Int J Coal Sci Technol (2017) 4(1):33–40
DOI 10.1007/s40789-017-0158-1

Modelling of a post-combustion CO_2 capture process using extreme learning machine

Fei Li[1] **· Jie Zhang**[1] **· Eni Oko**[2] **· Meihong Wang**[2]

Received: 21 August 2016 / Revised: 13 January 2017 / Accepted: 13 February 2017 / Published online: 23 February 2017

Abstract This paper presents modelling of a post-combustion CO_2 capture process using bootstrap aggregated extreme learning machine (ELM). ELM randomly assigns the weights between input and hidden layers and obtains the weights between the hidden layer and output layer using regression type approach in one step. This feature allows an ELM model being developed very quickly. This paper proposes using principal component regression to obtain the weights between the hidden and output layers to address the collinearity issue among hidden neuron outputs. Due to the weights between input and hidden layers are randomly assigned, ELM models could have variations in performance. This paper proposes combining multiple ELM models to enhance model prediction accuracy and reliability. To predict the CO_2 production rate and CO_2 capture level, eight parameters in the process were utilized as model input variables: inlet gas flow rate, CO_2 concentration in inlet flow gas, inlet gas temperature, inlet gas pressure, lean solvent flow rate, lean solvent temperature, lean loading and reboiler duty. The bootstrap re-sampling of training data was applied for building each single ELM and then the individual ELMs are stacked, thereby enhancing the model accuracy and reliability. The bootstrap aggregated extreme learning machine can provide fast learning speed and good generalization performance, which will be used to optimize the CO_2 capture process.

Keywords CO_2 capture · Neural networks · Data-driven modelling · Extreme learning machine

1 Introduction

Greenhouse emissions (GHE), mainly carbon dioxide (CO_2), is identified as the chief reason resulting in the global climate change, especially the global warming. The growing energy demand, due to rapid increasing population and development of industrialization, are directly linked to the increasing release of GHE. The target of a 50% reduction of CO_2 emission by 2050 comparing with the level in 1950 is set by the Intergovernmental Panel on Climate Change.

Carbon capture and storage (CCS) has been widely believed as an advanced technology to achieve CO_2 emission reduction, which captures, transports and stores CO_2. There are three major types of technologies applied for CCS: post-combustion, pre-combustion and oxyfuel combustion. Among these various CCS technologies, post-combustion CO_2 capture (PCC) process is considered as the most convenient way to reduce CO_2 emission from coal fired power plants, as it can retrofit the exiting power plant and be integrated into new ones. However, PCC process will generate a large amount of energy penalty, which reduces the efficiency and effectiveness of the power plant. The energy requirement is strongly influenced by the operation conditions, equipment dimensions and capture target of PCC process. Therefore, it is necessary to apply process optimisation in order to enhance the efficiency of CCS systems.

✉ Jie Zhang
jie.zhang@newcastle.ac.uk

[1] School of Chemical Engineering and Advanced Materials, Newcastle University, Newcastle upon Tyne NE1 7RU, UK

[2] Department of Chemical and Biological Engineering, University of Sheffield, Sheffield S1 3JD, UK

In order to optimize the operation of post-combustion CO_2 capture process, a reliable and accurate process model is necessary. In the past, researchers have proposed various kinds of modelling technologies, such as mechanistic models (Lawal et al. 2010; Biliyok et al. 2012; Posch and Haider 2013; Cormos and Daraban 2015) and data-driven models (Zhou et al. 2009, 2010; Sipocz and Assadi 2011). However, some problems have been raised up by using the above mentioned methods. For instance, the development of mechanistic model is not only time consuming, but also needs a huge volume of knowledge of the underlying first principles of the process. It is also computationally very demanding when using a detailed mechanistic model in process optimisation. Statistic models can overcome these problems and are efficient in building data driven models, but they still have a few shortcomings. It is shown in that statistical model is unable to describe the nonlinear relationships that possibly exits among the parameters (Zhou et al. 2010). In this case, another advanced modelling method, artificial neural networks (ANNs), is proposed to address the above weakness. However, feedforward neural networks trained by the back propagation (BP) learning algorithm have some issues: firstly, the most appropriate learning rate is difficult to determine; secondly, the presence of local minima affects the modelling results; then, networks would possibly be over trained leading to poor generalization performance; lastly, it is also time-consuming when applying gradient based learning (Huang et al. 2006).

