Electronics and Electrical Engineering

AIMS Electronics and Electrical Engineering, 4(3): 287–302.
DOI: 10.3934/ElectrEng.2020.3.287
Received: 30 April 2020
Accepted: 30 June 2020
Published: 13 July 2020

http://www.aimspress.com/journal/ElectrEng

Research article

Developing robust nonlinear models through bootstrap aggregated deep belief networks

Changhao Zhu* and Jie Zhang*

School of Engineering, Merz Court, Newcastle University, Newcastle upon Tyne NE1 7RU, UK

*** Correspondence:** Email: zhu5@newcastle.ac.uk; jie.zhang@newcastle.ac.uk.

Abstract: Deep belief network (DBN) has recently emerged as a powerful tool in building nonlinear data driven models. However, a single DBN model can still lack reliability especially when the amount of data available for modelling is limited. This paper proposes a bootstrap aggregated deep belief network (BAGDBN) to improve model reliability and robustness. In the proposed method, bootstrap re-sampling with replacement is applied to the original modelling data to generate multiple replications. A DBN model is developed on each replication of the original modelling data. These individual DBN models are then combined to form a BAGDBN model. The proposed method is demonstrated on two application examples, modelling of a conic water tank and inferential estimation of polymer melt index in an industrial polypropylene polymerization process. The application results demonstrate that the proposed BAGDBN models can give more reliable estimation and prediction than single DBN models.

Keywords: machine learning; bootstrap aggregated deep belief network; robustness; soft sensor; polypropylene polymerization

1. Introduction

With the increasing customer demands and government regulations, modern industrial facilities face more stringent requirements on produce quality, production efficiency, and emission reduction. Advanced process control is adopted to reduce the utility cost and increase the industrial product yields. The strategy of process control is to monitor and control the production process effectively using accurate measurements of key process variables and advanced control techniques like model

based predictive control. Due to the problems of the high cost of some instruments and delay of measurement on key quality variables, many important product quality variables cannot be measured online hindering their real time control. For example, the polymer melt index (MI) is not easy to be measured online in polypropylene polymerization process, which limits the performance of advanced process control [1]. The relationships between process variables and quality variables can be utilized to develop soft sensors. The inferential estimation of polymer MI can be achieved from the measured process variables if the relationships between them are known. However, detailed mechanistic models for complex industrial processes are usually very difficult to develop. In this case, data-driven modelling utilizing machine learning techniques should be capitalized. In past decades, data-driven soft sensors have been widely applied to industrial processes for online monitoring and process optimization. They have many advantages such as versatility, fast response, low cost and flexibility [2–5].

Data-driven empirical models become very popular during the past three decades. Empirical models can be developed rapidly based on a large amount of historical data collected in the process plants. Due to the fast development of machine learning and advanced process control techniques, data-driven soft sensors have many successful applications. The most notable linear modeling techniques are principal component regression (PCR) and partial least squares (PLS). Besides, nonlinear data-driven modelling techniques emerged for nonlinear process modelling, such as artificial neural networks (ANN) and support vector machine (SVM). ANN has been proved being capable of approximating any nonlinear functions. The conventional ANN is built as a shallow structure network. The key training parameters of ANN, weights between neurons in adjacent layers, are modified by using training methods such as the backpropagation algorithm [6]. However, conventional ANNs have difficulty in meeting the high demand of modelling accuracy when training with data from highly nonlinear industrial processes. Conventional ANNs often meet the problem of poor generalizations because of their shallow architecture [7]. Hinton presented deep learning technique which builds a network with deep architecture for modelling highly nonlinear systems [8]. One well known deep neural network is the deep belief network (DBN). DBN is developed based on restricted Boltzmann machine (RBM). It has strong generalization capability for modelling highly nonlinear systems. The deep learning algorithm is widely used in DBN, long short-term memory (LSTM) network, stacked autoencoder (SAE), and convolutional neural network (CNN) [9,10]. Many techniques based on deep learning technique are widely used in many areas, such as nonlinear process modelling, image classification and speech recognition [11,12].

Compared with conventional neural networks, the big difference of DBN is that it has a deep network structure. Conventional neural networks usually have no more than three layers in one network. When a neural network with more than three layers is trained using the conventional training methods, its performance cannot meet the high demand of accuracy and the training result is not reliable. In order to solve this problem, DBN is proposed by Hinton and it can achieve accurate and reliable results. Model input data are used to train a DBN model in an unsupervised way at first. DBN can extract deep features in raw input variables. Then, weights in the whole network are fine-tuned by the backpropagation algorithm with the target values involved. Shang et al. [13] applied DBN in the estimation of the 95% cut point of heavy diesel in a crude distillation unit. It gives more accurate estimations and shows stronger ability to represent highly non-linear processes than traditional data-driven modelling methods [13]. DBNs also have good performance for feature representation and fault diagnosis in chemical processes [14]. In order to model nonlinear dynamic

processes, LSTM is proposed by Hochreiter and Schmidhuber [15] and applied to soft-sensor development [16,17]. It solves the problem of the gradient vanishing and explosion in conventional recurrent neural networks. LSTM is widely applied in emotion-sensitive artificial listening and machine translation [18].

One of the most important criteria for data-driven modelling is robustness or reliability. Though DBN can extract more latent information from process data. However, the unsupervised training phase cannot guarantee the latent features from process data is appropriate for the supervised training phase. It causes the model being non-robust and the predictions could become unreliable. Model robustness can be used to judge the performance of models in real industrial applications. Among the various techniques for improving model robustness, the method of combining multiple models has been shown to be very effective. Bootstrap aggregated neural network was proposed by Zhang [19]. It is shown that this type of aggregated models can achieve great performance in many applications with improved robustness [20–24]. Based on the idea of stacking several networks to enhance model performance, many approaches were proposed in recent years. Stacked extreme learning machines was given by Zhou et al. [25] and deep analytic network (DAN) was proposed by Low and Teoh [26]. However, these methods were applied to 3D pattern recognition and classification. Few stacking-based deep networks have been applied to the inferential estimation of chemical product quality. In order to improve the robustness of DBN, this paper proposes bootstrap aggregated DNB (BAGDBN). Bootstrap re-sampling with replacement is applied to the original modelling data to generate multiple replications. A DBN model is developed on each replication and these individual DBN models are then combined. BAGDBN is developed in this study and demonstrated on two application examples, modelling of a conic tank and inferential estimation of polymer MI in an industrial polypropylene polymerization process.

