Applied Soft Computing 70 (2018) 1083–1096

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A comprehensive experimental evaluation of orthogonal polynomial expanded random vector functional link neural networks for regression

Najdan Vuković [a,*], Milica Petrović [b], Zoran Miljković [b]

[a] University of Belgrade, Faculty of Mechanical Engineering, Innovation Center, Kraljice Marije 16, 11120 Belgrade 35, Serbia
[b] University of Belgrade, Faculty of Mechanical Engineering, Production Engineering Department, Kraljice Marije 16, 11120 Belgrade 35, Serbia

ARTICLE INFO

Article history:
Received 1 August 2016
Received in revised form
29 September 2017
Accepted 2 October 2017
Available online 12 October 2017

Keywords:
Random vector functional link neural networks
Orthogonal polynomial
Ridge regression
Nonparametric statistical hypotheses testing

ABSTRACT

The Random Vector Functional Link Neural Network (RVFLNN) enables fast learning through a random selection of input weights while learning procedure determines only output weights. Unlike Extreme Learning Machines (ELM), RVFLNN exploits connection between the input layer and the output layer which means that RVFLNN are higher class of networks. Although RVFLNN has been proposed more than two decades ago (Pao, Park, Sobajic, 1994), the nonlinear expansion of the input vector into set of orthogonal functions has not been studied. The Orthogonal Polynomial Expanded Random Vector Functional Link Neural Network (OPE-RVFLNN) utilizes advantages from expansion of the input vector and random determination of the input weights. Through comprehensive experimental evaluation by using 30 UCI regression datasets, we tested four orthogonal polynomials (Chebyshev, Hermite, Laguerre and Legendre) and three activation functions (*tansig*, *logsig*, *tribas*). Rigorous non-parametric statistical hypotheses testing confirms two major conclusions made by Zhang and Suganthan for classification (Zhang and Suganthan, 2015) and Ren et al. for timeseries prediction (Ren, Suganthan, Srikanth, Amaratunga, 2016) in their RVFLNN papers: direct links between the input and output vectors are essential for improved network performance, and ridge regression generates significantly better network parameters than Moore-Penrose pseudoinversion. Our research shows a significant improvement of network performance when one uses *tansig* activation function and Chebyshev orthogonal polynomial for regression problems. Conclusions drawn from this study may be used as guidelines for OPE-RVFLNN development and implementation for regression problems.

1. Introduction

Artificial Neural Networks (ANN) are among the most used techniques of Machine Learning (ML). They are popular because of their ability to model/learn complex nonlinear mappings, which comes as a consequence of their universal approximation capability [1]. As a result, they have been applied for regression and classification, and for diverse problems in robotics and engineering [2,3].

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

ANN are typically trained by minimizing user-defined cost function and by applying gradient descent (GD) and back-propagating (BP) the error between the desired output and the output of the model [7]. GD learning uses gradient of the cost functions to find direction in which the cost function is minimized; then, the error between the input and the output is back-propagated to inputs to calculate the 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 is the convergence of the algorithm.

* Corresponding author.
E-mail addresses: nvukovic@mas.bg.ac.rs, najdanvuk@gmail.com (N. Vuković), mmpetrovic@mas.bg.ac.rs (M. Petrović), zmiljkovic@mas.bg.ac.rs (Z. Miljković).

https://doi.org/10.1016/j.asoc.2017.10.010
1568-4946/

One of the methods that bypasses iterative procedure and convergence issue applies 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 go back to the beginning of 1990s [8]. The attractiveness of this approach comes as a result of its simplicity but a proper theoretical framework is still being looked for.

In this research, we focus on a special type of SLFN where the connection between the input and the output is achieved through two channels: via hidden layer neurons and via direct link between the input and the output.

In this architecture, the input pattern is transformed through the nonlinear transformation and copied: one transformed version is directly fed to the output (direct link) while the other one is processed through the nonlinear function in the hidden layer of the network. Weights between the 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, and between the nonlinear transformation of the input pattern and the output. This ANN architecture is referred to as the Orthogonal Polynomial Expanded Random Vector Functional Link Neural Networks (OPE-RVFLNN).

This ANN architecture has two sub-architectures. The first one is based on the Functional Link Neural Networks (FLNN); these networks are formed of a single layer network in which the nonlinearity of the original input pattern is achieved by increasing the pattern dimension through some nonlinear function. Typically, depending on the various methods of function expansions, the FLNNs are categorized into four groups [9,10]:

1 Random vector FLNN, where the output is modelled as follows:
$$\mathbf{y} = \sum_{i=1}^{N_h} \beta_i g\left(\mathbf{A}_i \mathbf{x} + b_i\right) + \sum_{j=1}^{N} \beta_j \mathbf{x}$$

$g(\cdot)$ denotes nonlinear activation function, A_i and b_i are randomly generated input weight matrix and bias vector, respectively;

1 FLNN with multivariate polynomial basis function: the output is given as follows: $\mathbf{y} = \sum_{r=0}^{R} \sum_{i_1+\ldots+i_N=r} \alpha_{i_1\, i_2 \ldots i_N} x_1^{i_1} x_2^{i_2} \ldots x_N^{i_N}$, where R stands for the order of the polynomial;
2 FLNN with trigonometric basis functions: the input vector is nonlinearly transformed via trigonometric expansion $\left[x_1, \sin(\pi x_1), \sin(2\pi x_1), \ldots, \cos(n\pi x_1), \ldots, x_N, \ldots, \cos(n\pi x_N)\right]$;
3 FLNN with the expansion into orthogonal polynomials. Four orthogonal polynomials are used: Chebyshev, Legendre, Laguerre and Hermite.

The nonlinear transformation/expansion of the input pattern serves to enhance network's ability to model the nonlinear relationship and enhance the representation capabilities of the network. We focus on the nonlinear expansion of the input vector into orthogonal polynomials and try to capture higher order information.

The second sub-architecture of OPE-RVFLNN is conventional SLFN with one hidden layer; the input into this layer of processing units is the same nonlinearly expanded input pattern. This entire architecture is related to the Random Vector Functional Link Neural Networks (RVFLNN) [11]; as in RVFLNN, our architecture is formed of conventional processing layer of the hidden units and direct links between the input layer and the output layer. Recent research results have shown that this direct link and the hidden layer processing significantly improve network's performance [11]. However, unlike basic/standard RVFLNN, we apply nonlinear expansion of the input pattern into the set of orthogonal functions. In contrast to RVFLNN, other single layered neural network architectures (e.g. RBFs, ELMs etc.) do not exploit direct communication between the input layer and the output layer.

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 [12] where they utilize the advantage of the determination of the weights through the randomization and functional link network. Their version of RVFLNN has a set of enhancement nodes, which are similar to the conventional hidden layer in feedforward neural network; the learning algorithm uses the generalized delta rule. Another pioneering work is classic result by Schmidt et al. paper [8] in which the authors propose a simple neural network with randomly generated input weights; the authors build their model for classification purposes and prove that randomly generated input weights significantly reduce computational burden and enhance the classification accuracy. In [13], authors develop Random Activation Neural Net (RAWN) for fast learning in control applications, and show that their approach outperforms conventional backpropagation in terms of accuracy.

Recent advances in RVFLNN are given in [14], where input pattern is expanded. Their functional link-based models 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 the Laplace distribution. In [16] authors test RVFL performance for electricity demand. Their study shows that the direct link between the input and the output layer of RVFLNN significantly improves performance for time series prediction. Similarly, through rigorous experimental analysis, Zhang and Suganthan [11] experimentally evaluated RVFLNN performance for 121 benchmark classification problems. Their study reveals several important features for RVFLNN: the direct link between the input and the output is important for network performance, and that ridge regression generates better solutions than Moore-Penrose pseudoinverse. An interesting approach in terms of decentralized RVFLNN learning can be seen in [17], 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 [18]; the reader is kindly referred to it and the references therein.

A Random projection may lead to the suboptimal solutions and redundant information. To prevent this from happening in random feed forward neural networks, authors in [19] developed a Successive Projection Algorithm (SPA) for ensemble of networks, which utilizes three features: feature selection, pruning of neurons and ensemble selection. Through rigorous experimental procedure, authors demonstrate that their SPA algorithm with RVFL network outperforms ELM and other tested networks.

Interesting results in terms of RVFL development and implementation can be seen in [20]. Namely, through special algorithm called Ensemble Empirical Mode Decomposition (EEMD) authors demonstrate that the ensemble of RVFL networks shows better performance than ELM and MLP for crude oil price prediction (time-series).

