivalent ONNs by just manipulating activation functions at the equivalent ONN. As we have shown before, the activation function of each unit in the equivalent ONN depends on the corresponding current unit and the next unit of the original HONN. One can use the same conversion method described previously but the problem that we might face is the possibility of having two activation functions acting at the current unit, i.e., one goes to the next Sigma unit and another to the next PI unit. To solve this problem we suggest a new form of activation function called the dual activation function which acts at one Sigma unit, i.e., a Sigma units with two outputs. Converting a HONN of type MSPL-HONN topology yields this kind of dual activation function at the resultant equivalent ONN, see Figure 2-a that depicts a clear illustration of this case. Adopting this dual activation functionality at a Sigma unit, however, is not so hard to accomplish since each layer can be considered as consisted of two sub layers, a Sigma-sub layer and a PI-sub layer, as depicted in Figure 2-b.

The conversion of the above MSPL-HONN to an equivalent ONN is performed at each sub-layer and one has to find all the activation functions for the equivalent ONN as described in Section 4. Apparently, what we describe here will lead to a new concept in artificial neural networks which is dual activation function neuron, or multi output neuron. The following steps show how to convert MSPL-HONN to its equivalent ONN:

1. Define an ONN structure with the same number of layers and units as that of the MSPL-HONN.
2. Using the bottom up approach previously described, find the activation functions of the equivalent ONN. Each unit may have two activation functions depending on the current unit and the next unit and how PI and Sigma units are distributed.

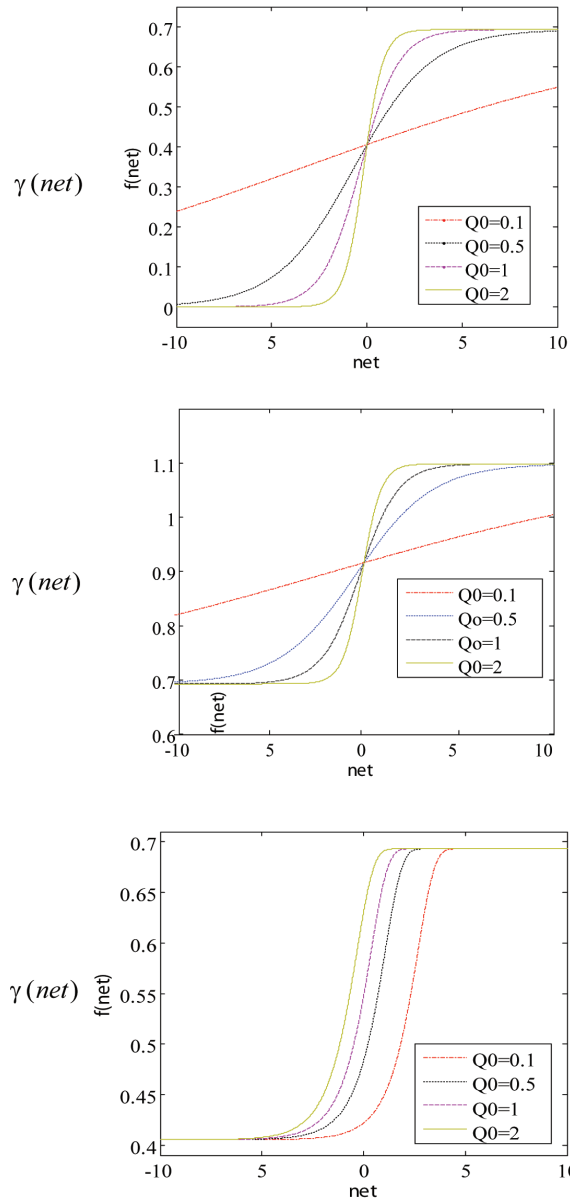
Which fulfills the proof that even a HONN with mixed Sigma and PI units is equivalent to an ONN.

5. ENSURING EQUIVALENCE IN SOFTWARE IMPLEMENTATIONS

Several issues should be considered in order to directly use previous software originally designed for ONN implementation to handle an ONN equivalent to a HONN. The most important issue here is eliminating the possibility of having complex neuron output. The complex neuron output occurs when the input to a PI unit is negative and the synaptic weight of that PI unit has a fractional value.

