Artificial neural networks with an infinite number of nodes

IOP Conf. Series: Journal of Physics: Conf. Series **915** (2017) 012006 doi:10.1088/1742-6596/915/1/012006

Artificial neural networks with an infinite number of nodes

K Blekas and I E Lagaris

Department of Computer Science and Engineering University of Ioannina, Ioannina 45500, Greece

E-mail: kblekas@cs.uoi.gr, lagaris@cs.uoi.gr

Abstract. A new class of Artificial Neural Networks is described incorporating a node density function and functional weights. This network containing an infinite number of nodes, excels in generalizing and possesses a superior extrapolation capability.

1. Introduction

Artificial Neural Networks (ANNs) are known to be universal approximators [1, 2] and naturally have been applied to both regression and classification problems. Usual types are the so called feed-forward "Multi-Layered Perceptrons (MLP)" and the "Radial Basis Function (RBF) networks". The usefulness of ANNs is indisputable and there is a vast literature describing applications in different fields, such as pattern recognition and data fitting [3, 4], the solution of ordinary and partial differential equations [5, 6, 7], stock price prediction [8], etc. ANNs can learn from existing data the underlying law (i.e. a function) that produced the dataset. The network parameters, commonly referred to as "weights", are adjusted by a "training" procedure so as to represent as closely as possible the generating function. When the data are outcomes of experimental measurements, they are bound to be corrupted by systematic errors and random noise, a fact that further complicates the "learning" task.

"Over-training" is a phenomenon where, although a network may fit a given dataset accurately, it fails miserably when used for interpolation. To evaluate the performance of a network, the dataset is usually split into the "Training-set" and the "Test-set". The ANN is trained using the former, and it is evaluated by examining its interpolation capability on the latter. If the interpolation is satisfactory, the network is said to "generalize" well, and hence it may be trusted for further use. Over-trained networks do not generalize well. An empirical observation is that given two networks with similar performance on the training set, the one with fewer parameters is likely to generalize better. Several techniques have been developed aiming to improve the network's generalization; for instance "weight decay" and "weight bounding" [9, 10], "network pruning" [11], etc. These methods make use of an additional set, the validation set, i.e. they partition the data into three subsets; the training, the validation and the test subset. Training may yield in several candidate models (i.e. networks with different weight values). Then the validation set is used to pick the best performing model, and finally the test set will give an estimate for the size of the expected error on new data.

Another framework for modeling is through sparse supervised learning models [4, 12] which utilize only a subset of the training data, discarding unnecessary samples based on certain

criteria. Sparse models widely used are, among others, the Lasso [13], the Support Vector Machines (SVMs) [14] and the Relevance Vector Machines (RVMs) [15]. Sparse learning applies either L_1 regularization, leading to penalized regression schemes [13], or sparse priors on the model parameters under the Bayesian framework [15, 16, 17, 18]. Quite recently, Toronto's "Dropout" technique [19] has appeared, stressing the importance of the subject and indicating the current keen interest of the scientific community. This approach employs a stochastic procedure for node removal during the training phase, thus reducing the number of model parameters, in an effort to avoid over-training.

In the present article we introduce a new type of neural network the FWNN, with weights that depend on a continuous variable, instead of the traditional discrete index. This network employs a small set of adjustable parameters and at the same time presents superior generalization performance as indicated by an ample set of numerical experiments. In addition to the interpolation capability we studied the extrapolation properties of FWNN, found to be quite promising. Presently, and in order to illustrate the new concept, we focus on particular forms of the involved weight and kernel functions, which however are not restrictive by any means. The idea of replacing the indexed parameters with continuous functions has been previously considered in [20], in a different setting and with rather limited implementation prospects. However since then, no further followups have been spotted in the literature.

In section 2, we introduce the proposed neural network with continuous weight functions, by associating it to an ordinary radial basis function network and presenting the process of the transition to the continuum. In section 3, we report comparative outcomes of numerical experiments conducted on both homemade datasets and on well known benchmarks established in the literature. Finally, in section 4, we summarize the strengths of the method, and consider different architectural choices that may become the subject of future research.

2. A new type of neural network

Radial basis functions (RBF) are known to be suitable for function approximation and multivariate interpolation [21, 22]. An RBF network with K Gaussian activation nodes, may be written as:

$$N_{RBF}(\mathbf{x};\theta) = w_0 + \sum_{j=1}^{K} w_j \phi(\mathbf{x};\mu_j,\sigma_j) = w_0 + \sum_{j=1}^{K} w_j \exp\left(-\frac{|\mathbf{x}-\mu_j|^2}{2\sigma_j^2}\right) \tag{1}$$

where $\mathbf{x}, \mu_j \in R^n$ and $\theta = \{w_j, \mu_j, \sigma_j\}_{j=0}^{K}$ collectively denote the network parameters to be determined via the training procedure. The total number of adjustable parameters is given by the expression

$$N_{var}^{RB} = K(2+n)+1 \tag{2}$$

which grows linearly with the number of network nodes.