Extreme learning machine (ELM) was proposed into address the issue of slow training in conventional feedforward neural networks (Huang et al. 2006). ELM is basically a single hidden layer feedforward neural network with randomly assigned weights between the input and hidden layers. The weights between the hidden and output layers are determined in a one-step regression type approach using generalised inverse. Thus, an ELM can be built very quickly. As the weights between the input and hidden layers are randomly assigned, correlations can exist among the hidden neuron outputs and variations in model performance can exist. This paper proposes using principal component regression (PCR) to obtain the weights between the hidden and output layers in order to overcome the correlation issue among hidden neuron outputs. This paper also proposes building multiple ELMs on bootstrap re-sampling replications of the original training data and then combining these ELMs in order to enhance model accuracy and reliability. The proposed method is applied to the dynamic model development of the whole post-combustion process plant.

This paper is structured as follows: Sect. 2 briefly presents post-combustion CO_2 capture process through chemical absorption. Extreme learning machine, a method for calculating output layer weights in ELM using PCR, and aggregating multiple ELM are given in Sect. 3. Application results and discussions are presented in Sect. 4. Section 5 draws some concluded remarks.

2 CO_2 capture process through chemical absorption

Figure 1 shows a typical post-combustion CO_2 capture process through chemical absorption. It consists of two major parts: an absorber and a stripper. In details, the flue gas from the power plant is pressured into the bottom of absorber and contacted counter-currently with lean MEA solution from the top side. The lean MEA solution will chemically absorb the CO_2 in flue gas, forming rich amine solution. The treated gas stream containing much lower CO_2 content is leaving from the top of absorber. Then the rich amine solution is pressured into the regenerator before preheating in the cross heat exchanger. In the stripper, CO_2 is separated from rich amine solution by the heat provided from the reboiler. The regenerated CO_2 is cooled in condenser and compressed for storage, and remaining solution (lean solution) is recycled to the cross heat exchanger to exchange heat with rich amine. The heat supplied in the reboiler, coming from the low pressure steam from power plant, is used to increase the temperature of solution, separate CO_2 from rich amine and vaporize the gas in stripper. This will result in a large energy consumption.

Two parameters are identified to affect the process performance: CO_2 capture level and CO_2 production rate. CO_2 capture level is the amount of CO_2 extracted from the

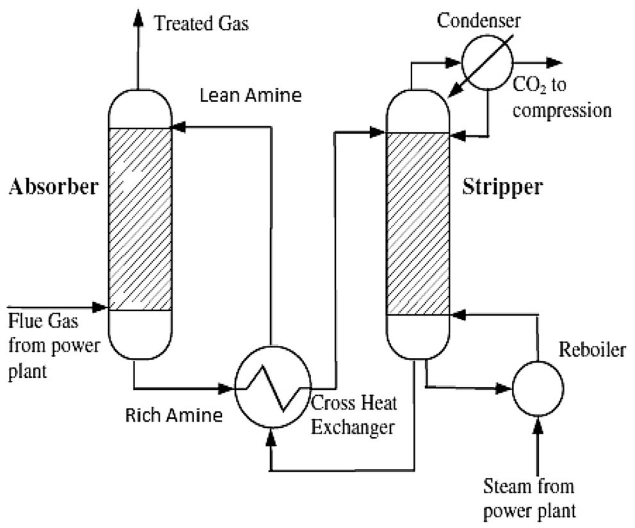

Fig. 1 Simplified process flow diagram of chemical absorption process for post-combustion capture plant

inlet flue gas in absorber column, which is calculated in Eq. (1).

$$\eta_{\text{co}_2\text{ capture}} = 1 - \frac{m_{\text{outlet co}_2} \times V_{\text{outlet gas}}}{m_{\text{inlet co}_2} \times V_{\text{inlet gas}}} \tag{1}$$

where, $m_{\text{outlet co}_2}$, $V_{\text{outlet gas}}$, $m_{\text{inlet co}_2}$ and $V_{\text{inlet gas}}$ represent CO$_2$ mass fraction in gas out of absorber, gas flow rate out of absorber, CO$_2$ mass fraction in inlet flow gas of absorber, and inlet gas flow rate of absorber, respectively.

CO$_2$ production rate represents the amount of CO$_2$ captured after the condenser, which is an indicator for the whole process because it is not affected by a single component of the process. It is calculated as in Eq. (2):

$$\partial_{\text{co}_2} = \dot{m}_{\text{co}_2} \times \tilde{v}_{\text{outlet gas}} \tag{2}$$

where ∂_{co_2} is CO$_2$ production rate after the condenser, $\dot{m}_{\text{co}_2}$ and $\tilde{v}_{\text{outlet gas}}$ are CO$_2$ mass fraction and gas flow rate of the outlet gas from stripper respectively.

