Information Sciences 000 (2015) 1–12

Contents lists available at ScienceDirect

Information Sciences

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

A comprehensive evaluation of random vector functional link networks

Le Zhang, P.N. Suganthan*

School of Electric and Electronic Engineering, NanYang Technological University, Singapore 639798, Singapore

ARTICLE INFO

Article history:
Received 14 May 2015
Revised 4 September 2015
Accepted 11 September 2015
Available online xxx

Keywords:
Random vector functional link networks
Ridge regression
Moore–Penrose pseudoinverse
Data classification

ABSTRACT

With randomly generated weights between input and hidden layers, a random vector functional link network is a universal approximator for continuous functions on compact sets with fast learning property. Though it was proposed two decades ago, the classification ability of this family of networks has not been fully investigated yet. Through a very comprehensive evaluation by using 121 UCI datasets, the effect of bias in the output layer, direct links from the input layer to the output layer and type of activation functions in the hidden layer, scaling of parameter randomization as well as the solution procedure for the output weights are investigated in this work. Surprisingly, we found that the direct link plays an important performance enhancing role in RVFL, while the bias term in the output neuron had no significant effect. The ridge regression based closed-form solution was better than those with Moore–Penrose pseudoinverse. Instead of using a uniform randomization in [−1,+1] for all datasets, tuning the scaling of the uniform randomization range for each dataset enhances the overall performance. Six commonly used activation functions were investigated in this work and we found that *hardlim* and *sign* activation functions degenerate the overall performance. These basic conclusions can serve as general guidelines for designing RVFL networks based classifiers.

1. Introduction

Single layer feedforward neural networks (SLFN) have been widely applied to solve problems such as classification and regression because of their universal approximation capability [14,17,20,31]. Conventional methods for training SLFN are back-propagation based learning algorithms [7,10]. These iterative methods suffer from slow convergence, getting trapped in a local minimum and being sensitivity to learning rate setting. Random Vector Functional Link Networks (RVFL), shown in Fig. 1, which is a randomized version of the functional link neural network network [8,25], shows that actual values of the weights from the input layer to hidden layer can be randomly generated in a suitable domain and kept fixed in the learning stage. Independently developed method in [35] also belongs to the family of randomized methods for training artificial neural networks with randomized input layer weights. This method [35] does not have direct links between the inputs and the outputs whereas RVFL has highly beneficial direct links.

RVFL was proposed in [28]. Learning and generalization characteristics of RVFL were discussed in [26]. In [17], Igelnik and Pao proved that the RVFL network is a universal approximator for a continuous function on a bounded finite dimensional set

* Corresponding author. Tel.: +65 670 5404; fax: +65 6793 3318.
E-mail address: epnsugan@ntu.edu.sg (P.N. Suganthan).

http://dx.doi.org/10.1016/j.ins.2015.09.025
0020-0255/© 2015 Elsevier Inc. All rights reserved.

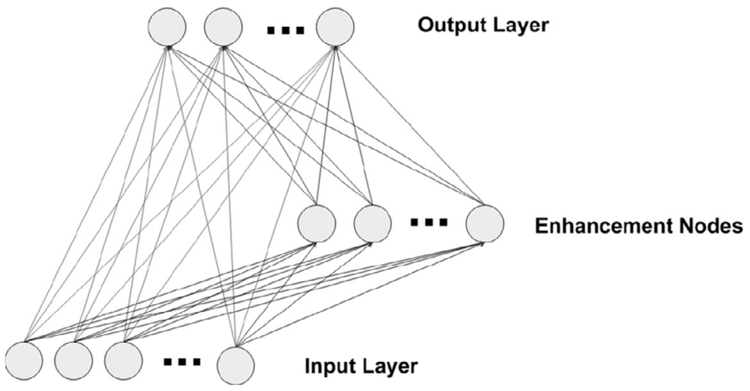

Fig. 1. The structure of RVFL. The input features are firstly transformed into the enhanced features by the enhancement nodes. Input weights and biases of the enhancement nodes are randomly generated. At the output layer, all the enhanced and original features are concatenated and fed into output neurons.

with a closed-form solution. From then on, RVFL has been employed to solve problems in diverse domains. A dynamic step-wise updating algorithm was proposed to update the output weights of the RVFL on-the-fly in [5] for both a new added pattern and a new added enhancement node. The RVFL network was investigated in [37] in the context of modeling and control. They [37] suggested to combine unsupervised placement of network nodes to the input data density with subsequent supervised or reinforcement learning of the linear parameters of the approximator. Modeling conditional probabilities with RVFL was reported in [15].

RVFL can also be combined with other learning methods. In [6], RVFL was combined with statistical hypothesis testing and self-organization of a number of enhancement nodes to generate a new learning system called a statistical self-organizing learning system (SSOLS) for remote sensing applications. In [16], expectation maximization was combined with RVFL to improve its performance. RVFL has also been investigated in ensemble learning framework. In [1], decorrelated RVFL ensemble was introduced based on the negative correlation learning. RVFL based multi-source data ensemble for clinker free lime content estimation in rotary kiln sintering processes can be found [21]. RVFL has also been widely applied to solve real-life problems. In [30], the authors reported the performance of a holistic-styled word-based approach to off-line recognition of English language script. Radial basis function neural net and RVFL were combined. Their approach, named as density-based random-vector functional-link net (DBRVFLN), was helpful in improving the performance of the word recognition. In [29], RVFL was used in MPEG-4 coder. In [38] RVFL was applied for pedestrian detection based on combination of multi-feature. In [39], RVFL was combined with Adaboost in the pedestrian detection system. In [23], the authors investigated the performance of hardware implementation methods for RVFL. In [34], distributed learning of RVFL was proposed where training data is distributed under a decentralized information structure.