It is a common practice in neural networks to use sigmoid squash shaped activation functions. This is necessary to make sure that the lower and upper bound of the neuron output are saturated, bounded and to approximate the McCulloch-Pits neuron that employs the step (hard limited) activation function. Since an ONN that is equivalent to some HONN contains $\ln()$ functions, it is important to choose the activation function(s) of that ONN carefully. This is because $\ln()$ functions yield complex values at $x < 0$, and their limit approaches $-\infty$ at $x=0$ which may cause unhandled exception as well as unexpected result. This problem can be easily resolved by making all signals passing into PI units to have positive non zero values, therefore, the output of any unit should be shifted to only positive non-zero values as

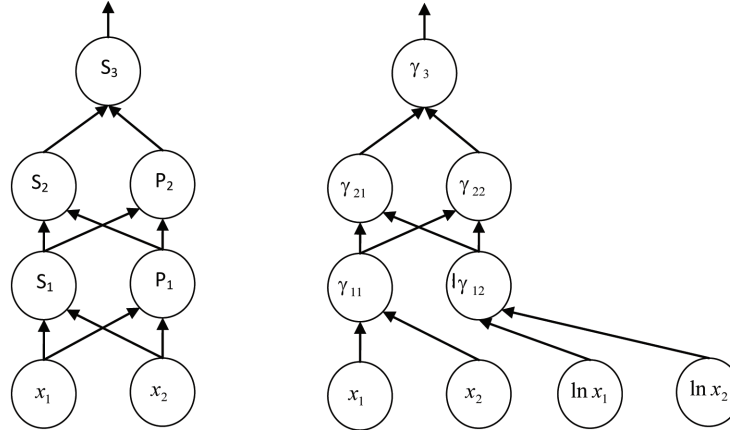
Figure 2. HONN with mixed PI and Sigma units at each layer: a) Shows dual activation function that results from conversion, b) Each layer can be divided into two sub-layers



well as input data and this remarkable solution can be done by adapting the activation function of the original HONN.

Is it possible to do this shifting and/or normalization? Duda et al., (2000) in their pattern classification book noted “In any particular problem, we can always scale the input region to lie in a hypercube, and this condition is not limiting.” As an example of how to shift the unit outputs to positive values let us consider the logistic activation function, we just have to add a tolerance value that makes it greater than zero, the following describes the case:

Figure 3. a) The activation function that result of converting a Sigma-PI interconnection to a Sigma-Sigma interconnection is given by $\gamma(net) = \ln\left[1 / \left(1 + \exp(-Q_0 net)\right) + \varepsilon\right]$ using $\varepsilon = 1$. b) The same activation function of (a) with $\varepsilon = 2$. c) The activation function that result from converting a PI-Sigma interconnection to a Sigma-Sigma interconnection is $\gamma(net) = \ln\left[1 / \left(1 + \exp(-Q_0 \exp(net))\right) + \varepsilon\right]$ using $\varepsilon = 1$ (using $\varepsilon = 2$ yields the same shape but shifted in the y-axis direction with nearly 0.3)