3 Bootstrap aggregated ELM

3.1 Single hidden layer neural networks

Figure 2 shows the structure of a single hidden layer feedforward neural network (SLFN). For N arbitrary distinct samples (x_i, t_i), where $x = [x_{i1}, x_{i2}, \ldots, x_{in}]^{\text{T}} \in R^n$ is a vector of network inputs and $t_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^{\text{T}} \in R^m$ is a vector of the target values of network outputs. The output of a standard SLFNs with Ñ hidden nodes and activation function $g(x)$ is shown in the following equation:

$$o_j = \sum_{i=1}^{\tilde{N}} \beta_i g_i(w_i x_j + b_i), \quad j = 1, \ldots, N \tag{3}$$

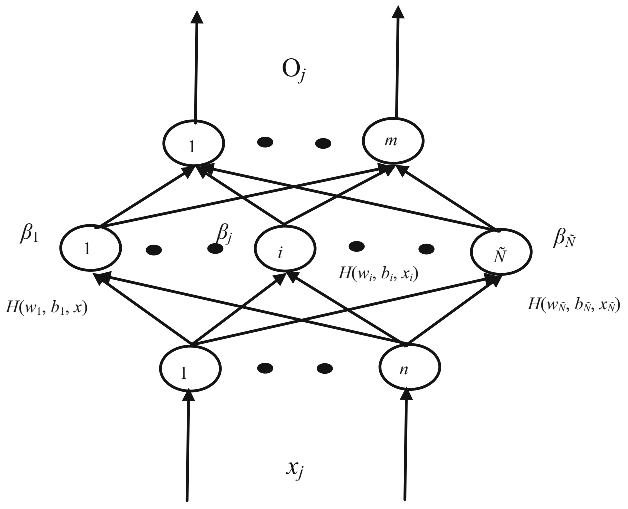

Fig. 2 The structure of single hidden layer feedforward networks

where $w_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^{\text{T}}$ is a vector of the weights between the ith hidden node and the input nodes, b_i is the bias of the ith hidden nodes, x_j is the jth input sample, $o_j = [o_{j1}, o_{j2}, \ldots, o_{jm}]^{\text{T}} \in R^m$ is a vector of the SLFN output corresponding to the jth input sample, $\beta_i \in R^m$ is a vector of the weight linking the ith hidden node and the output node. The output node is chosen to have linear activation function and the hidden layer neurons use the sigmoid activation function in this paper.

In theory, the standard SLFNs can approximate any continuous nonlinear functions with zero error, which means $\sum_{j=1}^{\tilde{N}} \|o_j - t_j\| = 0$. Specifically, there exits β_i, w_i and b_i to make:

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i(w_i \cdot x_j + b_i) = t_j, \quad j = 1, \ldots, N \tag{4}$$

The above equation can be written as $\boldsymbol{H\beta} = \boldsymbol{T}$, where:

$$H(w_1, \ldots, w_{\tilde{N}}, b_1, \ldots, b_{\tilde{N}}, x_1, \ldots, x_{\tilde{N}}) = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_{\tilde{N}} \cdot x_1 + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(w_1 \cdot x_N + b_1) & \cdots & g(w_{\tilde{N}} \cdot x_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}} \tag{5}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1^{\text{T}} \\ \vdots \\ \beta_{\tilde{N}}^{\text{T}} \end{bmatrix}_{\tilde{N} \times m} \quad \text{and} \quad \boldsymbol{T} = \begin{bmatrix} t_1^{\text{T}} \\ \vdots \\ t_N^{\text{T}} \end{bmatrix}_{N \times m} \tag{6}$$

In the above equations, $\boldsymbol{H}$ is called hidden layer output matrix of the neural network and the ith column of $\boldsymbol{H}$ is the ith hidden node output with respect to inputs $x_1, x_2, \ldots, x_N$. Training of SLFNs can be done through finding the minimum value of $E = \min\|H_{N\times N}\beta_{N\times m} - T_{N\times m}\|$.

SLFNs are usually trained by gradient-based learning algorithms, such as BP algorithm, which typically need many iterations and are typically slow. The process of training is to search the minimum value of $\|H_{N\times N}\beta_{N\times m} - T_{N\times m}\|$ by numerical optimisation methods. In this procedure, the parameters $\boldsymbol{\theta} = (\boldsymbol{\beta}, \boldsymbol{w}, \boldsymbol{b})$ is iteratively adjusted as below:

$$\boldsymbol{\theta} = \boldsymbol{\theta}_{k-1} - \eta \frac{\partial E(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{7}$$

where η is the learning rate. By using BP algorithm, the parameters are updated by error propagation from the output layer to the input layer.

