Neural Process Lett
DOI 10.1007/s11063-016-9566-2

A New Sparse Learning Machine

Mojtaba Nayyeri[1] · Alaleh Maskooki[2] · Reza Monsefi[1]

Abstract Many algorithms have been proposed so far for pruning and sparse approximation of feedforward neural networks with random weights in order to obtain compact networks which are fast and robust on various datasets. One drawback of the randomization process is that the resulted weight vectors might be highly correlated. It has been shown that ensemble classifiers' error depends on the amount of error correlation between them. Thus, decrease in correlation between output vectors must lead to generation of more efficient hidden nodes. In this research a new learning algorithm called New Sparse Learning Machine (NSLM) for single-hidden layer feedforward networks is proposed for regression and classification. In the first phase, the algorithm creates hidden layer with small correlation among nodes by orthogonalizing the columns of the output matrix. Then in the second phase, using L_1-norm minimization problem, NSLM makes the components of the solution vector become zero as many as possible. The resulted network has higher degree of sparsity while the accuracy is maintained or improved. Therefore, the method leads to a new network with a better generalization performance. Numerical comparisons on several classification and regression datasets confirm the expected improvement in comparison to the basic network.

Keywords Feedforward neural networks · Sparse learning machine · Approximation algorithm · L_1-norm minimization · Error correlation · Gram–Schmidt orthogonalization

✉ Reza Monsefi
monsefi@um.ac.ir

Mojtaba Nayyeri
mojtaba.nayyeri@alumni.um.ac.ir

Alaleh Maskooki
maskooki@mail.um.ac.ir

1 Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran

2 Department of Applied Mathematics, Ferdowsi University of Mashhad, Mashhad, Iran

Published online: 25 November 2016

1 Introduction

Single Hidden Layer Feedforward Networks (SLFNs) are capable of approximating complex nonlinear mappings arbitrarily well and thus are extensively used in various applications including classification and regression problems. Conditions, under which universal approximation of feedforward networks is valid, have been investigated in several literatures. Hornik [1] proved that feedforward networks with additive hidden nodes and continuous, bounded and non-constant activation function approximate any continuous mapping to any desired degree of accuracy over compact input sets. An extension of the above theorem is proposed by Leshno et al. [2] which states that a multilayer feedforward network with additive nodes and locally bounded piecewise continuous activation function can approximate any continuous function if and only if the network's activation function is non-polynomial. In addition, the universal approximation capability is generalized for randomized feedforward networks in some research works [3–5] and is investigated for a network trained with N distinct samples [6].

A major issue appears when using most neural network based methods, is choosing the appropriate number of hidden nodes which should be determined by trial and error process, since too few or too many hidden nodes lead to underfitting or overfitting respectively [7]. To improve generalization and avoid this issue, a large network is trained and unnecessary nodes are pruned during learning. In fact, the size of the network is automatically determined by the algorithm with respect to the desired accuracy. Several methods are proposed to create a sparse network. Sietsma and Dow [8] discussed neural network pruning methods in detail. Lazarevic et al. [9] presented several pruning algorithms in ensemble neural networks that remove redundant classifiers via identifying the groups of similar classifiers. Furthermore, Setiono [10] proposed a penalty function with two terms that helps prune unnecessary weights using simple criteria. In another research work [11], he presented a new pruning algorithm for neural network and obtained high accuracy rate for breast cancer diagnosis. Huang et al. [12] proposed a generalized growing and pruning Radial Basis Function (RBF) neural network for function approximation. In addition, Rong et al. [13] presented a pruning algorithm for Feedforward Neural Networks (FNN) with random weights using statistical methods for measuring the relevance of hidden nodes in contributing to the prediction accuracy of the classifier and prone/keep the irrelevant/relevant nodes, to obtain a compact network which is fast and robust on unseen data. Alcin et al. [14] introduced several sparse schemes of the randomized FNNs in which various greedy algorithms are used for sparse approximation of the output weight vector and investigated several greedy algorithms to design an efficient network being free of parameter adjustment and avoiding the singularity problem.

Recently, the study of L_1-norm penalty attracted significant attentions due to its ability for obtaining sparse models. 1-norm extreme learning machine (1-norm ELM), proposed by Balasundaram et al. [15], is a recent sparse model representation for regression and multiclass classification based on a linear programming problem whose solution is obtained by solving its dual exterior penalty problem using a fast Newton method. Sakar and Mammone [16] has formerly proposed a heuristic learning algorithm for growing neural tree networks based on minimizing the L_1-norm of the error and a pruning algorithm to improve the generalization performance, in pattern classification problem. They showed that L_1-norm minimization has better performance in terms of reducing the number of classification errors than the squared error minimization method used in backpropagation.