The remaining parts of this paper are organized as follows, DBN model and its main algorithm are presented in section 2. Section 3 introduces the main idea of BAGDBN model and the training process. In section 4, the cases studies on a water tank and a polypropylene polymerization process are given. The results and discussions are given in section 5. Conclusions of this study are drawn in section 6.

2. Deep belief network

2.1. Restricted Boltzmann machine

Deep belief network intends to improve the generalization capability of conventional neural network. In order to approximate various regions of a process, the model needs more hidden neurons added to the hidden layers. It is suggested that the networks with a deep structure can achieve reliable results in recent research [8]. In a DBN model, several restricted Boltzmann machines (RBMs) can be stacked and combined as one learning network. DBN is developed with a deep structure based on deep learning technique. Figure 1 shows the structure of DBN. It can be seen that this network contains three hidden layers, an input layer and an output layer. In Figure 1, **W** represents the weights of the network, **b** and **c** are bias of the network, y is the output of the network, and h represents the hidden layer in RBMs. It can be considered that DBN is a combination of stacking RBMs. Each hidden layer of DBN is regarded as one single RBM. Compared with traditional Boltzmann machine, the neurons in a hidden layer of DBN are not connected to each other.

The units in hidden layers are binary units and the visible input layer units are Gaussian units. The first phase of training is unsupervised training and the process operational data are used to train the DBN model without any target variables involved. The unsupervised training helps the DBN to mine more correlations than feed-forward neural network. The weights are adjusted in a desired region before the supervised training phase. After unsupervised training, DBN is fine-tuned by the backpropagation algorithm in the supervised training phase.

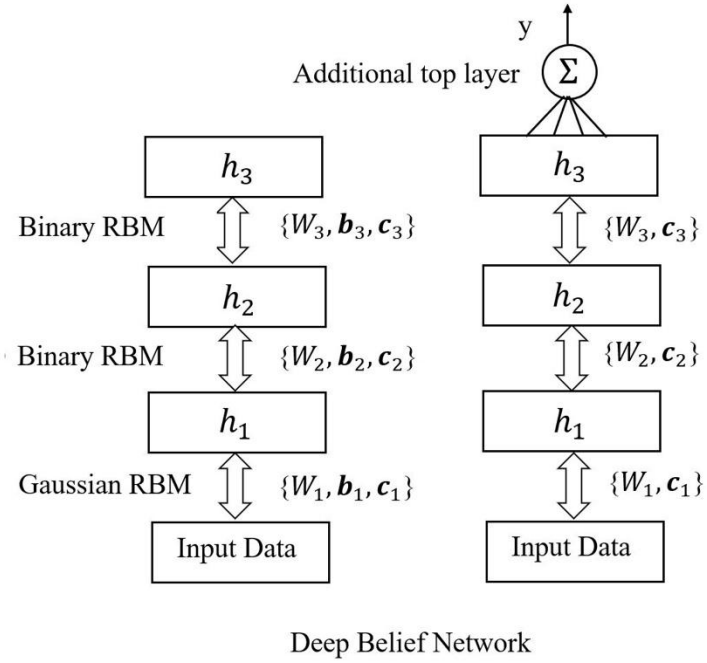

Figure 1. The architecture of DBN.

In the 1980s, Paul Smolensky developed Restricted Boltzmann machine [27]. Hinton et al. developed DBN by stacking RBMs as the layers of DBN [8]. To understand the basics of RBM, the probability function between visible units and hidden units need to be introduced first. Equation (1) shows the probability function,

$$P(\boldsymbol{v}, \boldsymbol{h}) = (\exp\{-\text{Energy}(\boldsymbol{v}, \boldsymbol{h})\})/Z \tag{1}$$

where Z represents a normalizing factor, $\boldsymbol{v}$ represents the vector of visible layer, $\boldsymbol{h}$ represents the vector of hidden layer which needs to be estimated by $\boldsymbol{v}$. The probability $P(\boldsymbol{v})$ increases when the energy function decreases. In the RBM, the energy function is given by,

$$Energy(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{b}^T\boldsymbol{v} - \boldsymbol{c}^T\boldsymbol{h} - \boldsymbol{h}^T\boldsymbol{W}\boldsymbol{v} \tag{2}$$

where $\boldsymbol{W}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ are parameters of the function. It should be noticed that the vector $\boldsymbol{v}$ and the vector $\boldsymbol{h}$ are both binary-valued. Binary RBMs are used as hidden layers in a DBN model. However, they cannot be used to deal with continuous variables. To overcome this issue, (2) can be extended to the energy function of Gaussian RBM,

$$Energy(\boldsymbol{v}, \boldsymbol{h}) = \sum_i \frac{(v_i - a_i)^2}{2\sigma_i^2} - \boldsymbol{c}^T\boldsymbol{h} - \boldsymbol{h}^T\boldsymbol{W}\boldsymbol{v} \tag{3}$$

where α_i is the mean of Gaussian distribution, σ_i is the standard deviation of Gaussian distribution for an input neuron. The samples of input data are commonly normalized to zero mean and unit variance in practical applications. Therefore, (3) can be changed to,

$$Energy(\boldsymbol{v},\boldsymbol{h}) = \frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{v}^T\boldsymbol{v} - \boldsymbol{b}^T\boldsymbol{v} - \boldsymbol{c}^T\boldsymbol{h} - \boldsymbol{h}^T\boldsymbol{W}\boldsymbol{v} \tag{4}$$

Hinton also described other forms of RBMs [28], but the DBN in this paper only uses Gaussian RBM and binary RBM.

