Information Sciences 382–383 (2017) 170–178

Contents lists available at ScienceDirect

Information Sciences

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

Insights into randomized algorithms for neural networks: Practical issues and common pitfalls

Ming Li, Dianhui Wang*

Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC 3086, Australia

ARTICLE INFO

Article history:
Received 15 August 2016
Revised 2 December 2016
Accepted 4 December 2016
Available online 7 December 2016

Keywords:
Randomized algorithms
Neural networks
Incremental learning
Function approximation

ABSTRACT

Random Vector Functional-link (RVFL) networks, a class of learner models, can be regarded as feed-forward neural networks built with a specific randomized algorithm, i.e., the input weights and biases are randomly assigned and fixed during the training phase, and the output weights are analytically evaluated by the least square method. In this paper, we provide some insights into RVFL networks and highlight some practical issues and common pitfalls associated with RVFL-based modelling techniques. Inspired by the folklore that "all high-dimensional random vectors are almost always nearly orthogonal to each other", we establish a theoretical result on the infeasibility of RVFL networks for universal approximation, if a RVFL network is built incrementally with random selection of the input weights and biases from a fixed scope, and constructive evaluation of its output weights. This work also addresses the significance of the scope setting of random weights and biases in respect to modelling performance. Two numerical examples are employed to illustrate our findings, which theoretically and empirically reveal some facts and limits of such class of randomized learning algorithms.

1. Introduction

Universal approximation capability of neural networks as the theoretical foundation for data modelling has been extensively studied in literature [5,6,8,17]. However, there are few works asserting theoretical bounds on the number of hidden nodes required [13,18]. In other words, a neural network model with fewer number of hidden nodes cannot ensure modelling performance whilst a larger number of hidden nodes may lead to over-fitting phenomenon which implies poor generalization performance.

To resolve this problem, constructive schemes, which starts with a small size of network then incrementally generates hidden nodes and output weights until an acceptable learning performance is achieved, have received considerable attention in the past years [1,12]. Although some convergence properties of these methods can be theoretically established, there exist some limitations in practice due to the extensive search for the hidden parameters. Randomized learning techniques for neural networks become popular in recent years because of their good potential in dealing with large-scale data analysis, fast dynamic modelling, and real-time data processing [4,14,19,20,23]. To the best of our knowledge, researches on randomized algorithms for training neural networks can be tracked back to the 1980s [2]. For single hidden layer feed-forward neural networks, Schmidt et al. tried to randomly assign the input weights and biases from $[-1,1]$ and calculate the output

* Corresponding author.
E-mail address: dh.wang@latrobe.edu.au (D. Wang).

http://dx.doi.org/10.1016/j.ins.2016.12.007
0020-0255/

weights by solving a linear least square problem [21]. Unfortunately, such an attempt on the scope setting lacks scientific justification and it could not ensure the universal approximation ability of the resulted model. In [15,16], Pao and Takefji proposed Random Vector Functional-link (RVFL) networks where the input weights and biases are randomly generated and then fixed. RVFL networks have direct links from the input layer to the output layer, its output weights can be calculated by using a pseudo-inverse of the hidden output matrix. In 1995, Igelnik and Pao established some significant results on the approximation capability of RVFL networks [10]. As indicated in their proofs, however, the scope of the input weights and biases is specified and data dependent. In [22], Ivan and his co-workers compared RVFL approximators with Barron's greedy learning framework [1], and examined the feasibility of RVFL networks. To simplify the specification of the random parameters of in RVFL networks, Husmeier suggested to use symmetric and adjustable intervals for approximating a class of nonlinear maps [9]. For more details about the history and recent developments of randomized method for training neural networks, readers may refer to an informative editorial [23].

In this paper, we address some practical issues and common pitfalls in RVFL-based data modelling. Specifically, we look into some impacts of the scope of random parameters on the model's performance, and empirically show that a widely-used setting (e.g $[-1, 1]$) is misleading. Two illustrations are presented to clarify some incorrect perceptions about the way to randomly assign the input weights and biases. Also, we provide a theoretical verification about the infeasibility of a class of incremental RVFL (IRVFL) networks for universal approximation. Simulation results align with the theoretical analysis, and demonstrate that IRVFL networks have technical limits to model nonlinear maps with arbitrary accuracy.

The remainder of the paper is organized as follows. Section 2 briefly reviews constructive neural networks and presents some practical considerations on RVFL networks. Section 3 provides our theoretical analysis that shows IRVFL networks with specific coefficients will not universally approximate a given map. Section 4 reports some numerical results, revealing and correcting some common pitfalls in randomized learning techniques. Section 5 concludes this paper with some remarks.

2. Related work

2.1. Constructive neural networks

Let $L_2(D)$ denote the space of all Lebesgue-measurable functions f: $R^d \to R$ on a compact set $D \subset R^d$, with the L_2 norm defined as $\| f\|_2 := \sqrt{\langle f, f\rangle} = (\int_D |f(x)|^2 dx)^{1/2}$ where the inner product of f_1 and f_2 is defined as $\langle f_1, f_2\rangle = \int_D f_1(x) f_2(x) dx$. For a target function f: $R^d \to R$, assume a single layer feed-forward network (SLFN) with $L-1$ hidden nodes has already been constructed, i.e., $f_{L-1}(x) = \sum_{j=1}^{L-1} \beta_j g_j(w_j^{\mathrm{T}} x + b_j)$ $(f_0 = 0)$. If the current residual error, denoted as $e_{L-1} = f - f_{L-1}$, is still unacceptable, the problem of incremental learning becomes how to add β_L, g_L (w_L and b_L) leading to $f_L = f_{L-1} + \beta_L g_L$ until the residual error $e_L = f - f_L$ reaches a pre-defined tolerance ϵ for the given task, i.e., $\|e_L\|_2 \le \epsilon$.

