Dear Author,

Here are the proofs of your article.

- You can submit your corrections **online**, via **e-mail** or by **fax**.
- For **online** submission please insert your corrections in the online correction form. Always indicate the line number to which the correction refers.
- You can also insert your corrections in the proof PDF and **email** the annotated PDF.
- For fax submission, please ensure that your corrections are clearly legible. Use a fine black pen and write the correction in the margin, not too close to the edge of the page.
- Remember to note the **journal title**, **article number**, and **your name** when sending your response via e-mail or fax.
- **Check** the metadata sheet to make sure that the header information, especially author names and the corresponding affiliations are correctly shown.
- **Check** the questions that may have arisen during copy editing and insert your answers/ corrections.
- **Check** that the text is complete and that all figures, tables and their legends are included. Also check the accuracy of special characters, equations, and electronic supplementary material if applicable. If necessary refer to the *Edited manuscript*.
- The publication of inaccurate data such as dosages and units can have serious consequences. Please take particular care that all such details are correct.
- Please **do not** make changes that involve only matters of style. We have generally introduced forms that follow the journal's style.
  Substantial changes in content, e.g., new results, corrected values, title and authorship are not allowed without the approval of the responsible editor. In such a case, please contact the Editorial Office and return his/her consent together with the proof.
- If we do not receive your corrections **within 48 hours**, we will send you a reminder.
- Your article will be published **Online First** approximately one week after receipt of your corrected proofs. This is the **official first publication** citable with the DOI. **Further changes are, therefore, not possible.**
- The **printed version** will follow in a forthcoming issue.

Please note

After online publication, subscribers (personal/institutional) to this journal will have access to the complete article via the DOI using the URL: http://dx.doi.org/[DOI].
If you would like to know when your article has been published online, take advantage of our free alert service. For registration and further information go to: http://www.link.springer.com.

Due to the electronic nature of the procedure, the manuscript and the original figures will only be returned to you on special request. When you return your corrections, please inform us if you would like to have these documents returned.

Metadata of the article that will be visualized in OnlineFirst

ArticleTitle	Functionally weighted neural networks: frugal models with high accuracy	
Article Sub-Title		
Article CopyRight	Springer Nature Switzerland AG (This will be the copyright line in the final PDF)	
Journal Name	SN Applied Sciences	

Corresponding Author	Family Name	**Blekas**
	Particle	
	Given Name	**Konstantinos**
	Suffix	
	Division	Department of Computer Science and Engineering
	Organization	University of Ioannina
	Address	45110, Ioannina, Greece
	Phone	
	Fax	
	Email	kblekas@cs.uoi.gr
	URL	
	ORCID	http://orcid.org/0000-0003-1085-8177
Author	Family Name	**Lagaris**
	Particle	
	Given Name	**Isaac E.**
	Suffix	
	Division	Department of Computer Science and Engineering
	Organization	University of Ioannina
	Address	45110, Ioannina, Greece
	Phone	
	Fax	
	Email	lagaris@cs.uoi.gr
	URL	
	ORCID	

	Received	21 February 2020
Schedule	Revised	
	Accepted	14 October 2020

Abstract	In this article, we introduce the “functionally weighted neural network,” a new addition to the rich collection of artificial neural networks. Instead of a finite number of discrete nodes, we consider an infinite number of continuously distributed nodes. The weights assume a functional form, and the sum over the nodes becomes an integral. The gain is a significant reduction in the number of adjustable parameters, accompanied by an enhanced generalization performance. To quantitatively assess the quality of this new network, we have performed numerical experiments on a number of benchmark datasets. Comparison with state-of-the-art techniques reveals the advantages of the proposed method and emphasizes its modeling potential.

Keywords (separated by '-')	Neural networks - Functional weights - Infinite number of nodes - Generalization - Function approximation
Footnote Information	

Research Article

Functionally weighted neural networks: frugal models with high accuracy

Konstantinos Blekas[1] · Isaac E. Lagaris[1]

Received: 21 February 2020 / Accepted: 14 October 2020

Abstract

In this article, we introduce the "functionally weighted neural network," a new addition to the rich collection of artificial neural networks. Instead of a finite number of discrete nodes, we consider an infinite number of continuously distributed nodes. The weights assume a functional form, and the sum over the nodes becomes an integral. The gain is a significant reduction in the number of adjustable parameters, accompanied by an enhanced generalization performance. To quantitatively assess the quality of this new network, we have performed numerical experiments on a number of benchmark datasets. Comparison with state-of-the-art techniques reveals the advantages of the proposed method and emphasizes its modeling potential.

Keywords Neural networks · Functional weights · Infinite number of nodes · Generalization · Function approximation

1 Introduction

Artificial neural networks (ANNs) have proved to be valuable tools in a host of different applications, such as function approximation and data fitting [2], solution of ordinary and partial differential equations [15–17], time-series prediction for the stock market [34], pattern recognition [2, 29], classification [35] and clustering [7], to name a few. ANNs are flexible modeling functions known for their excellent approximation capabilities [6, 10–12, 14] and have been termed "Universal Approximators." ANNs may be designed according to various architectures, the main structural elements being the number of hidden layers, the number of neurons and the type of activation functions. Deep neural networks (DNNs) are ANNs with multiple hidden layers and can model complex mappings between the input and output layers.

ANNs suffer from the issue of overfitting, i.e., producing a model that may be very accurate for a subset of data while failing to account for the rest. In DNNs, the overfitting issue is even more pronounced due to the extra layers that enable the fitting of outliers. Several techniques have been developed to combat overfitting known collectively under the name "Regularization Methods." Examples are node pruning [31], weight decay (or L_2 regularization) [1], weight bounding [20], sparsity (or L_1 regularization) and more recently the "dropout" technique [32], determinantal point processes (DPPs)[22], approximate empirical Bayes methods[37] that may be roughly described as random pruning. ANNs are trained using a so-called training set, and their performance is evaluated using a "test set." Networks that perform well are said to generalize. An overfit/overtrained network obviously does not generalize and therefore cannot be trusted for further use.

In the present article, we introduce a new type of ANN, the "functionally weighted neural network" (FWNN). Single-hidden-layer ANNs may be expressed as a linear combination of a number of parametric basis functions. Common forms are based on the logistic and Gaussian activation functions, namely:

✉ Konstantinos Blekas, kblekas@cs.uoi.gr; Isaac E. Lagaris, lagaris@cs.uoi.gr | [1]Department of Computer Science and Engineering, University of Ioannina, 45110 Ioannina, Greece.

