11-2018

Short-Term Wind Speed Forecasting via Stacked Extreme Learning Machine With Generalized Correntropy

Xiong Luo
University of Science and Technology Beijing (USTB), xluo@ustb.edu.cn

Jiankun Sun
University of Science and Technology Beijing

Long Wang
University of Science and Technology Beijing

Weiping Wang
University of Science and Technology Beijing

Wenbing Zhao
Cleveland State University, w.zhao1@csuohio.edu

See next page for additional authors

Short-Term Wind Speed Forecasting via Stacked Extreme Learning Machine With Generalized Correntropy

Xiong Luo, *Member, IEEE*, Jiankun Sun, Long Wang, *Member, IEEE*, Weiping Wang, Wenbing Zhao, *Senior Member, IEEE*, Jinsong Wu, *Senior Member, IEEE*, Jenq-Haur Wang, and Zijun Zhang, *Member, IEEE*

***Abstract*—Recently, wind speed forecasting as an effective computing technique plays an important role in advancing industry informatics, while dealing with these issues of control and operation for renewable power systems. However, it is facing some increasing difficulties to handle the large-scale dataset generated in these forecasting applications, with the purpose of ensuring stable computing performance. In response to such limitation, this paper proposes a more practical approach through the combination of extreme-learning machine (ELM) method and deep-learning model. ELM is a novel computing paradigm that enables the neural network (NN) based learning to be achieved with fast training speed and good generalization performance. The stacked ELM (SELM) is an advanced ELM algorithm under deep-learning framework, which works efficiently on memory consumption decrease. In this paper, an enhanced SELM is accordingly developed via replacing the Euclidean norm of the mean square error (MSE) criterion in ELM with the generalized correntropy criterion to further improve the forecasting performance. The advantage of the enhanced SELM with generalized correntropy to achieve better forecasting performance mainly relies on the following aspect. Generalized correntropy is a stable and robust nonlinear similarity measure while employing machine learning method to forecast wind speed, where the outliers may exist in some industrially measured values. Specifically, the experimental results of short-term and ultra-short-term forecasting on real wind speed data show that the proposed approach can achieve better computing performance compared with other traditional and more recent methods.**

***Index Terms*—Autoencoder, generalized correntropy, stacked extreme learning machine (SELM), wind speed forecasting.**

I. Introduction

WIND power energy has been one of the most widely used renewable energy resources in the world on the condition that wind power is renewable without the limitation of use and free of pollution [1]. Since the importance of renewable resources is clear, and many countries pay more attention to seeking for the efficient uses of them. Then, many research works have been conducted to explore the wind energy better. Wind speed forecasting is one of the most significant research directions since the strength of wind power is positively correlated with wind speed. Efficient wind speed forecasting is of great benefits to the utilization of wind energy [2], [3].

Generally, current wind forecasting models can be classified into two categories, i.e., physical models and statistical models. The former evaluate the wind speed via constructing a complex mathematical model with many parameters on the basis of these measured values. Here, the numeric weather prediction (NWP) is a well-known method, but it requires a large number of computational resources and time consumption [4]. Compared with physical models, statistical models utilize fewer parameters and the process of modeling is simpler. In many cases, these approaches treat previous history data as input data to model the short-term forecasting. There are a number of traditional statistical models used for short-term wind speed forecasting, such as autoregressive moving average, support vector machine regression, artificial neural network (ANN), and many others [5]–[7]. Recently, some popular learning algorithms have also

been applied in this field. For example, extreme learning machine (ELM) addresses this issue effectively [8], since it is considered as a novel computing paradigm enabling the neural network (NN) based learning with fast training speed and good generalization performance. Furthermore, with the advancement of deep learning used in the field of industry informatics [9]–[11], some models, such as deep neural network (DNN) [12], deep belief network [13], and deep Boltzmann machine [14], have been employed to forecast wind speed.

However, in statistical models, some large-scale and complex datasets are always fed into the model for training, which may cause high memory occupation. To avoid such limitations, the stacked extreme learning machine (SELM) was developed in the deep-learning structure, through the combination of the advantages of ELM [15]. SELM splits a large NN into several serially computed smaller ones to achieve small memory occupation, then more hidden neurons can be accordingly added to each layer in NN. Compared with other algorithms, SELM achieves higher learning accuracy and less memory occupation [15].

Among all renewable energy sources, wind energy is unstable due to many uncertain factors, including weather, temperature, altitude, and so on. These uncertainties and some other random fluctuations from inaccurate measurements always affect the quality of observation data, which may come with some outliers. Therefore, the wind forecasting modeling is challenging. The correntropy as a nonlinear similarity measure can be used in developing some machine learning algorithms [16], [17]. Furthermore, the generalized correntropy is a generalization form of correntropy, which substitutes generalized Gaussian density (GGD) function for the Gaussian kernel in the correntropy [18]. This paper, thus, proposes a more practical approach, an enhanced SELM, via incorporating the generalized correntropy into the SELM framework to deal with the issue of forecasting wind speed. In addition to generalized correntropy, some other techniques are also employed in the algorithm implementation to further improve the performance. The ELM-based autoencoder (ELM-AE) as a feature extraction strategy [19], is introduced to refine the input data, and the principle component analysis (PCA) technique in SELM is used to extract the major information of the hidden layers to reduce the calculation in the next layer [15].

Our contributions in this paper are as follows.

1) Instead of mean square error (MSE), generalized correntropy is used as the measure in ELM framework. The generalized correntropy is with a robust performance for outliers, and thus, it can further improve the wind speed forecasting under the consideration that various outliers may exist in the dataset.
2) The L_2 regularization is adopted on the cost function of ELM in the forecasting process to improve the stability and generalization performance further.
3) Specifically, through the comparisons with some recent methods, such as deep-learning-based DNN [12], the effectiveness of our proposed model is validated on the short-term and ultra-short-term forecasting experiments with real wind speed data from a commercial wind farm located in Shandong Province, China.

The rest of this paper is organized as follows. Section II provides an analysis of the backgrounds related to these methods used here. Section III presents our enhanced SELM approach and Section IV discusses the experimental results. Conclusion and further discussions are given in Section V.

II. Backgrounds

A. Stacked Extreme Learning Machine (SELM)

There is a single-hidden-layer feedforward network (SLFN). Let L be the number of the hidden neurons in this NN. And the input weights $\boldsymbol{w}_j$ and biases b_j of the jth hidden layer node $(1 \leqslant j \leqslant L)$ are randomly generated. For N training samples $\{(\boldsymbol{x}_i, \boldsymbol{t}_i)|\boldsymbol{x}_i \in \mathbb{R}^q, \boldsymbol{t}_i \in \mathbb{R}^n\}_{i=1}^N$ and the activation function $g(\cdot)$, where q is the number of the input features, n denotes the number of the output neural units, the main idea of ELM is to calculate the output weight vector $\boldsymbol{\beta}$ between the hidden layer and the output layer in SLFN by [20]

$$\boldsymbol{\beta} = \left(\mathbf{H}^\top \mathbf{H}\right)^{-1} \mathbf{H}^\top \mathbf{T} \tag{1}$$

where $\mathbf{H}$ represents the hidden layer output matrix, $\mathbf{T} = [[\boldsymbol{t}_i]_{i=1}^N]^\top$ denotes the target matrix.