Another ensemble of RVFL networks is given in [21], where authors aim to exploit the negative correlation learning to determine the output weights as to develop the ensemble of RVFL networks with high generalization ability.

The expansion of input vector/pattern into linearly independent elements constantly catches research attention. For example,

in [22] the input pattern is nonlinearly expanded using Chebyshev polynomials, because of its 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 the recursive least squares algorithm which guarantees convergence. Simulation results prove that this approach outperforms multilayer perceptron network in terms of accuracy for real time system identification. Similarly, in [23] authors nonlinearly expand input and train their FLNN it with robust H_∞ filter. Another application of Legendre orthogonal polynomial expansion is given in [24]; Legendre NN (LeNN) is used for modelling of nonlinear channel equalization problem with 4-QAM. This field has been great for the application of FLNN, for example, in [25] authors propose FLNN with Chebyshev expansion for nonlinear channel equalizer. In [26] 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). The reduction of noise on basis of FLNN can be seen in [27].

The performance of FLNN in terms of classification can be observed in [9,28,29]. Learning algorithm based on evolutionary techniques for FLNN training is shown in [28]. Features are estimated during the learning phase of the FLNN, which significantly increases time needed for learning but it improves the accuracy of the FLNN significantly. A survey of FLNN and related results may be seen in [9]. Authors provide a road map for higher order networks and FLNN, and introduce a novel scheme for FLNN training using the improved Particle Swarm Optimization (PSO) algorithm. The model is tested on classification problems and results obtained show that FLNN could achieve a high accuracy with lower complexity. Finally, the classification performance of FLNN is tested in [29], where new PSO algorithm is applied for training of FLNN. In experiments, the authors effectively show power of FLNN when it comes to model nonlinear decision boundaries.

Recent advances in deep learning inspired authors in [30] to utilize the best of two worlds: on one side, we have RVFL, while on the other we have Convolutional Neural Networks (CNN). The authors proposed Convolutional Random Vector Functional Neural Network for visual tracking in order to simplify the current state-of-the-art approach based on the pretraining using large auxiliary dataset such as ImageNet [31]. This interesting marriage of two worlds shows that direct links between the input and the output are of extreme importance for network performance.

Finally, in [10] one can find how to add nonlinearity in neural network by expanding the input vector into multivariate polynomial; their functional link neural network is defined in complex domain which does not need activation function. Authors show advantages of functional link neural network and how one could benefit from the expansion of input pattern/vector into higher dimension.

Our research differs from others along these lines. Firstly, OPE-RVFLNN adds direct connections between the nonlinear transformation of the input vector and the output layer. As thorough experimental studies have confirmed [11,16], these direct links have a great importance when it comes to modelling complex nonlinear mappings and they actually add value to the model which enhances its accuracy. Other generic FLNN results [9,14,19,24,29] do not make usage of these direct links.

However, a distinction needs to be made at this place. Namely, references [11,16] analyse performance of RVFL networks with and without direct links for classification and timeseries modelling, respectively. This paper demonstrates the importance of having RVFL with direct links for regression problems.

Our model differs from the generic RVFLNN in expansion of the input pattern into orthogonal polynomials. By expanding the input pattern into linearly independent polynomials, we capture higher order information, which is important for modelling of highly nonlinear data/problems. Therefore, in comparison with RVFLNN [11,14–16,18] our model establishes the direct link between the expansion of the input pattern and the output layer. Finally, our approach differs from [10]: our OPE-RVFL uses nonlinear expansion into orthogonal polynomial while authors in [10] apply nonlinear expansion into multivariate polynomial basis functions. Secondly, authors in [10] exploit complex domain to avoid need for activation functions; our approach relies on activation function being applied after nonlinear expansion into orthogonal polynomial.

At this place, we point out the main difference between the RVFL and the Extreme Learning Machines networks. ELM networks have drawn a lot research attention [32] due to their fast learning capabilities. As previous research results indicate [8,12,33] (and references therein) ELMs are just one variant of RVFL networks. Namely, both ELM and RVFL networks randomly choose input weights and train only output set of weights, but ELM omits direct links between the input vector and the output vector. In contrast, RVFL has additional set of weights for direct links between the input vector and the output vector. Therefore, ELMs represent only one minor subclass of RVFLNN.

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. To fully assess performance of neural network, we apply the non-parametric statistical hypothesis testing [34–37] and provide results for two pairwise comparisons and three multiple comparisons tests. Conclusions are given in the fourth part of the paper, where we point out the advantages, limitations and possible future directions for the research.

2. Orthogonal polynomial expanded random vector functional link neural networks

In mathematics, it is a well known fact that the nonlinear approximation of the orthogonal polynomial is very powerful technique for modelling of the unknown functional dependence. In this paper, we propose a 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 the hidden layer processing units. Let 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 y is given with FLNN and it is defined by the following equation:

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

β 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 the input vector x is transformed (expanded) through orthogonal polynomial, in other words:

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

Where $FE(\cdot)$ denotes the functional expansion of the input vector x. Now, as reader may see, the nonlinearly transformed input pattern is linearly combined with weights; there is another variant of FLNN in which the linear mapping is replaced with some nonlin-

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

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

ear activation function. For example, we may use logistic sigmoid (*logsig*) or tangent sigmoid (*tansig*). Table 1 shows activation functions used in this paper. Each h(·) is called functional link, base or hidden function.

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

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

In this research paper, we expand the input vector x into vector 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. Table 2 shows recurrence equation for each orthogonal polynomial [38]:

These simple recursive equations enable one to generate the orthogonal polynomial of input vector x of arbitrary degree of expansion N_e. Consider the expansion of input vector x using Chebyshev polynomial. Let x be two dimensional vector $\mathbf{x} \in \mathbb{R}^2$; $\mathbf{x} = [x_1\, x_2]^T$. The new pattern (vector) obtained by functional expansion using Chebyshev functions is given by: $\mathbf{h}(\mathbf{x}) = \mathbf{x}^* = [1,\ Ch_1(x_1), Ch_2(x_2), .., Ch_1(x_2), Ch_2(x_2), ...]^T$, where $Ch_i(x_j)$ denotes Chebysev polynomial of degree i of the input pattern j. It is important to point out the 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 the intervals of orthogonality are satisfied.

Typically, the training of these networks is performed using the 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 (Kalman filter, unscented filter, information filter [39] etc.).

2.2. Random vector functional links neural networks

Random vector functional links neural networks are closely related to the 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 following equation shows the network output for one training example:

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

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 β 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. To prevent numerical instabilities we introduce the 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 the complexity coefficient and I is the identity matrix of proper dimensions. This type of calculation for parameter β is known as ridge regression (regularized least square solutions). It can be shown that this solution for β is exactly the same solution for least-square regularization, i.e.

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

The ridge regression [40,41] enables determination of absolute values of the parameters (weights) and penalizes extremely large values. If we fail to address this issue it may cause high variance and result in overfitting of training data. If we have many correlated variables, their coefficients could be large, so the ridge regression is used to shrink weights towards zero. The complexity parameter λ controls by how much we would like to shrink our coefficients. If we set λ = 0 then we have standard Moore-Penrose inverse in which we seek no regularization (shrinkage) of weights.

2.3. Orthogonal polynomial expanded random vector functional links neural networks

In the first step, we expand the input pattern via nonlinear transformation, using one of the orthogonal polynomials given in Table 2, to form a new vector: $\mathbf{x}^* = FE(\mathbf{x})$. Fig. 1 shows OPE-RVFLNN architecture.