2.2. Learning algorithm for RBM

The objective of training RBM is to maximize the probability $P(\boldsymbol{v})$, which can be achieved by minimizing the energy function. From Gibbs sampling, $\boldsymbol{h}$ can only be sampled from $\boldsymbol{v}$ of visible layers. Based on the previous work, the gradient at a visible point $\boldsymbol{v}$ is can be formulated as:

$$\frac{\partial log P(\boldsymbol{v})}{\partial \theta} = \frac{\partial log \sum_h P(\boldsymbol{v},\boldsymbol{h})}{\partial \theta}$$

$$= \frac{\sum_h e^{-Energy\ (\boldsymbol{v},\boldsymbol{h})}\left(\frac{\partial[-Energy(\boldsymbol{v},\boldsymbol{h})]}{\partial \theta}\right)}{\sum_h e^{-Energy\ (\boldsymbol{v},\boldsymbol{h})}} - \frac{\sum_{\tilde{\boldsymbol{v}}}\sum_{\boldsymbol{h}} e^{-Energy\ (\tilde{\boldsymbol{v}},\boldsymbol{h})}\left(\frac{\partial[-Energy(\tilde{\boldsymbol{v}},\boldsymbol{h})]}{\partial \theta}\right)}{\sum_{\tilde{\boldsymbol{v}}}\sum_{\boldsymbol{h}} e^{-Energ\ (\tilde{\boldsymbol{v}},\boldsymbol{h})}}$$

$$= \sum_h P(\boldsymbol{h}|\boldsymbol{v})\frac{\partial[-Energy\ (\boldsymbol{v},\boldsymbol{h})]}{\partial \theta} - \sum_{\tilde{v}}\sum_h P(\tilde{\boldsymbol{v}},\boldsymbol{h})\frac{\partial[-Energy\ (\tilde{\boldsymbol{v}},\boldsymbol{h})]}{\partial \theta} \tag{5}$$

where $\boldsymbol{\theta} = \{\boldsymbol{W},\boldsymbol{b},\boldsymbol{c}\}$ is a vector of the network parameters. Computing the positive term in (5) is easy because the vector $\boldsymbol{v}$ is known. However, it is intractable to deal with the negative term in (5). The contrastive divergence is a useful method to overcome the issue of calculating second-order approximation of the negative term and it offers an effective solution [13]. The process of training RBM starts with training vectors on the visible units. Then hidden units $\boldsymbol{h}^{(t)}$ can be generated from $\boldsymbol{v}^{(t-1)}$ by Gibbs sampling and update visible units $\boldsymbol{v}^{(t)}$ from $\boldsymbol{h}^{(t)}$. It is named as Markov chain. After infinity times iterations of Gibbs sampling, the visible units $\boldsymbol{v}^{(\infty)}$and hidden units $\boldsymbol{h}^{(\infty)}$ are sampled. The correlation of $\boldsymbol{v}^{(\infty)}$ and $\boldsymbol{h}^{(\infty)}$ can be measured after sampling for a long time. However, in practical situation, just one iteration of Gibbs sampling can achieve a satisfied result and the learning algorithm works well.

2.3. Supervised training through back-propagation

Back-propagation is the most commonly used supervised training approach to train neural networks. After the unsupervised training phase, the backpropagation algorithm will fine tune the whole network in the supervised training phase. The errors between the network outputs and the corresponding labels are computed and backpropagated to the previous layer. Equation (6) shows the error terms,

$$\boldsymbol{Err}_j = \boldsymbol{O}_j\left(1 - \boldsymbol{O}_j\right)\left(\boldsymbol{T}_j - \boldsymbol{O}_j\right) \tag{6}$$

where $\boldsymbol{O}_j$ represents the network output for a training sample, $\boldsymbol{T}_j$ is the corresponding target value for the *j*th output neuron. The error term of hidden layers is formulated as,

$$\boldsymbol{Err}_j = \boldsymbol{O}_j\left(1-\boldsymbol{O}_j\right)\sum_k \boldsymbol{Err}_k \boldsymbol{w}_{jk} \tag{7}$$

where $\boldsymbol{w}_{jk}$ is the vector of weights connecting output layer and the last hidden layer, $\boldsymbol{Err}_k$ is the error term of output layer. During training, the weight updating is transferred from the output layer to the input layer. The formulas of weight updating are given as,

$$\boldsymbol{w}_{ij} = \boldsymbol{w}_{ij} + \eta \boldsymbol{Err}_j \boldsymbol{O}_i \tag{8}$$

$$\boldsymbol{c}_j = \boldsymbol{c}_j + \eta \boldsymbol{Err}_j \tag{9}$$

where η is the learning rate of the training process, $\boldsymbol{w}_{ij}$ and $\boldsymbol{c}_j$ are the vectors of weights and bias respectively. The learning rate needs to be properly selected. A large learning rate may miss the minimum whereas a small learning rate usually leads to slow training speed.

As described earlier, the training of DBN contains an unsupervised training phase and a supervised training phase. The initial weights are adjusted to an appropriate region in the unsupervised training procedure. The whole network is then fine-tuned by backpropagation in the supervised training phase to achieve accurate modelling results. The profuse latent information extracted from input variables during the unsupervised training is more interpretable. This semi-supervised method improves the robustness and generalization capability of the model with a deep architecture.

3. Bootstrap aggregated deep belief network

The main idea of BAGDBN is to develop multiple DBN models and then combine them to improve model prediction reliability and accuracy. In order to increase the diversity of these individual DBN models, each DBN model is developed from a replication of the original modelling data set generated through bootstrap resampling. Suppose that the size (number of samples) of the original modelling data set is *b*. The number of combined DBNs in a BAGDBN is *n*. The original process and quality data are sampled with replacement for *b* times to generate a bootstrap re-sampling replication. A DBN model is developed based on each of these replications. These developed DBN models are then combined. A diagram of BAGDBN is shown in Figure 2. These individual DBN models in a BAGDBN are trained to find the relationship between process data and quality data of processes. Predictions from these individual DBN models are then combined to obtain the final prediction of the BAGDBN model. The output of a BAGDBN can be formulated as,

$$f(X) = \sum_{i=1}^{n} w_i f_i(X) \tag{10}$$

where $f(X)$ is the output of BAGDBN, $f_i(X)$ is the output of the *i*th DBN, w_i is the aggregating weight of the *i*th BAGDBN, *n* is the number of DBN models in the BAGDBN model, and *X* is a vector of model inputs. Aggregating weights w_i can have big effects on the overall prediction and need to be determined properly for good modelling performance. In this paper, the aggregating weights, w_i, are set as the same value of 1/*n* for simplicity. It means the output of BAGDBN is an average of individual DBN outputs. It is shown in this study that this approach gives quite good performance.

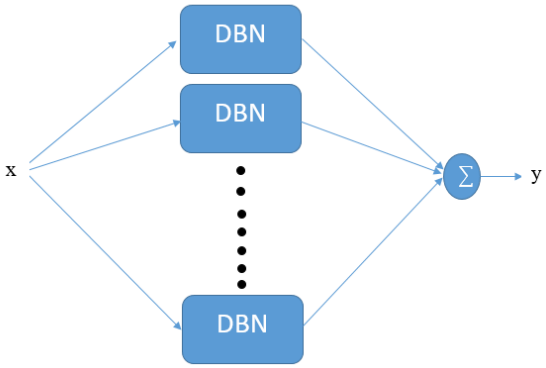

Figure 2. The structure of BAGDBN.