Consider a dataset $S = \{\mathbf{x}^{(l)}, f^{(l)}\}$, where $f^{(l)}$ is the desired output for the corresponding input $\mathbf{x}^{(l)}$, and let $T \subset S$ be a subset of S. The interpolating RBF network is then determined by minimizing the mean squared deviation over T:

$$E_{[T]}(\theta) \stackrel{\text{def}}{=} \frac{1}{\#T} \sum_{\mathbf{x}^{(l)}, f^{(l)} \in T} \left(N_{RBF}(\mathbf{x}^{(l)};\theta) - f^{(l)}\right)^2 \tag{3}$$

Let $\widehat{\theta} = \{\widehat{w}_j, \widehat{\mu}_j, \widehat{\sigma}_j\}_{j=1}^{K}$ be the minimizer of $E_{[T]}(\theta)$, i.e.

$$\widehat{\theta} = \arg\min_{\theta}\{E_{[T]}(\theta)\} \ . \tag{4}$$

The network's generalization performance is measured by the mean squared deviation, $E_{[S-T]}(\widehat{\theta})$, over the relative complement set $S-T$. In the neural network literature, T is usually referred to as the *"training"* set, while $S-T$ as the *"test"* set. A well studied issue is the proper choice for K, the number of nodes.

The training "error" $E_{[T]}(\widehat{\theta})$, is a monotonically decreasing function of K, while the test "error" $E_{[S-T]}(\widehat{\theta})$, is not. Hence we may encounter a situation where adding nodes, in an effort to reduce the training error, will result in increasing the test error, spoiling therefore the network's generalization ability. This is what is called "over-fitting" or "over-training", which is clearly undesirable. An analysis of this phenomenon, coined under the name "*bias-variance dillema*" may be found in [23]. Over-fitting is a serious problem and considerable research effort has been invested to find ways to deter it, leading to the development of several techniques such as model selection, cross-validation, early stopping, weight decay, and weight pruning [4, 12, 23, 24].

2.1. Functionally Weighted Neural Network (FWNN)

We propose a new type of neural network by considering an infinite number of hidden nodes. This is facilitated by introducing a node density function $\rho(s)=\frac{1}{1-s^2}$ of a continuous variable $s\in[-1,1]$ with diverging integral $\int_{-1}^{1}\rho(s)ds\to\infty$. We define the "Functionally Weighted Neural Network" in correspondence to (1) as:

$$N_{FW}(\mathbf{x};\theta)=\int_{-1}^{1}\frac{ds}{1-s^2}\tilde{w}(s)\exp\left(-\frac{|\mathbf{x}-\mu(s)|^2}{2\sigma^2(s)}\right)\ , \tag{5}$$

by applying the following transitions:

$$w_j\longrightarrow\tilde{w}(s),\ \mu_j\longrightarrow\mu(s),\ \sigma_j\longrightarrow\sigma(s),\ \sum_{j=1}^{K}\longrightarrow\int_{-1}^{1}\frac{ds}{1-s^2} \tag{6}$$

In order to use the Gauss–Chebyshev quadrature, we reset $\tilde{w}(s)=\sqrt{1-s^2}w(s)$ and the network becomes:

$$N_{FW}(\mathbf{x};\theta)=\int_{-1}^{1}\frac{ds}{\sqrt{1-s^2}}w(s)\exp\left(-\frac{|\mathbf{x}-\mu(s)|^2}{2\sigma^2(s)}\right) \tag{7}$$

The weight model-functions $w(s),\mu(s)$ and $\sigma(s)$ are parametrized and these parameters are collectively denoted by θ. We examined polynomial forms, i.e:

$$w(s)=\sum_{j=0}^{L_w}w_j s^j,\ \mu(s)=\sum_{j=0}^{L_\mu}\mu_j s^j\,,\text{and}\quad \sigma(s)=\sum_{j=0}^{L_\sigma}\sigma_j s^j\,. \tag{8}$$

Note that $\mu_j\ \forall j=0,\cdots,L_\mu$ and $\mu(s)$ are vectors in R^n. Therefore, the set of adjustable parameters becomes:

$$\theta=\{\{w_j\}_{j=0}^{L_w},\ \{\mu_{ij}\}_{i=1,j=0}^{n,\ L_\mu},\ \{\sigma_j\}_{j=0}^{L_\sigma}\} \tag{9}$$

with a total parameter number given by:

$$N_{var}^{FW}=(1+L_w)+n(L_\mu+1)+(L_\sigma+1)=L_w+nL_\mu+L_\sigma+n+2 \tag{10}$$

The mean squared deviation corresponding to Eq. (3) is given by:

$$E_{[T]}(\theta)\stackrel{\text{def}}{=}\frac{1}{\#T}\sum_{\mathbf{x}^{(l)},f^{(l)}\in T}\left(N_{FW}(\mathbf{x}^{(l)};\theta)-f^{(l)}\right)^2 \tag{11}$$

The above quantity serves as the objective function for the optimization procedure. Since $E_{[T]}(\theta)$ is continuously differentiable, Newton or Quasi-Newton methods are appropriate. Also, since such an objective function is expected to possess a multitude of local minima, a global procedure is necessary as usual [25].

2.2. Optimization strategy
Estimating the model parameters is accomplished by minimizing the mean squared error $E_{[T]}(\theta)$. Since this objective is multimodal, a global optimization technique known as "multistart" has been employed. This is a two-phase method, consisting of an exploratory global phase and a subsequent local minimum-seeking phase. In our case, the search during the local phase is performed via a Quasi-Newton approach with the BFGS update [26]. Since the network is expressed in a closed form, second derivatives may also be computed if a Newton method is to be preferred instead. In Multistart, a point θ is sampled uniformly from within the feasible region, $\theta \in S$, and subsequently a local search $\mathcal{L}$, is started from it leading to a local minimum $\widehat{\theta} = \mathcal{L}(\theta)$. If $\widehat{\theta}$ is a minimum found for the first time, it is stored, otherwise, it is rejected. The cycle goes on until a stopping rule [27] instructs termination.