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

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

The reader is kindly reminded that training inputs are given column-wise. At this place we emphasize that the weights w and bias b between the nonlinearly transformed input vector $\mathbf{x}^* =$

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

Chebyshev polynomials of the first kind	$T_0(x) = 1$
$\varphi_1(x) = Ch_0(x) = 1$	$T_1(x) = x$
$\varphi_2(x) = Ch_1(x) = x$	$T_2(x) = 2x^2 - 1$
$\varphi_n(x) = Ch_{n+1}(x) = 2\,Ch_n(x) - Ch_{n-1}(x)$	$T_3(x) = 4x^3 - 3x$
Orthogonal in interval [−1,1]	$T_4(x) = 8x^4 - 8x^2 + 1$
probabilists' Hermite	$T_5(x) = 16x^5 - 20x^3 + 5x$
$\varphi_1(x) = H_0(x) = 1$	$T_6(x) = 32x^6 - 48x^4 + 18x^2 - 1$
$\varphi_2(x) = H_n(x) = 2\,n\,H_{n-1}(x)$	$H_2(x) = x^2 - 1$
$\varphi_n(x) = H_{n+1}(x) = 2\,x\,H_n(x) - 2\,n\,H_{n-1}(x)$	$H_3(x) = x^3 - 3x$
Orthogonal in interval (−∞,∞)	$H_4(x) = x^4 - 6x^2 + 3$
Laguerre	$H_5(x) = x^5 - 10x^3 + 15x$
$\varphi_1(x) = L_0(x) = 1$	$H_6(x) = x^6 - 15x^4 + 45x^2 - 15$
$\varphi_2(x) = L_1(x) = 1 - x$	$L_2(x) = \frac{1}{2}\left(x^2 - 4x + 2\right)$
$\varphi_n(x) = L_{n+1}(x) =$	$L_3(x) = \frac{1}{6}\left(-x^3 + 9x^2 - 18x + 6\right)$
$= \frac{(2n+1-x)L_n(x) - n\,L_{n-1}(x)}{n+1}$	$L_4(x) = \frac{1}{24}\left(x^4 - 16x^3 + 72x^2 - 96x + 24\right)$
Orthogonal in interval [0,∞]	$L_5(x) = \frac{1}{120}\left(-x^5 + 25x^4 - 200x^3 + 600x^2 - 600x + 120\right)$
Legendre	$L_6(x) = \frac{1}{720}\left(x^6 - 36x^5 + 450x^4 - 2400x^3 + 5400x^2 \ldots\right.$
$\varphi_1(x) = 1$	$P_0(x) = 1$
	$P_1(x) = x$
$\varphi_2(x) = x$	$P_2(x) = \frac{1}{2}\left(3x^2 - 1\right)$
$\varphi_n(x) = (2n+1)xP_n(x) = (n+1)P_{n+1}(x) + \ldots$	$P_3(x) = \frac{1}{2}\left(5x^3 - 3x\right)$
	$P_4(x) = \frac{1}{8}\left(35x^4 - 30x^2 + 3\right)$
	$P_5(x) = \frac{1}{8}\left(63x^5 - 70x^3 + 15x\right)$
$+nP_n - 1(x).$	$P_6(x) = \frac{1}{16}\left(231x^6 - 315x^4 + 105x^2 - 5\right)$
Orthogonal in interval [−1,1]	

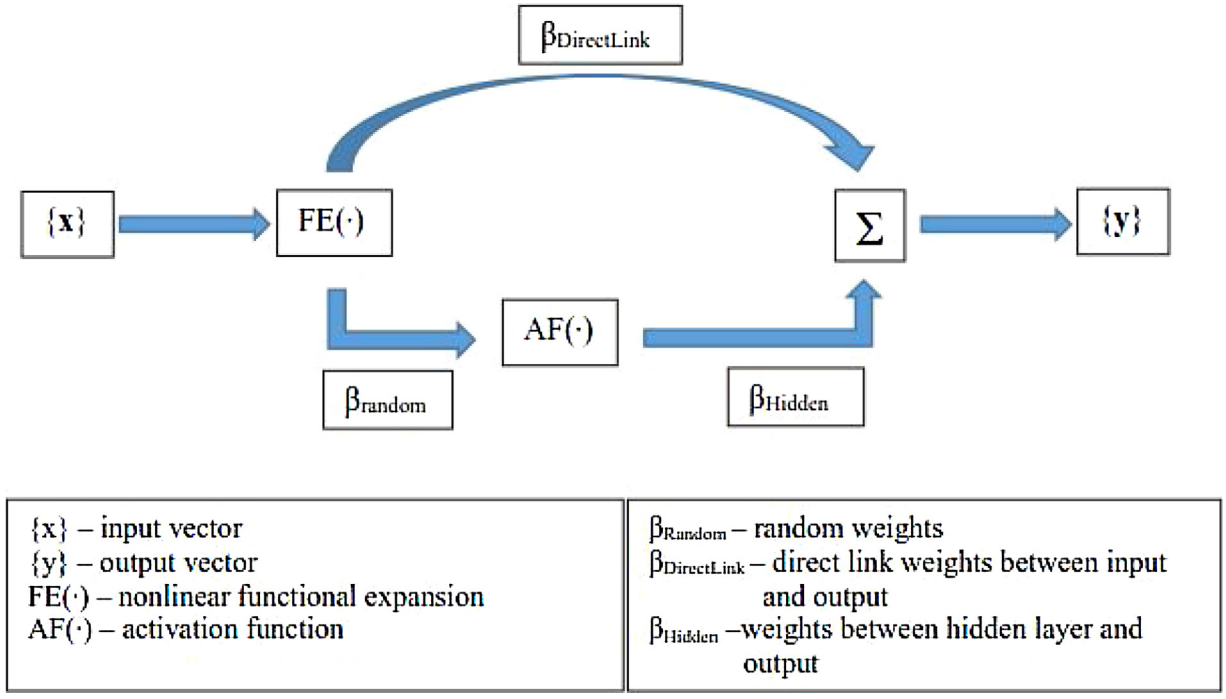

Fig. 1. OPE-RVFLNN architecture. Matrix β_{Random} consisting of input weights is randomly chosen while matrices of β_{Hidden} and $\beta_{DirectLink}$ weights are determined via ridge regression.

$FE(\mathbf{x})$ and the hidden layer are randomly generated from [−1,1] and [0,1] range respectively. OPE-RVFLNN expression is given below:

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

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

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

where λ controls how much we would like to shrink weights towards zero. In general, complexity coefficient λ is determined in cross-validation.

The regularization has a great importance in neural network modelling because it prevents overfitting of model to data. In case

of OPE-RVFLNN, we must prevent the model to learn random noise present in data; the ridge regression represents one way in which we can achieve smaller absolute value of weights and prevent increase of variance and overfitting to training data. The regularization has been within RVFLN community since 1990s, for example, in [42,43] authors use regularization to shrink weights of their network and solve problem of small singular values.

3. Experimental setup

In this section of the paper, we analyse the performance of OPE-RVFLNN over real world and simulated datasets in terms of accuracy, generalization and compactness of the network. The 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 laptop computer with Intel(R) CoreTMi5-2320 CPU @ 1.6 GHz (2.3 GHz) with 6 GB of RAM, running on 64-bit Windows 7.

At this point we emphasize the total number of networks we tested. Firstly, we have four orthogonal polynomials (Table 2). Then we adopted six 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. 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). We have used three different activation functions for the hidden layer (Table 1). Therefore, we have 3·4·5 = 60 OPE-RVFLNN neural networks. To assess influence of direct links on network performance, we tested all 60 OPE-RVFLNN networks with and without the direct links. Therefore, total number of tested feedforward networks is 60·2 = 120.

Before any training is done, we are to determine numerical values of the number of hidden neurons N_h and the ridge regression parameter λ. The 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 = [151050100150200]. After cross validation we should have 120 OPE-RVFLNN networks, each with its own optimal setting of N_h and λ. 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.

Table 3 shows characteristics of regression benchmark problems used in the experiments [44–46].

4. Statistical inference and experimental results

In the hypothesis testing, we draw inferences about populations given experimental results. The main foundations of this process are two hypotheses called the null hypothesis H_0 and the alternative hypothesis H_1. The null hypothesis states that no effect or difference is observed, while the alternative states the opposite. To reject the null hypothesis we need to set some threshold value (level of significance α) below which we are certain that H_0 is not valid. The smallest level of significance α at which we reject the null hypothesis we call the p value. In statistical hypothesis testing it is important to state the p value of the test because it tells how significant the test actually is: the lower the p value, the less evidence we have to accept the null hypothesis H_0 [47].

In contrast to the parametric test, where we assume that our data have certain distribution, non-parametric tests are based on the ranking of the algorithms under investigation. We can perform pairwise comparisons and multiple comparisons. A pairwise comparison assumes the usage of control algorithm (the common approach is to take the best performing algorithm); then, we proceed with testing equality between the control and other algorithms in the study ($1 \times K$ comparison). In contrast, multiple comparisons assume $K \times K$ comparisons where we test the equality between all algorithms.

In this study we have conducted two pairwise and three multiple comparisons. Before we proceed to experimental results we present some common notation used:

L – the number of datasets used in the study

K – the number of algorithms used for testing

By applying non-parametric statistical hypotheses testing, this study aims to give answers to these four questions:

1) Which of three used activation functions is better?
2) Which orthogonal polynomial expansion is better?
3) Is it better to have the direct link between the transformed input and the output layer?
4) Should we use the ridge regression or Moore-Penrose pseudoinverse?
5) Is it better to apply the non-iterative (batch) learning or iterative learning?