Consider an RVFL as demonstrated in Fig. 1. As mentioned before, the weights a_{ij} from the input to the enhancement nodes are randomly generated such that the activation functions $g(a_j^T x + b_j)$ are not all saturated. Following the approach in [1], all the weights are generated with the a uniform distribution within $[-S, +S]$ in this work, where S is a scale factor to be determined during the parameter tuning stage for each dataset. For RVFL, only the output weights β need to the determined by solving the following problem:

$$t_i = d_i^T \beta, \quad i = 1, 2, \ldots, P \tag{1}$$

where P is the number of data samples, t is the target and d is the vector version of the concatenation of the original features as well as the random features.[1] Directly solving the problem in Eq. (1) may lead to over-fitting. In practice, a regularization on the solution such as regularized least square or preference of the solution with smaller norm [3] can be adopted to obtain the solution. RVFL can be roughly divided into 2 classes based on the algorithm to obtain the output weights. One is iterative RVFL, which obtains the output weights in an iterative manner based on the gradient of the error function. The other one is closed-form based RVFL, which obtains the output weights in a single-step. The present work focuses on the closed-form based RVFL because of its efficiency. A straightforward solution within a single learning step can be achieved by the pseudo-inverse [17,27], among which Moore–Penrose pseudoinverse, $\beta = D^{+}T$, where D and T are the matrix versions of the features and targets by stacking the features and targets of all data samples, is most commonly used. Another alternative is the $L2$ norm regularized least square (or ridge regression), which solves the following problem:

$$\sum_i (t_i - d_i^T \beta)^2 + \lambda \|\beta\|^2; i = 1, 2, \ldots P. \tag{2}$$

The solution is given by $\beta = D(D^T D + \lambda I)^{-1} T$, where λ is the regularization parameter to be tuned.

[1] For notational simplicity, we use the same formulation for all cases no matter whether there are biases in the output neurons since representing the features for the output neurons with $d = [d, 1]$ is equivalent to having a bias term in the output neurons.

Table 1
Activation functions used in this work. s and y are the inputs and outputs, respectively.

Activation function	Formulation
Sigmoid	$y = \frac{1}{1+e^{-s}}$
Sine	$y = sine(s)$
Hardlim	$y = (sign(s) + 1)/2)$
Tribas	$y = max(1 - \|s\|, 0)$
Radbas	$y = exp(-s^2)$
Sign	$y = sign(s)$

Though there are many RVFL variants in the literature, some core features of RVFL remain unchanged. In this work, We choose the closed-from based RVFL and the following issues are investigated by using 121 UCI datasets as done in [11].

1. Effect of direct links from the input layer to the output layer.
2. Effect of the bias in the output neuron.
3. Performance of 6 commonly used activation functions as summarized in Table 1.
4. Performance of Moore–Penrose pseudoinverse and ridge regression (or regularized least square solutions) for the computation of the output weights.
5. Effect of range for randomly generated parameters in hidden neurons.

Issues 1−4 in the above list are discussed in Section 2.3 while issue 5 is discussed in Section 2.5.

2. Evaluation protocol

2.1. Datasets

All 121 datasets are from the UCI repository [22]. The details of the datasets are summarized in Table 2.

We follow the same procedure as in [11]. Randomized stratified sampling is employed to make sure one training and one test set are generated (each with 50% of the available patterns), where each class has the same number of training and test patterns. Parameter tuning is performed on this couple of sets to identify parameters with the best performance on the test set. There are two parameters in the present work. One is the number of hidden neurons, which is tuned over 3: 203 with a step-size of 20 [11]. The other one is λ in ridge regression in Eq. (2), which is set to be $(1/2)^C$ and C is $-5 : 1 : 14$ [11]. Then, with the selected values for the tunable parameters, a 4-fold cross validation is developed using the whole data. However, for some datasets where the training-testing partition is already available (such as annealing and audiology-std, among others), the classifier is trained on the predefined training set and evaluated on the test set. In this case, the test result is calculated on the test set [11]. Each input feature is normalized by removing the mean value and dividing by its $l2$ norm.

2.2. Different RVFL configurations

We evaluate 48 different closed-form based RVFL configurations listed as follows:

1. RVFL with and without bias in the output neuron.
2. RVFL with and without direct link from input layer to output layer.
3. The performance of 6 commonly used activation functions as summarized in Table 1.
4. RVFL with Moore–Penrose pseudoinverse and RVFL with ridge regression.

2.3. Results and discussion

Due to page limits, the detailed accuracy for each method is omitted in this paper and it can be downloaded from the authors' homepage.[2] In the following, we summarize the overall performance of the variants based on their rank on each dataset. The most straightforward way to compare classifiers is to compute the average accuracies over all datasets. However, their averages are meaningless if the results on different datasets are not comparable. Moreover, averaging of accuracies is also susceptible to outliers. They allow a classifier's excellent performance on one dataset to compensate for poor performance. Further, a total failure on one problem can submerge fair results on most others [9]. Hence, in this study, we follow the method in [9], and use the rank of each classifier to reflect its performance [11]. This approach ranks the algorithms for each data set separately, the best performing algorithm getting the rank of 1, the second best rank 2..., average ranks are assigned in case of ties. Based on the 121 rankings, we summarize the overall ranking of each method in Table 3.

[2] http://www.ntu.edu.sg/home/EPNSugan/.

Table 2
Datasets used in this work.