In [1], Barron provided the framework of greedy approximation by using the classical Jones iteration method [11]. For a target function f that belongs to closure of the convex hull of a given Hilbert space G, i.e., $f \in \overline{\mathrm{conv}(G)}$, Barron's constructive scheme aims at finding α_L and g_L that minimize $\|\alpha_L f_{L-1} + (1-\alpha_L) g_L - f\|_2$ at each step. However, this strategy is only applicable to $f \in \overline{\mathrm{conv}(G)}$ rather than all Lebesgue-measurable functions $f \in L_2(D)$. Besides, the ability of f_L may be constrained because it is generated via a convex combination of f_{L-1} and g_L, rather than optimizing the output weights over the parameter space.

In [12], the authors proposed an approach by optimizing certain objective functions. Their proposed schemes can ensure the convergence property of the constructed model, provided that g_L, which maximizes $\langle e_{L-1}, g_L\rangle^2 / \|g\|_2^2$, can be found in each incremental step. However, it is not easy to obtain an optimal g_L^* as the optimization process might be frequently plagued by local minima when performing gradient ascent methods. The optimization process required for building a new hidden node is very inefficient, as mentioned in [12]. For many real world applications with large scale dataset and/or with real-time data processing, it is impractical to iteratively find out the parameters of the hidden nodes. Thus, fast algorithms for generating a new hidden node (basis function) are being expected.

2.2. Some practical considerations on RVFL networks

As indicated in [7] and [22], some additional conditions on the families of functions to be approximated are requested to ensure successful data modelling with a class of randomized approximators. It should be noticed that the universal approximation theorem presented in [10] holds for certain appropriate range of random parameters in the hidden nodes. Some theoretical analysis on the feasibility of randomized basis function approximators is given in [7], which proved that in the absence of certain additional conditions one may observe an exponential growth of the number of terms needed to approximate a nonlinear map, and the resulted model may be very sensitive to the random parameters. These work motivates us to highlight some 'risky' aspects caused by the randomness in RVFL networks. In particular, the illogical way of simply selecting a trivial range $[-1, 1]$ for random assignment of the input weights and biases should be questioned and corrected.

3. Theoretical analysis

This part aims to provide a theoretical analysis on the infeasibility of IRVFL networks for universal approximation. For a target function f: $R^d \to R$, IRVFL networks start from generating one (hidden) node $g_1 = g_1(w_1^{\mathrm{T}} x + b_1)$ (g is the activation

function) by randomly assigning w_1 and b_1 from $[-\lambda, \lambda]$, $\lambda > 0$; then the output weight linking this newly added node to the output layer is calculated by $\beta_1 = \langle e_0, g_1\rangle / \|g_1\|_2^2$, where $e_0 = f$; renew the error residual as $e_1 = f - \beta_1 g_1(w_1^{\mathrm{T}} x + b_1)$; adding new hidden nodes (one by one) by repeating these steps.

Suppose a learner model with L hidden nodes, i.e., $f_L(x) = \sum_{j=1}^{L} \beta_j g_j(w_j^{\mathrm{T}} x + b_j)$, has been constructed, with the current residual error denoted as $e_L = f - f_L$. Our analysis below focuses on the feasibility of the above incremental learning process, in other words, whether or not e_L converges to zero or is less than an expected tolerance ϵ for sufficiently large L, if its decreasing rate is subject to certain limitation.

Theorem 1. *Let span (Γ) be dense in L_2 and $\forall g \in \Gamma$, $0 < \|g\| < b$ for some $b \in \mathbb{R}^+$. Suppose that g_L is randomly generated and β_L is given by*

$$\beta_L = \frac{\langle e_{L-1}, g_L\rangle}{\|g_L\|_2^2}. \tag{1}$$

For sufficiently large L, if the followings hold:

$$\frac{\|e_{L-1}\|_2^2 - \|e_L\|_2^2}{\|e_{L-1}\|_2^2} \le \varepsilon_L < 1, \tag{2}$$

and

$$\lim_{L\to\infty} \prod_{k=1}^{L} (1 - \varepsilon_k) = \varepsilon > 0. \tag{3}$$

Then, the constructive neural network with random weights, f_L, has no universal approximation capability, that is,

$$\lim_{L\to\infty} \|f - f_L\|_2 \ge \sqrt{\varepsilon}\|f\|_2. \tag{4}$$

Proof. Simple computation can verify that the sequence $\{\|e_L\|_2^2\}$ is monotonically decreasing and converges. Indeed,

$$\begin{aligned}
&\|e_L\|_2^2 - \|e_{L-1}\|_2^2 \\
&= \langle e_{L-1} - \beta_L g_L, e_{L-1} - \beta_L g_L\rangle - \langle e_{L-1}, e_{L-1}\rangle \\
&= -2\langle e_{L-1}, \beta_L g_L\rangle + \langle \beta_L g_L, \beta_L g_L\rangle \\
&= -2\frac{\langle e_{L-1}, g_L\rangle^2}{\|g_L\|_2^2} + \frac{\langle e_{L-1}, g_L\rangle^2}{\|g_L\|_2^2} \\
&= -\frac{\langle e_{L-1}, g_L\rangle^2}{\|g_L\|_2^2} \le 0,
\end{aligned} \tag{5}$$

which means $\{\|e_L\|_2^2\}$ is monotonically decreasing and converges because it is bounded by zero.