SN Applied Sciences _####################_ | https://doi.org/10.1007/s42452-020-03713-y

Journal : Large 42452	Article No : 3713	Pages : 12	MS Code : 3713	Dispatch : 29-10-2020

$$N_l(\boldsymbol{x};\theta) = w_0 + \sum_{k=1}^{K} \frac{w_k}{1 + exp\left(-(c_k^T \boldsymbol{x} + b_k)\right)} \tag{1}$$

(Logistic MLP)

$$N_G(\boldsymbol{x};\theta) = w_0 + \sum_{k=1}^{K} w_k \exp\left(-\frac{1}{2}\left|\frac{\boldsymbol{x} - \boldsymbol{\mu}_k}{\sigma_k}\right|^2\right) \tag{2}$$

(Gaussian RBF)

where θ, in both cases, stands collectively for the adjustable parameters and K is the number of neural nodes.

Our proposal introduces a neural network that employs a continuous nodal distribution $\rho(s)$, instead of a countable set of discrete nodes. The corresponding functionally weighted expressions for logistic and Gaussian activation functions may be cast as:

$$N_{Fl}(\boldsymbol{x};\theta) = \int \frac{w(s)}{1 + \exp\left(-(c(s)^T \boldsymbol{x} + b(s))\right)} \rho(s) ds \tag{3}$$

$$N_{FG}(;,\theta) = \int w(s) \exp\left(-\frac{1}{2}\left|\frac{\boldsymbol{x} - \boldsymbol{\mu}(s)}{\sigma(s)}\right|^2\right) \rho(s) ds \tag{4}$$

Preliminary results assessing the performance of FWNNs have been reported earlier [3] and have been presented at the Sofianos-2017 international symposium. The substitution of discrete weights by continuous functions has been also considered in [30], where, however, the activation is restricted to be an odd function, and the weights are either piecewise constant or piecewise affine functions. Polynomials have not been considered there, because the integrals involved cannot be expressed in a closed analytic form. To the best of our knowledge, this work has not been followed up.

In Sect. 2, we introduce the proposed neural network with continuous weight functions, by associating it with an ordinary radial basis function (RBF) network and presenting the process of the transition to the continuum. Technical details are given in Sect. 3, about the numerical quadrature, the training optimization methods and the software platforms used. In Sect. 4, we report the results of numerical experiments conducted on simulated homemade datasets as well as on established benchmarks from the literature. Finally, in Sect. 5, we summarize the strengths of the method and pose a few questions that may become the subject of future research.

2 Neural networks with infinite number of hidden units

Radial basis functions are known to be suitable for function approximation and multivariate interpolation [4, 27]. Assuming an n-dimensional input space, $\boldsymbol{x} \in R^n$, an RBF neural network consisting of K Gaussian nodes with parameters $\boldsymbol{\mu}_k \in R^n$ and $\sigma_k \in R$ is given by Eq. (2).

The set $\theta = \{w_0, \left(w_k, \boldsymbol{\mu}_k, \sigma_k\right)_{k=1}^K\}$ denotes collectively the network parameters to be determined via the training procedure. The total number of adjustable parameters is given by the expression

$$N_{var}^{RBF} = K(2+n) + 1 \tag{5}$$

which grows linearly with the number of network nodes. Consider a dataset $S = \{\boldsymbol{x}_i, t_i\}$, where t_i is the desired output (target) for the corresponding input $\boldsymbol{x}_i$. Let also $T \subset S$ be a subset of S with cardinality $\#T$. The approximating RBF network is then determined by minimizing the mean squared deviation over T:

$$E_{[T]}(\theta) \stackrel{\text{def}}{=} \frac{1}{\#T} \sum_{\boldsymbol{x}_i, t_i \in T} \left(N_G(\boldsymbol{x}_i;\theta) - t_i\right)^2 \tag{6}$$

Let $\hat{\theta} = \{\hat{w}_0, \left(\hat{w}_k, \hat{\boldsymbol{\mu}}_k, \hat{\sigma}_k\right)_{k=1}^K\}$ be the minimizer of $E_{[T]}(\theta)$, i.e.,

$$\hat{\theta} = \arg\min_{\theta}\{E_{[T]}(\theta)\}. \tag{7}$$

The network's generalization performance is measured by the mean squared deviation, $E_{[S-T]}(\hat{\theta})$, 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, which denotes the number of nodes in the neural network architecture.

The training "error" $E_{[T]}(\hat{\theta})$ is a monotonically decreasing function of K, while the test "error" $E_{[S-T]}(\hat{\theta})$ is not. Hence, we may encounter a situation where adding nodes, in an effort to reduce the training error, will result in an increase in the test error, spoiling therefore the network's generalization ability. This behavior is known as "overfitting" or "overtraining" and is clearly undesirable. An early analysis of this phenomenon coined under the name "*bias–variance dilemma*" may be found in [9]. Overfitting 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, regularization and weight pruning [2, 9, 13, 24, 25].

Journal : Large 42452 | Article No : 3713 | Pages : 12 | MS Code : 3713 | Dispatch : 29-10-2020

2.1 Functionally weighted neural network

We define the "*functionally weighted neural network*" (FWNN) to be the limit of the conventional ANN, as the number of nodes $K \to \infty$. The set of discrete nodes indexed by an integer (k) is replaced by a nodal distribution $\rho(s)$ that depends on a continuous variable (s). The FWNN may then be cast, in correspondence with Eq. (2), as:

$$N_{FG}(\boldsymbol{x};\theta) = \int_{-1}^{1} \mathrm{d}s\, \rho(s)\, \tilde{w}(s) \exp\left(-\frac{|\boldsymbol{x} - \boldsymbol{\mu}(s)|^2}{2\sigma^2(s)}\right), \tag{8}$$

by applying the following transitions:

$$w_k \longrightarrow \tilde{w}(s) \tag{9a}$$

$$\boldsymbol{\mu}_k \longrightarrow \boldsymbol{\mu}(s) \tag{9b}$$

$$\sigma_k \longrightarrow \sigma(s) \tag{9c}$$

$$\sum_{k=1}^{K} \longrightarrow \int_{-1}^{1} \mathrm{d}s\, \rho(s) \tag{9d}$$

The density function $\rho(s)$ should lead to an infinite number of nodes, i.e.

$$\int_{-1}^{+1} \rho(s)\, ds \to \infty. \tag{10}$$

For the density function, we have chosen the following form that satisfies (10):

$$\rho(s) = \frac{1}{1 - s^2} \tag{11}$$

The weight functions $\tilde{w}(s)$, $\boldsymbol{\mu}(s)$ and $\sigma(s)$ are parametrized, and these parameters are collectively denoted by θ. In this article, we have examined the following functional forms:

$$\tilde{w}(s) \equiv \sqrt{1 - s^2} w(s) = \sqrt{1 - s^2} \sum_{j=0}^{L_w} w_j s^j \tag{12a}$$

$$\boldsymbol{\mu}(s) = \sum_{j=0}^{L_\mu} \boldsymbol{\mu}_j s^j \tag{12b}$$

$$\sigma(s) = \sum_{j=0}^{L_\sigma} \sigma_j s^j \tag{12c}$$

Note that $\boldsymbol{\mu}(s)$ and $\boldsymbol{\mu}_j = \left(\mu_{jl}\right)_{l=1}^{n}$, $j = 0, \ldots, L_\mu$ are vectors in R^n.

