International Journal of Automation and Computing 17(1), February 2020, 44-54
DOI: 10.1007/s11633-019-1203-x

Developing Soft Sensors for Polymer Melt Index in an Industrial Polymerization Process Using Deep Belief Networks

Chang-Hao Zhu Jie Zhang

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

Abstract: This paper presents developing soft sensors for polymer melt index in an industrial polymerization process by using deep belief network (DBN). The important quality variable melt index of polypropylene is hard to measure in industrial processes. Lack of online measurement instruments becomes a problem in polymer quality control. One effective solution is to use soft sensors to estimate the quality variables from process data. In recent years, deep learning has achieved many successful applications in image classification and speech recognition. DBN as one novel technique has strong generalization capability to model complex dynamic processes due to its deep architecture. It can meet the demand of modelling accuracy when applied to actual processes. Compared to the conventional neural networks, the training of DBN contains a supervised training phase and an unsupervised training phase. To mine the valuable information from process data, DBN can be trained by the process data without existing labels in an unsupervised training phase to improve the performance of estimation. Selection of DBN structure is investigated in the paper. The modelling results achieved by DBN and feedforward neural networks are compared in this paper. It is shown that the DBN models give very accurate estimations of the polymer melt index.

Keywords: Polymer melt index, soft sensor, deep learning, deep belief network (DBN), unsupervised training.

1 Introduction

Much work on soft sensors in the research area of process control has done in the past few decades. This technique is widely implemented in industrial chemical processes. Soft sensors are very effective techniques for the estimation of some important product quality variables in industrial processes which cannot be measured effectively. In the area of process control, the issues of hardware instruments, such as unavailability or high cost, hinder the product quality control. To overcome these issues, empirical models can be developed based on process operational data obtained from real industrial processes. With such models, the difficult-to-measure quality variables can be estimated from easy-to-measure process variables[1]. This modelling technique based on historical process data has become increasingly popular in chemical processes in recent years. Such data driven models can be effectively used to reduce the cost of production in industrial processes and improve the efficiency.

Much successful research on process modelling based on multivariate statistical techniques has been completed in the last century. In 1901, principal component analysis (PCA) was proposed by Pearson[2]. This method was further developed by Harold Hotelling in the 1930s[3, 4]. Based on PCA, principal component regression (PCR) and partial least squares (PLS) have emerged as useful modelling methods to address the problem of co-linearity among the input variables[2]. Data-driven soft sensors based on PCR can be developed by using principal components as the predictor variables. As an improvement of PCR, PLS regression can model both the process data and quality data at the same time[5]. Wold et al.[6] first introduced PLS and Wold further developed it. There were many applications based on the PLS technique in process modelling. One limitation of PLS and PCR is that they are both linear techniques. They are not very effective when applied to nonlinear process modelling.

With the development of machine learning, many researches on developing soft sensors based on machine learning techniques have been reported in the past a few years. There are many successful process modelling techniques based on machine learning, such as support vector machine (SVM) and artificial neural networks (ANN). McCulloch and Pitts[7] proposed the original neural network in the 1940s. After 20 years, with the vast improvement of computer capability, neural networks became a popular research topic. The back-propagation algorithm was applied to ANN by Werbos[8] in 1975. The advantage of ANN is that they can be used to approximate any

Research Article
Special Issue on Improving Productivity Through Automation and Computing
Manuscript received March 15, 2019; accepted September 18, 2019; published online November 5, 2019
Recommended by Associate Editor Xian-Dong Ma

nonlinear functions. ANN gives very good performance on estimation and prediction of quality data. The back-propagation algorithm can deal with the exclusive-or problems. In the backpropagation training algorithm, the network weights between neurons are modified to distribute the errors back up from the output layer[8]. However, conventional ANN suffers from problems of local optima and lack of generalization capability. SVM can achieve accessible optima of training even when there is little training data[9]. However, when applying SVM to processes with large amount of modelling data, the pressure of computation will increase. In 2006, Hinton et al.[10] first introduced deep learning. Deep belief network (DBN) is one kind of the most well-known data-driven modelling techniques based on deep learning. It shows strong generalization capability in modelling highly nonlinear processes. This model is established with a deep architecture. Deep learning has many applications in speech recognition and images classification[11]. There are two training phases in the procedure of DBN training: unsupervised training followed by supervised training. Before supervised training, DBN will capture more information from nonlinear process input data to achieve more accurate prediction or estimation of quality data. It has shown significant performance in many other applications[12, 13].

Soft sensors for polymer melt index (MI) are established using DBN and applied to an industrial polypropylene polymerization process in this study. By using deep learning techniques, large amount of industrial process data samples without pre-existing labels can also be used by DBN models in the unsupervised training phase. However, these input data are useless for training the conventional feed-forward neural networks which just use supervised training. These process data samples help the DBN model in adjusting the weights in a desirable region. The information from process data were captured during the procedure of unsupervised training. It is shown in this paper that DBN models gave very accurate estimations of MI.

The rest of this paper is organized as follows. An introduction of ANN is given in Section 2. In Section 3, DBN model and the main principles of restricted Boltzmann machines (RBMs) and back-propagation are introduced. Section 4 introduces the case study of an industrial polypropylene polymerization process. The selection of DBN model architectures are discussed and the polymer melt index estimation results are given in Section 5. Section 6 summarizes the conclusions of this paper.

2 Artificial neural networks

The feed-forward neural network is one of the most well-known machine learning techniques. It can be used in solving many problems of prediction, classification, and pattern recognition. Much research of ANN has been reported in the past decades. In the initial form of simple perceptron invented by McCulloch and Pitts, the model calculates the weighted sum of input variables and then passes it to an activation function. Fig. 1 shows a simple perceptron structure.