From (2), we can easily prove that

$$\|e_L\|_2^2 \ge (1 - \varepsilon_L)\|e_{L-1}\|_2^2 \ge \prod_{k=1}^{L} (1 - \varepsilon_k)\|f\|_2^2. \tag{6}$$

Therefore, we get

$$\lim_{L\to\infty} \|e_L\|_2^2 \ge \lim_{L\to\infty} \prod_{k=1}^{L} (1 - \varepsilon_k)\|f\|_2^2 = \varepsilon\|f\|_2^2. \tag{7}$$

This completes the proof. □

Remark 1. It is easy to find a nonnegative decreasing sequence $\{\varepsilon_L\}$, for example, $\varepsilon_L = \frac{1}{4L^2}$, that satisfies

$$\lim_{L\to\infty} \prod_{k=1}^{L} (1 - \varepsilon_k) = \frac{2}{\pi} > 0. \tag{8}$$

Therefore, $\lim_{L\to\infty} \|e_L\|_2 = \sqrt{\frac{2}{\pi}}\|f\|_2$, if $(\|e_{L-1}\|_2^2 - \|e_L\|_2^2)/(\|e_{L-1}\|_2^2) \le \frac{1}{4L^2}$.

On the other hand, the probability of the event $(\|e_{L-1}\|_2^2 - \|e_L\|_2^2)/(\|e_{L-1}\|_2^2) \le \varepsilon_L$ can be predicted by using the theoretical results in [3,7], which provides a precise characterization of the folklore "all high-dimensional random vectors are almost always nearly orthogonal to each other". In particular, we can see that

$$\frac{\|e_{L-1}\|_2^2 - \|e_L\|_2^2}{\|e_{L-1}\|_2^2} = \frac{\langle e_{L-1}, g_L\rangle^2}{\|e_{L-1}\|_2^2\|g_L\|_2^2} = \langle z_1, z_2\rangle^2, \tag{9}$$

where $z_1 = \frac{e_{L-1}}{\|e_{L-1}\|_2}$ and $z_2 = \frac{g_L}{\|g_L\|_2}$ can be considered as two random vectors in R^N with $\|z_1\|_2 = \|z_2\|_2 = 1$. Based on the results in [3,7], we have

$$P\left(\frac{\|e_{L-1}\|_2^2 - \|e_L\|_2^2}{\|e_{L-1}\|_2^2} \le \varepsilon_L\right) \ge 1 - \exp^{-N\varepsilon_L}, \tag{10}$$

where $P(X)$ stands for the probability of the event X. Indeed, (8) implies the fact that IRVFL networks, with the maximum number of hidden nodes set as $L_{\max}$, is less feasible for data modelling with a specified tolerance $\eta > 0$ in the sense of probability, that is,

$$P\Big(\|f - f_{L_{max}}\|_2 \ge \eta\Big) \ge 1 - \gamma, \tag{11}$$

where γ is sufficiently small when $N\varepsilon_{L_{\max}}$ is very large.

Remark 2. For a sufficiently large N, each newly added hidden node produces one high-dimensional random variable. Therefore, the phenomenon of almost orthogonality in high dimensions is inevitable during the course of construction. Unlike the constructive scheme based on optimization techniques [12], for example, maximizing $\langle e_{L-1}, g_L\rangle^2/\|g_L\|_2^2$, the process of generating new hidden nodes randomly with a fixed scope $[-\lambda, \lambda]$ has no guarantee to produce a feasible model in practice.

4. Simulation results

In this section, we present a series of simulation results to illustrate the significance of the scope of random parameters in RVFL networks for modelling performance, and the infeasibility of a class of IRVFL networks for universal approximation. Two functions are employed in this simulation study. All experiments are carried out by using MATLAB 7.0 on a computer with 3.5GB RAM and 2.4 GHz Intel Core 2 Duo processor.

4.1. Experimental setup

The used sample data in simulations are generated by the following functions.

- The first function is given by

$$f_1(x) = 0.2e^{-(10x-4)^2} + 0.5e^{-(80x-40)^2} + 0.3e^{-(80x-20)^2},\ x \in [0, 1].$$

- The second function is a SinE function expressed by

$$f_2(x) = 0.8\exp(-0.2x)\sin(10x),\ x \in [0, 5].$$

Our objectives are to look into: (i) if the built randomized learner model with various scope settings can perform the regression task; (ii) if the IRVFL networks with the specific configuration as discussed above can approximation these nonlinear functions through randomized learning techniques. Specifically, we detail our experiments associated with the aforementioned objectives as followed:

- **Q1.** Given N training samples, choose different number of hidden nodes $L = \alpha N$, where α is selected from the set {0.2, 0.4, 0.6, 0.8, 1}. Is the hidden output matrix H (of size $N \times L$) full-rank or not, if randomly assigning the input weights and biases from certain scopes. Besides, can the trained model approximate the target functions?
- **Q2.** Given a number of training and test samples, can the error residual approach a sufficiently small tolerance ϵ, until L (relatively large) hidden nodes are generated in IRVFL networks (detailed in Section 3). In addition, is the error changing rate (i.e., $r_L = (\| e_{L-1}\|_2 - \| e_L\|_2)/\| e_{L-1}\|_2$) significant for reducing the error residual?

In this study, we generated randomly various numbers of both training and test data for f_1 and f_2 (with uniform distribution in [0,1] and [0,5], respectively). In particular, we take 1000 points as the training data and another 1000 points as the test data in our experiments for **Q2**. The sigmoidal activation function $g(x) = 1/(1 + \exp(-x))$ is applied in our simulations.

4.2. Significance of the scope setting

Tables 1 and 2 report some numerical results with various settings of the sample size N and the number of hidden nodes L. For each case, the same number of test samples are used for performance evaluation. In Tables 1–4 (applicable for Tables 5–8 as well), both average values and standard deviations (for rank values of H, training and test performance in RMSE: root-mean-square error) are reported over 100 independent trials. In Table 1, the scope for randomly assigning w and b is set as $[-1, 1]$. It is found that in all cases H is **NOT** invertible, at the same time, both the training and test errors are near 0.06, which indeed is far less than expected. In Table 2, where the scope is chosen as $[-200, 200]$, both the training and test errors become much smaller than that reported in Table 1, showing that the resulted learner model has much

Table 1
Performance of RVFL networks on f_1 when $w, b \in [-1, 1]$.