The set of adjustable parameters is then represented by:

$$\theta = \{ \left(w_j\right)_{j=0}^{L_w}, \left(\mu_{jl}\right)_{j=0,l=1}^{L_\mu, n}, \left(\sigma_j\right)_{j=0}^{L_\sigma} \} \tag{13}$$

with a total parameter number given by:

$$\begin{aligned} N_{var}^{FW} &= (1 + L_w) + n(L_\mu + 1) + (L_\sigma + 1) = \\ &= L_w + nL_\mu + L_\sigma + n + 2 \end{aligned} \tag{14}$$

The "cost" function $C(\theta)$, is formed by adding a regularization term $R(\theta)$ to the mean squared deviation of Eq. (6),

$$C(\theta) \overset{\text{def}}{=} E_{[T]}(\theta) + R(\theta) \tag{15}$$

$C(\theta)$ serves as the objective function for the optimization task, and from now on, we redefine $\hat{\theta}$ as $\hat{\theta} = \arg\min_\theta \{C(\theta)\}$. For the regularization term $R(\theta)$, the squared Euclidean ($L2$) norm multiplied by a penalty factor has been adopted.

3 Technical details

In this section, we present the numerical methods used in our calculations. Namely, we describe the employed integration technique, the optimization procedure, and we also refer to the relevant software.

Substituting the nodal density from Eq. (11) in Eq. (8) and using Eq. (12a), the FWNN may be rewritten as:

$$N_{FG}(\boldsymbol{x};\theta) = \int_{-1}^{1} \frac{\mathrm{d}s}{\sqrt{1 - s^2}} w(s) \exp\left(-\frac{|\boldsymbol{x} - \boldsymbol{\mu}(s)|^2}{2\sigma^2(s)}\right). \tag{16}$$

3.1 Approximating integrals

Integrals were estimated by the accurate Gauss–Chebyshev quadrature:

$$\int_{-1}^{1} \frac{\mathrm{d}s}{\sqrt{1 - s^2}} g(s) \approx \frac{\pi}{M} \sum_{i=1}^{M} g(s_i), \tag{17}$$

where

$$s_i = \cos\left(\frac{2i - 1}{2M}\pi\right).$$

The above explains our choice for the functional form of $\tilde{w}(s)$ in Eq. (12a). In our experiments, we have used $M = 100$. The number of integration points has been increased up to $M = 200$, without noticing any appreciable difference.

Journal : Large 42452	Article No : 3713	Pages : 12	MS Code : 3713	Dispatch : 29-10-2020

3.2 Learning procedure and software platforms

Determination of the FWNN parameters is accomplished by minimizing the cost function given in Eq. (15). Since objectives of this kind are known to be multimodal, global optimization should be considered. We have employed a simple stochastic global optimization technique known as "*Multistart*" [33]. This is a two-phase method, consisting of an exploratory global phase and a subsequent local minimum-seeking phase.

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 $\hat{\theta} = \mathcal{L}(\theta)$. If $\hat{\theta}$ is a minimum found for the first time, it is stored; otherwise, it is rejected. The cycle goes on until a stopping rule [18] instructs termination. An algorithmic presentation of Multistart is given below:

Simple Multistart Algorithm

1. **Initialize:** Set $k = 1$, sample $\theta \in S$ and set $\hat{\theta}_k = \mathcal{L}(\theta)$
2. If a termination rule applies, set $\hat{\theta} = \hat{\theta}_m$ and stop (note that m is the index with the property: $C(\hat{\theta}_m) = \min_i \{C(\hat{\theta}_i)\}$)
3. **Main iteration:** Sample $\theta \in S$ $\hat{\theta} = \mathcal{L}(\theta)$ If $\hat{\theta} \notin \{\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_k\}$, then $k \leftarrow k + 1$ and $\hat{\theta}_k \leftarrow \hat{\theta}$ Endif
4. Repeat from step 2.

The computer code was written in Python. For the local phase, we have relied on the quasi-Newton framework with the BFGS update, using the weak Wolfe–Powell conditions for the line search, that is contained in Pythons scipy. optimize library.

4 Numerical experiments, comparative analysis and extrapolation performance

A series of numerical experiments was devised for testing the performance of the proposed FWNN, by comparing its outcomes against those obtained by a number of established alternatives. We have considered both homemade simulated datasets and benchmarks that are widely used in the relevant scientific literature.

In our experiments, we have compared FWNN with MLP and RBF networks, as well as with Gaussian processes (GPs). For the neural networks, a host of architectural configurations (created by varying the number of the hidden nodes $K \in [5, 100]$) have been considered. MLPs were trained by the "Limited Memory BFGS" (L-BFGS) method that requires low memory computational resources and has proved to be quite efficient. For the RBFs, the exponential parameters were determined by K-means clustering, while the amplitudes were determined by linear regression. For the Gaussian processes, we have considered RBF kernels with automatic determination of its scalar parameter in the range $[10^{-5}, 10^{5}]$. For the experiments in all cases (MLP, RBF, GPs), the following values for the regularization parameter have been used: $\alpha = \{10^{-10}, 10^{-5}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 0, 10, 10^{2}, 10^{3}, 10^{5}\}$.We have noticed that in some cases the regularization parameter had a significant effect. For every experiment, only the best result of each approach is reported for comparison with the corresponding FWNN outcome. The reason for choosing Gaussian processes in our experimental study is first its modeling potential and second some neural networks become identical to a Gaussian process with a specific type of covariance function in the limit of infinite hidden units [25, 28]. Finally, we have used Python's Scikit-learn library for the implementation of the above three regression methodologies.

4.1 Experiments with simulated datasets

Several datasets were constructed by evaluating a number of selected test functions at preset sets of equidistant points. Four and three test functions have been employed for the 1*d* and for the 2*d* experiments, respectively, with their plots and formulas depicted in Figs. 1a–d and 2a–c.

Each dataset was divided into a training set and a test set. The target values of the training sets have been deliberately "contaminated" by addition of noise. On the other hand, the test sets have been left "clean," i.e., with no noise addition, so that one can make an assessment on the capability of the tested methods to filter out the noise and reveal the underlying function.