Seen from another perspective, diverse sets of classifiers are shown to outperform single predictors in multiple classifier systems. Brown et al. [17], reviewed existing explanation of

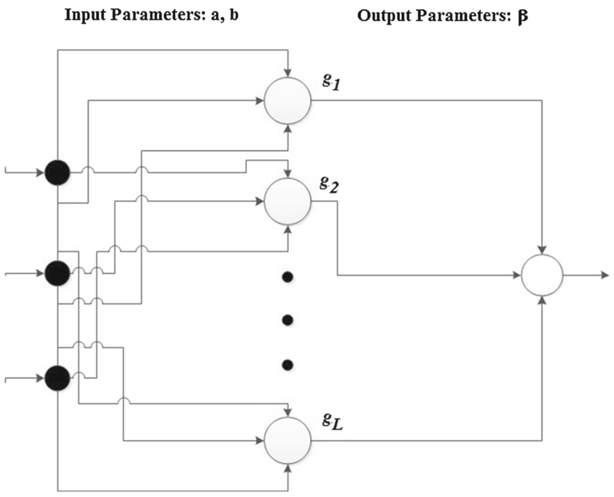

Fig. 1 Structure of a single hidden layer feedforward network: input layer, hidden layer, output layer

why ensembles with diverse errors perform well, including literature for both the regression and classification cases. They suggested that, the mean square error of ensemble classifiers depends on the amount of error correlation between individual units, quantified in terms of covariance. Based on this, we would ideally like the correlation between classifiers' outputs to be small. In order to obtain low (or even zero) correlation, diversity should be increased [18]. Tumer and Ghosh [19] quantified the need to reduce correlation between individual classifiers especially for a limited training data.

In this research, a New Sparse Learning Machine (NSLM) method is presented. In the next sections, we briefly state the primal linear programming ELM introduced by Balasundaram et al. [15] as well as the method used for obtaining its solution. The proposed NSLM is introduced in Sect. 3. Experimental results for accuracy and sparsity level of NSLM in comparison to the original 1-norm ELM on several regression and classification problems are illustrated in Sect. 4. Finally, Sect. 5 concludes this research.

2 1-norm Extreme Learning Machine (1-norm ELM)

Suppose $\{(x_i, t_i)\}_{i=1,\dots,N}$ be the set of N training samples, where $x_i = (x_{i1}, \dots, x_{id})^T \in \mathbb{R}^d$ is the ith input vector and $t_i \in \mathbb{R}$ is its desired output. The input weight vectors $a_j = (a_{j1}, \dots, a_{jd})^T \in \mathbb{R}^d$ and biases $b_j \in \mathbb{R}$ connect the input layer to the jth hidden node. Likewise, the output weight vector $(\beta_1, \dots, \beta_L) \in \mathbb{R}^L$ connects the L nodes of the hidden layer to the output node. Figure 1 illustrates the general structure of a single hidden layer feedforward neural network. The goal is to determine weight vectors such that the following equation holds:

$$t_i = \sum_{j=1}^{L} \beta_j . g(a_j, b_j, x_i) \quad \forall i = 1, \dots, N \tag{1}$$

where $g(a_j, b_j, x_i)$ is the output value of the jth hidden node for the input sample x_i. The equation system (1) can be expressed in a matrix form as follows:

$$H\beta = T \tag{2}$$

where

$$H = \begin{bmatrix} g(a_1, b_1, x_1) & \cdots & g(a_L, b_L, x_1) \\ \vdots & \ddots & \vdots \\ g(a_1, b_1, x_N) & \cdots & g(a_L, b_L, x_N) \end{bmatrix} \tag{3}$$

is the output matrix of the hidden layer and $T = (t_1, ..., t_N)^T \in \mathbb{R}^N$ is the desired output vector of the network.

From the view point of the network architecture, nodes in the hidden layer can be additive whose output value is determined by

$$g_j(x) = g\left(\langle a_j, x\rangle + b_j\right)$$

or RBF nodes with the following output

$$g_j(x) = g\left(\frac{||x - a_j||}{b_j}\right)$$

1-norm ELM with absolute loss, proposed by Balasundaram et al. [15] as a robust and sparse iterative method for regression and classification, is stated as follows: Given a training set of N samples, activation function g, L hidden nodes and the output vector T, the problem 1-norm ELM with absolute loss is formulated as the following minimization problem:

$$\min_{\beta\in\mathbb{R}^L} C\|H\beta - T\|_1 + \|\beta\|_1 \tag{4}$$

where $C > 0$ is a constant. Given $\beta = r - s$ such that $r, s \in \mathbb{R}^L$, and $H\beta - T = p - q$ such that $p, q \in \mathbb{R}^N$, the problem (4) is equivalent to solving the following linear programming problem:

$$\min_{r,s\in\mathbb{R}^L,\, p,q\in\mathbb{R}^N} e_L^T(r + s) + Ce_N^T(p + q) \tag{5}$$

subject to:

$$\begin{aligned} &H(r - s) - p + q = T \\ &r, s, p, q \geq 0 \end{aligned}$$

where e_L and e_N are the column vectors of ones of dimension L and N respectively.