Motivated by deep-learning models, SELM was proposed via a stacked ELM with multilayer NN structure [15]. SELM partitions a large ELM NN into multiple stacked small ELMs. The first layer is with an original ELM architecture, and the parameters of hidden layer neurons in the first layer are absolutely randomly generated, while those of the rest layers may be generated in a random way, only partially, on account that some parameters, such as $\boldsymbol{\beta}$, are propagated after being cut down to a lower dimension. The information of the input data is propagated to the next layer after the previous layer is trained, then the input information is transmitted from layer to layer until the last one. Hence, the multilayer ELM architecture constructs the deep-learning model. Provided that the first hidden layer output is denoted as $\mathbf{H}_1$, then the optimization problem of the first layer can be expressed as follows [20]:

$$\min_{\boldsymbol{\beta}_1} \left\{ \mathbb{L}_1 = \frac{1}{2}\|\boldsymbol{\beta}_1\|^2 + \frac{C}{2}\|\mathbf{T} - \mathbf{H}_1\boldsymbol{\beta}_1\|^2 \right\} \tag{2}$$

where $\boldsymbol{\beta}_1$ is the output weight vector of the first layer, and C is the tradeoff parameter between the training error and the norm of output weights.

Considering $\frac{\partial}{\partial \boldsymbol{\beta}_1}\mathbb{L}_1 = 0$, then $\boldsymbol{\beta}_1$ is obtained as follows:

$$\boldsymbol{\beta}_1 = \left(\frac{\mathbf{I}}{C} + \mathbf{H}_1^\top \mathbf{H}_1\right)^{-1} \mathbf{H}_1^\top \mathbf{T} \tag{3}$$

where $\mathbf{H}_1 = [g((\boldsymbol{w}_j^{(1)})^\top \boldsymbol{x}_i + b_j^{(1)})]_{i=1,\ldots,N;j=1,\ldots,L}$. Here, $\mathbf{W}_1 = [\boldsymbol{w}_j^{(1)}]_{j=1}^L$ and $\mathbf{b}_1 = [b_j^{(1)}]_{j=1}^L$ are the input weight and bias vector in the first layer, respectively.

There may be redundant information in the first hidden layer neurons. Hence, the dimension of $\boldsymbol{\beta}$ can be reduced into a lower level, i.e., from L to l, where L denotes the original feature dimension, l can be a specified value, and $L > l$. As the eigenvectors matrix $\mathbf{U} \in \mathbb{R}^{L\times L}$ is generated through PCA dimension reduction technique, we can obtain a new matrix $\mathbf{U}_r \in \mathbb{R}^{L\times l}$,

which consists of the eigenvectors corresponding the top l eigenvalues. The reduced hidden layer output matrix and the reduced output weight matrix can be expressed as follows:

$$\mathbf{H}_r = \mathbf{H}\mathbf{U}_r \tag{4}$$

$$\boldsymbol{\beta}_r = \mathbf{U}_r^\top \boldsymbol{\beta}. \tag{5}$$

When the number of hidden neurons is reduced to l, only $L - l$ hidden neurons in the next layer are inevitably, randomly generated and a new $\mathbf{H}_{r'}$ can be calculated. Then, this hidden layer output can be formed as follows:

$$\mathbf{H} = [\mathbf{H}_r, \mathbf{H}_{r'}]. \tag{6}$$

Here, $\boldsymbol{\beta}$ in this layer can be described through (3), and then, this $\boldsymbol{\beta}$ can be reduced in the same way as mentioned earlier. Conducting the iteration until the last layer and the dimension reduction procedure is not needed in the last layer and the output can be achieved:

$$\mathbf{O} = \mathbf{H}\boldsymbol{\beta} \tag{7}$$

where $\boldsymbol{\beta}$ is the value vector calculated until the last layer.

B. Extreme Learning Machine-Based Autoencoder (ELM-AE)

Traditional autoencoders usually choose the backpropagation algorithm for training, while ELM-AE directly uses ELM structure as the autoencoder [19]. Then, ELM-AE represents the original input data with a new useful feature representation via projecting the input features into a different dimensional space. ELM-AE follows the same basic form of the ELM, while taking the input data $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]^\top$ as the output data. Then

$$\mathbf{X} = \mathbf{H}\boldsymbol{\beta}. \tag{8}$$

Similarly, a random weight matrix and a random bias matrix are also required. The better choice in ELM-AE is to set the weight matrix $\mathbf{W} = [\boldsymbol{w}_j]_{j=1}^L$ and the bias matrix $\mathbf{b} = [b_j]_{j=1}^L$ orthogonally by $\mathbf{H} = \left[g(\boldsymbol{w}_j^\top \boldsymbol{x}_i + b_j)\right]_{i=1,\ldots,N;j=1,\ldots,L}$. Here, $\mathbf{H}$ denotes the hidden layer outputs. In addition, $\boldsymbol{w}_j^\top \boldsymbol{w}_j = 1$ and $b_j^\top b_j = 1$.

The higher dimensional projection $\boldsymbol{\beta}$ of input data can be calculated by

$$\boldsymbol{\beta} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^\top \mathbf{H}\right)^{-1} \mathbf{H}^\top \mathbf{X} \tag{9}$$

where $\mathbf{X}$ is the output of ELM structure, but it is the same with the input data in ELM-AE.

C. Generalized Correntropy

In information theoretic learning, correntropy has been a widely used nonlinear similarity measure method due to its robustness [16]. While in some cases, the default Gaussian kernel function in correntropy cannot achieve the best performance. Therefore, a universal form of the correntropy was proposed via substituting GGD function for the Gaussian kernel in correntropy [18], and that generalized correntropy has been used in machine learning [21].

Definition: Considering two random variables D and F, the original correntropy is defined by

$$V(D,F) = \mathbb{E}\left[\langle \Phi(D), \Phi(F)\rangle\right] = \mathbb{E}\left[\kappa(D,F)\right] \tag{10}$$

where $\mathbb{E}$ denotes the mathematical expectation, κ denotes a kernel function which is usually a Gaussian kernel in correntropy. Moreover, Φ is a nonlinear mapping from input data to a high-dimensional Hilbert space related to κ.

