Chemometrics and Intelligent Laboratory Systems 159 (2016) 31–44

Contents lists available at ScienceDirect

Chemometrics and Intelligent Laboratory Systems

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

Developments of two supervised maximum variance unfolding algorithms for process classification

Chihang Wei[a], Junghui Chen[b,*], Zhihuan Song[a,*]

[a] State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou, 310027 Zhejiang, China
[b] Department of Chemical Engineering, Chung Yuan Christian University, Chungli, Taoyuan 32023, Taiwan, ROC

ARTICLE INFO

Keywords:
Dimension reduction
Maximum variance unfolding
Process classification
Semi-definite optimization

ABSTRACT

Maximum variance unfolding (MVU) has recently proven to be a powerful dimension reduction method for nonlinear data with numerous mutually correlated measured variables. However, in classification work, MVU performs poorly since it is an unsupervised method without considering class information of data. In this paper, two novel supervised maximum variance unfolding (SMVU) algorithms, SMVU1 and SMVU2 are developed respectively. They extend MVU to supervised methods. Both SMVU1 and SMVU2 not only aim to find the embeddings that unfold the manifold in the reduced dimensional space but also group the within-class samples together and separate between-class samples. In SMVU1, between-class manifold structures are defined by the class separation constraints, and within-class manifold structures to be unfolded are defined by the objective function; in SMVU2, between-class manifold structures to be unfolded and within-class manifold structures to be folded are put together in the objective function. Additionally, a novel kernel function approximation algorithm is developed based on the Nyström method to handle new samples. The effectiveness of the proposed methods are illustrated through a simple nonlinear system and a real industrial polyethylene process. The study results show the proposed SMVUs significantly outperform the conventional MVU in classification work.

1. Introduction

Automation in the operating refineries and chemical plants is consistently increasing as witnessed during the last few decades because of their large processing rates and complex configurations of unit operations [1]. Moreover, it is needed because of the increasing demand on consistent product quality and good process performance, lack of adequately skilled labor and the necessity to lower costs in order to keep pace with rapidly growing global competition. In the modern plants with the extensive use of distributed control systems, process data have become abundant and the measured variables are often characterized by high dimensions. However, most of these dimensions are unnecessary and there are often severe dependencies in the variables because of a set of constraints, like mass and energy balances, or operating policies [2–9]. This implies that the independent variables or the important information among these data typically lie in a low dimensional manifold. This means that only a few intrinsic variables are needed to characterize the data variations and dimension reduction is necessary to handle this kind of data. With these features, multivariate statistical process monitoring (MSPM) approaches have been proposed to learn the data structure, to reduce the dimension and to extract significant interests for performing supervisory tasks such as process monitoring, fault detection and diagnostics [2–9].

Principal component analysis (PCA) [10] is one of the earliest papers on dimension reduction in MSPM. Although PCA has been proven to be useful in the process monitoring, the linear assumption limits its applicable area and performance [11]. The kernel technique has also been applied. It can extend traditional linear dimension reduction methods to other nonlinear dimension reduction methods. Kernel PCA (KPCA) [11,12] was proposed to generalize PCA to the nonlinear case. However, the algorithm maps the high dimensional data onto a lower dimensional space without considering the manifold structure because the performance of KPCA largely depends on the kernel function, but appropriate selection of kernel functions has been sporadically discussed in the research literature. Unlike KPCA defined by an artificially determined kernel function, maximum variance unfolding (MVU) was proposed and it can automatically learn the kernel space from the input data instead. It has been proven to be a powerful dimension reduction method for nonlinear data with numerous mutually correlated measured variables [13,14]. Ideally, MVU represents the intrinsic dimension of the data. More importantly, the boundary of the distribution region of the training samples in the input

* Corresponding authors.
E-mail addresses: jason@wavenet.cycu.edu.tw (J. Chen), songzhihuan@zju.edu.cn (Z. Song).

http://dx.doi.org/10.1016/j.chemolab.2016.09.009
Received 2 March 2016; Received in revised form 27 September 2016; Accepted 29 September 2016
Available online 30 September 2016

space is faithfully preserved. These features facilitate its process monitoring applications [15–17].