Datasets	Patterns	Features	Classes
Abalone	4177	8	3
Ac-inflam	120	6	2
Acute-nephritis	120	6	2
Adult	48842	14	2
Annealing	798	38	6
Arrhythmia	452	262	13
Audiology-std	226	59	18
Balance-scale	625	4	3
Balloons	16	4	2
Bank	45211	17	2
Blood	748	4	2
Breast-cancer	286	9	2
Bc-wisc	699	9	2
Bc-wisc-diag	569	30	2
Bc-wisc-prog	198	33	2
Breast-tissue	106	9	6
Car	1728	6	4
Ctg-10classes	2126	21	10
Ctg-3classes	2126	21	3
Chess-krvk	28056	6	18
Chess-krvkp	3196	36	2
Congress-voting	435	16	2
Conn-bench-sonar	208	60	2
Conn-bench-vowel	528	11	11
Connect-4	67557	42	2
Contrac	1473	9	3
Credit-approval	690	15	2
Cylinder-bands	512	35	2
Dermatology	366	34	6
Echocardiogram	131	10	2
Ecoli	336	7	8
Energy-y1	768	8	3
Energy-y2	768	8	3
Fertility	100	9	2
Flags	194	28	8
Glass	214	9	6
Haberman-survival	306	3	2
Hayes-roth	132	3	3
Heart-cleveland	303	13	5
Heart-hungarian	294	12	2
Heart-switzerland	123	12	2
Heart-va	200	12	5
Hepatitis	155	19	2
Hill-valley	606	100	2
Horse-colic	300	25	2
Ilpd-indian-liver	583	9	2
Image-segmentation	210	19	7
Ionosphere	351	33	2
Iris	150	4	3
Led-display	1000	7	10
Lenses	24	4	3
Letter	20000	16	26
Libras	360	90	15
Low-res-spect	531	100	9
Lung-cancer	32	56	3
Lymphography	148	18	4
Magic	19020	10	2
Mammographic	961	5	2
Miniboone	130064	50	2
Molec-biol-promoter	106	57	2
Molec-biol-splice	3190	60	3
Monks-1	124	6	2
Monks-2	169	6	2
Monks-3	3190	6	2
Mushroom	8124	21	2
Musk-1	476	166	2
Musk-2	6598	166	2
Nursery	12960	8	5

(*continued on next page*)

Table 2 (*continued*)

Datasets	Patterns	Features	Classes
OocMerl2F	1022	25	3
OocMerl4D	1022	41	2
OocTris2F	912	25	2
OocTris5B	912	32	3
Optical	3823	62	10
Ozone	2536	72	2
Page-blocks	5473	10	5
Parkinsons	195	22	2
Pendigits	7494	16	10
Pima	768	8	2
Pb-MATERIAL	106	4	3
Pb-REL-L	103	4	3
Pb-SPAN	92	4	3
Pb-T-OR-D	102	4	2
Pb-TYPE	105	4	6
Planning	182	12	2
Plant-margin	1600	64	100
Plant-shape	1600	64	100
Plant-texture	1600	64	100
Post-operative	90	8	3
Primary-tumor	330	17	15
Ringnorm	7400	20	2
Seeds	210	7	3
Semeion	1593	256	10
Soybean	307	35	18
Spambase	4601	57	2
Spect	80	22	2
Spectf	80	44	2
St-aus-credit	690	14	2
St-german-credit	1000	24	2
St-heart	270	13	2
St-image	2310	18	7
St-landsat	4435	36	6
St-shuttle	43500	9	7
St-vehicle	846	18	4
Steel-plates	1941	27	7
Synthetic-control	600	60	6
Teaching	151	5	3
Thyroid	3772	21	3
Tic-tac-toe	958	9	2
Titanic	2201	3	2
Rains	10	28	2
Twonorm	7400	20	2
Vc-2classes	310	6	2
Vc-3classes	310	6	3
Wall-following	5456	24	4
Waveform	5000	21	3
Waveform-noise	5000	40	3
Wine	179	13	3
W-qua-red	1599	11	6
W-qua-white	4898	11	7
Yeast	1484	8	10
Zoo	101	16	7

Details of the datasets. Some keys are: ac-inam = acute-inammation, bc = breastcancer, congress-vot = congressional-voting, ctg = cardiotocography, conn-benchsonar/ vowel = connectionist-benchmark-sonar-mines-rocks/vowel-deterding, pb = pittsburg-bridges, st = statlog, aus = australian, vc = vertebral-column, w-qua = wine-quality.

In order to give a detailed analysis of the results, we follow the method in [9] to test the significance of their differences. The statistical test is based on Friedman test. The Friedman test [12,13] is a non-parametric equivalent of the repeated-measures ANOVA. It ranks the algorithms for each dataset separately, (the best performing algorithm getting the rank of 1, the second best rank 2 and so on), as shown in Table 3. In case of ties, average ranks are assigned. Let r_i^j be the rank of the jth of k algorithms on the ith of N datasets. The Friedman test compares the average ranks of algorithms, $R_j = \sum_i r_i^j$. The null-hypothesis states that all the algorithms are equivalent and so their ranks R_j should be equal. Let N and k denotes the number of algorithms and datasets

Table 3
Average rank values[1] based on classification accuracies of RVFL variants.

Solution & activation function		−bias, −link[2]	+bias, −link[3]	−bias, +link[4]	+bias, +link[5]	F value[6]
Ridge Regression	Sigmoid	2.7975	2.7025	2.3306	2.1694	6.7829
	Sine	2.5868	2.6033	2.4174	2.3926	0.8843
	Hardlim	3.0785	3.0702	1.9091	1.9421	43.0537
	Tribas	2.6901	2.6736	2.2810	2.3554	3.3333
	Radbas	2.6612	2.6405	2.3182	2.3802	2.2775
	Sign	3.0455	3.0372	1.9545	1.9628	36.7475
Moore–Penrose pseudoinverse	Sigmoid	2.5868	2.5620	2.3926	2.4587	0.5639
	Sine	2.5702	2.6612	2.2769	2.4917	1.9702
	Hardlim	3.0785	3.0785	1.9732	1.9298	43.8826
	Tribas	2.5620	2.6116	2.4008	2.4256	0.7646
	Radbas	2.5000	2.7066	2.4008	2.3926	1.5576
	Sign	3.0950	3.0289	1.9404	1.9256	40.6768