The well-known zero-mean GGD function is defined by

$$\begin{aligned} G_{\alpha,\gamma}(D,F) = g_{\alpha,\gamma}(D-F) &= \frac{\alpha}{2\gamma\Gamma(\frac{1}{\alpha})} e^{-\left(\frac{|D-F|}{\gamma}\right)^\alpha} \\ &= \lambda e^{-\left(\frac{|D-F|}{\gamma}\right)^\alpha} \end{aligned} \tag{11}$$

where $\alpha > 0$ represents the shape parameter, $\gamma > 0$ represents the scale parameter, λ is used for a concise form, and $\Gamma(\cdot)$ denotes the gamma function. In the generalized form, both Gaussian and Laplacian distributions are contained as the special cases when α is equal to 2 and 1, respectively.

For generalized correntropy, the definition can be expressed as follows:

$$V(D,F) = \mathbb{E}\left[G_{\alpha,\gamma}(D,F)\right]. \tag{12}$$

Let $\{(d_i, f_i)\}_{i=1}^N$ be N data drawn from the joint probability density function. The generalized correntropic loss (GC-loss) function which is similar to correntropic loss (C-loss) in correntropy can be defined by:

$$\begin{aligned} \mathbb{J}_{\mathrm{GC-loss}}(D,F) &= G_{\alpha,\gamma}(0) - G_{\alpha,\gamma}(D,F) \\ &= G_{\alpha,\gamma}(0) - \mathbb{E}\left[g_{\alpha,\gamma}(d_i - f_i)\right] \\ &= \lambda\left\{1 - \mathbb{E}\left[e^{-\mu|e(i)|^\alpha}\right]\right\} \end{aligned} \tag{13}$$

where μ is equal to $\frac{1}{\gamma^\alpha}$ and $e(i)$ denotes $(d_i - f_i)$ of the ith sample.

Remark: The GC-loss can be used as the cost function of an adaptive system training problem through minimizing this function. Our proposed method is mainly designed using this property.

III. Proposed Algorithms

Here, through the use of generalized correntropy in ELM, the computing process of ELM with generalized correntropy is developed first. Then, after incorporating it into SELM, our proposed approach is presented.

A. Generalized Correntropy-Based Extreme Learning Machine (GC-Based ELM)

The generalized correntropy can be employed as the cost function of ELM via substituting the objective function for MSE used in ELM. In original cost function of ELM, the solution of the parameters is to minimize the MSE between the forecasting output and the target output. Correspondingly, the parameter problem can be also solved through minimizing the GC-loss function. Furthermore, after adding frequently-used L_2

regularization term to the cost function, we can obtain

$$\mathbb{J}_{\mathrm{GC-loss}}(\boldsymbol{\beta}) = \min_{\boldsymbol{\beta}} \left\{ \lambda \left(1 - \frac{1}{N} \sum_{i=1}^{N} e^{-\mu |\boldsymbol{t}_i - \boldsymbol{y}_i|^{\alpha}} \right) + \eta \|\boldsymbol{\beta}\|_F^2 \right\} \tag{14}$$

where $\boldsymbol{\beta}$ denotes the hidden layer output weight, $\boldsymbol{t}_i$ denotes the output of the sample $\boldsymbol{x}_i$ in training dataset, η is the regularization parameter, $\|\cdot\|_F^2$ denotes the Frobenius norm, and $\boldsymbol{y}_i$ is the forecasting output from ELM network:

$$\boldsymbol{y}_i = \boldsymbol{h}_i \boldsymbol{\beta} \tag{15}$$

where $\boldsymbol{h}_i$ is the hidden layer output of training sample $\boldsymbol{x}_i$.

Through minimizing the cost function, the optimal solution of (14) at the sth iteration can be expressed as follows:

$$\boldsymbol{\beta}_{s+1} = (\mathbf{H}^{\top}\mathbf{P}\mathbf{H} + \sigma \mathbf{I})^{-1}\mathbf{H}^{\top}\mathbf{P}\mathbf{T} \tag{16}$$

where σ is another regularization parameter, $\mathbf{P}$ denotes a diagonal matrix and each value of the diagonal element p_{ii} $(i = 1, 2, \ldots, N)$ is defined as follows:

$$p_{ii} = \sum_{i=1}^{N} \frac{\lambda\mu\alpha}{N} e^{-\mu |e(i)|^{\alpha}} |e(i)|^{\alpha - 2} \tag{17}$$

where $e(i) = \boldsymbol{t}_i - \boldsymbol{h}_i \boldsymbol{\beta}$.

The deduction for the above result is as follows.

Considering $\frac{\partial}{\partial \boldsymbol{\beta}} \mathbb{J}_{\mathrm{GC-loss}} = 0$, then

$$\begin{aligned}
&-\frac{\lambda\mu\alpha}{N} \sum_{i=1}^{N} \left(e^{-\mu |e(i)|^{\alpha}} |e(i)|^{\alpha - 1} \mathrm{sign}(e(i)) \boldsymbol{h}_i \right) + 2\eta\boldsymbol{\beta} = 0 \\
&\Rightarrow -\sum_{i=1}^{N} \left(\frac{\lambda\mu\alpha}{N} e^{-\mu |e(i)|^{\alpha}} |e(i)|^{\alpha - 2} e(i) \boldsymbol{h}_i \right) + 2\eta\boldsymbol{\beta} = 0 \\
&\Rightarrow \left(\sum_{i=1}^{N} \boldsymbol{h}_i^{\top} p_{ii} \boldsymbol{h}_i + \sigma \right) \boldsymbol{\beta} = \sum_{i=1}^{N} \boldsymbol{h}_i^{\top} p_{ii} \boldsymbol{t}_i \\
&\Rightarrow \boldsymbol{\beta} = \left(\sum_{i=1}^{N} \boldsymbol{h}_i^{\top} p_{ii} \boldsymbol{h}_i + \sigma \right)^{-1} \times \left(\sum_{i=1}^{N} \boldsymbol{h}_i^{\top} p_{ii} \boldsymbol{t}_i \right) \\
&\Rightarrow \boldsymbol{\beta} = (\mathbf{H}^{\top}\mathbf{P}\mathbf{H} + \sigma \mathbf{I})^{-1}\mathbf{H}^{\top}\mathbf{P}\mathbf{T}
\end{aligned} \tag{18}$$

where $\mathbf{H} = [[\boldsymbol{h}_i]_{i=1}^{N}]^{\top}$ and $\sigma = \frac{2\eta N}{\lambda\mu\alpha}$ is the regularization parameter.

Considering a large-scale dataset with training samples $\{(\boldsymbol{x}_i, \boldsymbol{t}_i) | \boldsymbol{x}_i \in \mathbb{R}^q, \boldsymbol{t}_i \in \mathbb{R}^n\}_{i=1}^{N}$, our GC-based ELM can be described in Algorithm 1.

