Journal of Process Control 24 (2014) 223–233

Contents lists available at ScienceDirect

Journal of Process Control

journal homepage: www.elsevier.com/locate/jprocont

Data-driven soft sensor development based on deep learning technique

Chao Shang[a,b], Fan Yang[a,b], Dexian Huang[a,b,*], Wenxiang Lyu[a,b]

[a] Department of Automation, Tsinghua University, Beijing 100084, China
[b] Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, China

ARTICLE INFO

Article history:
Received 8 April 2013
Received in revised form 6 January 2014
Accepted 26 January 2014
Available online 16 February 2014

Keywords:
Deep neural network
Nonlinear regression
Soft sensor
Data-driven technique

ABSTRACT

In industrial process control, some product qualities and key variables are always difficult to measure online due to technical or economic limitations. As an effective solution, data-driven soft sensors provide stable and reliable online estimation of these variables based on historical measurements of easy-to-measure process variables. Deep learning, as a novel training strategy for deep neural networks, has recently become a popular data-driven approach in the area of machine learning. In the present study, the deep learning technique is employed to build soft sensors and applied to an industrial case to estimate the heavy diesel 95% cut point of a crude distillation unit (CDU). The comparison of modeling results demonstrates that the deep learning technique is especially suitable for soft sensor modeling because of the following advantages over traditional methods. First, with a complex multi-layer structure, the deep neural network is able to contain richer information and yield improved representation ability compared with traditional data-driven models. Second, deep neural networks are established as latent variable models that help to describe highly correlated process variables. Third, the deep learning is semi-supervised so that all available process data can be utilized. Fourth, the deep learning technique is particularly efficient dealing with massive data in practice.

1. Introduction

Soft sensors have been extensively studied and implemented in the process industries over the past two decades. Typically, they are predictive models based on massive amounts of data available in the industrial processes, and are mainly responsible for online predictions of some variables that play an indispensable role in quality control as well as production safety, for the reason that hardware measuring instruments are unavailable or costly [1]. In general, one can broadly classify soft sensors into two types, namely, the first-principle models (white-box models) and data-driven models (black-box models). First-principle models are dependent on a prior mechanical knowledge and thus often unavailable since industrial processes are commonly too complicated to analyze, making the mechanical knowledge rather hard-won. Their data-driven counterparts, alternatively, give empirical models based on the historical data collected in the industrial process. Owing to their practical utility and independence on a priori knowledge, data-driven soft sensors have increasingly established themselves as popular and effective approaches [2,3]. A wide variety of statistical inference techniques and machine learning techniques have been employed in data-driven soft sensors, among which representative examples are *principal component regression* (PCR) that incorporates *principal component analysis* (PCA) with a regression model, *partial least squares* (PLS) regression, *support vector machine* (SVM), and *artificial neural network* (ANN).

There is one challenging issue for soft sensors that in real industrial scenario, processes are characterized by strongly correlated process variables. Usually, the number of such process variables as seen is much larger than its effective dimension, which is termed as *data rich but information poor* [4]. In such case, *latent variable* models are particularly suitable for describing low-dimensional subspaces with few loss of information, which give explanations to major variations in the process data. The PCR and PLS approaches are the most popular techniques handling data correlations in the process industries. Based on the PCA, the PCR [5] makes an effective tool with the co-linearity

Abbreviations: PCA, principal component analysis; PLS, partial least squares; SVM, support vector machine; ANN, artificial neural network; MLP, multi-layer perceptron; RBFN, radial basis function network; RBM, restricted Boltzmann machine; DBN, deep belief network; CDU, crude distillation unit; CD, contrastive divergence; NNPLS, neural network partial least squares; ASTM, American Society for Testing Materials; DCS, distributed control system.

* Corresponding author at: Department of Automation, Tsinghua University, Beijing 100084, China. Tel.: +86 10 62784964; fax: +86 10 62786911.
E-mail address: huangdx@tsinghua.edu.cn (D. Huang).

0959-1524/$ – see front matter © 2014 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jprocont.2014.01.012

problem. It first finds *principal components* (PCs) of the high dimensional process variables (X space) and then establishes a regression mapping from the PCs to a desired target variable. However, PCR fails to reflect the relationship between the process variables (X space) and the target variable (Y space) when establishing latent variables. On the contrary, the PLS technique [6] can model both X and Y spaces concurrently, which has found wide applications in soft sensor modeling. Nevertheless, both the pure PCR and pure PLS share several common drawbacks: (1) vast amounts of data are needed to generalize well; and (2) they possess a linear prototype and their nonlinear extensions [7–9] such as *neural network PLS* (NNPLS) and *kernel PLS* (KPLS) suffer from the difficulty in selections of nonlinear parameters.

In recent years, there has been a proliferation of research that applies machine learning approaches to soft sensors. The most-used ones are SVM and ANN. Formulated as a convex quadratic optimization problem, SVM [10] enjoys advantages of low computational costs and accessible optima. Since proposed, it has been gradually become a major method in many fields, such as artificial intelligence, pattern recognition and machine learning. In addition, it proves effective to tackle problems with small samples, and thus it has been extensively used as soft sensors [11,12]. However, there exists a remaining problem that the computational complexity grows exponentially with the number of training samples. On the other hand, ANN has held its own space in function approximation and pattern recognition. Overall, ANN represents a large class of model structures. The most common types are *multi-layer perceptron* (MLP) and *radial basis function networks* (RBFN). Qin [13] has pointed out that ANN is especially suitable for developing soft sensors and it indeed has been widely used [14–16]. However, ANN still suffers from an uncontrolled convergence speed and local optima. In addition, parameters of neural networks with deep structures (more than two layers) are difficult to optimize using traditional gradient descent. Despite some successful applications, both SVM and ANN provide no allowance for the latent variable subspace, leading to a weak interpretation capability.

Resembling the multi-level structure of the human brain, deep neural networks *should* also belong to the scope of ANN, but they have been of no use until in 2006 machine learning practitioners proposed the *deep learning* technique, a novel data-driven approach to training of deep neural networks. Since then, the deep learning has gained increasing attentions in machine learning and has motivated a plenty of successful applications in speech and image recognition [17,18] and natural language processing [19], where deep learning outperforms other traditional data-driven learning methods like PCR, SVM, and shallow ANN due to its remarkable representation ability. A detailed introduction to deep learning can be found in [20]. A deep network is trained by means of pre-training *deep belief network* (DBN) that in essence makes a nonlinear latent variable model being interpretable, and a regression model is established on the basis of the acquired latent variable model, making it suitable for handling highly correlated data. Moreover, unsupervised and supervised learning are properly integrated [21], yielding a semi-supervised model; this is just what traditional ones are not good at. Up to now, however, applications of deep learning have not been found in data-driven soft sensor modeling. To this end, this paper employs advantages of evolving deep learning techniques to deal with the abundant data and inherent changing nature of industrial processes for the development of soft sensors.

The remainder of this article proceeds as follows. In Section 2, drawbacks of shallow networks and advantages of deep learning as well as characteristics of industrial processes are overviewed, and the goodness that deep learning well fits soft sensor modeling is further demonstrated. Fundamentals of deep learning technique are detailed in Section 3. The modeling procedure of deep neural network based soft sensors is outlined in Section 4. An industrial case study about estimation of the heavy diesel cut point in the crude-oil distillation unit is employed in Section 5 to demonstrate the efficacy of the proposed approach. The final section gives concluding remarks.