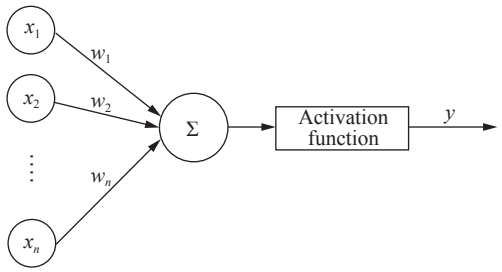

Fig. 1 Model of simple perceptron

As can be seen from Fig. 1, x_i, $i = 1, 2, \cdots, n$, are input variables and w_j, $j = 1, 2, \cdots, n$, are the corresponding weights for these input variables. McCulloch and Pitts[7] used the threshold function as the activation function. They proved universal computations can be performed by simple perceptrons if weights are chosen appropriately. However, a lot of complicated systems cannot be represented by this method[14]. Many other activation functions can be used, such as Heaviside step function, sigmoid function and Gaussian function. These activation functions are sometimes also named as transfer function in ANN research. The most popular activation function is the sigmoid function.

The characteristic of the sigmoid function is that it is an "S"-shaped curve as shown in Fig. 2.

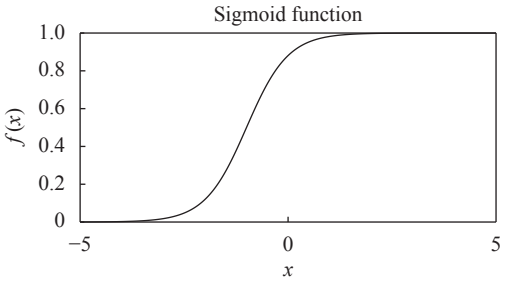

Fig. 2 "S"-shaped curve of sigmoid function

The sigmoid function maps its input values into a region from 0 to 1. In Fig. 2, the output value approaches to 1 when x approaches to $+\infty$, whereas the output approaches to 0 when x approaches to $-\infty$. It has the appropriate asymptotic properties. The sigmoid function is given by (1):

$$S(x) = \frac{1}{1 + \mathrm{e}^{-\beta x}} \tag{1}$$

where x represents the sum of weighted input values and β is a slope parameter.

The structure of ANN can be regarded as neurons ar-

ranged into inter-connected layers with weighted connections between neurons in adjacent layers. Basically, feedforward neural networks and recurrent networks are principally two types of ANN. Feed-forward networks have no feedback connections from the network outputs. Recurrent neural networks are a type of neural networks with feedback connections. In this work, feed-forward networks are also used for soft sensor development in the polypropylene polymerization process for the purpose of comparison with DBN.

Multilayer perceptrons are the most classic type of feed-forward networks. They can deal with more complicated problems than simple perceptrons. Neurons in the adjacent layers are connected unidirectionally without feedback loops. A multilayer perceptron model has at least three layers. It is commonly constructed by an input layer, a hidden layer and an output layer. The relationship between the neural network input and output variables can be learnt during the process of supervised training and stored as the trained network weights. The structure of a multilayer perceptron with two hidden layers is shown in Fig. 3.

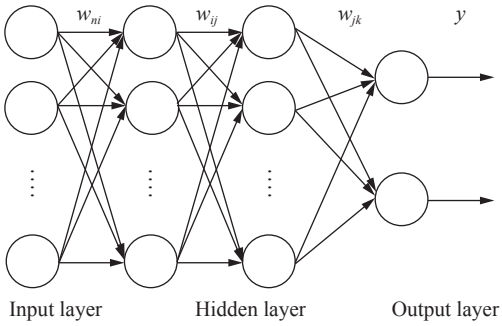

Fig. 3 Multilayer perceptron

Each unit in the input layer is a network input. The output of a unit in the hidden or output layer is calculated by passing the sum of the weighted outputs of the previous layer to an activation function as follows:

$$O_j = f\left(\sum_{i=1}^{n} w_{ij} I_i + b_j\right) \tag{2}$$

where O_j is the output value of the unit j in a particular layer, w_{ij} is the weight between this unit and the i-th unit of the immediate previous layer, I_i is the i-th input of this unit (i.e., the output value of the i-th unit in the previous layer), b_j is a bias, and f is the activation function. During network training, the weights and biases will be initialized as random values typically in a range between –0.1 and 0.1. Network weights are adjusted by using training algorithms to minimize the error terms between network outputs and target labels. After training, the relationship between system input variables and output variables can be represented by the trained neural networks.

The training process of a multilayer feedforward neural network is supervised training. The most commonly used supervised training algorithm is the backpropagation algorithm. Multilayer feedforward neural networks have the capability of modeling nonlinear processes. However, the process of polymerization is highly nonlinear. The structure of commonly used multilayer neural networks is shallow. When a feedforward neural network with more than three layers is training by backpropagation, the model always suffers from the problem of poor generalization. This modelling technique cannot meet the demand of the accuracy of the estimation. To achieve more accurate estimation of MI, DBN models are established in this study. DBN has a deep architecture and stronger generalization capability.

3 Deep belief networks

3.1 Structure of deep belief network

The limitation of traditional neural networks is that they usually have shallow structures. There are typically no more than three layers in a conventional neural network model. With this limitation, a neural network with shallow structure may not achieve satisfactory estimation performance when applied to highly nonlinear industrial processes. The actual industrial processes are commonly highly nonlinear. The shallow architecture of feedforward neural networks could lead to the lack of representation capability[15, 16]. To approximate various regions of processes, the model needs more hidden neurons added to the hidden layers. It is suggested in recent research that networks with a deep structure can achieve reliable results[15]. DBN has been successfully applied to many research areas, such as classification and recognition[17]. 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 a deep learning technique. Fig. 4 presents the basic architecture of DBN.

