University of Belgrade – Faculty of Mechanical Engineering

Najdan Vuković, Milica Petrović, Zoran Miljković

Orthogonal Polynomial Expanded Random Vector Functional Link Neural Networks for Regression and Classification

Working paper

esponding author:

Dr. Najdan Vuković

Assistant Research Professor/ Research Associate

University of Belgrade - Faculty of Mechanical Engineering

Innovation Center

Kraljice Marije 16; 11120 Belgrade 35; Serbia

e-mail: nvukovic@mas.bg.ac.rs or najdanvuk@gmail.com

Belgrade, 2016-2017

Abstract

Feedforward neural networks are among the most used architectures as models for regression. These networks are typically trained with gradient descent using back-propagation of the error between current network output and desired output; that is, weights between hidden layers of the network are determined in iterative procedure by minimizing the error function. However, this process is slow and prone to convergence to local minima. To avoid these pitfalls, one can set randomly input weights and through learning procedure determine only output weights. This approach leads us to Random Vector Functional Link Neural Network (RVFLNN) which through random selection of the input weights enables faster learning. On the other hand, nonlinear expansion of the input vector into higher dimensions enables capturing of higher order information from data. Orthogonal Polynomial Expanded Random Vector Functional Link Neural Network (OPE-RVFLNN) utilizes advantages from both of these approaches: on one hand, it enables nonlinear transformation/expansion of the input, whilst on the other, it introduces random determination of the input weights. In this research, four different nonlinear expansions based on orthogonal polynomials (Chebyshev, Hermite, Laguerre and Legendre) are applied and tested, with five different degrees of orthogonal polynomial expansion and three activation functions (*tansig*, *logsig*, *tribas*). Non-parametric statistical hypotheses testing reveals significant improvement in network performance when one uses *tansig* as activation function, fifth degree of expansion and Chebyshev orthogonal polynomial. Conclusions drawn from this study based on non-parametric statistical hypotheses testing may be used as guidelines for OPE-RVFLNN development and implementation.

Keywords: Random vector functional link neural networks, Orthogonal Polynomial, Ridge regression, nonparametric statistical hypotheses testing

1. Introduction

Artificial Neural Networks (ANN) are among the most used techniques of Machine Learning (ML). They are popular within research and software engineering community because of their ability to model/learn complex nonlinear mappings, which comes as consequence of their universal approximation capability [1]. As a result, they have been applied for regression and classification; if we take a look at a big picture we may see that ANN are successfully applied as models of diverse problems in robotics and engineering [2, 3], etc.

Much of research effort has been devoted to determination of optimal number of neurons (processing units) and optimal number of layers in the network. Single layer feedforward neural networks (SLFN) can achieve high accuracy using certain number of neurons, whereas Multilayer feedforward neural networks (MLFN) are able to model complex relations between input and output sets by exploiting inter-layer relations. There is a vast body of literature devoted to determination of optimal number of neurons for SLFN [4, 5, 6]. Recent improvements in MLFN within Deep Learning framework [7, 8] effectively demonstrate how these networks are used for complex tasks like speech and image recognition [7, 8] etc.

ANN are typically trained by minimizing user-defined cost function and by applying gradient descent (GD) and back-propagating (BP) the error between desired output and the output of the model. Research community has developed a great reservoir of possible training algorithms of ANN using GD and BP [9]. GD learning uses gradient of the cost functions to find direction in which the cost function is minimized; then, the error between input and output is back-propagated to inputs so as to calculate increment of the change of the weights. This procedure is applied iteratively until