To answer these questions we are going to use both pairwise and multiple comparison tests.

Before we start providing answers to research questions, we present RMSE for OPE-RVFLNN, RVFLNN, and ELM obtained on all 30 UCI datasets averaged over 100 independent trials; results are given in Table 4.

Table 4 reveals that OPE-RVFLNN and RVFLNN neural networks outperform ELM in 21 out of 30 cases. This is the expected behaviour of RVFLNN because ELM is RVFLNN but without direct links from the input layer to the output layer, and recent research results [11] have proven that the RVFLNN performance is improved when direct links are present. In case of OPE-RVFLNN, the nonlinear transformation of the input vector into nonlinear orthogonal polynomial leads to the improved performance of OPE-RVFLNN when compared to RVFLNN (in 18 cases out of 30).

Table 5 shows training and testing times for three tested neural networks. As these results indicate, in all tested cases, ELM is faster than its counterparties. This is expected behaviour because ELM has less parameters to optimize. Computational complexity of matrix inversion (7) is $O\left(n^3\right)$, where n denotes the number of dimensions of the input vector. Therefore, a higher number of dimensions implies higher computational complexity and memory requirements. The lower dimension of the input space insures lower computational burden but results in lower performance, as indicated in Table 4. OPE-RVFLNN has higher input dimension than both ELM and RVFLNN, which leads to the increased computational complexity but the bonus side is the improved accuracy (Table 4).

Table 5 reveals one interesting fact. As for testing times, although ELM has outperformed RVFL and OPE-RVFL in terms of training time in all test sets, when it comes to the testing speed

Table 3
30 benchmark regression datasets.

Dataset No.	Dataset Name	Description	Number of Examples	Number of Features
1.	Auto mpg	Predict fuel consumption (mpg)	386	7
2.	Weather Ankara	Predict temperature in Ankara given historic measurements	1609	9
3.	Weather Izmir	Predict temperature in Izmir given historic measurements	1461	9
4.	Mortgage	Predict the 30-year mortgage rate	1049	15
5.	Treasury	Predict one month CD rate	1049	15
6.	Abalone	Predict the age of abalone from physical measurements	4177	8
7.	Kinematics 8NH	Forward kinematics of an 8 link robot arm (nonlinear and highly noisy)	8192	8
8.	Kinematics 32NH	Forward kinematics of an 8 link robot arm (nonlinear and highly noisy)	8192	32
9.	California housing	Predict median value of the house	20460	8
10.	Boston housing	Predict housing values in Boston	506	13
11.	Computer activity	Predict usr, the portion of time that CPUs run in user mode from all attributes	8192	21
12.	Computer activity (smaller dataset)	Predict usr, the portion of time that CPUs run in user mode from restricted set of attributes	8192	12
13.	House 16h	Predict the median price of the house	22784	16
14.	Yacht hydrodynamics	Predict the hydrodynamic performance of sailing yachts	308	6
15.	Wisconsin breast cancer	Predict time to recur	194	32
16.	Triazines	Predict the activity from the descriptive structural attributes	186	60
17.	Pyramidines	The Inhibition of Dihydrofolate Reductase by Pyrimidines	74	27
18.	Machine CPU	Predict relative CPU performance	209	6
19.	Puma8NH	Predict the angular acceleration of one of the robot arm's links (nonlinear and highly noisy)	8192	8
20.	Puma32NH	Predict the angular acceleration of one of the robot arm's links (nonlinear and highly noisy)	8192	32
21.	bank8FM	Predict the fraction of bank customers who leave the bank because of full queues	8192	8
22.	bank32nh	Predict the fraction of bank customers who leave the bank because of full queues (nonlinear and highly noisy)	8192	32
23.	Delta elevators	Controlling the elevators of a F16 aircraft (target is variation)	9517	6
24.	Delta ailerons	Controlling the ailerons of a F16 aircraft (target is variation)	7129	5
25.	Ailerons	Predict the control action on the ailerons of the F-16 aircraft (target is absolute value)	13750	40
26.	Elevators	Predict the control action on the elevators of the F-16 aircraft (target is absolute value)	16599	18
27.	Airfoil noise data	Predict scaled sound pressure level	1503	5
28.	Diabetes	Predict the level of serum C-peptide	43	2
29.	Auto price	Predict the price of the car	159	15
30.	Friedman example	Obtain the value of the target variable Y	8000	10

ELM wins only in nine out of 30 occasions. RVFL has 11 wins while surprisingly OPE-RVFLNN takes eight victories (with two inconclusive cases). This demonstrates the potential of RVFL and OPE-RVFL for on-line data processing; although lagging in time needed for training, these two networks show impressive on-line usability.

The results given in Table 4 effectively show how nonlinear transformation of the input vector into set of orthogonal basis functions improves the performance of neural network, where the performance is measured with RMSE. In 18 out of 30 used datasets OPE-RVFLNN outperforms its "sibling" RVFLNN, which does not use orthogonal polynomial expansion of the input vector.

4.1. Pairwise comparison

Pairwise comparisons are the simplest comparisons one can perform. We will use two tests of this form: the Sign test. in which we estimate the required critical number of wins an algorithm needs to achieve in order to outperform its counterparty, and the Wilcoxon signed rank test where we use the ranking of algorithm and provide the insight whether performances of two algorithms are significantly different.

4.1.1. The Sign test

This test is very simple to conduct and to understand and yet it is the most intuitive one. Given performances of two algorithms this procedure tests how many times out of L, the algorithm i needs to outperform the algorithm j so that we can deduce that it is the better one. It is intuitive to figure out that if performances of two algorithms are equal than they both should have $L/2$ cases in which they outperform the other one; this is our null hypothesis. If it turns out that one of the algorithms has more/less victories then we can estimate the required number of wins it needs to have so that we can reject the null hypothesis of equal performance.

Given two algorithms, the number of wins is distributed according to the binomial distribution. If we have large number of cases, we may assume that the number of wins has Gaussian distribution with parameters $N\left(L, \sqrt{L}/2\right)$. When we apply z test then with certain p (for example 0.1) we say that the critical number of wins should be at least $L/2 + 1.645 \cdot \sqrt{L}/2$. Table 6 shows the required number of wins needed to achieve for four different levels of significance.

To perform this and forthcoming tests we need to declare the control algorithm. Figs. 2 and 3 depict boxplots of RMSE after 100 independent trials and reveal that the minimum RMSE is achieved by:

- the tangent hyperbolic (*tansig*) within the set of activation functions,
- the Chebyshev orthogonal polynomial within the set of orthogonal polynomials,

These are our control algorithms to be used for establishing differences between them and other members of their set. Tables 7 and 8 show results of the test and the detected level of significance.

Table 4
RMSE for 30 dataset obtained by OPE-RVFLNN, RVFLNN and ELM architectures. Better average results are underlined. OPE-RVFLNN and RVFLNN outperform ELM in majority of cases.

Dataset No.	Dataset Name	OPE-RVFL	RVFLNN	ELM
1.	Auto mpg	0.0788	0.0790	0.0817
2.	Weather Ankara	0.0171	0.0173	0.0189
3.	Weather Izmir	0.0190	0.0190	0.0211
4.	Mortgage	0.0082	0.0098	0.0322
5.	Treasury	0.0126	0.0129	0.0323
6.	Abalone	0.0954	0.0844	0.0809
7.	Kinematics 8NH	0.1289	0.1282	0.1409
8.	Kinematics 32NH	0.1319	0.1325	0.1502
9.	California housing	0.2161	0.2181	0.2218
10.	Boston housing	0.0743	0.0764	0.1035
11.	Computer activity	0.0552	0.0386	0.0586
12.	Computer activity (smaller dataset)	0.0657	0.0623	0.0636
13.	House 16h	0.1292	0.2222	0.1964
14.	Yacht hydrodynamics	0.0756	0.0918	0.1194
15.	Wisconsin breast cancer	0.4206	0.4177	0.3
16.	Triazines	0.8592	0.9574	0.9025
17.	Pyramidines	0.1654	0.256	0.4828
18.	Machine CPU	0.1394	0.1671	0.1102
19.	Puma8NH	0.1554	0.1574	0.1712
20.	Puma32NH	0.1565	0.1565	0.1565
21.	bank8FM	0.0776	0.0557	0.0474
22.	bank32nh	0.1322	0.1279	0.1328
23.	Delta elevators	0.0534	0.0534	0.0548
24.	Delta ailerons	0.0389	0.0389	0.0404
25.	Ailerons	0.0545	0.0547	0.0697
26.	Elevators	0.0917	0.0920	0.0939
27.	Airfoil noise data	0.1522	0.1474	0.1509
28.	Diabetes	0.3324	0.4218	0.2648
29.	Auto price	0.0862	0.0910	0.0859
30.	Friedman example	0.0541	0.0597	0.0953

Table 5
Training and testing times for 30 dataset obtained by OPE-RVFLNN, RVFLNN and ELM architectures.