The DBN shown in Fig. 4 has five layers, an input layer, an output layer and three hidden layers. In Fig. 4, W is the weight of the network, $\boldsymbol{b}$ and $\boldsymbol{c}$ are biases of the network. 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 the traditional Boltzmann machine, the neurons in a hidden layer of DBN are not connected to each other. However, the layers in a network have symmetrical connections with 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 mine more correlations than the feed-forward neural networks. 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.

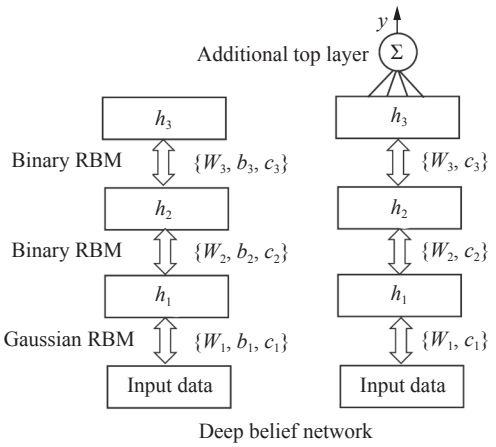

Fig. 4 Architecture of DBN

3.2 Restricted Boltzmann machines

In the 1980s, Smolensky[18] developed the restricted Boltzmann machine. Hinton et al.[10] developed DBN by stacking RBMs as the layers of DBN. A DBN contains stacked RBMs as shown in Fig. 4.

To understand the basics of RBM, the probability function between visible units and hidden units need to be introduced at first. Equation (3) shows the probability function

$$P(\boldsymbol{v},\boldsymbol{h})=\frac{\exp\{-Energy(\boldsymbol{v},\boldsymbol{h})\}}{Z} \tag{3}$$

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

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

where $\boldsymbol{W}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ are parameters of the function. It should be noted 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, (4) can be extended to energy function of Gaussian RBM:

$$Energy(\boldsymbol{v},\boldsymbol{h})=\sum_i\frac{(v_i-a_i)^2}{2\sigma_i^2}-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{h}-\boldsymbol{h}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{v} \tag{5}$$

where a_i is the mean of Gaussian distribution, σ_i is the standard deviation of Gaussian distribution for input neuron. The samples of input data are commonly normalized to zero mean and unit variance in practical applications. Therefore, (5) can be changed to

$$Energy(\boldsymbol{v},\boldsymbol{h})=\frac{1}{2}\boldsymbol{v}^{\mathrm{T}}\boldsymbol{v}-\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v}-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{h}-\boldsymbol{h}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{v}. \tag{6}$$

Hintons[19] also described other forms of RBM, but the DBN in this paper only uses Gaussian RBM and binary RBM.

3.3 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 $\boldsymbol{v}$ can be formulated as

$$\begin{aligned}\frac{\partial\log P(\boldsymbol{v})}{\partial\theta}&=\frac{\partial\log\sum_h P(\boldsymbol{v},\boldsymbol{h})}{\partial\theta}=\\&\frac{\sum_h \mathrm{e}^{-Energy(\boldsymbol{v},\boldsymbol{h})}\left(\frac{\partial[-Energy(\boldsymbol{v},\boldsymbol{h})]}{\partial\theta}\right)}{\sum_h \mathrm{e}^{-Energy(\boldsymbol{v},\boldsymbol{h})}}-\\&\frac{\sum_{\tilde{\boldsymbol{v}}}\sum_{\boldsymbol{h}} \mathrm{e}^{-Energy(\tilde{\boldsymbol{v}},\boldsymbol{h})}\left(\frac{\partial[-Energy(\tilde{\boldsymbol{v}},\boldsymbol{h})]}{\partial\theta}\right)}{\sum_{\tilde{\boldsymbol{v}}}\sum_{\boldsymbol{h}} \mathrm{e}^{-Energy(\tilde{\boldsymbol{v}},\boldsymbol{h})}}=\\&\sum_h P(\boldsymbol{h}|\boldsymbol{v})\frac{\partial[-Energy(\boldsymbol{v},\boldsymbol{h})]}{\partial\theta}-\\&\sum_{\tilde{v}}\sum_h P(\tilde{\boldsymbol{v}},\boldsymbol{h})\frac{\partial[-Energy(\tilde{\boldsymbol{v}},\boldsymbol{h})]}{\partial\theta}\end{aligned} \tag{7}$$

where $\theta=\{\boldsymbol{W},\boldsymbol{b},\boldsymbol{c}\}$ is a vector of the network parameters. Computing the positive term in (7) is easy because the vector $\boldsymbol{v}$ has been known. Calculating the negative term in (7) becomes intractable. 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[20, 21]. The process of training RBM starts with training vectors on the visible units. Then hidden units $\boldsymbol{h}^{(\boldsymbol{t})}$ can be generated from $\boldsymbol{v}^{(\boldsymbol{t-1})}$ by Gibbs sampling and update visible units $\boldsymbol{v}^{(\boldsymbol{t})}$ from $\boldsymbol{h}^{(\boldsymbol{t})}$. It is named as a Markov chain. After infinite 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 situations, just one iteration of Gibbs sampling can achieve a satisfactory result and the learning algorithm works well.