2. Overview of deep learning and its goodness of fit for soft sensor modeling

2.1. Remarkable representation ability of deep neural networks and limitations of traditional models

A great majority of traditional models, such as SVM, conventional MLP and RBFN, are able to approximate any continuous nonlinear mapping to arbitrary precision; however, they are considered to have *shallow architectures*, that is, with less than three layers of computation units. For example, the MLP with a single hidden layer consists of two layers of neurons; the SVM can be seen as a two-layer network, while the type of the kernel determines the number of units in the first layer as well as connecting configurations in the second layer. Recent research has shown that, with an inadequate depth of layers, these networks lack a powerful representation efficacy, and reveal limitations on some learning tasks consequently [22]. Typical difficulties are incurred when approximating "highly varying functions" that vary violently in some regions. In order to well approximate these highly varying regions, numerous units should be added to the shallow architectures [23]. Meanwhile, sufficient training samples in highly varying regions are required to guarantee desirable generalization. If the training samples are scarce, the highly varying functions are not to be properly represented by networks with shallow structure. Recent analysis, however, suggested that a highly varying function can be properly represented by a *deep architecture*, that is, with more than two layers of nonlinearities [22]. Nevertheless, the common gradient-descent optimization with random initial configurations is no good any longer because it is often stuck into poor local optima, and deep architectures have therefore always been a hot potato for training. A remarkable breakthrough was reported by [24] that a novel semi-supervised training algorithm, termed as deep learning, was proposed.

In the process industries, chemical devices are commonly composed of a wide variety of sub-systems. Meanwhile, biochemical reactions are usually highly varying in different configurations of constitutes. Obviously, the complicated chemical processes can be better described by deep hierarchical structures.

2.2. Unique advantages of deep learning: nonlinear latent structures

The deep learning technique mainly includes two phases, namely the unsupervised pre-training phase and the supervised back-propagation phase. In the unsupervised pre-training phase, a *deep belief network* (DBN) is pre-trained as initial weights of the subsequent supervised phase, and in the supervised back-propagation phase, the whole network is fine-tuned in a supervised manner.

Here a detailed introduction to DBN is given and its unique advantages are analyzed. It will be shown that the DBN regards the deep neural network as a latent variable model, which is beneficial for soft sensor modeling. The DBN adopts a multi-layer structure including one visible layer and several latent layers. In this study the number of layers of the DBN is defined as L. A DBN is hierarchically built by stacking a series of *restricted Boltzmann machines* (RBMs), as shown in Fig. 1. Each layer in a DBN is seen as an individual RBM. There are

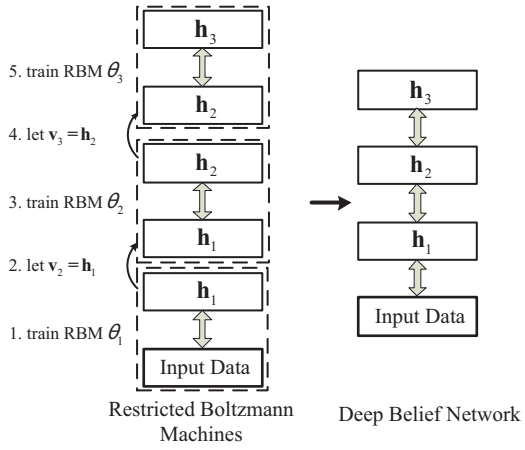

Fig. 1. Deep belief network structure.

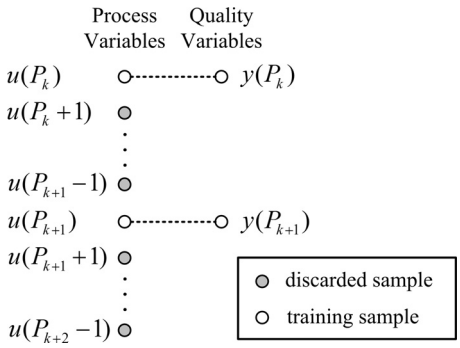

Fig. 2. Multi-rate sampling in soft sensor modeling. Process data discarded by traditional methods (gray dots) can be used for unsupervised training of deep neural networks, and the training sample used by traditional methods (white dots) can be used for supervised back-propagation.

two layers in an individual RBM $\theta_l (l = 1, 2, \ldots, L)$, namely the visible layer representing inputs $\mathbf{v}_l$ and the hidden layer representing latent variable $\mathbf{h}_l$, and the latent variable serve as the inputs of the next RBM θ_{l+1}. With hidden units $\mathbf{h}$ introduced, the RBM enjoys a latent variable model type. The training process of DBN involves a greedy layer-wise scheme from lower layers to higher layers. Here this process is illustrated by a simple example of a three-layer RBM. In Fig. 1, RBM θ_1 is trained first, and the hidden layer of the previous RBM is taken as the inputs of RBM θ_2, and then RBM θ_2 is trained, and next the RBM θ_3 can be accomplished in the same successive manner. During the training progress of each RBM, latent variables as features are individually extracted from its inputs by maximizing the probability $P(\mathbf{v}_l)$. Notice that the DBN training process is totally unsupervised as no target variable is involved. It can be understood that high-level latent variables are learnt on the basis of low-level latent variables, which make DBN a desired latent variable model allowing for powerful interpretation.

It is known that evident correlations (inherent features) exist between the chemical process variables and hence latent models are desired, such as some data-driven methods – the PCR, PLS and their nonlinear extensions – as aforementioned. Several recent results have revealed that deep neural networks help to mine much more complicated correlations than PCA and PLS, and have the capability of building low-dimensional nonlinear latent structures [25]; thus it is likely to utilize deep learning to capture correlations between process variables with latent variable models, making the soft sensor model more interpretable.

2.3. A semi-supervised strategy to incorporate all available process data

Another reliable reason for deep learning to be employed is that fast-sampled process data can be fully utilized. In chemical processes, the sampling rate of quality variable of interest can be extremely slow as compared with process variables, as shown in Fig. 2. Consequently, the number of quality samples is much less than that of process samples. For traditional soft sensor models like PLS, SVM and ANN, however, equal numbers of process samples and quality samples are used. Therefore, only a small number of process samples are taken into consideration and the rest abundant fast-rate process samples containing profuse information remain unused. In deep learning, these process data abandoned by previous methods can be fortunately used for unsupervised pre-training to extract explicit latent variables, facilitating the supervised back-propagation with target quality variable and their corresponding process samples. It is therefore plausible that with more data used, more accurate models would be attained. Therefore, it is worth attempting to apply the deep learning technique to soft sensor modeling.

3. Two-stage procedure of developing regression models with DBN

This section details the two-stage procedure of developing regression models using deep neural networks. The first step is pre-training, in which a DBN is trained solely with unsupervised input data. The second step is back-propagation, in which the regression neural network is trained with target data using the parameters attained in the first step for initialization. Section 3.1 introduces basics of RBM that acts as layer component of DBN. In Section 3.2, the gradient descent based learning algorithm for each individual RBM is clarified. Section 3.3 presents the supervised learning to obtain the regression model using target data.

