Computers and Chemical Engineering 67 (2014) 33–42

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

Cognitive fault diagnosis in Tennessee Eastman Process using learning in the model space

Huanhuan Chen [a,*], Peter Tiňo [b], Xin Yao [b]

[a] UBRI, School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China
[b] CERCIA, School of Computer Science, University of Birmingham, Birmingham B15 2TT, United Kingdom

ARTICLE INFO

Article history:
Received 11 April 2013
Received in revised form 14 February 2014
Accepted 25 March 2014
Available online 3 April 2014

Keywords:
Learning in the model space
Tennessee Eastman Process
Fault detection
Cognitive fault diagnosis
Reservoir computing
One class learning

ABSTRACT

This paper focuses on the Tennessee Eastman (TE) process and for the first time investigates it in a cognitive way. The cognitive fault diagnosis does not assume prior knowledge of the fault numbers and signatures. This approach firstly employs deterministic reservoir models to fit the multiple-input and multiple-output signals in the TE process, which map the signal space to the (reservoir) model space. Then we investigate incremental learning algorithms in this reservoir model space based on the "function distance" between these models. The main contribution of this paper is to provide a cognitive solution to this popular benchmark problem. Our approach is not only applicable to fault detection, but also to fault isolation without knowing the prior information about the fault signature. Experimental comparisons with other state-of-the-art approaches confirmed the benefits of our approach. Our algorithm is efficient and can run in real-time for practical applications.

1. Introduction

With the development of chemical industry, chemical processes become more complex. The product efficiency and consistency become essential. Therefore, on-line monitoring and fault diagnosis are gaining more attention for produce quality and plant safety. In recent years, there has been a lot of research in the design and analysis of fault diagnosis schemes for different dynamic systems (for example, Chen and Patton, 1999; Gertler, 1998). A significant part of the research has focused on linear dynamical systems, where it is possible to obtain rigorous theoretical results. More recently, considerable effort has been devoted to the development of fault diagnosis schemes for nonlinear systems with various kinds of assumptions and fault scenarios (Zhang et al., 2002, 2005; Yan and Edwards, 2007).

These traditional fault diagnosis approaches rely, to a large degree, on the mathematical model of the "normal" system. If such a mathematical model is available, then fault diagnosis can be achieved by comparing actual observations with the prediction of the model. Most autonomous fault diagnosis algorithms are based on this methodology. However, for complex chemical processes operating in dynamic environments, such mathematical models may not be accurate or even unavailable at all. Therefore, it is necessary to develop cognitive fault diagnosis methods based on the real-time data.

In a typical chemical process, there are a large number of input variables, measurement (output) variables in chemical plants. Some of these variables are highly correlated, which increases the difficulty to extract useful information in the diagnosis process. For example, a significant change in output variables may be driven by input variables or by faults. Most of the existing data-driven fault diagnosis approaches that rely on detection of output concept drift using signals cannot deal with this kind of situation. The usual methodology is to employ an estimator, such as a neural network, to approximate the mapping from input variables to output variables. Then, the difference between the exact observations and the predicted outputs are compared for fault diagnosis. As this methodology has employed the estimator to approximate the input-output mapping, it may reduce the false alarm rate. However, the estimator can only produce accurate results when given sufficient data in all kinds of situations, such as in the normal regime and various fault scenarios. In a practical chemical process, it is expensive to obtain all such data. The absence of training data would result in low fault detection rate.

To address these problems, we introduce a novel "learning in the model space" framework for dealing with fault detection and fault isolation when no or very limited knowledge is provided about the underlying system (Chen et al., 2014). In this framework, we do not assume that we know the type, the number or the functional form

* Corresponding author. Tel.: +86 551 63606730.
E-mail addresses: hchen@ustc.edu.cn (H. Chen), P.Tino@cs.bham.ac.uk (P. Tiňo), X.Yao@cs.bham.ac.uk (X. Yao).

http://dx.doi.org/10.1016/j.compchemeng.2014.03.015
0098-1354/

of the faults in advance. The core idea is to transform the signal into a higher dimensional "dynamical feature space" via reservoir computation models and then represent varying aspects of the signal through variation in the linear readout models trained in such dynamical feature spaces. In this way parts of the signal captured in a sliding window will be represented by the reservoir model with the readout mapping fitted in that window.

Reservoir computing (RC) (Lukoševičius and Jaeger, 2009) is a class of state space models based on a "fixed" randomly constructed state transition mapping, realized through so-called reservoir and an trainable (usually linear) readout mapping from the reservoir. In our formulation, the underlying reservoir will be the *same* throughout the signal – the differences in the signal characteristics at different times will be captured solely by the linear readout models and will be quantified in the function space of readout models.

We assume that for some sufficiently long initial period the system is in a 'normal/healthy' regime so that when a fault occurs the readout models characterizing the fault will be sufficiently 'distinct' from the normal ones. A variety of novelty/anomaly detection techniques can be used for the purposes of detection of deviations from the 'normal'. In this paper we will use one-class support vector machines (OCS) (Schölkopf et al., 2001) in the readout model space. As new faults occur in time they will be captured by our incremental fault library building algorithm operating in the readout model space.

The main contributions of this paper include:

- This paper *for the first time* investigates the *cognitive fault diagnosis on the TE process* without prior knowledge of the fault numbers and types. To our knowledge, there is no existing work on *cognitive fault diagnosis* on the TE process. All existing work on fault diagnosis on the TE process relies on the assumption that all the fault patters are known in advance.
- This paper also studies the strategy to dynamically construct fault dictionary in real time.

The rest of this paper is organized as follows. The background and the related work are reviewed in Section 2. Section 3 introduces deterministic reservoir computing and the framework of "learning in the model space", followed by the incremental one class learning algorithm for cognitive fault diagnosis in Section 3.2. The experimental results and analysis on Tennessee Eastman Process are reported in Section 4. Finally, Section 5 concludes the paper and presents some future work.

2. Background and related work

The fault diagnosis procedure can often be investigated in three steps: (i) fault detection is the process of determining whether a fault has occurred or not; (ii) fault isolation deals with the issue of determining the location/type of fault; and (iii) fault identification provides an estimate of the magnitude or severity of the fault. In some cases, the issues of fault isolation and fault identification are interwoven, since they both deal with determining the type of fault that has occurred.

Most automated fault diagnosis algorithms are based on the available mathematic models. However, for complex engineering systems operating in uncertain environments, such mathematical models may not be accurate or even unavailable at all. Therefore, it is necessary to develop cognitive fault diagnosis methods based on the observed data.

The data driven approaches are popular fault diagnosis methods when the system models are unclear, especially in distributed systems. A general learning methodology for fault diagnosis of nonlinear systems was first developed by Polycarpou and Helmicki (1995), where the stability and approximation properties of the learning scheme were rigorously investigated for the ideal case without modelling uncertainty. There have been other learning based approaches to fault detection and diagnosis, e.g. Vemuri and Polycarpou, 1997; Palade and Bocaniala, 2010; Venkatasubramanian et al., 2003; Kankar et al., 2011. Neural networks were used as learning algorithms for fault detection and diagnosis, e.g. Vemuri and Polycarpou, 1997; Venkatasubramanian et al., 2003; Palade and Bocaniala, 2010. In 2011, Barakat et al. (2011) proposed to use self adaptive growing neural network for faults diagnosis. They applied wavelet decomposition and used the variance and kurtosis of the decomposed signals as features to train neural networks.