3.4 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 back-propagation 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 back propagated to the previous layer. Equation (8) shows the error terms

$$\boldsymbol{Err}_j = \boldsymbol{O}_j\left(1-\boldsymbol{O}_j\right)\left(\boldsymbol{T}_j-\boldsymbol{O}_j\right) \tag{8}$$

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

$$\boldsymbol{Err}_j = \boldsymbol{O}_j\left(1-\boldsymbol{O}_j\right)\sum_k \boldsymbol{Err}_k \boldsymbol{w}_{jk} \tag{9}$$

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

$$\boldsymbol{w}_{ij} = \boldsymbol{w}_{ij} + \eta \boldsymbol{Err}_j \boldsymbol{O}_i \tag{10}$$

$$\boldsymbol{c}_j = \boldsymbol{c}_j + \eta \boldsymbol{Err}_j \tag{11}$$

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 model with a deep architecture.

4 Polypropylene polymerization process

Advanced monitoring, control, and optimization techniques are essential in modern industrial chemical processes to overcome the issue of high cost and improve the efficiency of production[22]. In this paper, DBN is used to develop soft sensors for a polypropylene production plant in China. In this plant, two continuous stirred tank reactors (CSTR) and two fluidized-bed reactors (FBR) are used to produce polypropylene as shown in Fig. 5. Propylene, hydrogen, and catalyst are fed to reactors. Reactants for the growing polymer particles are these gases and liquids fed to the reactors. They are also the provider of the heat transfer media. The melt index of polymer is a key polymer quality variable and should be closely monitored and controlled. MI of polypropylene depends on many factors like catalyst, reactor temperature and concentration of the reaction materials. For example, hydrogen can increase the polymerization rate of polypropylene. It mainly increases the initial polymerization rate of propylene[23]. The hydrogen concentration regulates the weight of polypropylene. Hydrogen can also delay the decay rate of the catalyst. Due to the difficulty of measuring polymer MI in this process, the relationship between MI and some process variables which can be measured easily during the process needs to be found. The inferential estimation of MI can be obtained by soft sensors. As this industrial process is very complicated, it is difficult to develop first principle models linking polymer MI with easy-to-measure process variables. Therefore, nonlinear data-driven models need to be utilised in developing soft sensors for this process.

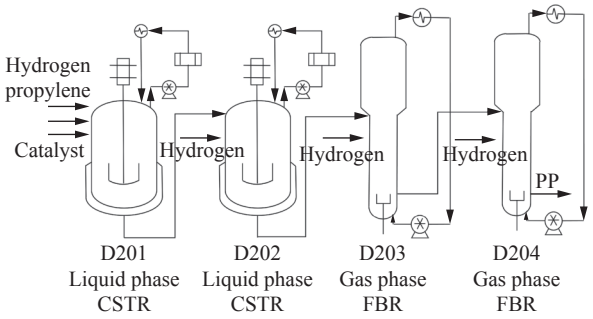

Fig. 5 Propylene polymerization process

The polypropylene grades are related to some key variables, such as reactant composition, reactor temperature and catalyst properties. The feedstock of D201 are propylene, hydrogen and catalyst. The co-monomer is added to D204. Several grades of polymers were produced within one month. Industrial process operational data covering this time period are available for this application. In this process, polymer MI were logged every two hours and process samples were logged every half hour. In fact, MI are only highly relevant to a few process variables. Based on the research of Zhang et al.[24], there are strong correlations between MI of polymer in reactor D204 and hydrogen concentration in reactor D201 and reactor D202. MI of polymer in reactor D201 is highly relevant with the hydrogen concentration and feed rate in reactor D201[24]. Concentration of hydrogen in D201 and D202, feed rate of hydrogen and MI of polypropylene in reactor D201 and D204 are shown in Figs. 6–8, respectively. Due to the industrial confidentiality, the units of these variables are not disclosed.

From Fig. 8, it can be observed that the MI data cover quite a wide range. Thus, the data are suitable for de-

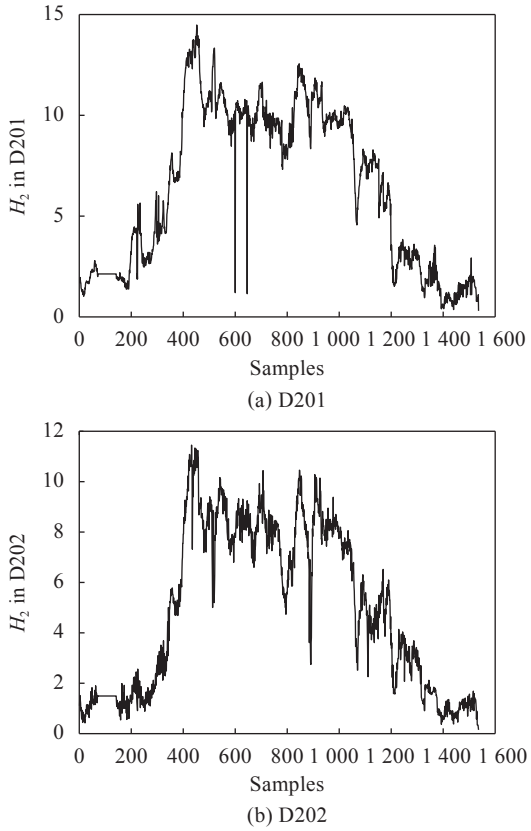

Fig. 6 Concentration of hydrogen in (a) D201 and (b) D202

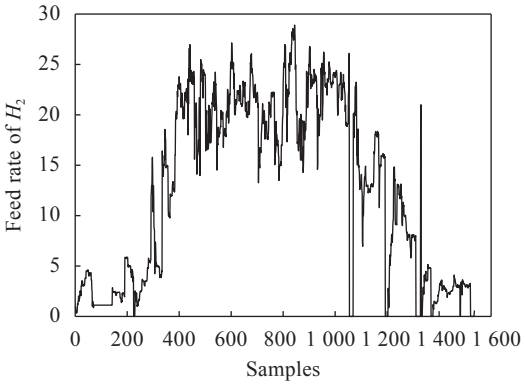

Fig. 7 Feed rate of hydrogen

veloping data-driven models. Soft sensors should extract the information from limited process data and quality data to obtain accurate estimation of MI. From the trends displayed in Figs. 6–8, it can be seen that MI is highly correlated with hydrogen feed rate and concentration.