3.1. Basics of RBM

First we introduce the basic energy-based models, in which each configuration of variables is assigned to a scalar value of energy. Let $\mathbf{v}$ denote the input vector of the RBM (visible layer), and then the probability of distribution through an energy function can be defined as:

$$P(\mathbf{v}) = \frac{\exp\{-\text{Energy}(\mathbf{v})\}}{Z}, \tag{1}$$

where $Z = \sum_{\mathbf{v}} \exp\{-\text{Energy}(\mathbf{v})\}$ is the normalizing factor. Notice that a lower energy is preferred to maximize the probability $P(\mathbf{v})$. By introducing a vector of hidden layer $\mathbf{h}$, the RBM framework is formulated as:

$$P(\mathbf{v}, \mathbf{h}) = \frac{\exp\{-\text{Energy}(\mathbf{v}, \mathbf{h})\}}{Z}. \tag{2}$$

Different energy-based models have different forms of energy function. In an RBM, the energy function is simplified as a second-order polynomial:

$$\text{Energy}(\mathbf{v}, \mathbf{h}) = -\mathbf{b}^{\mathrm{T}}\mathbf{v} - \mathbf{c}^{\mathrm{T}}\mathbf{h} - \mathbf{h}^{\mathrm{T}}\mathbf{W}\mathbf{v}, \tag{3}$$

where $\mathbf{v}$ and $\mathbf{h}$ are both vectors of binary values (0 or 1), and $\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{c}\}$ are parameters of the energy function. Such RBMs are called *binary* RBMs, and they are most-used ones in practical applications. Binary RBMs, however, only have the capability to deal with discrete inputs. In order to tackle continuous-valued inputs, binary RBMs are extended to *Gaussian* RBMs as follows [26]:

$$\text{Energy}(\mathbf{v}, \mathbf{h}) = \sum_i \frac{(v_i - a_i)^2}{2\sigma_i^2} - \mathbf{c}^{\mathrm{T}}\mathbf{h} - \mathbf{h}^{\mathrm{T}}\mathbf{W}\mathbf{v}, \tag{4}$$

where a_i and σ_i are the mean and the standard deviation of the Gaussian distribution for visible unit i. Here the input layer $\mathbf{v}$ is continuous-valued and hidden layer $\mathbf{h}$ is binary. In (4), $\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{c}\}$ as well as a_i and σ_i are parameters to be learnt in the training procedure. In practical use, however, each component of the input data is commonly normalized to zero mean and unit variance, so that Gaussian RBMs are simplified as normalized Gaussian RBMs:

$$\text{Energy}(\mathbf{v}, \mathbf{h}) = \frac{1}{2}\mathbf{v}^{\mathrm{T}}\mathbf{v} - \mathbf{b}^{\mathrm{T}}\mathbf{v} - \mathbf{c}^{\mathrm{T}}\mathbf{h} - \mathbf{h}^{\mathrm{T}}\mathbf{W}\mathbf{v}. \tag{5}$$

In the following, the normalized Gaussian unit will be discussed and thus (5) will be employed instead of (4).

Notice that with hidden layer $\mathbf{h}$ introduced, the RBM is of a latent variable model structure. There are many other types of RBMs that can be seen in [27], such as soft-max RBMs and rectified linear RBMs. In this study only binary units and Gaussian RBMs are used. Appendix A gives some conditional probabilities for RBM units that are of unique use in the RBM training procedure.

3.2. Gradient descent based learning algorithm for RBM

For RBM models, only the visible layer $\mathbf{v}$ is at hand and the hidden layer $\mathbf{h}$ is to be estimated. Therefore, it is reasonable to set the training objective of RBM as maximizing $P(\mathbf{v})$, which is the probability of the model simply on the training input data. Optimization is typically performed by the gradient descent state-of-the-art. By (2) and total probability formula, the gradient of the log-likelihood function at a single data point $\mathbf{v}$ is calculated as:

$$\begin{aligned}\frac{\partial \log P(\mathbf{v})}{\partial\theta} &= \frac{\partial \log\sum_{\mathbf{h}} P(\mathbf{v},\mathbf{h})}{\partial\theta} = \frac{\sum_{\mathbf{h}} \mathrm{e}^{-\text{Energy}(\mathbf{v},\mathbf{h})}\cdot(\partial[-\text{Energy}(\mathbf{v},\mathbf{h})]/\partial\theta)}{\sum_{\mathbf{h}} \mathrm{e}^{-\text{Energy}(\mathbf{v},\mathbf{h})}} - \frac{\sum_{\tilde{\mathbf{v}}}\sum_{\mathbf{h}} \mathrm{e}^{-\text{Energy}(\tilde{\mathbf{v}},\mathbf{h})}\cdot(\partial[-\text{Energy}(\tilde{\mathbf{v}},\mathbf{h})]/\partial\theta)}{\sum_{\tilde{\mathbf{v}}}\sum_{\mathbf{h}} \mathrm{e}^{-\text{Energy}(\tilde{\mathbf{v}},\mathbf{h})}} \\ &= \underbrace{\sum_h P(\mathbf{h}|\mathbf{v})\cdot\frac{\partial[-\text{Energy}(\mathbf{v},\mathbf{h})]}{\partial\theta}}_{\text{positive term}} - \underbrace{\sum_{\tilde{\mathbf{v}}}\sum_{\mathbf{h}} P(\tilde{\mathbf{v}},\mathbf{h})\cdot\frac{\partial[-\text{Energy}(\tilde{\mathbf{v}},\mathbf{h})]}{\partial\theta}}_{\text{negative term}},\end{aligned} \tag{6}$$

where $\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{c}\}$. Two terms in Eq. (6) are defined: the positive term represents the conditional expectation of $\partial[-\text{Energy}(\mathbf{v}, \mathbf{h})]/\partial\theta$ given the visible units $\mathbf{v}$, and the negative term represents the expectation of $\partial[-\text{Energy}(\mathbf{v}, \mathbf{h})]/\partial\theta$ of the joint distribution $P(\mathbf{v},\mathbf{h})$ with parameter θ. Note that $\partial[-\text{Energy}(\mathbf{v}, \mathbf{h})]/\partial\theta$ is easy to compute, making the positive term available with $P(\mathbf{h}|\mathbf{v})$ given in Appendix A. However, for RBMs with numerous units, computing the negative term by means of sampling becomes thoroughly intractable. As an effective solution, the *contrastive divergence* (CD) approximates the negative term by means of sampling approaches [28]. The details of the CD algorithm are given in Appendix B.

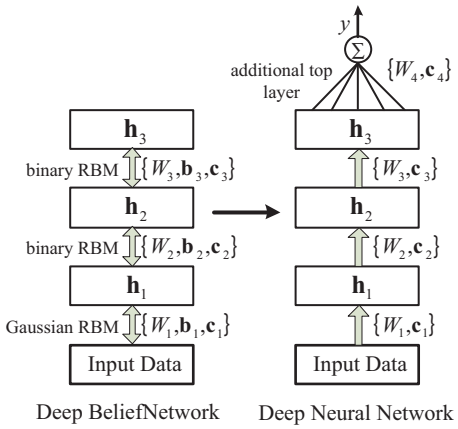

Fig. 3. Initializing a deep sigmoid network with 3 hidden layers after pre-training a DBN.

3.3. Training deep neural networks with a pre-trained DBN by back-propagation

As aforementioned in Section 2, stacking a series of RBMs constructs a DBN. Notice that the hidden layer in RBM describes the underlying features behind the input data. The hidden layers of high-level RBMs represent the high-level features because high-level RBMs are established based on the low-level features from the hidden layer of low-lever RBMs. Therefore, the DBN can be seen as a latent variable model with inner layer representing low-level features and output amounting to high-level features. In soft sensing modeling, because the process data are continuous and are not limited to a certain range, the bottom RBM is selected as the Gaussian unit as desired, and the rest are selected as binary units. Fig. 3 depicts the structure of DBN used in the present study.