minimum of the learning error criterion is reached. However, the main issue in this setup is convergence of the algorithm; the algorithm can get trapped in local minima which results in suboptimal values of network parameters and undesired network response.
One of the methods that bypasses iterative procedure and convergence issue is based on randomization of input weights of the ANN. These weights are randomly chosen from some range and kept fixed whilst training is performed for the output layer weights exclusively. The roots of this idea goes back to beginning of 1990s [10]. Attractiveness of this approach comes as a result of its simplicity, and has become great source of inspiration for community in terms of application for SLFN. However, a proper framework for application of randomization for MLFN learning is still being looked for.
In this research, we focus on special type of SLFN where connection between input and output is achieved through two channels: via hidden layer neurons and via direct link between the input and the output. This ANN architecture has two sub-architectures. The first one is based on Functional Link Neural Networks (FLNN); these networks are formed of single layer network in which nonlinearity of the original input pattern is achieved by increasing the pattern dimension through some nonlinear function. As reader may notice, these networks do not use conventional hidden layer but rather nonlinear enhancement of the input. Typically, one may use expansion in trigonometric polynomial, Chebyshev polynomial or Legendre polynomial. Nonlinear transformation/expansion of the input pattern serves to enhance network's ability to model nonlinear relationship and enhance representation capabilities of the network. The second sub-architecture is conventional SLFN with one hidden layer; the input into this layer of processing units is the same nonlinearly expanded input pattern.
This novel architecture closely resembles Random Vector Functional Link Neural Networks (RVFLNN) [11]; in contrast to other SLFN (like RBFs, ELMs etc) these networks have conventional processing layer of the hidden units and direct link between the input layer and the output layer. Recent research results have shown that this direct link and the hidden layer processing can significantly improve performance of these networks [11]. However, these networks do not apply nonlinear expansion of the input pattern into set of orthogonal functions. By applying this nonlinear transformation of the input vector we try to capture higher order information and use it for modelling. In architecture presented in this research, the input pattern is being transformed through nonlinear transformation and copied: one transformed version is directly fed to output (direct link) while the other one is processed through nonlinear function in the hidden layer of the network. Weights between nonlinear transformation of the input pattern and the hidden layer are randomly determined; the learning is performed between the hidden layer output and the output layer, as well as between nonlinear transformation of the input pattern and the output. This ANN architecture is referred to as Orthogonal Polynomial Expanded Random Vector Functional Link Neural Networks (OPE-RVFLNN).

1.1 Related works

One of the very first results in terms of research and development of RVFLNN has been taken by Pao et al. in [13] where they utilize the advantage of determination of the weights through randomization and functional link network. Their version of RVFLNN has a set of enhancement nodes which are similar to conventional hidden layer in feedforward neural network; the learning algorithm of the developed RVFLNN network is based on generalized delta rule. Another pioneering work in the field is classic results by Schmidt et al. paper [10] in which the authors propose simple neural network with randomly generated input weights; the authors build their model for classification purposes and prove that randomly generated input weights significantly reduces computational burden and enhances classification accuracy.

Recent advances in RVFLNN is given in [14], where input pattern is expanded into one of four possible cases: Chebyshev orthogonal polynomials, Legendre orthogonal polynomials, trigonometric function expansion or random vector expansion. Their functional link-based models involving different nonlinear expansions are applied and tested for audio and speech input signals. To enable robustness of the neural network with random weights authors in [15] propose new probabilistic learning algorithm in which error is distributed according to Laplace distribution. In [12] authors develop their RVFLNN model and through testing assess its performance for electricity demand. Their study showed that direct link between input and output layer of RVFLNN significantly improves performance for time series prediction. On the other hand, through rigorous experimental analysis of RVFLNN Zhang and Suganthan [11] experimentally evaluated RVFLNN performance for 121 benchmark classification problems. Their study reveals several important features for RVFLNN: direct link between input and output is important for network performance, and that ridge regression generates better solutions than Moore-Penrose pseudoinversion. Interesting approach in terms of decentralized RVFLNN learning can be seen in [16], where authors split training data into subsets, while training is performed in decentralized mode. At the very end, the new decentralized learning algorithm finds optimal values for the whole network structure. A broad survey of randomized algorithms for neural network training is published in [17]; reader is kindly referred to it and references therein.

Expansion of input vector/pattern into linearly independent elements has been around in the field for a while and it constantly catches attention. For example, in [18] one can see a single layer functional link ANN where input pattern is nonlinearly expanded using Chebyshev polynomials. Chebyshev series is used because of efficiency and fast convergence when compared to expansions in other set of polynomials. This results in a model that is linear in parameters and nonlinear in the inputs. The training scheme is based on recursive least squares algorithm which guarantees convergence of the Chebyshev neural network weights. Simulation results prove that this approach outperforms multilayer perceptron network in terms of accuracy for real time system identification. Similarly, in [19] authors develop FLNN model, trained with robust H_∞ filter, and use expansions of the input pattern into Chebyshev polynomials, Legendre polynomials and trigonometric expansion. Another application of Legendre orthogonal polynomial expansion instead of using the conventional trigonometric function expansion is given in [20]; Legendre NN (LeNN) is used for modelling of nonlinear channel equalization problem with 4-QAM. Study shows that LeNN outperforms multilayer perceptron and standard FLNN. This field has been great for application of FLNN, for example, in [21] authors propose FLNN with Chebyshev expansion for nonlinear channel equalizer; when compared to MLP their method shows better performance. In [22] we may see recurrent structure of FLNN and how this approach generates models with less computational complexity and lower error measured with Mean Squared Error (MSE). Reduction of noise on basis of FLNN can be seen in [23].
Performance of FLNN in terms of classification can be observed in [24, 25, 26]. New learning algorithm based on evolutionary techniques for FLNN training for classification problems is shown in [24]. Features are being estimated during learning phase of the FLNN which significantly increases time needed for learning but it improves accuracy of the FLNN significantly. Finally, a survey of FLNN and related results may be seen in [25]. Authors provide a road map for higher order networks and FLNN, and introduce a novel scheme for FLNN training using improved Particle Swarm Optimization (PSO) algorithm. Model is tested on classification problems and obtained results show that FLNN could achieve high accuracy with lower complexity. Finally, classification performance of FLNN is tested in [26], where new PSO algorithm is applied for