Algorithm 1: GC-based ELM.

Input:
A large training dataset:
$\{(\boldsymbol{x}_i, \boldsymbol{t}_i) | \boldsymbol{x}_i \in \mathbb{R}^q, \boldsymbol{t}_i \in \mathbb{R}^n, i = 1, 2, \ldots, N\}$;
The number of hidden layer nodes: L;
ELM activation function: $g(\cdot)$;
The termination threshold: τ;
The maximum number of iterations: $N_{\max}$;
The shape parameter of generalized correntropy: α;
The scale parameter of generalized correntropy: γ;
The regularization parameter: σ.
Output:
The forecasting output matrix $\mathbf{O}$;
The hidden layer weight $\boldsymbol{\beta}$.
(a) Generate the random input weight matrix $\mathbf{W} = [\boldsymbol{w}_j]_{j=1}^{L}$ and the random hidden layer bias $\mathbf{b} = [b_j]_{j=1}^{L}$.
(b) Compute the hidden layer output matrix $\mathbf{H}$ by:

$$\mathbf{H} = \left[g(\boldsymbol{w}_j^{\top} \boldsymbol{x}_i + b_j) \right]_{i=1,\ldots,N; j=1,\ldots,L}.$$

(c) Calculate the output weight $\boldsymbol{\beta}$:

$$\boldsymbol{\beta} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^{\top}\mathbf{H} \right)^{-1} \mathbf{H}^{\top}\mathbf{T}.$$

(d) Update the hidden layer weight $\boldsymbol{\beta}$:
(d-1) Update the auxiliary matrix $\mathbf{P} = \mathrm{diag}(p_{11}, p_{22}, \ldots, p_{NN})$ by:

$$p_{ii}^{s} = \sum_{i=1}^{N} \frac{\lambda\mu\alpha}{N} e^{-\mu |e(i)|^{\alpha}} |e(i)|^{\alpha - 2}$$

where $e(i) = \boldsymbol{t}_i - \boldsymbol{h}_i \boldsymbol{\beta}_s$ and s is the number of iterations;
(d-2) Calculate $\mathbb{J}_{\mathrm{GC-loss}}(\boldsymbol{\beta}_s)$ through (14);
(d-3) Judge whether the iteration termination condition is satisfied, that is when $s > N_{\max}$ or $\Delta\mathbb{J}_{\mathrm{GC-loss}}(\boldsymbol{\beta}_s) = |\mathbb{J}_{\mathrm{GC-loss}}(\boldsymbol{\beta}_{s+1}) - \mathbb{J}_{\mathrm{GC-loss}}(\boldsymbol{\beta}_s)| < \tau$, jump out and end the algorithm;
(d-4) Update $\boldsymbol{\beta}$:

$$\boldsymbol{\beta}_{s+1} = (\mathbf{H}^{\top}\mathbf{P}\mathbf{H} + \sigma \mathbf{I})^{-1}\mathbf{H}^{\top}\mathbf{P}\mathbf{T}$$

jump to (d-1).
(e) Output the forecasting matrix: $\mathbf{O} = \mathbf{H}\boldsymbol{\beta}$.

B. Generalized Correntropy-Based Stacked Extreme Learning Machine (GC-Based SELM)

According to these facts, including small memory requirement of SELM, the data preprocessing ability of ELM-AE, and the steady-state performance to outliers in generalized correntropy, the GC-based SELM is accordingly proposed on the basis of the deep-learning model. In the first layer of our model, the above developed GC-based ELM is applied after the ELM-AE model. Then, after the dimension reduction process of SELM, the model gets into the next layer. From layer 2 to the last layer, we repeat the similar process in the first layer, and the generated hidden layer neurons are only left after reduction.

Fig. 1 demonstrates the architecture of GC-based SELM. The input weight is optimized through the ELM-AE network, and then the hidden layer weight $\boldsymbol{\beta}$ can be calculated initially. After minimizing the GC-loss, the output of GC-based ELM is delivered to the PCA dimension reduction module. Then, the reduced $\boldsymbol{\beta}$ is used on SELM.

The description for our proposed GC-based SELM is shown in Algorithm 2.

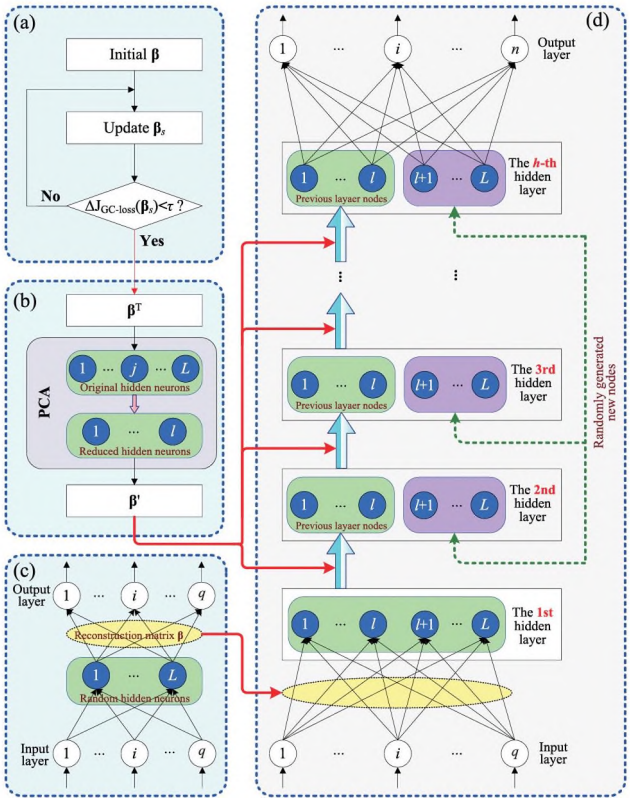


Fig. 1. Architecture of GC-based SELM. (a) GC-based ELM. (b) PCA. (c) ELM-AE. (d) SELM.

IV. Experimental Results and Discussions

In this section, we demonstrate the performance comparisons between our GC-based SELM and some other data-driven forecasting methods, including ANN [6], ELM [8], SELM [15], and more recent algorithm DNN [12]. The experimental results are achieved on dealing with short-term and ultra-short-term wind speed forecasting tasks.

