IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 17, 2018, accepted December 6, 2018, date of publication December 14, 2018,
date of current version January 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2886820

Mutual Information-Weighted Principle Components Identified From the Depth Features of Stacked Autoencoders and Original Variables for Oil Dry Point Soft Sensor

JIE WANG AND XUEFENG YAN

Key Laboratory of Advanced Control and Optimization for Chemical Processes of Ministry of Education, East China University of Science and Technology, Shanghai 200237, China

Corresponding author: Xuefeng Yan (xfyan@ecust.edu.cn)

The authors are grateful for the support of National Natural Science Foundation of China (21878081) and Fundamental Research Funds for the Central Universities under Grant of China (222201717006).

ABSTRACT In modern chemical process control, the application of data-driven soft sensor has become increasingly extensive. Feature extraction is an important step in soft sensor. A novel feature extraction and integration method based on stacked autoencoders (SAE) and mutual information (MI)-weighted principle component analysis (PCA) was proposed to solve the loss of information on shallow depth features and original variables in neural network models. First, an SAE model was trained to extract the features of the original variables with varying depths. Second, through an MI indicator, the original variables and features with strong dependency on the outputs were selected. Then, MI was used to assign varied weights to the features and original variables, and the PCA method was used to remove any possible redundancy between the original variables and features of varying depths to obtain the principle components. Finally, the principle components were used to construct a regressor, such as a neural network. The model was first tested using the Boston housing dataset as a benchmark and then applied to the soft sensor of a constant top oil dry point. The proposed model achieved optimal results in terms of the root mean squared error and r indicators in the experiments and was thus proved feasible and useful.

INDEX TERMS Feature extraction, mutual information, soft sensor, stacked autoencoder.

I. INTRODUCTION

The petrochemical industry is fundamental to energy, transportation, textiles, materials, electronics, and people's daily lives. In modern chemical industry processes, certain key quality variables, such as product quality and gas concentration, should be accurately measured to accomplish process control and system optimization. However, some physical facilities that measure these variables are particularly expensive and suffer from frequently long delays and extreme work environments. These drawbacks render the direct measurement of these variables difficult. Soft sensor is a technique that estimates the quality of variables that are difficult to measure on the basis of other easily measured process variables. It is used as a reliable and economical alternative to expensive physical measurement sensing. Currently, soft sensor technology is widely used in process control. Several data-based modeling methods have been used to build models according to the input and output data of processes. Improvements in distributed control systems have driven modern industries into the era of big data. As a result of the difficulty in obtaining the mechanisms of complex industrial objects and given the large volume of easily accessible data in industrial processes, data-driven soft sensor technology has attracted the interest of researchers [1]–[6].

In data-driven modeling methods, neural network (NN) models can well approximate nonlinear functions and have thus been widely applied [7], [8]. Existing soft sensor methods based on NN models remarkably improve the advanced control of complex processes and product quality. In the past few years, deep NNs have been rapidly developed.

Compared with traditional shallow NNs, deep NNs construct deep models to simulate the cognitive mechanism of the human brain so as to learn some abstract concepts. When data are entered into the network, the output of the previous layer continues to serve as the input of the next layer, and the features are learned layer by layer through the neurons of the hidden layers. In deep NNs, a large number of hidden layers indicates highly abstract learned features [9]. Deep NNs have many classic models, such as convolutional NNs, recurrent NNs, deep belief networks, and stacked autoencoders (SAEs). An autoencoder is an unsupervised learning model that learns the features of data by minimizing reconstruction errors. SAEs are formed when the features extracted from a previous autoencoder are used as inputs for the next autoencoder. SAEs can learn different depth features of input data [10]. Deep learning has also made good progress in soft sensor applications.

Qiu *et al.* [11] proposed a soft sensor model on the basis of deep NN and achieved better prediction results than a shallow architecture soft sensor model did under extremely complex scenarios. Yao and Ge [12] proposed a semi-supervised deep learning model on the basis of a hierarchical extreme learning machine and applied it to soft sensor. They used the deep network structure of autoencoders to extract unsupervised features with all process samples. Using a deep learning network, Yan *et al.* [13] proposed a method for soft sensor modeling that integrates an NN with denoising autoencoders. This modeling method could capture key information from data through a deep model and thus build soft sensors with excellent performance. The abovementioned studies pioneered the application of deep NNs to soft sensor modeling and provided new tools for soft sensor in industrial processes. These algorithms share one characteristic in common; that is, they mainly use abstract features in the final layers of the deep network and ignore the various levels of abstract information contained in the layers with varying depths. Shallow NNs have only one hidden layer. Once the original feature is transformed into hidden layer information, the feature then reaches the output layer. This phenomenon indicates that the shallow features of data can also complete prediction tasks. In traditional deep NN models, the information of the original variables is transmitted layer by layer through the network. The hidden layers only perform transmission functions. Ultimately, only the final hidden layer is directly linked to the output layer, although the features of the final hidden layer are transmitted by the input data through multiple hidden layers and serve as the abstract representation of the original variables.

However, a certain amount of information loss is inevitable in data compression and transmission. Different depth features in a network contain varied levels of original variable information, which may be beneficial to process modeling. We can use the original input variables and information from all layers in the deep NN to build our regression model. However, directly using these features could cause problems, such as excessive dimensions of or redundancy between features. Several authors have reported feature extraction methods. Hild *et al.* [14] proposed the method of maximization of Renyi's mutual information (MI) using a stochastic information gradient. mRMR [15] is a method for feature extraction that sets an indicator that takes relevance and redundancy into account to ensure the selection of features with maximum relevance and minimum redundancy. Another study [16] used ICA and an MI-based criterion for feature extraction. In the current study, we propose a new feature transformation method which has a better performance on the experiments.