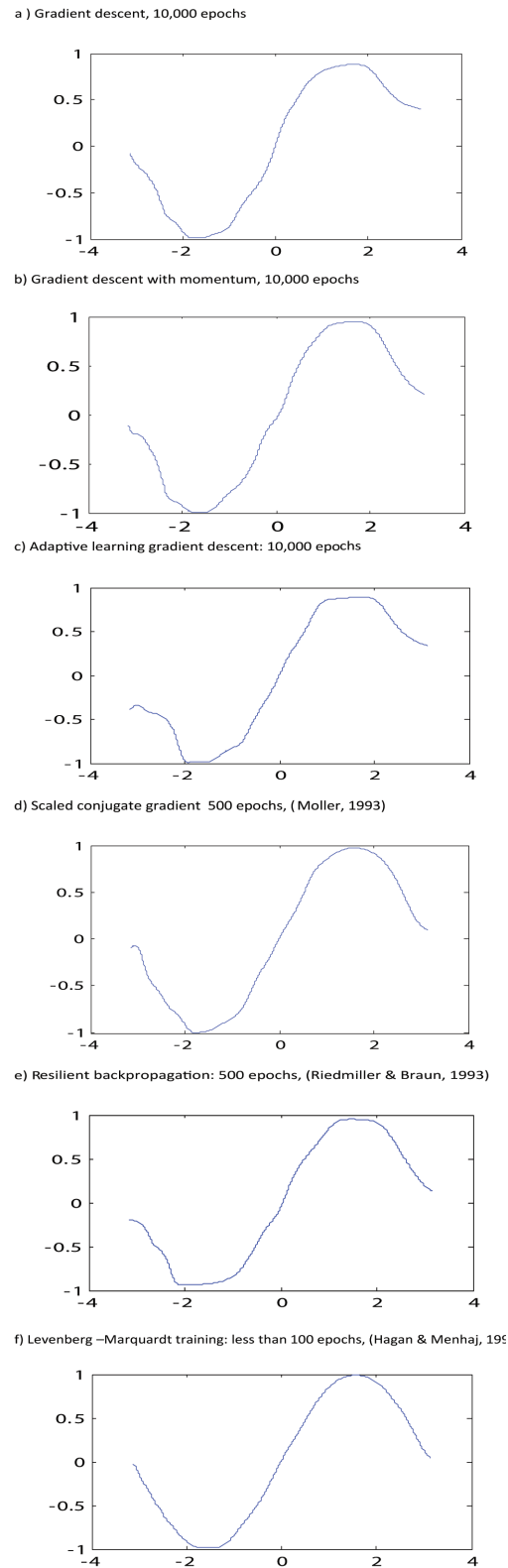


$$f(net) = \frac{1}{1 + \exp(-Q_0 net)} + \varepsilon, \quad (26)$$

where Q_0 is the temperature of the neuron, and $\varepsilon > 1$ is a value that is used to keep the range of the function above 0. Since a PI unit could be placed at the first hidden layer, it is also important to normalize inputs to lie within the unit hypercube I^n where $(I = [0, 1])$. Moreover, we might need to remove the effect of $\ln(0) \rightarrow -\infty$, and $\ln(1) \rightarrow 0$ at the input layer therefore it is also necessary to normalize the input using $(I = [1 + \tau, 2])$ such that $\tau > 0$, i.e., $I = (1, 2]$. As a matter of a fact, it is necessary to do this normalization of the input data even in the direct application of HONNs as a precaution of not setting the activation of some PI units to zero.

If one uses the form shown in (26), the equivalent ONN of some HONN may have at some of its units the following activation function $\gamma(net) = \ln\left[1 / \left(1 + \exp(-Q_0 net)\right) + \varepsilon\right]$, and the behavior of this function for different Q_0 and ε values is illustrated in Figure 3. One can see from Figure 3 that the activation function keeps its S-shape for $\varepsilon = 1$. It must be noted that the range of the activation function depicted in Figure 3 does not lie within $[-1, 1]$ as the sigmoid does, however, this will not pose a problem since this shift will be compensated in the synaptic weights of the network during training. Moreover, the upper part of Figure 3-b is smoother than the lower part, and Figure 3-c shows an asymmetric squash function, see how the upper part saturates faster than the lower part. This really gives an idea of how HONNs behave if they were converted to ONN. Finally, $\varepsilon = 2$ is also a good choice since its lower saturation bound is above zero.

Figure 4.



6. EXAMPLES

Example 1: Suppose that a HONN is consisted of 3 inputs, first hidden layer with 5 PI units with activation function f_1 and an output layer with 3 Sigma units with activation function f_2 . The equivalent ONN can be found as follows:

Construct an ONN with three inputs, first hidden layer with 5 sigma units, and an output layer with 3 sigma units. Now using the method described in the Sec. 3, we can find the activation functions of the first hidden layer γ_1 and the output layer γ_2 , they are given as follows:

$$\gamma_1() = \ln(f_1()) \quad (27)$$

$$\gamma_2() = f_2(\exp()) \quad (28)$$

Example 2: Suppose a HONN with 2 inputs, a PI unit with activation function P_1 and a Sigma unit with activation function S_1 at the first hidden layer, connected to another hidden layer with a PI and a Sigma units having P_2 and S_2 as activation functions respectively, then, the second hidden layer is connected to one Sigma unit output layer with activation function S_3 . The equivalent HONN is as follows:

Using the method described in Sections 3 & 5, the activation functions of the equivalent ONN are as given below:

$$\begin{aligned} \gamma_{31} &= S_3 \\ \gamma_{21} &= S_2, \\ \gamma_{22} &= P_2(\exp()) \end{aligned} \quad (29)$$

$$\begin{aligned} \gamma_{11} &= \begin{cases} S_1 & \text{for connection between } S_1 \text{ and } S_2 \\ \ln(S_1()) & \text{for connection between } S_1 \text{ and } P_2 \end{cases} \\ \gamma_{12} &= \begin{cases} P_1(\exp()) & \text{for connection between } P_1 \text{ and } S_2 \\ \ln(P_1(\exp())) & \text{for connection between } P_1 \text{ and } P_2 \end{cases} \end{aligned} \quad (30)$$

While Figure 4-a contains mixed PI and Sigma units, all units of the graph shown in Figure 4-b are Sigma.

