
Proceedings Paper:
Wei, H. orcid.org/0000-0002-4704-7346 (2019) Boosting wavelet neural networks using evolutionary algorithms for short-term wind speed time series forecasting. In: Rojas, I., Joya, G. and Catala, A., (eds.) Proceedings of the 2019 International Work-Conference on Artificial Neural Networks (Advances in Computational Intelligence). 2019 International Work-Conference on Artificial Neural Networks (Advances in Computational Intelligence), 12-14 Jun 2019, Gran Canaria, Spain. Lecture Notes in Computer Science (11506). Springer , pp. 15-26. ISBN 9783030205201

https://doi.org/10.1007/978-3-030-20521-8_2

Acepted by 2019 International Work-Conference on Artificial Neural Networks (Advances in Computational Intelligence), Gran Canaria, Spain, 12-14 June, 2019

Boosting Wavelet Neural Networks Using Evolutionary Algorithms for Short-Term Wind Speed Time Series Forecasting

H. L. Wei

Automatic Control and Systems Engineering, University of Sheffield, Sheffield, S1 3JD, UK
w.hualiang@sheffield.ac.uk

Abstract. This paper addresses nonlinear time series modelling and prediction problem using a type of wavelet neural networks. The basic building block of the neural network models is a ridge type function. The training of such a network is a nonlinear optimization problem. Evolutionary algorithms (EAs), including genetic algorithm (GA) and particle swarm optimization (PSO), together with a new gradient-free algorithm (called coordinate dictionary search optimization – CDSO), are used to train network models. An example for real speed wind data modelling and prediction is provided to show the performance of the proposed networks trained by these three optimization algorithms.

Keywords: Neural Network, Wavelet, Boosting, Optimization, Evolutionary Algorithms, Time Series, Wind Speed, Forecasting, Data-Driven Modelling.

1 Introduction

Many practical time series modelling problems can be described as follows. There is a response variable y (also known as output or dependent variable) that depends on a set of independent variables $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ (also known as input or explanatory variables). Usually, a number of observations of both the output and input variables are available, which are denoted by $\{y_k, \mathbf{x}_k\}$ (k =1, 2,…,N). The true quantitative representation of the relationship between the output y and the input **x** is in general not known. The central task of data modelling is to establish quantitative representations, e.g. mathematical models such as y = f(**x**) + e (where e is model error), to approximate the input-output relationship as close as possible.

There are a variety of methods and algorithms in the literature for dealing with different types of nonlinear data based modelling problems, such as system identification [1]-[4], data mining [5],[6], pattern recognition and classification [7], supervised statistical learning [8],[9]. Among these methods, system identification techniques provide a tool for deducing mathematical models from measured input and output data for dynamic processes. In general, the output signal y at time instant t depends on previous values of the input and output signals, such as $u_r(t-1)$, $u_r(t-2)$, …, $u_r(t- n_u)$, $y(t-1),…,y(t-n_y)$, where r is the number of exogenous input variables, n_y is the time lag

in the output, n_u is the time lag in the inputs. For a time series without any external input, y(t) only depends on the previous output values such as $y(t-1),\ldots,y(t-n_y)$.

There is a diversity of methods and approaches for building a good function to approximate the function f or F for a given problem, including machine learning and neural networks, among others. In recent years, boosted regression has attracted extensive attention due to the work of [9]-[11], which connects boosting to general regression models such as Gaussian, logistic and generalized linear models. In [12], a new form of boosted trees called aggregated boosted trees was proposed for ecological system modelling and prediction. In [13], a boosting ridge regression was proposed for solving a medical image processing problem. A boosted L1 regularized projection pursuit for additive model learning was proposed and applied to face caricature generation and gender classification in [14]. An image based regression algorithm using boosting method has been proposed for image detection and feature selection in [15]. In [16]-[18], a boosting method was integrated to projection pursuit regression [19] to construct neural networks for spatio-temporal system identification. In order to improve the accuracy of flood forecasting, boosting approaches were proposed and incorporated to kernel based modelling and forecasting systems in [20] and [21], respectively. Most recently, comparative studies have been conducted on random forest regression, gradient boosted regression and extreme gradient boosting to tackle wind energy prediction and solar radiation problem [22]. It has been shown that ensemble methods can improve the performance of support vector regression for individual wind farm energy prediction [22].