In our experiments, we compare the FWNN to the logistic MLP and Gaussian RBF networks with "weight decay"($L2$) regularization. For the evaluation, we use the almost insensitive to data scaling "Normalized Mean Squared Error" (NMSE) over the test set $[S - T]$, namely:

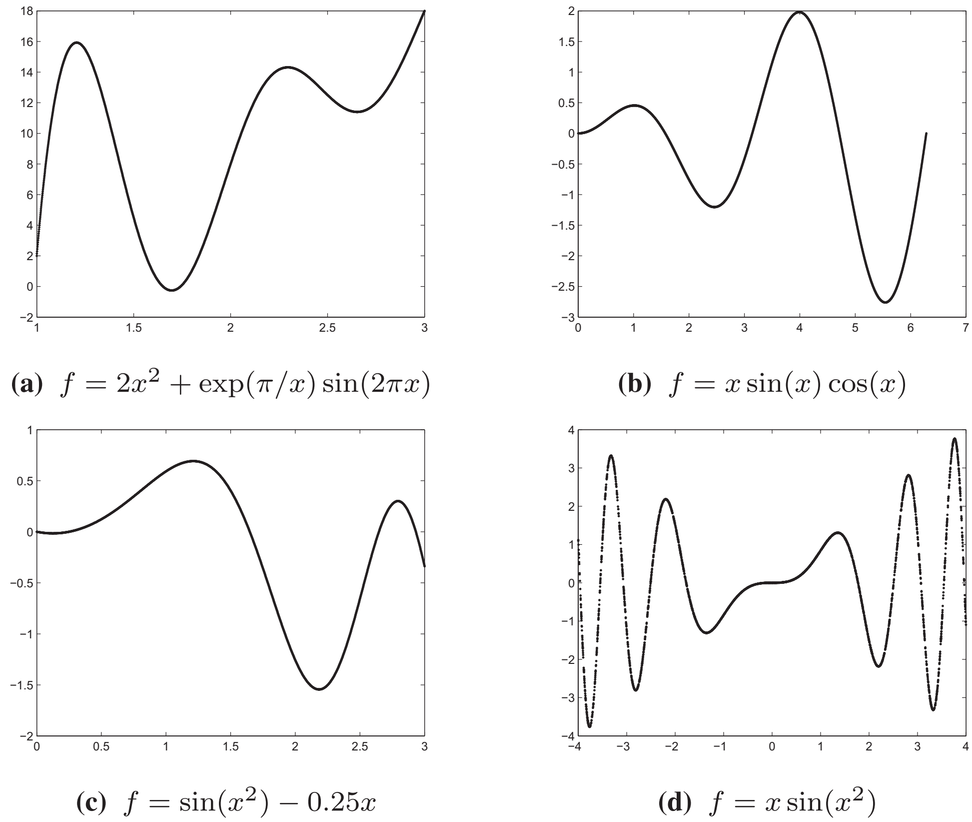
$$\mathrm{NMSE} = \frac{1}{\#[S-T]} \sum_{x_i, t_i \in [S-T]} \left(\frac{N(\boldsymbol{x}_i; \hat{\theta}) - t_i}{t_i} \right)^2 \times 100 \tag{18}$$

The experimental setup for the simulated datasets has been detailed in an earlier publication [3].

Two levels of signal-to-noise ratio were considered for generating the simulated training sets: *medium* ($-5\ dB$), and *large* ($-10\ dB$). For each noise level, 50 independent runs were performed and the corresponding NMSE mean and standard deviation are reported. For the FWNN, we have used throughout the following polynomial degrees:

Journal : Large 42452	Article No : 3713	Pages : 12	MS Code : 3713	Dispatch : 29-10-2020

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



$L_w = 5$, for the polynomial contained in $w(s)$
$L_\mu = 1$, for each of the $\boldsymbol{\mu}(s)$ polynomials
$L_\sigma = 1$, for the $\sigma(s)$ polynomial

As a consequence of the above settings, the total number of the FWNN adjustable parameters equals $2n + 8$.

The results are listed in Tables 1 and 2 for the 1*d* and 2*d* datasets accordingly. Notice that for the MLP and RBF networks, as well as for the Gaussian process, only the results corresponding to the best performing case are listed. By inspection, FWNN's generalization is superior, especially for large noise levels. This advantage becomes even more pronounced in the 2*d* case. While FWNN employs only ten and 12 parameters for the 1*d* and 2*d* datasets, MLP and RBF networks require a significantly larger number in the range $[31 - 301]$ and $[41 - 401]$, respectively, in order to achieve a comparable test error. For these datasets, a plethora of experiments and related results may be found in [3].

Additional experiments were conducted in order to study the generalization performance of the FWNN as a function of the number of network parameters. We have examined a limited number of cases; hence, our results are only indicative, not conclusive. In doing so, we have retained first-degree polynomials for both $\boldsymbol{\mu}(s)$ and $\sigma(s)$ and varied only the degree of the polynomial in $w(s)$. Accordingly, for the MLP and RBF networks, we have varied the number of hidden nodes. Again 50 independent experiments were performed for each case, and the corresponding NMSE mean was calculated. We have selected two artificial datasets, generated by the functions plotted in Figs. 1b and 2b. We have observed that for the FWNN, the dependence of NMSE on the number of parameters was significantly weaker.

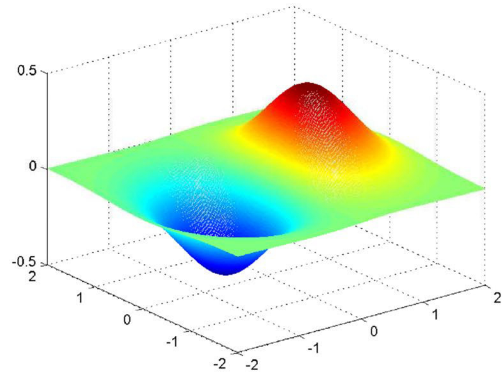
4.2 Extrapolation in one dimension

Consider an 1*d* dataset with points $x_1, x_2, \dots, x_M$ arranged in ascending order, and corresponding targets $y_1, y_2, \dots, y_M$. Let $N(x, \theta)$ be a network trained over the above set. Estimating the target value as $Y = N(X, \theta)$ at a point $X \in (x_j, x_{j+1})$ is called interpolation, while at a point $X \notin [x_1, x_M]$ is called extrapolation. It has been argued in [21] that artificial neural networks extrapolate rather poorly. To study the extrapolation potential of FWNN, the first two test functions of Fig. 1 have been employed, namely:

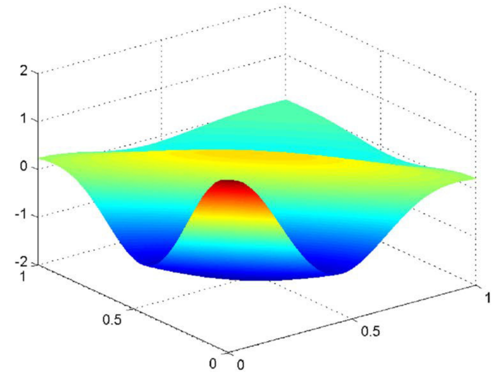
$$f(x) = 2x^2 + \exp(\pi/x)\sin(2\pi x) \text{ and}$$

$$f(x) = x\sin(x)\cos(x),$$

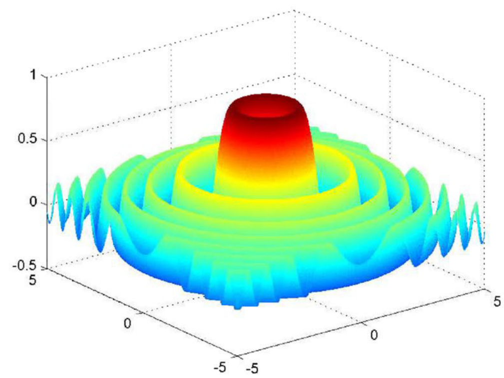
Journal : Large 42452	Article No : 3713	Pages : 12	MS Code : 3713	Dispatch : 29-10-2020



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



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



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

Fig. 2 Generating functions used for creating the 2*d* datasets. In each case, 100 training and 1000 testing points were used

Table 1 Comparison of the NMSE mean over the test set, resulting from 50 independent experiments, for the 1*d* datasets related to Fig. 1a–d

Method	NMSE over the test set			
	Medium noise	High noise	Medium noise	High noise
	Dataset 1(a)		**Dataset 1(b)**	
FWNN	0.63	1.43	**0.04**	**0.12**
MLP (best)	0.59 ($K = 30$)	1.73 ($K = 30$)	2.92 ($K = 100$)	5.43 ($K = 100$)
RBF (best)	1.17 ($K = 10$)	1.78 ($K = 10$)	1.19 ($K = 10$)	3.05 ($K = 10$)
GP (best)	**0.49** ($a = 0.1$)	**1.42** ($a = 1$)	1.17 ($a = 0.1$)	2.22 ($a = 1$)
	Dataset 1(c)		**Dataset 1(d)**	
FWNN	**0.03**	**0.24**	**1.29**	**2.01**
MLP (best)	3.67 ($K = 30$)	5.71 ($K = 10$)	23.96 ($K = 100$)	48.19 ($K = 100$)
RBF (best)	3.83 ($K = 20$)	6.55 ($K = 50$)	3.47 ($K = 80$)	5.77 ($K = 80$)
GP (best)	7.34 ($a = 0.1$)	8.76 ($a = 1$)	3.38 ($a = 0.1$)	5.59 ($a = 1$)

The indication "*best*" denotes the result of the optimal performer among a handful of trial configurations

for generating two datasets, each with 150 equidistant data points. The first 100 points were used for training, while the remaining 50 points labeled as $z_1, \ldots, z_{50}$ were used for evaluating the quality of extrapolation. We base the assessment for the extrapolation capability on the relative deviation at an extrapolation point defined by:

Table 2 Comparison of the NMSE mean over the test set, resulting from 50 independent experiments, for the 2*d* datasets related to Fig. 2a–c

Method	NMSE over the test set			
	Medium noise	High noise	Medium noise	High noise
	Dataset 2(a)		**Dataset 2(b)**	
FWNN	**11.14**	**22.83**	**1.55**	**4.66**
MLP (best)	19.84 ($K = 10$)	71.84 ($K = 10$)	2.34 ($K = 100$)	7.95 ($K = 100$)
RBF (best)	11.98 ($K = 50$)	51.73 ($K = 50$)	1.69 ($K = 50$)	8.11 ($K = 30$)
GP (best)	30.18 ($a = 0.001$)	53.09 ($a = 0.1$)	2.41 ($a = 0.1$)	7.52 ($a = 0.1$)
	Dataset 2(c)			
FWNN	**68.99**	**69.82**		
MLP (best)	110.71 ($K = 100$)	84.97 ($K = 100$)		
RBF (best)	86.18 ($K = 80$)	80.42 ($K = 80$)		
GP (best)	97.62 ($a = 0.1$)	90.55 ($a = 0.1$)		

The indication "*best*" denotes the result of the optimal performer among a handful of trial configurations

Table 3 Comparison of the extrapolation index *J*, for the two datasets related to Fig. 1a, b

Method	Fig. 1a dataset			Fig. 1b dataset		
	$r_b = 0.05$	$r_b = 0.15$	$r_b = 0.25$	$r_b = 0.05$	$r_b = 0.15$	$r_b = 0.25$
FWNN	25	50	50	24	35	38
MLP (best)	11	26	50	1	2	5
RBF (best)	8	22	33	7	10	13
GP (best)	18	31	39	17	23	26

The indication "*best*" denotes the result of the optimal performer among a handful of trial configurations. For the FWNN, only the standard configuration was used

$$r_i \equiv \frac{|f(z_i) - N(z_i, \hat{\theta})|}{\max\{1, |f(z_i)|\}} \tag{19}$$

By imposing an upper bound r_b, for the acceptable relative deviation, we determine *J*, the number of consecutive extrapolation points satisfying:

$$r_i < r_b,\ \forall\, i \le J \text{ and } r_{J+1} > r_b \tag{20}$$

Given a value for the upper bound r_b, inside a reasonable range $r_b \in [0, 0.25]$, the best method for extrapolation is the one with the highest value of *J*.

Table 3 contains the extrapolation results for three values of the upper bound, $r_b = \{0.05, 0.15, 0.25\}$. In particular, we show the mean values of the *J*-index that have resulted from 50 independent experiments. By inspection, it is clear that the FWNN outperforms the rival MLP and RBF networks, as well as the Gaussian processes. Further details and extrapolation experiments have been presented earlier in [3].

4.3 Experiments with real-world benchmarks

Additional experiments were performed on a variety of established benchmarks.

Table 4 Summary of the selected real-world datasets from the UCI repository

Dataset	*n*	#*T*	#[*S* − *T*]
abalon	8	1000	3177
airfoil	5	500	1003
bodyfat	13	100	152
concrete	8	500	530
CPU	12	500	7692
housing	13	200	306
mg	6	385	1000
pima	8	384	384
wine	11	1066	533

4.3.1 Experiments with UCI datasets

We have selected nine benchmarks from the UCI Machine Learning Repository[1]. which are briefly described in Table 4. Note that the last two datasets (*pima, wine*) are benchmarks used primarily for evaluating classification methods and contain data belonging to two and seven classes, respectively.

[1] These datasets are available at: http://mlr.cs.umass.edu/ml/datasets.html.