7. EXPERIMENTAL RESULTS

According to the conversion methods discussed in the previous sections, any software designed to simulate ONNs can be used directly to emulate HONNs. This can be done by just changing the activation function at the defined equivalent ONN. In this section, we shall use a neural network toolbox designed for neural networks and all what is needed is writing the source code for the activation functions between different units as that results from HONN conversion as was described in Sections 3 & 5. If one needs to consider HONNs with mixed units at each layer, then, the methods described in section 4 should be taking into account too.

The first problem discussed in this work is the typical nonlinear classification problem, the 2-bit parity problem also known as XOR problem. A PI-Sigma HONN will be used to solve this problem. The input/output data are as shown Table 1:

Since the first hidden layer is a PI layer, each input data is normalized by taking the $\ln(x+1)$, thus, 1's at the input layer are replaced by 1.0986 and a very simple PI-Sigma feed forward neural network with bias will be used. According to section 3 & 5, the activation function of the hidden PI units is $f(\text{net}) = 1/[1 + \exp(-2*\text{net})] + 1$, and the activation function of the output Sigma unit is $f(\text{net}) = 2/[1 + \exp(-2*\text{net})] - 1$. After converting this HONN to its equivalent ONN, it is trained using backpropagation of error, and the network converged well and gave the desired outputs.

In another experiment, a PI-Sigma HONN is used to approximate the Sine function. The input is $x = \{-\pi, 0.01-\pi, \dots, \pi\}$ which is then normalized using $x_{\text{norm}} = \ln[x + \min(x)+1]$ and the output (target) is $y = \sin(x_{\text{norm}})$, therefore, this network has one input and one output units. Furthermore, ten PI hidden units are used in each of the experiments and the transfer function acting at each PI unit is $f(\text{net}) = 1/[1 + \exp(-2*\text{net})] + 1$. Each HONN is converted to an equivalent ONN by changing the activation function as discussed previously. Results of the above mentioned function approximation experiment are shown in Figures. 5 & 6. Various training and/or optimization methods are used to train feedforward and cascade correlation (equivalent) ONNs derived from HONNs.

As shown in Figures. 5 & 6, the best function approximation results are obtained using feedforward trained using Levenberg–Marquardt training as shown in Figure 5-f, and may be the best of all is cascade correlation trained with Levenberg–Marquardt as shown in Figure 6.

8. CONCLUSION

HONNs are equivalent to ONNs! This is the major conclusion of this chapter that may open a new research era in artificial neural networks. The equivalence proof shade new ideas on neural networks mechanism and topological issues, such as, a new discovery that a neuron may have dual activation function depending on HONN topology, so, why not using this phenomena in ONN topology design too? In fact, one might generalize this idea on new triple or quadruple activation function modes and study a new form of artificial neural networks. Other important issues we thrived through this chapter is that one can design any HONN, then, a neural networks software converts it to its equivalent ONN and thus may makes use of all the available neural networks modifications, optimizations, and learning strategies that already exist for ONNs. This means saving lots of time and efforts that one needs to spend in modeling those issues for HONNs from scratch. Due to HONNs complexity, the success of HONNs optimization and training modeling might not be straightforward in getting results and/or convergence of a solution, but, using equivalent ONNs more options are available with flexibility too.

One might look at the equivalence philosophy discussed in this work as a justification of how HONNs performance is related to ONNs and judging whether they are superior or not. After simple normalization of inputs, a HONN and its equivalent ONN have the same number of units and synaptic connections and they only differ by the activation functions acting at each unit. Still they differs with respect to arithmetic operations architecture point of view (multiplications are more complicated than additions) and equivalent ONNs has more complicated activation functions which means more research needs to be done to give a conclusive answer to this issue.