$N = L$	Rank of H	Training RMSE	Test RMSE
300	10.68 ± 0.47	5.39e−2 ± 3.67e−4	5.27e−2 ± 3.51e−4
500	10.05 ± 0.22	6.96e−2 ± 2.40e−4	5.84e−2 ± 2.49e−4
1000	10.00 ± 0	6.40e−2 ± 4.56e−6	6.44e−2 ± 3.83e−6
1500	10.00 ± 0	5.71e−2 ± 3.43e−6	6.41e−2 ± 3.53e−6
2000	10.00 ± 0	6.40e−2 ± 2.37e−6	6.42e−2 ± 2.83e−6
2500	10.00 ± 0	5.89e−2 ± 3.33e−6	5.81e−2 ± 2.96e−6

Table 2
Performance of RVFL networks on f_1 when $w, b \in [-200, 200]$.

$N = L$	Rank of H	Training RMSE	Test RMSE
300	91.81 ± 7.12	8.68e−3 ± 7.29e−3	1.92e−1 ± 6.34e−1
500	142.08 ± 9.40	1.14e−3 ± 1.76e−3	1.79e−1 ± 9.70e−1
1000	258.30 ± 12.10	**1.74e−6 ± 4.94e−6**	**2.53e−6 ± 7.29e−6**
1500	333.35 ± 8.93	**1.25e−7 ± 1.71e−7**	**1.33e−7 ± 2.18e−7**
2000	366.65 ± 6.33	**6.62e−8 ± 3.30e−8**	**6.80e−8 ± 3.28e−8**
2500	389.98 ± 5.15	**5.55e−8 ± 2.46e−8**	**5.59e−8 ± 2.44e−8**

Table 3
Training performance of RVFL networks on f_1 when $w, b \in [-1, 1]$.

N	Training RMSE with $L = \alpha N$				
	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 1$
300	5.05e−2 ± 2.77e−5	5.00e−2 ± 2.74e−4	5.00e−2 ± 9.46e−5	5.05e−2 ± 1.74e−4	5.03e−2 ± 2.75e−4
500	5.41e−2 ± 3.44e−4	5.43e−2 ± 1.22e−5	5.42e−2 ± 1.79e−4	5.37e−2 ± 4.52e−4	5.42e−2 ± 2.60e−4
1000	6.65e−2 ± 1.35e−5	6.65e−2 ± 8.73e−6	6.65e−2 ± 6.21e−6	6.65e−2 ± 5.59e−6	6.65e−2 ± 4.68e−6
1500	6.01e−2 ± 7.83e−6	6.01e−2 ± 6.25e−6	6.01e−2 ± 4.62e−6	6.01e−2 ± 4.04e−6	6.01e−2 ± 3.52e−6
2000	6.45e−2 ± 9.49e−6	6.45e−2 ± 6.17e−6	6.45e−2 ± 4.84e−6	6.45e−2 ± 4.20e−6	6.45e−2 ± 3.84e−6
2500	6.17e−2 ± 7.98e−6	6.17e−2 ± 4.32e−6	6.17e−2 ± 3.82e−6	6.17e−2 ± 3.05e−6	6.17e−2 ± 2.80e−6

Table 4
Training performance of RVFL networks on f_1 when $w, b \in [-200, 200]$.

N	Training RMSE with $L = \alpha N$				
	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 1$
300	4.70e−2 ± 7.91e−3	3.23e−2 ± 7.85e−3	2.15e−2 ± 9.47e−3	1.42e−2 ± 7.23e−3	**7.85e−3 ± 6.14e−3**
500	5.07e−2 ± 1.75e−2	2.67e−2 ± 1.30e−2	**9.45e−3 ± 7.45e−3**	**3.88e−3 ± 3.65e−3**	**1.21e−3 ± 1.75e−3**
1000	2.12e−2 ± 9.52e−3	**3.39e−3 ± 3.30e−3**	**4.41e−4 ± 9.63e−4**	**2.92e−5 ± 6.17e−5**	**5.04e−6 ± 2.76e−5**
1500	**8.15e−3 ± 6.48e−3**	**4.76e−4 ± 9.06e−4**	**8.21e−6 ± 1.66e−5**	**1.74e−7 ± 2.50e−7**	**1.09e−7 ± 1.34e−7**
2000	**3.74e−3 ± 3.66e−3**	**5.28e−5 ± 2.09e−4**	**2.11e−7 ± 4.45e−7**	**5.47e−8 ± 2.58e−8**	**6.39e−8 ± 2.85e−8**
2500	**1.18e−3 ± 1.34e−3**	**1.89e−6 ± 4.02e−6**	**4.67e−8 ± 2.63e−8**	**5.21e−8 ± 2.56e−8**	**5.40e−8 ± 2.18e−8**

Table 5
Performance of RVFL networks on f_2 when $w, b \in [-1, 1]$.

$N = L$	Rank of H	Training RMSE	Test RMSE
300	18.40 ± 0.50	2.63e−1 ± 5.73e−3	2.70e−1 ± 7.34e−3
500	18.00 ± 0	2.83e−1 ± 3.21e−4	2.85e−1 ± 8.80e−4
1000	18.00 ± 0	2.82e−1 ± 2.74e−4	2.72e−1 ± 3.89e−4
1500	18.00 ± 0	2.74e−1 ± 3.12e−4	2.83e−1 ± 2.76e−4
2000	17.00 ± 0	2.81e−1 ± 2.76e−4	2.85e−1 ± 2.42e−4
2500	17.00 ± 0	2.82e−1 ± 2.13e−4	2.79e−1 ± 2.45e−4

Table 6
Performance of RVFL networks on f_2 when $w, b \in [-5, 5]$.