It is known that wavelet basis functions have the property of localization in both time and frequency [23]. Due to this inherent property, wavelet approximations provide the foundation for representing arbitrary functions economically using just a small number of basis functions, and this makes wavelet representations more adaptive compared with other basis functions [24]-[29]. This motivates us to develop a new family of neural networks where wavelet are used as the building blocks.

The training of such networks is a nonlinear optimization problem, which can be solved by using either a classical gradient based algorithm or a modern meta-heuristic search algorithm. In this study, two population based algorithms, namely genetic algorithm (GA) [30] and particle swarm optimization (PSOs) [31], together with a new derivative-free algorithm (called coordinate dictionary search optimization – CDSO), are applied to estimate the hyper-parameters of the wavelet network models.

2 Structure of the Wavelet Neural Network

2.1 The Framework of the Network

Following the commonly used notation, it is assumed that the system y is related to the input vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ as below:

$$y(t) = F[\mathbf{x}(t); \boldsymbol{\theta}] + e(t) \tag{1}$$

where F[•] is a function which is normally unknown or unavailable but can be approximated by a set of functions estimated through machine learning, system identification or other data modelling techniques, $\boldsymbol{\theta}$ is a model parameter vector which can be estimated from data, and e(t) is unmeasurable noise sequence.

This study considers to use a one-hidden-layer neural network to approximate the unknown function F as:

$$F[\mathbf{x}(t);\boldsymbol{\theta}] = \sum_{k=1}^{K} w_k g_k(\mathbf{x}(t);\boldsymbol{\theta}_k) + r_K(t) \tag{2}$$

where g_k (k=1,2,…, K) are basis functions whose structure and property are known, $\boldsymbol{\theta}_k$ are parameter vectors, w_k are weight coefficients (connection coefficients), K is the number of basis functions, and $r_K(t)$ is model error (residual).

2.2 Ridge Type Wavelet Basis Function

In this study, each of the functions g_k (k=1,2…, K) in (2) is chosen to be the ridge type wavelet, which is of the form:

$$h(x_1,...,x_n) = \psi(a_0 + a_1x_1 + ... + a_nx_n) = \psi(\boldsymbol{\theta}^T\mathbf{x}) \tag{3}$$

where ψ is a scalar function, a_0, a_1, .., a_n are called direction parameters, $\boldsymbol{\theta} = [a_0, a_1, ..., a_n]^T$, and $\mathbf{x} = [x_1, x_2, ..., x_n]^T$. The function ψ in (3) can be any functions with good representation properties including wavelet basis functions. Such a function is used as the elementary building block for model construction.

2.3 Training of the Network

Let $\mathbf{y}$ = [y(1), y(2),…, y(N)]^T be the observation vector of the output signal and $\mathbf{x}_k(t) = [x_k(t), x_k(t), ..., x_k(t)]^T$ be the observation vector (t =1,2, …, N) at instant t. Let $\psi(t) = \psi(\boldsymbol{\theta}^T\mathbf{x}(t))$, $\mathbf{g}(\Phi;\boldsymbol{\theta}) = [\psi(1), \psi(2), ..., \psi(N)]^T$, with $\Phi = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$.