To address the loss of information in deep networks and the problem caused by directly using these features in modeling, we propose a novel feature extraction and integration method that is based on SAEs and MI-weighted principle component analysis (PCA). This method uses SAEs to extract abstract features of different depths from original data. According to the MI values of the original variables and features with output, the original variables and features that are strongly associated with the output are selected. Following the weighing of these variables and features by the MI values, PCA is performed to remove any redundant information and obtain the principal components. Finally, these principal components are used to build a regression model, such as an NN.

The remainder of this paper is organized as follows. Section II introduces some of the related concepts used in this method. Section III introduces the specific steps of the proposed method. Section IV discusses the benchmark test on the Boston housing dataset and soft sensor model of the constant top oil dry point in an atmospheric tower. Section V summarizes the main contributions of this article.

II. RELATED CONCEPTS

A. NN

NNs are abstract computing models that simulate the human brain [17]. An NN's structure is mainly made up of three layers, namely, input, hidden, and output layers. These layers in the NN are connected such that information can be transmitted back and forth [18]. Figure 1 shows the structure of a three-layer NN. Assuming that the training set is $D = \{(\boldsymbol{x}^1, \boldsymbol{y}^1), (\boldsymbol{x}^2, \boldsymbol{y}^2), \ldots, (\boldsymbol{x}^n, \boldsymbol{y}^n)\}$, then $\boldsymbol{x}^i \in R^d, \boldsymbol{y}^i \in R^l$, which denotes that the input data have d attributes and that the dimension of the output data is l.

The NN in Figure 1 is a multilayer feed-forward NN with l output nodes, q hidden nodes, and d input nodes. The bias of the h^{th} node in the hidden layer is γ_h, and the bias of the j^{th} node in the output layer is θ_j. The connection weight between the i^{th} node of the input layer and the h^{th} node of the hidden layer is v_{ih}, and the connection weight between the h^{th} node of the hidden layer and the j^{th} node of the output layer is w_{hj}. We set the input received by the h^{th} node of the hidden layer as

$$\alpha_h = \sum_{i=1}^{d} v_{ih} x_i \tag{1}$$

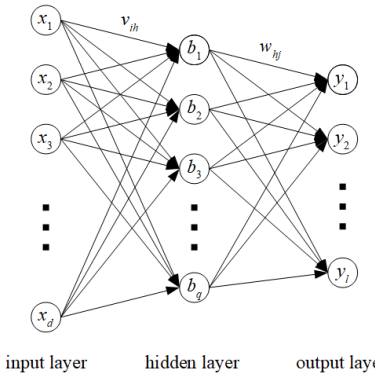

FIGURE 1. Structure of a neural network.

and the input value received by the j^{th} node in the output layer as

$$\beta_j = \sum_{h=1}^{q} w_{hj} b_h \tag{2}$$

where b_h is the output of the h^{th} node in the hidden layer. The activation function of the output and hidden layer nodes can be set to the sigmoid function, that is,

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

The function can compress a wide range of input values into range (0, 1) as output.

For the training data (x^k, y^k), we assume that the output value of the NN is $\hat{y}^k = (\hat{y}_1^k, \hat{y}_2^k, \ldots, y_l^k)$. The following formula could be obtained:

$$\hat{y}_j^k = f(\beta_j - \theta_j) \tag{4}$$

Hence, the square error of the NN is

$$E_k = \frac{1}{2} \sum_{j=1}^{l} (\hat{y}_j^k - y_j^k)^2 \tag{5}$$

where $1/2$ is added for convenience in the following derivation.

The NN in Figure 1 has a total of $(d + l + 1)q + l$ parameters to be determined, including $d \times q$ weights between the input and hidden layers, $q \times l$ weights between the hidden and output layers, q biases of the hidden layer nodes, and l biases of the output layer nodes. We can apply the back-propagation (BP) algorithm to estimate these parameters step by step [19].

B. SAE

As a special NN, the autoencoder can maximize the reconstruction of the original signal. To achieve this reconstruction, the autoencoder should obtain representative features from the input data, that is, the main components of the input data [20]. A single-layer autoencoder model can be represented as a simple NN that is composed of three layers, namely, input, hidden, and output layers. Figure 2 illustrates

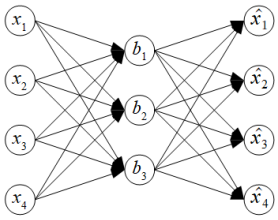

FIGURE 2. Structure of an autoencoder.

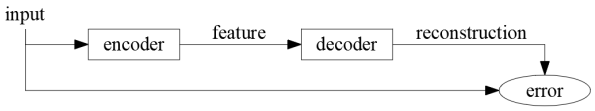

FIGURE 3. Encoding and decoding stages of an autoencoder.

an autoencoder with four nodes in the input and output layers and three nodes in the hidden layer.

As shown in Figure 3, the autoencoder model involves encoding and decoding stages. The encoding stage refers to the process of data transmission from the input layer to the hidden layer. The decoding stage refers to the process of data transmission from the hidden layer to the output layer.