Although the problem (5) is feasible and bounded and thus solvable, Balasundaram et al. [15] obtained the solution by solving its dual exterior penalty problem using Newton-Armijo algorithm, due to increase in number of unknowns and constraints of the primal form.

They showed that there exists $\bar{\theta} > 0$ such that for any $\theta \in (0, \bar{\theta}]$ the following equations:

$$\begin{aligned} r &= \frac{1}{\theta}(H^T u - e_L)_+, \quad s = \frac{1}{\theta}(-H^T u - e_L)_+ \\ p &= \frac{1}{\theta}(-u - Ce_N)_+, \quad q = \frac{1}{\theta}(u - Ce_N)_+ \end{aligned}$$

generate an exact solution to the primal problem (5), where u is the solution of the corresponding dual exterior penalty problem and x_+ is a vector whose ith component is $\max\{0, x_i\}$. For further details we refer the reader to [15].

The main advantage of using 1-norm ELM is that it leads to a sparse model in the sense that many components of the optimal output vector will become zero and thus there are much less number of hidden nodes in the network in comparison to randomized FNN.

3 The Proposed NSLM Method

In this section, a new learning algorithm called new sparse learning machine (NSLM) for single-hidden layer feedforward networks is proposed based on the notions discussed in the previous section which uses independent nodes in the hidden layer of the network for obtaining optimal weight vectors.

In practice, when the parameters of the hidden layer are created randomly, there might be a large number of hidden nodes that are highly correlated, which leads to inefficiency in the network. The main goal of NSLM is to take advantage of any potential sparsity by reducing the correlation between classifiers' outputs to improve generalization of the network.

In contrast to 1-norm ELM method in which the input weight vectors and the biases are randomly assigned at the beginning of the learning algorithm and remain fixed in the optimal network, in NSLM the input weights and biases are optimized through a least squares problem so as to estimate a diverse output set. The hidden layer output matrix H in 1-norm ELM method inherits the randomness characteristic from the input weights and thus is random. The idea of our proposed method is to transform the random output vectors of hidden nodes into a diverse set by orthogonalizing the columns of matrix H using Gram–Schmidt algorithm [23]. It can be shown that the matrix produced by the Gram–Schmidt orthogonalization algorithm generates the equal subspace to that of the original random matrix H in 1-norm ELM. The orthogonalizing procedure leads to an ideal decrease in correlation between output vectors of hidden nodes. Computational experiments on various regression and classification datasets confirm that, this method eventually results in an increase in sparsity level whereas the accuracy of the original network is maintained or improved.

3.1 Input Weights Estimation

At the beginning of the learning process, the input weight vector and the bias of the first hidden node are optimized.

Let $X = [X_1, ..., X_N]_{(d+1)\times N}$ be the matrix of input samples where $X_i = (x_{i1}, ..., x_{id}, 1)^T$, $\forall i \in \{1, ..., N\}$ and the desired output be the following row vector:

$$h_1(X) = [h_1(X_1), ..., h_1(X_N)] = \frac{1}{T_{max}}[t_1, ..., t_N]_{1\times N} \tag{6}$$

where $T_{max} = \max_{i=1,...,N} \{|t_i|\}$. We determine the optimal input weight vector $w_1^* = (a_{11}, ..., a_{1d}, b_1)$ of the first hidden node using the following optimization problem whose objective is to minimize the squared residual error along with a Tikhonov regularization term [22]:

$$\min_{w_1^T \in \mathbb{R}^{d+1}, \xi^T \in \mathbb{R}^N} \frac{1}{2}||w_1||_2^2 + \frac{1}{2}C_{in}||\xi||_2^2 \tag{7}$$

subject to:

$$w_1 X = g^{-1}(h_1(X)) - \xi$$

where $C_{in} > 0$ is a constant and ξ is the residual error vector. The optimal input weights of the first node w_1^* is generated using problem (7) such that its output vector $g(w_1^* X)$ has almost the same direction as target vector T in order to reduce the error as much as possible. As the length of the output vector can be adjusted later by the output weight β, we can multiply the target vector T by the positive constant value $\frac{1}{T_{max}}$ in this step such that it is restricted to the domain of the function $g^{-1}(.)$ which is supposed to be the inverse hyperbolic tangent in this research. Finally, including the Tikhonov regularization term in the objective function, we minimize the squared residual error as well as the input weight vector as much as possible. So, w_1^* will be adjusted such that $g(w_1^* X) \approx \frac{1}{T_{max}} T$. The constant value C_{in} in problem (7) is a regularization term for optimizing the effect of the two terms in the objective function and will be approximated by trial and error.