Journal : Large 42452 | Article No : 3713 | Pages : 12 | MS Code : 3713 | Dispatch : 29-10-2020

Table 5 Comparison of the NMSE mean over the test set, resulting from 50 independent experiments, for the nine UCI datasets

Method	UCI dataset					
	Train	Test	Train	Test	Train	Test
	Abalon $n = 8$		**Airfoil $n = 5$**		**Bodyfat $n = 13$**	
FWNN	3.44	**4.33**	0.06	**0.09**	1.75×10^{-5}	$\mathbf{3.03 \times 10^{-4}}$
MLP	3.43	4.47	0.11	0.12	4.23×10^{-3}	1.01×10^{-2}
(best)	($K = 100$)		($K = 100$)		($K = 10$)	
RBF	5.05	5.49	1.37	1.93	0.64	0.66
(best)	($K = 30$)		($K = 30$)		($K = 10$)	
GP	3.95	4.52	0.05	**0.09**	1.48×10^{-3}	4.32×10^{-4}
(best)	($a = 1e1$)		($a = 1e1$)		($a = 1e-5$)	
	concrete $n = 8$		**CPU $n = 12$**		**housing $n = 13$**	
FWNN	1.46	2.29	0.15	0.18	0.53	**1.99**
MLP	0.99	**1.91**	0.15	**0.17**	0.88	3.12
(best)	($K = 100$)		($K = 100$)		($K = 10$)	
RBF	21.36	21.72	2.81	2.95	7.06	12.67
(best)	($K = 20$)		($K = 10$)		($K = 30$)	
GP	0.71	2.46	0.11	**0.17**	0.93	3.21
(best)	($a = 1e-5$)		($a = 1e1$)		($a = 1e1$)	
	mg $n = 6$		**pima $n = 5$**		**wine $n = 11$**	
FWNN	1.27	**1.86**	4.86	5.67	1.05	**1.23**
MLP	1.21	1.99	4.43	5.83	0.97	1.25
(best)	($K = 10$)		($K = 30$)		($K = 100$)	
RBF	3.21	5.24	16.68	18.81	1.12	1.36
(best)	($K = 50$)		($K = 30$)		($K = 80$)	
GP	2.05	2.52	4.74	**5.48**	1.24	1.32
(best)	($a = 1e-1$)		($a = 1e-1$)		($a = 1$)	

The indication "*best*" denotes the result of the optimal performer among a handful of trial configurations

For each dataset and network architecture, 50 experiments were carried out. For these experiments, we have used 5^{th} degree polynomials ($L_w = L_\mu = L_\sigma = 5$) corresponding to a number of $6(n + 2)$ model parameters. For the MLP, RBF and GPs, we have experimented with a host of different architectural and regularization parameters, and in Table 5, we quote, for each of them, the best performing configuration. Observing these results, we note that FWNN outperforms all competitors in five (out of nine) datasets and in another dataset shares the top with GPs. MLP is top in one dataset and is tied at the top with GPs in another one. GPs is at the top in one dataset, while RBF failed to win the top in any of the UCI datasets.

Since the *pima* and *wine* datasets are classification benchmarks, the classification capability of FWNN has been tested. For this purpose, the classification accuracy is calculated as the percentage of the correctly classified test points within a tolerance (see [5]). The results are presented in Table 6 for four different tolerance values, namely: $\eta = 0.10, 0.25, 0.5$ and 1.0. In these experiments, FWNN together with GPs performs better than both the MLP and RBF networks. It is interesting to note the remarkable classification accuracy of the FWNN, particularly for the low tolerance value of $\eta = 0.10$.

4.3.2 Large-scale experiments

To further test the approximation quality of the FWNN, experiments on extensively studied complex, large datasets were performed. The datasets are summarized in Table 7. The *Sarcos* dataset is a robotic real-world benchmark [28], representing the inverse dynamics of a robotic seven-joint arm[2] related to rhythmic motions. The task is to map a 21-dimensional input space (seven joint positions, seven joint velocities, seven joint accelerations) to the corresponding seven joint torques.

The training in this case was performed using fifth-degree polynomials ($L_w = L_\mu = L_\sigma = 5$) corresponding to a total of 138 ($= 6n + 12$) parameters. The FWNN results along with results published by different authors using

[2] Sarcos dataset is available at http://www.gaussianprocess.org/gpml/data/.

Journal : Large 42452 | Article No : 3713 | Pages : 12 | MS Code : 3713 | Dispatch : 29-10-2020

Table 6 Classification accuracy for several tolerance values

Dataset	Classification accuracy (%)				
	FWNN	MLP (best)	RBF (best)	GPs (best)	Published
pima $\eta = 0.10$	**27.9**	20.3	10.0	21.6	
$\eta = 0.25$	**51.8**	46.6	27.1	45.1	
$\eta = 0.50$	**78.9**	75.3	56.0	**78.9**	77.7 ([8])
$\eta = 1.00$	**99.7**	98.4	78.9	99.2	
wine $\eta = 0.10$	**16.5**	14.6	12.2	14.1	
$\eta = 0.25$	38.6	35.4	30.8	33.4	43.2 ([5])
$\eta = 0.50$	62.3	61.1	58.9	59.8	62.4 ([5])
$\eta = 1.00$	89.0	89.3	87.4	88.7	89.0 ([5])

The indication "*best*" denotes the result of the optimal performer among a handful of trial configurations. In addition, we quote other published results

Table 7 Summary of the datasets used in our large-scale experiments

Dataset	*#features*	*#training*	*#testing*
Sarcos	21	44484	4449
Elevators	17	8752	7847
Kin40k	8	10000	30000
Pole Telecomm	26	10000	5000
Pumadyn32-nm	32	7168	1024

Note that the *Sarcos* dataset output has seven components (DOF)

Table 8 Mean and normalized mean squared errors for the SARCOS dataset

Method	Degree of freedom (DOF)						
	First	Second	Third	Fourth	Fifth	Sixth	Seventh
	Mean squared error (MSE)						
FWNN	**18.92**	**9.17**	**3.43**	**2.51**	**0.068**	**0.24**	**0.24**
Ref. [36]	31.08	22.68	9.08	9.73	0.13	0.83	0.43
	Normalized mean squared error (NMSE)						
FWNN	**0.031**	**0.011**	**0.015**	**0.003**	**0.031**	0.067	**0.009**
Ref. [23]	0.036	0.042	0.034	0.011	0.038	**0.056**	0.019

Each column relates to one of the seven torques

GPs are listed in Table 8 and compared favorably. In Fig. 3, the predicted versus the actual values are plotted, for all seven DOFs, rendering the model's performance obvious. We observe that all points are scattered symmetrically around and near to the diagonal $x = y$ line that represents the perfect match.