3. Experimental results
We have performed a series of numerical experiments in order to test the performance of FWNN, and compare it to that of its predecessors, namely the sigmoid MLP and Gaussian RBF networks Both "homemade", as well as established benchmarks from the literature, have been employed. The homemade datasets were constructed by evaluating known functions at a number of selected points. Each set is divided into a training set and a test set. The training sets have been further manipulated by adding noise to the target values, in order to simulate the effect of "corruption" due to measurement errors, while the test sets have been left "clean", i.e. without any noise addition. The training of the FWNN is performed simply by minimizing the mean squared deviation given by Eq.(11), while for the MLP and RBF networks we minimize the corresponding "error" given in Eq.(3) with the addition of a penalty term, to facilitate the necessary "weight decay" regularization. Without regularization the MLP and RBF networks are susceptible to over-training. Since the mission of the network is to filter out the noise and reveal the underlying generating function, we have the luxury, for the homemade datasets, to train using noise-corrupted data, and evaluate the test error over clean data.

In the experiments, 10-fold cross-validation was adopted for the training. To quantify and evaluate network performance, the "Normalized Mean Squared Error" (NMSE) over the test set $S - T$, was used, namely:

$$NMSE = \frac{\sum\limits_{x^{(l)}, f^{(l)} \in S-T} \left(N_{RBF}(\mathbf{x}^{(l)}; \widehat{\theta}) - f^{(l)} \right)^2}{\sum\limits_{x^{(l)}, f^{(l)} \in S-T} \left(f^{(l)} \right)^2} \times 100 \, . \tag{12}$$

Note that the above metric is proper for comparisons, being almost insensitive to data scaling.

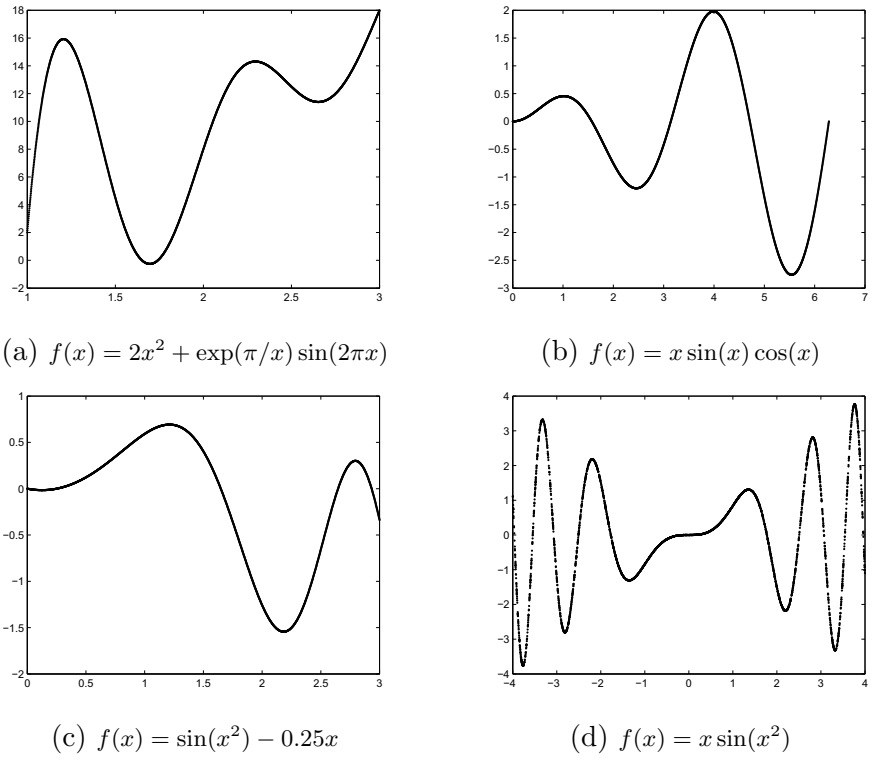
3.1. Experiments using artificial data
We conducted experiments with one- and two-dimensional data. Four functions were employed in one dimension and three in two dimensions. Graphical representations may be found in figures 1(a-d) and 2(a-c) correspondingly.

3.1.1. One-dimensional experiments For each function $f(x)$, an interval $[a, b]$ was chosen for the range of x values as shown in figures 1(a-d). The training set T, contains $L = 100$ equidistant points, determined by:

$$x_i = a + (i-1)\frac{b-a}{L-1}, \ \forall i = 1, 2, \cdots, L \tag{13}$$

The associated target values were then calculated as:

$$f_i = f(x_i) + \eta_i$$



(a) $f(x) = 2x^2 + \exp(\pi/x)\sin(2\pi x)$ (b) $f(x) = x\sin(x)\cos(x)$

(c) $f(x) = \sin(x^2) - 0.25x$ (d) $f(x) = x\sin(x^2)$

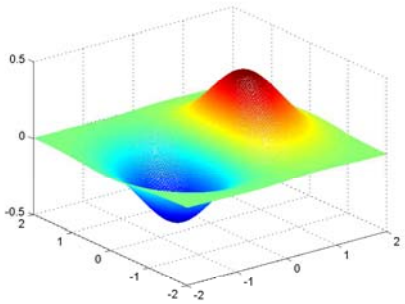
Figure 1. Generating functions used for creating the $1d$ datasets. In each case 100 training and 1000 testing points were used.

with η_i being white Gaussian noise. Three levels of Signal-to-Noise-Ratio (SNR) were considered: no noise ($0\ dB$), medium ($-5\ dB$) and high noise ($-10\ dB$). Similarly the test set was created by picking $L = 1000$ equidistant points using eq. (13). No noise was added to the corresponding target values.