MVU can learn the manifolds bottom up from the topology of the input data, but the data representatives on the learned manifold are not guaranteed to have desired properties such as classification. MVU only projects the raw data onto the manifold with lowest dimension without preserving the data topology at all when training samples with several classes are applied, the unfolded manifold structure of each class may be mixed together. Obviously, given this manifold, no linear classifier can easily separate the data samples of different classes since MVU is an unsupervised method. These features limit the applications of MVU in classification work while the MVU kernel can only be expected to perform well for large margin classification if the decision boundary on the unfolded manifold is approximately linear. There is no a priori reason, however, to expect this type of linear separability for different classes.

In the past, as plant operators and engineers often spent a substantial amount of time and efforts before the classes could be properly diagnosed, several algorithms for the automatic process classification had been developed based on the classification objective and data topology [7,18–21]. The major motivation of this paper is to develop two types of supervised MVUs. Supervised learning is the machine learning task of inferring a "machine" from the labeled training data. In supervised learning, each sample is a pair consisted of a set of input variables and a corresponding desired output value. A supervised learning algorithm analyzes the training data and produces an inferred "machine". The machine can be used for mapping new examples which are not the original training data to correctly determine the class labels. In this paper, the unsupervised MVU modeling is extended to two supervised ones, including Supervised Maximum Variance Unfolding 1 (SMVU1) and Supervised Maximum Variance Unfolding 2 (SMVU2). The purpose of the algorithms is to discover the natural boundary from the given data with the label annotations. Both algorithms aim at obtaining the separable unfolding of the low dimension and preserving the data structure at the same time. They are developed respectively based on the maximum variance embedding objective used in the existing semi-definite programming (SDP) algorithm [22] to unfold data and pull different class data apart. In SMVU1, between-class manifold structures are defined by the class separation constraints, and within-class manifold structures to be unfolded are defined by the objective function; in SMVU2, between-class manifold structures to be unfolded and within-class manifold structures to be folded are put together in the objective function. But the two optimization problems (SMVU1 and SMVU2) are not convex. A simplified optimization by reformulating the SMVU1 and SMVU2 problems in terms of the elements of the inner product matrix is developed. Then the computation kernel matrix is irrelevant to the dimension of the data. Despite the favorable properties of the kernel methods in terms of theory, empirical performance and flexibility, the constructed kernel matrix is valid for the training samples only; it cannot handle the new samples. In this work, the effectiveness of the Nyström method to scale kernel regression is used to get the approximate kernel function [23–25]. Then SMVU1 and SMVU2 algorithms learning from the labeled (or known) classes are developed. They can predict the unknown classes using the Bayesian classifier [26]. SMVU1 and SMVU2 use different types of constraints and objective functions respectively to construct between-class and within-class manifold structures. They have different scopes of applications. They will be described and discussed in detail later in this paper.

The rest of the paper is organized as follows. Section 2 gives the background knowledge of the MVU algorithm. Then the detailed SMVU1 and SMVU2 algorithms are discussed in Section 3; the differences among MVU, and SMVU1 and SMVU2 are also discussed in this section. Next a kernel approximation method is given in Section 4. Section 5 contains the SMVU1 and SMVU2 based process classification using the Gaussian naive classifier. Also, case studies of a simple nonlinear system and an industrial problem are presented to evaluate the proposed SMVU1 and SMVU2 algorithms in Section 6. Finally, conclusions are made.

2. Revisit of maximum variance unfolding

Before discussing the proposed algorithms, the basic background knowledge and concepts of MVU are introduced in this section. A manifold is a mathematical surface that behaves linear locally. Representing such data in its raw high dimensional-form is not necessary. Euclidean distances are only meaningful on a very local scale, and it is very hard to handle the whole data. Ideally one would like to have a representation that matches the intrinsic dimension of the data so that Euclidean distances can be globally meaningful.