The boosting procedure of the network is carried out in a stepwise manner; at each step a function that minimizes the projection error is determined. Starting with $\mathbf{r}_0 = \mathbf{y}$, the first step is to find an element vector $\mathbf{g}_1 = \mathbf{g}(\mathbf{X};\boldsymbol{\theta}_1)$ such that $(\boldsymbol{\theta}_1, w_1) = \arg\min_{(\boldsymbol{\theta},w)}\{\| \mathbf{r}_0 - w\mathbf{g}(\Phi;\boldsymbol{\theta}) \|^2\}$. The resulting residual vector is defined as $\mathbf{r}_1 = \mathbf{r}_0 - w_1\mathbf{g}_1$, which can be used as the "pseudo-reference" signal to find the second element vector $\mathbf{g}_2$, and so on. This procedure may repeat many times. At the kth step, we use the sum of squared errors, $\| \mathbf{r}_k \|^2$, to define a measure called the error-to-signal ratio: $ESR_k = \| \mathbf{r}_k \|^2 / \| \mathbf{r}_0 \|^2$ and use this to monitor the iterative procedure - when ESR becomes smaller than a pre-specified threshold value. This measure can be

used to define a criterion called the penalized error-to-signal ratio: $\mathrm{PESR}_k = [\mathrm{N}/(\mathrm{N}-\lambda k)]^2 \mathrm{ESR}_k$ [16]-[18], where λ is a small positive number which is normally in the range $1 \le \lambda \le 0.005\mathrm{N}$. The maximum number of basis functions included in the network can be determined as the number of iterations where PESR reaches its minimum.

The cost function in the algorithm is defined as

$$J_{k-1}(\boldsymbol{\theta}) = \| \mathbf{r}_{k-1} - w\mathbf{g}(\Phi;\boldsymbol{\theta}) \|^2 = \sum_{t=1}^{N} [r_{k-1}(k) - wg(\mathbf{x}(t);\boldsymbol{\theta})]^2 \tag{4}$$

which can be solved through a boosted regression algorithm which is briefly summarized below.

The Boosted Projection Pursuit Regression algorithm:

Initialization: $\mathbf{r}_0 = \mathbf{y}$; k=1; ESR(k) = 0; PESR(k) = 0;

while {$k \le K$}; //{K is the maximum number of iterations}//

$[\boldsymbol{\theta}_k; w_k] = \arg\min_{\boldsymbol{\theta}, w} \left\{ \| \mathbf{r}_{k-1} - w\mathbf{g}(\Phi;\boldsymbol{\theta}) \|^2 \right\}$;

$\mathbf{g}_k = \mathbf{g}(\Phi;\boldsymbol{\theta}_k)$;

$\mathbf{r}_k = \mathbf{r}_{k-1} - w_k \mathbf{g}_k$;

$\mathrm{ESR}(k) = \| \mathbf{r}_k \|^2 / \| \mathbf{r}_0 \|^2$;

$\mathrm{PESR}_k = [\mathrm{N}/(\mathrm{N}-\lambda k)]^2 \mathrm{ESR}_k$;

k = k +1;

end while

---

3 Network Training

The optimization of the parameters in the cost function (4) can be achieved by means of either a classical gradient based algorithm or a modern meta-heuristic search algorithm. Once the estimates of the required parameters are available, the The function F[•] in (2) can then be represented as a linear combination of the estimated functions g_k (k=1,2…,).

3.1 Evolutionary Algorithms

Two evolutionary algorithms, namely, genetic algorithm (GA) [30] and particle swarm optimization (PSO) [31] are considered in this study. Matlab toolbox for GA and PSO is available in Mathworks products (Matlab 2018b). A large amount of information on evolutionary algorithms is readily and easily available publically in the literature, descriptions for these algorithms are therefore omitted here to save space.

3.2 Coordinate Dictionary Search Optimization (CDSO) Algorithm

For comparison purpose, a new coordinate dictionary search optimization (CDSO) algorithm is presented in this section.