Table 1. XOR data

x_1	x_2	output
0	0	-1
0	1	1
1	0	1
1	1	-1

9. PROBLEMS: SOME OF THE BELOW PROBLEMS MAY SERVE AS SUGGESTIONS FOR FUTURE RESEARCHES

- P1 Convert the HONN that Giles & Maxwell (1987) used to solve the XOR classification into its equivalent ONN.
- P2 Given a fully connected HONN with a bias unit with the following topology: 3 inputs, 11 units (first) hidden layer of which 3 are PI units and 8 are Sigma units, 7 units at the second hidden layer of which 2 are Sigma units and 5 are PI units, and an output layer with one Sigma unit. Assume using $S()$ as Sigma units activation function and $P()$ as PI units activation function, find the equivalent ONN.
- P3 This chapter showed that HONNs can be converted to ONNs, about the converse? Show that it is possible to convert an ONN to an equivalent HONN. Hint: use the activation of one Sigma unit and complete the proof.
- P4 Assume using software that implement multi-layer neural network (ONN) that can be used for training, testing, cross validation, using different optimization methods. State what features should that software possess in order to be implemented on an ONN that was a HONN before conversion?
- P5 Consider ONN with a dual activation function at each unit and an ONN with one activation function at each unit, is there a difference in terms of expressive power and training time? Hint: Perform a training task using some dataset to see the difference.
- P6 In the discussion appeared previously in this chapter, the $\exp(\ln(\cdot))$ have been used to convert

Figure 5. Approximation of Sin function by using a feedforward PI-Sigma HONN implementing exponent weights, results showed the solution found using the equivalent ONN. One can map the final ONN to the corresponding HONN

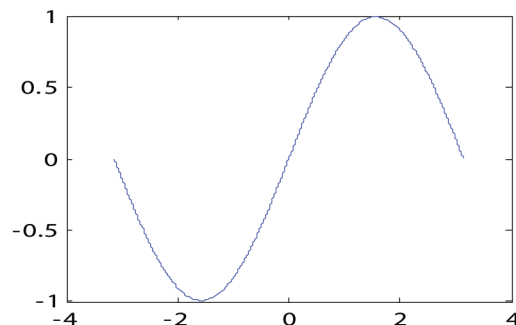
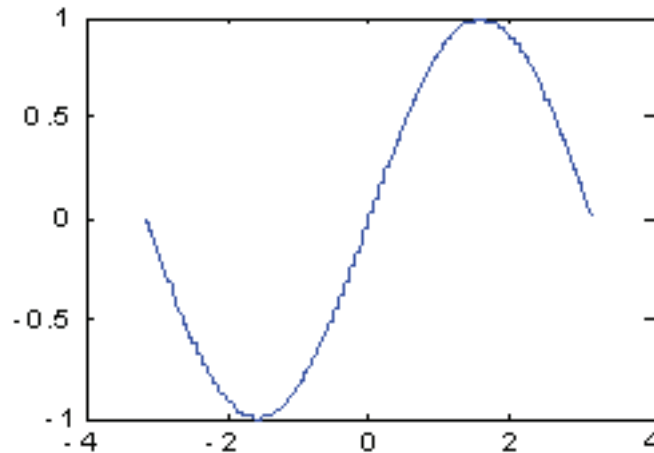


Figure 6. Approximation of sin function by using a cascade correlation PI-Sigma HONN after converting it to an equivalent ONN using cascade correlation (Fahlman & Lebiere C, 1990) implementing Levenberg–Marquardt as optimized training algorithm, less than 100 epochs are needed



the PI unit into a Sigma unit. Is it possible to propose another methodology? If the answer is yes please specify that methodology.

- P7 (a) Calculate the computational complexity of a HONN with d inputs, n hidden PI units, and c output Sigma units. (b) Calculate the computational complexity of the equivalent ONN and compare it with (a). Hint: to complete your discussion of this problem you are urged to read on the complexity of one multiplication and one addition operation (see for instance: The art of computer programming).
- P8 Calculate the computational complexity of an L layers mixed PI Sigma layer HONN, assume that the number of PI and Sigma units in each layer is L_p and L_s respectively. Calculate the computational complexity of the equivalent ONN.
- P9 Give the delta rule (the weight update rule used to train an ONN with backpropagation of error) to find Δw of a PI-PI interconnection layers that is converted to its ONN equivalent interconnection.
- P10 Please specify why is it that the weight of a PI unit appears as an exponent rather than a coefficient.
- P11 Is it possible that a HONN possess more expressive power (see Duda et al., 2000) and have too many local minima than an ONN with the same features. Hint: To be able to discuss this issue, suppose that an ONN_1 is defined with the same topology of some HONN, then convert the HONN to its equivalent ONN call it ONN_h. Compare the features of the two networks to reach a conclusion.
- P12 Let the activation of a PI unit be given by $net_j = \beta \prod_{i=1}^d x_i^{w_{ji}}$ where β is some coefficient that may be fixed or adapted using some training algorithm. Does a HONN with this β coefficient equivalent to an ONN? Justify your answer.