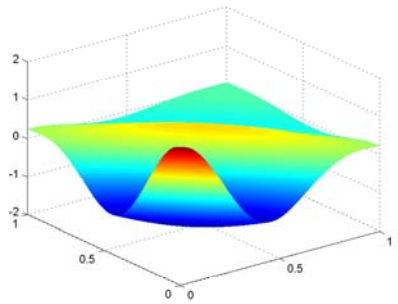
3.1.2. Two-dimensional experiments For each function $f(x_1, x_2)$, intervals $[a_1, b_1] \otimes [a_2, b_2]$ were chosen for the range of $x_1 \otimes x_2$ as shown in figures 2(a-c). For the training set, uniform grids of 15 points in each direction were used, resulting to a total of 225 training points. Again, we have added noise at three levels to the target values. Finally, for the test set, uniform grids of 100 points in each direction were used, resulting to a total of 10000 testing points (without any noise).

3.1.3. Procedural details FWNN, MLP and RBF networks have been investigated using the same datasets. For each noise level, 50 independent runs were executed and the corresponding mean NMSE value and its standard deviation are reported for both the training and the test sets. For the FWNN we used throughout the following values: $L_w = 5,\ L_\mu = 1$ and $L_\sigma = 1$, i.e. a fifth order polynomial for $w(s)$ and first order polynomials for $\mu(s)$ and $\sigma(s)$, corresponding to a total of $2n + 8$ parameters. For the case of MLP and RBF networks, four architectures, differing in the number of nodes, were used. In particular the instances of $K = 5, 10, 20$ and 30 nodes have been investigated, corresponding to $K(n + 2) + 1$ parameters.

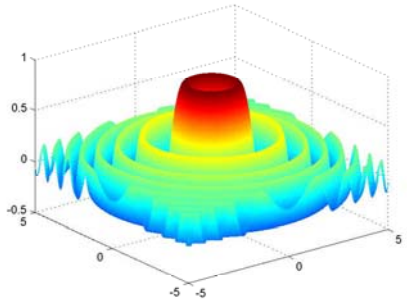
In table 1, the results for the mean NMSE and its standard deviation, estimated over fifty independent runs, are laid out for the $1d$-examples. Accordingly, the results for the $2d$-examples



(a) $f(x_1, x_2) = x_1 \exp(-(x_1^2 + x_2^2))$

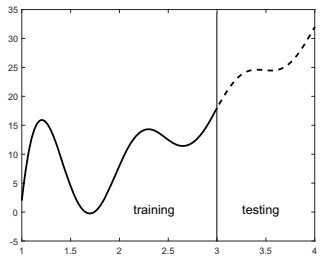


(b) $f(x_1, x_2) = \frac{\pi}{2} \exp(-2(x_1^2 + x_2^2)) \cos(2\pi(x_1 + x_2))$

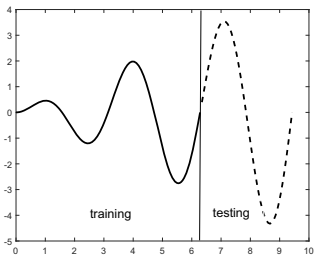


(c) $f(x_1, x_2) = \frac{\sin(x_1^2+x_2^2)}{\sqrt{x_1^2+x_2^2}}$

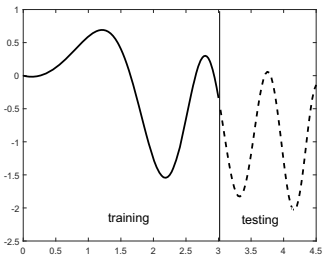
Figure 2. Generating functions used for creating the $2d$ datasets. In each case 225 training and 10000 testing points were used.



Dataset of figure 1a



Dataset of figure 1b



Dataset of figure 1c

Figure 3. The 3 datasets were created using three first generating functions shown in figure 1. They contain 150 equidistant points used for evaluating the extrapolation capabilities of the networks. The training sets (continuous lines) contain 100 points and the extrapolation test sets (dotted lines) 50 points.

are listed in table 2. All networks perform similarly for the $1d$ case, however FWNN has a clear edge in the case of highly corrupted data. This advantage becomes even more evident in the $2d$ case, where FWNN employs only 12 parameters to complete the task, yielding at the same time a superior level of generalization.

Table 1. NMSE mean and std comparison, for the $1d$ datasets related to figures 1(a-d), calculated over 50 independent experiments. The number in brackets is the number of parameters.

