IET Generation, Transmission & Distribution

Special Section: Next Generation of Synchrophasor-based Power System Monitoring, Operation and Control

IET Journals
The Institution of Engineering and Technology

Incremental broad learning for real-time updating of data-driven power system dynamic security assessment models

ISSN 1751-8687
Received on 6th September 2019
Revised 23rd February 2020
Accepted on 5th March 2020
E-First on 7th May 2020
doi: 10.1049/iet-gtd.2019.1371
www.ietdl.org

Chao Ren[1], *Yan Xu*[2] ✉

[1]*Interdisciplinary Graduate School, Nanyang Technological University, Singapore, Singapore*
[2]*School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, Singapore*
✉ *E-mail: eeyanxu@gmail.com*

Abstract: This work aims to real-time update the data-driven dynamic security assessment model for accuracy maintenance and improvement. Based on the random vector functional link neural network, an incremental broad learning model is developed as an alternative and much faster approach for deep learning. Besides, the proposed method can significantly reduce the computation burden caused by abundant model parameters of layers and filters. Three real-time increment scenarios are achieved: enhancement hidden nodes, features, and training instances. Distinguished from existing methods, the proposed method can fast remodel in broad expansion without retraining the whole model, since it can only calculate the pseudoinverse of new state matrix without the computations of the whole output weights matrix. Numerical simulation results show that the proposed method can improve the accuracy step by step and update the model within a very short time at the online stage, verifying its real-time computational efficiency and accuracy improvement.

Nomenclature

$\boldsymbol{X}$	input data
$\boldsymbol{Z}_i$	ith group of features
$\boldsymbol{Z}^n$	all the features groups
$\boldsymbol{Z}_a^n$	additional enhancement hidden nodes corresponding to new training instances
$\boldsymbol{H}_j$	jth group of enhancement hidden nodes
$\boldsymbol{H}^m$	all the enhancement hidden nodes groups
$\boldsymbol{H}_{ex_m}$	additional enhancement hidden nodes corresponding to features $\boldsymbol{Z}_{n+1}$
$\boldsymbol{H}_{a_m}$	additional enhancement hidden nodes corresponding to features $\boldsymbol{Z}_a^n$
$\boldsymbol{Y}$	output matrix
$\mu(\cdot)$	activation function for features layers
$\varphi(\cdot)$	activation function for enhancement hidden nodes layers
m	number of enhancement hidden nodes
n	number of features
$\boldsymbol{X_a}$, $\boldsymbol{Y_a}$	adding new training instances
$\boldsymbol{W}_n^m$	weight with n features and m enhancement hidden nodes
$\boldsymbol{W}_n^{m+1}$	weight with an increment of enhancement hidden nodes
$\boldsymbol{W}_{n+1}^m$	weight with an increment of features
${}^a\boldsymbol{W}_n^m$	weight with an increment of new training instances
$\boldsymbol{W}_{e_i}$	weight from input data to the ith group of features
$\boldsymbol{W}_{h_j}$	weight from features to the jth group of enhancement hidden nodes
$\boldsymbol{\beta}_{e_i}$	biases from input data to the ith group of features
$\boldsymbol{\beta}_{h_j}$	biases from features to the jth group of enhancement hidden nodes
$\boldsymbol{R}_n^m$	state matrix with n features and m enhancement hidden nodes
$\boldsymbol{R}_n^{m+1}$	state matrix with an increment of enhancement hidden nodes
$\boldsymbol{R}_{n+1}^m$	state matrix with an increment of features
${}^a\boldsymbol{R}_n^m$	state matrix with an increment of new training instances
$\boldsymbol{V}$	transformation symbol for state matrix
$\boldsymbol{P}$	transformation symbol for state matrix
$\boldsymbol{K}$	transformation symbol for state matrix

1 Introduction

Intelligent system (IS) techniques have been regarded as a promising approach to achieve real-time dynamic security assessment (DSA). The IS strategy belongs to data-driven methods that aim to train a learning model (e.g. a classifier or predictor) from a DSA database generated from a power system operating state prediction and off-line simulation. Once trained, the model can be applied for online pre-fault DSA using real-time power system measurements as the inputs, which can provide much faster assessment speed and better generalisation capability [1–5]. In recent years, due to the rapid integration of intermittent renewable energy sources (such as wind and solar power), the operation state of the power system becomes highly uncertain and variable, which poses a great challenge to the power system's DSA. Traditionally, the model updating is realised by time-consuming re-training and parameter tuning [6, 7], which may not be fast enough to accommodate the variations of the power system operating state. As a consequence, the off-line prepared DSA database may not be sufficient to accommodate the wide-range changes of the power system operating state, and the DSA accuracy will be deteriorated if significant changes happen. Therefore, there is a pressing need to online update the DSA model periodically and timely [1–3].

The conventional DSA method is the time-domain (T-D) simulation that relies on iteratively solving a high-dimensional differential-algebraic equation set. Such high computation complexity makes it slow to satisfy the real-time assessment needs. Based on phasor measurement unit measurements, data-driven approaches have been developed for DSA based on some machine learning algorithms. Some examples are decision trees [8–11], support vector machine [12], artificial neural networks [13], extreme learning machines (ELM) [5], random vector functional link neural network (RVFL) [14, 15], generative adversarial network [16], Lyapunov exponent [17], shapelet classification [18], imbalanced learning technique [19], probabilistic reliability method [20], integrated online learning method [21], random forest [22], semi-supervised learning techniques [23], transfer learning algorithm [24], deep spatial-temporal data-driven approach [25], and online sequential learning method [26], which can carry out DSA control actions on time. As an emerging learning technology, ensemble learning also shows higher DSA performance [6]. For ensemble learning, all the learning units are assembled to solve the

same problem. The accuracy of the entire DSA is improved due to the compensation between different learning units. Among them, randomised learning algorithms, including ELM and RVFL, show their advantages due to their ability to model stochastic and fast learning. Stochastic modelling can improve the learning diversity of integrated learning, while rapid learning ability can help reduce the burden of training [27].

Owing to the nature of statistical learning, DSA errors may persist, resulting in loss or error in DSA results [27–29]. As described in [6], if a DSA error is still unavoidable, it is more sensible to detect potential errors and avoid using incredible DSA results. In [27], several manually-tuned key parameters were further optimised under the multi-objective optimisation programming framework to find the best balance between DSA accuracy and efficiency. Therefore, it is sensible to apply effective algorithms to further improve the performance of DSA, including the accuracy and credibility of the assessment results.

However, it is clear that these methods fully depend on the trained model, but do not consider real-time updating for trained DSA model, once they need to real-time updating or the IS cannot achieve the practical standards in advance, the DSA model may become ineffective and has to be updated via re-training the whole model with the new parameters setting or larger database.