2D Case

Let $f(x_1, x_2)$ be a function defined in 2D, and the objective is to find its global minimum point, with a box constraint $a_i \le x_i \le b_i$ (i=1,2). We first define the basic unit search directions as $D_B = \{d_1, d_2, ..., d_8\}$, where the d_i's are:

$$\begin{matrix} d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 & d_8 \end{matrix}$$
$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \frac{1}{\sqrt{2}}\begin{bmatrix} -1 \\ 1 \end{bmatrix}, \frac{1}{\sqrt{2}}\begin{bmatrix} -1 \\ -1 \end{bmatrix}, \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

The eight unit directions are shown in Fig. 1, where the first four directions are in the first, third, fifth and seventh quadrant lines, while the other four are in the second, fourth, sixth and eighth quadrant lines, respectively.

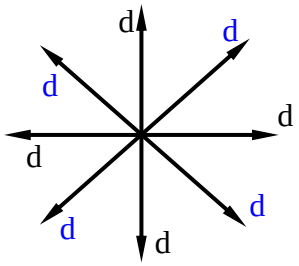

Fig. 1. An illustration of the basic unit search directions for 2D case functions.

Then we define the scaled search directions as below:

$$\begin{cases} D_{s_0} = s_0 D_B = \{s_0 d_1, s_0 d_2, ..., s_0 d_8\} \\ D_{s_1} = s_1 D_B = \{s_1 d_1, s_1 d_2, ..., s_1 d_8\} \\ ... \\ D_{s_{10}} = s_{10} D_B = \{s_{10} d_1, s_{10} d_2, ..., s_{10} d_8\} \end{cases} \tag{5}$$

where s_m (m =0,1,…,10) are scale coefficients which are defined as:

$$s_m = 10^{-m} r \tag{6}$$

The parameter r in (6) is adjustable, it determines the maximum step-size (learning rate) for network training. The dictionary used for parameter optimization is a combination of all these scaled dictionaries, that is, $D = D_{s_0} + D_{s_1} + ... + D_{s_{10}}$.

n-Dimensional Case

The 2D case can easily be extended to a general n dimensional case. Assume that a box constraint is given as $a_i \le x_i \le b_i$ (i =1,2, …, n). We define the basic coordinate search directions $D_1 = \{d_1, d_2, ..., d_{2n}\}$ as below:

$$\begin{matrix} d_1 & d_2 & & d_n & d_{n+1} & d_{n+1} & ... & d_{2n} \end{matrix}$$
$$\begin{bmatrix} 1 \\ 0 \\ ... \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ ... \\ 0 \end{bmatrix}, ..., \begin{bmatrix} 0 \\ 0 \\ ... \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ ... \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ ... \\ 0 \end{bmatrix}, ..., \begin{bmatrix} 0 \\ 0 \\ ... \\ -1 \end{bmatrix}$$

We then use the 2n unit vectors to generate new unit vectors as:

$$D_2 : \begin{cases} \{d_1, d_2\} \to \frac{1}{\sqrt{2}}[1,1,0,...,0,0]^T \\ \{d_1, d_n\} \to \frac{1}{\sqrt{2}}[1,0,0,...,0,1]^T \\ \quad ... \\ \{d_1, d_{n+2}\} \to \frac{1}{\sqrt{2}}[1,-1,0,...,0,1]^T \\ \{d_1, d_{2n}\} \to \frac{1}{\sqrt{2}}[1,0,0,...,0,-1]^T \\ \quad ... \\ \{d_{2n-1}, d_{2n}\} \to \frac{1}{\sqrt{2}}[0,0,0,...,-1,-1]^T \end{cases} \tag{7}$$

Note that group (7) comprises a total of $2n^2 - 2n$ unit vectors. The basic unit dictionary D_B is made up of all the elements of basic coordinate dictionary D_1 and all the elements of the group D_2. The basic dictionary therefore contains a total of $2n^2$ unit vectors, which are denoted by: $D_B = \{d_1, d_2, ..., d_M\}$, with $M = 2n^2$.