In fault detection and diagnosis, Tennessee Eastman (TE) process, created by the Eastman Chemical Co., has been widely used as a benchmark for evaluating process diagnosis methods (Fig. 2). In 2009, Yélamos et al. (2009) proposed to use support vector machines for fault diagnosis in chemical plants. In a specific application, neural network and support vector machines have been employed to identify ball bearings faults (Kankar et al., 2011). Principal component analysis (PCA) (Raich and Cinar, 1995, 1997; Kano et al., 2000), multiway PCA (Chen and McAvoy, 1998), partial PCA (Huang et al., 2000), nonlinear dynamic PCA (Lin et al., 2000), pattern recognition (Kassidas et al., 1998), Fisher discriminant analysis (FDA) (Chiang et al., 2004), PCA-wavelet (Akbaryan and Bishnoi, 2001), steady-state-based approach (Chen and Howell, 2002), support vector machines (SVM) (Chiang et al., 2004), and PCA-QTA (qualitative trend analysis) (Maurya et al., 2003) have all been applied to the TE process. Most of the previous methods are based on multivariate statistics, and several studies have used nonlinear or dynamic models to consider process dynamics and nonlinearity (Chen and McAvoy, 1998). Although data driven methods show good diagnostic performance, they either assume that all the fault patterns are known a priori, or are inapplicable for unknown faults, which is unrealistic for practical systems operating in an uncertain environment,

The fault diagnosis framework (Chen et al., 2014) used in this paper is able to identify new faults by employing the incremental one-class learning approach in the model space. Learning in the model space (Chen et al., 2014) is naturally applicable to the current industrial MIMO system, and the framework is robust to imperfection in data/signal, such as missing values, high dimensionality, etc.

3. The framework of learning in the model space

This section introduces the recently proposed "learning in the model space" framework (Chen et al., 2014), which includes multiple-input and multiple-output (MIMO) signal simulated by deterministic reservoir models, and the learning stage using incremental one-class learning with the 'model distance' as the input features.

3.1. Learning in the model space

Recently, Chen et al. (2014) proposed to use deterministic reservoir computing (DRC) (Rodan and Tiňo, 2012) to represent MIMO signal segments and to use incremental one-class learning for fault diagnosis. Learning in the model space is to use models fitted on parts of data as more stable and parsimonious representations of the data. Learning is then performed directly in the model space, instead of the original data space.

Reservoir computing (RC) (Lukoševičius and Jaeger, 2009) is a class of state space models based on a "fixed" randomly constructed state transition mapping, realized through so-called reservoir and an trainable (usually linear) readout mapping from the reservoir.

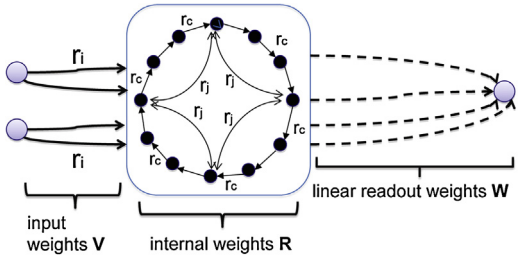

Fig. 1. Illustration of the deterministic reservoir model space. R is the reservoir weight matrix ($N \times N$), V is the input weight matrix ($N \times O$) to the reservoir, and O is the dimensionality of the time series.

DRC (Rodan and Tiňo, 2012) is a deterministic version of RC. The motivation to use DRC is because the traditional randomized RC is largely driven by a series of randomized model building stages, which could be unstable and difficult to understand, especially for fault diagnosis. Due to linear training, the DRC model can be trained fast and run in real-time.

Given the input signal $\mathbf{u}$ and output (target) signal $\mathbf{y}$, the reservoir model with N reservoir (state) units[1] is formulated as follows:

$$\mathbf{x}(t) = \tanh(R\ \ \mathbf{x}(t-1) + V\,\mathbf{u}(t)), \tag{1}$$

$$f(\mathbf{x}(t)) = W\mathbf{x}(t) + \mathbf{a}, \tag{2}$$

where $\mathbf{x}(t) = [x_1, \ldots, x_N]^T \in \Re^N$ is the state vector of reservoir activations, $\mathbf{u}(t)$ is the input signal at time t, R is the reservoir weight matrix ($N \times N$), V is the input weight matrix ($N \times O$) to the reservoir, O is the dimensionality of the time series, $\tanh(\cdot)$ is the state-transition function of the reservoir, $\mathbf{W}$ is the weight matrix ($O \times N$) from reservoir to output, and $f(\mathbf{x}(t))$ is the output of the linear readout from the reservoir. The state transition and output parts of the state space model are described by Eqs. (1) and (2), respectively.

This paper will focus on specific forms of ESN since they constitute one of the simplest, yet effective forms of RC. ESN has a "non-trainable" recurrent part ("the reservoir") (Eq. (1)) and a simple linear readout (Eq. (2)). Typically, the reservoir weights R and the input weights V to the reservoir are randomly generated so that the "Echo State Property" is satisfied. Loosely speaking, this means that the reservoir output would be independent of the initial conditions (Jaeger, 2001). Training of ESN can be efficiently performed through linear regression. For more details we refer the interested reader to e.g. Lukoševičius and Jaeger, 2009.

The downside of reservoir models is that their construction is largely driven by a series of randomized model building stages. Recently, Rodan and Tiňo (2012) proposed to use a simple deterministic constructed cycle reservoirs with regular jumps (DRC). This reservoir architecture has been shown to be comparable to (or better than) the traditional ESN on a wide variety of time series modeling and prediction tasks (Rodan and Tiňo, 2012). In DRC the reservoir nodes are connected in a uni-directional cycle with bi-directional shortcuts (jumps) (Fig. 1). All cyclic reservoir weights r_c have the same value; all jumps r_j share the same weight. This results in a simple, sparse and deterministically constructed reservoir coupling weight matrix R. Specifically, R is a very sparse matrix with r_c and r_j spreaded over, e. g. a network of 10 internal units with 2 as jump size, the matrix R is of the form as follows:

$$\begin{pmatrix}
0 & 0 & r_j & 0 & 0 & 0 & 0 & 0 & r_j & r_c \\
r_c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
r_j & r_c & 0 & 0 & r_j & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & r_c & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & r_j & r_c & 0 & 0 & r_j & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & r_c & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & r_j & r_c & 0 & 0 & r_j & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & r_c & 0 & 0 & 0 \\
r_j & 0 & 0 & 0 & 0 & 0 & r_j & r_c & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_c & 0
\end{pmatrix}$$

The input weight matrix V is highly constrained as well. The absolute weight values are the same r_i, the only difference is in the signs of individual weights, the sign pattern needs to be aperiodic. This has been analyzed empirically and theoretically in Rodan and Tiňo (2011, 2012). Algorithm 3.1 details how to train the reservoir.