Then, after unsupervised training phase, parameters $\{W_l, \mathbf{c}_l\}$ $(l = 1, \ldots, L-1)$ of the DBN are derived from $L-1$ well learnt RBMs. Once the unsupervised training step for DBN is accomplished, the whole neural network is to be trained with target value in a supervised manner. Then weights of the L-layer deep neural network are initialized as follows: parameters $\{W_l, \mathbf{c}_l\}$ $(l = 1, \ldots, L-1)$ (except the top layer parameters) are set the same as the DBN, and the top layer weights $\{W_L, \mathbf{c}_L\}$ are initialized stochastically. After that, the whole network can be fine-tuned by back-propagation in a supervised way using process data and corresponding quality data. Fig. 3 presents a schematic illustration on initializing a 4-layer deep neural network with a pre-trained DBN as well as an additional top layer.

The entire training procedure can be explained as follows. Training deep neural networks always yields poor results unless the initial weights are in desirable regions. Pre-training a DBN helps to find latent variables behind the data and set the weights adjacent to a good solution so that back-propagation becomes effective for training deep neural networks. In addition, because the neural network is established based on latent variables in DBN, the entire neural network can be seen as keeping some latent variable information and more interpretable.

4. Soft sensor modeling based on deep neural networks

In summary, the procedure of soft sensor modeling based on deep neural networks includes following steps:

Step 1: Select the secondary variables according to the process knowledge.
Step 2: Preprocess the data. First remove all potential outliers from the original samples, and then perform normalization so that all samples are zero mean and unit variance.
Step 3: Determine the architecture of the network, and pre-train the DBN in an unsupervised way by training each individual RBM successively.
Step 4: Initialize the parameters of the deep neural network with the pre-trained DBN.
Step 5: Fine-tune the deep neural network with target variable by back-propagation.
Step 6: Validate the deep network by an additional dataset. If there exist over-fitting, go back to Step 5.
Step 7: Test the soft sensor model in a test dataset. If dissatisfactory, go back to Step 3.

5. Industrial case study: crude-oil distillation unit

This section presents the development of a soft sensor based on deep neural networks in a petroleum refinery process. The soft sensor is primarily responsible for the real-time prediction of product quality with special interests, and its performance is compared to other traditional data-driven modeling approaches.

5.1. Case description

The *crude distillation unit* (CDU) is commonly regarded as one of the most essential production process in petrochemical industry. It usually consists of an atmospheric distillation unit and a vacuum distillation unit. The atmospheric distillation unit usually consists of a crude tower, a condenser and a stripper, as shown in Fig. 4. The CDU is a necessary step through which crude is first processed after entering a refinery. After desalted to remove impurities and preheated to a requisite temperature, crude is partitioned in CDU into different fractions, and typical products from CDU include naphtha, kerosene, light diesel and heavy diesel. Their boiling/flash/pour points are key

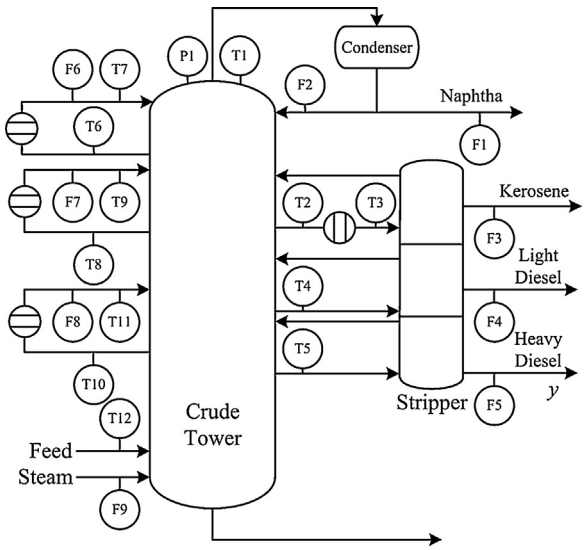

Fig. 4. Sketch of an atmospheric distillation unit and selection of process variables.

indices that are highly relevant for oil quality as well as yield of product, thereby having a great impact on the efficiency and cost of the entire process. Stringent requirement of petroleum product quality as a consequence impels the need that such quality indices should be monitored and controlled at all times. In common, there are two direct measuring approaches for these quality indices in real production scenarios. One approach is to use offline laboratory analysis at intervals of 8–24 h, not only involving a large time delay but also asking for massive manual effort. The other approach is to use online analyzers; however, it is usually expensive and suffers from a high malfunction probability. For these two reasons online soft sensors, with much less design effort and maintenance cost, are in extremely urgent need to provide real-time and reliable predictions for quality control purposes in petroleum refinery process.

Here a deep neural network based soft sensor is applied to the online quality prediction of the American Society for Testing Materials (ASTM) 95% cut point temperature of heavy diesel, which is the primary controlled variable in CDU. In this case study, all modeling data and test data come from the real process data that are measured and recorded in a petroleum refinery plant in northern China. In this plant, the quality variable, namely the ASTM 95% cut point of heavy diesel, is sampled by manual laboratory analysis at either 12:00 or 20:00 every day, in a typically irregular manner. It is to be additionally noted that the laboratory analysis usually takes more than one hour to execute so that it can no longer act as the feedback signal for proper quality control. The process variables, such as temperatures, pressures and material flows, are sampled and collected by the *distributed control system* (DCS) of the plant, the sampling interval of which is only one second.

5.2. Settings for soft sensing modeling

5.2.1. Variable selection

The process variables that are used as the inputs of soft sensors are selected according to a physical background. As shown in Fig. 4, there are 12 temperature variables (T1–T12), 10 flow rate variables (F1–F10), and one pressure variable (P1), in total 23 process variables directly collected by DCS.

First, the ASTM 95% cut point index is heavily influenced by the heat exchange within the crude tower. According to the energy balance within the crude tower, the change of heat can be calculated as the product of the corresponding flow and temperature difference. For example, the top recycle heat (No. 20 in Table 1) can be approximated as the production of recycle flow rate (F6) and the temperature difference in top recycles (T7–T6). Therefore, three heat values (Nos. 20, 25, and 30 in Table 1) are inferred based on the relevant flow and temperature values.

Then, it is known that the crude is partitioned into various products with different components at a certain ratio, and the composition of product determines its quality index. In the CDU process, each side-drawn flow rate as well as the recycle flow rate must be proportionally changed to the feed flow rate so that the composition of product would remain unchanged, thereby keeping the quality stable. For this reason, the quality index is profoundly affected by the ratio between each flow rate (F1–F9 in Table 1) and the feed flow rate (F10 in Table 1), and in the present study these ratio-transformed variables are selected as process variables (Nos. 4, 6, 10, 13, 16, and 34 in Table 1). Similarly, the recycle heat is in proportion to the corresponding flow rate, and thus should be divided by the feed flow rate to be selected. Both direct-measured variables and transformed variables are tabulated in Table 1, among which a collection of 16 variables are selected as process variables for soft sensor modeling (see column 4 in Table 1).

5.2.2. Selection of the historical data

In the present study, a dataset containing 351 process/quality samples, corresponding to a recent year of process operation, is selected. The dataset is separated into a training set with 251 samples and a test set with 100 samples. In addition, the unsupervised pre-training stage typically incurs in the development of deep neural networks. Therefore, special care should be taken for selection of the unsupervised data, which simply require a large number of process samples and no longer need the quality sample as regression targets. In this regard, 1724 samples of 'unlabeled' process variables are used for the unsupervised training of the DBN. It is noted that these unsupervised data cannot be utilized by traditional data-driven models.