3.2 Bootstrap aggregated ELM

Huang et al. have proved that, if the activation function $g(x)$ is infinitely differentiable in any interval and the number of hidden nodes is large enough, it is not necessary to adjust all the weighting parameters of the network (Huang et al. 2006). In other words, the weights and biases between the input and hidden layers can be randomly chosen. In order to get good performance, the required number of hidden nodes is not more than the number of input samples. Huang et al. have used a method of finding a least square solution of the linear equation $\boldsymbol{H\beta} = \boldsymbol{T}$ to obtain the weights between the hidden and output layers.

$$\boldsymbol{\beta} = \boldsymbol{H}^{\dagger}\boldsymbol{T} \tag{8}$$

where $\boldsymbol{H}^{\dagger}$ is the generalised inverse of $\boldsymbol{H}$.

However, as the hidden layer outputs can be collinear, the modelling performance would be poor by using least square solution to find the weights between the hidden and output layers. This would be especially true for ELM as they have randomly assigned hidden layer weights and typically large number of hidden neurons are required. This paper proposes using PCR to obtain the weights between the hidden and output layers to overcome the multi-collinearity problems. Instead of regressing $\boldsymbol{H}$ and $\boldsymbol{T}$ directly, the principal components of $\boldsymbol{H}$ matrix are used as regressors.

The matrix $\boldsymbol{H}$ can be decomposed into the sum of a series of rank one matrices through principal component decomposition.

$$\mathrm{H} = \mathrm{u}_1\mathrm{p}_1^{\mathrm{T}} + \mathrm{u}_2\mathrm{p}_2^{\mathrm{T}} + \cdots + \mathrm{u}_N\mathrm{p}_N^{\mathrm{T}} \tag{9}$$

In the above equation, $\boldsymbol{u}_i$ and $\boldsymbol{p}_i$ are the ith score vector and loading vector respectively. The score vectors are orthogonal, likewise the loading vectors, in addition they are of unit length. The loading vector $\boldsymbol{p}_1$ defines the direction of the greatest variability and the score vector $\boldsymbol{u}_1$, also known as the first principal component, represents the projection of each column of $\boldsymbol{H}$ onto $\boldsymbol{p}_1$. The first principal component is thus that linear combination of the columns in $\boldsymbol{H}$ explaining the greatest amount of variability ($\boldsymbol{u}_1 = \boldsymbol{Hp}_1$). The second principal component is that linear combination of the columns in $\boldsymbol{H}$ explaining the next greatest amount of variability ($\boldsymbol{u}_2 = \boldsymbol{Hp}_2$) subject to the condition that it is orthogonal to the first principal component. Principal components are arranged in decreasing order of variability explained. Since the columns in $\boldsymbol{H}$ are highly correlated, the first a few principal components can explain the majority of data variability in $\boldsymbol{H}$.

$$\mathrm{H} = \mathrm{U}_k\mathrm{P}_k^{\mathrm{T}} + \mathrm{E} = \sum_{i=1}^{k}\mathrm{u}_i\mathrm{p}_i^{\mathrm{T}} + \mathrm{E} \tag{10}$$

where $\boldsymbol{U}_k = [\boldsymbol{u}_1\boldsymbol{u}_2 \ldots \boldsymbol{u}_k]$, $\boldsymbol{P}_k = [\boldsymbol{p}_1\boldsymbol{p}_2 \ldots \boldsymbol{p}_k]$, k represents the number of principal components to retain, and $\boldsymbol{E}$ is a matrix of residuals of unfitted variation.

If the first k principal components can adequately represent the original data set $\boldsymbol{H}$, then regression can be performed on the first k principal components. The model output is obtained as a linear combination of the first k principal components of $\boldsymbol{H}$ as

$$\hat{\mathrm{T}} = \mathrm{U}_k\mathrm{w} = \mathrm{HP}_k\mathrm{w} \tag{11}$$

where $\boldsymbol{w}$ is a vector of model parameters in terms of principal components.

The least squares estimation of $\boldsymbol{w}$ is:

$$\mathrm{w} = \left(\mathrm{U}_k^{\mathrm{T}}\mathrm{U}_k\right)^{-1}\mathrm{U}_k^{\mathrm{T}}\mathrm{T} = \left(\mathrm{P}_k^{\mathrm{T}}\mathrm{H}^{\mathrm{T}}\mathrm{HP}_k\right)^{-1}\mathrm{P}_k^{\mathrm{T}}\mathrm{H}^{\mathrm{T}}\mathrm{T} \tag{12}$$

The model parameters in Eq. (8) calculated through PCR are then given by the following equation:

$$\beta = \mathrm{P}_k\mathrm{w} = \mathrm{P}_k\left(\mathrm{P}_k^{\mathrm{T}}\mathrm{H}^{\mathrm{T}}\mathrm{HP}_k\right)^{-1}\mathrm{P}_k^{\mathrm{T}}\mathrm{H}^{\mathrm{T}}\mathrm{T} \tag{13}$$

The number of principal components, k, to be retained in the model is usually determined through cross-validation (Wold 1978). The data set for building a model is partitioned into a training data set and a testing data set. PCR models with different numbers of principal components are developed on the training data and then tested on the testing data. The model with the smallest testing errors is then considered as having the most appropriate number of principal components.