The time delay of the industrial process can be found based on the cross-correlations analysis[24]. The data-driven model for inferential estimation of MI can be represented as

$$MI_1(t) = f_1[H_1(t), H_1(t-1), H_1(t-2), F(t-9), F(t-10), F(t-11)] \tag{12}$$

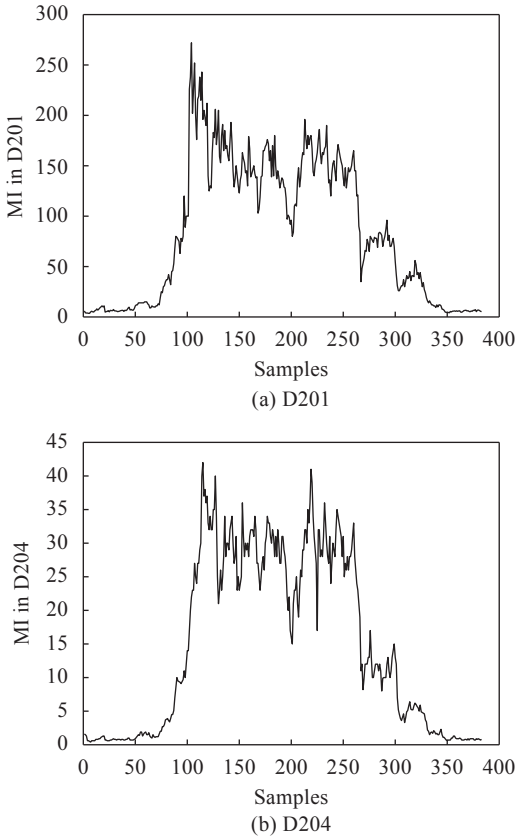

Fig. 8 Melt index in (a) D201 and (b) D204

$$MI_2(t) = f_2[H_1(t-7), H_1(t-8), H_1(t-9), H_2(t-6), H_2(t-7), H_2(t-8)] \tag{13}$$

where MI_1 and MI_2 are the MI in D201 and D204, respectively, H_1 and H_2 are the concentrations of hydrogen in D201 and D202, respectively, and F is the hydrogen feed rate to D201.

The original process data set contains 1 534 samples of process operational data and 383 samples of quality data (MI) which are available for the establishment of data driven DBN models. It indicates that the amount of process variable samples is larger than the amount of quality variable samples. There are only 383 samples of process variables that have corresponding quality variables. However, the rest of process variable samples can be utilized by DBN in the unsupervised training phase. By such means, DBN can capture much valuable information from process data. The estimation of MI achieved by DBN can be improved.

The data set for supervised training phase were separated into a training data set, a testing data set and an unseen validation data set. The partition of data sets for estimating MI_1 is presented by Table 1. The partition of data sets for estimating MI_2 is presented by Table 2.

The selections of model structure can be determined by the training data set and testing data set through cross-validation. The unseen validation data are useful to test the performance of the final developed DBN model.

Table 1 Partition of data sets for estimating MI_1

Data sets	Percentage	Number of samples
Training data	50%	192
Testing data	22%	85
Unseen validation data	28%	106

Table 2 Partition of data sets for estimating MI_2

Data sets	Percentage	Number of samples
Training data	52%	200
Testing data	18%	68
Unseen validation data	30%	115

It can be seen from Tables 1 and 2, 277 samples of training and testing variables were selected to fine tune DBN by backpropagation for MI_1 and DBN only use 268 samples of training and testing variables to fine tune DBN in supervised training phase for MI_2. During the unsupervised training phase of DBN models, only input data are required and target values are not required. Those input data samples without the corresponding output data are named as "unlabeled" process data. Therefore, in the unsupervised training phase of DBN models, samples of process variables without the corresponding MI data can also be utilized. However, those "unlabeled" process variables could not be used by other traditional neural networks for inferential estimation of product quality. For comparison, conventional neural network models were also developed.

5 Results and discussions

The model structures need to be determined first. In this study, 25 DBN models with different architectures were developed and compared to each other. The one giving the best performance on the testing data set was regarded as having the appropriate structure. These DBN models have one visible layer (input layer), one additional top layer (output layer) and two hidden layers. The learning rate in the unsupervised training phase is selected as 0.01. The learning rate in the supervised training phase is 0.001 5, The structures of 25 DBN models are shown in Table 3. Figs. 9 and 10 present the sum of squared errors (SSE) on the training data set and testing data set, respectively for these 25 DBN models for estimating MI_1.