A. Dataset and Experiment Environment Description

The wind speed data utilized in this paper are provided by a commercial wind farm located in Shandong Province, China. Here, there are two time series datasets. One is from May 13, 2015 00:00 to May 13, 2015 03:19. The other is from February 1, 2014 00:00 to June 27, 2014 23:50. After initially analyzing these data, we find that there are some outliers. The outliers are generated mainly from the imprecise observations, the change of the weather, and the differences of temperature at different time in accordance with the data provider. On the first time series, the time interval is chosen as 1 s, and we predict wind speed in the next 5 s, while on the other time series, the time interval is chosen as 10 min, and the average wind speed of the next 10, 30, 60 min will be predicted, respectively. Considering the size of the dataset, the same method is adopted to select the training and test samples in the following two experiments. For the dataset, random 7000 time series in the first 10 000 time series are chosen as the training data and random 1500 data in the next 5000 samples are set as validation data, while 1500 test data are sampled from the other 5000 data. The experiments are conducted on the MATLAB R2016a environment running on an Inter(R) Core(TM) i5-4200 M, 2.50 GHZ, 8.00 GB RAM Computer.

Algorithm 2: GC-based SELM.

Input:
The same input in **Algorithm 1**;
The number of hidden layers: h.
Output:
The forecasting output matrix $\mathbf{O}$.
(a) Apply GC-based ELM on the first layer:
(a-1) Generate the random input weight matrix $\mathbf{W} = [\boldsymbol{w}_j]_{j=1}^{L}$ and the random hidden layer bias $\mathbf{b} = [b_j]_{j=1}^{L}$ for layer 1.
(a-2) Reconstruct $\mathbf{W}' = [\boldsymbol{w}'_j]_{j=1}^{L}$ and $\mathbf{b}' = [b'_j]_{j=1}^{L}$ through ELM-AE.
(a-3) Compute the hidden layer output matrix $\mathbf{H}$ by:

$$\mathbf{H} = \left[g((\boldsymbol{w}'_j)^\top \boldsymbol{x}_i + b'_j)\right]_{i=1,\ldots,N;j=1,\ldots,L}.$$

(a-4) Calculate the output weight $\boldsymbol{\beta}$ through **Algorithm 1**.
(b) Cut down the number of the hidden neurons:
After applying dimension reduction technique on $\boldsymbol{\beta}$, we can record the reduced number l of hidden layer nodes, and the eigenvectors matrix $\mathbf{U}$. Then, $\mathbf{H}'$ and $\boldsymbol{\beta}'$ can be calculated by:

$$\boldsymbol{\beta}' = \mathbf{U}^\top \boldsymbol{\beta},$$

$$\mathbf{H}' = \mathbf{H}\mathbf{U}.$$

(c) Employ SELM from layer 2 to layer h:
(c-1) The input weight $\mathbf{W}_k$ and the the hidden layer bias $\mathbf{b}_k$ are generated randomly for the layer k with the hidden number $L - l$, and the corresponding $\mathbf{H}_{r'}$ with $L - l$ hidden neurons can be calculated.
(c-2) The new hidden layer matrix: $\mathbf{H} = [\mathbf{H}', \mathbf{H}_{r'}]$.
(c-3) Jump to (a-4).
(d) Compute the output of this architecture in the final layer: $\mathbf{O} = \mathbf{H}\boldsymbol{\beta}$.

B. Metrics

Here, four metrics are used on the evaluation of forecasting performance, i.e., mean absolute error (MAE), MSE, root mean square error (RMSE), and mean absolute percentage error (MAPE), like the usual operation in [22], [23]. They are as follows:

$$\mathrm{MAE} = \frac{1}{T}\sum_{t=1}^{T}|y_t - p_t| \tag{19}$$

$$\mathrm{MSE} = \frac{1}{T}\sum_{t=1}^{T}(y_t - p_t)^2 \tag{20}$$

TABLE I
COMPARISONS OF FORECASTING PERFORMANCE (MAE)

Second	Persistence	ANN		DNN		ELM		SELM		GC-based SELM	
	MAE	MAE	Imp (%)	MAE	Imp (%)	MAE	Imp (%)	MAE	Imp (%)	MAE	Imp (%)
1	0.7670	0.6670	13.03	0.7121	7.16	0.6851	10.68	0.6930	9.65	**0.6465**	15.71
2	0.8773	0.7116	18.89	0.7427	15.35	0.7383	15.85	0.7471	14.84	**0.6990**	20.32
3	0.9135	0.7466	18.26	0.7569	17.14	0.7553	17.32	0.7578	17.05	**0.7167**	21.55
4	0.9484	0.7790	17.86	0.7778	17.99	0.7892	16.78	0.7889	16.82	**0.7526**	20.65
5	0.9817	0.7706	21.50	0.7631	22.27	0.7759	20.97	0.7802	20.53	**0.7423**	24.39

TABLE II
COMPARISONS OF FORECASTING PERFORMANCE (MSE)

Second	Persistence	ANN		DNN		ELM		SELM		GC-based SELM	
	MSE	MSE	Imp (%)	MSE	Imp (%)	MSE	Imp (%)	MSE	Imp (%)	MSE	Imp (%)
1	0.9608	0.7345	23.55	0.8257	14.07	0.7513	21.81	0.7776	19.07	**0.6803**	29.19
2	1.2266	0.8209	33.07	0.8867	27.71	0.8805	28.22	0.8907	27.38	**0.7914**	35.48
3	1.3068	0.8988	31.22	0.9149	29.99	0.9101	30.36	0.9235	29.33	**0.8246**	36.90
4	1.4321	0.9430	34.15	0.9466	33.90	0.9755	31.88	0.9869	31.08	**0.8902**	37.84
5	1.5047	0.9302	38.18	0.9220	38.73	0.9381	37.65	0.9604	36.18	**0.8733**	41.96

$$\mathrm{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t - p_t)^2} \tag{21}$$

$$\mathrm{MAPE} = \frac{1}{T}\sum_{t=1}^{T}\frac{|y_t - p_t|}{y'_t} \tag{22}$$

where y_t denotes the observed wind speed at time point t, p_t is the forecasting value at time point t, and $y'_t = \frac{1}{T}\sum_{t=1}^{T} y_t$.

To compare forecasting performance between our model and other traditional models, the persistent forecasting model is set as the benchmark. Then, the relative performance based on the benchmark can be calculated through $\mathrm{Imp} = \frac{E_p - E}{E_p}$, where E is one of the error metrics listed from (19) to (22), and E_p is the corresponding E of the persistent model.

C. Parameters Selection

In SELM, there are three parameters that should be set in advance. Applying generalized correntropy into SELM, extra parameters are added into the model. All the best choices of these parameters in this paper are achieved through grid searching with cross-validation. The parameters and their bounds are listed as: the regularization coefficient C from 2^{-20} to 2^{20}, the number of hidden layers h from 2 to 7, the number of hidden neurons in each hidden layer L from 50 to 2000, the shape parameter α from 0.5 to 4, the scale parameter γ from 0.001 to 0.5, and another regularization parameter σ from 2^{-20} to 2^{20}. More detailed discussions for these parameters are provided in Section IV-D. In the cross-validation phase, the original validation dataset is split into tenfolds for the parameters selection. As for the two initial variables $\boldsymbol{w}$ and b in each hidden layer, they are recommended to be randomly generated by a uniform distribution ranged within $[-1, 1]$ and within $[0, 1]$, respectively.