The BAGDBN model shown in Figure 2 contains *n* (e.g. *n* = 30) individual DBN models. This would suggest that the training effort of a BAGDBN model is *n* times more than training a single DBN model. However, this is not the case. In developing a single DBN model, a number of DBN models with different structures (e.g. different number of layers and hidden neurons) need to be developed and the one giving the best performance on the testing data is considered to have the appropriate structure. This is also known as the cross-validation based procedure for network structure determination. In developing a BAGDBN model, this cross-validation based procedure for network structure determination is only required for the first DBN. For the building of subsequent DBN models, the appropriate structure(s) identified in building the first DBN model can be used. Thus the computation time for building a BAGDBN model containing *n* individual DBN models is much less than *n* times the computation time for building a single DBN model.

4. Case studies

BAGDBN models are compared with single DBN models here on two case studies: modelling tank level in a conic water tank and inferential estimation of MI in an industrial polypropylene polymerization process.

4.1. A conic water tank

As shown in Figure 3, the conic tank has a water inlet stream and a water outlet stream [29]. The water level of the tank can be controlled by adjusting the inlet flow rate. The rate of change in the volume of water in the tank can be formulated using mass balance as:

$$\frac{dV}{dt} = Q_{in} - Q_{out} \tag{11}$$

where V is the volume of water in the tank, Q_{in} and Q_{out} represent the inlet flow rate and the outlet flow rate separately. The level of conic tank regulates the outlet flow rate, Q_{out}, as shown in (12):

$$Q_{out} = k\sqrt{h} \tag{12}$$

where h represents the tank level and k is a parameter reflecting value opening. Then, the water volume of the conic tank can be formulated as shown in (13):

$$V = \pi h(r^2 + \frac{hr}{tan\theta} + \frac{h^2}{3(tan\theta)^2}) \tag{13}$$

where r is the radius of the tank bottom, θ is the angle between the tank boundary and the horizontal plane. Therefore, the first principle dynamic model can be formulated as:

$$\frac{dh}{dt} = \frac{Q_i - k\sqrt{h}}{\pi(r^2 + \frac{hr}{tan\theta} + \frac{h^2}{3(tan\theta)^2})} \tag{14}$$

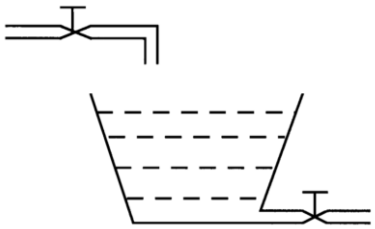

Figure 3. A conic water tank.

The first principle dynamic model is implemented in MATLAB for process simulation [29]. It can be seen from (14) that the process is clearly nonlinear. The parameters of the process model are set as fixed values, $k = 34.77cm^{2.5}/s$, $r = 10cm$, and $\theta = 60^{\circ}$. The sampling time of the dynamic model is 10 seconds. Simulated measurement noises with the distribution $N(0, 0.5)$ are added to the tank level. It is assumed that the first principle model is unavailable. A data-driven model must be developed. The developed multi-step ahead prediction model of water level can be formulated as,

$$\hat{y}(t) = [\hat{y}(t-1), \hat{y}(t-2), u(t-1), u(t-2)] \tag{15}$$

where $\hat{y}$ is the prediction of water level, t represents the discrete time, and u is the inlet water flow rate.

All the process operating data were auto scaled and divided into 3 parts. The first part is the training data set for training BAGDBN model. The second part is the testing data set which is for the determination of model structures and guarding against overfitting. The last part is the unseen validation data which is for testing the final BAGDBN model to confirm the model generalization capability. The partition of these data sets is given in Table 1.

Table 1. Partition of data sets for modelling tank level.

Data set	Percentage	Number of samples
Training data	47%	188
Testing data	19%	77
Unseen validation data	34%	133

4.2. A polypropylene polymerization process

The industrial process of polypropylene polymerization is located in China. This industrial process contains two continuous stirred tank reactors (CSTR) and two fluidized-bed reactors (FBR)

in series. Figure 4 shows the polypropylene polymerization process. To improve the efficiency of polypropylene polymerization process and reduce the operating costs, advanced monitoring and control techniques need to be applied. As a key product quality variable, MI indicates the polymer quality and needs to be closely monitored and controlled. However, MI cannot be measured on-line. Inferential estimation of MI is thus required for the advanced monitoring and control of this process.

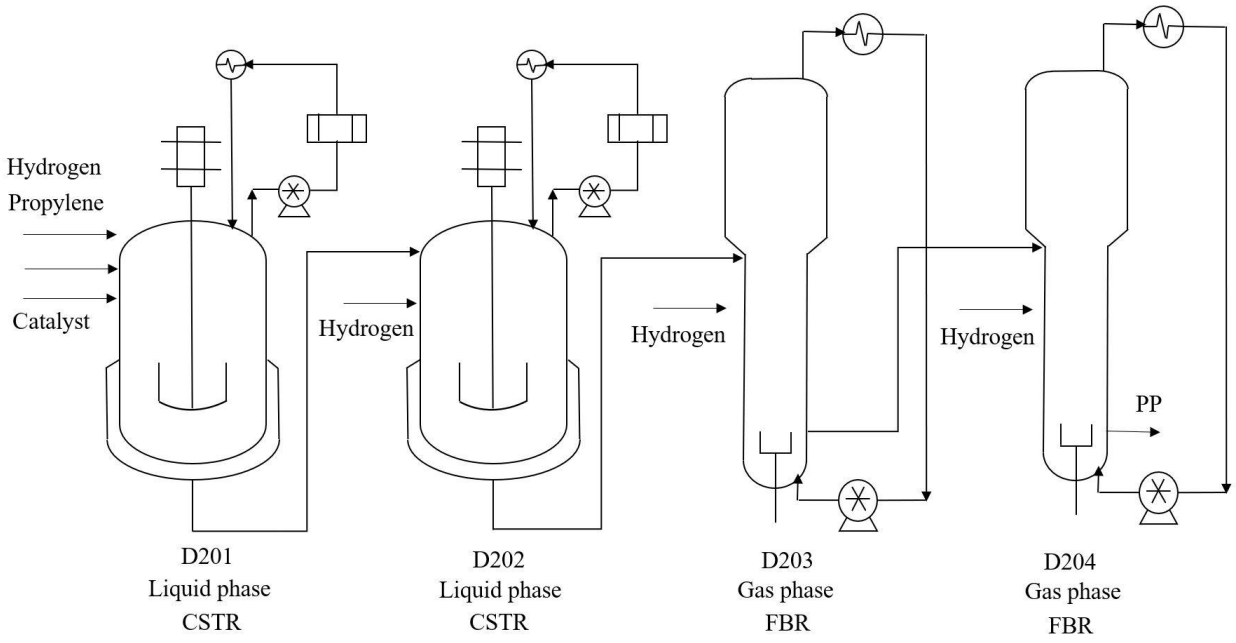

Figure 4. A propylene polymerization process.