From Figs. 9 and 10, the 7th DBN model gives the best generalization performance on the testing data set. The 6th DBN model gives the second lowest value of error on the testing data set. The 12th to the 25th DBN models have lower training errors than the 7th DBN model. However, those models give larger testing errors than the 7th DBN model. Thus, the 12th to the 25th DBN models are likely have suffered from overfitting and

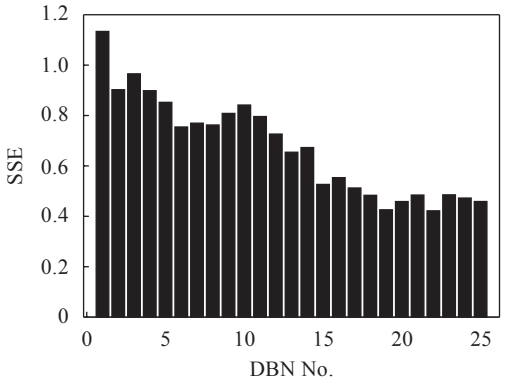

Fig. 9 SSE on training data for estimating MI_1

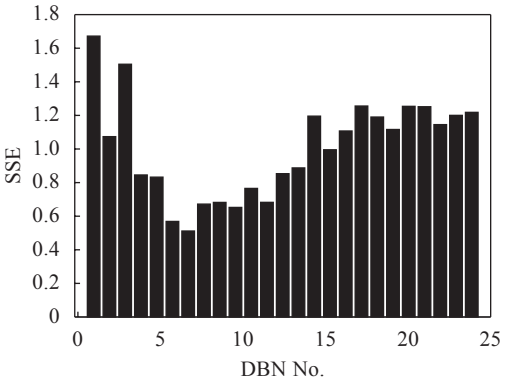

Fig. 10 SSE on testing data for estimating MI_1

their structures are not appropriate to be selected. From the results given by Figs. 9 and 10, the number of neurons in the first hidden layer can be considered as 5. From Table 3, it can be seen that these 25 DBN models have close numbers of neurons in the first and second hidden layers. The first step of this investigation is to confirm that the 7th DBN gave the best performance among these 25 DBN models. To avoid the situation that some DBN models not included in Table 3 might give better performance, the second step is to further investigate the number of neurons in the second hidden layer. Nine additional DBN models with neurons in the second hidden layer ranging from 2 to 10 are constructed. The values of error terms on the training and testing data of these DBN models are shown in Table 4.

From Table 4, it can be seen that the training error of the 7th DBN is the smallest. However, its testing error is not the smallest. The testing errors from the 6th to the 9th DBN increased. Therefore, the estimation on testing data obtained by the 6th to the 9th DBNs are overfitted. The 4th DBN (i.e., the 7th DBN model in Table 3) has the lowest testing error among all DBN models. This indicates that the 4th DBN model has better performance than other models and its structure should be adopted.

In order to demonstrate the advantage of using those input data samples without corresponding target values as additional training data in the unsupervised training

Table 3 DBN models with different structures

No.	Neurons in 1st hidden layer	Neurons in 2nd hidden layer	No.	Neurons in 1st hidden layer	Neurons in 2nd hidden layer
1	2	1	14	8	7
2	2	2	15	9	9
3	3	3	16	9	8
4	3	2	17	10	10
5	4	4	18	10	9
6	4	3	19	11	11
7	5	5	20	11	10
8	5	4	21	12	12
9	6	6	22	12	11
10	6	5	23	13	13
11	7	7	24	13	12
12	7	6	25	14	13
13	8	8			

Table 4 Errors of DBN models with different structures for estimating MI_1

No.	Neurons in 1st hidden layer	Neurons in 2nd hidden layer	SSE (training)	SSE (testing)
1	5	2	0.7562	0.5819
2	5	3	0.8204	0.6193
3	5	4	0.7824	0.5945
4	5	5	0.7696	0.5118
5	5	6	0.8206	0.5773
6	5	7	0.7271	0.5742
7	5	8	0.6723	0.5859
8	5	9	0.7628	0.6071
9	5	10	0.7372	0.6322

phase, a DBN model trained only using the input data samples with the pre-existing labels in the unsupervised training phase was also developed. This is represented by DBN No. 1 in Table 5, where DBN No. 2 was built by using "unlabeled" process data without corresponding MI samples as well. DBN No. 2 in Table 5 is in fact the 4th DBN model in Table 4. The two DBN models in Table 5 have the same structure. It can be seen from Table 5 that the first DBN model has larger SSE values on the training, testing and validation data set than the second DBN model. Therefore, DBN can extract more features from the "unlabeled" data. DBN No. 2 gives better performance than DBN No. 1.

Seven conventional single hidden layer feedforward neural network models were also established for the purpose of comparison. The SSE values of these conventional feedforward neural networks with different structures on the training and testing data are given in Table 6. From Table 6, the 4th neural network has the lowest SSE on the testing data set for estimating MI_1 and the 3rd neural network has the lowest SSE on the testing data for

Table 5 Errors of DBN models for estimating MI_1 with different input data

DBN No.	SSE (training)	SSE (testing)	SSE (validation)
1	1.6203	0.8905	0.7024
2	0.7696	0.5118	0.6851

Table 6 Errors of neural networks with different structures

No.	Neurons in hidden layer	MI_1		MI_2	
		SSE (training)	SSE (testing)	SSE (training)	SSE (testing)
1	2	1.3256	0.7446	1.6025	0.6855
2	3	0.7949	0.8221	1.5185	0.7374
3	4	0.7924	0.6527	1.5035	0.6564
4	5	0.7675	0.6323	1.3650	0.6883
5	6	0.6347	0.6532	1.1009	0.8214
6	7	0.5124	1.1054	0.7844	0.8305
7	8	0.4201	0.8895	0.5108	1.6024

estimating MI_2.