In order to solve (7), the method of Lagrange multipliers [21] is used. The Lagrangian is:

$$L(w_1, \xi, \alpha) = \frac{1}{2} w_1 w_1^T + \frac{1}{2} C_{in} \xi \xi^T - \left(w_1 X - g^{-1}(h_1(X)) + \xi \right) \alpha \tag{8}$$

where $\alpha \in \mathbb{R}^N$ is the Lagrange multiplier. Thus, the following system of equations holds:

$$\frac{\partial L}{\partial w_1} = 0, \quad \frac{\partial L}{\partial \xi} = 0, \quad \frac{\partial L}{\partial \alpha} = 0 \tag{9}$$

which yields:

$$w_1 = \alpha X^T, \quad C_{in} \xi = \alpha, \quad w_1 X - g^{-1}\left(h_1(X)\right) + \xi = 0 \tag{10}$$

Substituting the first two equations of (10) into the last equation yields:

$$w_1 = g^{-1}\left(h_1(X)\right) \left(X^T X + \frac{1}{C_{in}} I \right)^{-1} X^T \tag{11}$$

Similarly, substituting the second equation of (10) into the first one and then using the last equation yields:

$$w_1 = g^{-1}\left(h_1(X)\right) X^T \left(X X^T + \frac{1}{C_{in}} I \right)^{-1} \tag{12}$$

In order to increase the calculation speed, Eq. (11) or (12) is used when $N < d$ or $N > d$ respectively.

Therefore, the optimal weight vector of the first hidden node is obtained using Eq. (11) or (12). Before calculating the other $L - 1$ optimal weight vectors, a matrix $H_{N \times L}$ is generated as defined in (2) using $g(w_1^* X)^T$ as the first column and $g(w_j X)^T$ as the jth column of the matrix, where $w_j \in \mathbb{R}^{d+1}$, $j \in \{2, ..., L\}$ are random input weight vectors. Afterwards, using Gram-Schmidt orthogonalization algorithm [23], a matrix $\bar{H}_{N \times L} = [v_1, ..., v_L]$ with a set of L orthogonal columns v_j, $j \in \{1, ..., L\}$ is derived from H. The algorithm is stated in Table 1.

It is known that the columns of matrix $\bar{H}$ produced by the Gram-Schmidt algorithm, generates the equal subspace to that of H . In the next step, the goal is to reproduce the remaining $L - 1$ random hidden nodes such that they are as diverse as possible and span the same space as 1-norm ELM.

Similar to the definition (6) for estimating the input vector corresponding to the first hidden node, let the desired output of the jth hidden node be the following row vector:

$$h_j(X) = [h_j(X_1), ..., h_j(X_N)] = \frac{1}{v_{jmax}} [v_{j1}, ..., v_{jN}]_{1 \times N}, \quad \forall j = 2, ..., L \tag{13}$$

where $v_{jmax} = \max_{i=1,...,N} \{|v_{ji}|\}$ and v_{ji} is the ith component of the column vector v_j. Following the same process for calculating w_1^*, the optimal input weight vectors $w_j^* = (a_{j1}, ..., a_{jd}, b_j)$, $j \in \{2, ..., L\}$ are obtained via replacing w_1 by w_j and $h_1(X)$ by $h_j(X)$ in problem (7). Replacing the desired output vector of the jth hidden node by the corresponding modified column of $\bar{H}$ in problem (7), we can determine the optimal input parameters of jth hidden node w_j^* such that $g(w_j^* X)$ estimates the direction of v_j.

Table 1 Gram-Schmidt orthogonalization algorithm

```
Input an arbitrary basis $\{u_1, ..., u_L\}$ for an $L$-dimensional space $V$.
Output an orthogonal basis $\{v_1, ..., v_L\}$ for $V$.
– Let $v_1 = u_1$
– For $j = 2$ to $L$ do
–     Let $\sigma$ be a zero vector
–     For $k = 1$ to $j - 1$ do
–         Put $\sigma = \sigma + \frac{u_j^T v_k}{||v_k||^2} v_k$
–     End
–     Put $v_j = u_j - \sigma$
– End
```

According to the Theorem 2.1 in [24], column vectors of H defined by (2) is theoretically capable of spanning the whole solution space. However in practice, it might fail to do so since input parameters of hidden nodes are obtained by simulating the continuous probability distribution function based on finite samples of MATLAB or other software. Therefore, H might contain columns which are highly correlated. Using Gram-Schmidt process, we determine the optimal input weights w_j^*, $j \in \{1, ...L\}$ to produce L independent column vectors of the optimal output matrix H^* defined in (14) including one column having a close direction to the target vector. Taking advantage of L_1 norm and the above mentioned properties of H^*, the method leads to a more compact network architecture.