To address the real-time DSA model updating problem, this study proposes an incremental broad learning method for accuracy maintenance and improvement of the data-driven DSA models [30–32]. The main contributions of the proposed method are as follows:

(i) The proposed method takes the merits of a broad learning structure to reduce the time-consuming training process caused by numerous model parameters, including layers setting and filters setting.
(ii) The proposed incremental broad learning method only needs to calculate the pseudoinverse of new state matrix without the computations of the whole output weights matrix. Therefore, it does not need re-training the whole model to continuously maintain and improve the online DSA accuracy.
(iii) Three different increment scenarios are achieved: enhancement hidden nodes, features, and new training instances. The snapshot performance can improve the DSA accuracy step by step within a very short time.

This paper is organised as follows: Section 2 presents the incremental broad learning network and its training mechanism in detail; the proposed online updating model is described in Section 3; Section 4 demonstrates the tested performance of the proposed approach on New England 10-machine 39-bus system; and Section 5 concludes the whole paper.

2 Incremental broad learning network

The broad learning algorithm is evolved from RVFL [31] and can achieve high performance without the need for deep learning structures that suffer from time-consuming training of massive parameters in multi-layers. Its structure comprises four parts (input $\boldsymbol{X}$, features $\boldsymbol{Z}^n$, enhancement hidden nodes $\boldsymbol{H}^m$, and output $\boldsymbol{Y}$). The input data $\boldsymbol{X}$ are firstly transformed into random features $\boldsymbol{Z}^n$ by feature mappings that are further randomly connected by non-linear activation functions to form the enhancement hidden nodes $\boldsymbol{H}^m$. Then, features $\boldsymbol{Z}^n$ together with enhancement hidden nodes $\boldsymbol{H}^m$ are connected to the output $\boldsymbol{Y}$ as (1), where the output weights $\boldsymbol{W}$ are determined by a linear regression method

$$\boldsymbol{Y} = \boldsymbol{R}_n^m \boldsymbol{W}_n^m = [\boldsymbol{Z}^n(\boldsymbol{X}) \big| \boldsymbol{H}^m(\boldsymbol{X})]\ \boldsymbol{W}_n^m \tag{1}$$

where (see (2)) , where $\mu(\cdot)$ and $\varphi(\cdot)$ are the activation functions for features layers and enhancement hidden nodes layers, respectively; $\boldsymbol{H}_j$ and $\boldsymbol{H}^m$ represent the jth group of enhancement hidden nodes and all the enhancement hidden nodes, respectively; $\boldsymbol{Z}_i$ and $\boldsymbol{Z}^n$ represent the ith group of features and all the features, respectively; state matrix is $\boldsymbol{R}_n^m = [\boldsymbol{Z}^n(\boldsymbol{X}) \big| \boldsymbol{H}^m(\boldsymbol{X})]$, m and n represent the number of enhancement hidden nodes and features; $\boldsymbol{W}_n^m$ represents weight with n features and m enhancement hidden nodes; $\boldsymbol{W}_{e_i}$ and $\boldsymbol{\beta}_{e_i}$ are the weight and biases from input data to the ith group of features, respectively; $\boldsymbol{W}_{h_j}$ and $\boldsymbol{\beta}_{h_j}$ are the weight and biases from features to the jth group of enhancement hidden nodes, respectively.

Incremental learning works for a new input arriving or an expansion of the features and enhancement hidden nodes. The main advantage of broad learning is its flexible structure, which allows it for broad learning to rapidly remodel in broad expansion without retraining the whole model [28]. With the dynamically increment of (i) enhancement hidden nodes and (ii) features and (iii) added online training instances (dashed arrows in Fig. 1*a*–*c*), the whole model can be updated in real-time and achieve desired accuracy performance. The principle of the three increment conditions and corresponding scenarios are described in the following sections.

2.1 Increment of enhancement hidden nodes

During online application, due to the unexpected changes in system operating state from the offline generated DSA database, the trained model may not maintain a satisfactory accuracy. One solution is to expand the network by adding the $(m + 1)$th group of enhancement hidden nodes, and the new state matrix $\boldsymbol{R}_n^{m+1}$ can be formulated as

$$\boldsymbol{R}_n^{m+1} = [\boldsymbol{R}_n^m \big| \boldsymbol{H}_{m+1}] \tag{3}$$

Then the new output weights $\boldsymbol{W}_n^{m+1}$ with the increment of enhancement nodes can be updated by

$$\boldsymbol{W}_n^{m+1} = \left(\boldsymbol{R}_n^{m+1}\right)^+ \boldsymbol{Y} = \begin{bmatrix} \boldsymbol{W}_n^m - V\boldsymbol{K}^{\mathrm{T}}Y \\ \boldsymbol{K}^{\mathrm{T}}Y \end{bmatrix} \tag{4}$$

where the pseudoinverse of $\boldsymbol{R}_n^{m+1}$ is

$$\left(\boldsymbol{R}_n^{m+1}\right)^+ = \begin{bmatrix} \boldsymbol{R}_n^m - V\boldsymbol{K}^{\mathrm{T}} \\ \boldsymbol{K}^{\mathrm{T}} \end{bmatrix} \tag{5}$$

where $\boldsymbol{V}$, $\boldsymbol{P}$, and $\boldsymbol{K}$ are transformation symbols for new state matrix $\boldsymbol{R}_n^{m+1}$ as

$$\begin{aligned} \boldsymbol{R}_n^m &= [\boldsymbol{Z}^n(\boldsymbol{X}) \big| \boldsymbol{H}^m(\boldsymbol{X})] \\ \boldsymbol{Z}_i &= \left[\mu\left(\boldsymbol{X}\boldsymbol{W}_{e_i} + \boldsymbol{\beta}_{e_i}\right)\right],\ i = 1, 2, \ldots, n \\ \boldsymbol{Z}^n &= [\boldsymbol{Z}_1, \ldots,\ \boldsymbol{Z}_n] \\ \boldsymbol{H}_j &= \left[\varphi\left(\boldsymbol{Z}^n\boldsymbol{W}_{h_j} + \boldsymbol{\beta}_{h_j}\right)\right],\ j = 1, 2, \ldots, m \\ \boldsymbol{H}^m &= [\boldsymbol{H}_1, \ldots,\ \boldsymbol{H}_m] \end{aligned} \tag{2}$$