$N = L$	Rank of H	Training RMSE	Test RMSE
300	43.71 ± 0.69	1.82e−3 ± 8.63e−4	1.01e−1 ± 6.29e−2
500	44.27 ± 0.51	1.60e−3 ± 5.37e−4	2.26e−3 ± 6.97e−4
1000	44.06 ± 0.24	**1.58e−3 ± 2.98e−4**	**1.66e−3 ± 2.89e−4**
1500	44.43 ± 0.50	**1.42e−3 ± 3.03e−4**	**1.62e−3 ± 3.58e−4**
2000	43.63 ± 0.49	**1.67e−3 ± 2.05e−4**	**1.72e−3 ± 2.20e−4**
2500	43.66 ± 0.48	**1.70e−3 ± 1.86e−4**	**1.75e−3 ± 1.81e−4**

Table 7
Training performance of RVFL networks on f_2 when $w, b \in [-1, 1]$.

N	Training RMSE with $L = \alpha N$				
	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 1$
300	2.60e−1 ± 2.61e−3	2.52e−1 ± 1.10e−2	2.56e−1 ± 6.15e−3	2.56e−1 ± 5.77e−3	2.47e−1 ± 1.19e−2
500	2.65e−1 ± 1.94e−3	2.64e−1 ± 8.29e−4	2.64e−1 ± 1.74e−3	2.63e−1 ± 3.26e−3	2.64e−1 ± 4.77e−4
1000	2.74e−1 ± 1.26e−3	2.73e−1 ± 5.07e−4	2.73e−1 ± 3.87e−4	2.73e−1 ± 2.99e−4	2.73e−1 ± 2.96e−4
1500	2.86e−1 ± 1.40e−3	2.88e−1 ± 1.27e−3	2.85e−1 ± 6.18e−4	2.85e−1 ± 2.86e−4	2.85e−1 ± 2.32e−4
2000	2.82e−1 ± 1.63e−3	2.83e−1 ± 1.18e−3	2.81e−1 ± 4.52e−4	2.81e−1 ± 1.35e−3	2.83e−1 ± 2.84e−4
2500	2.86e−1 ± 6.68e−4	2.84e−1 ± 2.29e−3	2.86e−1 ± 3.50e−4	2.86e−1 ± 2.92e−4	2.86e−1 ± 2.97e−4

Table 8
Training performance of RVFL networks on f_2 when $w, b \in [-5, 5]$.

N	Training RMSE with $L = \alpha N$				
	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 1$
300	5.95e−2 ± 3.55e−2	1.02e−2 ± 1.04e−2	4.78e−3 ± 4.89e−3	**2.40e−3 ± 1.04e−3**	**1.90e−3 ± 8.50e−4**
500	2.08e−2 ± 1.48e−2	3.78e−3 ± 2.72e−3	**2.11e−3 ± 1.14e−3**	**1.85e−3 ± 6.28e−4**	**1.57e−3 ± 5.21e−4**
1000	5.26e−3 ± 5.00e−3	**2.14e−3 ± 9.67e−4**	**1.64e−3 ± 4.22e−4**	**1.70e−3 ± 3.25e−4**	**1.57e−3 ± 2.12e−4**
1500	3.18e−3 ± 1.58e−3	**2.24e−3 ± 4.21e−4**	**1.83e−3 ± 3.65e−4**	**1.70e−3 ± 2.50e−4**	**1.48e−3 ± 3.09e−4**
2000	**2.67e−3 ± 1.10e−3**	**1.91e−3 ± 3.27e−4**	**1.64e−3 ± 2.04e−4**	**1.81e−3 ± 1.49e−4**	**1.75e−3 ± 1.90e−4**
2500	**2.49e−3 ± 6.38e−4**	**2.01e−3 ± 3.08e−4**	**1.96e−3 ± 1.88e−4**	**1.90e−3 ± 1.49e−4**	**1.71e−3 ± 2.15e−4**

Table 9
Training and test performance, real error decreasing rate, and the frequency of cases when $E_L \leq 0.001$ for IRVFL networks with $L = 5000$ and $L = 10,000$.

Target	Scope Setting	$L = 5000$			$L = 10,000$			p
		E_L	r_L	$\tilde{E}_L$	E_L	r_L	$\tilde{E}_L$	
f_1	$\lambda = 1$	0.0867	1.0528e−6	0.0894	0.0861	6.0383e−7	0.0889	0%
	$\lambda = 5$	0.0701	3.3742e−7	0.0720	0.0689	5.2599e−6	0.0708	0%
	$\lambda = 10$	0.0608	7.1617e−6	0.0629	0.0594	9.7421e−7	0.0617	0%
	$\lambda = 50$	0.0511	7.2501e−6	0.0556	0.0501	2.5366e−6	0.0546	0%
	$\lambda = 100$	0.0470	3.6098e−6	0.0513	0.0443	1.3200e−5	0.0483	0%
	$\lambda = 150$	0.0440	2.2082e−5	0.0481	0.0403	3.1085e−5	0.0442	0%
	$\lambda = 200$	0.0419	3.4586e−5	0.0459	0.0393	7.2569e−5	0.0411	0%
f_2	$\lambda = 1$	0.3722	6.0199e−7	0.3691	0.3715	1.8746e−7	0.3687	0%
	$\lambda = 5$	0.3558	9.5618e−7	0.3618	0.3538	1.4427e−6	0.3594	0%
	$\lambda = 10$	0.3130	6.4169e−6	0.3149	0.3008	3.0559e−6	0.3033	0%
	$\lambda = 50$	0.1996	3.2870e−5	0.2046	0.1753	1.1726e−5	0.1798	0%
	$\lambda = 100$	0.1569	5.8076e−5	0.1616	0.1186	2.2033e−4	0.1223	0%
	$\lambda = 150$	0.1429	2.9097e−5	0.1474	0.0979	1.6259e−4	0.1012	0%
	$\lambda = 200$	0.1348	1.6353e−4	0.1411	0.0868	9.4113e−5	0.0909	0%