training of FLNN. In experiments authors effectively show power of FLNN when it comes to model nonlinear decision boundaries.

There are several lines which sets apart this paper from previous research results. Firstly, if we take generic FLNN position, than our OPE-RVFLNN model adds direct connections between nonlinear transformation of the input vector and the output layer. As thorough experimental studies have undoubtedly confirmed [11, 12], these direct links have great importance when it comes to modelling complex nonlinear mappings and they actually add value to model which enhances its accuracy. Other generic FLNN results [13, 18, 20, 25, 26] do not make usage of these direct links. On the other hand, when we take RVFLNN position than our model differs from generic RVFLNN in expansion of the input pattern into orthogonal polynomials. By expanding the input pattern into linearly independent polynomials we try to capture as much as possible of higher order information, important for modelling of highly nonlinear data/problems. Therefore, in comparison with RVFLNN [11, 12, 14, 15, 16] our model establishes direct link between expansion of the input pattern and the output layer. Final line which separates our paper from others is thorough usage of non-parametric statistical hypothesis testing. Namely, we apply almost all statistical procedures proposed in [27-30]; in contrast, in [11] one can find only Friedman test with Nemenyi post-hoc procedure, which is one multiple comparison test with post-hoc procedure. In our research we provide results for two pairwise comparisons and three multiple comparisons tests with additional 10 post-hoc procedures. Conclusions drawn from this study of OPE-RVFLNN performance for regression problems show high statistical significance.

In the next section of the paper we introduce OPE-RVFLNN using FLNN and RVFLNN as basis for it. The third section of the paper brings experimental results obtained on real world datasets. In the fourth part of the paper we provide conclusions drawn from experimental analysis and point out advantages, limitations and possible future directions for the research.

2. Orthogonal Polynomial Expanded Random Vector Functional Link Neural Networks

In mathematics, it is well known fact that non-linear approximation of the orthogonal polynomial is very powerful technique in terms of modelling of the unknown functional dependence, hence combination of the FLNN, RVFLNN and expansion of input set into orthogonal polynomial should result in the neural network able to learn complex nonlinear relations between input-output sets of data. In this paper we propose method that utilizes the RVFLNN input–output pattern and the nonlinear approximation capabilities of orthogonal polynomial for regression. In the reminder we provide basic information.

2.1 Functional Link Neural Network

FLNN is a single layer neural structure in which input vector is transformed (expanded) before entering hidden layer processing units. Let $\mathbf{x}$ be an input vector $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$ output vector, and let $\mathbf{X} = \{\mathbf{x}(i)\}_{i=1}^{N}$ and $\mathbf{Y} = \{\mathbf{y}(i)\}_{i=1}^{N}$ be input and output sets of data (respectively), where N denotes total number of data points. Now, j-th element of output m dimensional vector $\mathbf{y}$ is given with FLNN and it is defined by the following equation:

$$\mathbf{y}_i = \sum_{j=1}^{N_h} \beta_j \mathbf{x}^*(k) = \boldsymbol{\beta}\left(\mathbf{x}^*(k)\right)^T \tag{1}$$

$\boldsymbol{\beta}$ is vector (matrix) of output layer weights, N_h is the number of the hidden neurons. The most important feature of FLNN is the fact that input vector $\mathbf{x}$ is transformed (expanded) through orthogonal polynomial, in other words:

$$\mathbf{x}^* = FE(\mathbf{x}) \tag{2}$$

where $FE(\cdot)$ denotes functional expansion of the input vector $\mathbf{x}$. Now, as reader may see, nonlinearly transformed input pattern is linearly combined with weights; there is another variant of FLNN in which linear mapping is replaced with some nonlinear activation function. For example, we may use logistic sigmoid (*logsig*) or tangent sigmoid (*tansig*). Activation functions used in this paper are given in Table 1. Each $\mathbf{h}(\cdot)$ is called functional link, base or hidden function.