The estimations of MI_1 on the unseen validation data by DBN and the conventional feedforward neural network are shown in Fig. 11. In Fig. 11, the solid, dashed, and dotted lines represent, respectively, the actual values of MI_1, the estimations by DBN, and the estimations by the conventional feedforward neural network. It can be seen from Fig. 11 that the estimations by the DBN model are generally closer to the corresponding actual values of MI_1 than those by the feedforward neural network. The SSE values of both DBN and neural network are presented in Table 7. It can be seen from Table 7 that the SSE of DBN on training data set is larger than that of the neural network. However, the SSE values of DBN on testing and unseen validation data set are much smaller than those of the neural network. The strong generalization

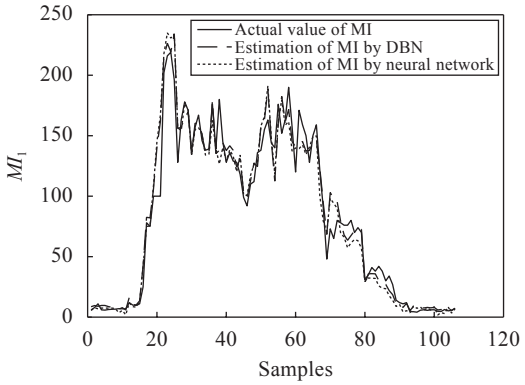

Fig. 11 Estimation of MI_1 by DBN and neural network

Table 7 SSE of estimating MI_1

Models	SSE (training)	SSE (testing)	SSE (validation)
Neural network	0.7675	0.6323	0.8243
DBN	0.7696	0.5118	0.6851

capability of DBN was proved by the inferential estimation of MI_1. It gave better performance than the feed-forward neural network. The profuse latent information from process data were extracted by DBN during the unsupervised training phase. Overall, the DBN model gives more accurate estimations of MI_1.

Fig. 12 compares the estimations of MI_2 by DBN and conventional feedforward neural network on the unseen validation data. In Fig. 12, the solid, dashed, and dotted lines represent, respectively, the actual values of MI_2, the estimations by DBN, and the estimations by the conventional feedforward neural network. From Fig. 12, it can be seen that both models give similar performance when MI values are high. However, when MI values are low, the DBN model gives better estimations. Table 8 shows the SSE values in the estimation of MI_2. The SSE of DBN on training data is larger than that of neural network. The SSE values of DBN on testing and unseen validation data set are much smaller than those of the neural network model. The results in Fig. 12 and Table 8 indicate that

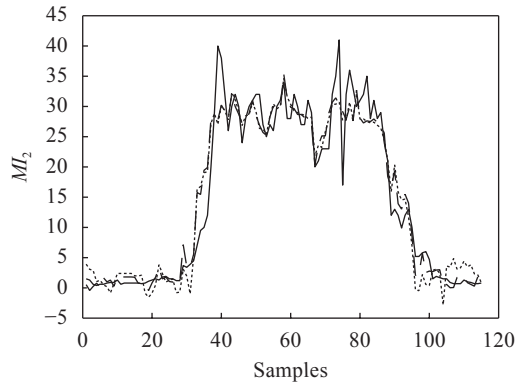

Fig. 12 Estimation of MI_2 by DBN and neural network

Table 8 SSE of estimating MI_2

Models	SSE (training)	SSE (testing)	SSE (validation)
Neural network	1.503 5	0.656 4	0.991 5
DBN	1.517 0	0.434 2	0.856 0

the estimations of MI_2 achieved by DBN are more reliable and accurate than those from the conventional feedforward neural network.

6 Conclusions

DBN models for the on-line inferential estimation of the polymer melt index in an industrial polymerization process are developed in this paper. DBN can be developed with a deep structure. The profuse latent information from the process variables can be extracted by DBN. The "unlabeled" process data, which ware useless to the conventional neural network models, can be used in the unsupervised training stage of DBN. It is shown in this paper that the accuracy of inferential estimation of polymer MI can be improved by this means. Selection of DBN structure is investigated in the paper. The appropriate structures of DBN for the estimation of MI_1 and MI_2 are selected. DBN has much better performance compared with the results from conventional feedforward neural networks. The study demonstrates that DBN is very suitable for developing nonlinear data-driven models for the inferential estimation of polymer melt index. The proposed DBN model could be extended for developing multi-step ahead prediction models in the future. The network structure of DBN can be further optimized to improve the robustness.

Acknowledgments

The work was supported by National Natural Science Foundation of China (No. 61673236) and the European Union (No. PIRSES-GA-2013-612230).

Open Access

References

[1] M. T. Tham, G. A. Montague, A. J. Morris, P. A. Lant. Soft-sensors for process estimation and inferential control. *Journal of Process Control*, vol. 1, no. 1, pp. 3–14, 1991. DOI: 10.1016/0959-1524(91)87002-F.

[2] K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901. DOI: 10.1080/14786440109462720.

[3] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, vol. 24, no. 6, pp. 417–441, 1933. DOI: 10.1037/h0071325.

[4] H. Hotelling. Relations between two sets of variates. *Breakthroughs in Statistics: Methodology and Distribution*, S. Kotz, N. L. Johnson, Eds., New York, USA: Springer, pp. 321–377, 1992. DOI: 10.1007/978-1-4612-4380-9_14.

[5] H. Wold. Estimation of principal components and related models by iterative least squares. *Multivariate Analysis*, P. R. Krishnaiah, Ed., New York, USA: Academic Press, pp. 391–420, 1966.