Similar to the 2D case, we define the scaled search directions as:

$$D_{s_m} = s_m D_B = \{s_m d_1, s_m d_2, ..., s_m d_{2n}\} \tag{8}$$

where s_m (m =0,1,…,10) are scaling coefficients which are the same as in in (6). The dictionary used for optimization is a combination of all these scaled dictionaries, that is, $D = \bigcup_{m=0}^{10} D_{s_m}$.

Outline of the CDSO Algorithm

The implementation of the proposed CDSO algorithm briefly summarised below.

Initialization: Number of decision variables (dimension);
Constraint boundary [lb, ub];
Maximum search distance (parameter R);
Maximum iteration (itMax);
Tolerate threshold (Tol)
Guessed initial condition (x0);

1. Generate the dictionary D
2. t =1;
3. x(t) = x0;
4. Find the best direction (denoted by dbest);
5. Record the current best solution xbest(t) = x(t) + dbest;
6. Check the stop criterion;
7. t = t+1; x0 = xbest(t);

Repeat 3 to 7 until the specified stop criterion is met.

---

4 Case Study – Wind Speed Forecasting

The proposed method is applied to real wind speed data, which were acquired from our research collaborators. The hourly wind speed data were collected at the Berkhout wind station, Netherlands, for the period of January-December 2004. The data were measured by the Royal Netherlands Meteorological Institute. For demonstration purpose, we use the data of November 2004 to train the network model and use the data of December of 2004 to test the model prediction performance. The training data (1-30 November 2004) and test data (1-31 December 2004) are shown in Fig. 2.

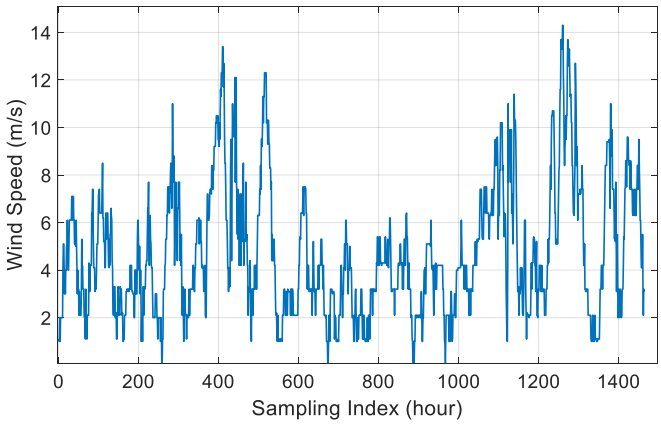

Fig. 2. Graphical illustration of the hourly wind speed data (1 November – 31 December 2004).

4.1 The Model

Let the value of wind speed at time instant t be designated by y(t) (t = 1, 2, …, N). We are interested in predicting y(t) using the previous values at the time instants t-1, t-2, … , t-n. We consider the following model:

$$y(t) = F[y(t-1), y(t-2), ..., y(t-n)] + e(t) \tag{9}$$

For convenience of description, we use $x_j(t)$ to denote y(t-j), with j = 1,2, …, n. So model (9) can be written as:

$$y(t) = F[x_1(t), x_2(t), ..., x_n(t)] + e(t) \tag{10}$$

We then use the training data to train a wavelet neural network model by the three algorithms: GA, PSO and CDSO. The following well-known sinc function (also known as the Shannon wavelet scaling function) [32] is used as the basis function to build the network model:

$$\varphi(x) = \frac{\sin \pi x}{\pi x} \tag{11}$$

With the above sinc function, the ridge type function $\psi(\boldsymbol{\theta}^T \mathbf{x})$ is:

$$\psi(\boldsymbol{\theta}^T \mathbf{x}) = \varphi(\boldsymbol{\theta}^T \mathbf{x}) = \frac{\sin\left(\pi[a_0 + a_1 x_1(t) + ... + a_n x_n(t)]\right)}{\pi[a_0 + a_1 x_1(t) + ... + a_n x_n(t)]} \tag{12}$$

where $\boldsymbol{\theta} = [a_0, ..., a_n]^T$ and $\mathbf{x} = [x_1(t), ..., x_n(t)]^T$. Note that the Shannon wavelet scaling function (11) is not differentiable, meaning that the conventional gradient descent type algorithms cannot be directly used to train the associated wavelet neural network.

