A Backward Learning Algorithm in Polynomial Echo State Networks

Cuili Yang(✉), Xinxin Zhu, and Junfei Qiao

Faculty of Information Technology,
Beijing University of Technology Beijing Key Laboratory
of Computational Intelligence and Intelligence System,
Beijing 100124, People's Republic of China
{clyang5,junfeiq}@bjut.edu.cn,
804340106@qq.com, 1205580412@qq.com

Abstract. Recently, the polynomial echo state network (PESN) has been proposed to incorporate the high order information of input features. However, there are some redundant inputs in PESN, which results in high computational cost. To solve this problem, a backward learning algorithm is designed for PESN, which is denoted as BL-PESN for short. The criterion for input features removing is designed to prune the insignificant input features one by one. The simulation results illustrate that the proposed approach has better prediction accuracy and less testing time than other ESNs.

Keywords: Polynomial echo state network · Subset selection · Backward learning algorithm

1 Introduction

Echo state network (ESN) has drawn great interest in the research field [1]. Compared with the traditional gradient based recurrent neural networks (RNNs), the ESN has faster convergence speed with better performance. The core structure of an ESN is the reservoir, which is consisted of a large number of recurrent connected neurons. In an ESN, the internal weights and input weights are randomly given, while the weights connecting the reservoir and the readouts are trained by the standard linear regression routines. Recently, the ESN has been adopted in different fields, such as time series prediction [2], the pattern extraction [3], speech recognition [4], and adaptive control [5].

Recently, many various ESN schemes have been explored, such as augmented complex ESNs [6], the minimum complexity ESN [7], the robust ESN [8]. However, in the above methods, the high order information of input features are not mentioned. To solve this problem, the polynomial ESN (PESN) was investigated [9] by employing the polynomial functions of complete input features into output weights. The experimental results showed that PESN performed better than traditional ESNs in terms of accuracy and testing speed.

X. B. Zhai et al. (Eds.): MLICOM 2019, LNICST 294, pp. 501–509, 2019.
https://doi.org/10.1007/978-3-030-32388-2_43

However, some redundant input features of PESN may be incorporated to construct the polynomial output weights, which not only increase the computational cost but also worse the testing accuracy. Thus, it is essential to prune the redundant features. To solve this problem, the forward and backward learning algorithms provide an effective way. The forward learning algorithms start with an empty model, then gradually add the term with the largest decrease in the cost function [10]. While the backward learning algorithms involve an over-sized network and delete the unnecessary term one by one [11]. Compared with backward learning algorithms, the forward learning algorithms are always sensitive to the initial conditions. Thus, the backward learning algorithm is focused in this paper. Recently, several backward learning algorithms have been developed for ESNs. For example, a pruning and regularization algorithm was proposed in [12] to prune the insignificant connection of the reservoir. Then, a sensitive iterative pruning algorithm was designed to remove the the least sensitive reservoir neurons.

In this paper, a backward learning algorithm is designed for polynomial ESN (BL-PESN) to remove the redundant input features. The BL-PESN is started with a p-order PESN in which the complete input features are chosen as output weights. Then, the redundant or insignificant features are pruned one by one until the required criterion is satisfied. Finally, some experiments are carried out to illustrate the effectiveness of the proposed method.

The rest of the paper is organized as follows. Section 2 describes the ESN and the PESN briefly. The criterion of pruning feature for BL-PESN is given in Sect. 3. In Sect. 4, three experiments are done. Section 5 concludes the paper.

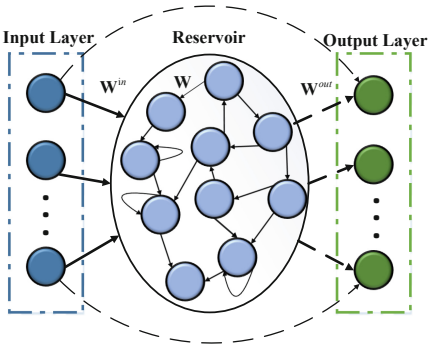


Fig. 1. The structure of the ESN without output feedback.

2 Preliminaries

In this section, the original ESN (OESN) and the PESN are briefly described, respectively.