Algorithm 3.1. Reservoir Training Algorithm

1: **Input:** Set of input signals $\mathbf{u}_1, \ldots, \mathbf{u}_T$; output signals $\mathbf{y}_1, \ldots, \mathbf{y}_T$; parameters (number of reservoir units N; DRC weights (r_c, r_j, r_i); ridge regression parameter λ (the parameter λ was chosen by cross validation).
2:**Output:** the trained reservoir weight $\mathbf{W}$.
3: Construct the input weight matrix V using r_i and the state transition matrix R using r_c and r_j.
4: **for** each time step t, $t = 1, \ldots, T$**do**
5: Drive the reservoir state evolution with the input sequence $\mathbf{x}(t) = \tanh(R\,\mathbf{x}(t-1) + V\mathbf{u}(t))$ (Eq. (1)).
6: **end for**
7: Construct the state matrix $\mathbf{X} = [\mathbf{x}(1); \ldots; \mathbf{x}(T)]$ and the output matrix $\mathbf{Y} = [\mathbf{y}(1); \ldots; \mathbf{y}(T)]$ by accumulating the state evolution and the output signal, respectively.
8: Given the linear readout mapping from the reservoir, the weight W is calculated by ridge regression, i.e. $W = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T Y$.

The main idea of "Learning in the Model Space framework" is that, provided the reservoir is able to represent a rich set of features of the input–output mapping, the model-based representation of the input–output mapping will be given by the linear readout mapping[2] $f(\mathbf{x}; \mathbf{u})$ operating on reservoir activations $\mathbf{x}$ to minimize the normalized mean square error (NMSE) between the model predictions $f(\mathbf{x}; \mathbf{u})$ and targets $\mathbf{y}(t)$.

In this framework, it is necessary to generate the model space from the original signal space. One possible way is to identify parameterized models with their parameter vectors and work in the parameter space. This, however, will make the learning highly dependent on the particular model parameterization used. A more satisfying approach is to use parameterization-free notions of distance or similarities between the models.

To simplify the notation, we will denote the readout $f(\mathbf{x}; \mathbf{u}_i)$ fitted to sequence $\mathbf{u}_i$ by $f_i(\mathbf{x})$ in this section. In the model space, the

[1] In this paper, the value of N is fixed to 100. Based on our previous results (Rodan and Tiňo, 2012), a larger N might lead to better performance. Instead of optimizing the number, we fix the number just to demonstrate that the algorithm can work well even with a non-optimal parameter N. In general, larger N would lead to larger computational overhead with small performance gain. As in other applications of ESN, The danger of over-fitting is minimal, since the only adaptive part is the linear readout (Lukoševičius and Jaeger, 2009).

[2] Note that in our formulation, the underlying dynamic reservoir will be the *same* throughout the signal – the differences in the signal characteristics at different times will be captured solely by the linear readout models.

m-norm distance between models $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ ($f_1, f_2 : \Re^N \to \Re^O$) is defined as follows:

$$L_m(f_1, f_2) = \left(\int_C D_m(f_1(\mathbf{x}), f_2(\mathbf{x}))\, d\mu(\mathbf{x}) \right)^{1/m},$$

where $D_m(f_1(\mathbf{x}), f_2(\mathbf{x})) = \|f_1(\mathbf{x}) - f_2(\mathbf{x})\|^m$ is a function to measure the difference between $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$, $\mu(\mathbf{x})$ is the probability density function of the input domain $\mathbf{x}$, and C is the integral range. In this paper, we adopt $m = 2$ and first assume that $\mathbf{x}$ is uniformly distributed. Of course, non-uniform $\mu(\mathbf{x})$ can be adopted as well by using samples generated from it or by estimating it directly using e.g. Gaussian mixture models.

In the following, we demonstrate the application of the distance definition in the model space for linear readout models of reservoir models. The readout model of reservoir computing can be represented by the following equation

$$f(\mathbf{x}) = W\mathbf{x} + \mathbf{a},$$

where $\mathbf{x} = [x_1, \ldots, x_N]^T$ is a reservoir state vector, N is the number of reservoir units in the model, W are the parameters ($O \times N$ matrix) in the model, O is the output dimensionality, and $\mathbf{a} = [a_1, \ldots, a_o] \in \Re^O$ is the bias vector of output nodes.

The distance between two readouts from the *same* reservoir can be calculated based on the follow equation (Chen et al., 2014):

$$L_2(f_1, f_2) = \left(\int_C \|W\mathbf{x}\|^2 + \|\mathbf{a}\|^2 d\mathbf{x} \right)^{1/2} = \left(\frac{2^N}{3} \sum_{j=1}^{N} \sum_{i=1}^{O} w_{i,j}^2 + 2^N \|\mathbf{a}\|^2 \right)^{1/2}$$

where $\mathbf{w}_i^T$ is the ith row of W, $w_{i,j}$ is the (i, j)th element of W, $f_1(\mathbf{x}) = W_1\mathbf{x} + \mathbf{a}_1$, $f_2(\mathbf{x}) = W_2\mathbf{x} + \mathbf{a}_2$, $W = W_1 - W_2$ and $\mathbf{a} = \mathbf{a}_1 - \mathbf{a}_2$.

Scaling of the squared model distance ($L_2^2(f_1, f_2)$) by 2^{-N} we obtain

$$\frac{1}{3} \sum_{j=1}^{N} \sum_{i=1}^{O} w_{i,j}^2 + \|\mathbf{a}\|^2,$$

which differs from the squared Euclidean distance of the readout parameters

$$\sum_{j=1}^{N} \sum_{i=1}^{O} w_{i,j}^2 + \|\mathbf{a}\|^2,$$

by the factor 1/3 applied to the differences in the linear part W of the affine readouts. Hence, more importance is given to the 'offset' than 'orientation' of the readout mapping.

The above analysis assumed that the distribution of $\mathbf{x}$ is uniform in the integral range C. When the distribution of $\mathbf{x}$ is non-uniform, sampling techniques and analytical techniques using e.g. a Gaussian mixture model can be employed to calculate the distance. Please refer to Chen et al. (2014) for details.

3.2. Incremental one class learning for cognitive fault diagnosis

In fault diagnosis, it is relatively cheap and simple to obtain measurements from a normally working system. In contrast, sampling from faulty situations requires the system to break down in various ways to obtain faulty measurement examples. Therefore, it will be very expensive, or completely impractical to construct a fault library.

We will follow (Chen et al., 2014) and employ the incremental one-class learning algorithm to identify unknown faults and construct a fault library dynamically, which will facilitate fault isolation based on this library. One-class classification (Schölkopf et al., 2001) is a special type of classification algorithm. It tries to learn from a training set containing only the points of one particular class, i.e. one class, to distinguish this particular class of points from all other possible classes.

The incremental one class learning is to use each one-class learner to represent each fault/sub-fault segment by using the "learning in the model space" approach. In the beginning, a normal one-class learner Θ_0 will be constructed based on the normal MIMO signal segments. When the sliding window is moving forward, the one-class learner Θ_0 will be applied to judge whether a fault occurs. If a fault, judged by Θ_0, is detected, we will train a new one-class-learner Θ_i to represent fault i. Then, we keep monitoring the signal and determine whether the ongoing signal segment belongs to either normal state or a known fault. If neither, a new one-class learner Θ_i will be built and included in the model library. The algorithm includes the following major steps:

1 Normal data preparation by applying deterministic reservoir model to the sliding windows in the first t steps, i.e. the "normal" regime is sequentially induced.
2 Calculate the pairwise model distance matrix $\mathbf{L}_2(f_i, f_j)$ and employ one class SVMs (OCS) to obtain the normal class Θ_0. In one class SVMs, Gaussian RBF kernel is employed with the data distance replaced by the *model distance* $L_2(f_i, f_j)$;

$$\phi_\sigma(f_i, f_j) = \exp\left\{ -\sigma \cdot L_2(f_i, f_j) \right\}.$$

3 With the sliding window moving forward, if a new f_j belongs to an existing model Θ_k,[3] update the existing Θ_k with this new data f_j, and empty the candidate pool.[4] Otherwise, put the "point" f_j in the candidate pool.
4 If the number of data points in the candidate pool exceeds half of the window size, construct a new one-class learner Θ_{k+1} and empty the candidate pool.