[6] S. Wold, M. Sjöström, L. Eriksson. PLS-regression: A basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems*, vol. 58, no. 2, pp. 109–130, 2001. DOI: 10.1016/S0169-7439(01)00155-1.

[7] W. S. McCulloch, W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943. DOI: 10.1007/BF02478259.

[8] P. Werbos. Beyond Regression: New Fools for Prediction and Analysis in the Behavioral Sciences, Ph. D. dissertation, Harvard University, Boston, USA, 1974.

[9] K. Desai, Y. Badhe, S. S. Tambe, B. D. Kulkarni. Soft-sensor development for fed-batch bioreactors using support vector regression. *Biochemical Engineering Journal*, vol. 27, no. 3, pp. 225–239, 2006. DOI: 10.1016/j.bej.2005.08.002.

[10] G. E. Hinton, S. Osindero, Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006. DOI: 10.1162/neco.2006.18.7.1527.

[11] A. Mnih, G. E. Hinton. A scalable hierarchical distributed language model. In *Proceedings of the 21st International Conference on Neural Information Processing Systems*, Curran Associates Inc., Vancouver, Canada, pp. 1081–1088, 2009.

[12] F. Li, J. Zhang, C. Shang, D. X. Huang, E. Oko, M. H. Wang. Modelling of a post-combustion CO_2 capture process using deep belief network. *Applied Thermal Engineering*, vol. 130, pp. 997–1003, 2018. DOI: 10.1016/j.applthermaleng.2017.11.078.

[13] Z. J. Yao, J. Bi, Y. X. Chen. Applying deep learning to individual and community health monitoring data: A survey. *International Journal of Automation and Computing*, vol. 15, no. 6, pp. 643–655, 2018. DOI: 10.1007/s11633-018-1136-9.

[14] A. K. Jain, J. C. Mao, K. M. Mohiuddin. Artificial neural networks: A tutorial. *Computer*, vol. 29, no. 3, pp. 31–44, 1996. DOI: 10.1109/2.485891.

[15] Y. Bengio, O. Delalleau, N. Le Roux. The curse of highly variable functions for local kernel machines. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, MIT Press, Vancouver, Canada, pp. 107–114, 2006.

[16] Y. Bengio, Y. LeCun. Scaling learning algorithms towards AI. *Large-scale Kernel Machines*, L. Bottou, O. Chapelle, D. DeCoste, J. Weston, Eds., Cambridge, USA: MIT Press, pp. 1–41, 2007.

[17] Y. C. Tang, R. Salakhutdinov, G. Hinton. Robust Boltzmann machines for recognition and denoising. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Providence, USA, pp. 2264–2271, 2012.

[18] P. Smolensky. Information Processing in Dynamical Systems: Foundations of Harmony Theory, Technical Report CU-CS-321-86, University of Colorado, Boulder, USA, 1986.

[19] G. E. Hinton. A practical guide to training restricted Boltzmann machines. *Neural Networks: Tricks of the Trade*, G. Montavon, G. B. Orr, K. R. Müller, Eds., Berlin Heidelberg, Germany: Springer, pp. 599-619, 2012. DOI: 10.1007/978-3-642-35289-8_32.

[20] M. Á. Carreira-Perpiñán, G. E. Hinton. On contrastive divergence learning. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, The Society for Artificial Intelligence and Statistics, Barbados, pp. 33–40, 2005.

[21] C. Shang, F. Yang, D. X. Huang, W. X. Lyu. Data-driven soft sensor development based on deep learning technique. *Journal of Process Control*, vol. 24, no. 3, pp. 223–233, 2014. DOI: 10.1016/j.jprocont.2014.01.012.

[22] S. Z. Gao, X. F. Wu, L. L. Luan, J. S. Wang, G. C. Wang. PSO optimal control of model-free adaptive control for PVC polymerization process. *International Journal of Automation and Computing*, vol. 15, no. 4, pp. 482–491, 2018. DOI: 10.1007/s11633-016-0973-7.

[23] J. B. P. Soares, A. E. Hamielec. 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*, vol. 37, no. 20, pp. 4607–4614, 1996. DOI: 10.1016/0032-3861(96)00286-8.

[24] J. Zhang, Q. Jin, Y. Xu. Inferential estimation of polymer melt index using sequentially trained bootstrap aggregated neural networks. *Chemical Engineering & Technology*, vol. 29, no. 4, pp. 442–448, 2006. DOI: 10.1002/ceat.200500352.

Chang-Hao Zhu received the B. Sc. degree in mechanical engineering from Shandong University, China in 2014, and the M. Sc. degree in electrical power from Newcastle University, UK in 2016. Currently, he is a Ph. D. degree candidate in chemical engineering at the School of Engineering, Newcastle University, UK.

His research interests include process control, machine learning, development of data driven soft sensor and their applications industrial chemical processes.

E-mail: c.zhu5@newcastle.ac.uk

ORCID iD: 0000-0003-1801-0787

Jie Zhang received the B. Sc. degree in control engineering from Hebei University of Technology, China in 1986, and the Ph. D. degree in control engineering from City University, UK in 1991. He is a Reader in the School of Engineering, Newcastle University, UK. He has published over 300 papers in international journals, books and conferences. He is a senior member of IEEE, a member of the IEEE Control Systems Society, IEEE Computational Intelligence Society, and IEEE Industrial Electronics Society. He is on the Editorial Boards of a number of journals including *Neurocomputing* published by Elsevier.

His research interests are in the general areas of process system engineering including process modelling, batch process control, process monitoring, and computational intelligence.

E-mail: jie.zhang@newcastle.ac.uk (Corresponding author)

ORCID iD: 0000-0002-9745-664X