Table 1. Activation functions used in this paper. $\mathbf{h(s)}$ is the output of the activation function and $\mathbf{s} = \mathbf{w}^T\mathbf{x} + \mathbf{b}$.

logsig	$\mathbf{h}(\mathbf{s}) = \dfrac{1}{1+\exp(-\mathbf{s})}$
tansig	$\mathbf{h}(\mathbf{s}) = \dfrac{\exp(\mathbf{s})-\exp(-\mathbf{s})}{\exp(\mathbf{s})+\exp(-\mathbf{s})}$
tribas	$\mathbf{h}(\mathbf{s}) = \max(1-\lvert\mathbf{s}\rvert)$

Therefore, taking the possibility of nonlinear transformation in the hidden layer the general expression for FLNN is given as:

$$\mathbf{y}_i = \sum_{j=1}^{N_h} \beta_j h\left(\mathbf{x}^*(k)\right) = \boldsymbol{\beta}\mathbf{h}^T\left(\mathbf{x}^*(k)\right) \tag{3}$$

In this research paper, we expand input vector x into vector $\mathbf{x}^*$ using one of four different orthogonal polynomials: Chebyshev, Hermite, Legendre and Laguerre. The main characteristics of orthogonal polynomials is that any two subsequent polynomials in the sequence are orthogonal to each other with respect to weight under inner product. Each orthogonal polynomial can be defined using its recurrence equations and for each orthogonal polynomial expansion used in this paper, we show their recurrence equation [31]:

Table 2. Recursive formulae for orthogonal polynomials and their first seven monomials

Chebyshev polynomials of the first kind $\varphi_1(x) = Ch_0(x) = 1$ $\varphi_2(x) = Ch_1(x) = x$ $\varphi_n(x) = Ch_{n+1}(x) = 2Ch_n(x) - Ch_{n-1}(x)$ Orthogonal in interval [-1,1]	$T_0(x) = 1$ $T_1(x) = x$ $T_2(x) = 2x^2 - 1$ $T_3(x) = 4x^3 - 3x$ $T_4(x) = 8x^4 - 8x^2 + 1$ $T_5(x) = 16x^5 - 20x^3 + 5x$ $T_6(x) = 32x^6 - 48x^4 + 18x^2 - 1$

probabilists' Hermite $\varphi_1(x) = H_0(x) = 1$ $\varphi_2(x) = H_n(x) = 2nH_{n-1}(x)$ $\varphi_n(x) = H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x)$ Orthogonal in interval (-∞,∞)	$H_0(x) = 1$ $H_1(x) = x$ $H_2(x) = x^2 - 1$ $H_3(x) = x^3 - 3x$ $H_4(x) = x^4 - 6x^2 + 3$ $H_5(x) = x^5 - 10x^3 + 15x$ $H_6(x) = x^6 - 15x^4 + 45x^2 - 15$
Laguerre $\varphi_1(x) = L_0(x) = 1$ $\varphi_2(x) = L_1(x) = 1 - x$ $\varphi_n(x) = L_{n+1}(x) =$ $= \frac{(2n+1-x)L_n(x) - nL_{n-1}(x)}{n+1}$ Orthogonal in interval [0,∞)	$L_0(x) = 1$ $L_1(x) = -x + 1$ $L_2(x) = \frac{1}{2}(x^2 - 4x + 2)$ $L_3(x) = \frac{1}{6}(-x^3 + 9x^2 - 18x + 6)$ $L_4(x) = \frac{1}{24}(x^4 - 16x^3 + 72x^2 - 96x + 24)$ $L_5(x) = \frac{1}{120}(-x^5 + 25x^4 - 200x^3 + 600x^2 - 600x + 120)$ $L_6(x) = \frac{1}{720}\begin{pmatrix} x^6 - 36x^5 + 450x^4 - 2400x^3 + 5400x^2 \ldots \\ \ldots - 4320x + 720 \end{pmatrix}$
Legendre $\varphi_1(x) = 1$ $\varphi_2(x) = x$ $\varphi_n(x) = (2n+1)xP_n(x) = (n+1)P_{n+1}(x) + \ldots$ $+nP_n - 1(x).$ Orthogonal in interval [-1,1]	$P_0(x) = 1$ $P_1(x) = x$ $P_2(x) = \frac{1}{2}(3x^2 - 1)$ $P_3(x) = \frac{1}{2}(5x^3 - 3x)$ $P_4(x) = \frac{1}{8}(35x^4 - 30x^2 + 3)$ $P_5(x) = \frac{1}{8}(63x^5 - 70x^3 + 15x)$ $P_6(x) = \frac{1}{16}(231x^6 - 315x^4 + 105x^2 - 5)$