3.2 Output Weights Estimation

In this phase, the 1-norm ELM with absolute loss [15] is applied to estimate the output weight vector β of the network using the following output matrix:

$$H^* = \begin{bmatrix} g(w_1^* X_1) & \cdots & g(w_L^* X_1) \\ \vdots & \ddots & \vdots \\ g(w_1^* X_N) & \cdots & g(w_L^* X_N) \end{bmatrix} \tag{14}$$

where w_j^*, $j \in \{1, ..., L\}$ are the optimal input vectors obtained from the previous phase. The optimal output weight vector β is obtained from the following problem:

$$\min_{\beta \in \mathbb{R}^L} C_{out} ||H^* \beta - T||_1 + ||\beta||_1 \tag{15}$$

Regarding the method used for problem (4), optimal $\beta = r - s$ for (15) is obtained through solving its corresponding dual exterior penalty problem using Newton-Armijo algorithm. The steps of the proposed NSLM algorithm are presented in Fig. 2. In the next section, we will show that substituting the random matrix H in 1-norm ELM by H^* defined above yields an output vector with a higher degree of sparsity in comparison to the original method while avoiding underfitting.

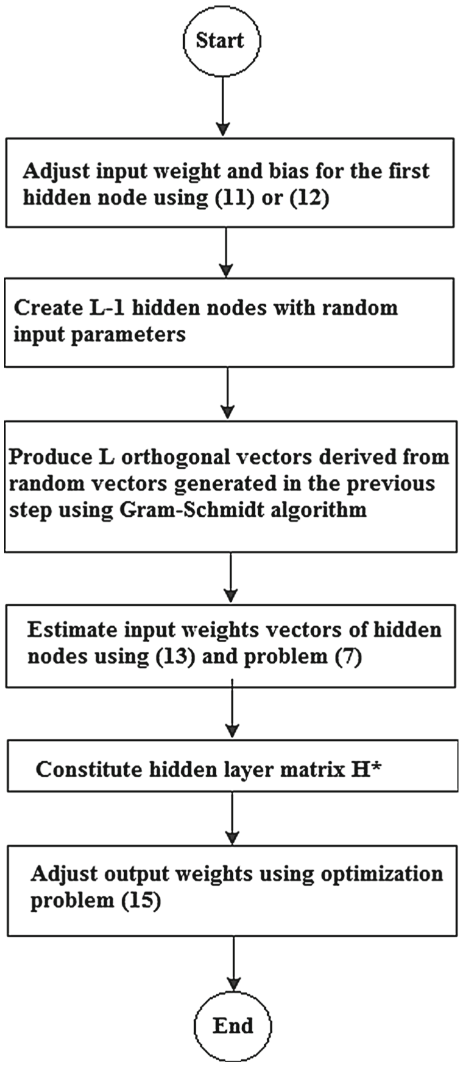
Start
Adjust input weight and bias for the first hidden node using (11) or (12)
Create L-1 hidden nodes with random input parameters
Produce L orthogonal vectors derived from random vectors generated in the previous step using Gram-Schmidt algorithm
Estimate input weights vectors of hidden nodes using (13) and problem (7)
Constitute hidden layer matrix H*
Adjust output weights using optimization problem (15)
End

Fig. 2 The flowchart of NSLM algorithm

4 Computational Experiments

In this section, the performance of NSLM is compared with 1-norm ELM using several classification and regression datasets. All codes are implemented in MATLAB environment.

To verify the effectiveness of the proposed NSLM method, numerical comparisons are carried out on six binary and three multiclass datasets for classification and eight datasets for regression. The characteristics of datasets are shown in Tables 2 and 3. All datasets are taken from UCI machine learning repository [20].

In all the experiments, the original data are normalized by the following formulation:

$$x_{ij} = \frac{x_{ij} - x_j^{min}}{x_j^{max} - x_j^{min}} \quad \forall (i, j)$$

Table 2 Classification datasets

Data	Samples	Attributes	Classes
Australian Credit	690	14	2
Breast Cancer	699	9	2
Diabetes	768	9	2
Glass	214	9	6
Iris	150	4	3
Liver	345	6	2
WDBC	569	30	2
Wine	178	13	3
Cleveland	297	13	2

Table 3 Regression datasets

Data	Samples	Attributes
Autoprice	159	15
Clouds	108	7
Housing	506	13
Quakes	2178	3
Strikes	635	6
Baskballs	96	4
Bodyfat	252	13
Pyrim	74	27

where $x_j^{min} = \min_{i \in \{1,\ldots,N\}}\{x_{ij}\}$ and $x_j^{max} = \max_{i \in \{1,\ldots,N\}}\{x_{ij}\}$ for the jth attribute, $j \in \{1, \ldots, d\}$.