Network Architecture		*without noise*		*medium noise*		*high noise*	
		train	*test*	*train*	*test*	*train*	*test*
dataset 1(a)							
FWNN (10)		0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	5.57 ± 0.81	0.89 ± 0.58	16.95 ± 1.01	$\mathbf{1.47 \pm 0.59}$
NN_5	MLP	0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	5.98 ± 0.92	$\mathbf{0.86 \pm 0.45}$	15.88 ± 2.54	2.59 ± 0.83
(16)	RBF	0.15 ± 0.11	0.12 ± 0.08	6.45 ± 0.92	0.95 ± 0.54	17.56 ± 2.70	3.46 ± 2.33
NN_{10}	MLP	0.00 ± 0.00	0.00 ± 0.00	6.46 ± 0.79	0.93 ± 0.39	15.88 ± 3.67	2.45 ± 1.27
(31)	RBF	0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	6.01 ± 1.11	1.71 ± 1.51	13.87 ± 1.86	4.19 ± 1.30
NN_{20}	MLP	0.00 ± 0.00	0.00 ± 0.00	5.66 ± 0.65	1.46 ± 1.19	15.67 ± 1.53	2.61 ± 0.83
(61)	RBF	0.00 ± 0.00	0.00 ± 0.00	5.13 ± 0.98	1.80 ± 1.72	11.70 ± 2.89	8.29 ± 6.24
NN_{30}	MLP	0.00 ± 0.00	0.00 ± 0.00	5.79 ± 1.47	1.75 ± 1.22	14.25 ± 2.66	4.46 ± 2.78
(91)	RBF	0.00 ± 0.00	0.00 ± 0.00	5.67 ± 1.10	1.29 ± 0.68	16.27 ± 3.17	2.90 ± 1.11
dataset 1(b)							
FWNN (10)		0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	12.53 ± 1.61	$\mathbf{2.07 \pm 1.25}$	25.94 ± 3.67	$\mathbf{4.23 \pm 1.64}$
NN_5	MLP	0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	12.40 ± 1.67	2.50 ± 0.98	28.04 ± 7.16	5.49 ± 3.37
(16)	RBF	0.18 ± 0.11	0.27 ± 0.17	12.59 ± 2.30	2.60 ± 1.21	24.19 ± 3.48	5.40 ± 2.30
NN_{10}	MLP	0.00 ± 0.00	0.00 ± 0.00	12.37 ± 1.67	2.13 ± 1.17	25.00 ± 2.57	4.40 ± 1.22
(31)	RBF	0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	10.84 ± 3.04	2.69 ± 1.38	25.32 ± 4.22	7.86 ± 4.55
NN_{20}	MLP	0.00 ± 0.00	0.00 ± 0.00	11.49 ± 0.91	3.19 ± 1.46	26.48 ± 4.57	6.74 ± 2.67
(61)	RBF	0.00 ± 0.00	0.00 ± 0.00	7.80 ± 1.69	5.99 ± 1.65	24.30 ± 3.83	6.73 ± 2.44
NN_{30}	MLP	0.00 ± 0.00	0.00 ± 0.00	11.05 ± 2.06	5.66 ± 2.87	23.80 ± 3.75	10.88 ± 5.15
(91)	RBF	0.00 ± 0.00	0.00 ± 0.00	9.29 ± 2.39	5.34 ± 2.28	21.96 ± 3.98	8.85 ± 3.49
dataset 1(c)							
FWNN (10)		0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	32.52 ± 3.56	$\mathbf{3.52 \pm 2.09}$	54.75 ± 2.84	$\mathbf{11.11 \pm 3.51}$
NN_5	MLP	0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	33.20 ± 5.42	7.50 ± 3.05	57.74 ± 7.01	15.20 ± 6.71
(16)	RBF	0.08 ± 0.06	0.10 ± 0.10	32.40 ± 3.75	6.72 ± 1.12	50.93 ± 4.72	15.60 ± 11.58
NN_{10}	MLP	0.00 ± 0.00	0.00 ± 0.00	32.85 ± 4.65	5.55 ± 3.60	55.08 ± 7.36	16.02 ± 8.98
(31)	RBF	0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	26.58 ± 4.53	9.59 ± 3.78	55.41 ± 6.89	16.13 ± 9.00
NN_{20}	MLP	0.00 ± 0.00	0.00 ± 0.00	29.74 ± 5.84	9.88 ± 6.02	53.98 ± 8.19	16.37 ± 8.53
(61)	RBF	0.00 ± 0.00	0.00 ± 0.00	21.97 ± 2.60	17.66 ± 7.50	51.80 ± 6.66	25.50 ± 14.95
NN_{30}	MLP	0.00 ± 0.00	0.00 ± 0.00	27.13 ± 4.94	13.96 ± 10.68	48.00 ± 6.39	28.45 ± 16.78
(91)	RBF	0.00 ± 0.00	0.00 ± 0.00	15.54 ± 2.19	24.98 ± 7.40	47.76 ± 6.39	22.27 ± 15.76
dataset 1(d)							
FWNN (10)		0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	7.92 ± 1.59	$\mathbf{2.77 \pm 0.93}$	15.31 ± 2.54	$\mathbf{6.80 \pm 0.53}$
NN_5	MLP	95.08 ± 5.23	95.12 ± 5.23	99.03 ± 3.38	99.10 ± 4.29	98.55 ± 1.66	100.02 ± 1.76
(16)	RBF	60.50 ± 12.49	59.22 ± 13.19	61.32 ± 15.14	57.54 ± 16.47	56.35 ± 9.84	52.69 ± 11.09
NN_{10}	MLP	0.05 ± 0.05	0.05 ± 0.02	7.21 ± 0.97	3.83 ± 1.94	17.33 ± 4.21	9.94 ± 6.02
(31)	RBF	0.51 ± 1.58	0.32 ± 0.99	13.22 ± 6.95	7.46 ± 5.15	19.82 ± 5.31	11.76 ± 5.54
NN_{20}	MLP	0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	7.06 ± 1.37	4.04 ± 1.35	16.10 ± 2.98	8.50 ± 2.80
(61)	RBF	0.00 ± 0.00	0.00 ± 0.00	8.60 ± 1.92	5.40 ± 1.82	18.96 ± 8.45	11.22 ± 11.19
NN_{30}	MLP	0.00 ± 0.00	0.00 ± 0.00	5.65 ± 1.31	4.66 ± 2.15	15.16 ± 3.14	12.15 ± 6.11
(91)	RBF	0.00 ± 0.00	0.00 ± 0.00	7.22 ± 2.00	3.29 ± 0.61	14.58 ± 3.31	9.08 ± 3.15