Table 1
Process variables to be selected for soft sensors.

No.	Tag	Explanatory variables	Selected variables with prior knowledge	Selected variables without prior knowledge
1	P1	Top pressure	√	√
2	T1	Top temperature	√	√
3	F1	Top flow rate		√
4	F1/F10	Relative top flow rate	√	
5	F2	Reflux flow rate		√
6	F2/F10	Relative reflux flow rate	√	
7	T2	Side-drawn 1# tray temperature	√	√
8	T3	Side-drawn 1# reboiling temperature	√	√
9	F3	Side-drawn 1# flow rate		√
10	F3/F10	Relative side-drawn 1# flow rate	√	
11	T4	Side-drawn 2# tray temperature	√	√
12	F4	Side-drawn 2# flow rate		√
13	F4/F10	Relative side-drawn 2# flow rate	√	
14	T5	Side-drawn 3# tray temperature	√	√
15	F5	Side-drawn 3# flow rate		√
16	F5/F10	Relative side-drawn 3# flow rate	√	
17	T6	Top recycle drawn temperature		
18	T7	Top recycle returned temperature		
19	F6	Top recycle flow rate		
20	F6 × (T7 − T6)	Top recycle heat		√
21	F6 × (T7 − T6)/F10	Relative top recycle heat	√	
22	T8	Middle recycle 1# drawn temperature		
23	T9	Middle recycle 1# returned temperature		
24	F7	Middle recycle 1# flow rate		
25	F7 × (T9 − T8)	Middle recycle 1# heat		√
26	F7 × (T9 − T8)/F10	Relative middle recycle 1# heat	√	
27	T10	Middle recycle 2# drawn temperature		
28	T11	Middle recycle 2# returned temperature		
29	F8	Middle recycle 2# flow rate		
30	F8 × (T11 − T10)	Middle recycle 2# heat		√
31	F8 × (T11 − T10)/F10	Relative middle recycle 2# heat	√	
32	T12	Feed temperature	√	√
33	F9	Steam flow rate		√
34	F9/F10	Relative steam flow rate	√	
35	F10	Feed flow rate		√

The performance of data-driven soft sensors may heavily deteriorate because of the outliers that industrial processes commonly involve. In practical applications, because of sensor malfunction, communication exception or database shutdown, collected data may stick to a fixed value for some time, or fall out of the normal operating range unexpectedly, thereby being unreliable for modeling. The acquisition of such a normal range usually requires a priori knowledge elicited from process industry practitioners. In this study, 11 samples have been first pruned out accordingly. Then outliers are detected with the three-sigma rule. Five samples that violate the constraint $|x_i(k) - \bar{x}_i| < 3\sigma_i$ are considered as outliers and further removed, where $\bar{x}_i$ denotes the mean of data of ith variable and σ_i denotes the corresponding standard deviation. Finally the process variables are normalized to zero mean and unit variance by the following linear transformation:

$$x_i^*(k) = \frac{x_i(k) - \bar{x}_i}{\sigma_i}. \tag{7}$$

5.2.3. Settings of deep neural network based soft sensor

A deep 16-20-16-1 neural network is built for quality prediction, and the network architecture is selected by cross-validation. To avid over-fitting, 100 samples in the training set are used for validation in the back-propagation step, while other 151 samples are used for gradient descent. If the validation error begins to increase, it indicates over-fitting occurs and back-propagation should be stopped. Due to the randomness in neural network learning, the training procedure is commonly repeated many times and the result with least validation error is selected. In this study, the training procedure is repeated 10 times.

5.3. Results and discussions

5.3.1. Comparison with traditional data-driven approaches

In the present study, four traditional data-driven modeling approaches, namely, the single hidden layer neural network, SVM, PLS and NNPLS, are applied to soft sensor development for comparison. Because deep neural networks have the characteristics of both a deep nonlinear structure and a latent variable model, we make a comparison study from these two different facets. On one hand, ANN with a single hidden layer, SVM and NNPLS are introduced as traditional nonlinear shallow models. On the other hand, PLS and NNPLS are introduced as traditional latent variable models.

For a single hidden layer neural network, the number of neurons in the hidden layer is the only parameter to be selected. In this study, the number of neurons is selected as 45 according to 10-fold cross-validation. Here the coefficients of the weight matrix are uniformly initialized in range $[-0.5/fan\_in, 0.5/fan\_in]$, where fan_in is the number of input neurons in that weight matrix's layer [29]. Because the learning algorithm is stochastic, the training procedure is repeated 10 times and the result with the best performance is selected.

For conventional SVM, the most-used Gaussian kernel $\exp\{-\gamma||x_i - x_j||^2/2\}$ is adopted in this study. The regularization parameter C, the precision threshold ε, and the kernel parameter γ should be carefully determined. The cross-validation strategy is available for parameter

Table 2
Comparison of modeling results of several methods (with prior knowledge).

	Deep neural network	Single hidden layer neural network	SVM	PLS	NNPLS
Training error	1.84	2.23	1.70	3.25	2.73
Test error	3.02	3.72	3.35	4.17	3.83

Table 3
Distribution of the absolute values of test errors of deep learning and SVM.

	$\leq 1\,^{\circ}C$	$\leq 2\,^{\circ}C$	$\leq 3\,^{\circ}C$	$\leq 4\,^{\circ}C$
Deep learning	21%	45%	75%	85%
SVM	21%	44%	64%	77%

selection in SVM, and then all training data can be used to attain the model. In this study, therefore, the training set with 251 samples is used for the 10-fold cross-validation to determine the parameters. The parameters are selected as $C = 1.2$, $\varepsilon = 0.01$ and $\gamma = 0.07$. Then the model is developed with the 251 training samples in the training set.

For pure PLS, the efficacy of soft sensor is heavily influenced by the number of latent variables, and thus they should be carefully selected. In this application, the number of latent variables in PLS model was selected as 11 by 10-fold cross-validation with the training set, the same way as used for SVM. After that, the entire training set is used to train the PLS model.

For NNPLS, a neural network based nonlinear extension of PLS, it is necessary to select the number of latent variables and neurons. In this study, the number of latent variables is selected as 11 and that of neurons is selected as 2 by means of 10-fold cross-validation.

The *root mean squared error* (RMSE) is employed to evaluate the performance of different algorithms. Table 2 summarizes the modeling result of the above methods.

From columns 1 to 5 in Table 2, it can be seen that, the soft sensor based on deep neural network outperforms other traditional data-driven methods, and the model proves reliable from the generalization viewpoint. From columns 1 to 3 in Table 2, the deep neural network has better representation ability than traditional shallow models such as the single hidden layer neural network and SVM. Although the neural network with a single hidden layer uses 45 nonlinear neurons, more than the deep neural network does, its performance is much poorer. The SVM in this study generalizes worse than the deep neural network. Specifically, Table 3 further summarizes the distribution of the absolute value of test error of these two approaches. It can be seen that 75% of the prediction errors fall into the range of $\pm 3\,^{\circ}C$ in our method, as compared to 64% in SVM. Deep learning significantly reduces the number of estimations with large errors compared with SVM, indicating better generalization ability. From a practical point of view, estimations with large errors will typically have a serious influence on the quality control loop performance, deteriorating the product quality, because the estimation value of the soft sensor acts as the feedback signal in quality control loops. Therefore, the deep learning technique has practical advantages because of its better generalization and less number of estimations with large errors. It is convincing that, in comparison with shallow network structures, multi-layer deep structure is more efficient and powerful in approximation. In addition, the traditional shallow neural network and SVM make no allowance for latent variables; however, the deep neural network builds latent variable models by pre-training DBN, which is able to make full use of massive process data. With more process data employed, deep neural networks are able to extract more process information and thus enjoy better representation ability.