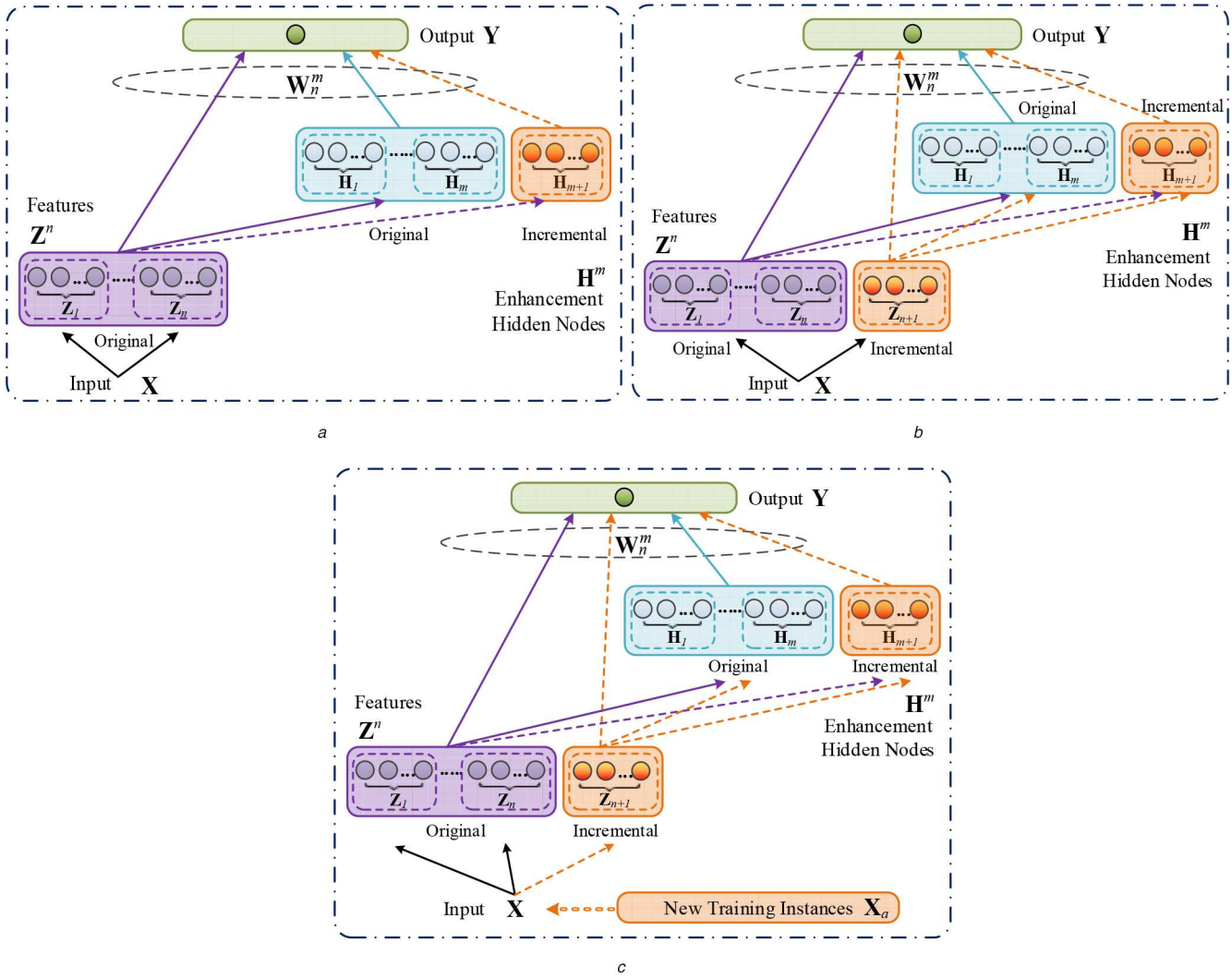

Fig. 1 *Different structure of the incremental broad learning for*
(a) Increment of enhancement hidden nodes, ***(b)*** Increment of features, ***(c)*** Increment of enhancement hidden nodes, features, and new training instances

Input: Inputs for instances $\mathbf{X}$, number of features n, number of the group of enhancement hidden nodes m.
Output: Weight $\mathbf{W}$.
begin
  for i =1 to n **do**:
    Randomly assign $\mathbf{W}_{e_i}$, $\boldsymbol{\beta}_{e_i}$.
    Calculate $\mathbf{Z}_i = [\mu(\mathbf{X}\mathbf{W}_{e_i} + \boldsymbol{\beta}_{e_i})]$.
  end for
  Obtain the features group $\mathbf{Z}^n = [\mathbf{Z}_1, \ldots, \mathbf{Z}_n]$.
  for j =1 to m **do**:
    Randomly as sign $\mathbf{W}_{h_j}$, $\boldsymbol{\beta}_{h_j}$.
    Calculate $\mathbf{H}_j = [\varphi(\mathbf{Z}^n\mathbf{W}_{h_j} + \boldsymbol{\beta}_{h_j})]$.
  end for
  Obtain the enhancement hidden nodes group $\mathbf{H}^m = [\mathbf{H}_1, \ldots, \mathbf{H}_m]$.
  Set $\mathbf{R}_n^m$ and calculate $(\mathbf{R}_n^m)^+$ by Eq. (1), (2).
  repeat
    # Increment of enhancement hidden nodes
    Randomly assign $\mathbf{W}_{h_{m+1}}$, $\boldsymbol{\beta}_{h_{m+1}}$.
    Calculate $\mathbf{H}_{m+1} = [\varphi(\mathbf{Z}^n\mathbf{W}_{h_{m+1}} + \boldsymbol{\beta}_{h_{m+1}})]$.
    Update $\mathbf{R}_n^{m+1}$ by Eq. (3).
    Calculate $(\mathbf{R}_n^{m+1})^+$ and $\mathbf{W}_n^{m+1}$ by Eq. (3), (4), (5), (6).
    $m = m + 1$.
  until Satisfy training error threshold
end

Fig. 2 *Algorithm 1: incremental broad learning: increment of enhancement hidden nodes*

$$\begin{aligned} \boldsymbol{V} &= (\boldsymbol{R}_n^m)^+ \boldsymbol{H}_{m+1} \\ \boldsymbol{P} &= \boldsymbol{H}_{m+1} - \boldsymbol{R}_n^m \boldsymbol{V} \\ \boldsymbol{K}^{\mathrm{T}} &= \begin{cases} \boldsymbol{P}^+ & \boldsymbol{P} \neq 0 \\ \left(1 + \boldsymbol{V}^{\mathrm{T}}\boldsymbol{V}\right)^{-1} \boldsymbol{V}^{\mathrm{T}} (\boldsymbol{R}_n^m)^+ & \boldsymbol{P} = 0 \end{cases} \end{aligned} \tag{6}$$

As can be seen, the algorithm based on such conditions only needs to calculate the new weights related to the $(m+1)$th group of enhancement hidden nodes, which can be highly computationally efficient. The complete computation process for the increment of the enhancement hidden nodes scenario is summarised in Algorithm 1 (see Fig. 2), and the structure is illustrated in Fig. 1*a*.