Table 2. NMSE mean and std comparison, for the $2d$ datasets related to figures 2(a-c), calculated over 50 independent experiments. The number in brackets is the number of parameters.

Network Architecture		*without noise*		*medium noise*		*high noise*	
		train	*test*	*train*	*test*	*train*	*test*
dataset 2(a)							
FWNN (12)		0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	46.59 ± 3.04	$\mathbf{2.89 \pm 0.83}$	71.73 ± 2.08	$\mathbf{4.66 \pm 2.19}$
NN_5 (21)	MLP	4.30 ± 0.24	3.82 ± 0.26	51.97 ± 3.94	10.57 ± 2.00	74.88 ± 3.88	20.43 ± 11.11
	RBF	0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	48.63 ± 2.66	3.36 ± 1.16	71.68 ± 2.50	7.42 ± 3.14
NN_{10} (41)	MLP	0.11 ± 0.00	0.09 ± 0.00	49.82 ± 3.41	8.08 ± 1.60	73.13 ± 4.26	15.83 ± 5.00
	RBF	0.00 ± 0.00	0.00 ± 0.00	47.07 ± 2.52	5.37 ± 2.05	71.27 ± 2.76	9.92 ± 5.31
NN_{20} (81)	MLP	0.02 ± 0.00	0.02 ± 0.00	48.72 ± 3.23	7.43 ± 1.92	74.71 ± 3.35	16.65 ± 3.97
	RBF	0.00 ± 0.00	0.00 ± 0.00	45.33 ± 3.21	6.27 ± 2.41	68.74 ± 2.99	13.25 ± 5.96
NN_{30} (121)	MLP	0.02 ± 0.00	0.01 ± 0.00	45.98 ± 3.10	7.11 ± 1.66	71.73 ± 5.28	14.84 ± 3.78
	RBF	0.00 ± 0.00	0.00 ± 0.00	46.19 ± 4.18	6.79 ± 3.26	69.39 ± 2.93	12.11 ± 6.35
dataset 2(b)							
FWNN (12)		0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	51.23 ± 4.69	$\mathbf{5.14 \pm 0.71}$	73.91 ± 4.35	$\mathbf{9.53 \pm 2.59}$
NN_5 (21)	MLP	0.19 ± 0.05	0.17 ± 0.05	51.16 ± 0.66	5.42 ± 2.14	74.34 ± 8.98	19.27 ± 26.57
	RBF	3.19 ± 1.69	2.90 ± 1.58	51.75 ± 3.94	8.79 ± 4.61	72.21 ± 2.67	15.11 ± 3.49
NN_{10} (41)	MLP	0.01 ± 0.01	0.01 ± 0.00	50.50 ± 2.93	6.60 ± 2.63	70.33 ± 2.87	12.88 ± 6.68
	RBF	0.03 ± 0.01	0.03 ± 0.01	47.61 ± 2.46	6.15 ± 0.76	69.41 ± 2.66	13.37 ± 5.14
NN_{20} (81)	MLP	0.01 ± 0.00	0.00 ± 0.00	49.82 ± 2.80	7.17 ± 2.62	71.81 ± 2.80	11.03 ± 2.81
	RBF	0.00 ± 0.00	0.00 ± 0.00	45.87 ± 2.77	11.12 ± 3.61	68.07 ± 2.53	16.04 ± 4.35
NN_{30} (121)	MLP	0.01 ± 0.00	0.00 ± 0.00	48.64 ± 2.13	6.09 ± 2.09	70.67 ± 2.81	11.57 ± 4.04
	RBF	0.00 ± 0.00	0.00 ± 0.00	45.45 ± 2.83	11.05 ± 3.32	66.68 ± 3.76	24.91 ± 8.30
dataset 2(c)							
FWNN (12)		22.35 ± 1.42	$\mathbf{20.95 \pm 1.18}$	64.43 ± 2.90	$\mathbf{26.76 \pm 2.05}$	84.27 ± 2.85	$\mathbf{63.93 \pm 18.58}$
NN_5 (21)	MLP	96.44 ± 2.32	96.32 ± 2.60	96.95 ± 1.80	96.72 ± 1.79	98.51 ± 1.86	98.23 ± 1.72
	RBF	65.32 ± 4.34	63.69 ± 4.42	83.19 ± 2.17	66.98 ± 4.11	89.61 ± 2.75	72.10 ± 3.48
NN_{10} (41)	MLP	87.89 ± 9.93	87.26 ± 10.36	98.44 ± 1.34	97.82 ± 1.50	97.68 ± 2.46	98.70 ± 2.15
	RBF	62.23 ± 5.85	$60.59 \pm 5.88)$	78.88 ± 2.52	$67.88 \pm 2.89)$	85.54 ± 3.77	81.04 ± 11.50
NN_{20} (81)	MLP	89.16 ± 9.73	88.85 ± 10.03	97.17 ± 2.84	96.86 ± 2.74	97.00 ± 3.22	98.43 ± 1.63
	RBF	47.15 ± 3.15	45.25 ± 3.07	69.85 ± 4.77	67.12 ± 10.16	79.51 ± 6.39	94.86 ± 13.43
NN_{30} (121)	MLP	71.89 ± 2.69	70.90 ± 2.88	95.58 ± 3.54	99.46 ± 3.69	94.79 ± 2.87	91.89 ± 4.07
	RBF	47.15 ± 3.15	40.07 ± 2.22	64.67 ± 4.46	68.29 ± 4.51	74.41 ± 5.84	111.65 ± 20.54