Dataset No.	Dataset Name	ELM Training	RVFL Training	OPE-RVFL Training	ELM Testing	RVFL Testing	OPE-RVFL Testing
1.	Auto mpg	0.0042	0.0158	0.0281	0.0046	0.0023	0.0312
2.	Weather Ankara	0.0045	0.0173	0.0342	0.0054	0.0032	0.0368
3.	Weather Izmir	0.0103	0.0578	0.2568	0.0132	0.0067	0.2363
4.	Mortgage	0.0008	0.0044	0.0102	0.0006	0.0001	0.0122
5.	Treasury	0.0007	0.0047	0.0117	0.0008	0.0002	0.0132
6.	Abalone	0.0033	0.0219	0.1002	0.004	0.0012	0.0958
7.	Kinematics 8NH	0.0075	0.0299	0.0284	0.0699	0.0024	0.0071
8.	Kinematics 32NH	0.0085	0.0342	0.0316	0.0826	0.0029	0.0078
9.	California housing	0.0255	0.1166	0.1714	0.4615	0.0051	0.0235
10.	Boston housing	0.0019	0.0105	0.0108	0.0266	0.0003	0.002
11.	Computer activity	0.0018	0.0116	0.0116	0.0331	0.0004	0.0022
12.	Computer activity (smaller dataset)	0.0096	0.0458	0.0681	0.2062	0.0011	0.0095
13.	House 16h	0.0068	0.0306	0.0151	0.003	0.0315	0.0023
14.	Yacht hydrodynamics	0.0079	0.0372	0.0175	0.0033	0.0365	0.0026
15.	Wisconsin breast cancer	0.0232	0.3197	0.0626	0.0065	0.1123	0.0031
16.	Triazines	0.0015	0.0105	0.0053	0.0005	0.0117	0.0001
17.	Pyramidines	0.0017	0.0129	0.0057	0.0005	0.0132	0.0001
18.	Machine CPU	0.0085	0.1335	0.0243	0.002	0.044	0.0003
19.	Puma8NH	0.0058	0.0185	0.0156	0.0033	0.0233	0.0038
20.	Puma32NH	0.0067	0.0209	0.0207	0.0046	0.0279	0.0044
21.	bank8FM	0.0245	0.068	0.1674	0.0126	0.0793	0.0102
22.	bank32nh	0.0014	0.0056	0.005	0.0003	0.0084	0.0003
23.	Delta elevators	0.0013	0.0065	0.0066	0.0003	0.0108	0.0003
24.	Delta ailerons	0.0091	0.0255	0.0701	0.0033	0.0312	0.0027
25.	Ailerons	0.0055	0.0175	0.0596	0.0035	0.0266	0.0383
26.	Elevators	0.0069	0.0215	0.0667	0.0063	0.0317	0.0446
27.	Airfoil noise data	0.0243	0.1803	0.2792	0.0219	0.3128	0.1731
28.	Diabetes	0.0013	0.0066	0.0245	0.0003	0.0103	0.0122
29.	Auto price	0.0015	0.0067	0.0272	0.0004	0.0118	0.0149
30.	Friedman example	0.009	0.0745	0.1207	0.0052	0.1335	0.0726

With this simple procedure, we can test one interesting feature of OPE-RVFLNN and provide an answer to the following question: should we have the direct link between the nonlinearly expanded input vector and the output vector? As research and results given in [11,16] indicate, RVFLNN can only benefit from these direct links. OPE-RVFLNN with direct links is referred to as OPE-RVFLNN DL, while OPE-RVFLNN without direct links is OPE-RVFLNN NDL; OPE-RVFLNN DL is the control algorithm. Fig. 4 and Table 9 show results.

The results of the statistical test as an answer to the fifth research question is given in Table 10.

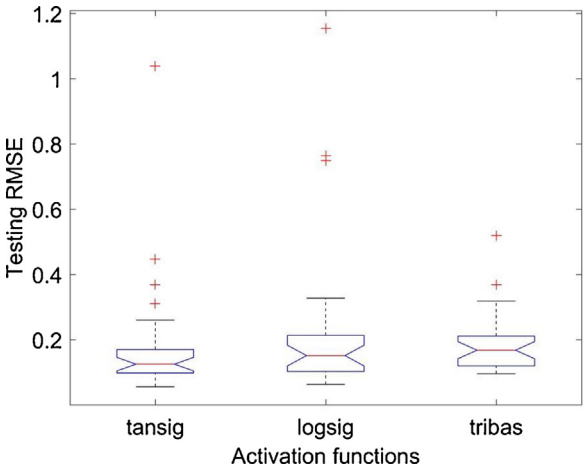

Fig. 2. Testing RMSE for OPE-RVFLNN achieved with three different activation functions.

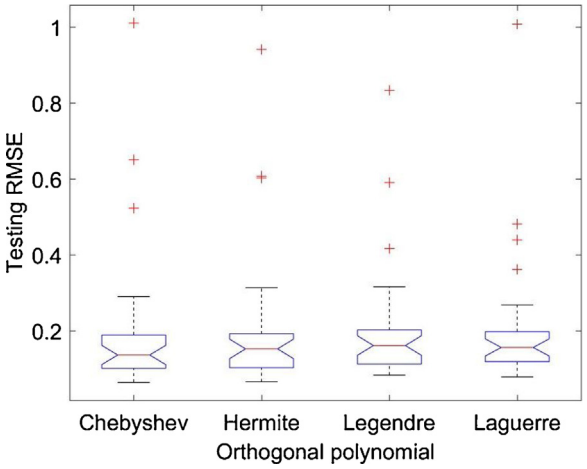

Fig. 3. Testing RMSE for OPE-RVFLNN achieved with four different orthogonal polynomials.

Table 6
Required number of wins for the two-tailed sign test when using 30 datasets and different levels of α. The algorithm is significantly better if it outperforms others on at least number of cases as given in the table.

α	0.1	0.05	0.02	0.01
Percentile	90	95	98	99
z_α	1.645	1.96	2.33	2.58
Critical Number of Wins	19.505	20.3677	21.3810	22.0656

Table 7
The Sign test for pairwise comparison for activation functions using *tansig* as the control; it shows a significant improvement over *logsig* and *tribas*.

	logsig	*tribas*
Wins	28	23
Loses	2	7
Detected differences	$\alpha = 0.01$	$\alpha = 0.01$

Table 8
The Sign test for pairwise comparison for orthogonal polynomials functions using *Chebyshev* as the control; it shows a significant improvement over Hermite, Legendre and Laguerre.

	Hermite	Legendre	Laguerre
Wins	23	25	23
Loses	7	5	7
Detected differences	$\alpha = 0.01$	$\alpha = 0.01$	$\alpha = 0.01$

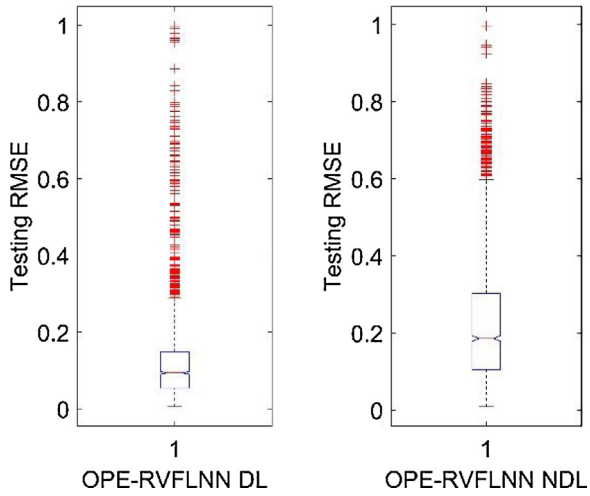

Fig. 4. Testing RMSE for OPE-RVFLNN DL versus OPE-RVFLNN NDL.

Table 9
The Sign test for pairwise comparison using OPE-RVFLNN with direct link as the control.