2.2 Increment of features

During online application, the originally selected features may not be sufficient to timely represent the online power system operating state. To address this condition, a traditional way is to retrain the model with the more feature inputs and reset the parameters for new network structures, which will cost more time to retrain, especially, for deep network structures. Based on the proposed method, the whole model can be efficiently constructed without retraining the whole network from the beginning. The network is expanded by adding the $(n+1)$th group of features to new state matrix $\boldsymbol{R}_{n+1}^m$ as

$$\boldsymbol{R}_{n+1}^m = \left[\boldsymbol{R}_n^m | \boldsymbol{Z}_{n+1} | \boldsymbol{H}_{ex_m}\right] \tag{7}$$

where $\boldsymbol{H}_{ex_m}$ is additional output of the enhancement nodes corresponding to the $(n+1)$th group of features $\boldsymbol{Z}_{n+1}$. Then the new output weights $\boldsymbol{W}_{n+1}^m$ with the increment of features and corresponding to enhancement hidden nodes can be updated by the following equation:

$$\boldsymbol{W}_{n+1}^m = (\boldsymbol{R}_{n+1}^m)^+ \boldsymbol{Y} = \begin{bmatrix} \boldsymbol{W}_n^m - V\boldsymbol{K}^{\mathrm{T}} Y \\ \boldsymbol{K}^T Y \end{bmatrix} \tag{8}$$

where the pseudoinverse of $\boldsymbol{R}_{n+1}^m$ is

$$(\boldsymbol{R}_{n+1}^m)^+ = \begin{bmatrix} \boldsymbol{R}_n^m - V\boldsymbol{K}^{\mathrm{T}} \\ \boldsymbol{K}^T \end{bmatrix} \tag{9}$$

where $\boldsymbol{V}$, $\boldsymbol{P}$, and $\boldsymbol{K}$ are transformation symbols for new state matrix $\boldsymbol{R}_{n+1}^m$ as

$$\begin{aligned} \boldsymbol{V} &= (\boldsymbol{R}_n^m)^+ [\boldsymbol{Z}_{n+1} | \boldsymbol{H}_{ex_m}] \\ \boldsymbol{P} &= [\boldsymbol{Z}_{n+1} | \boldsymbol{H}_{ex_m}] - \boldsymbol{R}_n^m \boldsymbol{V} \\ \boldsymbol{K}^{\mathrm{T}} &= \begin{cases} \boldsymbol{P}^+ & \boldsymbol{P} \neq 0 \\ \left(1 + \boldsymbol{V}^T \boldsymbol{V}\right)^{-1} \boldsymbol{V}^{\mathrm{T}} (\boldsymbol{R}_n^m)^+ & \boldsymbol{P} = 0 \end{cases} \end{aligned} \tag{10}$$

where the new weights related to the $(n+1)$th group of feature inputs is similar to the enhancement hidden nodes. As can be seen, the algorithm based on such condition only needs to calculate the new weights related to the $(n+1)$th group of feature inputs, thus resulting in highly computationally efficient. The complete computation process for the increment of feature inputs scenario is summarised in Algorithm 2 (see Fig. 3), and its structure is illustrated in Fig. 1*b*.

2.3 Increment of new training instances

During online application, more instances may be available from online information [6]. To utilise these newly generated instances, a traditional way is to attach the new instances to the original DSA database and retrain the model. However, this is certainly time-consuming. The proposed method can easily adapt to the online training data by only updating the corresponding weights. For this condition, we expand the network by adding new training instances $\{\boldsymbol{X_a}, \boldsymbol{Y_a}\}$ to new state matrix ${}^a\boldsymbol{R}_n^m$ as

$${}^a\boldsymbol{R}_n^m = \begin{bmatrix} \boldsymbol{R}_n^m \\ [\boldsymbol{Z}_a^n | \boldsymbol{H}_{a_m}] \end{bmatrix} \tag{11}$$

where $\boldsymbol{Z}_a^n$ represents the group of the incremental features updated by $\boldsymbol{X_a}$, and $\boldsymbol{H}_{a_m}$ represents the output of enhancement hidden nodes corresponding to $\boldsymbol{Z}_a^n$. Then the new output weights ${}^a\boldsymbol{W}_n^m$ with the increment of new instances can be updated by the following equation:

$${}^a\boldsymbol{W}_n^m = \boldsymbol{W}_n^m + \boldsymbol{K}\left(\boldsymbol{Y}_a^{\mathrm{T}} - [\boldsymbol{Z}_a^n | \boldsymbol{H}_{a_m}] \boldsymbol{W}_n^m\right) \tag{12}$$

where the pseudoinverse of ${}^a\boldsymbol{R}_n^m$ is

$$({}^a\boldsymbol{R}_n^m)^+ = \left[(\boldsymbol{R}_n^m)^+ - \boldsymbol{K}\boldsymbol{V}^{\mathrm{T}} | \boldsymbol{K}\right] \tag{13}$$

where $\boldsymbol{V}$, $\boldsymbol{P}$, and $\boldsymbol{K}$ are transformation symbols for new state matrix ${}^a\boldsymbol{R}_n^m$ as

$$\begin{aligned} \boldsymbol{V} &= \left((\boldsymbol{R}_n^m)^{\mathrm{T}}\right)^+ [\boldsymbol{Z}_a^n | \boldsymbol{H}_{a_m}]^{\mathrm{T}} \\ \boldsymbol{P} &= [\boldsymbol{Z}_a^n | \boldsymbol{H}_{a_m}]^{\mathrm{T}} - (\boldsymbol{R}_n^m)^{\mathrm{T}} \boldsymbol{V} \\ \boldsymbol{K}^{\mathrm{T}} &= \begin{cases} \boldsymbol{P}^+ & \boldsymbol{P} \neq 0 \\ \left(1 + \boldsymbol{V}^{\mathrm{T}} \boldsymbol{V}\right)^{-1} \boldsymbol{V}^{\mathrm{T}} \left((\boldsymbol{R}_n^m)^{\mathrm{T}}\right)^+ & \boldsymbol{P} = 0 \end{cases} \end{aligned} \tag{14}$$

To summarise, when the current model needs to be online updated, the proposed method only needs to calculate the pseudoinverse of the new state matrix without re-training. Thus, it can achieve real-time DSA updating. A complete computation process for the three increment scenarios is summarised in Algorithm 3 (see Fig. 4), and its structure is illustrated in Fig. 1*c*.

3 Implementation of proposed IS

The whole framework of the proposed method illustrated in Fig. 5 can be divided into three stages. At the offline training stage, the historical DSA database is utilised to train a DSA model via a broad learning algorithm, which can be a classifier or predictor. The feature inputs to the model are P/Q power generation, load demand, and bus voltage magnitudes. The *RELIEF-F* algorithm is used to select the critical features which are mostly related to the stability status. At the online application stage, the real-time power system measurement will be sent to the trained DSA model, then the system can obtain the final DSA result and activate the preventive control (i.e. generation rescheduling) if necessary. If the accuracy monitoring judges that it does not meet the actual accuracy requirements, the trained DSA model needs to online update via updating model. At the online updating stage, the current DSA model can be continuously updated by an increment of features or enhancement hidden nodes. In the meantime, all the online measurements can generate new training instances which can also be used to online update the current DSA model for performance maintenance or improvement.