[1] Each row represents a distinct comparison. Lower value indicates higher accuracy.
[2] RVFL without bias and without direct link.
[3] RVFL with bias and without direct link.
[4] RVFL without bias and with direct link.
[5] RVFL with bias and with direct link.
[6] Statistic value derived from Eq. (4).

respectively, when N and k are large enough, the Friedman statistic.

$$\chi_F{}^2 = \frac{12N}{k(k+1)}\left[\sum_j R^2{}_j - \frac{k(k+1)^2}{4}\right], \tag{3}$$

is distributed according to $\chi_F{}^2$ with $k-1$ degrees of freedom under the null-hypothesis. In that case, Friedman's $\chi_F{}^2$ is undesirably conservative. A better statistic is

$$F_F = \frac{(N-1)\chi_F{}^2}{N(k-1)-\chi_F{}^2}, \tag{4}$$

which is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom.

If the null-hypothesis is rejected, which means the differences among the algorithms are statistically significant, the Nemenyi post-hoc test [24] can be used to check whether the performance of two among k classifiers is significantly different. If the corresponding average ranks of two different algorithms differ by at least the critical difference

$$CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}}, \tag{5}$$

the performances of them are considered as significantly different. In Eq. (5), critical values q_α are based on the Studentized range statistic divided by $\sqrt{2}$. α is the significance level and is set to be 0.05 in this work.

2.3.1. Effect of direct link and bias

We report in this section the effect of the direct link and bias in various RVFL configurations. In order to have reasonable comparisons, we keep other issues (the activation functions and the solution for the output weights) fixed. In Table 3, each column stands for an RVFL variant based on direct link and bias. Interestingly, we find that the superiority of the bias cannot be observed. However, the direct link is much more important than the bias term. It can be easily demonstrated that the $5th-6th$ columns have lower ranks than those in $3rd-4th$ columns. The direct link from the input layer to the output layer can serve as a regularization for the randomization thereby making the RVFL to achieve overall better performance than the RVFL variants without the direct link.

In our case ($k = 4, N = 121$), the critical values for the F distribution with 3 and 360 degrees of freedom is 2.6 and CD for the Nemenyi test is 0.4264. So, for those rows with F value larger than 2.6, their difference is statistically significant. Further in those rows, the pair of columns whose ranks differ by more than 0.4264 are statistically significantly different. In Table 4, we summarize the statistical significance of the rank differences for those rows where the F value is larger than 2.6.

2.3.2. Activation function

In this section, we give a detailed analysis of different activation functions. Following the similar procedure introduced in Section 2.3 to test the significance of difference of classifier's performance, it is easy to get the critical value for F distribution with 5 and 600 degrees of freedom as 2.21 and the CD for the following Nemenyi test as 0.6196.

From Table 5, we can see that the *radbas* activation function always achieves better performance in all cases. *hardlim* and *sign* activation functions lead to the penultimate worst and the worst performances, respectively. For Ridge Regression based methods, there is a clear pattern

$$radbas > sine > tribas > sig > hardlim > sign \tag{6}$$

Table 4
Result of statistical test.[1] Statistically significant differences are marked with –[2] or ++[3].

(a) Ridge regression with "sigmoid"

	–b,–d	+b,–d	–b,+d	+b,+d
–b,–d			––	––
+b,–d				––
–b,+d	++			
+b,+d	++	++		

(b) Ridge regression with "hardlim"

	–b,–d	+b,–d	–b,+d	+b,+d
–b,–d			––	––
+b,–d			––	––
–b,+d	++	++		
+b,+d	++	++		

(c) Ridge regression with "sign"

	–b,–d	+b,–d	–b,+d	+b,+d
–b,–d			––	––
+b,–d				––
–b,+d	++			
+b,+d	++	++		

(d) M—P[4] pseudoinverse with "hradlim"

	–b,–d	+b,–d	–b,+d	+b,+d
–b,–d			––	––
+b,–d				––
–b,+d	++			
+b,+d	++	++		

(e) M—P pseudoinverse with "sign"

	–b,–d	+b,–d	–b,+d	+b,+d
–b,–d			––	––
+b,–d				––
–b,+d	++			
+b,+d	++	++		

[1] "+" ,"-","b" and "d" have the same meaning as in Table 3.
[2] " ––" means the method in the row is statistically significantly worse than the method in the column.
[3] "++" means the method in the row is statistically significantly better than the method in the column.
[4] "M–P" stands for Moore–Penrose.

where ">" means that the method on the left performs better than the method on the right. However, for the Moore–Penrose pseudoinverse based methods, *tribas* always achieves the 4th rank. We also find that there is no clear superiority between *sigmoid* and *sine* activation functions.

2.3.3. Closed-form solution

Both the Ridge Regression and Moore–Penrose pseudoinverse lead to closed-form solution for RVFL. In order to investigate the effect of these two methods on different RVFL variants, we keep the activation functions and the network structure (bias, direct link) to be the same and compare the performances. Hence, it leads to 24 pairs of comparisons in total. Sign test [36] is employed to test the statistical significance since each comparison only involves two classifiers.

If the two algorithms compared are equivalent as assumed under the null-hypothesis, each should win on approximately $N/2$ out of N datasets. The number of wins is distributed according to the binomial distribution. For a greater number of datasets, the number of wins is under the null-hypothesis distributed according to $N(N/2, \sqrt{N}/2)$, which allows for the use of z-test: if the number of wins is at least $N/2 + 1.96\sqrt{N}/2$ (or, for a quick rule of a thumb, $N/2 + \sqrt{N}$), the algorithm is significantly better with $p < 0.05$. In this case, if one algorithm wins more than 71.28 times on 121 datasets, then it is considered as statistically significantly better than the other one. These cases are highlighted in Table 6. Table 6 summarizes the results for each pair of comparisons. The entry in each column represents the number of times ridge regression (pseudoinverse) is better than pseudoinverse (ridge regression) for the same activation function. For example, the first column means ridge regression is better than Moore–Penrose pseudoinverse in 59 of 121 datasets and worse in 57 of 121 datasets. Generally, ridge regression leads to a better performance for almost all cases.