4.2 Model Performance

Primary simulation results suggest that the time delay in (9) can be chosen as n =4. With this choice, the three optimization algorithms (GA, PSO, and CDSO) were used to train wavelet neural network models. The final wavelet networks trained by the three algorithms contain 22, 9, and 9 basis functions, respectively. To ensure that the "best" (i.e. better optimized) basis function is added to the network model in each iteration, both GA and PSO algorithms are run 10 times in each search iteration and the basis function that gives the best performance is included in the model.

The changes of the penalized error-to-signal ratio (PESR) for the three algorithm (GA, PSO and CDSO), calculated on the training data, are shown in Fig. 3. Note that initially PESR decreases with the increase of the number of the basis functions included in the network, but somewhere in some later stage it begins to increase due to the effect of the penalty factor (see Section 2.2). Therefore, the boosting procedure can be terminated at an iteration k^* where $PESR(k^*) > PESR(k^*-1)$, to avoid overfit-

ting. The value of k^* for PSO, CDSO and GA is 9, 9, and 22, respectively, suggesting that the best network models trained by the three algorithms should include 9, 9, and 22 basis functions, respectively.

The PESR values of the best models generated by PSO (with 9 basis functions), CDSO (with 9 basis functions) and GA (with 22 basis functions) are 0.0334, 0.0296, and 0.0238, using respectively. In terms of time complexity, it turns out that the CPU time for PSO, CDSO and GA to achieve the three best models is 48.23 s, 61.33 s and 436.70 s, respectively.

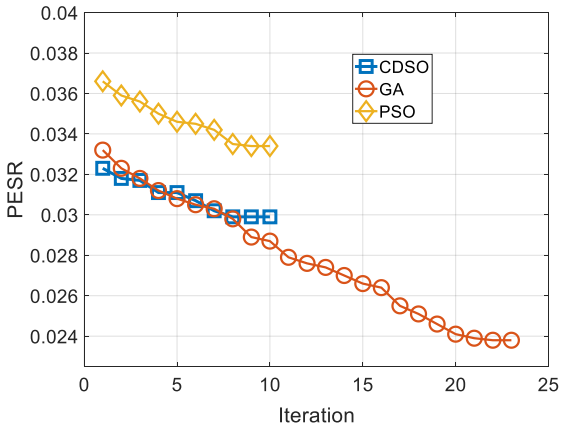

Fig. 3. The change plot of PESR on the training data.

For an illustration, the model predicted values, produced by the network trained using the CDSO algorithm over the test data (1-31 December 2004), are plotted in Fig. 4, where the corresponding measurements are also displayed for a comparison purpose. It can be seen that the obtained wavelet neural network model shows excellent prediction performance.

5 Conclusion

The main focus of the work has been paid on a type of boosted additive models. The main contributions are as follows. First, a framework of the model established based on a ridge type function was proposed. The main advantage of the proposed framework is that it allows high dimensional data modelling problems. Second, a boosted projection pursuit regression algorithm was presented. With such an algorithm, we can conveniently build a model step by step, until it achieves a good approximation. Third, we showed that either a derivative-free algorithm or an evolutionary algorithm can be used to train the networks. Given the fact that in many applications the cost functions may not be differentiable, we therefor proposed a coordinate dictionary

search (CDSO) algorithm, which works well for training the network models when integrated to the boosted projection pursuit regression algorithm.