The inputs of this polymerization process are hydrogen, propylene and catalyst. They are fed to the first reactor in Figure 4. They are the reactants to produce products and the provider of the heat transfer media. Co-monomer are added to reactor D204. This industrial process is too complicated to develop a first principle model for inferential estimation of MI. However, MI of polymer is related to many factors, such as temperature, catalyst, feedstock and heat transfer media. The initial polymerization rate of polypropylene is regulated by hydrogen [30]. Therefore, there are correlations between MI and some input variables. A BAGDBN is developed to estimate MI from the on-line measured process variables.

The process operating data set provided by the plant cover 30 days of process operation with different grades of polymer produced. Polymer melt index was measured every two hours in the laboratory. Measurements of process variables were logged every half hour. However, not all of them are highly relevant to polymer MI. The polymer MI is highly correlated with the hydrogen concentration and hydrogen feed rate in reactor D201 [1].

The measurements of MI were available every two hours whereas measurements of the process variables were recorded every half hour. This means that the amount of process data is larger than that of quality data. There are many process operational samples without the corresponding MI samples. Among the process data samples, 1151 samples are without corresponding quality data samples. In this study, only 383 pairs of input and output samples can be used in the procedure of supervised training. In training DBN models, the ‘unlabeled’ process data have valuable information to be mined

through the procedure of pre-training. The latent information from the process variables can be extracted by BAGDBN.

Before the training of BAGDBN, the original data were split into 3 subsets, training, testing and unseen validation data sets. All the variables were scaled to within the interval from 0 to 1 before they were used for network training. Table 2 shows the partition of data sets.

Table 2. Partition of data sets for the estimation of polymer melt index.

Data set	Percentage	Number of samples
Training data	57%	217
Testing data	15%	59
Unseen validation data	28%	107

5. Results and Discussions

5.1. Multi-step ahead prediction of water level

In this case study, 30 different training and testing data sets were resampled from the initial process data sets by using bootstrap resampling with replacement. An individual DBN model was developed on each replication of the original data set. The 30 DBN models were combined into a BAGDBN model. Figure 5 gives the mean squared errors (MSE) on the training and testing data set and on the unseen validation data set from individual DBN models. The MSE values are for scaled data.

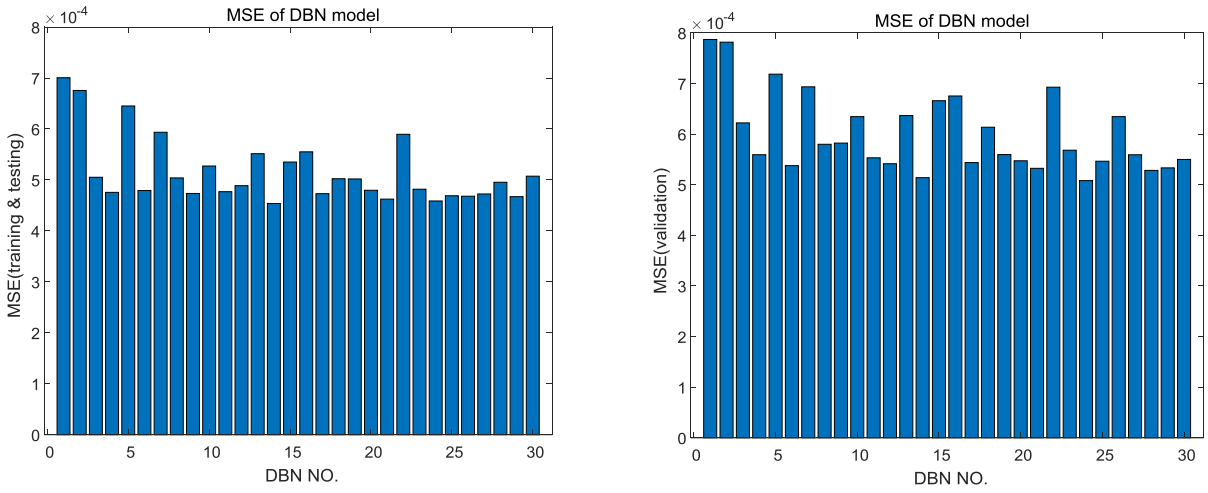

Figure 5. MSE values on the training, testing and validation data from individual DBN models.

Figure 5 indicates that the individual DBN models give various performances and their performances on the training and testing data and on the unseen validation data are not consistent. From the comparison of the two plots in Figure 5, the 10th and 13th DBN models gave similar performances on the training and testing data set. However, the MSE of the 10th DBN model on the unseen validation data is smaller than that of the 13th DBN model. The 28th DBN gives smaller MSE

on the training and testing data than the 26th DBN, but it gives larger MSE on the validation data set than the 26th DBN. These results indicate the non-robust or unreliable nature of single DBN models.

The objective of the proposed BAGDBN modelling approach is to improve the robustness of DBN models. BAGDBN models are developed by combining individual DBN models. The MSE values of predictions from BAGDBN models combining different numbers of DBN models on the training and testing data and on the unseen validation data are shown in Figure 6. The MSE values are for scaled data.