For the remaining (*Elevators, Kin40k, Pole Telecomm, Pumadyn32-nm*) datasets,[3] the FWNN results are listed in Table 9 along with results provided by a state-of-the-art Gaussian process approach reported in [19]. In spite its simplicity, the FWNN's performance is better or similar to that of a sophisticated, high-demanding, state-of-the-art method.

5 Discussion and conclusions

In the present article, we have proposed a new type of neural networks, the FWNN, in which the weights are functions of a continuous variable. This may be interpreted as a neural network with an infinite number of hidden nodes. In the conducted numerical experiments, the FWNN exceeded in generalization performance the MLP and RBF networks, as well as the Gaussian processes. This is evidence of robustness, reliability and modeling potential.

The FWNN has a number of interesting properties. There is ample experimental evidence that the generalization performance is superior. This may be related to the fact that the number of required parameters is limited, which in turn prevents serious overtraining.

3FL01 [3] All data were downloaded from http://www.dcc.fc.up.pt/~ltorgo/
3FL02 Regression/DataSets.html.

Journal : Large 42452 | Article No : 3713 | Pages : 12 | MS Code : 3713 | Dispatch : 29-10-2020

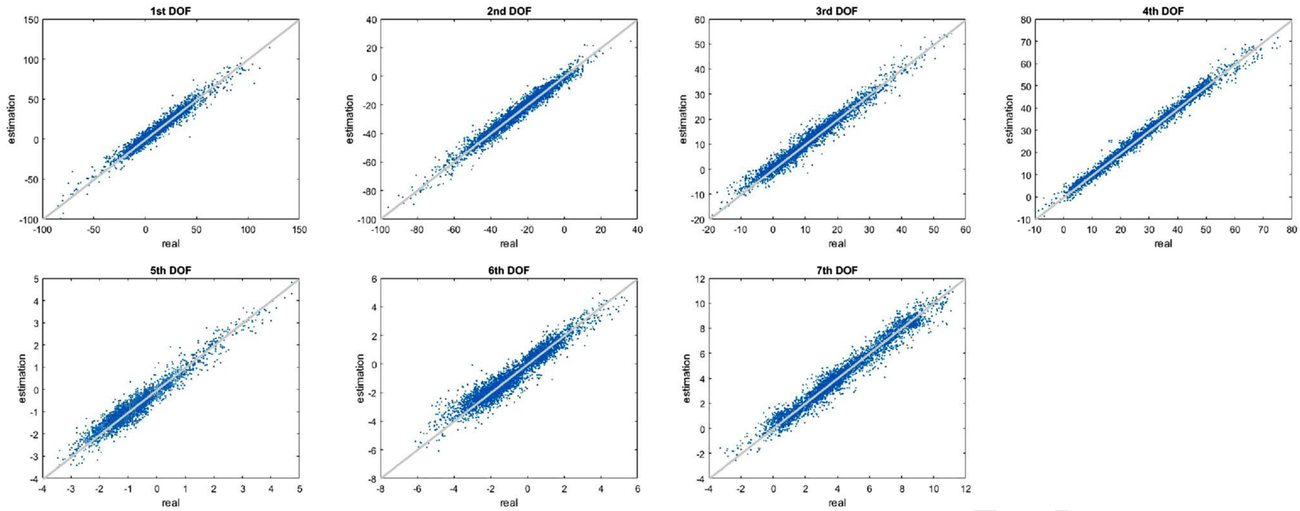


Fig. 3 Plots of the predicted (*y*-axes) versus the actual (*x*-axes) values of the 4484 test cases, for each of the seven DOFs in the SARCOS dataset. The diagonal line (thin) denotes the perfect match

Table 9 Comparison of the results (NMSE criterion) depicted with the proposed FFWN and those published in the literature

Method	Experimental dataset			
	Elevators	*Kin40k*	*Pole Telecomm*	*Pumadyn-32nm*
FWNN	**0.010**	0.022	**0.009**	**0.040**
Published [19]	0.115	**0.0120**	0.011	0.045

The positions of the Gaussian centers are determined by the $\boldsymbol{\mu}(s)$ and the corresponding widths by $\sigma(s)$, with $s \in [-1, 1]$. In the case of studying simulated datasets, we have used an affine form; hence, the $\mu(s)$ curve is a straight-line segment joining the two end points $\boldsymbol{\mu}(-1)$ and $\boldsymbol{\mu}(+1)$ in R^n. The widths are linearly increasing or decreasing with s, depending on the sign of σ_1. In spite of that this might seem to be a severe constraint, it has not degraded the network's performance. We credit this to the infinite number of nodes that render the approximation of any function feasible [6, 10]. In the case of real benchmarks, the affine model imposes an overly strict constraint, and thus it was replaced by a higher order polynomial, at the expense of some extra parameters. The Gaussian centers then may lie on a parabolic or a cubic locus, and the widths acquire higher adaptability.

The attractive features of the proposed FWNN may be briefly summarized as:

1. Frugal model, incorporating a small number of adjustable parameters.
2. Resistant to overtraining.
3. Superior interpolation and extrapolation performance.

We consider that some issues need further investigation and will become part of our future research effort. In particular,

- The model behavior when using different density functions.
- The effect caused by choosing different functional forms for the weights.
- The difference in using other than Gaussian kernels.
- The possibility of extending the shallow architecture to deep.

Furthermore, we would like to assess the effectiveness of FWNN in complex problems, such as solving partial and ordinary differential equations [15–17], modeling interatomic potentials [26], forecasting time series [34] and more a task that is underway.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest

Journal : Large 42452 | Article No : 3713 | Pages : 12 | MS Code : 3713 | Dispatch : 29-10-2020

Appendix: Derivatives

Since the optimization methods used for the training need derivative information, we list the FWNN first order derivatives.

Let us define for convenience the following quantity:

$$y(\boldsymbol{x}, \mu, \sigma) = \frac{|\boldsymbol{x} - \mu(s)|^2}{\sigma^2(s)} = \sum_{i=1}^{n} \left(\frac{x_i - \mu_i(s)}{\sigma(s)} \right)^2 \tag{21}$$

The network (16), is then rewritten as:

$$N_{FW}(\boldsymbol{x};\theta) = \int_{-1}^{1} \frac{ds}{\sqrt{1 - s^2}} w(s) \times \exp\left(-\frac{1}{2} y(\boldsymbol{x}, \mu, \sigma) \right) \tag{22}$$

and its partial first-order derivatives w.r.t. w, μ, and σ are given by:

$\forall\, r = 0, 1, \dots, L_w$

$$\frac{\partial N_{FG}(\boldsymbol{x};\theta)}{\partial w_r} = \int_{-1}^{1} \frac{ds}{\sqrt{1 - s^2}} s^r \exp\left(-\frac{1}{2} y(\boldsymbol{x}, \mu, \sigma) \right) \tag{23}$$