Among the above five methods, deep neural networks, PLS and NNPLS are latent variable models. It can be seen that the deep neural network has evident advantages. The test error of the PLS is the largest because of its inefficiency dealing with process nonlinearities. The NNPLS combines neural network and PLS to extract nonlinear latent variables, and achieves evident improvements compared with PLS; however, it is still not as outstanding as the deep network. It can be seen that the deep neural network makes a better latent variable model by using a pre-trained DBN.

5.3.2. Analysis of practical implementation complexity

For deep learning technique, there is one major concern that whether the learning algorithm is computationally affordable for industrial implementation being feasible. Here its practical implementation complexity is analyzed from two aspects: the modeling effort needed to design the soft sensor, and the computational effort needed to make a prediction after the soft sensor is put into use.

First, the deep neural network based soft sensors are practically feasible and they enjoy specific advantages dealing with large amounts of data. On one hand, the engineering effort needed to train the soft sensing model is affordable although the training of deep neural network seems troublesome. In this study, the training procedure of the deep neural network takes less than 300 s on a PC with Intel® Core® i5 1.80 GHz, 8 GB RAM using MATLAB® 2012a. By contrast, the SVM, PLS and NNPLS have a relatively shorter training time within 1 min. Such a relatively high computational cost of deep neural networks is still acceptable because soft sensors for online quality predictions are usually established in advance in an off-line manner. As long as the offline modeling time is acceptable, the prediction accuracy is expected to be as high as possible because of economic concerns. On the other hand, the modeling complexity of deep neural networks grows linearly with the number of samples. However, the modeling complexity of SVM grows exponentially with the number of samples, for the reason that training SVM amounts to a quadratic programming (QP) problem. In the presence of vast data, the computational complexity of deep neural networks is much lower than kernel methods like SVM. Consequently, with the equipment of advanced DCS and laboratory analyzers, more data would be accumulated and thus the deep neural network is much more beneficial for massive modeling data.

Another practical issue of soft sensor is its online computational cost when put into use in real production scenarios. Once a group of new samples are collected by DCS, they are immediately fed to the soft sensor as inputs, and then it usually takes some time for computers to calculate the online inference of quality indices. If the soft sensing model is over complicated, the online computation time may be considerably long, even longer than the sampling interval of the DCS. For example, if the online computation takes two seconds but the sampling interval of the DCS is only one second, the interval of estimation of quality index would be no longer expected to be same as that of DCS, and thus unable to meet the requirements of real-time quality control. Hence the computational complexity of soft sensing model

Table 4
Verifications of the efficacy of unsupervised pre-training.

	A: Randomly initialized deep neural network	B: Deep learning without prior knowledge	C: Deep learning with prior knowledge
Training error	6.20	1.95	1.84
Test error	7.95	3.04	3.02

is a major concern in industrial implementations. For nonlinear models, the computational complexity primarily depends on the number of nonlinear units, such as sigmoid function $1/[1+\exp(-v)]$ and radial basis function $\exp\{-\gamma v^2/2\}$. For SVM and some kernel methods, because the regression hyper-plane is "supported" by the training samples, the computational complexity of SVM can be approximated as $O(n_{SV})$, where n_{SV} is the number of support vectors. Alternatively, for deep neural networks, nonlinear neurons are connected much more compactly to describe the desired function, as stated in Section 2. For example, the SVM possesses 247 support vectors in this study, making 247 nonlinear units in the model. The deep neural network, however, not only yields much fewer nonlinear units (36 units in this study) but also achieves a remarkable better performance. In addition, when more and more process data are collected to update the soft sensor over time, the SVM would suffer from a large number of support vectors, leading to a rapidly increasing computational complexity. By contrast, the computational cost of deep neural networks would not increase once its structure has been determined. This implies that the deep neural network is much more beneficial for online computations.

5.3.3. Verifications of the efficacy of unsupervised pre-training

Additionally, in order to examine the efficacy of unsupervised pre-training when dealing with deep neural networks, two comparison experiments A and B are performed, respectively. They are typically used for comparison with the performance of the proposed soft sensor based on original deep neural network. For readers' convenience, the development of soft sensor based on our method is denoted as Experiment C. First, in Experiment A, the deep neural network with the same structure is trained without unsupervised pre-training. After the weights are initialized randomly, the whole deep neural network is trained simply with the supervised back-propagation phase. The network is directly trained after all weights are initialized randomly. Here the coefficients of the weight matrix are uniformly sampled in range $[-0.5/fan\_in, 0.5/fan\_in]$, where *fan_in* is the number of input neurons in that weight matrix's layer [29]. Next, in Experiment B, a group of the untransformed input variables is used to develop soft sensors. As aforementioned in Section 5.2.1, the input variables have been selected and further transformed in light of a priori physical knowledge. Instead of these transformed variables, here the direct-measured flow rates and recycle heats are used themselves (column 5 in Table 1), in total 17 process variables as the inputs of the soft sensor in Experiment B. For a fair comparison, both Experiments B and C are repeated 10 times and the performance with least validation error is selected for comparisons.

The results of Experiments A, B and C are tabulated in Table 4. From columns 1 and 3 in Table 4, it can be seen that the deep neural network without unsupervised pre-training almost loses its effectiveness. The solution is prone to local minima and it is indeed a tough task to train a deep network with random weights, as is adopted in training a shallow neural network traditionally. This negative result demonstrates that the greedy layer-wise training of the DBN drives the initial weights of deep networks into a preferable region adjacent to an optimal point and benefits the supervised learning. From columns 2 and 3 in Table 4, it can be seen that the deep learning still enjoys a desirable performance in spite of insufficient a priori knowledge. It is obvious that a higher input dimension amounts to more parameters to be learned, and the network is easier to over-fit. Generally, in order to ensure a certain degree of precision, the number of data needed must increase exponentially with the input dimension. Although more inputs are incurred in Experiment B, and both Experiments B and C utilize the same amount of training data, they yield approximately the same degree of precision with negligible disparity. This gives reliable evidence to our premise that the unsupervised pre-training in deep learning technique helps extract complex nonlinear correlation features in input variables. Because industrial processes often involve evidently correlated data, with this comparison, the deep learning is believed to be particularly suitable for learning complicated process mechanisms in modeling soft sensors.

It is worth further mentioning that the unsupervised pre-training phase takes 240–250 s approximately while the back-propagation phase only takes 40–50 s. The back-propagation phase occupies a small amount of the training time, and it further indicates the unsupervised pre-training effectively drives the initial weights closer to a desirable optimum, facilitating the back-propagation step to approach the optimum without too much effort.

6. Conclusion

This paper introduced the deep learning technique as a novel data-driven soft sensor modeling method. Advantages of deep learning and characteristics of soft sensor modeling were analyzed, and the deep learning technique was employed to develop soft sensors. The proposed method was applied to estimation of the 95% cut point of heavy diesel in a CDU. The predictions made by the deep neural network match real values much better in comparison with other traditional data-driven methods. Such case study results demonstrate that the deep learning technique can extract nonlinear latent variables effectively, making the neural network a desired latent variable model. The deep learning technique is therefore particularly suitable for soft sensor modeling, and we believe that it will have promising industrial applications such as process monitoring, fault diagnosis and advanced control.