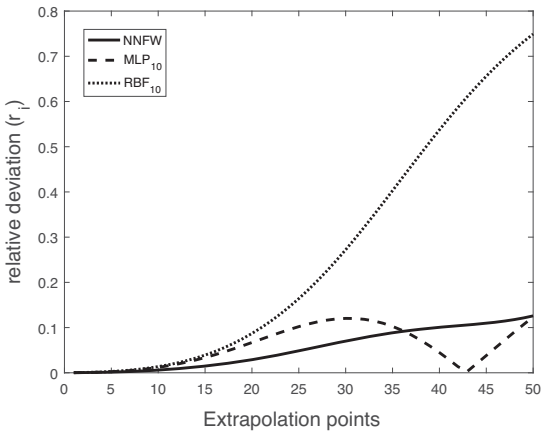
3.2. Extrapolation

FWNN is evaluated as trustworthy as far as interpolation is concerned. It would be quite interesting to explore its extrapolation capability as well. Although this was not the main goal of the present article, we were tempted to examine a few one-dimensional cases to obtain an indication of how its extrapolation capability compares to that of the traditional MLP and RBF networks. We describe briefly the methodology used for the evaluation of the extrapolation performance. We construct a grid of 150 equidistant points. The first 100 points are used for training the networks. The remaining 50 points, $x_1, x_2, \cdots, x_{50}$, are used for evaluating the extrapolation quality. Three such datasets were used as depicted in figure 3.

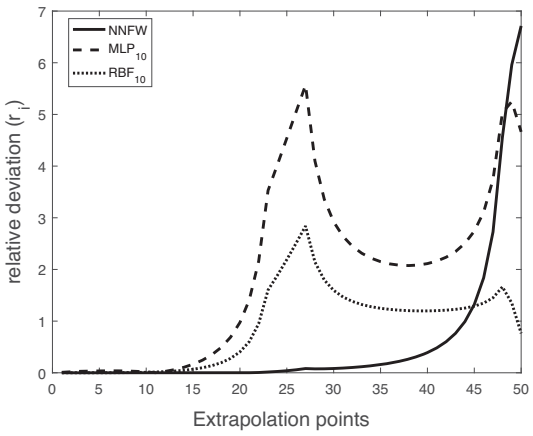
Let $f(x)$ and $N(x)$ denote the generating function and the corresponding trained network; then we may consider as a measure for the network's extrapolation capability, the relative

Table 3. Comparison of the extrapolation index J, for the 3 datasets.

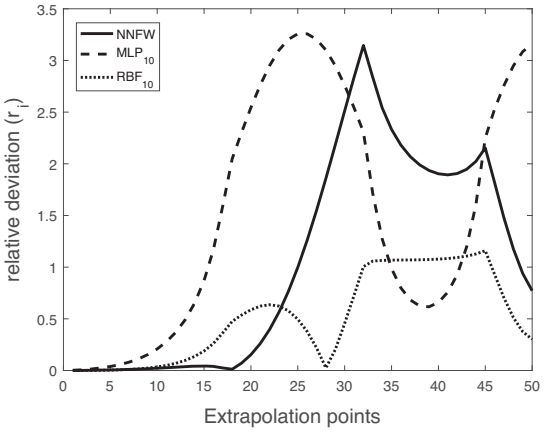
Network Architecture	*Deviation bound d*				
	0.05	0.10	0.15	0.20	0.25
dataset of Fig. 1a					
FWNN	25 ± 1	38 ± 2	50 ± 0	50 ± 0	50 ± 0
MLP 10 nodes	15 ± 2	25 ± 3	31 ± 4	34 ± 5	50 ± 0
RBF 10 nodes	15 ± 2	19 ± 2	22 ± 3	27 ± 5	29 ± 4
dataset of Fig. 1b					
FWNN	24 ± 3	25 ± 3	35 ± 2	37 ± 2	38 ± 2
MLP 10 nodes	12 ± 2	14 ± 2	15 ± 3	16 ± 1	16 ± 1
RBF 10 nodes	14 ± 2	16 ± 2	17 ± 2	18 ± 1	19 ± 1
dataset of Fig. 1c					
FWNN	18 ± 1	20 ± 1	21 ± 1	21 ± 1	21 ± 1
MLP 10 nodes	6 ± 2	8 ± 1	9 ± 1	10 ± 1	11 ± 1
RBF 10 nodes	11 ± 1	13 ± 2	14 ± 2	15 ± 1	16 ± 1



(a) Dataset: figure 1a



(b) Dataset: figure 1b



(c) Dataset: figure 1c

Figure 4. FWNN, MLP and RBF relative-deviations r_i, defined in eq. (14), evaluated at 50 equidistant points in the extrapolation domain, for the quoted datasets.