In the above description, the assumption is that the system is running normally in the first t steps. The window size m should be relatively large (e.g. >300 time steps) to accurately train the dynamic models (e.g. deterministic reservoir computing in this paper). The sliding window is moved forward by one step at a time, which can reduce fault detection delays.

4. Experimental studies

This section presents experimental results for Tennessee Eastman Challenge Process. This paper investigates fault detectability and fault isolationability using a number of approaches. In order to compare with previous methodologies, this section studies the supervised setting and the cognitive setting, respectively. The supervised setting assumes that we have plenty training data on various faults and the cognitive setting assumes that we do not know the fault data at all. The only assumption in the cognitive setting is that the system is running normally in the first several days. In this section, we also compare our cognitive approach with some other popular fault detection algorithms in Section 4.5.

4.1. Tennessee Eastman Challenge Process

TE process was proposed by Downs and Vogel (1993) to provide a realistic industrial process for evaluating process

[3] If the new point f_j is classified to more than one model by one-class SVMs, count the point in the last model due to sequential correlation.
[4] The candidate pool is used to save the "outlier points", which do not belong to existinig fault classes, until there are sufficient points to train a new one-class learner.

Table 1
Manipulated (input) variables of the TE process

Variable	Description	Variable	Description
XMV(1)	D feed flow (stream 2)	XMV(7)	Separator pot liquid flow (stream 10)
XMV(2)	E feed flow (stream 3)	XMV(8)	Stripper liquid product flow
XMV(3)	A feed flow (stream 1)	XMV(9)	Stripper steam valve
XMV(4)	Total feed flow (stream 4)	XMV(10)	Reactor cooling water flow
XMV(5)	Compressor recycle valve	XMV(11)	Condenser cooling water flow
XMV(6)	Purge valve (stream 9)	XMV(12)	Agitator speed

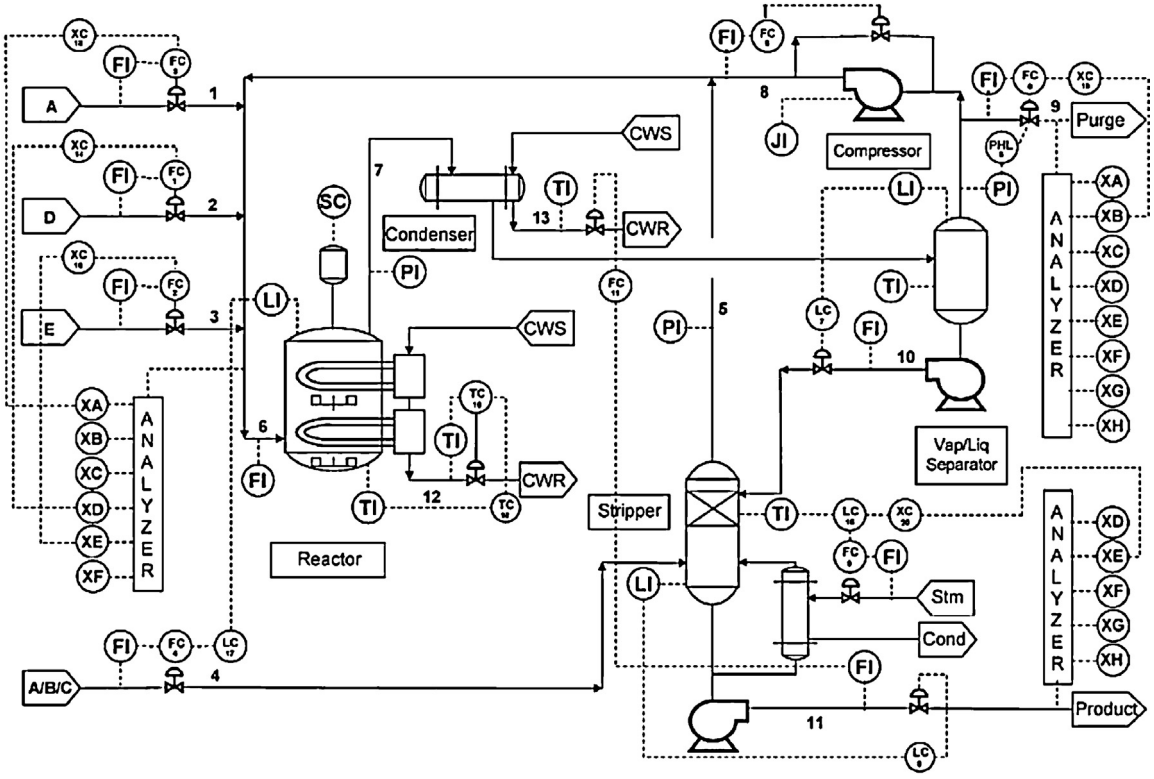

Fig. 2. Process flow of the TE process (Downs and Vogel, 1993).

control and monitoring methods. There are five major units, including a reactor, condenser, recycle compressor, vapor/liquid separator, and product stripper, and eight components, A–H (Fig. 2).

In TE process simulator,[5] we can simulate the process with different operating/initialization conditions, and inject faults at specific time. There are 12 manipulated (input) variables and 41 measured (output) variables (Tables 1 and 2). In the simulator, we simulate the system for 192 h. The system runs normally in the first 96 h, then we inject faults to the process from the beginning of 97 h to end. The sampling interval of 12 manipulated and 41 measured variables is three minutes. All the process measurements include Gaussian noise. In the TE process, 20 preprogrammed faults (Table 3) are included, 15 of which are known and 5 are unknown.

To our knowledge, there is no existing work on *cognitive fault diagnosis* on the TE process. All existing work on fault diagnosis on the TE process relies on the assumption that all the fault patters are known in advance. This paper *for the first time* investigates the cognitive fault diagnosis on the TE process.

4.2. Experimental settings

In our experiments, to evaluate the "learning in the model space" framework for fault diagnosis, two experimental settings, i.e. supervised setting and cognitive setting, are employed.

In the supervised setting, we aim to demonstrate the superiority of the model space representation compared with the signal space using a number of existing classifiers. Since many traditional fault diagnosis approaches are based on supervised learning, this setting can be employed to compare model space approaches with those traditional fault diagnosis approaches. In cognitive setting, we compare the incremental one class learning algorithm with other unsupervised algorithms using the model space representations. In addition, the ability to dynamically construct fault dictionary of our algorithm is demonstrated.

In both settings, the signal space is generated by selecting p consecutive points, i.e. $\{\mathbf{s}_t, \ldots, \mathbf{s}_{t+p-1}\}$, where $\mathbf{s}_t = (u_{t,1}, \cdots, u_{t,V}, y_{t,1}, \ldots, y_{t,O})^T$, us and ys are the inputs and outputs, respectively, as a training point by re-arranging these p points to one vector, and V is the dimensionality of the input signal. The order p will be selected in the range [1, 30].[6]

[5] Available at http://web.mit.edu/braatzgroup/TE_process.zip.

[6] The sampling interval of TE process is 3 min, and the upper bound of the lag equals to 90 min.