Additive nodes are used for the hidden layer of both networks with hyperbolic tangent as the activation function. The input weights and biases of the hidden nodes are selected randomly with uniform distribution from the interval $[-1, 1]$ at the beginning and are used as the initial input for both algorithms. The penalty parameter is set to $\theta = 10^{-4}$. The optimal values of parameters L, C_{in} and C_{out} are determined through comparison of mean accuracies by varying values of the initial number of hidden nodes L over the set $\{10, 100, 200, 400, 800, 1000\}$, C_{in} over $\{0.01, 0.1, 1, 10, 100, 1000\}$ and C_{out} over $\{32, 64, 128, 256, 512, 1024, 2048\}$ and then choosing the triple (L, C_{in}, C_{out}) corresponding to the highest mean accuracy for each method.

The mean accuracy corresponding to each triple (L, C_{in}, C_{out}) is obtained by performing 10 independent trials. For each trial, the total samples are randomly permuted and then evenly divided for training and testing. The mean value of 10 test accuracies is returned as the mean accuracy of the corresponding triple. For classification datasets, the highest mean accuracy among all triples (L, C_{in}, C_{out}) for both algorithms is shown under "MaxAcc" in Table 4 and the corresponding triple is also illustrated in Table 5. Similarly for regression datasets, the lowest Root Mean Square Error (RMSE) among all triples is shown under "MinRMSE" in Table 6 and the corresponding optimal triple is illustrated in Table 7. The value "STD" shows the standard deviation of 10 test accuracies corresponding to MaxAcc and MinRMSE, and is obtained from the following equation:

Table 4 Classification results

Dataset	Method	MaxAcc (%)	STD (%)	#Nodes
Australian Credit	NSLM	**86.46**	**1.47**	**9**
	1-norm	84.29	1.82	73
Breast Cancer	NSLM	95.00	1.21	**5.60**
	1-norm	**95.27**	**1.16**	47.70
Diabetes	NSLM	**77.68**	**1.14**	**6.10**
	1-norm	68.52	4.03	74.50
Glass	NSLM	**46.73**	**3.63**	**5.20**
	1-norm	34.11	17.43	27.10
Iris	NSLM	**86**	**30.37**	**2.50**
	1-norm	54.67	45.03	3.30
Liver	NSLM	**66.13**	**3.85**	**5.10**
	1-norm	52.03	27.63	6
WDBC	NSLM	93.93	1.10	**18**
	1-norm	**94.86**	**1.03**	52.80
Wine	NSLM	90.56	**2.81**	**10.10**
	1-norm	**92.70**	3.36	31.90
Cleveland	NSLM	**65.37**	**4.20**	**8.70**
	1-norm	48.99	7.25	8.90

Table 5 Optimal parameters for classification datasets

Dataset	Method	L	C_{out}	C_{in}
Australian Credit	NSLM	800	512	100
	1-norm	100	256	
Breast Cancer	NSLM	800	1024	0.10
	1-norm	100	64	
Diabetes	NSLM	1000	256	1000
	1-norm	100	32	
Glass	NSLM	10	64	10
	1-norm	100	32	
Iris	NSLM	200	256	1
	1-norm	10	1024	
Liver	NSLM	800	32	1
	1-norm	10	128	
WDBC	NSLM	200	1024	10
	1-norm	100	256	
Wine	NSLM	1000	512	1
	1-norm	100	512	
Cleveland	NSLM	100	32	100
	1-norm	10	256	

Table 6 Regression results

Dataset	Method	MinRMSE (%)	STD (%)	#Nodes
Autoprice	NSLM	**1.04**	**0.25**	**2.50**
	1-norm	1.57	0.35	2.60
Clouds	NSLM	**0.38**	**0.11**	**2.40**
	1-norm	0.78	0.29	22.60
Housing	NSLM	0.12	**0.01**	**6.10**
	1-norm	**0.10**	0.02	36.10
Quakes	NSLM	**0.18**	**0.01**	**2.90**
	1-norm	0.24	0.06	3.70
Strikes	NSLM	**0.30**	**0.00**	**3.80**
	1-norm	0.42	0.12	43.30
Baskballs	NSLM	**0.25**	**0.10**	2
	1-norm	0.47	0.39	**1.90**
Bodyfat	NSLM	0.14	0.09	**2.30**
	1-norm	**0.13**	**0.03**	19.80
Pyrim	NSLM	0.14	0.07	**4**
	1-norm	**0.12**	**0.04**	19.10

Table 7 Optimal parameters for regression datasets

Dataset	Method	L	C$_{out}$	C$_{in}$
Autoprice	NSLM	200	128	0.01
	1-norm	10	32	
Clouds	NSLM	10	32	10
	1-norm	800	32	
Housing	NSLM	100	32	10000
	1-norm	100	256	
Quakes	NSLM	200	1024	0.01
	1-norm	10	512	
Strikes	NSLM	10	512	1
	1-norm	400	128	
Baskballs	NSLM	100	1024	100
	1-norm	10	512	
Bodyfat	NSLM	1000	32	1
	1-norm	800	512	
Pyrim	NSLM	800	64	10
	1-norm	800	64	

$$std_{(C_{in},C_{out})} = \frac{1}{10}\left(\sum_{i=1}^{10}(acc(i) - a\bar{c}c)^2\right)^{\frac{1}{2}}, \quad \forall(C_{in}, C_{out})$$

where $acc(i)$ is the test accuracy of ith test set and $a\bar{c}c$ is the mean value of 10 test accuracies. Non-zero components of the optimal weight vector are enumerated after each training phase