In this study, we use e and d to represent the encoding and decoding operations, respectively. When the activation function is set as a sigmoid function, the mathematical expression of the two stages is presented as follows:

$$b_h = e(\alpha_h) = f(\sum_{i=1}^{d} v_{ih} x_i + \gamma_h) \tag{6}$$

$$\hat{x}_j = d(\beta_j) = f(w_{hj} b_h + \theta_j) \tag{7}$$

In the autoencoder network, the model first encodes the input vector $\boldsymbol{x}$ into vector $\boldsymbol{b}$ then decodes vector $\boldsymbol{b}$ into vector $\hat{\boldsymbol{x}}$. The decoded vector $\hat{\boldsymbol{x}}$ should approximate the input vector $\boldsymbol{x}$ optimally. The degree of approximation can be measured by reconstruction errors. Here, we use the square loss function with the following mathematical expression:

$$L(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \sum_{i=1}^{d} (x_i - \hat{x}_i)^2 \tag{8}$$

SAE is a type of deep NN that is individually made up of multiple connected autoencoders, in which the next autoencoder is responsible for re-encoding the hidden layer representation of the previous autoencoder. Whereas the first autoencoder can only learn the shallow first-level features of the input data, the second-level autoencoder can learn high-level features and new modes on the basis of the combination of the first-level features. Therefore, the SAE model can mine the representation of various levels of abstract features of input data [24]. The hidden layer can learn extremely complex abstract features. Thus, SAEs are applicable to difficult tasks, such as face recognition, handwritten character recognition, and object detection in images. An SAE can be stacked into many layers, and a deep SAE can considerably improve the performance of deep NN models while

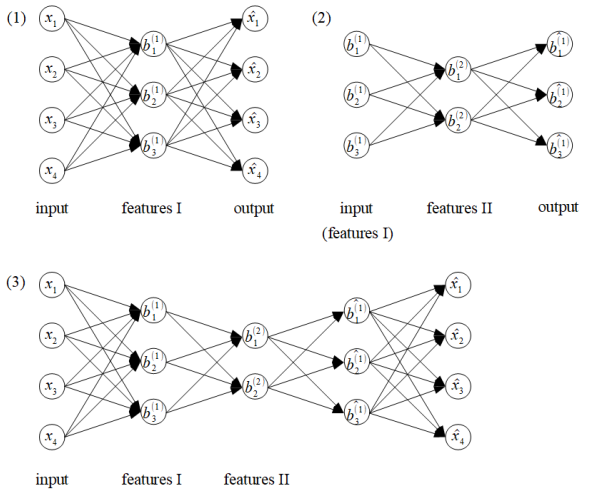

FIGURE 4. SAE construction process.

increasing the amount of computation. The features extracted from the SAE can be used as an input into the classifier or regressor to predict input data. Figure 4 shows the construction process of the SAE with two autoencoders structured as 4-3-4 and 3-2-3.

The specific training steps of the SAEs are as follows:

(1) The first autoencoder is trained using the original input $\boldsymbol{x}$, which can learn the first-level features $\boldsymbol{b}$ of the original input.

(2) Raw data are used as input for the previously trained autoencoders. For each input $\boldsymbol{x}^{(k)}$, the corresponding first-level features $\boldsymbol{b}^{(1)(k)}$ can be obtained. Then, we use these first-level features as the input of the second autoencoder to learn the high-level features $\boldsymbol{b}^{(2)(k)}$.

(3) The SAE is obtained by combining these two single-layer autoencoders.

C. PCA

PCA can remove correlations between variables via orthogonalization. In this way, most information from the original data can be retained. At the same time, the original data can be reorganized, and the redundant parts can be eliminated to obtain unrelated principal components. As a result, dimensions are reduced, and redundancies are removed [22]. The main steps of PCA are detailed below.

(1) Construction of observation sample matrix

On the basis of the collected data, an $n \times p$ dimension matrix is formed; here, n and p are the instance and number of attributes of collected data, respectively.

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_1 \\ \boldsymbol{X}_2 \\ \vdots \\ \boldsymbol{X}_n \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \tag{9}$$

(2) Construction of observation sample covariance matrix

$E\{[X - E(X)][Y - E(Y)]\}$ is the covariance of the random variables X and $Y_{\text{represented by}} cov(X, Y)$. Hence, we obtain the following:

$$cov(X, Y) = E\{[X - E(X)][Y - E(Y)]\} \tag{10}$$

The covariance matrix of vector $\boldsymbol{X} = [\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n]^T$ can be calculated according to Eq. (11) to form a symmetric matrix $\boldsymbol{C}$ as follows:

$$C = \begin{bmatrix} cov(\boldsymbol{X}_1, \boldsymbol{X}_1) & cov(\boldsymbol{X}_1, \boldsymbol{X}_2) & \cdots & cov(\boldsymbol{X}_1, \boldsymbol{X}_n) \\ cov(\boldsymbol{X}_2, \boldsymbol{X}_1) & cov(\boldsymbol{X}_2, \boldsymbol{X}_2) & \cdots & cov(\boldsymbol{X}_2, \boldsymbol{X}_n) \\ \vdots & \vdots & \ddots & \vdots \\ cov(\boldsymbol{X}_n, \boldsymbol{X}_1) & cov(\boldsymbol{X}_n, \boldsymbol{X}_2) & \cdots & cov(\boldsymbol{X}_n, \boldsymbol{X}_n) \end{bmatrix} \tag{11}$$

(3) Eigenvalue decomposition of the covariance matrix $\boldsymbol{C}$

$$\boldsymbol{Q}^T \boldsymbol{C} \boldsymbol{Q} = \boldsymbol{\Lambda} = \begin{Bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{Bmatrix} \tag{12}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix with λ_i $(i = 1, 2, \ldots, n)$ as its eigenvalue and $\boldsymbol{Q}$ is an orthogonal matrix. Then, according to the numeric value of the eigenvalue, the eigenvectors are re-ordered to obtain a new matrix:

$$\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots \boldsymbol{u}_n) = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ u_{21} & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{n1} & u_{n2} & \cdots & u_{nn} \end{bmatrix} \tag{13}$$

(4) Calculation of the rate of cumulative contribution

Cumulative contribution rate refers to the proportion of the sum of the first k eigenvalues and the sum of all eigenvalues

$$\rho = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{n} \lambda_i}. \tag{14}$$

The number of principal components that should be extracted is not fixed but is determined on the basis of the fact that the cumulative contribution rate ρ reaches a certain percentage. Thus, we set the condition as $\rho > 95\%$. When the cumulative variance contribution rates of the largest k eigenvalues meet the requirement, k principal components can be selected.