Table 2
Measured (output) variables of the TE process. The sampling interval is 3 min.

Variable	Description	Variable	Description
XMEAS(1)	A feed (stream 1)	XMEAS(22)	Condenser cooling water outlet temp
XMEAS(2)	D feed (stream 2)	XMEAS(23)	Composition of A (stream 6)
XMEAS(3)	E feed (stream 3)	XMEAS(24)	Composition of B (stream 6)
XMEAS(4)	Total feed (stream 4)	XMEAS(25)	Composition of C (stream 6)
XMEAS(5)	Recycle flow (stream 8)	XMEAS(26)	Composition of D (stream 6)
XMEAS(6)	Reactor feed rate (stream 6)	XMEAS(27)	Composition of E (stream 6)
XMEAS(7)	Reactor pressure	XMEAS(28)	Composition of F (stream 6)
XMEAS(8)	Reactor level	XMEAS(29)	Composition of A (stream 9)
XMEAS(9)	Reactor temp	XMEAS(30)	Composition of B (stream 9)
XMEAS(10)	Purge rate (stream 9)	XMEAS(31)	Composition of C (stream 9)
XMEAS(11)	Separator temp	XMEAS(32)	Composition of D (stream 9)
XMEAS(12)	Separator level	XMEAS(33)	Composition of E (stream 9)
XMEAS(13)	Separator pressure	XMEAS(34)	Composition of F (stream 9)
XMEAS(14)	Separator underflow (stream 10)	XMEAS(35)	Composition of G (stream 9)
XMEAS(15)	Stripper level	XMEAS(36)	Composition of H (stream 9)
XMEAS(16)	Stripper pressure	XMEAS(37)	Composition of D (stream 11)
XMEAS(17)	Stripper underflow (stream 11)	XMEAS(38)	Composition of E (stream 11)
XMEAS(18)	Stripper temperature	XMEAS(39)	Composition of F (stream 11)
XMEAS(19)	Stripper steam flow	XMEAS(40)	Composition of G (stream 11)
XMEAS(20)	Compressor work	XMEAS(41)	Composition of H (stream 11)
XMEAS(21)	Reactor cooling water outlet temp		

Table 3
Faults defined in the TE process (Downs and Vogel, 1993). The sampling interval is 3 min. The fault magnitudes are obtained by comparing the fault and normal data.

Fault ID	Description	Type	Magnitude
IDV1	A/C Feed ratio, B Composition constant (stream 4)	Step	203%
IDV2	B Composition, A/C Ratio constant (stream 4)	Step	105%
IDV3	D Feed temperature (stream 2)	Step	5%
IDV4	Reactor cooling water inlet temperature	Step	9%
IDV5	Condenser cooling water inlet temperature	Step	15%
IDV6	A Feed loss (stream 1)	Step	342%
IDV7	C Header pressure loss – reduced availability (Stream 4)	Step	25%
IDV8	A, B, C Feed composition (stream 4)	Random variation	736%
IDV9	D Feed temperature (stream 2)	Random variation	8%
IDV10	C Feed temperature (stream 4)	Random variation	112%
IDV11	Reactor cooling water inlet temperature	Random variation	567%
IDV12	Condenser cooling water inlet temperature	Random variation	8%
IDV13	Reaction kinetics	Slow drift	16%
IDV14	Reactor cooling water valve	Sticking	1285%
IDV15	Condenser cooling water valve	Sticking	5%
IDV16	Unknown	Random variation	78%
IDV17	Unknown	Random variation	557%
IDV18	Unknown	Step	57%
IDV19	Unknown	Random variation	73%
IDV20	Unknown	Random variation	310%

We generate 3000 time steps for normal signal and each fault signal, respectively, and employ a sliding window (size 500) to generate a series of signal segments, which are employed to train the deterministic reservoir model.

The fault detection ability is measured by fault detection rate (FDR) and false alarm rate (FAR). In fault isolation, the performance is measured by *precision*, *recall* (or sensitivity), and *specificity*. The *precision*, *recall* and *specificity* are defined as follows:

$$precision = \frac{tp}{tp+fp},$$
$$recall = \frac{tp}{tp+fn},$$
$$specificity = \frac{tn}{tn+fp},$$

where *tp*, *tn*, *fp*, *fn* indicate true positive, true negative, false positive, false negative, respectively. Their definitions are detailed as follow:

- true positive *tp*: fault signal correctly diagnosed as fault.
- true negative *tn*: normal signal correctly diagnosed as normal.
- false positive *fp*: fault signal incorrectly identified as normal.
- false negative *fn*: normal signal incorrectly identified as fault.

Based on these definitions, the fault detection (FD) rate is defined as $FD = (tp/tp + tn + fp + fn)$. The false alarm rate (FAR) is defined as $FD = (fn/tp + tn + fp + fn)$.

Precision measures the proportion of positive test results that are true positives, also referred to as positive predictive value. *Recall* measures the proportion of actual faults which are correctly

Table 4
Algorithms and parameters in supervised setting

Algorithm	Parameters
CART	–
NaiveBayes	–
Bagging	Number of trees: 100
Boosting	Number of trees: 100
SVM	σ Gaussian kernel parameter C Soft margin parameter
OCS	σ Gaussian kernel parameter ν The upper bound of outliers

Table 5
Comparisons of model space and signal space using supervised learning techniques in terms of fault detection rate (FDR) and false alarm rate (FAR). The ∗ means the difference between model and signal representations is statistically significant. The reported results are based on 10 runs of 5-fold cross validation.

	CART		Bagging		Boosting		SVM		OCS	
	Model	Signal	Model	Signal	Model	Signal	Model	Signal	Model	Signal
FDR	99.00	87.27∗	99.85	93.36∗	96.09	92.05∗	99.99	94.36∗	93.37	49.74∗
FAR	0.16	0.15	0.04	0.07∗	0.07	2.14∗	0.14	0.29∗	0.87	0.14∗

Table 6
Comparisons of model space and signal space using supervised learning techniques for each fault in terms of fault detection rate (details of Table 5). The reported results are based on 10 runs of 5-fold cross validation.

	CART		Bagging		Boosting		SVM		OCS	
	Model	Signal	Model	Signal	Model	Signal	Model	Signal	Model	Signal
IDV1	99.87	99.52	100	100	100	100	100	100	100	100
IDV2	99.59	99.80	100	100	99.75	100	100	100	100	100
IDV3	89.44	9.84	97.04	0.00	70.00	58.11	99.86	66.36	24.53	2.56
IDV4	97.84	99.72	100	100	95.31	100	100	100	99.59	4.52
IDV5	97.52	89.51	100	99.85	80.91	99.16	100	100	44.03	8.41
IDV6	99.95	100	100	100	100	100	100	100	100	100
IDV7	99.73	99.61	100	100	98.15	100	99.99	100	100	47.54
IDV8	100	98.89	100	100	100	99.31	100	99.93	100	99.89
IDV9	99.31	58.43	100	83.52	94.19	67.10	100	77.38	99.20	2.25
IDV10	99.70	87.93	100	99.28	98.77	86.56	100	95.06	100	14.27
IDV11	98.88	94.47	100	99.91	98.97	98.94	100	86.67	100	4.72
IDV12	100	98.86	100	100	100	99.89	100	99.23	100	99.41
IDV13	99.93	99.56	100	100	99.96	100	100	99.13	100	100
IDV14	99.78	99.30	100	100	99.96	100	100	99.83	100	88.24
IDV15	99.19	57.97	100	86.60	93.17	66.38	100	84.72	100	2.73
IDV16	99.59	86.69	100	99.34	95.88	79.09	100	83.15	100	6.58
IDV17	99.68	98.99	100	100	99.47	99.96	100	100	100	60.94
IDV18	99.90	99.69	100	100	99.96	99.96	100	100	100	99.58
IDV19	99.40	79.39	100	100	99.01	95.34	100	97.36	100	35.61
IDV20	99.80	87.32	100	98.71	98.35	91.15	100	98.43	100	17.53

identified as such and *specificity* measures the proportion of normal data which are correctly identified.