D. Multistep Second-Level Forecasting

Here, we show our model performance for the multistep prediction on second-level dataset. The input can be expressed as $X = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{n\times m}$, and the output is $Y = [y_1, y_2, \ldots, y_n] \in \mathbb{R}^{n\times r}$, where n is the number of the training or test data, m denotes the number of the input features, and r denotes the number of the output neurons. Here, m is set to 10 as the usual implementation in [23], and it means that the wind speed at a time point and the next 9 wind speed data, i.e., $x_i = [wp_i, wp_{i+1}, \ldots, wp_{i+9}]$. And r is set to 5, and it means that the wind speed of next 5 s after 10 s in the input samples, i.e., $y_i = [wp_{i+10}, wp_{i+11}, \ldots, wp_{i+14}]$.

In this experiment, the parameters are set as follows. The regularization coefficient C in ELM, SELM, and GC-based SELM is equal to 2^{10}, and another induced regularization parameter σ in GC-based ELM is set to 2^{-10}. The number of hidden layers and total hidden neurons in each layer in SELM and GC-based SELM are 2 and 500, respectively. The scale parameter γ and the shape parameter α induced by generalized correntropy are set to 0.05 and 3, respectively. The number of hidden neurons in ANN and ELM are chosen as 10 and 500, respectively. The depth of DNN is set as 4, and only hyperbolic tangent function $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ is used as an activation function. The number of hidden neurons of each layer is 35-35-35.

MAE, MSE, RMSE, and MAPE are four different metrics used to compare the measured value and the real value. Here, our method is used to calculate multistep wind speed and the results on those four metrics are compared to persistent model, ANN, DNN, ELM, and SELM. The 5-step forecasting result is shown from Table I–IV.

In these tables, we can observe that all of these models achieve low measures of forecasting errors. Among all the four metrics, GC-based SELM performs a lower value than other models, and it means that a higher accuracy is achieved. In addition,

TABLE III
COMPARISONS OF FORECASTING PERFORMANCE (RMSE)

Second	Persistence	ANN		DNN		ELM		SELM		GC-based SELM	
	RMSE	RMSE	Imp (%)	RMSE	Imp (%)	RMSE	Imp (%)	RMSE	Imp (%)	RMSE	Imp (%)
1	0.9802	0.8571	12.57	0.9087	7.30	0.8668	11.58	0.8818	10.04	**0.8248**	15.85
2	1.1075	0.9060	18.19	0.9416	14.98	0.9383	15.27	0.9438	14.78	**0.8896**	19.67
3	1.1431	0.9481	17.07	0.9565	16.33	0.9539	16.55	0.9610	15.94	**0.9081**	20.56
4	1.1967	0.9711	18.85	0.9729	18.70	0.9877	17.47	0.9934	16.98	**0.9435**	21.16
5	1.2267	0.9645	21.37	0.9602	21.72	0.9686	21.04	0.9800	20.11	**0.9345**	23.82

TABLE IV
COMPARISONS OF FORECASTING PERFORMANCE (MAPE)

Second	Persistence	ANN		DNN		ELM		SELM		GC-based SELM	
	MAPE	MAPE	Imp (%)	MAPE	Imp (%)	MAPE	Imp (%)	MAPE	Imp (%)	MAPE	Imp (%)
1	0.0616	0.0536	13.03	0.0572	7.16	0.0550	10.68	0.0557	9.65	**0.0519**	15.71
2	0.0703	0.0570	18.89	0.0595	15.35	0.0592	15.85	0.0599	14.84	**0.0560**	20.32
3	0.0732	0.0598	18.26	0.0607	17.14	0.0605	17.32	0.0607	17.05	**0.0574**	21.55
4	0.0760	0.0624	17.86	0.0623	17.99	0.0632	16.78	0.0632	16.82	**0.0603**	20.65
5	0.0786	0.0617	21.50	0.0611	22.27	0.0621	20.97	0.0625	20.53	**0.0595**	24.39

TABLE V
TIME CONSUMPTION ON SECOND-LEVEL DATASET

Algorithm	Training time (s)	Test time (s)
Persistence	-	0.21e-4
ANN	3.6825	0.0104
DNN	2.2690	0.0134
ELM	7.5625	0.1875
SELM	8.5331	0.2530
GC-based SELM	41.2950	0.2571

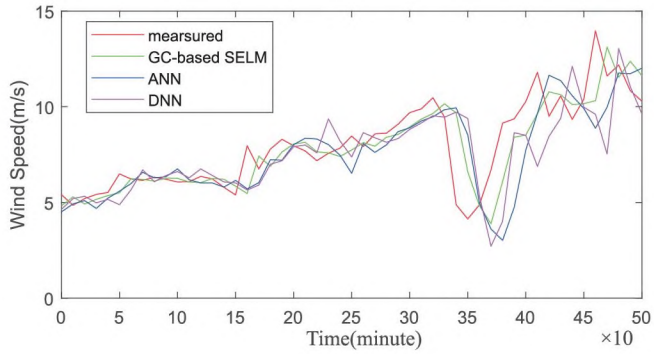


Fig. 2. 10 min-ahead forecasting.

the results in the tables show that all the models except the persistent model have degressive accuracy with the decrease of the prediction point on this dataset. Hence, the most accurate result we can obtain is to predict the next time point.

With respect to the time consumption, GC-based SELM takes more time to train data during the iteration process in computing similarity measure with generalized correntropy. In the test phase, both ELM and its variant algorithms take more time than ANN and DNN in our experiments, and the benchmark persistent method consumes less time. The details are shown in Table V.

E. Multistep Minute-Level Forecasting

Similarly, the parameters are set to achieve best performance. Specifically, the regularization coefficient C is equal to 2^{10} and another induced regularization parameter σ is set to 2^{-10}. The number of hidden layers and total hidden neurons in each layer are 3 and 100, respectively. The scale parameter γ and the shape parameter α induced by generalized correntropy are set to 0.05 and 3, respectively. The number of hidden neurons in ANN is chosen as 10.

The experiments mentioned above are on second-level data, and the following experiments are all on the basis of minute-level data. Considering that the most used forecasting points are about 10, 30, and 60 min later, respectively, the average of the next 10, 30, and 60 min forecasting data are set as the output in the following experiments, while there is a similar input date format of continuous ten data comparing with second-level forecasting. Here, with the purpose of clearly showing the computational results, only two widely used statistical models, i.e., ANN and DNN, are selected in the comparisons.