4 Simulation results

The proposed model is tested on the New England 39-bus system (in Fig. 6), which represents a standard benchmark power system

Input: Inputs for instances $\mathbf{X}$, number of features n, number of the group of enhancement hidden nodes m.
Output: Weight $\mathbf{W}$.
begin
  for i =1 to n **do**:
    Randomly assign $\mathbf{W}_{e_i}$, $\boldsymbol{\beta}_{e_i}$.
    Calculate $\mathbf{Z}_i = [\mu(\mathbf{X}\mathbf{W}_{e_i} + \boldsymbol{\beta}_{e_i})]$.
  end for
  Obtain the features group $\mathbf{Z}^n = [\mathbf{Z}_1, \ldots, \mathbf{Z}_n]$.
  for j =1 to m **do**:
    Randomly as sign $\mathbf{W}_{h_j}$, $\boldsymbol{\beta}_{h_j}$.
    Calculate $\mathbf{H}_j = [\varphi(\mathbf{Z}^n\mathbf{W}_{h_j} + \boldsymbol{\beta}_{h_j})]$.
  end for
  Obtain the enhancement hidden nodes group $\mathbf{H}^m = [\mathbf{H}_1, \ldots, \mathbf{H}_m]$.
  Set $\mathbf{R}_n^m$ and calculate $(\mathbf{R}_n^m)^+$ by Eq. (1), (2).
  repeat
    # Increment of enhancement features
    Randomly assign $\mathbf{W}_{e_{n+1}}$, $\boldsymbol{\beta}_{e_{n+1}}$.
    Calculate $\mathbf{Z}_{n+1} = [\mu(\mathbf{X}\mathbf{W}_{e_{n+1}} + \boldsymbol{\beta}_{e_{n+1}})]$.
    Randomly assign $\mathbf{W}_{ex_i}$, $\boldsymbol{\beta}_{ex_i}$, i=1, …, m.
    Calculate $\mathbf{H}_{ex_m}$ =
      $[\varphi(\mathbf{Z}_{n+1}\mathbf{W}_{ex_i} + \boldsymbol{\beta}_{ex_i}), .. \varphi(\mathbf{Z}_{n+1}\mathbf{W}_{ex_m} + \boldsymbol{\beta}_{ex_m})]$.
    Update $\mathbf{R}_{n+1}^m$ by Eq. (7).
    Calculate $(\mathbf{R}_{n+1}^m)^+$ and $\mathbf{W}_{n+1}^m$ by Eq. (7), (8), (9), (10).
    $n = n + 1$.
  until Satisfy training error threshold
end

Fig. 3 *Algorithm 2: incremental broad learning: increment of features*

for stability analysis [33]. The numerical simulation is conducted on a 64-bit computer with an Intel Core i7 CPU of 3.4-GHz and 32-GB RAM. The T-D simulation is implemented in the transient security assessment tool (TSAT). The proposed algorithms are implemented in the MATLAB platform.

Input: Inputs for instances $\mathbf{X}$, number of features n, number of the group of enhancement hidden nodes m.
Output: Weight $\mathbf{W}$.
begin
  for i =1 to n **do**:
    Randomly assign $\mathbf{W}_{e_i}$, $\boldsymbol{\beta}_{e_i}$.
    Calculate $\mathbf{Z}_i = [\mu(\mathbf{X}\mathbf{W}_{e_i} + \boldsymbol{\beta}_{e_i})]$.
  end for
  Obtain the features group $\mathbf{Z}^n = [\mathbf{Z}_1, \ldots, \mathbf{Z}_n]$.
  for j =1 to m **do**:
    Randomly as sign $\mathbf{W}_{h_j}$, $\boldsymbol{\beta}_{h_j}$.
    Calculate $\mathbf{H}_j = [\varphi(\mathbf{Z}^n\mathbf{W}_{h_j} + \boldsymbol{\beta}_{h_j})]$.
  end for
  Obtain the enhancement hidden nodes group $\mathbf{H}^m = [\mathbf{H}_1, \ldots, \mathbf{H}_m]$.
  Set $\mathbf{R}_n^m$ and calculate $(\mathbf{R}_n^m)^+$ by Eq. (1), (2).
  repeat
    if *Increment of enhancement hidden nodes* **then**
      Randomly assign $\mathbf{W}_{h_{m+1}}$, $\boldsymbol{\beta}_{h_{m+1}}$.
      Calculate $\mathbf{H}_{m+1} = [\varphi(\mathbf{Z}^n\mathbf{W}_{h_{m+1}} + \boldsymbol{\beta}_{h_{m+1}})]$.
      Update $\mathbf{R}_n^{m+1}$ by Eq. (3).
      Calculate $(\mathbf{R}_n^{m+1})^+$ and $\mathbf{W}_n^{m+1}$ by Eq. (3), (4), (5), (6).
      $m = m + 1$.
    else if *Increment of features* **then**
      Randomly assign $\mathbf{W}_{e_{n+1}}$, $\boldsymbol{\beta}_{e_{n+1}}$.
      Calculate $\mathbf{Z}_{n+1} = [\mu(\mathbf{X}\mathbf{W}_{e_{n+1}} + \boldsymbol{\beta}_{e_{n+1}})]$.
      Randomly assign $\mathbf{W}_{ex_i}$, $\boldsymbol{\beta}_{ex_i}$, i=1, …, m.
      Calculate $\mathbf{H}_{ex_m} =$
        $[\varphi(\mathbf{Z}_{n+1}\mathbf{W}_{ex_i} + \boldsymbol{\beta}_{ex_i}), .. \varphi(\mathbf{Z}_{n+1}\mathbf{W}_{ex_m} + \boldsymbol{\beta}_{ex_m})]$.
      Update $\mathbf{R}_{n+1}^m$ by Eq. (7).
      Calculate $(\mathbf{R}_{n+1}^m)^+$ and $\mathbf{W}_{n+1}^m$ by Eq. (7), (8), (9), (10).
      $n = n + 1$.
    else if *Increment of new training instances* **then**
      Add new instances $\{\mathbf{X}_a, \mathbf{Y}_a\}$.
      Calculate $\mathbf{Z}_a^n = [\mu(\mathbf{X}_a\mathbf{W}_{e_1} + \boldsymbol{\beta}_{e_1}), \ldots, \mu(\mathbf{X}_a\mathbf{W}_{e_n} + \boldsymbol{\beta}_{e_n})]$.
      Calculate $\mathbf{H}_{a_m} = [\varphi(\mathbf{Z}_a^n\mathbf{W}_{h_1} + \boldsymbol{\beta}_{h_1}), .. \varphi(\mathbf{Z}_a^n\mathbf{W}_{h_m} + \boldsymbol{\beta}_{h_m})]$.
      Update $^a\mathbf{R}_n^m$ by Eq. (11).
      Calculate $(^a\mathbf{R}_n^m)^+$ and $(^a\mathbf{W}_n^m)^+$ by Eq. (11), (12), (13), (14).
    end if
  until Satisfy training error threshold
end