(5) Calculation of principal components

$$\boldsymbol{Z} = \boldsymbol{P}^T \boldsymbol{X} \tag{15}$$

where $\boldsymbol{P}$ is the matrix of the first k columns of the eigenvector matrix $\boldsymbol{U}$. Therefore, the k principal components are calculated as follows:

$$\begin{cases} \boldsymbol{Z}_1 = u_{11}x_{11} + u_{12}x_{12} + \cdots + u_{1n}x_{1n} \\ \boldsymbol{Z}_2 = u_{21}x_{21} + u_{22}x_{22} + \cdots + u_{2n}x_{2n} \\ \vdots \\ \boldsymbol{Z}_k = u_{k1}x_{k1} + u_{k2}x_{k2} + \cdots + u_{kn}x_{kn} \end{cases} \tag{16}$$

D. MI

In probability theory, the MI of two random variables is a measurement of their interdependence [23]. The MI of the random variables X and Y is denoted as $I(X; Y)$ and mathematically defined as

$$I(X; Y) = H(X) - H(X|Y) \tag{17}$$

where $H(X)$ is the information entropy of X and $H(X|Y)$ is the conditional information entropy of X when given Y. The information entropy and conditional information entropy of the random variables are mathematically defined as follows

$$H(X) = -\sum_{i=1}^{n} P(X_i) \log(P(X_i)) \tag{18}$$

$$H(X|Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} P(X = x_i, Y = y_i) \log(P(X = x_i|Y = y_i)) \tag{19}$$

where n and m represent the number of discrete states of X and Y, respectively; and $P(X = x_i, Y = y_i)$ represents the joint probability distribution of X and Y.

On the basis of the definition, we determine that if the joint distribution $p(x, y)$ of two variables can be obtained, then their correlation can be obtained by calculating the MI, and the dependency among the variables can be analyzed. For continuous data, we can use the histogram method to transform them into discrete data so as to estimate probability density.

III. SAE–MIPCA-BASED FEATURE EXTRACTION AND INTEGRATION METHOD

The SAE method can construct features with varying depths in different hidden layers of a network. These features are high-level representations of input variables and may contain information that is helpful for modeling. The traditional deep NN only uses the features in the final hidden layers to obtain the output. However, the features of the hidden layer become increasingly abstract as the depth of the network increases. In fact, original variables and certain shallow features in a deep NN also contain information with output. Hence, the traditional deep NN may suffer from information loss. The combination of original variables and the different level features of the original variables should contain more information than original variables do. However, mixing them directly will cause problems such as multicollinearity or feature dimensions that are too high; the outcome is detrimental to modeling. To avoid these problems, we can use the MI index to select output-related variables and features and weigh them prior to the PCA. This step is deemed as supervised PCA. That is, the variables with strong dependence on the output will play important roles in PCA, whereas the importance of variables with a week dependency on the output will be suppressed. Therefore, these processing steps can construct features with more information than original variables do and with minimal redundancy.

After obtaining the feature output by MIPCA, we can use their features to train a regressor.

The proposed method can be summarized in detail as follows.

Data Preprocessing Stage:

(1) To attain a uniform scale for each attribute of the data and improve the stability of the model, we should normalize each dimension of the input and output of the dataset such that the range of each feature in the dataset lies in the interval of [0, 1] before training.

(2) We divide the entire dataset into a training set X_{train}, a validation set $X_{validation}$, and a test set X_{test} with ratios of 60%, 20%, and 20%, respectively. The training set is used to train the model offline. The validation set is served to adjust the hyper-parameters of the model to find the optimal model. The test set is used to observe the prediction performance of the model online.

Offline Modeling Stage:

(3) We train an SAE consisting of m autoencoders on X_{train} to obtain the features of m different depths $\{B^1, B^2, \ldots, B^m\}$, where B^i is the feature extracted from the hidden layer of the autoencoder in the SAE. Each autoencoder adopts a strategy of early stopping during training to avoid over-fitting. That is, when the reconstruction error of each autoencoder is no longer reduced on the validation set during s steps, the training is stopped in advance to avoid over-fitting. In this manner, we can obtain features with enhanced generalization capabilities. The quantity of autoencoders and nodes of each autoencoder that belongs to the SAE can be set as the parameter with the best effect through multiple experiments. Typically, when the number of training samples is large and the dimension of original variables is small, we can set a large number of layers of the SAE because the risk of over-fitting is small. Conversely, when the dimension of the original variables is large but the number of training samples is insufficient, the number of layers of the SAE cannot be set too high to avoid over-fitting. The number of nodes in the hidden layer of each autoencoder can be set according to experience. Thus, we set it to 75% of the number of nodes in the input layer as the initial number. Then, the number is adjusted on the basis of the reconstruction error of the autoencoder on the validation set.

(4) The effective MI threshold MI_{th} is determined as follows. Exactly 1,000 random vectors are randomly generated by a uniform distribution in the range of [0,1]. The dimension of each vector is consistent with the number of training samples. Then, the MI of the 1,000 random vectors is calculated with the training data outputs. The 50th largest MI value is selected as threshold MI_{th}. Thus, for a certain feature, if the MI with the output is greater than the threshold MI_{th}, the feature has a certain causal relationship with the output at a 95% confidence level.

(5) The variables V_{re} with strong relevance to the output values are selected from the original variables and extracted features. Specifically, we calculate the MI between variables in X_{train} and $\{B^1, B^2, \ldots, B^m\}$ with the outputs of the

training set. If the MI of a variable with the output value is higher than MI_{th}, then the variable is selected. Otherwise, the variable is discarded.