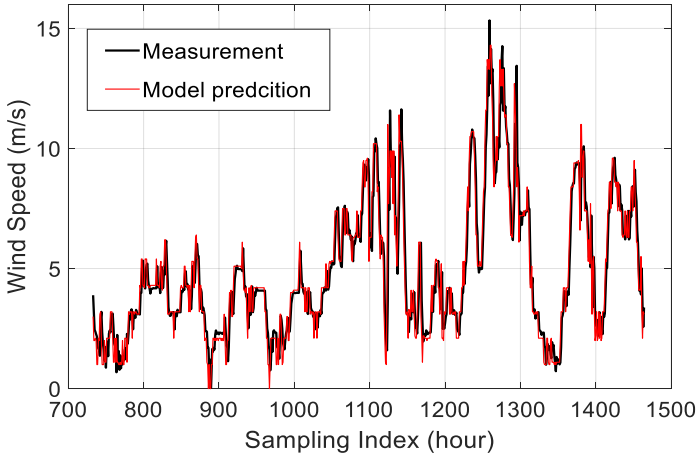

Fig. 4. A comparison of the model predicted values with the measurements on the test data (1 - 31 December 2004).

It is worth mentioning that the properties of the proposed boosted projection pursuit regression algorithm and the coordinate dictionary search algorithm have not been fully investigated. There are still several open questions that remain to be explored and answered. For example, in addition to Shannon wavelet scaling function, there perhaps exist many better alternative choices (e.g. Gaussian wavelet, radial basis function could be one of them); we will explore these in future work. We would also carry out further assessments on the performance of the proposed method and compare with traditional feedforward neural networks and state-of-the-art approaches.

While CDSO, GA and PSO algorithms all provide a zeroth-order optimization approach, meaning that they do not need gradient information, it does not necessarily mean that these methods would always be superior to gradient based algorithms. In this respect, it would be interesting to integrate the gradient boosting machine (GBM) to the boosted projection pursuit regression algorithm, to explore the advantage of GBM and investigate the potential to improve the performance of gradient-free algorithms.

Acknowledgments

This work was supported in part by the Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/I011056/1, the Platform Grant EP/H00453X/1, and

the EU Horizon 2020 Research and Innovation Programme Action Framework under grant agreement 637302.

References

1. Ljung, L.: System Identification: Theory for the User, Prentice-Hall: Upper Saddle River, N.J. (1987).
2. Sodestrom, T., Stoica, P.: System Identification, Prentice Hall: Upper Saddle River, N.J. (1988).
3. Nelles, O.: Nonlinear System Identification. Springer-Verlag: Heidelberg, Berlin (2011).
4. Billings, S.A.: Non-linear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains, Wiley: London (2013).
5. Han, J., Kamber, M.: Data Mining: Concepts and Techniques, Morgan Kaufmann, San Francisco, Calif, USA (2001).
6. Witten, I. H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann: San Francisco, Calif, USA (2005).
7. Bishop, C. M.: Neural networks for pattern recognition, Oxford University Press (1995).
8. Vapnik, V.: The Nature of Statistical Learning Theory (2nd ed.), Springer: New York (1999).
9. Friedman, J., Hastie, T., Tibshirani, R.: The Elements of Statistical Learning, Springer: New York (2001).
10. Friedman, J.: Greedy function approximation: a gradient boosting machine. Annals of Statistics 29, 1189–1232 (2001).
11. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: A statistical view of boosting. Annals of Statistics 28, 337–407 (2000).
12. Zhou, S. K., Georgescu, B., Zhou, X.S., Comaniciu, D.: Image based regression using boosting method. In Proc. 10th IEEE International Conference on Computer Vision (ICCV'05), pp. 541-548, IEEE, Beijing, China (2005).
13. De'ath, G.: Boosted trees for ecological modeling and prediction. Ecology 88(1), 243–251 (2007).
14. Zhou, S., Zhou, J., Comaniciu, D.: A boosting regression approach to medical anatomy detection. In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Minneapolis, MN (2007).
15. Zhang, X., Liang, L., Tang, X., Shum, H.: L1 regularized projection pursuit for additive model learning. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08), Anchorage, AK, USA (2008).
16. Wei, H.-L., Billings, S. A.: Generalized cellular neural networks (GCNNs) constructed using particle swarm optimization for spatio-temporal evolutionary pattern identification. International Journal of Bifurcation and Chaos 18(12), 3611-3624 (2008).
17. Wei, H.-L., Billings, S. A., Zhao, Y., Guo, L.: Lattice dynamical wavelet neural networks implemented using particle swarm optimization for spatio–temporal system identification. IEEE Transactions on Neural Networks 20(1), 181-185 (2009).
18. Wei, H.-L., Billings, S. A., Zhao, Y., Guo, L.: An adaptive wavelet neural network for spatio-temporal system identification. Neural Networks 23 (10), 1286-1299 (2010).
19. Friedman, J. H., Stuetzle, W.: Projection pursuit regression. J. Amer. Statist. Assoc. 76(376), 817–823 (1981).