Fig. 4 *Algorithm 3: incremental broad learning: increment of enhancement hidden nodes, features, and new training instances*

4.1 Database generation

Simulations are performed on a New England 10-machine 39-bus system to test the proposed method. The operating instances are generated via Monte–Carlo method [5], which randomly samples wind and load within its predetermined range. Specifically, given the forecasted load demand level for each bus, Monte–Carlo simulation is run to sample uncertain power variations. Then, for each uncertain power demand scenario, the power output of the generators is determined by optimal power flow calculation. The contingencies considered in the study are the three-phase faults on bus 3 with inter-area corridor trip (i.e. trip line 3-4) and cleared 0.25 s after their occurrences. In doing this, a large number of operating points will be obtained. Then, given the contingency set, time-domain simulation is performed to evaluate the dynamic security label for each operating point. The security conditions subject to the selected contingency are obtained by running a time-domain simulation using the TAST package. Finally, 10,000 operating instances with security conditions were obtained. In the study, 8000 are randomly sampled as the training dataset and the remaining 2000 serve as testing dataset.

4.2 Feature selection

The power system has high dimensions, which means that these features can represent its operation state. However, the core objective of the power system is stability, so only important features are useful inputs. The *RELIEF-F* algorithm [34] is a feature weight algorithm, which assigns features with different weights according to the correlation of each feature and label, and can also remove features with weights less than a specific threshold. The higher the weight of features, the stronger the classification ability of features, and vice versa, the weaker the classification ability of features. Therefore, the features with high positive weight are selected as the key features. The total number of features is 240 including power generation/load demand, bus voltage magnitude, branch flow etc.

4.3 Testing results

The proposed method is tested by applying the testing instances of the New England 10-machine 39-bus test system to show the online updating performance. The selected basic case performance and three increment conditions and corresponding scenarios for online testing are listed in Table 1. From left to right, the columns of Table 1 are the method, the number of new training instances, the number of features, the number of enhancement hidden nodes, testing accuracy, the accumulative training time, and accumulative

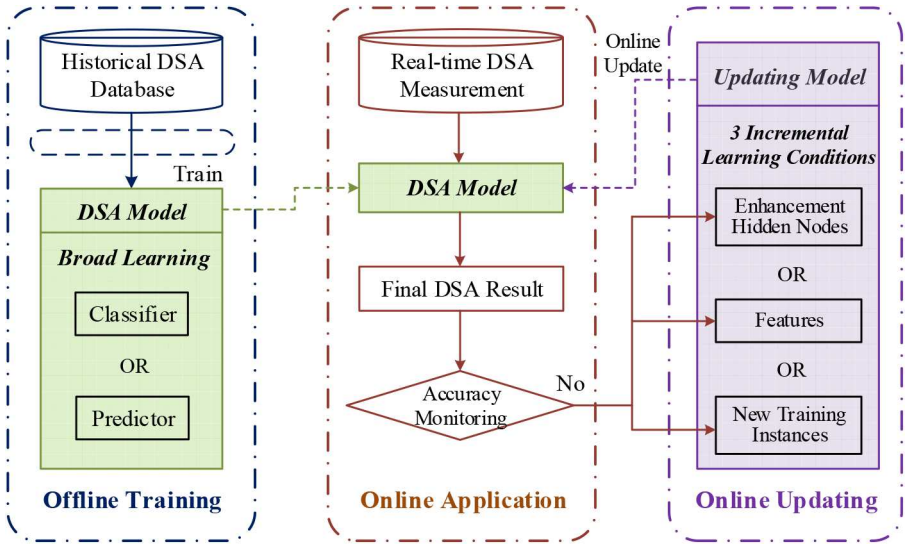

Fig. 5 *Framework of the proposed method*

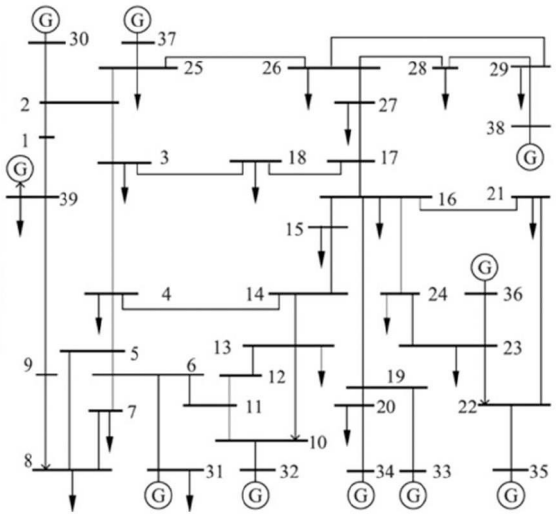

Fig. 6 *New England 10-machine 39-bus system*

testing time, respectively. In Tables 2–4, there are columns also for each additional training time and each additional testing time. Among them, accumulative training time and accumulative testing time represent the whole training time and testing time, respectively, and each additional training time and each additional testing time represent the training time and testing time at the current step, respectively. From Table 1, it is clear that the testing accuracy of the three different increment scenarios (see Algorithms 1–3 (Figs. 2–4)) all can be maintained and gradually improved (0.35–0.45% higher) compared to the basic case. More specifically, the proposed method can achieve the high DSA accuracy effectively and efficiently within a very short time after four steps increment steps for three different increment conditions, which can save much time in the training process and further verifies that the proposed methods are suitable for DSA online updating without re-training the whole model. The detail snapshot performance of three different increment conditions and corresponding scenarios are displayed in Tables 2–4.

The snapshot results of increment of enhancement hidden nodes are shown in Table 2: for the basic case, 8000 training examples are utilised to train DSA model, all the 240 features and 200 enhancement hidden nodes are employed, resulting in a 97.65%

Table 1 Classification accuracy and computational efficiency performance on New England test system by three different incremental broad learning algorithms

Method	Number of training instances	Number of features	Number of enhancement nodes	Testing accuracy (%)	Accumulative training times (s)	Accumulative testing times (s)
basic case	8000	240	400	**98.15**	**0.3212**	0.0474
increment of enhancement nodes (Algorithm 1 (Fig. 2))	8000	240	$200 \underset{50\times4}{\longrightarrow} 400$	**98.50**	**0.5806**	0.0817
increment of features (Algorithm 2 (Fig. 3))	8000	$80 \underset{40\times4}{\longrightarrow} 240$	$200 \underset{(20+30)\times4}{\longrightarrow} 400$	**98.55**	**0.7587**	0.0836
increment of input instances & feature nodes & enhancement nodes (Algorithm 3 (Fig. 4))	$2000 \underset{1500\times4}{\longrightarrow} 8000$	$80 \underset{40\times4}{\longrightarrow} 240$	$200 \underset{(20+30)\times4}{\longrightarrow} 400$	**98.60**	**0.4035**	0.0673

The bold values are to highlight the performance of accuracy and computational efficiency.