These simple recursive equations enables one to generate orthogonal polynomial of input vector x of arbitrary degree of expansion N_e. As an illustrative example, consider expansion of input vector x using Chebyshev polynomial. Let x be two dimensional vector $x \in \mathbb{R}^2; x = [x_1\ x_2]^T$. New pattern (vector) obtained by functional expansion using Chebyshev functions is given by: $\mathbf{h}(\mathbf{x}) = \mathbf{x}^* = [1, Ch_1(x_1), Ch_2(x_2), .., Ch_1(x_2), Ch_2(x_2), \ldots]^T$, where $Ch_i(x_j)$ denotes Chebysev polynomial of degree i of the input pattern j. The same procedure is applied for Hermite, Legendre and Laguerre's expansions. It is important to point out range of orthogonality of these polynomials.

As Table 2 shows Chebyshev and Legendre polynomials are orthogonal with respect to the weight in the same interval, whilst Hermite and Laguerre have different interval. This fact is used in experimental setup to normalize input values into [0,1] range so that all intervals of orthogonality are satisfied .

Typically, training of these networks is performed using gradient descent with backpropagation of the error [9]. However, any type of optimization which assumes calculation of the gradient of the cost function is applicable, for example Kalman filter, unscented filter, information filter [32] etc.

2.2 Random vector functional links neural networks

Random vector functional links neural networks are closely related to conventional SLFN but with additional links going from the input layer to the output layer of the network. Therefore, two channels of communication between the input layer and the output layer are established: the direct one and via the hidden layer. For one training example it looks like this:

$$\mathbf{y}_i = \sum_{m=1}^{N_i} \beta_m \mathbf{x}(k) + \sum_{j=1}^{N_h} \beta_j \mathbf{h}(\mathbf{x}(k)) = \boldsymbol{\beta}\, \mathbf{H}^T(\mathbf{x}(k)) \tag{4}$$

Consider hidden layer matrix formed of all available input data, where N denotes total number of data points used in training:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1(\mathbf{x}(1)) & \mathbf{h}_2(\mathbf{x}(1)) & \dots & \mathbf{h}_N(\mathbf{x}(1)) \\ \mathbf{h}_1(\mathbf{x}(2)) & \ddots & \dots & \mathbf{h}_N(\mathbf{x}(2)) \\ \vdots & \dots & \ddots & \vdots \\ \mathbf{h}_1(\mathbf{x}(N)) & \mathbf{h}_2(\mathbf{x}(N)) & \dots & \mathbf{h}_N(\mathbf{x}(N)) \end{bmatrix} \tag{5}$$

Now, if we write RVFLNN equation in matrix form and solve for output layer weights $\boldsymbol{\beta}$ we obtain:

$$\mathbf{Y} = \mathbf{H}\boldsymbol{\beta} \Rightarrow \boldsymbol{\beta} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{Y} = \mathbf{H}^{+}\mathbf{Y} \tag{6}$$

This direct calculation is known as Moore-Penrose pseudoinversion of the matrix. Another alternative is to prevent numerical instabilities in inversion of **H** so we introduce additional unity matrix multiplied with some constant, i.e.

$$\boldsymbol{\beta} = \left(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I}\right)^{-1}\mathbf{H}^T\mathbf{Y} \tag{7}$$

where λ is coefficient and $\mathbf{I}$ is identity matrix of proper dimensions. This type of calculation for parameter $\boldsymbol{\beta}$ is known as ridge regression (regularized least square solutions). It can be shown that this solution for $\boldsymbol{\beta}$ is exactly the same solution for least-square regularization, i.e.

$$\boldsymbol{\beta} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\left\|\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}\right\|_2^2 + \frac{\lambda}{2}\left\|\boldsymbol{\beta}\right\|_2^2 \tag{8}$$

Now, what makes this solution so interesting? The reader may recall that matrix **H** is formed of the input vector and functional links **h**, which are given as activation functions of input vector and input layer weights. The least square regularized solution which we showed never calculates input layer weights **w**; this is because we set it randomly and then perform minimization and regression. Put differently, input layer weights are chosen in the beginning of the learning process and they are independent of the training data. This approach is well known in the literature [11, 16, 17] and it enables fast training. Input layer weights are drawn from [-1,1] domain.