From Fig. 2–4, they demonstrate the curves of the forecasting wind speeds and the real wind speeds from the observation. The lines in Fig. 2 represent the 10 min-ahead predicted results of the wind speed from May 31, 2015 12:10 to May 31, 2015 20:30. And these lines in Figs. 3 and 4 represent the 30 min-ahead wind speed from May 31, 2015 03:40 to May 31, 2015 12:00 and 60 min-ahead wind speed from May 28, 2015 07:10 to May 28, 2015 15:30, respectively. From these figures, we can find that the result generated by GC-based SELM is closer to the actual data, compared with another two methods. It can achieve more accurate forecasting performance than ANN and DNN. The MAEs and MSEs of those three methods also verify this conclusion. Actually, the MAEs of our method on 10, 30, and 60 min are 0.7547, 1.0994, and 1.4336, respectively. They are

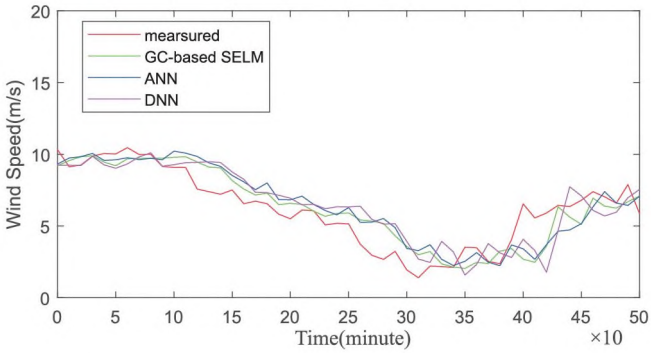


Fig. 3. 30 min-ahead forecasting.

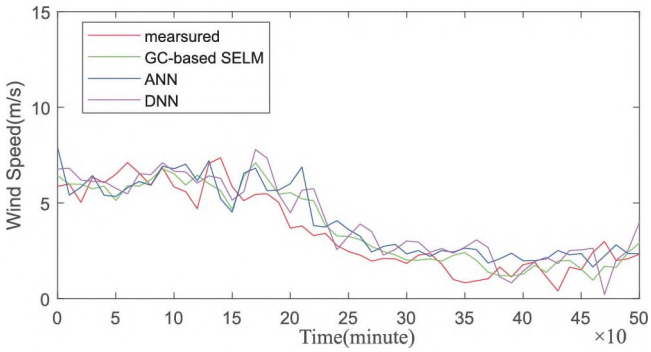


Fig. 4. 60 min-ahead forecasting.

obviously lower than the value 0.9531, 1.2328, 1.4520 of ANN, and the value 1.0947, 1.4428, 1.5397 of DNN. For the metric MSE, our method also achieves the lowest error among three methods, and it is 30% lower than ANN and 50% lower than DNN on 10 min ahead forecasting.

The time consumption on minute-level dataset is similar to that on second-level dataset. During the training phase, GC-based SELM spends 45.6227 s on average, and the average time consumptions on ANN and DNN are 0.5040 s and 2.8954 s, respectively. And in the test phase, compared with 0.0130 s and 0.0638 s spent by ANN and DNN respectively, GC-based SELM takes a little more time consumption 0.0600 s with higher forecasting accuracy. Considering that the calculation of the generalized correntropy is implemented in accordance with iterative mechanism, while the original β can be solved by (1) directly, the most time-consuming part belongs to the generalized correntropy cost function. On the other hand, the multilayer ELM architecture is also an important factor of the time consumption.

V. Conclusion

Motivated by deep-learning model, this paper has proposed a stacked ELM method using generalized correntropy on wind speed forecasting problem. Through the use of our proposed algorithm, the wind speed time series is modeled via replacing the cost function with generalized correntropy. In consideration of the robustness of generalized correntropy, this model can achieve better performance on observation data with outliers, compared with other methods.

From the experiments on multistep second-level and multistep minute-level wind speed forecasting tasks respectively, the performance of the method with generalized correntropy has been verified. Compared with some traditional and more recent models, including ANN, DNN, ELM, and SELM, our GC-based SELM achieves higher forecasting accuracy with a little more time consumption. In many industrial cases, well-trained models are used for production directly, while training time is less important. Then, the test time consumption in our method is acceptable with the same order of magnitude against other compared algorithms. The more accurate wind speed interval forecasting always means a significant improvement in wind farm operational control on robust optimization. To further improve the forecasting performance of this model, there is still much work to do along this direction, such as the optimization for the dimension reduction process [24] and the cost function of ELM-AE. Moreover, considering that the wind speed forecasting is similar to some other problems, such as stock forecasting, house price forecasting, we will also apply our method to these fields in the future work.

References

[1] Z. Zhang, Q. Zhou, and A. Kusiak, "Optimization of wind power and its variability with a computational intelligence approach," *IEEE Trans. Sustain. Energy*, vol. 5, no. 1, pp. 228–236, Jan. 2014.

[2] C. Wan, Z. Xu, P. Pinson, Z. Y. Dong, and K. P. Wong, "Probabilistic forecasting of wind power generation using extreme learning machine," *IEEE Trans. Power Syst.*, vol. 29, no. 3, pp. 1033–1044, May 2014.

[3] H. Long and Z. Zhang, "A two-echelon wind farm layout planning model," *IEEE Trans. Sustain. Energy*, vol. 6, no. 3, pp. 863–871, Jul. 2015.

[4] C. Wan, Y. Song, Z. Xu, G. Yang, and A. H. Nielsen, "Probabilistic wind power forecasting with hybrid artificial neural networks," *Electr. Power Compon. Syst.*, vol. 44, no. 15, pp. 1656–1668, 2016.

[5] J. L. Torres, A. García, M. D. Blas, and A. D. Francisco, "Forecast of hourly average wind speed with ARMA models in Navarre (Spain)," *Sol. Energy*, vol. 79, no. 1, pp. 65–77, Jul. 2005.

[6] M. Bilgili, B. Sahin, and A. Yasar, "Application of artificial neural networks for the wind speed prediction of target station using reference stations data," *Renew. Energy*, vol. 32, no. 14, pp. 2350–2360, Nov. 2007.

[7] A. Kusiak, H. Zheng, and Z. Song, "Short-term prediction of wind farm power: A data mining approach," *IEEE Trans. Energy Convers.*, vol. 24, no. 1, pp. 125–136, Mar. 2009.

[8] H. Liu, H. Tian, and Y. Li, "Four wind speed multi-step forecasting models using extreme learning machines and signal decomposing algorithms," *Energy Convers. Manage.*, vol. 100, pp. 16–22, Aug. 2015.