2.4. Overall comparison

In this section, we present an overall comparison of the RVFL variants. Since we have already found that *hardlim* and *sign* activation functions consistently performed poorly, RVFL with these two activation functions will be excluded in this comparison. Hence, an overall comparison with 36 RVFL variants is presented in Table 7. In the same way, pairs of methods whose ranks differ by more than 4.5589 are statistically significantly different.

Table 5
Statistical significance test for different activation functions. The values in bracket stands for their average rank. Lower values stands for better performance. The mark √ indicates that these two methods are statistically significantly different.

(a) −bias, −link, Ridge regression

	sigmoid (3.18)	sine (2.67)	hardlim (4.83)	tribas (2.95)	radbas (2.48)	sign (4.89)
sigmoid (3.18)			√		√	√
sine (2.67)			√			√
hardlim (4.83)	√	√		√	√	
tribas (2.95)			√			√
radbas (2.48)	√		√			√
sign (4.89)	√	√		√	√	

(b) +bias, −link, Ridge regression

	sigmoid (3.16)	sine (2.69)	hardlim (4.80)	tribas (2.98)	radbas (2.49)	sign (4.88)
sigmoid (3.16)			√		√	√
sine (2.69)			√			√
hardlim (4.80)	√	√		√	√	
tribas (2.98)			√			√
radbas (2.49)	√		√			√
sign (4.88)	√	√		√	√	

(c) −bias, +link, Ridge regression

	sigmoid (3.45)	sine (2.91)	hardlim (4.36)	tribas (2.98)	radbas (2.76)	sign (4.55)
sigmoid (3.45)			√		√	√
sine (2.91)			√			√
hardlim (4.36)	√	√		√	√	
tribas (2.98)			√			√
radbas (2.76)	√		√			√
sign (4.55)	√	√		√	√	

(d) +bias, +link, Ridge regression

	sigmoid (3.37)	sine (2.98)	hardlim (4.37)	tribas (3.02)	radbas (2.71)	sign (4.54)
sigmoid (3.37)			√		√	√
sine (2.98)			√			√
hardlim (4.37)	√	√		√	√	
tribas (3.02)			√			√
radbas (2.71)	√		√			√
sign (4.54)	√	√		√	√	

(e) −bias, −link, Moore—Penrose pseudoinverse

	sigmoid (2.79)	sine (2.82)	hardlim (4.68)	tribas (3.36)	radbas (2.60)	sign (4.75)
sigmoid (2.79)			√			√
sine (2.82)			√			√
hardlim (4.68)	√	√		√	√	
tribas (3.36)			√		√	√
radbas (2.60)			√	√		√
sign (4.75)	√	√		√	√	

(f) +bias, −link, Moore—Penrose pseudoinverse

	sigmoid (2.73)	sine (2.96)	hardlim (4.68)	tribas (3.34)	radbas (2.59)	sign (4.69)
sigmoid (2.73)			√			√
sine (2.96)			√			√
hardlim (4.68)	√	√		√	√	
tribas (3.34)			√		√	√
radbas (2.59)	√		√			√
sign (4.69)	√	√		√	√	

(g) −bias,+link, Moore—Penrose pseudoinverse

	sigmoid (3.10)	sine (2.95)	hardlim (4.19)	tribas (3.56)	radbas (2.93)	sign (4.27)
sigmoid (3.10)			√			√
sine (2.95)			√			√
hardlim (4.19)	√	√		√	√	
tribas (3.56)			√			√
radbas (2.93)			√			√
sign (4.27)	√	√		√	√	

(h) +bias, +link, Moore—Penrose pseudoinverse

	sigmoid (3.17)	sine (3.05)	hardlim (4.08)	tribas (3.58)	radbas (2.99)	sign (4.12)
sigmoid (3.17)			√			√
sine (3.05)			√			√
hardlim (4.08)	√	√		√	√	
tribas (3.58)			√			√
radbas (2.99)			√			√
sign (4.12)	√	√		√	√	

2.5. Range of the random parameters

In [17], the authors indicate that the performance of the RVFL may depend on the ranges of uniformly distributed random weights. However, this issue has been untouched in the literature to the best of the authors' knowledge. In this work, we investigate this issue by introducing one scaling factor S to control the ranges of the randomization. This process can also shed light on the effect of saturation of hidden neurons in RVFL. Based on the performance in previous section, we choose ridge regression based RVFL with *radbas* activation function because it achieves the best performance among all variants.

In previously section, the random weights are generated with uniform distribution in $[-1,1]$, as done exactly in [35], while the biases are in $[0,1]$. In this section, the random weights and biases are generated with uniform distribution in $[-S, S]$ and $[0, S]$ respectively, where S is a positive scaling factor. In this work, we set $S = 2^t$, t is set to be $-5{:}0.5{:}5$. The performances 21 RVFL configurations with direct links and bias are summarized in Fig. 2. It is obvious that all RVFL variants perform poorly when the range of the random parameters becomes either too large or too small. Another interesting conclusion is the commonly adopted approach that $S = 1$ for the randomization may not lead to the optimal performance.

Table 6
Comparisons between ridge regression and Moore–Penrose pseudoinverse. The entry in each column represents the number of times ridge regression (pseudoinverse) is better than pseudoinverse (ridge regression) for the same activation function. Statistically significant columns are highlighted.