deviation r_i at point x_i given by:

$$r_i \equiv \frac{|f(x_i) - N(x_i)|}{\max\{1, |f(x_i)|\}} \tag{14}$$

Let J be the maximum number of consecutive points x_i starting from x_1 and satisfying $r_i < d$, where $d \in [0, 0.25]$, is a bound for the relative deviation. Namely J is determined so that:

$$r_i < d, \ \forall \ i \leq J \text{ and } \ r_{J+1} > d$$

Given a value for d, the best network for extrapolation is the one corresponding to the highest J. We list our findings for $d = 0.05, 0.10, 0.15, 0.20, 0.25$ in table 3. The mean values and the standard deviations of the J-index, have resulted from the conduction of 50 independent experiments. In figure 4(a-c) we compare the relative deviations r_i, $\forall i = 1, \cdots, 50$, of the FWNN, MLP and RBF networks, for three quoted datasets. In all cases the proposed FWNN, displays superior extrapolation performance.

4. Discussion and Conclusions

In this study we have proposed a new type of neural network, the FWNN, in which all of its parameters are functions of a continuous variable and not of a discrete index. This maybe interpreted as a neural network with an infinite number of hidden nodes and significantly restricted number of model parameters. In the conducted experiments on a suite of benchmark datasets, FWNN achieved superior generalization performance compared to its peers, namely the conventional MLP and RBF networks. In addition, FWNN is robust and effective even on difficult problems.

The FWNN has interesting properties. The important issue of generalization is being served superbly. The number of necessary parameters is rather limited, resisting so over-training. The positions of the Gaussian centers are determined by the arc $\mu(s), \forall s \in [-1, 1]$ and the corresponding spherical spreads by $\sigma(s)$. In the present article we have used linear forms, hence the arc is a straight line segment joining the two end points $\mu(-1)$ and $\mu(+1)$ in R^n. The spreads are linearly increasing or decreasing with s, depending on the sign of σ_1. Although this seems to be a severe constraint, it has not degraded the network's performance. This may be understood noting that the infinite number of nodes, renders the approximation of any function feasible, while the existence of the constraint, deters over-training by substantially restricting the number of adjustable parameters. If in some cases the linear model for $\mu(s)$ or $\sigma(s)$ imposes an overly strict constraint, it can be replaced by a more flexible quadratic, or cubic model, at the expense of some extra parameters. The Gaussian centers will then be lying on a parabolic, or a cubic arc correspondingly and the spreads will exhibit a higher level of adaptivity.

References

[1] Cybenko G 1989 *Mathematics of Control, Signals, and Systems* 303–314
[2] Hornik K 1991 *Neural Networks* 251–257
[3] Ripley B D 1996 *Pattern Recognition and Neural Networks* (Campridge University Press)
[4] Bishop C 2006 *Pattern Recognition and Machine Learning* (Springer)
[5] Lagaris I E, Likas A and Fotiadis D I 1997 *Computer Physics Communications* 1–14
[6] Lagaris I E, Likas A and Fotiadis D I 1998 *IEEE Transactions on Neural Networks* 987–1000
[7] Lagaris I E, Likas A and Papageorgiou D G 2000 *IEEE Transactions on Neural Networks* **11** 1041–1049
[8] Wang J Z, Wang J J, Zhang Z G and Guo S P 2011 *Expert Systems with Applications* **38** 14346–14355
[9] Bartlett P 1998 *IEEE Transactions on Information Theory* **44** 525–536
[10] MacKay D J C 1992 *Neural Computation* **4** 415–447
[11] Sietsma J and Dow R J 1991 *Neural Networks* **4** 67–79
[12] Murphy K 2012 *Machine Learning: A Probabilistic Perspective* (MIT Press)
[13] Tibshirani R 1996 *Journal of the Royal Statistical Society B* **58** 267–288

[14] Meyer D, Leisch F and Hornik K 2003 *Neurocomputing* **55** 169–186
[15] Tipping M 2001 *Journal of Machine Learning Research* **1** 211–244
[16] Seeger M 2008 *Journal of Machine Learning Research* **9** 759–813
[17] Tzikas D, Likas A and Galatsanos N 2009 *IEEE Trans. on Neural Networks* **20** 926–937
[18] Blekas K and Likas A 2014 *Knowl. Inf. Syst.* **39** 241–264
[19] Srivastav N, Hinton G, Krizhevsky A, Sutskever I and Salakhutdinov R 2014 *Journal of Machine Learning Research* **15** 1929–1958
[20] Roux N and Bengio Y 2007 *Eleventh International Conference on Artificial and Statistics (AISTATS)* pp 404–411
[21] Powell M 1985 *IMA conference on "Algorithms for the Approximation of Functions and Data* (RMCS Shrivenham)
[22] Broomhead D S and Lowe D 1988 *Complex Systems* **2** 321–355
[23] Geman S, Bienenstock E and Doursat R 1992 *Neural Computation* **4** 1–58
[24] Krogh A and Hertz J 1992 *Advances in Neural Information Processing Systems* vol 4 pp 950–957
[25] Voglis C and Lagaris I 2006 *Neural, Parallel & Scientific Computations* **14** 231–240
[26] Fletcher R 1970 *Computer Journal* **13** 317–322
[27] Lagaris I E and Tsoulos I G 2008 *Applied Mathematics and Computation* **197** 622–632