[9] Z. Zeng, Z. Li, D. Cheng, H. Zhang, K. Zhan, and Y. Yang, "Two-stream multi-rate recurrent neural network for video-based pedestrian re-identification," *IEEE Trans. Ind. Inf.*, vol. 14, no. 7, pp. 3179–3186, Jul. 2018.

[10] X. Chang, Y. L. Yu, Y. Yang, and E. P. Xing, "Semantic pooling for complex event analysis in untrimmed videos," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 8, pp. 1617–1632, Aug. 2017.

[11] X. Luo, D. Zhang, L. T. Yang, J. Liu, X. Chang, and H. Ning, "A kernel machine-based secure data sensing and fusion scheme in wireless sensor networks for the cyber-physical systems," *Future Gener. Comput. Syst.*, vol. 61, pp. 85–96, Aug. 2016.

[12] M. Dalto, J. Matusko, and M. Vasak, "Deep neural networks for ultra-short-term wind forecasting," in *Proc. IEEE Int. Conf. Ind. Technol.*, pp. 1657–1663, Jun. 2015.

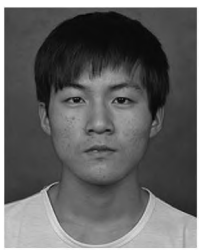
[13] H. Z. Wang, G. B. Wang, G. Q. Li, J. C. Peng, and Y. T. Liu, "Deep belief network based deterministic and probabilistic wind speed forecasting approach," *Appl. Energy*, vol. 182, pp. 80–93, Nov. 2016.

[14] C. Y. Zhang, C. L. P. Chen, M. Gan, and L. Chen, "Predictive deep Boltzmann machine for multiperiod wind speed forecasting," *IEEE Trans. Sustain. Energy*, vol. 6, no. 4, pp. 1416–1425, Oct. 2015.

[15] H. Zhou, G. B. Huang, Z. Lin, H. Wang, and Y. C. Soh, "Stacked extreme learning machines," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 2013–2025, Sep. 2015.
[16] W. Liu, P. P. Pokharel, and J. C. Príncipe, "Correntropy: Properties and applications in non-Gaussian signal processing," *IEEE Trans. Signal Process.*, vol. 55, no. 11, pp. 5286–5298, Nov. 2007.
[17] X. Luo *et al.*, "Towards enhancing stacked extreme learning machine with sparse autoencoder by correntropy," *J. Franklin Inst.*, vol. 355, no. 4, pp. 1945–1966, Mar. 2018.
[18] B. Chen, L. Xing, H. Zhao, N. Zheng, and J. C. Príncipe, "Generalized correntropy for robust adaptive filtering," *IEEE Trans. Signal Process.*, vol. 64, no. 13, pp. 3376–3387, Jul. 2016.
[19] L. L. C. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong, "Representational learning with ELMs for big data," *IEEE Intell. Syst.*, vol. 28, no. 6, pp. 31–34, Nov./Dec. 2013.
[20] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, pp. 489–501, Dec. 2006.
[21] L. Chen, H. Qu, and J. Zhao, "Generalized correntropy based deep learning in presence of non-Gaussian noises," *Neurocomputing*, vol. 278, pp. 41–50, Feb. 2018.
[22] Z. Song, Y. Jiang, and Z. Zhang, "Short-term wind speed forecasting with markov-switching model," *Appl. Energy*, vol. 130, pp. 103–112, Oct. 2014.
[23] G. Li and J. Shi, "On comparing three artificial neural networks for wind speed forecasting," *Appl. Energy*, vol. 87, no. 7, pp. 2313–2320, Jul. 2010.
[24] Z. Li, F. Nie, X. Chang, and Y. Yang, "Beyond trace ratio: Weighted harmonic mean of trace ratios for multiclass discriminant analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 10, pp. 2100–2110, Oct. 2017.

Xiong Luo (M'16) received the Ph.D. degree in computer applied technology from Central South University, Changsha, China, in 2004.

He is currently a Professor with the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, China. He has published extensively in his areas of interest in several journals, such as IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON HUMAN–MACHINE SYSTEMS, and IEEE ACCESS. His research interests include machine learning, cloud computing, and computational intelligence.



Jiankun Sun is currently working toward the Ph.D. degree at the University of Science and Technology Beijing, Beijing, China.

His research interests include machine learning and computational intelligence.

Long Wang (S'16–M'17) received the M.S. degree in computer science with distinction from University College London, London, U.K., in 2014, and the Ph.D. degree in systems engineering and engineering management from City University of Hong Kong, Hong Kong, in 2017.

He is currently an Associate Professor with the University of Science and Technology Beijing, Beijing, China. His research interests include machine learning, computational intelligence, and computer vision.

Dr. Wang is an Associate Editor for IEEE ACCESS and an Academic Editor for *PLOS ONE*.

Weiping Wang received the Ph.D. degree in physical electronics from the Beijing University of Posts and Telecommunications, China, in 2015.

She is currently an Associate Professor with the University of Science and Technology Beijing, Beijing, China. Her research interests include neural networks and computational intelligence.

Wenbing Zhao (S'99–M'02–SM'14) received the Ph.D. degree in electrical and computer engineering from the University of California, Santa Barbara, CA, USA, in 2002.

He is currently a Professor with the Department of Electrical Engineering and Computer Science, Cleveland State University, Cleveland, OH, USA. He has more than 120 academic publications. His research interests include dependable distributed systems and machine learning.

Dr. Zhao is the recipient of best paper awards at several international conferences.

Jinsong Wu (SM'11) received the Ph.D. degree in electrical and computer engineering from the Department of Electrical and Computer Engineering, Queen's University at Kingston, Kingston, ON, Canada, in 2006.

He is currently with the Department of Electrical Engineering, Universidad de Chile, Santiago, Chile. His research interests include green information and communication technologies, and smart grids.

Dr. Wu was the Founder and Founding Chair of IEEE Technical Committee on Green Communications and Computing. He is an Area Editor for the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING.

Jenq-Haur Wang received the Ph.D. degree in computer sciences from National Taiwan University, Taipei, Taiwan, in 2002.

He is currently an Associate Professor with the Department of Computer Science and Information Engineering, National Taipei University of Technology, Taipei, Taiwan. His research interests include data mining and network security.

Zijun Zhang (M'12) received the B.Eng. degree in systems engineering and engineering management from the Chinese University of Hong Kong, Hong Kong, in 2008, and the M.S. and Ph.D. degrees in industrial engineering from the University of Iowa, Iowa City, IA, USA, in 2012 and 2009, respectively.

He is currently an Associate Professor with the School of Data Science, City University of Hong Kong, Hong Kong. His research interests include data mining and computational intelligence with applications in wind energy, HVAC, and wastewater processing domains.