	(a) −bias, −link,					
	Sigmoid	Sine	Hardlim	Tribas	Radbas	Sign
Ridge Regression	59	61	*80*	*78*	63	*80*
Moore–Penrose pseudoinverse	57	56	*36*	*38*	54	*37*
	(b) +bias, −link,					
	Sigmoid	Sine	Hardlim	Tribas	Radbas	Sign
Ridge Regression	57	70	*81*	*77*	63	*81*
Moore–Penrose pseudoinverse	58	48	*36*	*40*	54	*37*
	(c) −bias, +link,					
	Sigmoid	Sine	Hardlim	Tribas	Radbas	Sign
Ridge Regression	64	57	68	*76*	58	63
Moore–Penrose pseudoinverse	52	56	46	*42*	59	51
	(d) +bias, +link,					
	Sigmoid	Sine	Hardlim	Tribas	Radbas	Sign
Ridge Regression	68	61	67	*78*	63	63
Moore–Penrose pseudoinverse	49	56	47	*39*	53	51

Table 7
Overall comparison based on overall ranks of 32 RVFL variants.[1] Lower values in rank reflects better performance.

Method	RR, −b,−d,sigmoid,	RR, +b,−d,sigmoid,	RR, −b,+d,sigmoid,	RR, +b,+d,sigmoid
Rank	18.3182	18.0579	17.0000	16.6860
method	RR, −b,−d,sine,	RR,+b,−d,sine,	RR,−b,+d,sine,	RR,+b,+d,sine
Rank	15.1157	15.2603	14.5455	14.9421
method	RR, −b,−d,tribas,	RR,+b,−d,tribas,	RR,−b,+d,tribas,	RR,+b,+d,tribas
Rank	16.7810	16.7479	15.1446	15.5702
method	RR, −b,−d,radbas,	RR,+b,−d,radbas,	RR,−b,+d,radbas,	RR,+b,+d,radbas
Rank	13.9545	14.0083	13.2686	13.4256
Method	MP, −b,−d,sigmoid,	MP, +b,−d,sigmoid,	MP, −b,+d,sigmoid,	MP, +b,+d,sigmoid
Rank	17.0124	16.7149	16.5124	16.9298
method	MP, −b,−d,sine,	MP,+b,−d,sine,	MP,−b,+d,sine,	MP,+b,+d,sine
Rank	17.3223	18.2066	15.7107	16.3678
method	MP, −b,−d,tribas,	MP,+b,−d,tribas,	MP,−b,+d,tribas,	MP,+b,+d,tribas
Rank	20.8636	20.9876	19.7934	20.1612
method	MP, −b,−d,radbas,	MP,+b,-d,radbas,	MP,−b,+d,radbas,	MP,+b,+d,radbas
Rank	15.4504	16.5785	15.0496	15.5124

[1] RR and MP stand for ridge regression and Moore–Penrose pseudoinverse, respectively. "+","-","b" and "d" are the same meaning as in Table 3.

Table 8
Average rank values of RVFL based on accuracies for different scale factor values over 121 datasets. Each column stands for a distinct comparison. Lower rank means higher accuracy. All RVFLs use ridge regression solution and *radbas* activation function.

Method	$S = 2^{-1.5}$	$S = 2^{-1}$	$S = 2^{-0.5}$	$S = 1$	$S = 2^{0.5}$	$S = 2^{1}$	$S = 2^{1.5}$
−b,−d	2.6446	2.5992	2.5992	2.6612	2.7231	2.6322	2.6653
+b,−d	2.6529	2.5785	2.6446	2.6405	2.6736	2.6860	2.7355
−b,+d	2.4298	2.4752	2.3140	2.3182	2.3430	2.4215	2.3182
+b,+d	2.2727	2.3471	2.4421	2.3802	2.2603	2.2603	2.2810

For RVFL without direct link, setting $t > 0$ for scaling factor S to increase the discrimination power of the features in the hidden neurons may make more neurons to saturate. This can be compensated by either having more hidden neurons or the direct link from the input layer to the output layer. On the other hand, setting $t < 0$ for scaling factor S to reduce the possibility of neuronal saturation may reduce the discrimination power of the features in the hidden neurons. Again, this can be compensated to some degree by having more hidden neurons or the direct link from the input layer to output layer. In Tables 8 and 9, we also present rank values based on accuracies and number of hidden neurons of different RVFL variants for different scaling factor S with $t = -1.5 : 0.5 : 1.5$ since their performance is consistently better than others as indicated in Fig. 2. The average rank values based on accuracies of RVFL when all parameters (N, λ, S) are tuned are presented in Table 10.

Results in Tables 8 and 10 are consistent with the previous subsections which clearly indicate the advantage of the direct link. Moreover, RVFL with direct link achieves better accuracy with less number of hidden neurons. The direct link from input layer