20. Li, S., Ma, K., Jin, Z., Zhu, Y.: A new flood forecasting model based on SVM and boosting learning algorithms. In Proc. IEEE Congress on Evolutionary Computation (CEC'16), pp.1343–1348, Vancouver, BC, Canada (2016).
21. Zhang, D., Zhang, Y., Niu, Q., Qiu, X.: Rolling forecasting forward by boosting heterogeneous kernels. In: Phung, D., Tseng, V., Webb, G., Ho, B., Ganji, M., Rashidi, L. (eds), Advances in Knowledge Discovery and Data Mining (AKDD'18). Lecture Notes in Computer Science, vol 10937. Springer, Cham (2018).
22. Torres-Barrána, A., Alonsoa, A., Dorronsoroa, J. R.: Regression tree ensembles for wind energy and solar radiation prediction. Neurocomputing 326–327, 151-160 (2019).
23. Mallat, S.: A Wavelet Tour of Signal Processing. Academic Press: San Diego (1998).
24. Wei, H.-L., Billings, S. A.: A unified wavelet-based modelling framework for non-linear system identification: the WANARX model structure. International Journal of Control 77(4), 351–366 (2004).
25. Billings, S.A, Wei, H.-L.: The wavelet-NARMAX representation: a hybrid model structure combining polynomial models with multiresolution wavelet decompositions. International Journal of Systems Science 36(3), 137–152 (2005).
26. Wei, H.-L., Billings, S.A.: Long term prediction of non-linear time series using multiresolution wavelet models. International Journal of Control 79(6), 569–580 (2006).
27. Li, Y., Cui, W., Guo, Y.Z. et al.: Time-varying system identification using an ultra-orthogonal forward regression and multiwavelet basis functions with applications to EEG. IEEE Transactions on Neural Networks and Learning Systems 29(7), 2960-2972 (2018).
28. Li, Y., Lei, M., Guo, Y., Hu, Z, Wei, H.-L.: Time-varying nonlinear causality detection using regularized orthogonal least squares and multi-wavelets with applications to EEG. IEEE Access 6, 17826-17840 (2018).
29. Li, Y., Lei, M., Cui, W et al.: A parametric time frequency-conditional Granger causality method using ultra-regularized orthogonal least squares and multiwavelets for dynamic connectivity analysis in EEGs. IEEE Transactions on Biomedical Engineering (in press).
30. T. Back, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press: Oxford, UK (1996).
31. Kennedy, J. and Eberhart, R.: Particle swarm optimization. In: Proc. IEEE Conf. Neural Networks, vol. 4, pp.1942–1948, Piscataway, NJ (1995).
32. Cattani, C.: Shannon wavelets theory. Mathematical Problems in Engineering, art. no. 164808 (2008).