2.3 Orthogonal Polynomial Expanded Random vector functional links neural networks

In this subsection, we make usage of both FLNN and RVFLNN to form new network architecture OPE-RVFLNN. Now, as a first step we expand the input pattern via nonlinear transformation using one of the orthogonal polynomials given in Table 2 to form new vector: $\mathbf{x}^* = FE(\mathbf{x})$.

In the next step this vector is passed to the hidden layer where activation function is applied and simultaneously linked with the output layer. Therefore, matrix **H** is enlarged with expanded vectors for each available data point, i.e.

$$\mathbf{H} = \begin{bmatrix} \mathbf{x}^*(1) & \mathbf{x}^*(1) & \cdots & \mathbf{x}^*(1) \\ \mathbf{x}^*(2) & \mathbf{x}^*(2) & \cdots & \mathbf{x}^*(2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}^*(N) & \mathbf{x}^*(N) & \cdots & \mathbf{x}^*(N) \\ \mathbf{h}_1\left(\mathbf{w}_1\mathbf{x}^*(1)+\mathbf{b}_1\right) & \mathbf{h}_2\left(\mathbf{w}_1\mathbf{x}^*(1)+\mathbf{b}_1\right) & \cdots & \mathbf{h}_N\left(\mathbf{w}_1\mathbf{x}^*(1)+\mathbf{b}_1\right) \\ \mathbf{h}_1\left(\mathbf{w}_2\mathbf{x}^*(2)+\mathbf{b}_2\right) & \ddots & \cdots & \mathbf{h}_N\left(\mathbf{w}_2\mathbf{x}^*(2)+\mathbf{b}_2\right) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{h}_1\left(\mathbf{w}_N\mathbf{x}^*(N)+\mathbf{b}_N\right) & \mathbf{h}_2\left(\mathbf{w}_N\mathbf{x}^*(N)+\mathbf{b}_N\right) & \cdots & \mathbf{h}_N\left(\mathbf{w}_N\mathbf{x}^*(N)+\mathbf{b}_N\right) \end{bmatrix} \quad (9)$$

The reader is kindly reminded that training inputs are given column-wise. At this place we emphasize that weights **w** and bias **b** between the nonlinearly transformed input vector $\mathbf{x}^* = FE(\mathbf{x})$ and the hidden layer are randomly generated from [-1,1]and [0,1] range respectively. Therefore, OPE-RVFLNN expression is given below:

$$\mathbf{y}_i = \sum_{m=1}^{N_e} \beta_m \mathbf{x}^*(k) + \sum_{j=1}^{N_h} \beta_j \mathbf{h}\left(\mathbf{x}^*(k)\right) = \boldsymbol{\beta}\,\mathbf{H}^T\left(\mathbf{x}^*(k)\right) \quad (10)$$

where N_e denotes degree of expansion of the input vector **x**. When matrix **H** is formed the vector of unknown parameters is then easily determined using ridge regression:

$$\boldsymbol{\beta} = \left(\mathbf{H}^T\mathbf{H} + \lambda\,\mathbf{I}\right)^{-1}\mathbf{H}^T\mathbf{Y} \quad (11)$$

3. Experimental setup

In this section of the paper we analyze performance of OPE-RVFLNN over real world and simulated datasets in terms of accuracy, generalization and compactness of the network. Performance of the newly proposed feedforward neural network architecture is tested in the experimental process over several benchmark problems for regression.

All codes are written and run in Matlab 9.01 environment; experiments are conducted on desktop computer with Intel(R) CoreTMi5-2320 CPU @ 1.6 GHz (2.3 GHz) with 4 GB of RAM, running on 64-bit Windows 7.