Table 2 Snapshot performance of classification accuracy and computational efficiency on New England test system by an increment of enhancement hidden nodes

Number of training instances	Number of features	Number of enhancement nodes	Testing accuracy (%)	Each additional training times (s)	Accumulative training times (s)	Each additional testing times (s)	Accumulative testing times (s)
8000 (base case)	240	200	97.65	0.2949	0.2949	0.0471	0.0471
8000	240	200→250	**97.95**	**0.0654**	0.3603	0.0094	0.0565
8000	240	250→300	**98.05**	**0.0671**	0.4274	0.0073	0.0638
8000	240	300→350	**98.35**	**0.0737**	0.5011	0.0071	0.0709
8000	240	350→400	**98.50**	**0.0795**	0.5806	0.0108	0.0817

The bold values are to highlight the performance of accuracy and computational efficiency.

Table 3 Snapshot performance of classification accuracy and computational efficiency on the New England test system by an increment of features

Number of training instances	Number of features	Number of enhancement nodes	Testing accuracy (%)	Each additional training times (s)	Accumulative training times (s)	Each additional testing times (s)	Accumulative testing times (s)
8000 (base case)	80	200	97.50	0.2774	0.2774	0.0416	0.0416
8000	80→120	200→250	**97.95**	**0.1120**	0.3894	0.0116	0.0532
8000	120→160	250→300	**98.30**	**0.1151**	0.5045	0.0103	0.0635
8000	160→200	300→350	**98.45**	**0.1251**	0.6296	0.0101	0.0736
8000	200→240	350→400	**98.55**	**0.1291**	0.7587	0.0100	0.0836

The bold values are to highlight the performance of accuracy and computational efficiency.

Table 4 Snapshot performance of classification accuracy and computational efficiency on the New England test system by an increment of enhancement hidden nodes, features, and new training instances

Number of training instances	Number of features	Number of enhancement nodes	Testing accuracy (%)	Each additional training times (s)	Accumulative training times (s)	Each additional testing times (s)	Accumulative testing times (s)
2000 (base case)	80	200	97.00	0.1029	0.1029	0.0466	0.0466
2000→3500	80→120	200→250	**97.70**	**0.0626**	0.1655	0.0053	0.0519
3500→5000	120→160	250→300	**98.25**	**0.0618**	0.2273	0.0054	0.0573
5000→6500	160→200	300→350	**98.45**	**0.0856**	0.3129	0.0048	0.0621
6500→8000	200→240	350→400	**98.60**	**0.0906**	0.4035	0.0052	0.0673

The bold values are to highlight the performance of accuracy and computational efficiency.

testing accuracy. Through four steps incremental schemes, the testing accuracy can be improved continuously, increased by 0.85% to 98.50%. In addition, it can be observed that for each incremental time, the computational time is very small, ranging from 0.0654 to 0.0795 s (far less than the basic case training time 0.2949 s), which can indicate that the model does not need to retain and is suitable for DSA online updating increment of enhancement hidden nodes. To use additional 200 enhancement hidden nodes to update the original DSA model, the total computation time of the proposed method is only 0.5806 s. Besides, the testing times of this increment scenario also meet the practical requirement.

Then, the snapshot results of increment of features are shown in Table 3: for the basic case, 8000 examples are applied to train DSA model, 80 features and 200 enhancement hidden nodes are utilised, leading to a 97.50% testing accuracy. With the four increment steps (add additional 40 features and 50 enhancement hidden nodes each step), the testing accuracy can continuously increase, up to 98.55% in the end. Besides, it is clear that for each time of increment, the needed computation time is also very minor, which ranges from 0.1120 to 0.1291 s. To update such a model with 160 additional features, and 200 enhancement hidden nodes, the accumulative computation time of the proposed method is just 0.7587 s, which can prove compliance and is also suitable for online DSA updating.

The snapshot performance of all three increments (including, enhancement hidden nodes, feature inputs, and new training instances) are given in Table 4: for the base case, 2000 instances are used to train the original DSA model, 80 features and 200 enhancement hidden nodes are used, leading to a 97.00% testing accuracy. With the four incremental steps (add additional 1500 training instances, 40 features, and 50 enhancement hidden nodes each step), the testing accuracy can continuously increase, up to 98.60% in the end. Moreover, it is important to note that for each time of increment, the needed computation time is very minor, which ranges from 0.0618 to 0.0906 s. To summarise, to update the model with 6000 additional training instances, 160 additional features, and 200 enhancement hidden nodes, the total computation time of the proposed method is just 0.4035 s, which allows its online implementation.

5 Conclusion

In this study, an incremental broad learning method is proposed to real-time update the trained DSA model for accuracy maintenance and improvement. Instead of retraining the whole model, the proposed method only needs to calculate the pseudoinverse of the new state matrix without the computations of the whole output weights matrix. Even though adding 6000 additional training instances, 160 additional features, and 200 enhancement hidden nodes, the accumulative training time of the proposed method is still minor as 0.4035 s, which can fully allow real-time model updating. The snapshot performances of three different increment scenarios are all shown that the proposed method can effectively improve the assessment accuracy (0.85–1.60% higher). Meanwhile, each step additional testing times (0.0519–0.0673 s) meet the practical requirements and can ensure that the trained model can make the final decision as soon as possible. To the best of our knowledge, similar works have not been reported in the literature, and the proposed method can be a very promising method to solve similar problems in power engineering.

6 Acknowledgment

This work was supported in part by the Ministry of Education (MOE), Republic of Singapore, under grant AcRF TIER 1 2019-T1-001-069 (RG75/19). Y.X. work was supported by the Nanyang Assistant Professorship from Nanyang Technological University, Singapore. C.R. work was supported by the Interdisciplinary Graduate Programme Scholarship from Nanyang Technological University, Singapore.

7 References

[1] Kundur, P., Paserba, J., Ajjarapu, V., *et al.*: 'Definition and classification of power system stability IEEE/CIGRE joint task force on stability terms and definitions', *IEEE Trans. Power Syst.*, 2004, **19**, (3), pp. 1387–1401

[2] Dong, Z.Y., Xu, Y., Zhang, P., *et al.*: 'Using IS to assess an electric power system's real-time stability', *IEEE Intell. Syst.*, 2013, **28**, (4), pp. 60–66

[3] Zhang, P., Li, F., Bhatt, N.: 'Next-generation monitoring, analysis, and control for the future smart control center', *IEEE Trans. Smart Grid*, 2010, **1**, (2), pp. 186–192

[4] Xu, Y., Guan, L., Dong, Z.Y., *et al.*: 'Transient stability assessment on China southern power grid system with an improved pattern discovery-based method'. 2010 Int. Conf. on Power System Technology (POWERCON), Hangzhou, People's Republic of China, 2010, pp. 1–6

[5] Xu, Y., Dong, Z., Meng, K., *et al.*: 'Real-time transient stability assessment model using extreme learning machine', *IET Gener. Transm. Distrib.*, 2011, **5**, (3), pp. 314–322