4.3. Supervised setting

In the supervised setting, we demonstrate the superiority of the model space compared with the signal space using a number of existing classifiers. Since many traditional fault diagnosis approaches use supervised learning, this setting can be employed to compare model space representation with those traditional fault diagnosis approaches.

Table 5 reports the comparisons of the representations of model space and signal space using a number of supervised learning algorithms for fault detection ability, including classification and regression trees (CART), naive Bayes, support vector machines (SVMs), one class support vector machine (OCS), Bagging (100 trees) and Adaboosting (100 trees).

Since the default setting of MATLAB is to optimize the classification and regression trees (CART) algorithm,[7] we follow the default setting in MATLAB for CART. Bagging and Adaboosting are ensemble algorithms with decision trees (CARTs) as based learners (CARTs have been optimized by MATLAB). They have only one parameter[8] to specify, i.e. the number of trees in the ensembles. We are using a popular choice (100 decision trees) in our comparisons. The parameters of SVMs and one-class SVMs are optimized by 5-fold cross validation. The parameters used in supervised setting are reported in Table 4.

The reported results in Tables 5–7 are based on 10 runs of 5-fold cross validation. Tables 5 and 6 report the performance for fault detection, and Table 7 reports the performance for fault isolation.

Based on these tables, the model space representation usually achieves statistically significantly better results. SVM achieves the best performance for fault detection rate (99.99%) and Bagging achieves the smallest false alarm rate (0.04%).

In OCS, it distinguishes one class of points from all other possible points by learning from a training set containing only the points of that class. It means that only the partial data, i.e. 'normal' class, is employed. Therefore, OCS achieves the worst performance in terms of FDR (93.37%) and FAR (0.87%) in the model space.

Table 7 reports the fault isolation ability in both model and signal spaces. The performance is measured by *Precision*, *recall* (or sensitivity) and *specificity*. Based on this table, Bagging and SVM outperform other classifiers in terms of three metrics. Adaboost.M2 seems to overfit the noise and leads to inferior results.

All these results demonstrate that the model space representation of the signal scan achieve better classification performance than using the original signal representation with the same kind of classifiers, which confirms the benefits to use the model space rather than the signal space in fault diagnosis.

4.4. Cognitive setting

To evaluate the incremental one class learning for cognitive fault diagnosis, in this setting we compare with other unsupervised algorithms in the model space. In addition, the ability to dynamically construct fault dictionary of our algorithm is demonstrated.

In cognitive setting, we do not know which data are faulty and the fault classes. The only assumption in the cognitive setting is that the initial part of the signal is normal. In this setting, we employ some clustering algorithms to compare with our proposed

[7] In matlab function 'classregtree', the default is to compute the full tree and the optimal sequence of pruned subtrees.
[8] Different variants of Bagging and Adaboosting may require more parameters.

Table 7
Comparisons of fault isolation by using the model space and signal space using supervised learning techniques. The ∗ means the difference between model and signal representations is statistically significant. The reported results are based on 10 runs of 5-fold cross validation.

	Precision		Recall		Specificity	
	Model	Signal	Model	Signal	Model	Signal
CART	97.74	66.78∗	97.72	66.07∗	99.88	98.21∗
SVM	99.82	67.54∗	99.82	66.41∗	100	98.23∗
NaiveBayes	93.60	43.49∗	91.93	45.65∗	99.58	97.14∗
Bagging	99.98	80.51∗	99.98	79.63∗	100	98.93∗
Adaboost.M2	29.96	15.40∗	36.79	21.77∗	96.67	95.88∗

algorithm. These algorithms include affinity propagation (AP) (Frey and Dueck, 2007), Kmeans, agglomerative hierarchical cluster tree (Hclustering), landmark-based spectral clustering (LSC) (Chen and Cai, 2011), and statistic test based approach "Hotelling's *T*-squared statistic test (T2)" (Cho et al., 2005), one class SVMs (Schölkopf et al., 2001) (only for fault detection). Table 8 summaries all the algorithms and their parameters employed in this paper. The parameters of one class SVMs will be optimized by 5-fold cross validation using the first 500 data points.

The reported results in Tables 9–11 are based on 10 runs of 5-fold cross validation. Tables 9 and 10 report the performance for fault detection, and Table 11 reports the performance for fault isolation in both model and signal spaces.

Based on these tables, the model space representation usually achieves statistically significantly better results. The cognitive approach achieves the best performance for fault detection rate (97.98%) and lowest false alarm rate (0.00%). In the model space, each data point is a series of signal segments selected using a sliding window. Therefore, the two data points could be generated with overlapping signal segments, which indicates they are not independent. Therefore, T2 achieves the inferior performance in the model space than in the signal space. The clustering algorithms are all very good in terms of false alarm rate in both model and signal spaces. This might indicate that the normal signal segments within the sliding window of TE process in both model and signal space form easily identified clusters. These clustering algorithms achieve much better performance in the model space than in the signal space in terms of fault detection rate, indicating that the clusters representing various faults are easily identified in the model space.

Table 11 reports the fault isolation ability in both model and signal spaces. In this table, we also report the true number of classes and the discovered classes (i.e. number of faults plus normal class) using a number of algorithms for each data set. Since in the cognitive setting, there is no prior information about faults, the clustering algorithms always discover more classes than the actual number of faults by decomposing each true fault into several subclusters. Since the number of discovered faults does not equal to the true number of faults, we compare each true cluster and these discovered clusters and merge those discovered clusters with the maximizing overlap with each true cluster to a pseudo-cluster. The performance metrics are obtained by comparing true clusters and merged clusters. Based on this table, the cognitive algorithm can achieve a very good performance in these three metrics and significantly outperform other algorithms. As a base algorithm, kmeans generates inferior results in all three metrics.

Tables 1–10 have reported the proposed approach on a common MIMO system with 20 programmed noise and disturbances defined as IDV1 to IDV20, which are called 'faults' in this paper. These disturbances include several types, including step (IDV1–IDV7), random variation (IDV8–IDV12), slow drift (IDV13), sticking (IDV14–IDV15), and some unknown types (IDV16–IDV20).

Based on Table 10, the cognitive approach performs well on most of the faults, i.e. achieving 100% fault detection rates, except on step disturbances: IDV1 (97.12%), IDV2 (76.95%), and unknown disturbance IDV17 (97.12%). It seems that the cognitive approach is robust against random disturbances IDV8-IDV12 (noise).