Now, at this point we need to emphasize the total number of networks we tested. Firstly, we have four possible settings of expansion in orthogonal polynomial (Table 2). Then we adopted six possible orders of expansion of input pattern into orthogonal polynomial; as Table 2 shows, the first element of the series is always constant and equal to one, while the second one is the input vector itself, so when we say that there is no expansion in orthogonal polynomials we actually take only first two elements of the expansion (constant and linear term). Furthermore, we have used three

different activation functions for the hidden layer (Table 1). Therefore, we have 3·4·5=60 OPE-RVFLNN neural networks. However, we need to assess if direct link influences network performance, so we test all 60 OPE-RVFLNN networks with and without direct link between the nonlinearly transformed input pattern and output. This gives us two more possible combinations, hence, the final number of tested feedforward networks is 60·2=120.
Before any training is done we are to determine numerical values of two independent parameters for each network: the number of hidden neurons N_h and the ridge regression parameter λ. Parameter setting is performed in cross-validation procedure in which 70% of data is used for training of neural models while 30% is used for testing/validation. This procedure is repeated 50 times for each dataset; the optimal setting of parameters for each network is chosen using minimum RMSE calculated over testing/validation set as main criterion. In cross validation ridge regression parameter λ was set to be $(1/2)^C$, where C=[0:20]; the number of hidden neurons N_h was varied in the following range N_h=[1 5 10 50 100 150 200]. Therefore, after cross validation we should have 120 OPE-RVFLNN networks, each with its own optimal setting of N_h and λ. Having determined the optimal setting the real training and testing of network can be performed. The training of networks is done in similar manner (70% of data is used for training and 30% for testing, 50 independent trials for each dataset), but using the optimal setting of N_h and λ for each different OPE-RVFLNN. The accuracy of the network is measured with root mean square error (RMSE) defined as:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N_{test}} \sum_{n=1}^{N_{test}} \left(\mathbf{y}_t - \hat{\mathbf{y}}_t\right)^T \left(\mathbf{y}_t - \hat{\mathbf{y}}_t\right)} \tag{12}$$

The lower numerical value of RMSE for testing set implies better generalization.

4. Statistical inference and experimental results

By applying non-parametric statistical hypotheses testing, this study aims to give answers to these questions:
1) Which of three used activation functions is better?
2) Which orthogonal polynomial expansion is better?
3) Is it better to have direct link between transformed input and the output layer?
4) Which degree of expansion generates optimal results?

To answer these questions we are going to use both pairwise and multiple comparison tests [27-39].

Funding

This research is financed by Serbian Government under grant TR35004 (2010-2016). Their support is gratefully acknowledged.

References

1. K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Networks, 4(2) (1991), 251-257.

2. Z. Miljković, N. Vuković, M. Mitić, Neural extended Kalman filter for monocular SLAM in indoor environment, Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science,230(5) (2016), 856-866.

3. N. Vuković, Z. Miljković, Robust sequential learning of feedforward neural networks in the presence of heavy-tailed noise, Neural Networks, 63 (2015), 31-47.

4. G. B., Huang, P. Saratchandran, N. Sundararajan, A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. IEEE Transactions on Neural Networks, 16(1) (2005), 57-67.

5. N. Vuković, Z. Miljković, A growing and pruning sequential learning algorithm of hyper basis function neural network for function approximation, Neural Networks, 46 (2013), 210-226.

6. M. Bortman, M. Aladjem, A growing and pruning method for radial basis function networks. IEEE Transactions on Neural Networks, 20(6) (2009), 1039-1045.

7. J. Schmidhuber, Deep learning in neural networks: An overview, Neural Networks, 61 (2015), 85-117.

8. Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature, 521(7553) (2015), 436-444.

9. Z. Miljković, D. Aleksendrić, Artificial neural networks—solved examples with theoretical background (In Serbian). University of Belgrade-Faculty of Mechanical Engineering, (2009)

10. W.F. Schmidt, M. A. Kraaijveld, R. P. W. Duin, Feedforward neural networks with random weights, in: Proceedings of the 11th IEEE International Conference on Pattern Recognition, 1992.

11. L. Zhang, P. N. Suganthan, A comprehensive evaluation of random vector functional link networks, Information Sciences, doi:10.1016/j.ins.2015.09.025 (2015).

12. Y. Ren, P.N. Suganthan, N. Srikanth, G. Amaratunga, Random vector functional link network for short-term electricity load demand forecasting, Information Sciences, doi:10.1016/j.ins.2015.11.039, (2016).

13. Y.H. Pao, G.H. Park, D. J. Sobajic, Learning and generalization characteristics of the random vector functional-link net, Neurocomputing, 6(2) (1994), 163-180.