[6] Xu, Y., Dong, Z.Y., Zhao, J.H., *et al.*: 'A reliable intelligent system for real-time dynamic security assessment of power systems', *IEEE Trans. Power Syst.*, 2012, **27**, (3), pp. 1253–1263

[7] Ren, C., Xu, Y., Zhang, Y., *et al.*: 'A multiple randomized learning based ensemble model for power system dynamic security assessment'. 2018 IEEE Power & Energy Society General Meeting (PESGM), Portland, OR, USA, 2018, pp. 1–5

[8] Kamwa, I., Samantaray, S., Joos, G.: 'Catastrophe predictors from ensemble decision-tree learning of wide-area severity indices', *IEEE Trans. Smart Grid*, 2010, **1**, (2), pp. 144–158

[9] Wehenkel, L., Pavella, M., Euxibie, E., *et al.*: 'Decision tree based transient stability method a case study', *IEEE Trans. Power Syst.*, 1994, **9**, (1), pp. 459–469

[10] Guo, T., Milanovic, J.V.: 'The effect of quality and availability of measurement signals on accuracy of on-line prediction of transient stability using decision tree method'. 4th IEEE/PES Innovative Smart Grid Technologies Europe (ISGT EUROPE), Lyngby, Denmark, 2013, pp. 1–5

[11] He, M., Vittal, V., Zhang, J.: 'Online dynamic security assessment with missing PMU measurements: a data mining approach', *IEEE Trans. Power Syst.*, 2013, **28**, (2), pp. 1969–1977

[12] Rajapakse, A.D., Gomez, F., Nanayakkara, K., *et al.*: 'Rotor angle instability prediction using post-disturbance voltage trajectories', *IEEE Trans. Power Syst.*, 2010, **25**, (2), pp. 947–956

[13] Zhou, Z.-H., Jiang, Y.: 'Medical diagnosis with C4. 5 rule preceded by artificial neural network ensemble', *IEEE Trans. Inf. Technol. Biomed.*, 2003, **7**, (1), pp. 37–42

[14] Zhang, Y., Xu, Y., Dong, Z.Y.: 'Robust ensemble data analytics for incomplete PMU measurements-based power system stability assessment', *IEEE Trans. Power Syst.*, 2018, **33**, (1), pp. 1124–1126

[15] Zhang, Y., Xu, Y., Dong, Z.: 'A robust ensemble model for PMU-based power system on-line dynamic security assessment considering incomplete measurements', *IET Gener. Transm. Distrib.*, 2017, **11**, (18), pp. 4484–4491

[16] Ren, C., Xu, Y.: 'A fully data-driven method based on generative adversarial networks for power system dynamic security assessment with missing data', *IEEE Trans. Power Syst.*, 2019, **34**, (6), pp. 5044–5052

[17] Dasgupta, S., Paramasivam, M., Vaidya, U., *et al.*: 'Real-time monitoring of short-term voltage stability using PMU data', *IEEE Trans. Power Syst.*, 2013, **28**, (4), pp. 3702–3711

[18] Zhu, L., Lu, C., Sun, Y.: 'Time series shapelet classification based online short-term voltage stability assessment', *IEEE Trans. Power Syst.*, 2016, **31**, (2), pp. 1430–1439

[19] Zhu, L., Lu, C., Dong, Z.Y., *et al.*: 'Imbalance learning machine based power system short-term voltage stability assessment', *IEEE Trans. Ind. Inf.*, 2017, **13**, (5), pp. 2533–2543

[20] Wang, Y., Vittal, V., Abdi-Khorsand, M., *et al.*: 'Probabilistic reliability evaluation including adequacy and dynamic security assessment', *IEEE Trans. Power Syst.*, 2020, **35**, (1), pp. 551–559
[21] Li, H., Diao, R., Zhang, X., *et al.*: 'An integrated online dynamic security assessment system for improved situational awareness and economic operation', *IEEE Access*, 2019, **7**, (1), pp. 162571–162582
[22] Liu, C., Tang, F., Leth Bak, C.: 'An accurate online dynamic security assessment scheme based on random forest', *Energies*, 2018, **11**, (7), p. 1914
[23] Liu, R., Verbič, G., Ma, J.: 'A new dynamic security assessment framework based on semi-supervised learning and data editing', *Electr. Power Syst. Res.*, 2019, **17**, (2), pp. 221–229
[24] Ren, C., Xu, Y.: 'Transfer learning-based power system online dynamic security assessment: using one model to assess many unlearned faults', *IEEE Trans. Power Syst.*, 2020, **35**, (1), pp. 821–824
[25] Huang, T., Guo, Q., Sun, H., *et al.*: 'A deep spatial-temporal data-driven approach considering microclimates for power system security assessment', *Appl. Energy*, 2019, **237**, pp. 36–48
[26] Zhang, R., Xu, Y.: 'Data-driven dynamic security assessment and control of power systems: an online sequential learning method', *J. Energy Eng.*, 2019, **145**, (5), p. 04019019
[27] Zhang, Y., Xu, Y., Dong, Z., *et al.*: 'Intelligent early-warning of power system dynamic insecurity risk: towards optimal accuracy-earliness tradeoff', *IEEE Trans. Ind. Inf.*, 2017, **13**, (5), pp. 2544–2554
[28] Sun, K., Likhate, S., Vittal, V., *et al.*: 'An online dynamic security assessment scheme using phasor measurements and decision trees', *IEEE Trans. Power Syst.*, 2007, **22**, (4), pp. 1935–1943
[29] Zhao, J.H., Dong, Z.Y., Zhang, P.: 'Mining complex power networks for blackout prevention'. Proc. 13th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining, San Jose, California, USA, 2007, pp. 986–994
[30] Chen, C.P., Liu, Z.: 'Broad learning system: an effective and efficient incremental learning system without the need for deep architecture', *IEEE Trans. Neural Netw. Learn. Syst.*, 2018, **29**, (1), pp. 10–24
[31] Chen, C.P., Liu, Z.: 'Broad learning system: structural extensions on single-layer and multi-layer neural networks'. 2017 Int. Conf. on Security, Pattern Analysis, and Cybernetics (SPAC), Shenzhen, People's Republic of China, 2017
[32] Ross, D.A., Lim, J., Lin, R.-S., *et al.*: 'Incremental learning for robust visual tracking', *Int. J. Comput. Vis.*, 2008, **77**, (1–3), pp. 125–141
[33] Meng, K., Dong, Z., Wong, K., *et al.*: 'Speed-up the computing efficiency of power system simulator for engineering-based power system transient stability simulations', *IET Gener. Transm. Distrib.*, 2010, **4**, (5), pp. 652–661
[34] Sikonja, M., Kononenko, I.: 'Theoretical and empirical analysis of relief and relief', *Mach. Learn.*, 2003, **53**, pp. 23–69