(6) We obtain the principal component V_{pc} of the reserved variables V_{re} using the MI-weighted PCA. That is, the matrix of reserved features W_{re} is multiplied with a diagonal matrix (such that each element is the MI of each reserved feature with outputs) to increase the importance of features with large MI with output and decrease the importance of features with small MI. This step yields the weighted reserved features of W_{re}. Then, the PCA algorithm is used to process W_{re} to remove possible redundancies between these features due to the relevance of different depths in the network. The number of principal components k retained by the PCA can be considered to meet the cumulative contribution rate of 95%. Thus, we obtain the principle components of V_{pc}.

(7) We use V_{pc} as input data and label y as output to train a regressor. The regressor should be simplified and easily set up as an NN with one or two hidden layers because the input already contains the components of the original variables and different depth features.

Online Prediction Stage:

(8) The test data set X_{test} is processed using trained network structures and parameters to obtain the prediction values. Specifically, we first use the SAE that was trained in step (3) to acquire the features of different depths of the test data. Then, we reserve the variables of the same nodes for the test data and features according to the variables reserved in step (5). We use the same MI weights and PCA parameters in step (6) to obtain the MI-weighted principal components. Finally, the principal components of the feature are input into the regressor that was trained in step (7), and the output value of the regressor should be inversely normalized to obtain the predicted value $\hat{y}$.

IV. CASE STUDY

A. BENCHMARK TEST ON BOSTON HOUSING DATASET

The dataset was originally published by Du *et al.* [21]. The data in the dataset date back to 1978. Exactly 506 instances cover information on the 14 attributes of houses in various suburbs in Boston, Massachusetts. This dataset is often used as a benchmark to test algorithm performance. The hardware environment for the experiment was an Intel i5-4200u CPU with 8 GB RAM.

We used the first 13 attributes as independent variables and the 14th attribute as the dependent variable that we sought to predict. After normalizing each dimension of the independent and dependent variables into a range of [0, 1], we randomly selected 304 instances from 506 samples as the training set, 101 instances as the validation set, and 101 instances as the test set.

We trained the SAE to extract the different depth features of the original attributes. After several experiments, the structure of the SAE was determined to be formed by three stacking autoencoders. The structures of the three autoencoders were 13-10-13, 10-7-10, and 7-4-7, respectively. The training process involved the use of the adaptive moment estimation (Adam) optimization algorithm. Compared with the traditional BP algorithm, the Adam algorithm is more efficient and stable. At the same time, the reconstruction error on the validation set was observed after each parameter update. If the error on the validation set no longer dropped within $s = 30$ iterations, then the training would be stopped in advance to avoid the over-fitting of the autoencoder model. In this manner, features including the 13 original variables and $10 + 7 + 4 = 21$ different depth abstract features were obtained for each input sample. Then, we calculated the MI between each feature with output. After 1,000 times of random sampling, the MI threshold was determined to be 0.1626 with 95% confidence. Figure 5 shows the MI of each feature with output and the value of threshold MI_{th}. MI was calculated using the histogram method, and the box number was set to 10.

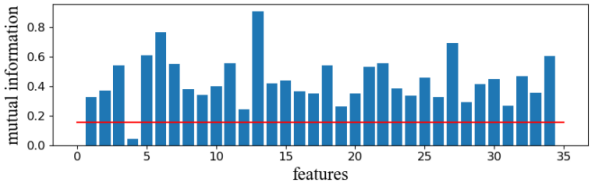

FIGURE 5. MI of each feature with output and the value of the threshold.

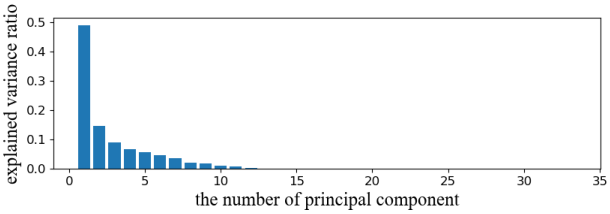

FIGURE 6. Explained variance contribution ratio of each principle component.

Figure 5 shows that only the fourth feature's MI with output fell below the threshold. Hence, we concluded that the fourth feature was independent of the output and thus discarded it without proceeding to the subsequent calculation. The remaining features were multiplied by MI and processed by PCA. Figure 6 displays the corresponding variance contribution of each principal component.

After calculation, the cumulative variance contribution rate of the first eight principal components reached 95.03%. Therefore, the first eight principal components were retained as input features of the next regressor.

Finally, we should use these principal components to build the regressor. We selected three types of regressor, namely, NNs, support vector regression (SVR), and random forest (RF). After several experiments, the NN's structure became three layers. The quantities of nodes in each layer were set to 8, 8, and 1 for the input, hidden, and output layers, respectively. The activation function could use the sigmoid function due to the small number of layers. Similar to the abovementioned step in training the autoencoders, the Adam optimization algorithm and early stopping strategy were also

used when training this NN. The number of waiting steps for the early stopping strategy was also set to 30. The parameters of the SVR model were as follows: penalty parameter c was set to 1, and the kernel function used RBF. The parameters of the RF model were as follows: the number of trees was set to 500, and the minimum number of samples to split was set to 2. At this point, the offline training stage of the model was completed.

In the online prediction stage, the test data were brought into the network to obtain the output value of the model. Then, the output value was processed by anti-normalization to obtain the final prediction value $\hat{y}$.