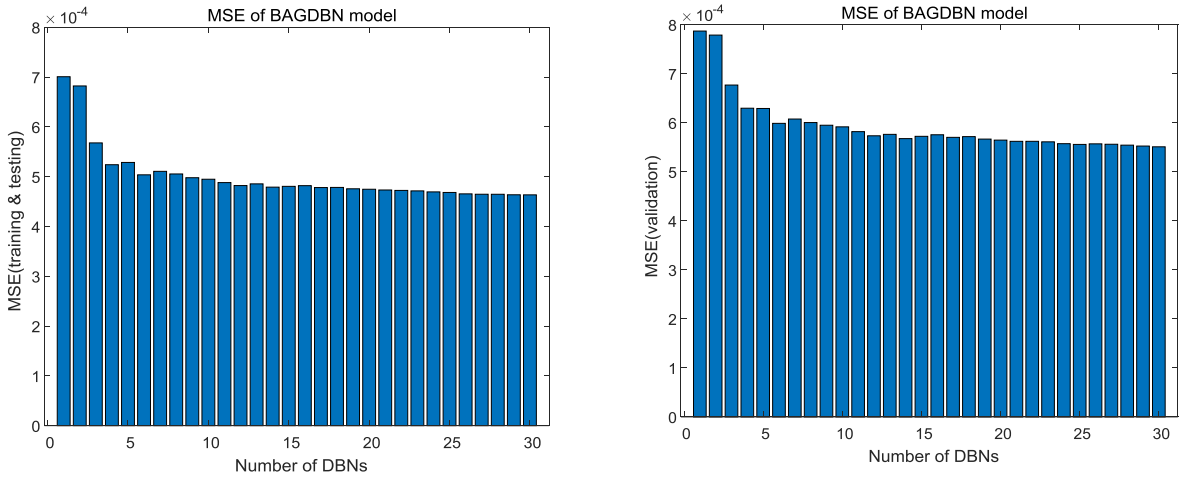

Figure 6. MSE values on the training, testing and validation data from BAGDBN models.

In Figure 6, the x-axis represents the number of DBN models included in a BAGDBN model. The first bar represents the first DBN model, the second bar represents aggregating the first two DBN models, and the last bar represents aggregating all the 30 DBN models. It can be seen from Figure 6 that the MSE values decrease as more DBN models are combined. BAGDBN models give very consistent performance on training and testing data set and unseen validation data set. The results in Figure 6 demonstrate that BAGDBN models are more robust or more reliable than single DBN models. Figure 6 also shows that, as long as sufficient number of DBN models are included (about 10), the performance of BAGDBN models is insensitive to the numbers of individual DBN models.

The multi-step ahead predictions of water level is given in Figure 7 and Table 3 compares the MSE values (on the original data scale) between the BAGDBN model and a single DBN model. In the BAGDBN model, 30 DBN models are combined. The multi-step ahead predictions of water level achieved by BAGDBN are more accurate than those from a single DBN. Table 3 indicates that the MSE of BAGDBN is smaller than that of DBN. It means that BAGDBN gives more reliable and accurate predictions than DBN.

Table 3. MSE of predictions by DBN and BAGDBN on the unseen validation data.

Model	MSE
DBN	0.7717
BAGDBN	0.5026

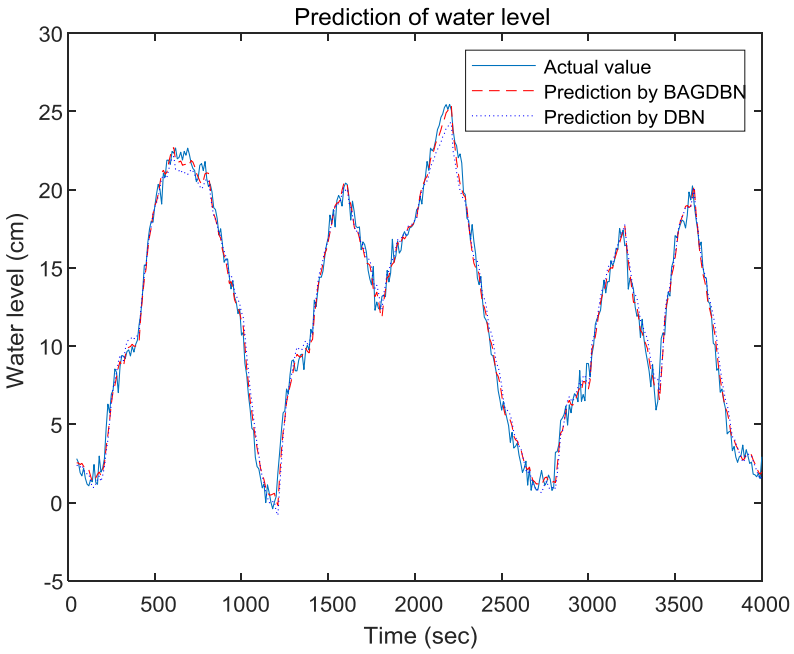

Figure 7. Multi-step ahead predictions of water level.

5.2. Estimation of polymer melt index

As with the previous case study, 30 replications of the training and testing data sets were generated from the original data set through bootstrap resampling with replacement. These data sets were used to train BAGDBN models. During the procedure of unsupervised training, the input variables without corresponding target samples were used to pre-train BAGDBN. After that, BAGDBN were fine-tuned using resampled training and testing data sets through supervised training. Figure 8 shows the MSE values of individual DBN models on the training and testing data and unseen validation data. Note that the MSE values are for scaled data. It can be seen from Figure 8 that the 1st, 2nd and 18th DBN models give similar performance on the training and testing data set. However, the MSE values of the 2nd and 18th DBN models on the unseen validation data are much smaller than that of the 1st DBN model. The 28th DBN gives smaller MSE on the training and testing data than the 29th DBN, but it gives larger MSE on the unseen validation data set than the 29th DBN. These indicate that single DBN models lack robustness or reliability.

Figure 9 shows the MSE values of BAGDBN models with different number of DBN models combined. In Figure 9, the first bar represents the first DBN model, the second bar represents aggregating the first two DBN models, and the last bar represents aggregating all the 30 DBN models. Again the MSE values are for scaled data. It can be seen from Figure 9 that the MSE values on the training and testing data and on the unseen validation data have similar trends. These MSE values decrease with the number of DBN models and then stabilize. Figure 9 also shows that, as long as sufficient number of DBN models are included (about 10), the performance of BAGDBN models is insensitive to the numbers of individual DBN models. The results in Figure 8 and Figure 9 indicate that BAGDBN models are more reliable and robust than single DBN models.