As shown in (Zhang 1999; Li et al. 2015), combining several networks can improve the prediction accuracy on unseen data and give a better generalization performance. The bootstrap re-sampling replication of the original training data is used for training individual networks and the overall output of the aggregated neural networks is a weighted combination of the individual neural network outputs (Fig. 3).

Therefore, the procedure of building bootstrap aggregated ELM model can be summarized as follows:

Given an activation function $g(x)$, and number of hidden nodes $\tilde{N}$,

Step 1: Apply bootstrap re-sampling to produce n (e.g. $n = 50$) replications of the original training data set, $(x_i, t_i)_1, \ldots, (x_i, t_i)_n$, $x_i \in R^n$, $t_i \in R^m$, $i = 1, \ldots, N$

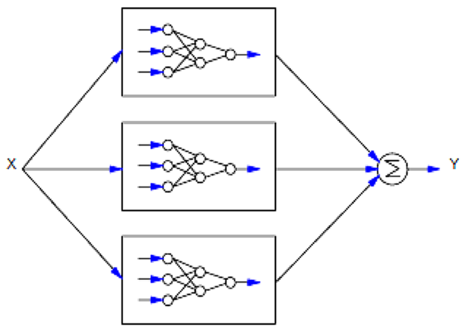

Fig. 3 A bootstrap aggregated neural network

Table 1 Performance comparison of BA-ELM and BA-NNs for CO_2 production rate

Learning algorithm	Time (CPU time) (s)		Training accuracy (MSE)	Validation accuracy (MSE)
	Training time	Verifying time (2nd batch)		
Bootstrap aggregated ELM (BA-ELM)	163.4422	0.7176	0.0488	0.0441
Bootstrap aggregated neural networks (BA-NNs)	1726.4	0.2964	0.0219	0.0771

Step 2: On each bootstrap replication of the original training data, build an ELM model:

Step 2(a): Randomly assign hidden layer weights w_i and bias b_i, $i = 1\ldots \tilde{N}$

Step 2(b): Calculate the hidden layer output matrix $\boldsymbol{H}$

Step 2(c): Calculate the output weights $\boldsymbol{\beta}$ by PCR

Step 3: Combine the n (e.g. $n = 50$) ELM models by averaging their predictions

It has been suggested that, the model prediction confidence bounds can be calculated from individual predictions by using bootstrap aggregated neural networks (Zhang 1999; Li et al. 2015). The standard error of the ith predicted value is calculated as

$$\boldsymbol{\sigma_e} = \left\{ \frac{1}{\boldsymbol{n}-1} \sum_{\boldsymbol{b}=1}^{\boldsymbol{n}} \left[\boldsymbol{y}\left(\boldsymbol{x}_i; \boldsymbol{W}^b\right) - \boldsymbol{y}(\boldsymbol{x}_i;) \right]^2 \right\}^{\frac{1}{2}} \qquad (14)$$

where $y(x_i) = \sum_{b=1}^{n} y(x_i; W^b)/n$ and n is the number of neural networks. The 95% prediction confidence bounds can be calculated as $y(x_i;) \pm 1.96\sigma_e$. It indicates a 95% confidence interval which will contain the true process output with a probability of 0.95. A narrower confidence bound is preferred as it indicates the associated model prediction is more reliable.

4 Performance evaluation

The simulated dynamic process operation data in (Li et al. 2015) were used to build data-driven models. The simulated data were generated from the mechanistic model implemented in gPROMS at University of Hull with a sampling time of 5 s. The data were divided into three groups: training data (56%), testing data (24%), and unseen validation data (20%). Furthermore, the constructed model used the input data of the second batch in which the lean solution flow rate has a step change, to verify its accuracy. To demonstrate the good performance of bootstrap aggregated ELM, its results are compared with those from (Li