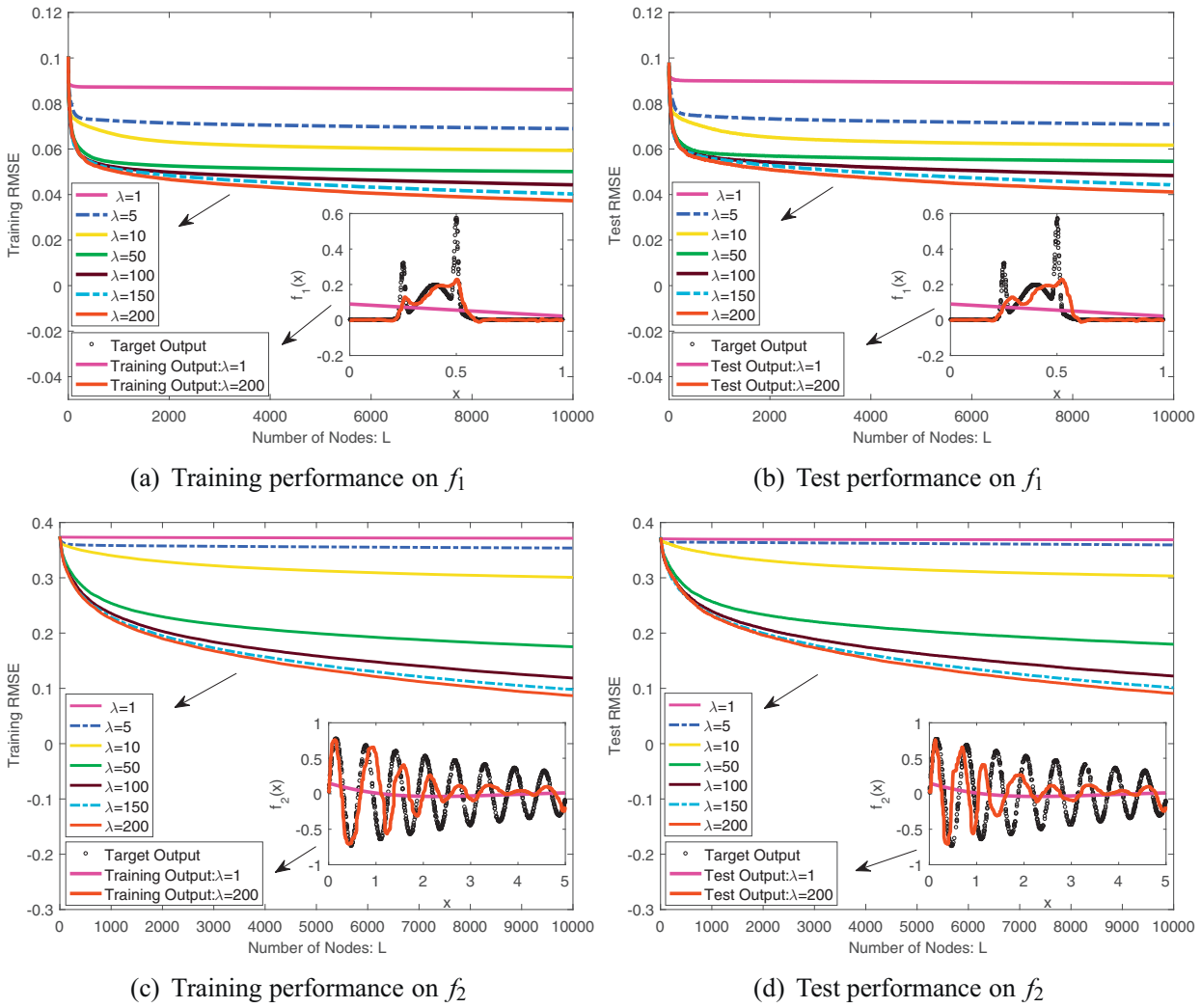
better performance in both learning and generalization. Interestingly, the hidden layer output matrix H led by this special scope is **NOT** full-rank as well.

In Tables 3 and 4, we have tried different settings of N and $L = \alpha N$, $\alpha = [0.2, 0.4, 0.6, 0.8, 1]$. As indicated in Table 3 where $w, b \in [-1, 1]$, no matter in which circumstance (N, L), the corresponding training error is far more larger than expected. Conversely, if $w, b \in [-200, 200]$, there are several pairs of (N, L) (highlighted with bold values in Table 4) that can lead to acceptable and preferable performance.

Similarly, the reported results of f_2 are in accordance with what we have mentioned for f_1. That is to say, the scope $[-1, 1]$ for randomly assigning w and b lead to the same outcome, i.e., H is **NOT** full-rank, the training and test performance is far away from acceptable levels. In contrast, RVFL networks with the specific scope $[-5, 5]$ performs favourably in both learning and generalization, as shown in Table 8.

4.3. Infeasibility of IRVFL networks

Table 9 presents the training and test errors (RMSE), denoted as E_L and $\tilde{E}_L$, and the error changing rate r_L for $L = 5000$ and $L = 10000$, with different settings of λ. The records of E_L, $\tilde{E}_L$, and r_L are average values over 100 independent trials. For each setting of λ, given $\epsilon = 0.001$ as a training error tolerance, the frequency of cases with $E_L \leq \epsilon$ (denoted by p) is calculated over all 200 trials for $L = 5000$ and $L = 10000$. It is believed that such large sized IRVFL networks are enough to demonstrate its effectiveness and feasibility for approximation. It is interesting to see that the value of r_L for each λ is very small but the resulted training and test error are still far away from the given tolerance ϵ. Also, it can be clearly seen in Fig. 1, the residual errors keep decreasing but still stay above a level that is far away from zero. For each setting of λ,



(a) Training performance on f_1 (b) Test performance on f_1

(c) Training performance on f_2 (d) Test performance on f_2

Fig. 1. Error decreasing trend of IRVFL networks with different settings of λ: (a) and (b) for f_1, (c) and (d) for f_2.

the value of p (based on all the trials of $L = 5000$ and $L = 10000$) is equal to zero, which possibly implies that the errors converge to a positive constant rather than zero. The approximation results for the case $\lambda = 200$ (on both the training and test data) are plotted as subgraphs inside of Fig. 1 (a)–(d), by which the infeasibility of IRVFL networks can be observed.