	OPE-RVFLNN NDL
Wins	25
Loses	5
Detected differences	$\alpha = 0.01$

Table 10
The Sign test for pairwise comparison using OPE-RVFLNN with ridge regression (OPE-RVFLNN RR) as the control.

	OPE-RVFLNN RR
Wins	24
Loses	6
Detected differences	$\alpha = 0.01$

Table 11
The Sign test for pairwise comparison using OPE-RVFLNN with batch learning (non-iterative NI) as the control.

	OPE-RVFLNN Non-Iterative
Wins	14
Loses	16
Detected differences	–

In similar manner, we proceed to provide answer related to which regression we should use: Moore-Penrose or ridge regression? Table 10 summarizes results of sign test for pairwise comparison; OPE-RVFLNN with ridge regression (OPE-RVFLNN RR) outperforms OPE-RVFLNN with Moore-Penrose regression (OPE-RVFLNN MP) in 24 out of 30 cases. Detected high level of significance points towards safe rejection of the null hypothesis of equality between two regression approaches.

Final test in this part of the paper is related to assess which learning approach is better: non-iterative learning or iterative learning – the sixth research question. Table 11 shows results.

Now, this test reveals that none of the tested learning approaches has an advantage. The non-iterative learning outperforms iterative one in 14 out of 30 cases, which is not enough to make any conclusion. However, what we can say is that Sign test failed to reject the null hypothesis, meaning that two tested approaches show same behaviour; there are no statistical significant differences between two learning approaches and their performances could be perceived as equal.

Four out of five tested cases (activation functions, choice or orthogonal polynomial, direct link, regression choice) depict high level of significance (α=0.01) which indicates the strong evidence against the hull hypothesis of equality between the algorithms. Based on the Sign test for pairwise comparison we may conclude these four facts (>stands for "is better than" or "it outperforms"; this symbol is used in Decision theory and Economics to denote "is preferred to"):

1) *tansig* > *tribas* > *logsig*
2) Chebyshev > Legendre; no significant difference is detected between the performances of Hermite and Laguerre orthogonal polynomial expansion when Chebyshev is used as the control method.
3) OPE-RVFLNN DL > OPE-RVFLNN NDL
4) Ridge regression > Moore-Penrose pseudoinverse

However, the Sign test fails to reject the null in the case of non-iterative learning versus iterative learning; this indicates that these two algorithms have the identical performance.

The final remark in this section of the paper is devoted to the results given in Table 4. As it can be seen, direct linked networks (OPE-RVFLNN and RVFLNN) outperform ELM in 21 out of 30 test cases, which means that the level of significance is 0.05 (Table 5). When compared to RVFLNN, OPE-RVFLNN shows better performance in 18 cases, and when we exclude three identical performances from analysis, the detected level of significance is 0.1. The performance of OPE-RVFLNN is better than the performance of RVFLNN for dataset known to be highly nonlinear (Kinematics32HN, California housing, Boston housing, Puma8NH, Elevators, Ailerons). Both networks show the identical performance for Puma32NH, Delta elevators and Delta ailerons, while RVFLNN outperforms OPE-RVFLNN in case of bank32nh and Kinematic8NH. These results indicate that the nonlinear expansion of the input vector into set of orthogonal polynomial improves the performance of the network when highly nonlinear datasets is being modelled/processed.

4.1.2. The Wilcoxon signed rank test

The Wilcoxon signed rank test for pairwise comparison is statistical procedure aimed to detect the significant difference between the means of two samples. The test may be seen as t test for the nonparametric statistical inference.

The test is performed in the following manner. Firstly, we subtract two data samples, which gives us the vector of differences d_i, and then we rank them according to their absolute values. Now, we compute two sums and use them as our statistics:

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i)$$
$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \quad (13)$$

As it may be seen, in case of $d_i = 0$ we split them equally between two samples. The test statistic is defined as the minima of these two: $T = \min(R^+, R^-)$. The test procedure proceeds with the comparison of this statistic with critical values given by Wilcoxon distribution for L degrees of freedom. If T is equal or smaller than the critical value the null hypothesis of equality is rejected with defined level of significance α (deduced from the table of critical values) and associated p value. This test is more sensitive than t test because it takes into account absolute differences between two samples; it emphasizes greater and smaller differences. Tables 12–14 provide detailed information. We stress out that sum of R^+ and R^- should be equal to $R^+ + R^- = \frac{L\cdot(L+1)}{2}$, which in our case results in 465.

Table 12
The Wilcoxon signed rank test for pairwise comparison for activation functions using *tansig* as the control; it shows a significant improvement over *logsig* and *tribas*. In both cases the detected level of significance is α=0.01.

	tansig versus *logsig*	*tansig* versus *tribas*
p value	0.000006983783147	0.005667172691498
R^+	451	367
R^-	14	98

Table 13
The Wilcoxon test for pairwise comparison for orthogonal polynomials functions using *Chebyshev* as the control; it shows a significant improvement over Hermite (α=0.02), Legendre (α=0.05) and Laguerre (α=0.02).

	Chebyshev versus Hermite	Chebyshev versus Legendre	Chebyshev versus Laguerre
p value	0.011748106348303	0.025637124030658	0.013194169865414
R^+	355	341	353
R^-	110	124	112

Table 14
The Wilcoxon signed rank test for pairwise comparison for detecting significant differences between OPE-RVFLNN DL and OPE-RVFLNN NDL; OPE-RVFLNN DL shows a significant improvement over OPE-RVFLNN NDL with the detected level of significance is $\alpha = 0.01$.

	OPE-RVFLNN DL and OPE-RVFLNN NDL
p value	0.002
R^+	383
R^-	82

Table 15
The Wilcoxon signed rank test for pairwise comparison for detecting significant differences between OPE-RVFLNN with ridge regression and OPE-RVFLNN with Moore-Penrose; OPE-RVFLNN with ridge regression shows a significant improvement over OPE-RVFLNN with Moore-Penrose.

	OPE-RVFLNN with ridge regression and OPE-RVFLNN with Moore-Penrose
p value	1.149921754357014E-6
R^+	420
R^-	45

The Wilcoxon signed rank test for pairwise comparison reveals some interesting facts. Firstly, when performance of activation functions is analysed we can notice very low associated p values; both levels of significance are lower than $\alpha = 0.01$ which indicates strong evidence against the null hypothesis. Finally, when we compare the performance in terms of orthogonal polynomials we see that the detected level of significance is $\alpha = 0.02$ for Hermite and Laguerre whilst for Legendre $\alpha = 0.02$; this indicates that Chebyshev is a significant improvement over Hermite, Legendre and Laguerre orthogonal expansion.

The next usage of Wilcoxon signed rank test is devoted to detecting possible differences between OPE- RVFLNN DL and OPE-RVFLNN NDL. Table 14 shows results. As one may see, direct links between the nonlinearly transformed/expanded input pattern and the output layer outperform the generic OPE-RVFLNN NDL.

The Wilcoxon signed rank test performed to detect whether the ridge regression is better choice than Moore-Penrose pseudoinversion (fifth research question) is given in Table 15.

As Table 15 shows, Wilcoxon signed rank test confirms that OPE-RVFLNN with ridge regression generates lower RMSE over test set, meaning there is a significant difference between the means of two samples.

The final usage of the Wilcoxon signed rank test is to see if performances of non-iterative learning and iterative learning statistically differ. The results are given in Table 16.

Table 16
The Wilcoxon signed rank test for pairwise comparison for detecting significant differences between the OPE-RVFLNN with non-iterative learning and the OPE-RVFLNN with iterative learning.

	OPE-RVFLNN with non-iterative learning and OPE-RVFLNN with iterative learning
p value	0.9918
R^+	233
R^-	232

The Wilcoxon signed rank test (Table 16) fails to reject the null hypothesis in case when we test the non-iterative versus iterative learning. This is confirmed with extremely great p value, which indicates a strong evidence in backing up the null hypothesis: there are no statistically significant differences in performances of two tested approaches.

4.2. Multiple comparisons with a control method

4.2.1. Multiple sign test

This test enables researcher to quickly figure out if there are differences between the control and the others. The test procedure is carried out in the following manner:

- Compute signed differences between other algorithms and control algorithm: $d_j^i = x_j^i - x_1^i$, where d_j^i stands for the difference between the j-th algorithm and the control algorithm achieved on i-th test set; in the same manner x_j^i and x_1^i represent the performance of the j-th algorithm achieved on i-th test set and the performance of the control algorithm on i-th testes, respectively;
- Let r_j denote the number of differences that have less frequently appearing sign within d_j^i;
- Denote with M_j and M_1 medians of two samples (*1* stands for control and *j* for one of the algorithms); to test H_0:$M_j \geq M_1$ versus H_1:$M_j < M_1$, we need to count number of pluses and determine if it is less than or equal to critical value defined with $K-1$ degrees of freedom
- To test H_0:$M_j \leq M_1$ versus H_1:$M_j > M_1$, we need to count number of minuses and determine if it is less than or equal to critical value defined with $K-1$ degrees of freedom.
- Table of critical values may be found in [34,35].