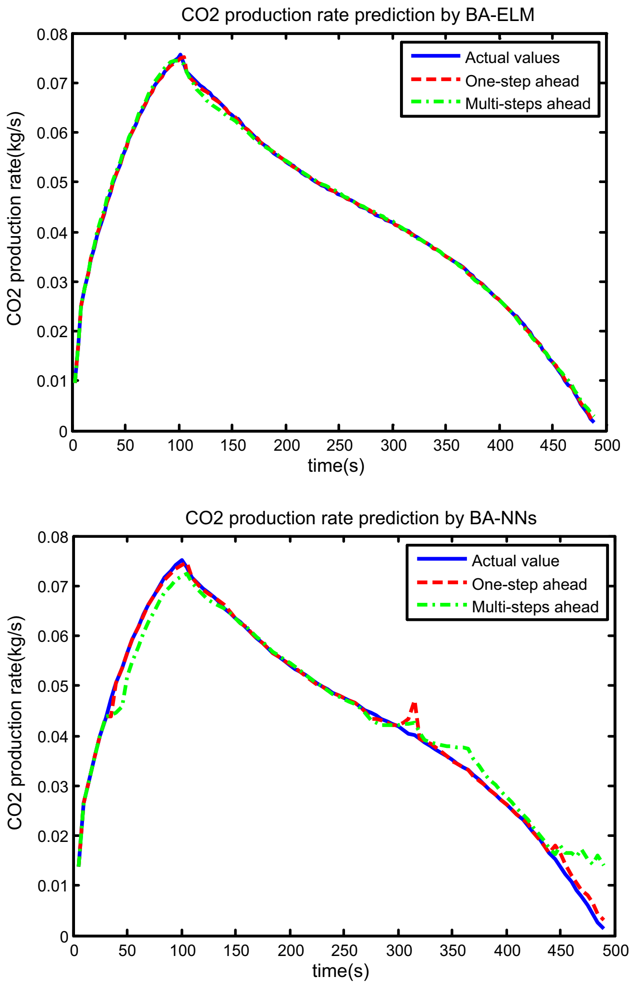

Fig. 4 Dynamic model prediction of CO_2 production rate using BA-ELM (*top*) and BA-NNs (*bottom*)

et al. 2015). Before training, the data should be scaled to zero mean and unit variance. Both bootstrap aggregated neural network (BA-NNs) and BA-ELM models combine 30 neural networks. In addition, the numbers of hidden neurons used in BA-NNs and BA-ELM are selected within the range of 2–20 and 40–100 respectively. All models with the number of hidden neurons in the above ranges are developed and tested on the testing data. The models give

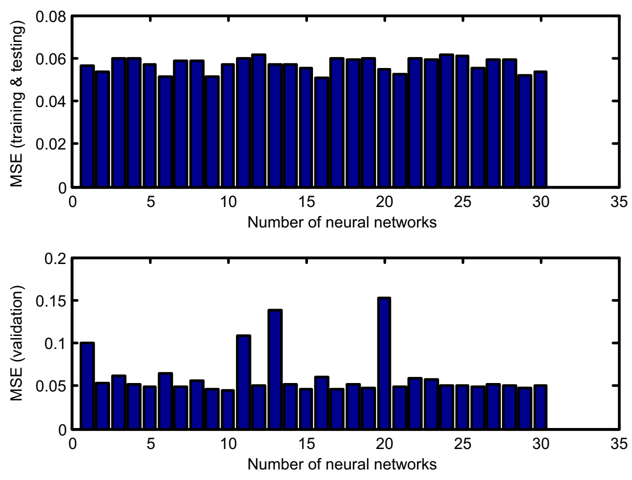

Fig. 5 MSE of CO_2 production rate for individual ELM models

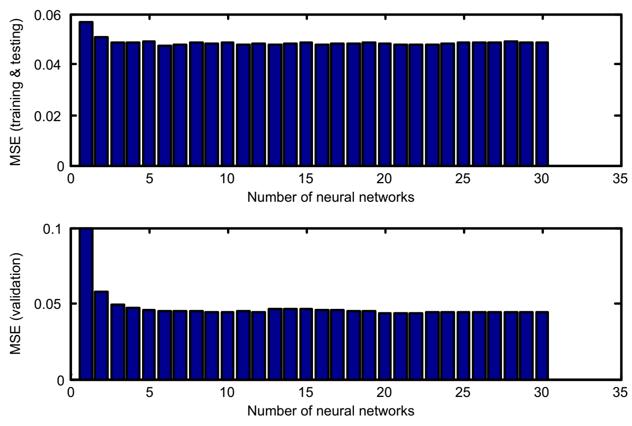

Fig. 6 MSE of CO_2 production rate for bootstrap aggregated ELM

the smallest mean squared errors (MSE) are considered as having the appropriate number of hidden neurons. The reason for ELM having more hidden neurons is due to the random nature of hidden layer weights in ELM and small number of hidden neurons would usually not be able to provide adequate function representation. The form of the dynamic model is shown in Eq. (15).