2.1 OESN

The general structure of an ESN without output feedback is shown in Fig. 1. It is composed of n input units, m readouts and N recurrent connected neurons called the reservoir. $\mathbf{W}^{in}$ and $\mathbf{W}$ are the input weight matrix and reservoir weight matrix. The output weights $\mathbf{W}^{out}$ is trained by linear regression method. At the time step k, the vectors of input, output and internal state are denoted by $\mathbf{u}(k) = [u_1(k), u_2(k), ..., u_n(k)]^T$, $\mathbf{y}(k) = [y_1(k), y_2(k), ..., y_m(k)]^T$, and $\mathbf{x}(k) = [x_1(k), x_2(k), ..., x_N(k)]^T$, respectively. The internal state dynamic and the output equations of the ESN can be described as below

$$\mathbf{s}(k) = \mathbf{g}(\mathbf{W}\mathbf{s}(k-1) + \mathbf{W}^{in}\mathbf{u}(k)) \tag{1}$$

$$\mathbf{y}(k) = \mathbf{W}^{out}[\mathbf{s}(k)^T, \mathbf{u}(k)^T]^T = \mathbf{W}^{out}\mathbf{x}(k) \tag{2}$$

where $\mathbf{g}(\cdot)$ is the activation function and $\mathbf{x}(k) = [\mathbf{s}(k)^T, \mathbf{u}(k)^T]^T$ is the concatenation of the internal state $\mathbf{s}(k)$ and input vector $\mathbf{u}(k)$.

Denote $\mathbf{X} = [\mathbf{x}(1), \mathbf{x}(2), ..., \mathbf{x}(L)]$ as the internal state matrix, and the corresponding matrix of targets as $\mathbf{T} = [\mathbf{t}(1), \mathbf{t}(2), ..., \mathbf{t}(L)]$, with $\mathbf{t}(k) = [t_1(k), t_2(k), ..., t_m(k)]^T$. The least-squares estimation of $\mathbf{W}^{out}$ is calculated as

$$\mathbf{W}^{out} = \mathbf{T}\mathbf{X}^{\dagger} = \mathbf{T}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1} \tag{3}$$

where $\mathbf{X}^{\dagger}$ is the Moore-Penrose generalized inverse of $\mathbf{X}$.

2.2 PESN

In OESN, the high order information of the inputs is not considered. To solve this problems, the polynomial echo state network (PESN) is presented. In PESN, the p-order polynomial function of full input features is constructed as the output weights, which is expressed as below

$$\mathbf{w}_i(\mathbf{u}(k)) = \mathbf{w}_{i00} + \sum_{q=1}^{p}\sum_{j=1}^{n}\mathbf{w}_{ijq}^T u_j^q(k) = \mathbf{B}_i\mathbf{z}(k) \tag{4}$$

where $i = 1, ..., N+n$, p is the polynomial order. Define $M = np+1$, $\mathbf{z}(k) \in \mathbb{R}^M$ and $\mathbf{B}_i \in \mathbb{R}^{m\times M}$ are given by

$$\mathbf{z}(k) = [1, u_1(k), ..., u_n(k), u_1^p(k), ..., u_n^p(k)]^T \tag{5}$$

$$\mathbf{B}_i = [\mathbf{w}_{i00}, \mathbf{w}_{i11}, ..., \mathbf{w}_{in1}, ..., \mathbf{w}_{i1p}, ..., \mathbf{w}_{inp}] \tag{6}$$

Similar to Eq. (2), the output of the PESN at the time step k is described as

$$\mathbf{y}(k) = \mathbf{W}^{out}(\mathbf{u}(k))\mathbf{x}(k) \tag{7}$$

where $\mathbf{W}^{out}(\mathbf{u}(k))$ is generated as below

$$\begin{aligned}\mathbf{W}^{out}(\mathbf{u}(k)) &= [\mathbf{w}_1(\mathbf{u}(k)), \mathbf{w}_2(\mathbf{u}(k)), ..., \mathbf{w}_{N+n}(\mathbf{u}(k))] \\ &= [\mathbf{B}_1\mathbf{z}(k), \mathbf{B}_2\mathbf{z}(k), .., \mathbf{B}_{N+n}\mathbf{z}(k)] \\ &= \mathbf{B}(\mathbf{I}_{N+n} \otimes \mathbf{z}(k))\end{aligned} \tag{8}$$