The comparison between RVFL networks and IRVFL networks is reported in Table 10, where we only report the results of some specific settings of λ and L. It can be found that the performance of IRVFL networks is far worse than that of RVFL networks, with the same setting of λ and L. In Table 10, some results of RVFL networks are highlighted to show that for some reasonable scopes (like $\lambda = 100$ or $\lambda = 200$ for f_1, $\lambda = 5$ or 10 for f_2), it is possible for RVFL networks to successfully approximate these nonlinear maps. We plot the real approximation results of IRVFL networks and RVFL networks in Fig. 2, where the dashed curves are the results of IRVFL networks while the solid curves represent the outcomes of RVFL networks.

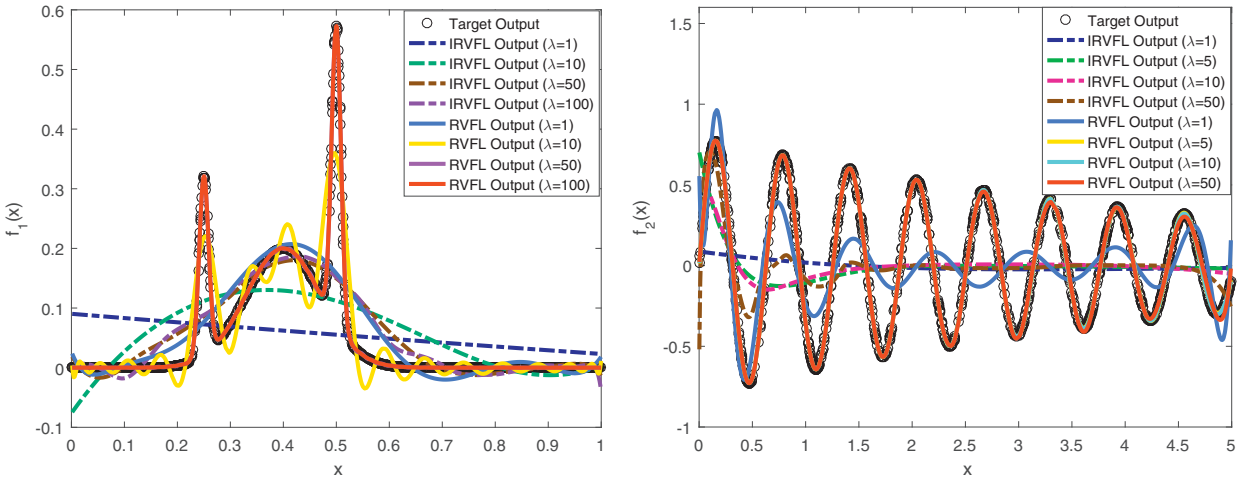
4.4. Discussion and summary

Clearly, our concerns in **Q1** and **Q2** have been addressed with empirical evidence. Specifically, Tables 1–8 verify that the hidden output matrix H is **NOT** full-rank with very high probability, if the input weights and biases are randomly assigned from certain scope (either $[-1, 1]$, $[-5, 5]$, or $[-200, 200]$). In particular, the scope $[-1, 1]$ makes RVFL networks unable to model these two nonlinear maps, no matter how many training samples are provided or what sized RVFL networks are used. However, there exist certain scopes (not unique) that can make RVFL networks to perform reasonably good for the given functions. Here, we summarize our findings related to **Q1** as followed.

- **R1**: Given a target function f and its N arbitrary distinct samples $\{x_i, t_i\}_{i=1}^N$, where $x_i \in R^n$ and $t_i \in R$. For any RVFL network with L hidden nodes ($L = N$) and the sigmoidal activation function $g(x) = 1/(1 + \exp(-x))$, there exist some

Table 10
Performance comparison between RVFL networks and IRVFL networks with different settings of λ and L.

Target	Algorithms	Training RMSE		Test RMSE	
		$L = 200$	$L = 500$	$L = 200$	$L = 500$
f_1	IRVFL ($\lambda = 1$)	0.0993	0.0991	0.1023	0.1022
	IRVFL ($\lambda = 100$)	0.0706	0.0662	0.0732	0.0681
	IRVFL ($\lambda = 200$)	0.0704	0.0643	0.0730	0.0661
	RVFL ($\lambda = 1$)	0.0676	0.0676	0.0693	0.0693
	RVFL ($\lambda = 100$)	0.0161	**2.84e−4**	0.0165	**3.15e−4**
	RVFL ($\lambda = 200$)	0.0230	**1.25e−3**	0.0237	**1.35e−3**
f_2	IRVFL ($\lambda = 1$)	0.3610	0.3601	0.3739	0.3731
	IRVFL ($\lambda = 5$)	0.3477	0.3473	0.3605	0.3599
	IRVFL ($\lambda = 10$)	0.3446	0.3390	0.3566	0.3504
	RVFL ($\lambda = 1$)	0.2777	0.2771	0.2702	0.2693
	RVFL ($\lambda = 5$)	**5.34e−3**	**1.85e−3**	**5.47e−3**	**1.92e−3**
	RVFL ($\lambda = 10$)	**8.35e−4**	**6.03e−6**	**8.47e−4**	**6.32e−6**



(a) f_1 (b) f_2

Fig. 2. Demonstration of test results for IRVFL networks and RVFL networks with different settings of λ: (a) for f_1 with $L = 500$; (b) for f_2 with $L = 200$.

special scopes such that if the input weights and biases are randomly assigned from such scope, the hidden output matrix H is **NOT** full-rank and $\|H\beta - T\| = \gamma > 0$ with probability one.