$$y(t) = f(y(t-1), u_1(t-1), u_2(t-1), \ldots, u_8(t-1)) \tag{15}$$

where y represents CO_2 capture level or CO_2 production rate, u_1 to u_8 are, respectively, inlet gas flow rate, CO_2 concentration in inlet flue gas, inlet gas temperature, inlet gas pressure, MEA circulation rate, lean loading, lean solution temperature, and reboiler temperature. Equation (15) represents a first order nonlinear dynamic model which is of the lowest order. For practical applications, model of the least complexity is generally preferred. If the low order nonlinear dynamic model could not give

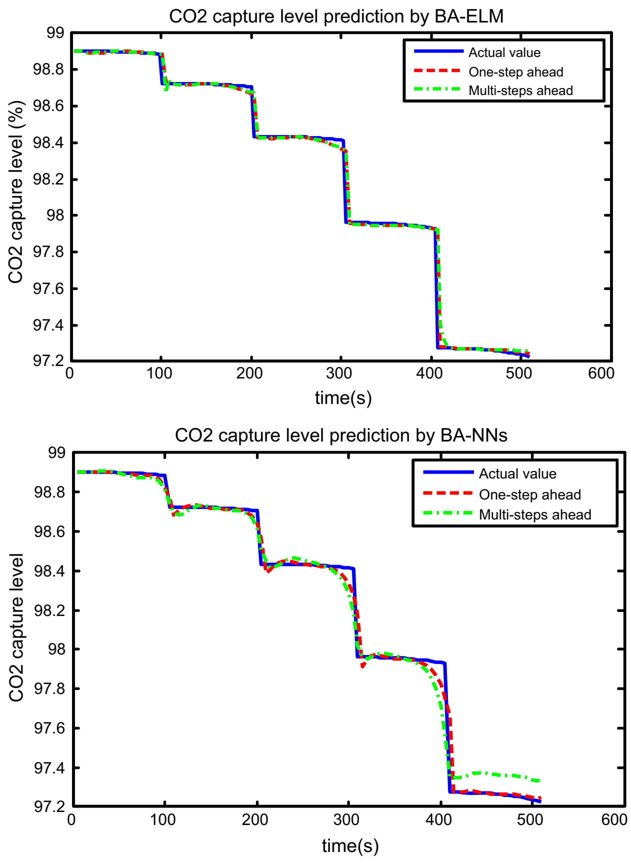

Fig. 7 Dynamic model prediction of CO_2 capture level using BA-ELM (*top*) and BA-NNs (*bottom*)

satisfactory performance, then higher order nonlinear dynamic models should be considered.

When developing the two different models, it is clearly seen that BA-ELM model is very simple because its training only needs one iteration. The performance comparison of the bootstrap aggregated neural networks and bootstrap aggregated ELM is shown in Table 1. The training CPU time of BA-ELM is about nine times lower than that of BA-NNs. The short training time of BA-ELM is due to the fact that each individual ELM is trained in one step without the need of gradient based iterative training. The verification time of BA-ELM is longer than that of BA-NN as the individual ELMs have more hidden neurons than the individual networks in BA-NN. The MSE value on the unseen validation data from BA-NNs is higher than that from BA-ELM. This could be due to the training of some neural networks in BA-NN might have been trapped in local minima or over fitted the noise. The results given in Table 1 demonstrate that BA-ELM is able to train faster and perform better than BA-NNs. The performance of one-step ahead predictions and multi-step ahead predictions of CO_2 production rate in BA-ELM and BA-NNs is indicated in Fig. 4. Clearly, the prediction using BA-ELM model is

Table 2 Performance comparison of BA-ELM and BA-NNs for CO_2 capture level

Learning algorithm	Time (CPU time) (s)		Training accuracy (MSE)	Validation accuracy (MSE)
	Training time	Verifying time (2nd batch)		
Bootstrap aggregated ELM (BA-ELM)	292.8919	0.8112	0.0034	0.00043
Bootstrap aggregated neural networks (BA-NNs)	1902.1	0.5148	0.0030	0.0015

much better than that using BA-NNs model, especially after 92 steps for the long range prediction.

The MSE values of CO_2 production rate for individual ELM models can be seen in Fig. 5. The performance on the unseen validation data is not in accordance with that on the training and testing data. For instance, the prediction on the unseen validation data by the 20th ELM is the worst, however, its performance on the training and testing data is better than many of the individual ELM models. This clearly demonstrates that single network has non-robust nature. Nevertheless, when several individual networks are combined together to build the model, the weakness can be addressed easily. Figure 6 indicates the MSE values on model building data by aggregating different numbers of ELM models. The first bar in Fig. 6 represents the first individual ELM model shown in Fig. 5, the second bar represents the combination of the first two individual ELM models, and the last bar represents combining all the individual ELM models. Look into the trends of top and bottom plots in Fig. 6, the prediction performance of bootstrap aggregated ELM on the unseen validation data is consistent with that on the training and testing data. In other words, combining several ELM models is able to get more accurate predictions on the training and testing data, as well as on the unseen validation data, than single ELM models. Furthermore, the MSE values in Fig. 6 indicates that, the aggregated ELM model provides more accurate predictions than single ELM models, when comparing with the MSE values in Fig. 5.