where $\otimes$ is the Kronecker product, $\mathbf{I}_{N+n} \in \mathbb{R}^{(N+n)\times(N+n)}$ is an unit matrix, and the unknown parameter matrix $\mathbf{B} \in \mathbb{R}^{m\times M(N+n)}$ is shown as below

$$\mathbf{B} = [\mathbf{B}_1, ..., \mathbf{B}_{N+n}] \tag{9}$$

Substituting the Eq. (8) into Eq. (7), one obtains

$$\mathbf{y}(k) = \mathbf{B}\left(\mathbf{x}(k) \otimes \mathbf{z}(k)\right) \tag{10}$$

In this context, calculating the solution of $\mathbf{B}$ is equivalent to finding the least square solution of the following linear formula

$$J = \arg\min_{\mathbf{B}} \left\{ C \left\|\mathbf{B}\right\|_F^2 + \left\|\mathbf{B}\bar{\mathbf{X}} - \mathbf{T}\right\|_F^2 \right\} \tag{11}$$

where $\|\cdot\|_F$ denotes the Frobenius-norm, $C > 0$ is the ridge parameter, $\bar{\mathbf{X}} \in \mathbb{R}^{M(N+n)\times L}$, is given as

$$\bar{\mathbf{X}} = [\mathbf{x}(1) \otimes \mathbf{z}(1), \mathbf{x}(2) \otimes \mathbf{z}(2), ..., \mathbf{x}(L) \otimes \mathbf{z}(L)] \tag{12}$$

By setting $\frac{dJ}{d\mathbf{B}} = 0$, one gets

$$\hat{\mathbf{B}} = \mathbf{T}\bar{\mathbf{X}}^T\left(\bar{\mathbf{X}}\bar{\mathbf{X}}^T + C\mathbf{I}\right)^{-1} \tag{13}$$

3 The Proposed BS-PESN

In this paper, BL-PESN is proposed to prune the insignificant input features. In the following, the details of the proposed BL-PESN are presented.

Firstly, Eq. (5) can be reorganized as

$$\mathbf{z}(k) = [1, u_1(k), ..., u_1^p(k), ..., u_n(k), ..., u_n^p(k)]^T = [1, \mathbf{z}_i(k), ..., \mathbf{z}_n(k)]^T \tag{14}$$

where $\mathbf{z}_i(k) = [u_i(k), ..., u_i^p(k)]$, $i = 1, ..., n$. Following Eq. (12), $\bar{\mathbf{X}}$ is given as

$$\begin{aligned}\bar{\mathbf{X}} &= [\mathbf{x}(1) \otimes \mathbf{z}(1), \mathbf{x}(2) \otimes \mathbf{z}(2), ..., \mathbf{x}(L) \otimes \mathbf{z}(L)] \\ &= \begin{bmatrix} \mathbf{x}(1) & \mathbf{x}(2) & \cdots & \mathbf{x}(L) \\ \mathbf{x}(1) \otimes \mathbf{z}_1^T(1) & \mathbf{x}(2) \otimes \mathbf{z}_1^T(2) & \cdots & \mathbf{x}(L) \otimes \mathbf{z}_1^T(L) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}(1) \otimes \mathbf{z}_n^T(1) & \mathbf{x}(2) \otimes \mathbf{z}_n^T(2) & \cdots & \mathbf{x}(L) \otimes \mathbf{z}_n^T(L) \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{X}}_0 \\ \bar{\mathbf{X}}_1 \\ \vdots \\ \bar{\mathbf{X}}_n \end{bmatrix}\end{aligned} \tag{15}$$

where $\bar{\mathbf{X}}_0 = \mathbf{X}$, $\bar{\mathbf{X}}_i = \left[\mathbf{x}(1) \otimes \mathbf{z}_i^T(1), \mathbf{x}(2) \otimes \mathbf{z}_i^T(2), ..., \mathbf{x}(L) \otimes \mathbf{z}_i^T(L)\right]$. Correspondingly, the matrix $\mathbf{B}$ in Eq. (9) is rewritten as