Acknowledgements

This work was supported by the National Basic Research Program of China (2012CB720505), the National Natural Science Foundation of China (21276137) and Tsinghua University Initiative Scientific Research Program.

Appendix A. Conditional probabilities for RBM units

For binary RBM, the conditional probabilities $P(\mathbf{h}|\mathbf{v})$ and $P(\mathbf{v}|\mathbf{h})$ are derived as:

$$P(h_j = 1|\mathbf{v}) = \frac{e^{c_j + W_j\mathbf{v}}}{1 + e^{c_j + W_j\mathbf{v}}} = \text{sigm}(c_j + W_j\mathbf{v}), \tag{A1}$$

$$P(v_i = 1|\mathbf{h}) = \frac{e^{b_i + W_i^{\mathrm{T}}\mathbf{h}}}{1 + e^{b_i + W_i^{\mathrm{T}}\mathbf{h}}} = \text{sigm}(b_i + W_i^{\mathrm{T}}\mathbf{h}), \tag{A2}$$

where W_j is the jth row of $\mathbf{W}$, W_i is the ith column of $\mathbf{W}$, and $\text{sigm}(v) = 1/[1 + \exp(-v)]$ is the sigmoid function. It can be seen that for binary RBM input units $\mathbf{v}$ and hidden units $\mathbf{h}$ are symmetrical.

For Gaussian RBM, the conditional probabilities $P(\mathbf{h}|\mathbf{v})$ and $P(\mathbf{v}|\mathbf{h})$ are given as:

$$P(h_j = 1|\mathbf{v}) = \frac{e^{c_j + W_j\mathbf{v}}}{1 + e^{c_j + W_j\mathbf{v}}} = \text{sigm}(c_j + W_j\mathbf{v}), \tag{A3}$$

$$P(v_i|\mathbf{h}) = \frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{1}{2}(v_i - b_i - W_i^{\mathrm{T}}\mathbf{h})^2\right\} \sim N(b_i + W_i^{\mathrm{T}}\mathbf{h}, 1). \tag{A4}$$

It can be seen that $\mathrm{P}(v_i|\mathbf{h})$ follows a normal distribution with mean of $b_i + W_i^{\mathrm{T}}\mathbf{h}$ and unit variance.

Appendix B. Contrastive divergence for training RBMs

The CD algorithm deals with the approximation of the negative term:

$$\sum_{\tilde{\mathbf{v}}}\sum_{\mathbf{h}} P(\tilde{\mathbf{v}}, \mathbf{h}) \cdot \frac{\partial[-\text{Energy}(\tilde{\mathbf{v}}, \mathbf{h})]}{\partial\theta}. \tag{B1}$$

The two-stage *Gibbs Sampler* is an effective approximate sampling approach. A Markov chain $(\mathbf{v},\mathbf{h})$ is constructed (Fig. B1) by iterating the following steps ($t = 1,2,\ldots$):

1. Sample $\mathbf{h}^{(t)}$ from $P(\mathbf{h}|\mathbf{v} = \mathbf{v}^{(t-1)})$,
2. Sample $\mathbf{v}^{(t)}$ from $P(\mathbf{v}|\mathbf{h} = \mathbf{h}^{(t)})$,

where $\mathbf{v}^{(0)} = \mathbf{v}$. It has been proved that the chain will converge to the true joint distribution $P(\mathbf{v},\mathbf{h})$ if aperiodic and irreducible. It means that $\mathbf{v}^{(\infty)}$ and $\mathbf{h}^{(\infty)}$ are ideally sampled from the distribution $P(\mathbf{v},\mathbf{h})$. Nevertheless, in order to compute new gradients, the Gibbs sampling must be performed all the time, making the training rather expensive. By replacing $\{\mathbf{v}^{(\infty)},\mathbf{h}^{(\infty)}\}$ with $\{\mathbf{v}^{(1)},\mathbf{h}^{(1)}\}$ CD algorithm adopts a second-order approximation of the negative term, which has been found to be practically tractable and effective, even if there is only one sample (Fig. B1).

For both binary and Gaussian RBMs, the term $\partial[-\text{Energy}(\mathbf{v}, \mathbf{h})]/\partial\theta$ is computed as:

$$\frac{\partial[-\text{Energy}(\mathbf{v}, \mathbf{h})]}{\partial W_{ij}} = h_j v_i, \quad \frac{\partial[-\text{Energy}(\mathbf{v}, \mathbf{h})]}{\partial b_i} = v_i, \quad \frac{\partial[-\text{Energy}(\mathbf{v}, \mathbf{h})]}{\partial c_j} = h_j, \tag{B2}$$

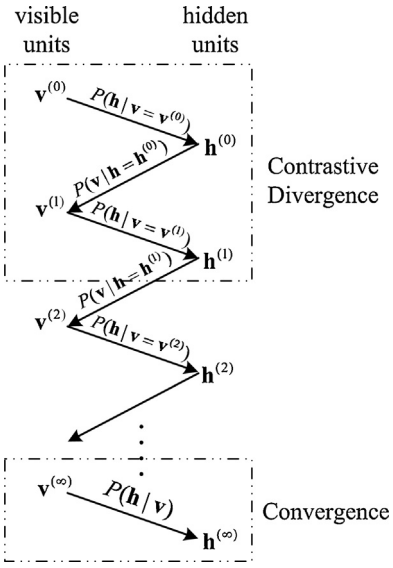

Fig. B1. Markov chain in two-stage Gibbs sampler and contrastive divergence for training RBM.

and the gradient of the log-likelihood function can be approximated as:

$$\Delta W_{ij} = \frac{\partial \log P(\mathbf{v})}{\partial W_{ij}} = \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}) \cdot h_j v_i - \sum_{\tilde{\mathbf{v}}} \sum_{\mathbf{h}} P(\tilde{v}, \mathbf{h}) \cdot h_j \tilde{v}_i \approx \mathrm{E}(h_j^{(0)}|\mathbf{v}^{(0)}) \cdot v_i^{(0)} - h_j^{(1)} v_i^{(1)} \approx \mathrm{E}(h_j^{(0)}|\mathbf{v}^{(0)}) \cdot v_i^{(0)} - E(h_j^{(1)}|\mathbf{v}^{(1)}) \cdot E(v_i^{(1)}|\mathbf{h}^{(0)})$$

$$= \mathrm{sigm}(c_j + W_j \mathbf{v}^{(0)}) \cdot v_i^{(0)} - \mathrm{sigm}(c_j + W_j \mathbf{v}^{(1)}) \cdot \mathrm{sigm}(b_i + W_i^{\mathrm{T}} \mathbf{h}^{(0)}), \tag{B3}$$

$$\Delta b_i = \frac{\partial \log P(\mathbf{v})}{\partial b_i} = \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}) \cdot v_i - \sum_{\tilde{\mathbf{v}}} \sum_{\mathbf{h}} P(\tilde{v}, \mathbf{h}) \cdot \tilde{v}_i \approx v_i^{(0)} - \mathrm{E}(\mathbf{v}_i^{(1)}|\mathbf{h}^{(0)}) = v_i^{(0)} - \mathrm{sigm}(b_i + W_{\cdot i}^{T} \mathbf{h}^{(0)}), \tag{B4}$$

$$\Delta c_j = \frac{\partial \log P(\mathbf{v})}{\partial c_j} = \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}) \cdot h_j - \sum_{\tilde{\mathbf{v}}} \sum_{\mathbf{h}} P(\tilde{v}, \mathbf{h}) \cdot h_j \approx \mathrm{E}(\mathbf{h}_j^{(0)}|\mathbf{v}^{(0)}) - E(\mathbf{h}_j^{(1)}|\mathbf{v}^{(1)}) = \mathrm{sigm}(c_j + W_j \mathbf{v}^{(0)}) - \mathrm{sigm}(c_j + W_j \mathbf{v}^{(1)}), \tag{B5}$$

where $\mathbf{h}^{(0)}$,$\mathbf{v}^{(1)}$, and $\mathbf{h}^{(1)}$ are sampled from the one-step Markov chain, as shown in Fig. B1. In the above equations, the mean field approximation is performed to avoid biphasic fluctuation, that is, to use the conditional expectation instead of the binary states sampled from the one-step Markov chain.