We choose to test H_0:$M_j \geq M_1$ versus H_1:$M_j < M_1$, with $K-1$ degrees of freedom. Now, we can use Tables 7 and 8 (Single Sign test) as reference. In the case of testing *tansig* activation function versus two others our critical number of pluses is nine. When we take a look at Table 7 we may see that *logsig* and *tribas* outperformed *tansig* on less than nine occasions (two in case of *logsig* and seven in case of *tribas*); therefore, we reject the null hypothesis and conclude that *tansig* is better than *logsig* and *tribas*.

The final case is to test which orthogonal polynomial to use. The critical value for three degrees of freedom is eight. Observing Table 8 we see that the null can be rejected and we conclude that Chebyshev expansion is better than others.

4.2.2. Rank-based non-parametric tests: the Friedman, Friedman aligned Ranks and Quade test

The Friedman test [48,49] can be seen as nonparametric case of the parametric two-way analysis of variance. The test provides researcher with answer to the following question: for $K \geq 2$ samples of data, can we observe at least two samples of data with different median values? In Friedman's test the null hypothesis states equality of medians of two data samples, hence no direction of difference can be deduced from this test. The test procedure is simple and it assumes following steps:

1. Rank all algorithms according to their average performance for each problem i; the best algorithm will take rank 1, the second best 2 etc. Denote these ranks as r_j^i, meaning rank of algorithm j calculated on test problem i;
2. For each tested algorithm j compute average of its ranks achieved on all datasets, i.e.

$$R_j = \frac{1}{L}\sum_{j=1}^{L} r_j^i \tag{14}$$

The Friedman statistics F_F is than given by:

$$F_F = \frac{12L}{K(K+1)}\left[\sum_j R_j^2 - \frac{K(K+1)^2}{4}\right] \tag{15}$$

This is distributed as χ^2 with $(K-1)$ degrees of freedom. When we compare F_F with critical values of χ^2 we may make decision if we are allowed to reject the null hypothesis.

Besides Friedman test, Iman and Davenport derived another statistic to compensate conservatism of generic Friedman:

$$F_{ID} = \frac{(L-1)\,\chi_F^2}{L(K-1)-\chi_F^2} \tag{16}$$

which undergoes F distribution with $(K-1)$ and $(L-1)(K-1)$ degrees of freedom. It is straightforward to find critical value $F(K-1,(K-1)(L-1))$ and conclude whether we reject the null hypothesis or not.

Besides these two tests, one can employ Friedman Aligned Rank test, where we take into account average performance achieved by all tested algorithms on i-the test problem. Then we calculate differences of this value and performance by j-th algorithm and rank algorithms based on these differences. These differences are usually called aligned observations, hence the name for the test. The statistic for Friedman Aligned Test is:

$$F_{AR} = \frac{(K-1)\left[\sum_{j=1}^{K}\left(R_j^i\right)^2 - \frac{K\cdot L^2\,(K\cdot L+1)^2}{4}\right]}{\frac{K\cdot L\cdot(K\cdot L+1)(2\cdot K\cdot L+1)}{6} - \frac{1}{K}\sum_{i=1}^{L}\left(R_j^i\right)^2} \tag{17}$$

F_{AR} statistic is compared to χ^2 with $(K-1)$ degrees of freedom.

Final procedure takes into account how hard/difficult the i-th problem really is and this is invoked into testing procedure by calculating the difference (range) between the greatest and the smallest performance of j-th algorithm on i-th test problem: $Range_i = \max_j x_j^i - \min_j x_j^i$. Then, we assigned ranks for the problems according to their *Range*. The reader is kindly referred to [34] and references therein for additional information.

Having briefly introduced and described Friedman, Friedman Aligned Ranks and Quade test we proceed with testing procedures and provide experimental results obtained. Results are given in Tables 17 and 18.

The results of non-parametric statistical tests indicate that in all tests we can reject the null hypothesis based on fact that computed test statistic is greater than the critical value. This is reinforced with computed p values; all are extremely low. Therefore, given these results of three statistical tests we can safely conclude that *tansig* (Table 17) and Chebyshev polynomial (Table 18) generate significantly different results when compared directly to other members of their sets.

Table 17
Friedman, Friedman Aligned and Quade ranks, test statistics, *p* values and critical values for activation functions used in the study.

		tansig	*logsig*	*tribas*
Friedman	Rank	1.3	2.2	2.5
	Test statistic, p value and critical value	$F_F = 23.4$ p value = 8.293840555451126E-6 $Critical\_value = \chi^2_2 = 5.9915$ $F_{ID} = 18.54$ p value = 5.950661074414772E-6 $Critical\_value = F(2, 58) = 3.15$		
Friedman Aligned	Rank	24.63	55.5	56.36
	Test statistic, p value and critical value	$F_{AR} = 22.09$ p value = 1.590852110378016E-5 $Critical\_value = \chi^2_2 = 5.9915$		
Quade	Rank	1.39	2.34	2.26
	Test statistic, p value and critical value	$F_Q = 7.71$ p value = 0.001069661677932122 $Critical\_value = F(2, 58) = 3.15$		

Table 18
Friedman, Friedman Aligned and Quade ranks, test statistics, *p* values and critical values for orthogonal polynomial used in the study.

		Chebyshev	Hermite	Legendre	Laguerre
Friedman	Rank	1.63	2.23	3.2	2.93
	Test statistic, p value and critical value	$F_F = 27$ p value = 5.887377367730373E-6 $Critical\_value = \chi^2_3 = 7.8147$ $F_{ID} = 12.43$ p value = 7.703966960387538E-7 $Critical\_value = F(3, 87) = 2.7094$			
Friedman Aligned	Rank	38.46	54.4	79.5	69.63
	Test statistic, p value and critical value	$F_{AR} = 24.33$ p value = 2.1290844862464908E-5 $Critical\_value = \chi^2_3 = 7.8147$			
Quade	Rank	1.9	3.33	2.94	2.82
	Test statistic, p value and critical value	$F_Q = 10.08$ p value = 9.053939521736683E-6 $Critical\_value = F(3, 87) = 2.7094$			

5. Concluding remarks and future research directions

In this study we have conducted a comprehensive experimental evaluation of Orthogonal Polynomial Expanded Random Vector Functional Link feedforward neural networks (OPE-RVFLNN) for regression. We have tested three types of activation functions, five degrees of nonlinear expansion into orthogonal polynomial and four different orthogonal polynomials; therefore, 120 OPE-RVFLNN neural networks have been tested using 30 USI datasets. Training results have been subjected to non-parametric statistical tests by applying both pairwise and multiple comparisons, to assess optimal setting of OPE-RVFLNN parameters and evaluate its overall performance. Thorough non-parametric statistical analysis points out these main conclusions from our study:

1 Performance of OPE-RVFLNN benefits from direct links between the nonlinearly expanded input layer and the output layer. As stated in the introductory part of the paper, this is the first research showing importance of direct links for RVFL implementation for regression problems. Previous research efforts demonstrate benefits from direct links for classification [11] and timeseries prediction [16]. In this study, we confirm findings in [11,16]. The importance of direct links for network's performance is shown in [30] as well;
2 All conducted statistical tests have confirmed that *tansig* activation function generates statistically significantly better results than other activation functions used in the study. Based on the results of non-parametric statistical tests we can safely conclude: *tansig* > *logsig* > *tribas*
3 The nonlinear expansion into Chebyshev orthogonal polynomial outperforms Legendre, Laguerre and Hermite expansions; non-parametric statistical test have revealed that the second best choice would be Hermite. The conducted statistical analysis points out following conclusion:

Chebyshev > Hermite > Laguerre > Legendre

However, we point out that some statistical test remained inconclusive when it comes to the comparison between Chebyshev and Hermite. Therefore, their performances are almost identical.