$$\mathbf{B} = [\mathbf{B}_0, \mathbf{B}_1, ..., \mathbf{B}_n] \tag{16}$$

It can be determined that pruning the insignificant input features from PESN is equivalent to deleting $\bar{\mathbf{X}}_i$ from $\bar{\mathbf{X}}$. It has been proved that the cost function is gradually increased with an decreasing number of features. The smaller the

increase value on the cost function incurred by pruning one feature is, the less important this feature is viewed to be. Hence, it can be justified which feature is insignificant from the increase value. Define a full index set $P = \{1, ..., n\}$, BL-PESN is initialized with $\bar{\mathbf{X}}_P = \bar{\mathbf{X}}$ in Eq. (15), and $l = 0$.

Assuming that BL-PESN is at the lth iteration. The cost function is described as below

$$\hat{J}^{(l)} = \min_{\mathbf{B}_P} \left\{ J^{(l)} = C \left\| \mathbf{B}_P \right\|_F^2 + \left\| \mathbf{B}_P \bar{\mathbf{X}}_P - \mathbf{T} \right\|_F^2 \right\} \tag{17}$$

If at the $(l+1)$th iteration, the ith feature is pruned according to a certain criterion. Equation (17) becomes

$$\hat{J}_{-i}^{(l+1)} = \min_{\mathbf{B}_{P\setminus\{i\}}} \left\{ J_{-i}^{(l+1)} = C \left\| \mathbf{B}_{P\setminus\{i\}} \right\|_F^2 + \left\| \mathbf{B}_{P\setminus\{i\}} \bar{\mathbf{X}}_{P\setminus\{i\}} - \mathbf{T} \right\|_F^2 \right\} \tag{18}$$

Here, $\bar{\mathbf{X}}_P = \begin{bmatrix} \bar{\mathbf{X}}_{P\setminus\{i\}} \\ \bar{\mathbf{X}}_i \end{bmatrix}$, $\mathbf{B}_P = \left[\mathbf{B}_{P\setminus\{i\}}, \mathbf{B}_i \right]$. Equation (17) can be rewritten as

$$\begin{aligned} \hat{J}^{(q)} &= \arg\min_{\mathbf{B}_p} \left\{ J^{(q)} = C \left\| \left[\mathbf{B}_{P\setminus\{i\}}, \mathbf{B}_i \right] \right\|_F^2 + \left\| \left[\mathbf{B}_{P\setminus\{i\}}, \mathbf{B}_i \right] \begin{bmatrix} \bar{\mathbf{X}}_{P\setminus\{i\}} \\ \bar{\mathbf{X}}_i \end{bmatrix} - \mathbf{T} \right\|_F^2 \right\} \\ &= \arg\min_{\mathbf{B}_i} \left\{ \hat{J}_{-i}^{(q+1)} + \left\| \mathbf{B}_i \bar{\mathbf{X}}_i \right\|_F^2 + C \left\| \mathbf{B}_i \right\|_F^2 - 2tr \left(\left(\hat{\mathbf{B}}_{P\setminus\{i\}} \bar{\mathbf{X}}_{P\setminus\{i\}} - \mathbf{T} \right)^T \mathbf{B}_i \bar{\mathbf{X}}_i \right) \right\} \end{aligned} \tag{19}$$

where $tr(\cdot)$ represents the trace of square matrix.

By setting $\frac{d\hat{J}^{(q)}}{d\mathbf{B}_i} = 0$, one gets

$$\hat{\mathbf{B}}_i = \left(\hat{\mathbf{B}}_{P\setminus\{i\}} \bar{\mathbf{X}}_{P\setminus\{i\}} - \mathbf{T} \right) \bar{\mathbf{X}}_i^T \left(\bar{\mathbf{X}}_i \bar{\mathbf{X}}_i^T + C\mathbf{I} \right)^{-1} \tag{20}$$

Substituting (20) into (19) obtains

$$\hat{J}^{(q)} = \hat{J}_{-i}^{(q+1)} - tr \left(\left(\hat{\mathbf{B}}_{P\setminus\{i\}} \bar{\mathbf{X}}_{P\setminus\{i\}} - \mathbf{T} \right)^T \hat{\mathbf{B}}_i \bar{\mathbf{X}}_i \right) \tag{21}$$