In this test, two indicators, namely, root mean squared error (*RMSE*) and Pearson correlation coefficient (r), were used to evaluate the performance of the algorithm. Eqs. (20) and (21) provide their mathematical definitions.

$$RMSE = \sqrt{\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (\hat{y}^i - y^i)^2} \tag{20}$$

$$r = \frac{\sum_{i=1}^{n_{test}} (y^i - \bar{y})(\hat{y}^i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{n_{test}} (y^i - \bar{y})^2} \sqrt{\sum_{i=1}^{n_{test}} (\hat{y}^i - \bar{\hat{y}})^2}} \tag{21}$$

where $\hat{y}^i$ is the prediction value of the i^{th} test data, y^i is the true value of the i^{th} test data, $\bar{\hat{y}}$ is the mean of the prediction values, and $\bar{y}$ is the mean of the true values. To eliminate the fluctuation of the model performance caused by the random distribution of data, we conducted 20 random assignments, fitted 20 models, and averaged 20 prediction performances as the total performance of the model.

To further explain the prediction performance of the model, we trained six models for comparison. Three of these models were NN, SVR, and RF, and they directly used the original variables as input. Their parameters were also equal to the regressors in the proposed method. The other three methods were built by changing the MIPCA in the proposed method by mRMR as the method for selecting the features.

Figure 7 shows the prediction and true values of different modeling methods on the test set. The abscissa of each point in the figure represents the prediction value, and the ordinate represents the true value. The closer the data point is to the dotted line of the symmetric axis, the better the prediction effect of the model is. Table 2 shows the performance metrics of the various methods.

On the basis of Figure 7, we found that the points predicted by the proposed method were close to the dotted line in each sub-graph; thus, the prediction was accurate. The same conclusion can be drawn in Table 1. The *RMSE* and r indexes were the best in each group of regressors, followed by SAE-mRMR for each group. This finding showed that the features that were transformed by the SAE-MIPCA method had better representation capabilities to build accurate models on the benchmark dataset compared with the other methods.

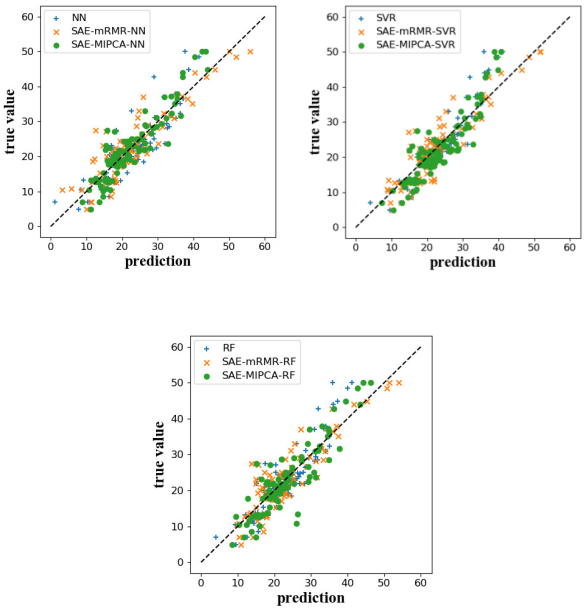

FIGURE 7. Prediction and true values of modeling methods on the test set.

TABLE 1. Prediction accuracy indicators of different algorithms.

Method	*RMSE*	r	Time(s)
SAE-MIPCA-NN	3.746	0.915	25.54
NN	3.878	0.908	4.29
SAE-mRMR-NN	3.851	0.910	26.32
SAE-MIPCA-SVR	3.881	0.906	22.63
SVR	4.194	0.898	0.49
SAE-mRMR-SVR	4.118	0.901	22.84
SAE-MIPCA-RF	4.121	0.901	23.48
RF	4.193	0.898	0.95
SAE-mRMR-RF	4.174	0.899	23.86

B. SOFT SENSOR MODEL OF CONSTANT TOP OIL DRY POINT IN AN ATMOSPHERIC TOWER

Atmospheric distillation towers are the primary production facilities for refineries. Their production level directly affects the utilization rate of crude oil and the economic benefits of enterprises. As a typical multilateral distillation tower, the atmospheric tower extracts kerosene, gasoline, diesel, and other products from the sidelines. The following quality indicators need to be controlled for the products of atmospheric tower distillation: dry point of constant top oil, flash point, freezing point, distillation range of gasoline, condensation point of diesel, 95% distillation temperature, and so on. The main product at the top of the atmospheric tower is the constant top oil, which this study aims to analyze through experimentation. If the dry point of the constant top oil is extremely high, then the heavy components of the produced oil will be extremely high and affect the quality of the oil. Therefore, we build a soft sensor model of the constant top oil using the proposed method.

In this experiment, the following independent variables were selected: total outlet temperature, total flow, tower top temperature, tower top pressure, top reflux output energy, top product volume, constant light oil flow, constant first-line flow, constant second-line flow, constant third-line flow, constant top cycle output energy, constant first middle output energy, constant second middle output energy, vaporization section temperature, and stripping steam flow. These factors can be obtained directly (or counted indirectly) through the DCS system in real time. In addition, the properties of crude oil directly affect the dry point of the constant top oil. However, the properties of crude oil are often difficult to obtain due to the frequent transformation of refined crude oil. We could use the artificial analysis value of the dry point of the constant top oil in the previous few moments as an indirect representation of the properties of refined crude oil. If too many moments were selected, the complexity of the model would increase. Through several experiments, selecting the artificial analysis values of the dry point in the four moments before the current moment as independent variables was deemed appropriate.

The experiment collected 150 samples and randomly selected 90 instances as the training set, 30 instances as the validation set, and 30 instances as the test set. The proposed method was used to model the soft sensor of the atmospheric tower. After several experiments, the SAE's parameters were set as follows: the input layer had 19 nodes, and three autoencoders were added in turn. The structure of each autoencoder was 19-15-19, 15-11-15, and 11-7-11. After 1,000 times of random sampling, the MI threshold MI_{th} was set to 0.3652. Figure 8 shows the original variables of the input layer and the MI values of the different depth features as extracted by the SAE with dry points.