For normalized Gaussian units, the gradient of the log-likelihood function can likewise be approximated as:

$$\Delta W_{ij} = \frac{\partial \log P(\mathbf{v})}{\partial W_{ij}} = \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}) \cdot h_j v_i - \sum_{\tilde{\mathbf{v}}} \sum_{\mathbf{h}} P(\tilde{\mathbf{v}}, \mathbf{h}) \cdot h_j \tilde{v}_i \approx \mathrm{E}(h_j^{(0)}|\mathbf{v}^{(0)}) \cdot v_i^{(0)} - h_j^{(1)} v_i^{(1)} \approx \mathrm{E}(h_j^{(0)}|\mathbf{v}^{(0)}) \cdot v_i^{(0)} - \mathrm{E}(h_j^{(1)}|v^{(1)}) \cdot \mathrm{E}(v_i^{(1)}|\mathbf{h}^{(0)})$$

$$= \mathrm{sigm}(c_j + W_j \mathbf{v}^{(0)}) \cdot v_i^{(0)} - \mathrm{sigm}(c_j + W_j \mathbf{v}^{(1)}) \cdot (b_i + W_i^{\mathrm{T}} \mathbf{h}^{(0)}), \tag{B6}$$

$$\Delta b_i = \frac{\partial \log P(\mathbf{v})}{\partial b_i} = \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}) \cdot v_i - \sum_{\tilde{\mathbf{v}}} \sum_{\mathbf{h}} P(\tilde{v}, \mathbf{h}) \cdot \tilde{v}_i \approx v_i^{(0)} - \mathrm{E}(v_i^{(1)}|\mathbf{h}^{(0)}) = v_i^{(0)} - (b_i + W_i^{\mathrm{T}} \mathbf{h}^{(0)}), \tag{B7}$$

$$\Delta c_j = \frac{\partial \log P(\mathbf{v})}{\partial c_j} = \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}) \cdot h_j - \sum_{\tilde{\mathbf{v}}} \sum_{\mathbf{h}} P(\tilde{\mathbf{v}}, \mathbf{h}) \cdot h_j \approx \mathrm{E}(h_j^{(0)}|\mathbf{v}^{(0)}) - E(h_j^{(1)}|\mathbf{v}^{(1)}) = \mathrm{sigm}(c_j + W_j \mathbf{v}^{(0)}) - \mathrm{sigm}(c_j + W_j \mathbf{v}^{(1)}). \tag{B8}$$

References

[1] M.T. Tham, G.A. Montague, A. Julian Morris, P.A. Lant, Soft-sensors for process estimation and inferential control, J. Process Control 1 (1991) 3–14.
[2] P. Kadlec, B. Gabrys, S. Strandt, Data-driven soft sensors in the process industry, Comput. Chem. Eng. 33 (2009) 795–814.
[3] M. Kano, M. Ogawa, The state of the art in chemical process control in Japan: good practice and questionnaire survey, J. Process Control 20 (2010) 969–982.
[4] D. Dong, T.J. McAvoy, Nonlinear principal component analysis-based on principal curves and neural networks, Comput. Chem. Eng. 20 (1996) 65–78.
[5] K. Pearson, LIII. On lines and planes of closest fit to systems of points in space, Lond. Edin. Dub. Philos. Mag. J. Sci. 2 (1901) 559–572.
[6] H. Wold, Estimation of principal components and related models by iterative least squares, Multivar. Anal. 1 (1966) 391–420.
[7] P. Geladi, B.R. Kowalski, Partial least-squares regression: a tutorial, Anal. Chim. Acta 185 (1986) 1–17.
[8] S.J. Qin, Recursive PLS algorithms for adaptive data modeling, Comput. Chem. Eng. 22 (1998) 503–514.
[9] S.J. Qin, T.J. McAvoy, Nonlinear PLS modeling using neural networks, Comput. Chem. Eng. 16 (1992) 379–391.
[10] V. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[11] W.W. Yan, H.H. Shao, X.F. Wang, Soft sensing modeling based on support vector machine and Bayesian model selection, Comput. Chem. Eng. 28 (2004) 1489–1498.
[12] K. Desai, Y. Badhe, S.S. Tambe, B.D. Kulkarni, Soft-sensor development for fed-batch bioreactors using support vector regression, Biochem. Eng. J. 27 (2006) 225–239.
[13] S.J. Qin, Neural networks for intelligent sensors and control – practical issues and some solutions, in: Neural Systems for Control, 1997, pp. 213.
[14] S. Park, C. Han, A nonlinear soft sensor based on multivariate smoothing procedure for quality estimation in distillation columns, Comput. Chem. Eng. 24 (2000) 871–877.
[15] L. Fortuna, S. Graziani, M.G. Xibilia, Soft sensors for product quality monitoring in debutanizer distillation columns, Control Eng. Pract. 13 (2005) 499–508.
[16] P. Kadlec, B. Gabrys, Adaptive local learning soft sensor for inferential control support, in: Proc. 2008 International Conference on Computational Intelligence for Modelling Control & Automation, IEEE, 2008, pp. 243–248.
[17] G.E. Dahl, D. Yu, L. Deng, A. Acero, Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition, IEEE Trans. Audio Speech Lang. Process. 20 (2012) 30–42.
[18] Y. Tang, R. Salakhutdinov, G.E. Hinton, Robust Boltzmann machines for recognition and denoising, in: Proc. 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2012, pp. 2264–2271.
[19] A. Mnih, G.E. Hinton, A scalable hierarchical distributed language model, Adv. Neural Inf. Process. Syst. 21 (2009) 1081–1088.
[20] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (2009) 1–127.
[21] Y. Bengio, A. Courville, P. Vincent, Representation Learning: A Review and New Perspectives, 2012 arXiv:1206.5538.[hep-th].
[22] Y. Bengio, O. Delalleau, N. Le Roux, The curse of highly variable functions for local kernel machines, Adv. Neural Inf. Process. Syst. 18 (2006) 107.
[23] Y. Bengio, Y. LeCun, Scaling learning algorithms towards AI, in: Large-Scale Kernel Machines, 2007, pp. 34.
[24] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (2006) 1527–1554.
[25] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (2006) 504–507.
[26] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, Adv. Neural Inf. Process. Syst. 19 (2007) 153.
[27] G.E. Hinton, A practical guide to training restricted Boltzmann machines, Momentum 9 (2010) 1.
[28] M.A. Carreira-Perpinan, G.E. Hinton, On contrastive divergence learning, in: Artificial Intelligence and Statistics, 2005, p. 17.
[29] A. Cichocki, R. Unbehauen, Neural Networks for Optimization and Signal Processing, Wiley, New York, 1993.