Figure 7 shows the performance comparison of one-step-ahead predictions and multi-step-ahead predictions of CO_2 capture level using BA-ELM and BA-NNs models. It is clear seen from the bottom graph both one-step-ahead predictions and multi-step-ahead predictions from BA-NN are reasonably accurate though some errors are observable, but the long range predictions (green line) are not accurate after 82 steps (410 s). However, in the top graph, the accurate one-step-ahead predictions and multi-step-ahead predictions from BA-ELM are very encouraging, indicating that the model has captured the underlying dynamics of the process. Such accurate long range predictions can be further used for model predictive control and real-time optimisation applications.

The performance comparison of the bootstrap aggregated neural networks and bootstrap aggregated ELM for CO_2 capture level is shown in Table 2. The training CPU time of BA-ELM is six times lower than that of BA-NN, while its verifying CPU time is a little bit longer than the latter one. This is because each network in the BA-ELM has more hidden neurons than each network in BA-NN. Looking into the comparison of the accuracy, the mean squared error (MSE) values on training data in both models are almost same, while the MSE value of BA-ELM on validation data is three times lower than that of BA-NNs. This shows that BA-ELM has a faster training speed and better generalization performance than BA-NNs, which has been proved in Huang et al. (2006). The faster training speed of BA-ELM is due to the ELMs are trained in a one-step procedure without the need of gradient based iterative procedure.

5 Conclusions

The BA-ELMs is demonstrated as a powerful tool to model the post-combustion CO_2 process, which can be trained much faster and is more accurate than the BA-NNs models. It gives a good generalization performance on unseen data, because the aggregation of multiple ELM can make the model avoid being trapped into local minima and overfitting problems. As ELM can be trained very quickly without iterative network weight updating, aggregating multiple ELMs does not pose any computational issues in model development. The model will be used to optimize the CO_2 capture process in the future. The model prediction confidence bounds provided by the BA-ELM can be incorporated in the optimisation objective function to enhance the reliability of the optimisation (Zhang 2004). Nevertheless, the BA-ELM still exits some problems. For instance, the number of hidden neurons is quite large, which may increase the model computation burden in optimisation studies. Further works on BA-ELM will be carried out to address these shortcomings.

Acknowledgements The work was supported by the EU through the project “Research and Development in Coal-fired Supercritical Power Plant with Post-combustion Carbon Capture using Process Systems Engineering techniques” (Project No. PIRSES-GA-2013-612230) and National Natural Science Foundation of China (61673236).

References

Biliyok C, Law A, Wang MH, Seibert F (2012) Dynamic modelling, validation and analysis of post-combustion chemical absorption CO_2 capture plant. Int J Greenhouse Gas Control 9:428–445

Cormos AM, Daraban IM (2015) Dynamic modelling and validation of amine-based CO_2 capture plant. Appl Therm Eng 74:202–209

Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489–501

Lawal A, Wang M, Stephenson P, Koumpouras G, Yeung H (2010) Dynamic modelling and analysis of post-combustion CO_2 chemical absorption process for coal-fired power plants. Fuel 89(10):2791–2801

Li F, Zhang J, Oko E, Wang MH (2015) Modelling of a post-combustion CO_2 capture process using neural networks. Fuel 151:156–163

Posch S, Haider M (2013) Dynamic modelling of CO_2 absorption from coal-fired power plants into an aqueous monoethanolamine solution. Chem Eng Res Des 91(6):977–987

Sipocz NTF, Assadi M (2011) The use of artificial neural network models for CO_2 capture plants. Apply Energy 88(7):2368–2376

Wold S (1978) Cross validatory estimation of the number of components in factor and principal components models. Technometrics 20:397–404

Zhang J (1999) Developing robust non-linear models through bootstrap aggregated neural networks. Neurocomputing 25:93–113

Zhang J (2004) A reliable neural network model based optimal control strategy for a batch polymerisation reactor. Ind Eng Chem Res 43(4):1030–1038

Zhou Q, Chan CW, Tontiwachiwuthikul P, Idem R, Gelowitz D (2009) A statistical analysis of the carbon dioxide capture process. Int J Greenhouse Gas Control 3(5):535–544

Zhou Q, Wu YX, Chan CW, Tontiwachiwuthikul P (2010) Applications of three data analysis techniques for modelling the carbon dioxide capture process. In: 2010 23rd Canadian conference on electrical and computer engineering (Ccece)