Table 8 Total execution time for a random trial on classification datasets

Dataset	Method	Acc(%)	#Nodes	Time(s)
Australian Credit	NSLM	**86.09**	**9**	4.85
	1-norm	84.06	79	**1.06**
Breast Cancer	NSLM	94.29	**3**	2.40
	1-norm	**94.84**	41	**1.18**
Diabetes	NSLM	**75.26**	**6**	4.58
	1-norm	46.09	62	**1.11**
Glass	NSLM	**52.34**	**8**	**0.06**
	1-norm	42.99	28	0.16
Iris	NSLM	**93.33**	**4**	0.20
	1-norm	80	**4**	**0.04**
Liver	NSLM	**64.16**	**6**	1.87
	1-norm	63.95	7	**0.06**
WDBC	NSLM	**94.03**	**18**	0.91
	1-norm	91.90	65	**0.87**
Wine	NSLM	88.76	**10**	5.21
	1-norm	**95.51**	29	**0.10**
Cleveland	NSLM	**67.11**	**8**	0.14
	1-norm	42.57	9	**0.05**

and 10 values corresponding to MaxAcc and MinRMSE are averaged as the average number of hidden nodes in the optimal network which is shown under "#Nodes". Similar to [15], we solve the problem (15) using Newton's method without Armijo step size. In Tables 4, 6, 8 and 9, the best results are shown in boldface for each dataset, with respect to higher accuracy, lower deviation, smaller network size and less execution time.

As can be seen from Table 4 for classification and Table 6 for regression problem, the average number of hidden nodes in NSLM has dramatically decreased in most of the experiments, as was expected. The increase in sparsity level is up to 12 times (on Diabetes dataset) for classification and up to 11 times (on Strikes dataset) for regression in comparison to 1-norm ELM. Furthermore, NSLM achieves better accuracies on more than two third of the experiments. This accuracy increase is significant in some classification instances. In the remaining instances, there is a slight difference between the accuracies obtained for the two methods, meaning that NSLM could maintain or improve the accuracies with a much more compact architecture.

In order to evaluate the practicality of our proposed method, we also measured the total execution time of a random trial in seconds on each dataset for the two algorithms, using the optimal parameters obtained in the previous experiments. The results are reported under "Time" in Tables 8 and 9 along with the accuracies obtained for classification under "Acc" and for regression under "RMSE". Also, the number of hidden nodes obtained for this random trial is shown under the "#Nodes" column. Since the structure of NSLM is unchanged in comparison to 1-norm network, the testing time is the same for both methods. According to Table 8, NSLM needs more time for training than 1-norm. However, this is not always the case in regression instances, as can be seen in Table 9. The time increase is due to the orthogonalization process, particularly when the number of attributes is large. However, we are less concerned with the time of learning, which is spent in the laboratory, than making an accurate and efficient predictor to perform well in practice.

Table 9 Total execution time for a random trial on regression datasets

Dataset	Method	RMSE(%)	#Nodes	Time(s)
Autoprice	NSLM	**0.82**	**2**	0.19
	1-norm	1.36	9	**0.10**
Clouds	NSLM	**0.64**	**2**	**0.07**
	1-norm	2.91	20	0.88
Housing	NSLM	**0.11**	**11**	**0.32**
	1-norm	0.38	39	0.47
Quakes	NSLM	**0.21**	**3**	**1.20**
	1-norm	2.33	5	11.10
Strikes	NSLM	**0.30**	**3**	**0.11**
	1-norm	1.06	43	2.84
Baskballs	NSLM	**0.35**	3	0.12
	1-norm	0.55	**1**	**0.08**
Bodyfat	NSLM	0.43	**1**	2.54
	1-norm	**0.11**	11	**0.96**
Pyrim	NSLM	**0.05**	**4**	**1.60**
	1-norm	1.14	18	2.21

5 Conclusion

In this research a novel learning algorithm called New Sparse Learning Machine (NSLM) is proposed, for regression and multiclass classification.

In practice, when using randomized feedforward networks, a large number of hidden nodes are generated, which may be highly correlated, that leads to inefficiency in the size of the network. It has been proven that, the mean square error of ensemble classifiers depends on the amount of error correlation between individual units. The main idea of our proposed approach is to achieve any potential sparsity in the network by reducing the correlation between classifiers outputs to improve its performance on test data. NSLM transforms the random output vectors of hidden nodes into a diverse set by orthogonalizing the columns of the output matrix of the hidden layer using Gram–Schmidt algorithm. The orthogonalizing procedure leads to an ideal decrease in correlation between output vectors of hidden nodes. Computational experiments on various regression and classification datasets confirm that, the proposed approach eventually results in better generalization performance and significantly smaller network size.