Hence, when removing ith input feature at the $(q+1)$th iteration, the error value can be expressed as

$$\Delta_{-i} = \hat{J}_{-i}^{(q+1)} - \hat{J}^{(q)} = tr \left(\left(\hat{\mathbf{B}}_{P\setminus\{i\}} \bar{\mathbf{X}}_{P\setminus\{i\}} - \mathbf{T} \right)^T \hat{\mathbf{B}}_i \bar{\mathbf{X}}_i \right) \tag{22}$$

Define the criterion of pruning feature as

$$s = \arg\min_{i \in P} \left\{ \Delta_{-i} = tr \left(\left(\hat{\mathbf{B}}_{P\setminus\{i\}} \bar{\mathbf{X}}_{P\setminus\{i\}} - \mathbf{T} \right)^T \hat{\mathbf{B}}_i \bar{\mathbf{X}}_i \right) \right\} \tag{23}$$

According to Sherman-Morrison formula [14], we can calculate $\hat{\mathbf{B}}_{P\setminus\{i\}}$ in Eq. (23) with a fast speed.

Firstly, denote $(\bar{\mathbf{X}}_P\bar{\mathbf{X}}_P^T + C\mathbf{I})^{-1}$ by $\mathbf{D}^{(l)}$, $\mathbf{D}^{(l)}$ is decomposed in the block form as

$$\mathbf{D}^{(l)} = \begin{bmatrix} \mathbf{D}_{11} & \mathbf{D}_{12} & \mathbf{D}_{13} \\ \mathbf{D}_{12}^T & \mathbf{D}_{22} & \mathbf{D}_{32}^T \\ \mathbf{D}_{13}^T & \mathbf{D}_{32} & \mathbf{D}_{33} \end{bmatrix} \tag{24}$$

where $\begin{bmatrix} \mathbf{D}_{12}^T & \mathbf{D}_{22} & \mathbf{D}_{32}^T \end{bmatrix}$ is the corresponding row to the ith feature. Hence, when the ith feature is pruned at the $(l+1)$th iteration,

$$\mathbf{D}_{-i}^{(l+1)} = \begin{bmatrix} \mathbf{D}_{11} & \mathbf{D}_{13} \\ \mathbf{D}_{13}^T & \mathbf{D}_{33} \end{bmatrix} - \begin{bmatrix} \mathbf{D}_{12} \\ \mathbf{D}_{32} \end{bmatrix} \mathbf{D}_{22}^{-1} \begin{bmatrix} \mathbf{D}_{12}^T & \mathbf{D}_{32}^T \end{bmatrix} \tag{25}$$

Correspondingly, the output weights can be updated as below

$$\hat{\mathbf{B}}_{P\setminus\{i\}} = \mathbf{T}\mathbf{X}_{P\setminus\{i\}}^T\mathbf{D}_{-i}^{(l+1)} \tag{26}$$

3.1 The Flowchart of BL-PESN Algorithm

The complete process of BL-PESN is given as below

- Step 1. Input the training samples $\{(\mathbf{u}(k), \mathbf{t}(k))_{k=1}^L | \mathbf{u}(k) \in \mathbb{R}^n, \mathbf{t}(k) \in \mathbb{R}^m\}$, generate the matrices $\mathbf{W}^{in}$ and $\mathbf{W}$ randomly, define the reservoir state matrix $\mathbf{X} = [\mathbf{x}(1), \mathbf{x}(2), ..., \mathbf{x}(L)]$ and the target output matrix $\mathbf{T} = [\mathbf{t}(1), \mathbf{t}(2), ..., \mathbf{t}(L)]$.
- Step 2. Determine the matrices $\mathbf{z}^T(k)$ in Eq. (14) and $\bar{\mathbf{X}}$ in Eq. (15).
- Step 3. Choose the ridge parameter C, and decide the maximum algorithm iteration $n_{\max}(0 \leq n_{\max} \leq n)$. In the initialization, let $P = \{1, ..., n\}$, and $l = 0$. Set $\bar{\mathbf{X}}_P = \bar{\mathbf{X}}$, $\mathbf{D}^{(0)} = (\bar{\mathbf{X}}_P^T\bar{\mathbf{X}}_P + C\mathbf{I})^{-1}$, and calculate $\hat{\mathbf{B}}_P = \mathbf{T}\bar{\mathbf{X}}_P^T\mathbf{D}^{(0)}$.
- Step 4. Determine the index s in Eq. (23) and find $\bar{\mathbf{X}}_s$ from $\bar{\mathbf{X}}$.
- Step 5. Calculate $\hat{\mathbf{B}}_{P\setminus\{s\}}$ in Eq. (26). Meanwhile, let $P \leftarrow P\setminus\{s\}$, and $l \leftarrow l+1$.
- Step 6. If $l \geq n_{\max}$ is satisfied , go to Step 7; otherwise, turn to Step 4.
- Step 7. Choose the network with the most important set of input features.