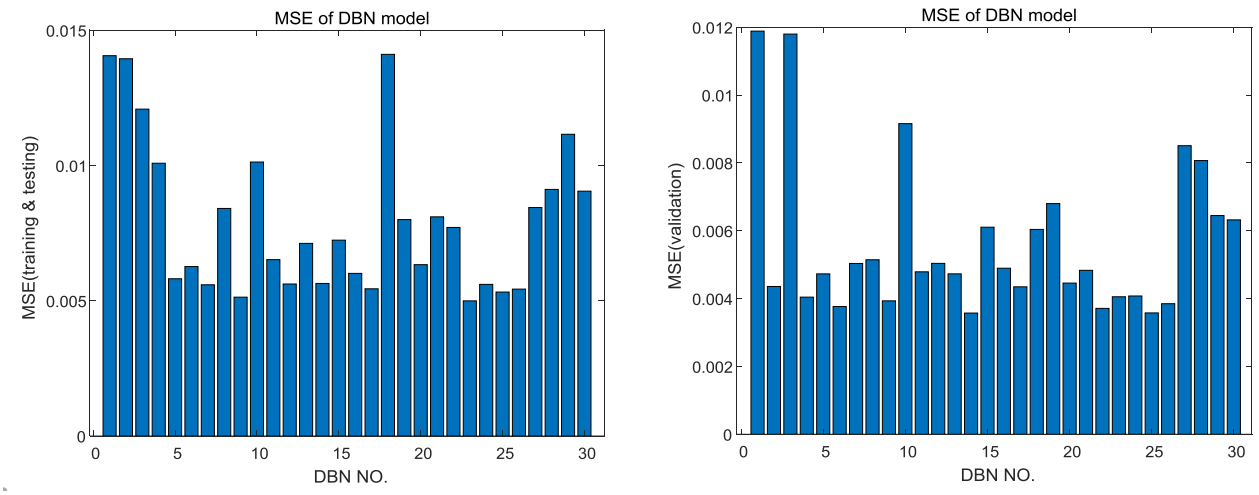

Figure 8. MSE values on training, testing and validation data from single DBN models.

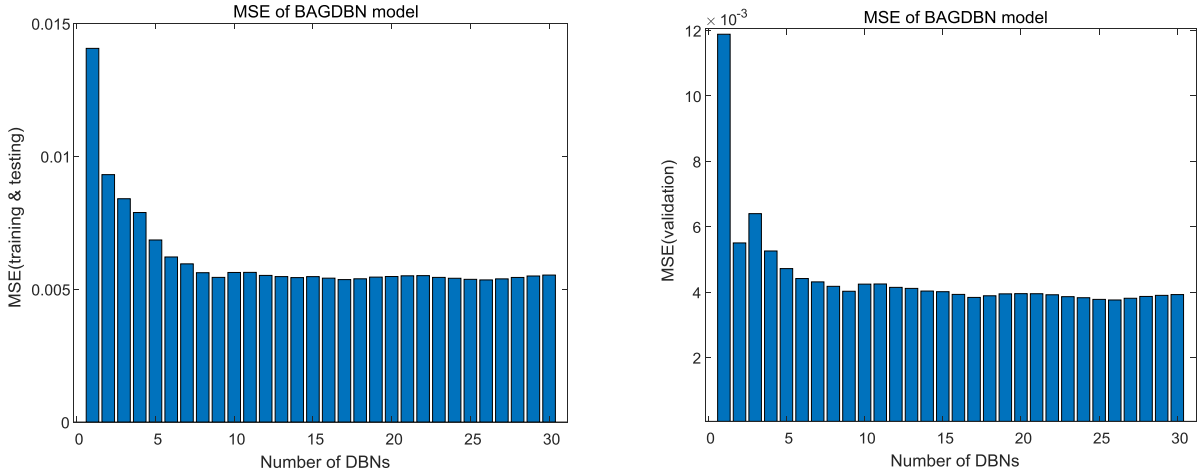

Figure 9. MSE values on training, testing and validation data from BAGDBN models.

Figure 10 shows the estimation of MI (on the original scale) achieved by DBN and BAGDBN. Table 4 gives the RMSE (on scaled data) from a conventional feedforward neural network, BAGDBN and DBN on the unseen validation data. It can be seen from Figure 10 that the BAGDBN model gives more accurate estimations than DBN. Table 4 shows that DBN gives smaller RMSE values than conventional neural network on the unseen validation data. The RMSE values from BAGDBN are smaller than those from the conventional neural network and DBN. Hence, the advantage of BAGDBN over DBN is clear.

Table 4. RMSE values of estimation from conventional neural network, DBN and BAGDBN.

Model	RMSE (training & testing)	RMSE (validation)
Conventional Neural Network	0.085	0.088
DBN	0.089	0.082
BAGDBN	0.072	0.061

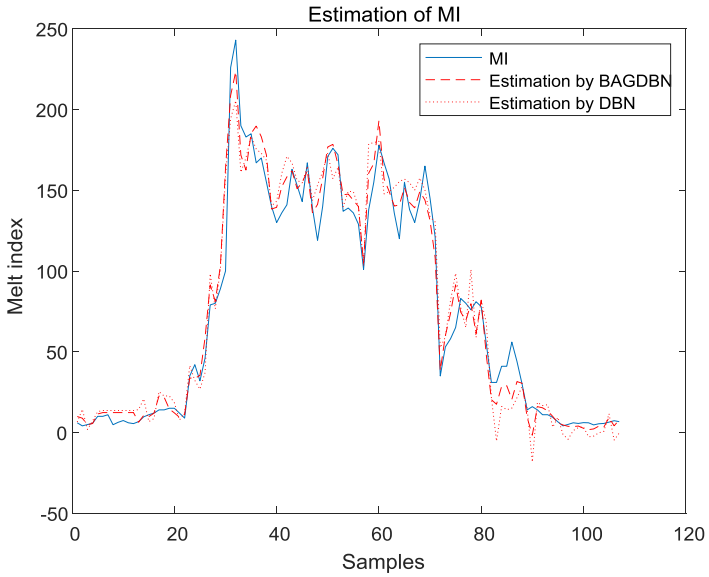

Figure 10. Estimation of polymer melt index.