According to [15], L_1-norm minimization problem tends to make components of the optimal solution vector become zero. One case of potential interest is using a similar procedure to obtain more sparsity in the number of input weights as well. It seems to be particularly beneficial for high dimensional datasets, since it can do the learning and the feature selection duties simultaneously. Furthermore, applying the orthogonalization procedure introduced for NSLM in other neural network based learning algorithms to improve generalization, also opens new areas for research.

Acknowledgements The authors would like to thank the anonymous referee for the constructive comments and suggestions that help improve the quality of this manuscript.

References

1. Hornik K (1991) Approximation capabilities of multilayer feedforward networks. Neural Netw 4:251–257. doi:10.1016/0893-6080(91)90009-T
2. Leshno M, Lin VY, Pinkus A, Schocken S (1993) Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Netw 6:861–867. doi:10.1016/S0893-6080(05)80131-5
3. Igelnik B, Pao YH (1995) Stochastic choice of basis functions in adaptive function approximation and the functional-link net. IEEE Trans Neural Netw 6:1320–1329. doi:10.1109/72.471375
4. Park J, Sandberg IW (1991) Universal approximation using radial-basis-function networks. Neural Comput 3:246–257. doi:10.1162/neco.1991.3.2.246
5. Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17:879–892. doi:10.1109/TNN.2006.875977
6. Huang GB, Babri H (1998) Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. IEEE Trans Neural Netw 9:224–229. doi:10.1109/72.655045
7. Anders U, Korn O (1999) Model selection in neural networks. Neural Netw 12:309–323. doi:10.1016/S0893-6080(98)00117-8
8. Sietsma J, Dow RJ (1988) Neural net pruning-why and how. In: IEEE international conference on neural networks, pp 325–333. doi:10.1109/ICNN.1988.23864
9. Lazarevic A, Obradovic Z (2001) Effective pruning of neural network classifier ensembles. In: Proceedings of IJCNN'01 IEEE international joint conference on neural networks, vol 2, pp 796–801. doi:10.1109/IJCNN.2001.939461
10. Setiono R (1997) A penalty-function approach for pruning feedforward neural networks. Neural Comput 9:185–204. doi:10.1162/neco.1997.9.1.185
11. Setiono R (1996) Extracting rules from pruned neural networks for breast cancer diagnosis. Artif Intell Med 8:37–51. doi:10.1016/0933-3657(95)00019-4
12. Huang GB, Saratchandran P, Sundararajan N (2005) A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. IEEE Trans Neural Netw 16:57–67. doi:10.1109/TNN.2004.836241
13. Rong HJ, Ong YS, Tan AH, Zhu Z (2008) A fast pruned-extreme learning machine for classification problem. Neurocomputing 72:359–366. doi:10.1016/j.neucom.2008.01.005
14. Alcin OF, Sengur A, Ghofrani S, Ince MC (2014) GA-SELM Greedy algorithms for sparse extreme learning machine. Measurement 55:126–132. doi:10.1016/j.measurement.2014.04.012
15. Balasundaram S, Gupta D (2014) 1-Norm extreme learning machine for regression and multiclass classification using Newton method. Neurocomputing 128:4–14. doi:10.1016/j.neucom.2013.03.051
16. Sakar A, Mammone RJ (1993) Growing and pruning neural tree networks. IEEE Trans Comput 42:291–299. doi:10.1109/12.210172
17. Brown G, Wyatt J, Harris R, Yao X (2005) Diversity creation methods: a survey and categorisation. Inf Fusion 6:5–20. doi:10.1016/j.inffus.2004.04.004
18. Hsu KW, Srivastava J (2010) Relationship between diversity and correlation in multi-classifier systems. In: Zaki MJ, Yu JX, Ravindran B, Pudi V (eds) Advances in knowledge discovery and data mining. Springer, Berlin, pp 500–506. doi:10.1007/978-3-642-13672-647
19. Tumer K, Ghosh J (1996) Error correlation and error reduction in ensemble classifiers. Connect Sci 8:385–404. doi:10.1080/095400996116839
20. Lichman M (2013) UCI Machine learning repository. University of California Irvine, CA. http://archive.ics.uci.edu/ml. Accessed 22 March 2016
21. Bertsekas DP (1999) Nonlinear programming, 2nd edn. Athena Scientific, Cambridge. ISBN: 1-886529-00-0
22. Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, New York
23. Cohen H (1993) A course in computational algebraic number theory. Springer-Verlag New York, Inc., New York
24. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501. doi:10.1016/j.neucom.2005.12.126