4 Simulation Results and Discussion

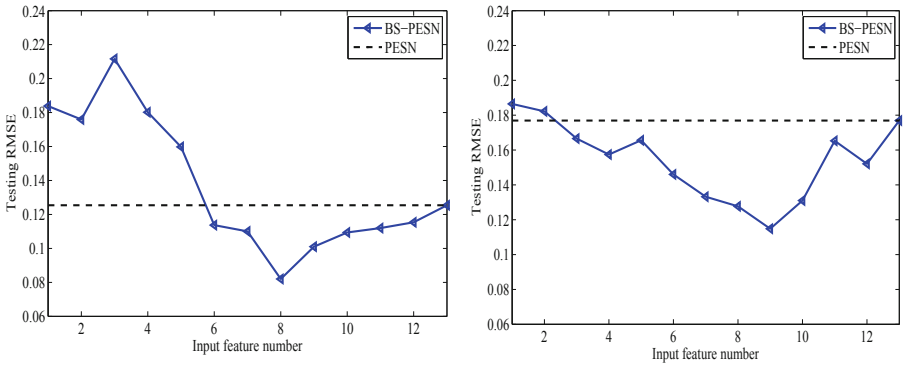
In this section, the performance of the proposed BL-PESN is compared with the PESN and the OESN. The ridge parameter C is chosen from a wide range of candidates $\{2^{-20}, 2^{-19}, ..., 2^{19}, 2^{20}\}$ for each dataset. The characteristics of regression datasets and the nearly optimal parameter C are shown in Table 1.

To facilitate comparisons among different algorithms, the performance index root mean square error (RMSE) is defined by

$$\text{RMSE} = \sqrt{\sum_{k=1}^{L} (\mathbf{t}(k) - \mathbf{y}(k))^2 / L} \tag{27}$$

Table 1. Information of the regression benchmark problems

Datasets	#Training samples	#Testing samples	#Features	#Outputs	p	$log_2 C$
Boston housing	350	156	13	1	1	-1
					2	-3
Stocks domain	450	500	9	1	1	-7
					2	-5
Wine-white	2600	2298	11	1	1	-3
					2	-3



(a) $p = 1$. (b) $p = 2$.

Fig. 2. Experimental results for Boston housing datasets. (Color figure online)

Table 2. Detailed experimented results of regression datasets.

Datasets	p	Algorithms	#nodes	#Feature	Training RMSE	Testing RMSE	Training time (s)	Testing time (s)
Boston housing	0	OESN	200	13	**0.0413**	0.1107	0.3769	0.3205
	1	PESN	72	13	0.0579	0.0751	1.2310	0.0995
		BS-PESN	72	**8**	0.0665	**0.0702**	4.8133	**0.0877**
	2	PESN	38	13	**0.0574**	0.0755	1.5312	0.0649
		BS-PESN	38	**9**	0.0582	0.0721	6.5818	**0.0641**
Stocks domain	0	OESN	250	9	0.0364	0.0828	0.2686	0.0608
	1	PESN	100	9	**0.0331**	0.0437	0.4525	0.3528
		BS-PESN	100	**5**	0.0422	0.0428	1.9434	**0.2641**
	2	PESN	53	9	0.0391	0.0411	0.4454	0.3378
		BS-PESN	53	**6**	0.0354	**0.0401**	3.2356	**0.3447**
Wine-white	0	OESN	500	11	**0.0946**	0.1513	0.3608	0.1012
	1	PESN	84	11	0.1083	0.1254	7.0332	5.6358
		BS-PESN	84	**8**	0.1169	0.1259	8.9192	**5.2360**
	2	PESN	45	11	**0.1013**	0.1545	10.7579	8.2422
		BS-PESN	45	**6**	0.1160	**0.1225**	11.3372	**7.2810**

where $\mathbf{y}(k)$ and $\mathbf{t}(k)$ are the estimated output and target output at time k. The smaller the RMSE value stands for better performance for each algorithm.