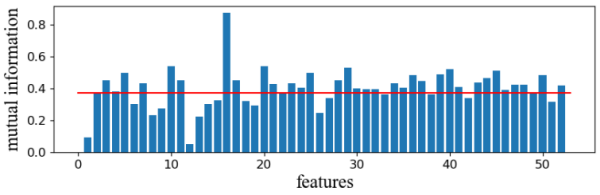

FIGURE 8. MI value of each feature with dry point.

Figure 8 shows that 35 variables whose MI values with dry points exceeded the threshold were reserved. After multiplying these variables with their MI, the PCA algorithm was used to remove possible redundancies and to simplify the variables. Figure 9 depicts the ratio of the explained variance of each principal component.

The cumulative variance contribution rate of the first 11 principal components reached 96.26%. Therefore, we reserved 11 principal components as the input features of the following regressor.

We continued to use NN, SVR, and RF as regressors. After several experiments, the structure of the NN was set as follows: the input layer had 11 nodes, the hidden layer had 15 nodes, and the output layer had 1 node. The activation function adopted the sigmoid function. Similarly, the Adam parameter optimization algorithm was used during training, whereas the early stopping strategy was used to avoid the over-fitting of the regressor. The parameters of SVR were as follows: penalty parameter c was set to 1, and the kernel function used the RBF kernel function. The parameters of the RF model were as follows: the number of trees was set to 500, and the minimum number of samples to split was set to 2.

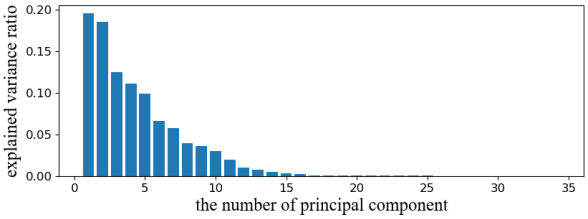

FIGURE 9. Explained variance ratio of each principal component.

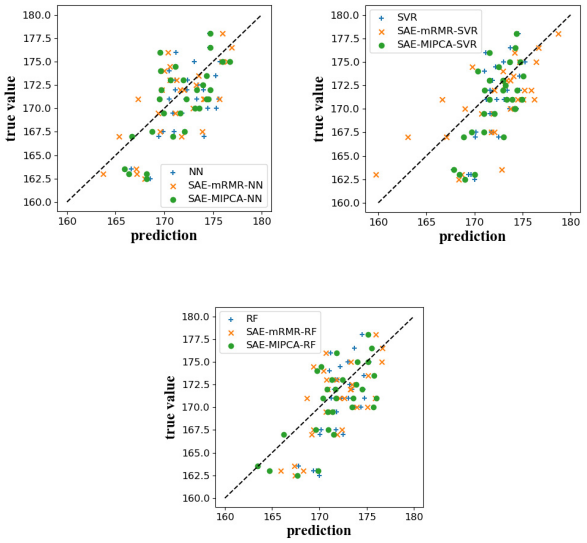

FIGURE 10. Prediction results and true values of dry point.

During the online test, the test data were brought into the network, where the output values of the model should be anti-normalized to obtain the final prediction value $\hat{y}$. The experiment was repeated 20 times, and the average of the 20 prediction performances was taken as the performance of the model.

For comparison, the NN, SVR, RF, SAE-mRMR-NN, SAE-mRMR-SVR, and SAE-mRMR-RF models were trained. Figure 10 provides the prediction results and true values of the dry point. Table 2 presents the prediction accuracy indicators *RMSE* and r.

The results in Table 2 show that when the regressor was set as NN or RF, the models with SAE to extract high level features underwent considerable improvement in accuracy. However, when the regressor was SVR, the improvement was minimal. Conversely, the accuracy of the models using MIPCA as the feature transformer was slightly higher than

TABLE 2. Prediction accuracy indicators of different algorithms.

Method	*RMSE*	*r*	Time(s)
SAE-MIPCA-NN	2.991	0.711	19.65
NN	3.571	0.625	3.48
SAE-mRMR-NN	3.075	0.701	20.83
SAE-MIPCA-SVR	3.447	0.684	16.27
SVR	3.480	0.675	0.24
SAE-mRMR-SVR	3.446	0.682	15.73
SAE-MIPCA-RF	2.965	0.717	16.91
RF	3.479	0.675	0.75
SAE-mRMR-RF	3.112	0.695	16.45

that of the models using mRMR with the same regressor. The results of the experiment indicated the effectiveness of the method in soft sensor for constant top oil dry points.

V. CONCLUSION

For the original NN model, only the last hidden layer is directly linked to the output layer. Moreover, the direct output-related information that may be contained in the input layer and the previous hidden layer does not directly affect the output in the network and may be lost in the process of transmission. This study proposed to use SAEs to extract the features of different levels and then filter the original variables and features with high MI using the outputs. The proposed method uses MI-weighted PCA to strengthen the weights of features with strong dependency on outputs and weaken the weights of features with minimal dependency on outputs. At the same time, this method removes the possible redundancies of different depths in those features. We used the principal components of the weighted features to construct a regressor, such as the NN. The benchmark test on the Boston housing dataset and soft sensor model of constant top oil dry points showed that the modeling accuracy of this method was better than that of other models in the experiments. This result proved the effectiveness of the proposed method in dry point soft sensor for constant top oil.

The proposed method did show certain drawbacks. For example, the structure of the SAE and NN regressor in the method was determined only after multiple attempts during the experiments; hence, the process was time consuming. In addition, the improvement of the model performance for different regressions varied, although the previous feature extraction process remained unchanged. For the NN regressor, this satisfactory feature extraction process improved the regression performance. However, in some cases of SVR, the promotion was not as big as that for NN.

These issues need to be addressed in future research.