On the Equivalence Between Ordinary Neural Networks and Higher Order Neural Networks

P13 The Taylor expansion of the trigonometric sin function gives the following

$$\sin(x) = \sum_{j=0}^{\infty} (-1)^j x^{2j+1} / (2j+1)!$$

- Construct a HONN to approximate this function.
- Find the equivalent ONN call it ONN_h.
- Use a software tool to train the ONN_h that you designed in (a).
- Compare your results with the following ONN, multi-layer feedforward with the following topology: one input, 10 hidden Sigma units with tanh() activation function, one output Sigma unit with a linear activation function.

Hint: to generate the training set calculate 15 values of sin from $-\pi$ to π , in other words the training set is

$T_R = \{(x, y) : x = -\pi, -\pi + \pi/7, \dots, \pi; y = \sin(x)\}$ Of course, x is the input and y is the target used in training. You may also generate the testing set $\{T_s\}$ as you did in with the training keeping in mind that $T_R \cap T_s = \emptyset$ and $\text{size}(T_R) < \text{size}(T_s)$.

P14 Show that a HONN that has d inputs implementing the translated multiplicative neuron (Iyoda et al, 2003) with its output given by:

$$z_k = f\left(b_k \prod_{j=1}^{n_h} (y_j - w_{kj})\right)$$

$$y_j = f\left(\sum_{i=1}^d (w_{ji} x_i - w_{0j})\right)$$

is equivalent to an ONN.

Hint: The above translated multiplicative NN has Sigma units at the first hidden layer, and PI units at the output layer. Assume that $k=1,2,..,c$ and n_h is the number of output PI units. More precisely, show that the equivalent ONN has the topology: d inputs, n_h fully connected Sigma units at the first hidden layer, n_h (two synapse connections) Sigma units at the second hidden layer, and c fully connected Sigma units at the output layer.

P15 The Higher order Functional Neural Networks HOFNNs (Giles & Maxwell, 1987) is used as a one layer neural network, a multiplication of the inputs that are connected to output units. The output of a neuron of a second order HOFNN is given by:

$$z_k = f\left(w_0 + \sum_{i=1}^d w_i x_i + \sum_{i1=1}^d \sum_{i2=1}^d w_{i1,i2} x_{i1} x_{i2}\right)$$

- Show that the above HOFNN can be represented with the HONN of this chapter that has an activation as shown in Eq. (19) of this work. Therefore, this HOFNN is also equivalent to an ONN.
- Use mathematical induction to show that HOFNN of any order is equivalent to ONN.
- Propose a method to construct multi layer HOFNN, then a conversion to their equivalent ONN to be trained with backpropagation of error.

P16: Given a HONN with d inputs, n_h PI units and c output Sigma units. Find the total number of Sigma units in the equivalent ONN. Does this justify the outperformance of HONN to ONN saying that it needs less size?

P17: Give the delta rule used to train the backpropagation of error algorithms for the equivalent ONN shown in Figure4 (b).