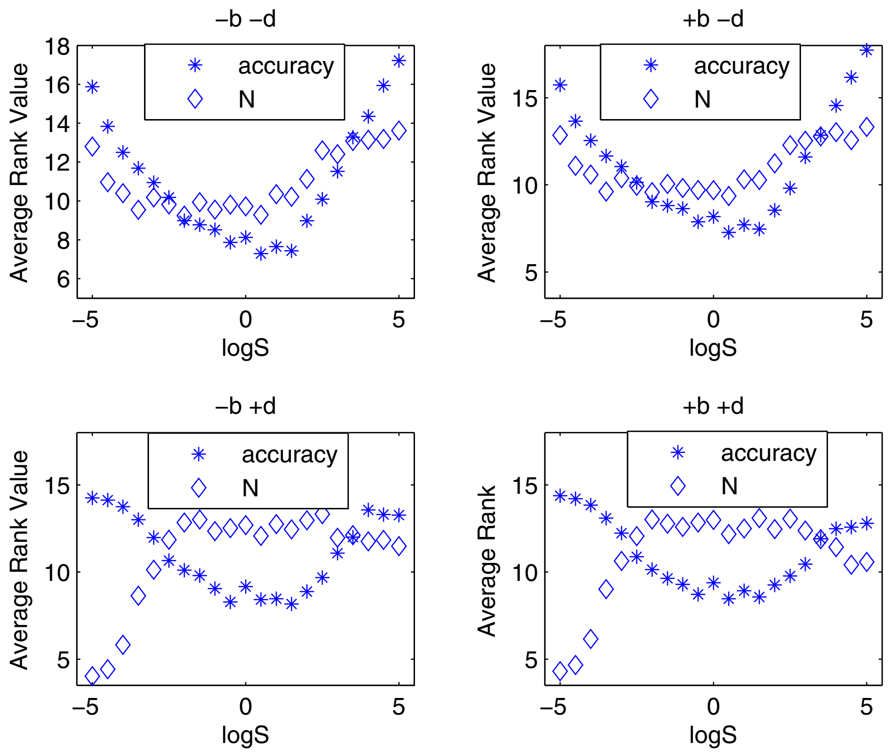

Fig. 2. Performance of RVFL based for different ranges of randomization. Smaller rank indicates better accuracy and less number of hidden neurons. *N* stands for the number of hidden neurons corresponding to the testing accuracy used in the ranking.

Table 9
Average rank values of RVFL based on the number of hidden neurons for different scale factor values over 121 datasets. Each column stands for a distinct comparison. Lower rank means less number of hidden neurons. All RVFLs use ridge regression solution and *radbas* activation function.

Method	$S = 2^{-1.5}$	$S = 2^{-1}$	$S = 2^{-0.5}$	$S = 1$	$S = 2^{0.5}$	$S = 2^{1}$	$S = 2^{1.5}$
−b,−d	2.6818	2.6529	2.6322	2.5868	2.6653	2.6736	2.6942
+b,−d	2.6488	2.7025	2.5785	2.5785	2.6488	2.6570	2.6281
−b,+d	2.3760	2.3140	2.3760	2.4091	2.3636	2.3926	2.3099
+b,+d	2.2934	2.3306	2.4132	2.4256	2.3223	2.2769	2.3678

Table 10
Average rank values of RVFL based on accuracies when all parameters (N, λ, S) are tuned. Lower rank means higher accuracy. All RVFLs use ridge regression solution and *radbas* activation function.

	−b,−d	+b,−d	−b,+d	+b,+d
Rank	2.8140	2.6653	2.3430	2.1777

to output layer severs as a standing out regularization and make a high chance for RVFL to achieve a better performance with high possibility than those without direct link. That is, with smaller number of random hidden neurons, direct link in RVFL leads to a thinner and simpler model than those without. For a given set of observations or data, there is always an infinite number of possible hypotheses fit the same data. According to the Occams Razor principle, one should choose from a set of otherwise equivalent models of a given phenomenon the simplest one. For example, it is possible for us to further increase the performance of RVFL if we enlarg the number of random hidden neurons. “Super flat” RVFL models (i.e. with a large number of hidden neurons) are more likely to overfit the available data. This is also in line with the statistical learning theory [18] that advocates for learning with lower complexity models. Hence, the tuning range for the number of hidden neurons can be relatively narrower for RVFL with direct links.

3. Concluding remarks

In this work we presented extensive and comprehensive evaluation of variants of RVFL with closed-form solution by using 121 UCI datasets [11]. The conclusion of our investigations are as follows:

1. the effect of the direct links from the input layer to the output layer. It turns out that the direct links lead to better performance than those without in all cases as seen in Table 3.
2. the effect of the bias in the output layer. It turns out that the bias term in the output neurons only has mixed effects on the performance, as it may or may not improve performance. Hence, bias can be a tunable network configuration depending on the specific problem.
3. effect of scaling the randomization range of input weights and biases. We show scaling down the randomization range of input weights and biases to avoid saturating the neurons may risk at degenerating the discrimination power of the random features. However, this can be compensated by having more hidden neurons or direct link. Scaling the randomization range of input weights and biases up to enhance the discrimination power of the random features may risk saturating the neurons. Again, this can be compensated by having more hidden neurons or combining with the direct link from the input to the output layer. However, for reasons explained in Section 2.5, we prefer lower model complexity.
4. the performance of 6 commonly used activation functions summarized in Table 1. It turns out that *radbas* function always leads to a better performance. *hardlim* and *sign* activation functions lead to penultimate worst and worst performances, respectively.
5. the performance of Moore–Penrose pseudoinverse and ridge regression (or regularized least square) solutions for the output weights. It turns out that with one more parameter (λ in Eq. (2)) to tune, ridge regression based RVFL shows better performance than the Moore–Penrose pseudoinverse based RVFL.

This work sets a basis for future research for random vector functional link network. Future studies and developments of RVFL may include:

1. performance of RVFL ensemble. Neural network has low bias and high variance [4]. Hence, the performance of RVFL can be significantly improved by ensemble methods.
2. performance of kernel methods with RVFL. Recent research [2,32,33] shows the success of random features for large-scale kernel machines. Hence, it is worthy to investigate kernel machines with random features extracted from RVFL.
3. performance of deep RVFL structure. Recent work in computer vision and machine learning community demonstrates the success of deep neural networks [19]. Hence, how to design a good deep RVFL structure for a specific problem is an open problem now.

Acknowledgment

The authors would like to thank the Guest Editors and the reviewers for their valuable comments. In particular, authors thank the managing Guest Editor Associate Professor Dianhui Wang for suggesting us to investigate the scaling of randomization. Results presented in Section 2.5 show overall performance enhancement due to tuning the scaling of randomization.