REFERENCES

[1] X. Yan and W. Zhao, "4-CBA concentration soft sensor based on modified back propagation algorithm embedded with ridge regression," *Intell. Automat. Soft Comput.*, vol. 15, no. 1, pp. 41–51, Mar. 2009.

[2] J. F. de Canete, P. D. Saz-Orozco, R. Baratti, M. Mulas, A. Ruano, and A. Garcia-Cerezo, "Soft-sensing estimation of plant effluent concentrations in a biological wastewater treatment plant using an optimal neural network," *Expert Syst. Appl.*, vol. 63, pp. 8–19, Nov. 2016.

[3] Y. Liu, C. Yang, Z. Gao, and Y. Yao, "Ensemble deep kernel learning with application to quality prediction in industrial polymerization processes," *Chemometrics Intell. Lab. Syst.*, vol. 174, pp. 15–21, Mar. 2018.

[4] J. Tang, J. Zhang, Z. Wu, Z. Liu, T. Chai, and W. Yu, "Modeling collinear data using double-layer GA-based selective ensemble kernel partial least squares algorithm," *Neurocomputing*, vol. 219, pp. 248–262, Jan. 2017.

[5] L. Xu, J. Wang, H. Zhang, and T. A. Gulliver, "Performance analysis of IAF relaying mobile D2D cooperative networks," *J. Franklin Inst.*, vol. 354, no. 2, pp. 902–916, Jan. 2017.

[6] L. Xu, J. Wang, H. Zhang, and T. A. Gulliver, "Outage performance for IDF relaying mobile cooperative networks," *Mobile Netw. Appl.*, vol. 23, no. 6, pp. 1496–1501, Dec. 2018.

[7] I. B. R. Nogueira *et al.*, "A quasi-virtual online analyser based on an artificial neural networks and offline measurements to predict purities of raffinate/extract in simulated moving bed processes," *Appl. Soft Comput.*, vol. 67, pp. 29–47, Jun. 2018.

[8] A. Jiménez, G. Beltrán, M. P. Aguilera, and M. Uceda, "A sensor-software based on artificial neural network for the optimization of olive oil elaboration process," *Sens. Actuators B, Chem.*, vol. 129, no. 2, pp. 985–990, Feb. 2008.

[9] C. Shang, F. Yang, D. Huang, and W. Lyu, "Data-driven soft sensor development based on deep learning technique," *J. Process Control.*, vol. 24, no. 3, pp. 223–233, Mar. 2014.

[10] H. Ghodrati and A. B. Hamza, "Nonrigid $_3$D shape retrieval using deep auto-encoders," *Appl. Intell.*, vol. 47, no. 1, pp. 44–61, Jul. 2017.

[11] Y. Qiu, Y. Liu, and D. Huang, "Date-driven soft-sensor design for biological wastewater treatment using deep neural networks and genetic algorithms," *J. Chem. Eng. Jpn.*, vol. 49, no. 10, pp. 925–936, Oct. 2016.

[12] L. Yao and Z. Ge, "Deep learning of semisupervised process data with hierarchical extreme learning machine and soft sensor application," *IEEE Trans. Ind. Electron.*, vol. 65, no. 2, pp. 1490–1498, Feb. 2018.

[13] W. Yan, D. Tang, and Y. Lin, "A data-driven soft sensor modeling method based on deep learning and its application," *IEEE Trans. Ind. Electron.*, vol. 64, no. 5, pp. 4237–4245, May 2017.

[14] K. E. Hild, D. Erdogmus, K. Torkkola, and J. C. Principe, "Feature extraction using information-theoretic learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 9, pp. 1385–1392, Sep. 2006.

[15] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1226–1238, Aug. 2005.

[16] T. Trappenberg, J. Ouyang, and A. Back, "Input variable selection: Mutual information and linear mixing measures," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 1, pp. 37–46, Jan. 2006.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[18] M. Hou *et al.*, "A new hybrid constructive neural network method for impacting and its application on tungsten price prediction," *Appl. Intell.*, vol. 47, no. 1, pp. 28–43, Jul. 2017.

[19] B. Wang, X. Gu, L. Ma, and S. Yan, "Temperature error correction based on BP neural network in meteorological wireless sensor network," *Int. J. Sensor Netw.*, vol. 23, no. 4, pp. 265–278, Jan. 2017.

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533, Oct. 1986.

[21] B. Du, W. Xiong, J. Wu, L. Zhang, L. Zhang, and D. Tao, "Stacked convolutional denoising auto-encoders for feature representation," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 1017–1027, Apr. 2017.

[22] N. Japkowicz, S. J. Hanson, and M. A. Gluck, "Nonlinear autoassociation is not equivalent to PCA," *Neural Comput.*, vol. 12, no. 3, pp. 531–545, 2000.

[23] R. Martín, R. Aler, and I. M. Galván, "A filter attribute selection method based on local reliable information," *Appl. Intell.*, vol. 48, no. 1, pp. 35–45, Jan. 2018.

[24] D. Harrison, Jr., and D. L. Rubinfeld, "Hedonic housing prices and the demand for clean air," *J. Environ. Econ. Manage.*, vol. 5, no. 1, pp. 81–102, Mar. 1978.

JIE WANG received the B.E. degree in automation from Harbin Engineering University, Harbin, China, in 2016. He is currently pursuing the M.E. degree with the Key Laboratory of Advanced Control and Optimization for Chemical Processes of the Ministry of Education, East China University of Science and Technology, Shanghai, China. His research interests include data mining, machine learning, and soft sensors.

XUEFENG YAN received the Ph.D. degree from Zhejiang University, China. He is currently a Professor with the East China University of Science and Technology. His research interests include complex chemical process modeling, optimizing and controlling, process monitoring, fault diagnosis, and intelligent information processing.