Figure 2 shows the experimental results for Boston housing. In each figure, there are two lines, in which the black dash line represents the PESN, and the blue solid line with triangle marks is represented as BL-PESN. For each blue line, there are always some marks below the dash line, which indicated that BS-PESN can obtain better performance than the PESN. Due to space limitation, the similar results of other regression experiments are omitted.

The comparisons between the proposed BL-PESN, PESN and OESN are shown in Table 2. The results are average over 30 independent runs. Obviously, the BL-PESN always needs fewer input features to construct the p-order polynomial function, thus it needs the less testing time than the PESN. The testing RMSE values of BL-PESN are always the smallest among all the compared methods, which means that the BL-PESN is able to improve the generalization by pruning some insignificant features from the PESN.

5 Conclusions

In this paper, a backward learning algorithm is designed for PESN to prune the insignificant input features. The BL-PESN begins with a complete PESN, and gradually prunes the insignificant input features one by one until the required criterion is satisfied. The simulation results show that the proposed BL-PESN has better prediction accuracy and training speed than other ESNs.

Acknowledgement. This work was supported by the National Natural Science Foundation of China under Grants 61603012, 61533002 and 61890930-5, the Beijing Municipal Education Commission Foundation under Grant KM201710005025, the Major Science and Technology Program for Water Pollution Control and Treatment of China (2018ZX07111005), the National Key Research and Development Project under Grants 2018YFC1900800-5.

References

1. Jaeger, H., Haas, H.: Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)
2. Shi, Z., Han, M.: Support vector echo-state machine for chaotic time-series prediction. IEEE Trans. Neural Netw. **18**, 359–372 (2007)
3. Levy, B., Attux, R., Zuben, F.: Self-organization and lateral interaction in echo state network reservoirs. Neurocomputing **138**, 297–309 (2014)
4. Skowronski, M., Harris, J.: Automatic speech recognition using a predictive echo state network classifier. Neural Netw. **20**, 414–423 (2007)
5. Han, S., Lee, J.: Fuzzy echo state neural networks and funnel dynamic surface control for prescribed performance of a nonlinear dynamic system. IEEE Trans. Indus. Elect. **61**(2), 1099–1112 (2014)
6. Xia, Y., Jelfs, B., Van Hulle, M., Príncipe, J., Mandic, D.: An augmented echo state network for nonlinear adaptive filtering of complex noncircular signals. IEEE Trans. Neural Netw. **22**(1), 74–83 (2011)
7. Rodan, A., Tino, P.: Minimum complexity echo state network. IEEE Trans. Neural Netw. **22**(1), 131–144 (2011)

8. Li, D., Han, M., Wang, J.: Chaotic time series prediction based on a novel ro bust echo state network. IEEE Trans. Neural Netw. Learn. Syst. **23**(5), 787–799 (2012)
9. Yang, C., Qiao, J., Han, H., Wang, L.: Design of polynomial echo state networks for time series prediction. Neurocomputing **290**, 148–160 (2018)
10. Miller, A.: Subset Selection in Regression. Chapman and Hall, London (2002)
11. Cotter, S., Kreutz-Delgado, K., Rao, B.: Backward sequential elimination for sparse vector subset selection. Sig. Process. **81**(9), 1849–1864 (2001)
12. Dutoit, X., Schrauwen, B., Campenhout, J., Stroobandt, D., Brussel, H., Nuttin, M.: Pruning and regularization in reservoir computing. Neurocomputing **72**(7–9), 1534–1546 (2009)
13. Liu, S., Trenkler, G., Hadamard, K.: Kronecker and other matrix products. Int. J. Inform. Syst. Sci. **4**(1), 160–177 (2008)
14. Zhang, X.: Matrix Analysis and Applications. Tsinghua University Press, Beijing (2004)