According to Tables 3, 6 and 10, it can be observed that small fault magnitudes often lead to relatively small fault detection rates. For example, with IDV3 (5%), IDV4 (9%) and IDV15 (5%), the signal based algorithms always have very small fault detection rates, e.g. CART (9.84) with IDV3, OCS (4.52) with IDV4 in Table 10, and AP (26.75) with IDV15. However, the model space based approaches are relatively robust to small fault magnitudes. For example, CART performed much better in model space (89.44) than in signal space (9.84) with IDV3; the same as OCS (99.59 vs. 4.52) with IDV4 and AP (100 vs. 26.75) with IDV15. This is understandable since model space based approaches focus on 'model/function change' with disturbance while signal space based approaches focus more on 'signal change', which could be easily disturbed by imposed faults.

All these results including fault detection/isolation confirm the benefits of using the model space rather than the signal space in fault diagnosis.

4.5. Benchmark comparisons

In the literature, most of the algorithms focus on fault detection approaches available for TE process. Therefore, we will compare our *cognitive* approach with some popular fault detection algorithms for the TE process.

Table 8
Algorithms and Parameters in Cognitive Setting

Algorithm	Parameters
T2	–
AP	–
Kmeans	Number of clusters: 50
Hclustering	Number of clusters: 50
LSC	Number of clusters: 50
OCS	σ Gaussian kernel parameter
	ν The upper bound of outliers
Cognitive	σ Gaussian kernel parameter
	ν The upper bound of outliers
	N Number of nodes in reservoir (100)

Table 9
Comparisons of the model space and signal space in terms of fault detection rate (FDR) and false alarm rate (FAR) using cognitive learning techniques. The ∗ means the difference between model and signal representations is statistically significant. The reported results are based on 10 runs of 5-fold cross validation.

	FDR (%)		FAR (%)	
	Model	Signal	Model	Signal
T2	39.24	46.12∗	10.00	10.00
OCS	94.41	51.85∗	8.08	2.7∗
Kmeans	84.29	44.29∗	0.00	0.00
Hclustering	92.00	20.25∗	0.00	0.00
LSC	91.55	48.02∗	0.00	0.00
AP	96.37	69.18∗	0.00	0.00
cognitive	97.98	60.96∗	0.00	0.00

Table 10
Comparisons of the model space and signal space for each fault in terms of fault detection rate using "cognitive" learning techniques (details of Table 9). The reported results are based on 10 runs of 5-fold cross validation.

	T2		OCS		Kmeans		Hclustering		LSC		AP		Cognitive	
	Model	Signal	Model	Signal	Model	Signal	Model	Signal	Model	Signal	Model	Signal	Model	Signal
IDV1	97.98	100	100	100	0	6.01	0.00	6.01	0.00	6.01	100	100	97.12	12.71
IDV2	100	100	100	100	40.06	0.00	40.06	0.00	40.06	0.00	100	83.23	76.95	12.71
IDV3	16.33	9.60	30.26	6.60	100	0.00	100	0.00	100	0.00	100	83.03	100	10.57
IDV4	29.01	10.86	100	8.15	100	93.99	100	93.99	100	93.99	100	83.03	100	99.22
IDV5	3.75	9.70	58.12	6.79	59.94	100	100	6.01	100	100	100	99.80	100	91.27
IDV6	0.86	100	100	100	84.05	86.23	100	0.10	100	83.41	100	75.45	100	100
IDV7	35.54	20.08	100	48.30	59.94	6.01	100	0.00	100	5.23	100	98.80	100	17.94
IDV8	25.46	93.50	100	99.90	94.81	0.00	100	0.00	100	0.10	100	22.36	100	34.92
IDV9	17.10	7.76	100	4.46	100	0.00	100	0.00	100	0.10	100	39.32	100	8.73
IDV10	37.46	16.49	100	19.01	100	83.41	100	4.56	100	77.01	100	27.15	100	96.90
IDV11	0.19	10.57	100	8.63	100	94.18	100	0.29	100	96.80	100	94.81	100	100
IDV12	75.70	93.21	100	100	75.12	68.77	100	0.00	99.04	100	100	100	100	97.09
IDV13	30.45	97.87	100	100	100	3.88	100	0.00	100	6.01	100	86.83	100	8.63
IDV14	0.86	20.66	100	90.11	100	0.00	100	0.00	100	0.10	100	36.93	100	19.69
IDV15	29.88	9.31	100	4.07	85.98	45.49	100	0.00	93.76	63.24	100	26.75	100	91.27
IDV16	82.52	9.80	100	10.28	87.70	97.87	100	93.99	98.08	98.93	100	64.27	100	97.09
IDV17	100	53.25	100	65.47	98.27	6.01	100	6.01	100	6.01	100	100	97.21	67.99
IDV18	0.00	100	100	100	100	0.00	100	0.00	100	27.16	100	30.54	100	73.81
IDV19	83.67	43.84	100	40.06	100	93.99	100	93.99	100	96.22	100	31.54	100	99.13
IDV20	18.06	15.91	100	25.32	100	100	100	100	100	10	100	99.80	100	100

Table 11
Comparisons of the model space and signal space in terms of fault isolation ability using cognitive learning techniques. The $*$ means the difference between model and signal representations is statistically significant. The reported results are based on 10 runs of 5-fold cross validation.

	Precision		Recall		Specificity		nclass	
	Model	Signal	Model	Signal	Model	Signal	Model	Signal
Kmeans ($K=50$)	67.88	43.82*	70.21	42.53*	98.47	97.14*	50	50
Hclustering	76.13	24.30*	80.04	19.28*	99.03	95.96*	50	50
LSC	76.92	36.05*	76.58	37.66*	98.82	96.91*	50	50
AP	89.10	51.54*	90.15	51.81*	99.51	97.59*	453	923
Cognitive	98.58	48.90*	97.15	41.00*	99.86	97.10*	606	139

Table 12
Comparisons of our cognitive approach with other approaches in terms of fault detection rate (FDR) and false alarm rate (FAR).

	Cognitive	PCA	DPCA	ICA	MICA	FDA	PLS	TPLS	MPLS	SAP
FDR	97.98	73.79	82.07	80.97	80.37	79.57	82.60	84.39	84.39	80.54
FAR	0.00	6.38	15.13	2.63	1.5	6.38	7.12	12.13	10.75	1.25

These compared algorithms include principle component analysis (PCA), dynamic PCA (DPCA) (Ku et al., 1995) to deal with autocorrelation of process variable, independent component analysis (ICA), modified ICA (MICA) (Lee et al., 2006; Zhang and Zhang, 2010), fisher discriminant analysis (FDA) (Chiang et al., 2004), partial least squares (PLS), total projection to latent structure (TPLS) approach (Zhou et al., 2010), modified approach (MPLS) (Yin et al., 2011), and subspace aided approach (SAP) (Ding et al., 2009). Based on the settings of (Yin et al., 2012), all the parameters of those compared algorithms are optimized.

In this comparison, we follow the same methodology as previous experiments in this paper. That is, we simulate the system for 192 h. The system in the first 96 h is normal and the system in the rest hours is faulty. The sampling interval is three minutes. Based on the simulated signals, 3000 time steps for normal signal and each fault signal are generated, respectively.

The results are reported in Table 12. Based on these results, our approach outperforms all other algorithms. All these results confirm the benefits of our algorithm in cognitive fault diagnosis for the TE process.

5. Conclusion

In this paper, we introduce a recently-proposed framework "learning in the model space" to diagnose faults in the Tennessee Eastman Process. Instead of conducting fault diagnosis in the signal space, this paper investigates "learning in the model space" framework that represents the multiple-input and multiple-output data as a series of models trained using the signal segments within a sliding window. By investigating the characteristic of these trained models using a learning approach in the model space, we can identify and isolate faults effectively.