References

[1] M. Alhamdoosh, D. Wang, Fast decorrelated neural network ensembles with random weights, Inf. Sci. 264 (2014) 104–117.
[2] S. An, W. Liu, S. Venkatesh, Face recognition using kernel ridge regression, in: Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2007, pp. 1–7.
[3] P.L. Bartlett, 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. Theor. 44 (2) (1998) 525–536.
[4] L. Breiman, Arcing classifier (with discussion and a rejoinder by the author), The Ann. Stat. 26 (3) (1998) 801–849.
[5] C.P. Chen, J.Z. Wan, A rapid learning and dynamic stepwise updating algorithm for flat neural networks and the application to time-series prediction, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 29 (1) (1999) 62–72.
[6] H.-M. Chi, O.K. Ersoy, A statistical self-organizing learning system for remote sensing classification, IEEE Trans. Geosci. Remote Sens. 43 (8) (2005) 1890–1900.
[7] L. Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Handwritten digit recognition with a back-propagation network, in: Advances in Neural Information Processing Systems, 1990, pp. 396–404.
[8] S. Dehuri, S.-B. Cho, A comprehensive survey on functional link neural networks and an adaptive pso–bp learning for cflnn, Neural Comput. Appl. 19 (2) (2010) 187–205.
[9] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[10] J.S. Denker, W. Gardner, H.P. Graf, D. Henderson, R. Howard, W. Hubbard, L.D. Jackel, H.S. Baird, I. Guyon, Neural network recognizer for hand-written zip code digits, in: Advances in Neural Information Processing Systems, 1989, pp. 323–331.
[11] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. 15 (1) (2014) 3133–3181.
[12] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. American Stat. Assoc. 32 (200) (1937) 675–701.
[13] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Ann. Math. Stat. 11 (1) (1940) 86–92.
[14] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[15] D. Husmeier, J.G. Taylor, Modelling conditional probabilities with committees of RVFL networks, in: Artificial Neural Networks–ICANN'97, Springer, 1997, pp. 1053–1058.
[16] D. Husmeier, J.G. Taylor, Neural networks for predicting conditional probability densities: improved training scheme combining EM and RVFL, Neural Netw. 11 (1) (1998) 89–116.

[17] 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.
[18] M.J. Kearns, U.V. Vazirani, An Introduction to Computational Learning Theory, MIT press, 1994.
[19] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Procedings of Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[20] M. Leshno, V.Y. Lin, A. Pinkus, S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural Netw. 6 (6) (1993) 861–867.
[21] W. Li, D. Wang, T. Chai, Multisource data ensemble modeling for clinker free lime content estimate in rotary kiln sintering processes, IEEE Trans. Syst. Man Cybern.: Syst. 45 (2) (2015) 303–314, doi:10.1109/TSMC.2014.2332305.
[22] M. Lichman, UCI machine learning repository, 2013, (http://archive.ics.uci.edu/ml).
[23] J.M. Martínez-Villena, A. Rosado-Muñoz, E. Soria-Olivas, Hardware implementation methods in random vector functional-link networks, Appl. Intell. 41 (1) (2014) 184–195.
[24] P. Nemenyi, Distribution-free Multiple Comparisons, Princeton University, 1963.
[25] Y. Pao, Adaptive pattern recognition and neural networks, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.
[26] 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.
[27] Y.-H. Pao, S.M. Phillips, The functional link net and learning optimal control, Neurocomputing 9 (2) (1995) 149–164.
[28] Y.-H. Pao, S.M. Phillips, D.J. Sobajic, Neural-net computing and the intelligent control of systems, Int. J. Control 56 (2) (1992) 263–289.
[29] G.H. Park, Y.J. Lee, S.R. LeClair, Intelligent rate control for MPEG-4 coders, Eng. Appl. Artif. Intell. 13 (5) (2000) 565–575.
[30] G.H. Park, Y.H. Pao, Unconstrained word-based approach for off-line script recognition using density-based random-vector functional-link net, Neurocomputing 31 (1) (2000) 45–65.
[31] J. Park, I.W. Sandberg, Universal approximation using radial-basis-function networks, Neural Comput. 3 (2) (1991) 246–257.
[32] A. Rahimi, B. Recht, Random features for large-scale kernel machines, in: Advances in Neural Information Processing Systems, 2007, pp. 1177–1184.
[33] C. Saunders, A. Gammerman, V. Vovk, Ridge regression learning algorithm in dual variables, in: Proceedings of the 15th International Conference on Machine Learning, Morgan Kaufmann, 1998, pp. 515–521.
[34] S. Scardapane, D. Wang, M. Panella, A. Uncini, Distributed learning for random vector functional-link networks, Inf. Sci. 301 (2015) 271–284.
[35] W.F. Schmidt, M. Kraaijveld, R.P. Duin, Feedforward neural networks with random weights, in: Proceedings of the 11th IAPR International Conference on Pattern Recognition, IEEE, 1992, pp. 1–4.
[36] D.J. Sheskin, Handbook of parametric and nonparametric statistical procedures, CRC Press, 2003.
[37] I. Tyukin, D. Prokhorov, Feasibility of random basis function approximators for modeling and control, in: Proceedings of IEEE International Symposium on Intelligent Control, 2009, pp. 1391–1396.
[38] Z. Wang, S. Yoon, S.J. Xie, Y. Lu, D.S. Park, Random vector functional-link net based pedestrian detection using multi-feature combination, in: Proceedings of the 6th International Congress on Image and Signal Processing (CISP), 2, IEEE, 2013, pp. 773–777.
[39] Z. Wang, S. Yoon, S.J. Xie, Y. Lu, D.S. Park, A high accuracy pedestrian detection system combining a cascade adaboost detector and random vector functional-link net, The Scientific World J. 2014 (2014).