1 Ridge regression [40–43] generates better results in terms of accuracy than Moore-Penrose pseudoinverse. Therefore: Ridge regression > Moore-Penrose pseudoinverse. It is important to stress out that authors in [11] have drawn the same conclusion when they tested RVFL performance.
2 Both Sign test and Wilcoxon signed rank test failed to reject the null hypothesis of equality when comparing non-iterative and iterative learning. This shows that the non-iterative and iterative learning have similar performance; there are no evidence that any of these two learning approaches is better when we compare their accuracy.
3 Although the OPE-RVFL has higher computational complexity than ELM and RVFL, it shows the reasonable usability when it comes to the on-line processing of data. On some datasets, it even outperformed both ELM and RVFL in terms of testing time.

Extensions of this approach would be in these research directions:

1. How can we find optimal (with respect to user defined criterion of optimality) number of neurons and degree of expansion into orthogonal polynomial?
2. Would additional nonlinear layers enhance the representation capability of OPE-RVFLNN as it does in deep learning? How can we utilize OPE-RVFLNN and deep learning technology to get network with ability to extract and learn features from data? The recent research result given in [30] shows that this approach offers promising insights and directions.
3. Would OPE-RVFLNN network benefit from drop-out approach [50]? How would research effort in [30] perform with drop-out and OPE-RVFL implementation?
4. In real time engineering applications we need to have a system robust to outliers [3], able to detect them and process without stopping, so the research direction would be to enable processing of outliers as data are coming;

These guidelines will be in focus of the forthcoming research, especially in terms of visual tracking and recognition [30].

Funding

This research is financed by Serbian Government under grant TR35004 (2010–2017). Their support is gratefully acknowledged.

Acknowledgments

We would like to sincerely express our gratitude to Guest Editor of this special issue for constant support and understanding he has given us in the review process. We would like to acknowledge anonymous reviewers as well, because their constructive criticism, helpful insights and expert guidance significantly helped us improve initial draft of the paper.

References

[1] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (2) (1991) 251–257.
[2] Z. Miljković, N. Vuković, M. Mitić, Neural extended Kalman filter for monocular SLAM in indoor environment, Proc. Inst. Mech. Eng. Part C: J. Mech. Eng. Sci. 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 Netw. 63 (2015) 31–47.
[4] N. Vuković, Z. Miljković, A growing and pruning sequential learning algorithm of hyper basis function neural network for function approximation, Neural Netw. 46 (2013) 210–226.
[5] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Netw. 61 (2015) 85–117.
[6] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[7] Z. Miljković, D. Aleksendrić, Artificial Neural Networks—Solved Examples with Theoretical Background, University of Belgrade-Faculty of Mechanical Engineering, 2009 (In Serbian).
[8] W.F. Schmidt, M.A. Kraaijveld, R.P.W. Duin, Feedforward neural networks with random weights, Proceedings of the 11th IEEE International Conference on Pattern Recognition (1992).
[9] S. Dehuri, S.B. Cho, A comprehensive survey on functional link neural networks and an adaptive PSO-BP learning for CFLNN, Neural Comput. Appl. 19 (2) (2010) 187–205.
[10] M.F. Amin, R. Savitha, M.I. Amin, K. Murase, Orthogonal least squares based complex-valued functional link network, Neural Netw. 32 (2012) 257–266.
[11] L. Zhang, P.N. Suganthan, A comprehensive evaluation of random vector functional link networks, Inf. Sci. 367–368 (C) (2016) 1094–1105, http://dx.doi.org/10.1016/j.ins.2015.09.025.
[12] 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.
[13] H.T. Braake, G.V. Straten, Random activation weight neural net (RAWN) for east non-iterative training, Eng. Appl. Artif. Intell. 8 (1) (1995) 71–80.
[14] D. Comminiello, S. Scardapane, M. Scarpiniti, P. Parisi, A. Uncini, Functional link expansions for nonlinear modeling of audio and speech signals, 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, Inf. Sci. 313 (2015) 62–78.
[16] Y. Ren, P.N. Suganthan, N. Srikanth, G. Amaratunga, Random vector functional link network for short-term electricity load demand forecasting, Inf. Sci. 367–368 (C) (2016) 1078–1093, http://dx.doi.org/10.1016/j.ins.2015.11.039.
[17] S. Scardapane, D. Wang, M. Panella, A. Uncini, Distributed learning for random vector functional-link networks, Inf. Sci. 301 (2015) 271–284.
[18] L. Zhang, P.N. Suganthan, A survey of randomized algorithms for training neural networks, Inf. Sci. (2016) 146–155, http://dx.doi.org/10.1016/j.ins.2016.01.039.
[19] D. Mesquita, J. Gomes, L. Rodrigues, S. Oliveira, R. Galvão, Building selective ensembles of randomization based neural networks with the successive projections algorithm, Appl. Soft Comput. (2017), http://dx.doi.org/10.1016/j.asoc.2017.08.007.
[20] L. Tang, W. Yao, Y. Lean, A non-iterative decomposition-ensemble learning paradigm using RVFL network for crude oil price forecasting, Appl. Soft Comput. (2017), http://dx.doi.org/10.1016/j.asoc.2017.02.013.
[21] M. Alhamdoosh, D. Wang, Fast decorrelated neural network ensembles with random weights, Info. Sci. 264 (2014) 104–117.
[22] S. Purwar, I.N. Kar, A.N. Jha, On-line system identification of complex systems using Chebyshev neural networks, Appl. Soft Comput. 7 (1) (2007) 364–372.
[23] 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, Appl. Soft Comput. 13 (7) (2013) 3324–3334.
[24] J.C. Patra, P.H. Meher, G. Chakraborty, Nonlinear channel equalization for wireless communication systems using Legendre neural networks, Signal Process. 89 (11) (2009) 2251–2262.
[25] W.D. Weng, C.S. Yang, R.C. Lin, A channel equalizer using reduced decision feedback Chebyshev functional link artificial neural networks, Inf. Sci. 177 (13) (2007) 2642–2654.
[26] 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, Inf. Sci. 181 (17) (2011) 3677–3692.
[27] 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, Appl. Soft Comput. 23 (2014) 51–60.
[28] S. Dehuri, S.B. Cho, Evolutionarily optimized features in functional link neural network for classification, Expert Syst. Appl. 37 (6) (2010) 4379–4391.
[29] S. Dehuri, R. Roy, S.B. Cho, A. Ghosh, An improved swarm optimized functional link artificial neural network (ISO-FLANN) for classification, J. Syst. Softw. 85 (6) (2012) 1333–1345.
[30] L. Zhang, P.N. Suganthan, Visual tracking with convolutional random vector functional link network, IEEE Trans. Cybern. (2016) 1–11.
[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: a large-scale hierarchical image database, Proceedings of 2009 IEEE Conference on Computer Vision and Pattern Recognition (2009) 248–255.
[32] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.
[33] L.P. Wang, C.R. Wan, Comments on 'The extreme learning machine', IEEE Trans. Neural Netw. 19 (2008) 1494–1495.
[34] 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 Evol. Comput. 1 (1) (2011) 3–18.
[35] 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, Inf. Sci. 180 (10) (2010) 2044–2064.
[36] 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 Syst. Appl. 36 (4) (2009) 7798–7808.
[37] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[38] M. Abramowitz, I.A. Stegun, Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables, vol. 55, Courier Corporation, 1964.
[39] N. Vuković, Machine Learning of Intelligent Mobile Robot Based on Artificial Neural Networks, Doctoral Dissertation, University of Belgrade–Faculty of Mechanical Engineering, 2012 (in Serbian).
[40] G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning, Springer, New York, 2013.
[41] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning Theory, CRC Press, 2001 (10th repriniting 2013).
[42] C.P. Chen, J.Z. Wan, A rapid learning and dynamic stepwise updating algorithm for flat neural networks and the application to time-series prediction, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 29 (1999) 62–72.
[43] H. Li, C.P. Chen, H.-P. Huang, Fuzzy Neural Intelligent Systems: Mathematical Foundation and the Applications in Engineering, CRC Press, 2010.
[44] K. Bache, M. Lichman, UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, 2017, http://archive.ics.uci.edu/ml, (Accessed 22 July 2016).
[45] L. Torgo, Regression DataSets, http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html, (Accessed 2 July 2016).

[46] H. A. Guvenir I. Uysal, Function Approximation Repository, http://funapp.cs.bilkent.edu.tr/DataSets/, (Accessed 2 July 2016).
[47] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, CRC Press, 2003.
[48] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Am. Stat. Assoc. 32 (200) (1937) 675–701.
[49] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Ann. Math. Stat. 11 (1) (1940) 86–92.
[50] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.