This paper for the first time investigates TE process without prior knowledge of the fault numbers and types based on the cognitive fault diagnosis. This paper also studies the strategy to dynamically construct fault dictionary in real time. The extensive experiments and comparisons with other algorithms validate the effectiveness of the proposed approach for the TE process.

In the cognitive setting, the number of discovered faults is often larger than the true number of faults due to the absence of information about faults. Our future work will focus on this issue and

try to minimize the difference between the number of discovered faults and the number of true faults.

Acknowledgments

Huanhuan Chen was supported by the National Natural Science Foundation of China under Grants 61203292, 61311130140 and the One Thousand Young Talents Program. Xin Yao was supported by a Royal Society Research Merit Award. This work was supported by the European Union Seventh Framework Programme under grant agreement No. INSFO-ICT-270428.

References

Akbaryan F, Bishnoi P. Fault diagnosis of multivariate systems using pattern recognition and multisensor data analysis technique. Comput Chem Eng 2001;25(9):1313–39.
Barakat M, Druaux F, Lefebvre D, Khalil M, Mustapha O. Self adaptive growing neural network classifier for faults detection and diagnosis. Neurocomputing 2011;74(18):3865–76.
Chen X, Cai D. Large scale spectral clustering with landmark-based representation. In: Proceedings of the twenty-fifth AAAI conference on artificial intelligence (AAAI'11); 2011. p. 313–8.
Chen J, Howell J. Towards distributed diagnosis of the tennessee eastman process benchmark. Control Eng Pract 2002;10(9):971–87.
Chen G, McAvoy T. Predictive on-line monitoring of continuous processes. J Process Control 1998;8(5):409–20.
Chen J, Patton RJ. Robust model-based fault diagnosis for dynamic systems. Norwell, Massachusetts, USA: Kluwer Academic Publishers; 1999.
Chen H, Tiňo P, Rodan A, Yao X. Learning in the model space for cognitive fault diagnosis. IEEE Trans Neural Networks Learning Syst 2014;25(1):124–36.
Chiang L, Kotanchek M, Kordon A. Fault diagnosis based on fisher discriminant analysis and support vector machines. Comput Chem Eng 2004;28(8):1389–401.
Cho J, Lee J, Wook Choi S, Lee D, Lee I. Fault identification for process monitoring using kernel principal component analysis. Chem Eng Sci 2005;60(1):279–88.
Ding S, Zhang P, Naik A, Ding E, Huang B. Subspace method aided data-driven design of fault detection and isolation systems. J Process Control 2009;19(9):1496–510.
Downs JJ, Vogel EF. A plant-wide industrial process control problem. Comput Chem Eng 1993;17(3):245–55.
Frey B, Dueck D. Clustering by passing messages between data points. Science 2007;315(5814):972–6.
Gertler JJ. Fault detection and diagnosis in engineering systems. Boca Raton, Florida, USA: CRC Press; 1998.
Huang Y, Gertler J, McAvoy T. Sensor and actuator fault isolation by structured partial PCA with nonlinear extensions. J Process Control 2000;10(5):459–69.
Jaeger H. The echo state approach to analysing and training recurrent neural networks. Tech. rep. German National Research Center for Information Technology; 2001.
Kankar P, Sharma S, Harsha S. Fault diagnosis of ball bearings using machine learning methods. Expert Syst Applicat 2011;38(3):1876–86.
Kano M, Nagao K, Hasebe S, Hashimoto I, Ohno H, Strauss R, Bakshi B. Comparison of statistical process monitoring methods: application to the eastman challenge problem. Comput Chem Eng 2000;24(2):175–81.
Kassidas A, Taylor P, MacGregor J. Off-line diagnosis of deterministic faults in continuous dynamic multivariable processes using speech recognition methods. J Process Control 1998;8(5):381–93.
Ku W, Storer R, Georgakis C. Disturbance detection and isolation by dynamic principal component analysis. Chemometr Intell Lab Syst 1995;30(1):179–96.
Lee J, Qin S, Lee I. Fault detection and diagnosis based on modified independent component analysis. AIChE J 2006;52(10):3501–14.
Lin W, Qian Y, Li X. Nonlinear dynamic principal component analysis for on-line process monitoring and diagnosis. Comput Chem Eng 2000;24(2):423–9.
LukošEvičIus M, Jaeger H. Reservoir computing approaches to recurrent neural network training. Comput Sci Rev 2009;3(3):127–49.
Maurya M, Rengaswamy R, Venkatasubramanian V. Qualitative trend analysis of the principal components: application to fault diagnosis. Comput Aided Chem Eng 2003;15:968–73.
Palade V, Bocaniala C. Computational intelligence in fault diagnosis. London, UK: Springer Publishing Company, Incorporated; 2010.
Polycarpou M, Helmicki A. Automated fault detection and accommodation: a learning systems approach. IEEE Trans Syst Man Cybernet 1995;25(11):1447–58.
Raich A, Cinar A. Multivariate statistical methods for monitoring continuous processes: assessment of discrimination power of disturbance models and diagnosis of multiple disturbances. Chemometr Intell Lab Syst 1995;30(1):37–48.
Raich A, Cinar A. Diagnosis of process disturbances by statistical distance and angle measures. Comput Chem Eng 1997;21(6):661–73.
Rodan A, Tiňo P. Minimum complexity echo state network. IEEE Trans Neural Networks 2011;99:1–14.
Rodan A, Tiňo P. Simple deterministically constructed cycle reservoirs with regular jumps. Neural Comput 2012;24(7):1822–52.
Schölkopf B, Platt J, Shawe-Taylor J, Smola A, Williamson R. Estimating the support of a high-dimensional distribution. Neural Comput 2001;13(7):1443–71.
Vemuri A, Polycarpou M. Neural-network-based robust fault diagnosis in robotic systems. IEEE Trans Neural Networks 1997;8(6):1410–20.
Venkatasubramanian V, Rengaswamy R, Kavuri S, Yin K. A review of process fault detection and diagnosis: Part III: process history based methods. Comput Chem Eng 2003;27(3):327–46.
Yélamos I, Escudero G, Graells M, Puigjaner L. Performance assessment of a novel fault diagnosis system based on support vector machines. Comput Chem Eng 2009;33(1):244–55.
Yan X, Edwards C. Nonlinear robust fault reconstruction and estimation using a sliding mode observer. Automatica 2007;43(9):1605–14.
Yin S, Ding S, Zhang P, Hagahni A, Naik A. Study on modifications of PLS approach for process monitoring. In: Proceedings of the 18th IFAC world congress, vol. 18; 2011. p. 12389–94.
Yin S, Ding S, Haghani A, Hao H, Zhang P. A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process. J Process Control 2012;22(9):1567–81.
Zhang Y, Zhang Y. Fault detection of non-gaussian processes based on modified independent component analysis. Chem Eng Sci 2010;65(16):4630–9.
Zhang X, Polycarpou M, Parisini T. A robust detection and isolation scheme for abrupt and incipient faults in nonlinear systems. IEEE Trans Automat Control 2002;47(4):576–93.
Zhang X, Parisini T, Polycarpou M. Sensor bias fault isolation in a class of nonlinear systems. IEEE Trans Automat Control 2005;50(3):370–6.
Zhou D, Li G, Qin S. Total projection to latent structures for process monitoring. AIChE J 2010;56(1):168–78.