$\forall\, k = 1, \dots, n$ and $r = 0, 1, \dots, L_\mu$

$$\frac{\partial N_{FG}(\boldsymbol{x};\theta)}{\partial \mu_{kr}} = \int_{-1}^{1} \frac{ds}{\sqrt{1 - s^2}} w(s) \exp\left(-\frac{1}{2} y(\boldsymbol{x}, \mu, \sigma) \right) \times \frac{x_k - \mu_k(s)}{\sigma^2(s)} s^r \tag{24}$$

$\forall\, r = 0, 1, \dots, L_\sigma$

$$\frac{\partial N_{FG}(\boldsymbol{x};\theta)}{\partial \sigma_r} = \int_{-1}^{1} \frac{ds}{\sqrt{1 - s^2}} w(s) \exp\left(-\frac{1}{2} y(\boldsymbol{x}, \mu, \sigma) \right) \times \frac{|\boldsymbol{x} - \mu(s)|^2}{\sigma^3(s)} s^r \tag{25}$$

References

1. Bartlett P (1998) The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Trans Inf Theory 44:525–536
2. Bishop C (2006) Pattern recognition and machine learning. Springer, Berlin
3. Blekas K, Lagaris I (2017) Artificial neural networks with an infinite number of nodes. J Phys Conf Ser 915(1):012006
4. Broomhead D, Lowe D (1988) Multivariable functional interpolation and adaptive networks. Complex Syst 2:321–355
5. Cortez P, Cerdeira A, Almeida F, Matos T, Reis J (2009) Modeling wine preferences by data mining from physicochemical properties. Decis Support Syst 47(4):547–553
6. Cybenko G (1989) Approximations by superposition of sigmoidal functions. Mathe Control Signals Syst 2:303–314
7. Du KL (2010) Clustering: a neural network approach. Neural Netw 23(1):89–107. https://doi.org/10.1016/j.neunet.2009.08.007
8. Duch W (2010) Datasets used for classification: comparison of results. http://fizyka.umk.pl/kis-old/projects/datasets.html
9. Geman S, Bienenstock E, Doursat R (1992) Neural networks and the bias/variance dilemma. Neural Comput 4(1):1–58
10. Hornik K (1991) Approximation capabilities of multilayer feedforward networks. Neural Netw 4:251–257
11. Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2(5):359–366
12. Ismailov VE (2014) On the approximation by neural networks with bounded number of neurons in hidden layers. J Math Anal Appl 417(2):963–969
13. Krogh A, Hertz J (1992) A simple weight decay can improve generalization. Adv Neural Inf Process Syst 4:950–957
14. Kůrková V (1992) Kolmogorov's theorem and multilayer neural networks. Neural Netw 5(3):501–506
15. Lagaris I, Likas A, Fotiadis D (1997) Artificial neural network methods in quantum mechanics. Comput Phys Commun 104:1–14
16. Lagaris IE, Likas A, Fotiadis DI (1998) Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans Neural Netw 9:987–1000
17. Lagaris IE, Likas A, Papageorgiou DG (2000) Neural network methods for boundary value problems with irregular boundaries. IEEE Trans Neural Netw 11:1041–1049
18. Lagaris IE, Tsoulos IG (2008) Stopping rules for box-constrained stochastic global optimization. Appl Math Comput 197:622–632
19. Lazaro-Gredilla M, Quinonero-Candela J, Rasmussen C, Figueiras-Vidal A (2010) Sparse spectrum Gaussian process regression. J Mach Learn Res 11:1865–1881
20. MacKay D (1992) Bayesian interpolation. Neural Comput 4(3):415–447
21. Marcus GF (1998) Rethinking eliminative connectionism. Cogn Psychol 37:243–282
22. Marier Z, Sra S (2016) Diversity networks: neural network compression using determinantal point processes. In: International conference on learning representations (ICLR)
23. Meier F, Schaal S (2016) Drifting Gaussian processes with varying neighborhood sizes for online model learning. In: IEEE international conference on robotics and automation (ICRA), pp 264–269
24. Murphy K (2012) Machine learning: a probabilistic perspective. MIT Press, Cambridge
25. Neal RM (1996) Bayesian learning for neural networks, vol 118. Lecture notes in statistics. Springer, Berlin
26. Onat B, Cubuk ED, Malone BD, Kaxiras E (2018) Implanted neural network potentials: application to Li–Si alloys. Phys Rev B 97:094106
27. Powell M (1985) Radial basis functions for multivariable interpolation: a review. In: IMA conference on "Algorithms for the Approximation of Functions and Data". RMCS Shrivenham
28. Rasmussen C, Williams CI (2006) Gaussian processes for machine learning. MIT Press, Cambridge
29. Ripley BD (1996) Pattern recognition and neural networks. Cambridge University Press, Cambridge
30. Roux N, Bengio Y (2007) Continuous neural networks. In: Eleventh international conference on artificial intelligence and statistics (AISTATS), pp 404–411

Journal : Large 42452	Article No : 3713	Pages : 12	MS Code : 3713	Dispatch : 29-10-2020

31. Sietsma J, Dow R (1991) Creating artificial neural networks that generalize. Neural Netw 4:67–79
32. Srivastav N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15:1929–1958
33. Voglis C, Lagaris I (2006) A global optimization approach to neural network training. Neural Parallel Sci Comput 14:231–240
34. Wang JZ, Wang JJ, Zhang ZG, Guo SP (2011) Forecasting stock indices with back propagation neural network. Expert Syst Appl 38:14346–14355
35. Zhang GP (2000) Neural networks for classification: a survey. IEEE Trans Syst Man Cybern Part C (Appl Rev) 30(4):451–462
36. Zhao H, Stretcu O, Negrinho R, Smola A, Gordon G (2017) Efficient multi-task feature and relationship learning. arXiv:1702.04423
37. Zhao H, Tsai YH, Salakhutdinov R, Gordon G (2019) Learning from the experience of others: approximate empirical Bayes in neural networks. In: International conference on learning representations (ICLR)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Journal : **Large 42452**	Article No : **3713**	Pages : **12**	MS Code : **3713**	Dispatch : **29-10-2020**

Journal:	**42452**
Article: **3713**	

Author Proof

Author Query Form

Please ensure you fill out your response to the queries raised below and return this form along with your corrections

Dear Author

During the process of typesetting your article, the following queries have arisen. Please check your typeset proof carefully against the queries listed below and mark the necessary changes either directly on the proof/online grid or in the 'Author's response' area provided below

Query	Details Required	Author's Response
AQ1	The labels in figure (3) is not readable. Please provide a new figure with legible labels in Vector EPS or tiff / jpeg format with 600 dpi resolution.	
AQ2	Please provide a definition for the significance of bold values in Tables 1, 2, 5, 6, 8 and 9.	