REFERENCES

- Al-Rawi, M. (2006). *Learning affine invariant pattern recognition using high-order Neural Networks*. International Conference on Artificial Intelligence and Machine Learning, (24-29.) Sharm El Sheikh, Egypt.
- Al-Rawi, M., & Bisher, A. (2005). *An improved neural network builder that includes graphical networks and PI nodes*. *Human Computer Interaction*. (229-237), International Conference on Human Computer Interaction, Al-Zaytoonah University, Amman, Jordan.
- Abdelbar, A. (1998). Achieving superior generalization with a high order neural network. *Neural Computing & Applications*, 7(2), 141-146.
- Artyomov, E., & Yadid-Pecht, O. (2005). Modified high-order neural network for invariant pattern recognition. *Pattern Recognition Letters*, 26(6), 843–851.
- Clark, J. (1995). Tracking, code recognition, and memory management with high-order neural networks. *Applications and Science of Artificial Neural Networks*, SPIE 2492, 964-973.
- Duda, R., Hart, P., & Strok, D. (2000). *Pattern classification*. 2nd Ed., New York: Wiley-Interscience.
- Elman, J. (1990). Finding structure in time. *Cognitive Science*, 14, 179-211
- Fahlman, S., & Lebiere, C. (1990). The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems*, 2, 524-532, San Mateo, CA: Morgan Kaufmann.
- Foltyniewicz, R. (1995). Efficient high order neural network for rotation, translation and distance invariant recognition of gray scale images. *Computer Analysis of Images and Patterns. Lecture Notes in Computer Science*, 970, 424-431.
- Fukushima, K., & Wake, N. (1991). Handwritten alphanumeric character recognition by the Neocognition. *IEEE Transactions on Neural Networks*, 2(3), 355-365.
- Ghosh, J., & Shin, T. (1992). Efficient higher-order neural networks for classification and function approximation. *International Journal of Neural Systems*, 3(4), 323-350.
- Giles, C., Griffin, R., & Maxwell, T. (1988). Encoding geometric invariant in higher-order neural networks. In Anderson D (ed), *Neural Information Processing Systems* (pp.301-309), Proceedings of American Institute of Physics Conference..
- Giles, C., & Maxwell, T. (1987). Learning invariance and generalization in high-order neural networks. *Applied Optics*, 26, 4972-4978.
- Hagan, M., & Menhaj, M. (1994). Training feed forward network with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6), 989-993.
- He, Z, & Siyal, M. (1999). Improvement on higher-order neural networks for invariant object recognition. *International Journal of Computer Vision*, 10(1), 49-55.
- Hughen, J., & Hollon, K. (1991). Millimeter wave radar stationary-target classification using a high-order neural network, *SPIE*, 1469, 341-350.

On the Equivalence Between Ordinary Neural Networks and Higher Order Neural Networks

- Iyoda, E., Nobuhara, H., & Hirota, K. (2003). A solution for the N-bit parity problem using a single translated multiplicative neuron. *Neural Processing Letters*, 18(3), 213-218.
- Knuth, D. (1997). *Fundamental Algorithms*. 3rd Ed. Reading, MA: Addison-Wesley.
- Kohonen, T. (2000). *Self organizing networks*: 3rd edition. New York: Springer series in information sciences.
- Kosmatopoulos, E., Polycarpou, M., Christodoulou, M., & Ioannou, P. (1995). High-Order Neural Network Structures for Identification of Dynamical Systems. *IEEE Transactions on Neural Networks*, 2 (6), 422-431.
- Kulisch, U. (2002). *Advanced Arithmetic for the Digital Computer*. 1st Ed. New York: Springer;
- Moller, M. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4), 525-533.
- Minsky, M., & Papert, S. (1969). *Perceptrons*. Cambridge, MA, MIT Press.
- Perantonis, S., & Lisboa, P. (1992). Translation, rotation, and scale invariant pattern recognition by high order neural networks and moment classifiers. *IEEE Transactions on Neural Networks*, 3(2), 241-251.
- Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: the Rprop algorithm. In *Proceedings of the International Conference on Neural Networks*, San Francisco.
- Rovithakis, G. (1999). Robustifying nonlinear systems using high-order neural network controllers. *IEEE Transactions on Automatic Control*, 44(1), 102-108.
- Rovithakis, G., Chalkiadakis, I., & Zervakis, M. (2004). High-order neural network structure selection for function approximation applications using genetic algorithms. *IEEE Transactions Systems Man Cybernetics (B)*, 34(1), 150-158.
- Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning internal representations by error propagation. In Rumelhart D, McClelland J (eds) *Parallel Data Processing* (pp 318-362). Cambridge, MA: The MIT Press.
- Spirkovska, L., & Reid, M. (1993). Coarse-coded higher-order neural networks for PSRI object recognition. *IEEE Trans on Neural Networks*, 4(2), 276 – 283.
- Thimm, G. (1998). *Optimization of high order perceptrons*. A PhD dissertation, Signal processing laboratory of the Swiss federal institute of technology.
- Thimm, G., & Fiesler, E. (1997). High-order and multilayer perceptron initialization. *IEEE Transactions on Neural Networks*, 8(2), 249-259.