6. Conclusions

A novel data-driven modelling approach through integrating multiple DBN is proposed in this paper. BAGDBN improves robustness of data-driven nonlinear models and achieves accurate estimations of process quality data. In this study, multiple DBNs are established based on different bootstrap resampling replications from the original process modelling data set and are combined as one BAGDBN model. By aggregating multiple DBN models, the failure of a particular DBN model can be compensated by other DBN models. The effectiveness of BAGDBN is demonstrated on two application examples, dynamic modelling of a conic water tank and inferential estimation of MI in an industrial polypropylene polymerization process. A BAGDBN model gives better multi-step ahead prediction performance than a single DBN model. It is also more robust than a single DBN model in that it can give consistent good performance on different sets of data. In the estimation of polymer MI, the BAGDBN model gives more accurate and reliable predictions than conventional neural network and DBN models. In polypropylene polymerization process, there are a large number of process data samples without the corresponding quality data samples and they cannot be used by conventional supervised training models. However, these unlabeled data samples can be used in the unsupervised training phase of DBN and BAGDBN, which can extract more latent information to improve the estimation of polymer MI. One limitation of BAGDBN is the long training time required as more DBN models need to be trained. This will be improved in the future by developing new BAGDBN algorithms through sequential training and selective combination of individual DBN models.

Acknowledgments

The work has been supported by the EU (Project No. PIRSESGA-2013-612230) and National Natural Science Foundation of China (61673236).

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. Zhang J, Jin Q, Xu Y (2006) Inferential estimation of polymer melt index using sequentially trained bootstrap aggregated neural networks. *Chemical Engineering & Technology: Industrial Chemistry-Plant Equipment-Process Engineering-Biotechnology* 29: 442–448.
2. Shao W, Yao L, Ge Z, et al. (2018) Parallel computing and SGD-based DPMM for soft sensor development with large-scale semisupervised data. *IEEE T Ind Electron* 66: 6362–6373.
3. Shao W, Ge Z, Song Z (2019) Quality variable prediction for chemical processes based on semi-supervised Dirichlet process mixture of Gaussians. *Chem Eng Sci* 193: 394–410.
4. Yuan X, Ou C, Wang Y, et al. (2020) Deep quality-related feature extraction for soft sensing modeling: A deep learning approach with hybrid VW-SAE. *Neurocomputing* 396: 375–382.
5. Yuan X, Zhou J, Huang B, et al. (2019) Hierarchical quality-relevant feature representation for soft sensor modeling: a novel deep learning strategy. *IEEE T Ind Inform* 16: 3721–3730.
6. Werbos P (1974) Beyond regression: new fools for prediction and analysis in the behavioral sciences. Harvard University.
7. Rosa E, Yu W (2016) Randomized algorithms for nonlinear system identification with deep learning modification. *Inform Sciences* 364: 197–212.
8. Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. *Neural Comput* 18: 1527–1554.
9. Yuan X, Huang B, Wang Y, et al. (2018) Deep learning-based feature representation and its application for soft sensor modeling with variable-wise weighted SAE. *IEEE T Ind Inform* 14: 3235–3243.
10. Yuan X, Ou C, Wang Y, et al. (2019) A layer-wise data augmentation strategy for deep learning networks and its soft sensor application in an industrial hydrocracking process. *IEEE T Neur Net Lear*.
11. Mnih A, Hinton GE (2009) A scalable hierarchical distributed language model. *Advances in Neural Information Processing Systems*, 1081–1088.
12. Li F, Zhang J, Shang C, et al. (2018) Modelling of a post-combustion CO2 capture process using deep belief network. *Appl Therm Eng* 130: 997–1003.
13. Shang C, Yang F, Huang D, et al. (2014) Data driven soft sensor development based on deep learning technique. *J Process Contr* 24: 223–233.
14. Wang Y, Pan Z, Yuan X, et al. (2020) A novel deep learning based fault diagnosis approach for chemical process with extended deep belief network. *ISA Transactions* 96: 457–467.
15. Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9: 1735–1780.
16. Yuan X, Li L, Wang Y (2019) Nonlinear dynamic soft sensor modeling with supervised long short-term memory network. *IEEE T Ind Inform* 16: 3168–3176.
17. Yuan X, Li L, Shardt Y, et al. (2020) Deep learning with spatiotemporal attention-based LSTM for industrial soft sensor model development. *IEEE T Ind Electron*.
18. Wöllmer M, Schuller B, Eyben F, et al. (2010) Combining long short-term memory and dynamic Bayesian networks for incremental emotion-sensitive artificial listening. *IEEE J-STSP* 4: 867–881.

19. Zhang J (1999) Developing robust non-linear models through bootstrap aggregated neural networks. *Neurocomputing* 25: 93–113.
20. Zhang J, Pantelelis NG (2011) Modelling and optimisation control of polymer composite moulding processes using bootstrap aggregated neural network models. *2011 International Conference on Electric Information and Control Engineering*, 2363–2366.
21. Kaunga DL, Zhang J, Ferguson K (2013) Reliable modelling of chemical durability of high-level waste glass using bootstrap aggregated neural networks. *2013 Ninth International Conference on Natural Computation*, 178–183.
22. Mohammed KJR, Zhang J (2013) Reliable optimisation control of a reactive polymer composite moulding process using ant colony optimisation and bootstrap aggregated neural networks. *Neural Comput Appl* 23: 1891–1898.
23. Osuolale FN, Zhang J (2018) Exergetic optimization of atmospheric and vacuum distillation system based on bootstrap aggregated neural network models. *Exergy for A Better Environment and Improved Sustainability* 1: 1033–1046.
24. Mukherjee A, Zhang J (2008) A reliable multi-objective control strategy for batch processes based on bootstrap aggregated neural network models. *J Process Contr* 18: 720–734.
25. Zhou H, Huang GB, Lin Z, et al. (2015) Stacked extreme learning machines. *IEEE T Syst Man Cy B* 45: 2013–2025.
26. Low CY, Teoh ABJ (2017) Stacking-based deep neural network: Deep analytic network on convolutional spectral histogram features. *2017 IEEE International Conference on Image Processing (ICIP)*, 1592–1596.
27. Smolensky P (1986) Information processing in dynamical systems: Foundations of harmony theory. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* 1: 194–281.
28. Hinton GE (2012) A practical guide to training restricted Boltzmann machines. *Neural networks: Tricks of the trade*, 599–619.
29. Zhang J, Morris AJ (2000) Long range predictive control of nonlinear processes based on recurrent neurofuzzy network models. *Neural Comput Appl* 9: 50–59.
30. Soares JBP, Hamielec AE (1996) Kinetics of propylene polymerization with a non-supported heterogeneous Ziegler-Natta catalyst—effect of hydrogen on rate of polymerization, stereoregularity, and molecular weight distribution. *Polymer* 37: 4607–4614.

AIMS Press