14. D. Comminiello, S. Scardapane, M. Scarpiniti, P. Parisi, A. Uncini, Functional link expansions for nonlinear modeling of audio and speech signals, in: Proceedings of the 2015 IEEE International Joint Conference on Neural Networks (IJCNN), (2015).

15. F. Cao, Y. Hailiang, D. Wang, A probabilistic learning algorithm for robust modeling using neural networks with random weights, Information Sciences, 313 (2015), 62-78.

16. S. Scardapane, D. Wang, M. Panella, A. Uncini, Distributed learning for random vector functional-link networks." Information Sciences, 301 (2015), 271-284.

17. L. Zhang, P. N. Suganthan, A Survey of Randomized Algorithms for Training Neural Networks, Information Sciences, doi:10.1016/j.ins.2016.01.039, (2016), 146-155.

18. S. Purwar, I. N. Kar, A. N. Jha, On-line system identification of complex systems using Chebyshev neural networks, Applied Soft Computing, 7(1) (2007), 364-372.

19. H. K. Sahoo, P. K. Dash, N. P. Rath, NARX model based nonlinear dynamic system identification using low complexity neural networks and robust H∞ filter, Applied Soft Computing, 13(7) (2013), 3324-3334.

20. J.C. Patra, P.H. Meher, G. Chakraborty, Nonlinear channel equalization for wireless communication systems using Legendre neural networks, Signal Processing, 89(11) (2009), 2251-2262.

21. W.D. Weng, C. S. Yang, R. C. Lin, A channel equalizer using reduced decision feedback Chebyshev functional link artificial neural networks, Information Sciences, 177(13) (2007), 2642-2654.

22. H. Zhao, X. Zeng, J. Zhang, T. Li, Y. Liu, D. Ruan, Pipelined functional link artificial recurrent neural network with the decision feedback structure for nonlinear channel equalization, Information Sciences, 181(17) (2011), 3677-3692.

23. S.K. Behera, D. P. Das, B. Subudhi, Functional link artificial neural network applied to active noise control of a mixture of tonal and chaotic noise, Applied Soft Computing, 23 (2014), 51-60.

24. S. Dehuri, S. B. Cho, Evolutionarily optimized features in functional link neural network for classification, Expert Systems with Applications, 37(6) (2010), 4379-4391.

25. S. Dehuri, S. B. Cho, A comprehensive survey on functional link neural networks and an adaptive PSO–BP learning for CFLNN, Neural Computing and Applications, 19(2) (2010), 187-205.

26. S. Dehuri, R. Roy, S.B. Cho, A. Ghosh, A. An improved swarm optimized functional link artificial neural network (ISO-FLANN) for classification, Journal of Systems and Software, 85(6) (2012), 1333-1345.

27. J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation, 1(1) (2011), 3-18.

28. S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Information Sciences, 180(10) (2010), 2044-2064.

29. J. Luengo, S. García, F. Herrera, A study on the use of statistical tests for experimentation with neural networks: Analysis of parametric test conditions and non-parametric tests, Expert Systems with Applications, 36(4) (2009), 7798-7808.

30. J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine learning research, 7 (2006), 1-30.

31. M .Abramowitz, I. A. Stegun, Handbook of mathematical functions: with formulas, graphs, and mathematical tables (Vol. 55). Courier Corporation, (1964).

32. N. Vuković, Machine Learning of Intelligent Mobile Robot Based on Artificial Neural Networks, Doctoral dissertation (in Serbian). University of Belgrade – Faculty of Mechanical Engineering, 2012.

33. K. Bache, M. Lichman, UCI machine learning repository. Irvine, CA: University of California, School of Information and Computer Science., http://archive.ics.uci.edu/ml, (last date of access: July 2, 2016).

34. L. Torgo, Regression DataSets, http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html, (last date of access: July 2, 2016).

35. H. A. Guvenir I. Uysal, Function Approximation Repository, http://funapp.cs.bilkent.edu.tr/DataSets/ , (last date of access: July 2, 2016).

36. D.J. Sheskin, Handbook of parametric and nonparametric statistical procedures, CRC Press, 2003.

37. M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of American Statistical Association, 32 (200) (1937) 675–701.

38. M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, The Annals of Mathematical Statistics, 11 (1) (1940) 86–92.

39. http://sci2s.ugr.es/sicidm/ last date of access: July 2, 2016.