Based on Tables 9, 10, Figs. 1, and 2 , it can be concluded that IRVFL networks are ineffective and infeasible for data modelling. It is found that the error decreasing rate of IRVFL networks will become very small after a few hidden nodes are generated, if the scope for random parameters is fixed in advance. In practice, given a tolerance for error decreasing rate, the incremental process of IRVFL networks is convergent but with an error residual still quite large. Furthermore, if the scope for randomly assigning w and b are rigidly fixed in one scale (also noted in [7]), there is a high probability that $g_{L+1} \approx g_L$ (under a certain degree of precision), which means the following incremental procedures are useless, as demonstrated in Fig. 1.

- **R2**: For some target functions, IRVFL networks generated incrementally by adding new hidden node g_L with random input weights and biases from a fixed scope, and the output weights calculated by $\beta_L = \langle e_{L-1}, g_L \rangle / \|g_L\|_2^2$ will not have a chance to build a universal approximator, if the condition (3) is met.

5. Conclusions

This paper aims to draw attention on some practical issues and common pitfalls in using RVFL networks for data modelling. The significance of the scope for randomly assigning hidden parameters in RVFL networks, as well as the algebraic property (rank) of the hidden output matrix are addressed through numerical examples. It is risky to apply RVFL networks for data modelling with a fixed scope setting (i.e., [−1,1]). Also, as a technical contribution to the working field, we point out the ineffectiveness and infeasibility of IRVFL networks that are incrementally generated by randomly assigning the input

weights and biases. Our experimental results demonstrate some weaknesses and limits of RVFL networks for building fast prediction models, if improper settings and configurations are used. Our findings reported in this paper will help in getting better and correct understandings on the randomized learning techniques for neural networks.

Further researches along this direction can be both theoretical aspects and algorithmic developments. For instance, efficiently constructing IRVFL networks that ensure the universal approximation capability; developing more reliable and secured RVFL models for streaming data analysis.

References

[1] A.R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, IEEE Trans. Inf. Theory 39 (3) (1993) 930–945.
[2] D.S. Broomhead, D. Lowe, Multivariable functional interpolation and adaptive networks, Complex Syst. 2 (1988) 321–355.
[3] T. Cai, J. Fan, T. Jiang, Distributions of angles in random packing on spheres, J. Mach. Learn. Res. 14 (1) (2013) 1837–1864.
[4] C. Cui, D. Wang, High dimensional data regression using lasso model and neural networks with random weights, Inf. Sci. 372 (2016) 505–517.
[5] T. Chen, H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, IEEE Trans. Neural Netw. 6 (4) (1995) 911–917.
[6] G. Cybenko, Approximations by superpositions of a sigmoidal function, mathematics of control, Signals Syst. 2 (1989) 303–314.
[7] A.N. Gorban, I.Y. Tyukin, D.V. Prokhorov, K.I. Sofeikov, Approximation with random bases: pro-et contra, Inf. Sci. 364 (2016) 129–145.
[8] K. Hornik, M. Stinchcombe, H. White, Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks, Neural Netw. 3 (5) (1990) 551–560.
[9] D. Husmeier, Random vector functional link (RVFL) networks, Neural Networks for Conditional Probability Estimation: Forecasting Beyond Point Predictions-Springer, 1999. Chapter 6
[10] B. Igelnik, Y.H. Pao, Stochastic choice of basis functions in adaptive function approximation and the functional-link net, IEEE Trans. Neural Netw. 6 (6) (1995) 1320–1329.
[11] L.K. Jones, A simple lemma on greedy approximation in hilbert space and convergence rates for projection pursuit regression and neural network training, Ann. Stat. 20 (1) (1992) 608–613.
[12] T.Y. Kwok, D.Y. Yeung, Objective functions for training new hidden units in constructive neural networks, IEEE Trans. Neural Netw. 8 (5) (1997) 1131–1148.
[13] Y. LeCun, L. Bottou, G.B. Orr, K.R. Müller, Efficient backprop, in: Lecture Notes in Computer Science, volume 1524, 1998, pp. 9–50.
[14] M.W. Mahoney, Randomized algorithms for matrices and data, Found. Trends Mach. Learn. 3 (2) (2011) 123–224.
[15] 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.
[16] Y.H. Pao, Y. Takefji, Functional-link net computing: theory, system architecture, and functionalities, IEEE Comput. 25 (5) (1992) 76–79.
[17] J. Park, I. Sandberg, Universal approximation using radial basis function networks, Neural Comput. 3 (1991) 246–257.
[18] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by backpropagating errors, Nature 323 (1986) 533–536.
[19] S. Scardapane, D. Wang, M. Panella, A. Uncini, Distributed learning for random vector functional-link networks, Inf. Sci. 301 (2015) 271–284.
[20] S. Scardapane, D. Wang, M. Panella, A decentralized training algorithm for echo state networks in distributed big data applications, Neural Netw. 78 (2016) 65–74.
[21] W.F. Schmidt, M.A. Kraaijveld, R.P. Duin, Feedforward neural networks with random weights, in: Proceedings of 11th IAPR International Conference on Pattern Recognition, in: Conference B: Pattern Recognition Methodology and Systems, volume II, 1992, pp. 1–4.
[22] I. Tyukin, D. Prokhorov, Feasibility of random basis function approximators for modeling and control, in: Proceedings of IEEE Multi-Conference on Systems and Control, Saint Petersburg, Russia, 2009, pp. 1391–1396.
[23] D. Wang, Editorial: randomized algorithms for training neural networks, Inf. Sci. 364–365 (2016) 126–128.