MVU, also known as semi-definite embedding, has recently been proposed as a special variation of KPCA utilized for nonlinear dimension reduction. The basic idea of MVU is based on the finding that the high dimensional data lie on a low dimensional manifold. The kernel matrix of KPCA, $\mathbf{K}$, is obtained by projecting the input data onto a higher dimensional feature space. The projection can be done by subjectively specifying a certain kernel function. Unlike KPCA, MVU directly constructs a kernel matrix from training samples so that the data manifold in the input space can be unfolded in the kernel feature space $\mathbf{\Phi}$ implicitly defined by $\mathbf{K}$. This also makes the manifold unfolded in the reduced space of MVU [13,14]. The problem of learning $\mathbf{K}$ is casted to an instance of SDP [22,27–29]. It is convex and it does not suffer from local optima. In particular, given an input set $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N, \mathbf{x}_n \in R^D$ and an unfolding space $\mathbf{Y} = \{\mathbf{y}_n = \mathbf{\Phi}(\mathbf{x}_n)\}_{n=1}^N$, $\mathbf{y}_n \in R^d$ where $d < D$, one considers a map $\mathbf{\Phi}$: $\mathbf{x} \to \mathbf{y}$ so that the outputs $\mathbf{Y}$ can be found and the inputs and the learned outputs are k-locally isometric, or at least approximately isometric. Here N is the number of samples while D and d are the dimensions of the input and the learned manifolds. Thus, the objective function is to unfold a manifold based on the observations, where any "fold" between two samples on a manifold serves to decrease the Euclidean distance between them. To unfold the manifold in $\mathbf{\Phi}$, an objective function that measures the sum of pairwise squared distances between the outputs $\{\mathbf{y}_n = \mathbf{\Phi}(\mathbf{x}_n)\}_{n=1}^N$ is maximized:

$$\max \Gamma = \max \frac{1}{2N} \sum_{ij} \|\mathbf{\Phi}(\mathbf{x}_i) - \mathbf{\Phi}(\mathbf{x}_j)\|^2 \tag{1}$$

By maximizing Eq. (1), the outputs are pulled as far apart as possible subject to some constraints, including isometry and centering.

- Isometry: This constraint is to preserve local manifold structure in the kernel space. The isometry between the discrete point sets can be translated into various sets of equality constraints on the inputs and the outputs. Let $\mathbf{S} \subset R^{N \times N}$ be a binary adjacency matrix which can tell whether there is an edge between $\mathbf{x}_i$ and $\mathbf{x}_j$ formed by pairwise connecting all the k-nearest neighbors. Thus, $\{\mathbf{x}_n\}_{n=1}^N$ and $\{\mathbf{y}_n = \mathbf{\Phi}(\mathbf{x}_n)\}_{n=1}^N$ are locally isometric if $\mathbf{x}_i$ and $\mathbf{x}_j$ are themselves neighbors ($\mathbf{S}_{ij} = 1$) or common neighbors of another point in the data set ($[\mathbf{S}^T\mathbf{S}]_{ij} = 1$). The local isometry constraints can be written as

$$\|\mathbf{\Phi}(\mathbf{x}_i) - \mathbf{\Phi}(\mathbf{x}_j)\|^2 = \|\mathbf{x}_i - \mathbf{x}_j\|^2 = D_{ij} \text{ for all } (i, j) \text{ with } \mathbf{S}_{ij} = 1 \text{ or } [\mathbf{S}^T\mathbf{S}]_{ij} = 1 \tag{2}$$

- Centering: the centering constraint,

$$\sum_i \mathbf{\Phi}(\mathbf{x}_i) = 0 \tag{3}$$

is also imposed to remove a translational degree of freedom from the final solution.

The optimization problem is to maximize the variance of the outputs $\{\mathbf{\Phi}(\mathbf{x}_n)\}_{n=1}^N$ (Eq. (1)) subject to the constraints that they are

locally isometric to the inputs $\{\mathbf{x}_n\}_{n=1}^N$ (Eq. (2)) and centered on the origin (Eq. (3)). But the optimization as stated above is not convex, as it involves maximizing a quadratic form subject to quadratic equality constraints. For simple optimization, the problem is reformulated in terms of the elements of the inner product matrix, $\mathbf{K}_{ij} = \mathbf{\Phi}(\mathbf{x}_i)^T\mathbf{\Phi}(\mathbf{x}_j)$. With the terms on the right-hand side of Eq. (1) expanded and the outputs centered on the origin, now the objective function in Eq. (1) can be directly expressed in terms of the inner product matrix $\mathbf{K}_{ij}$ of the outputs $\mathbf{\Phi}(\mathbf{x}_i)$ and $\mathbf{\Phi}(\mathbf{x}_j)$,

$$\Gamma = \frac{1}{2N}\sum_{ij}\|\mathbf{\Phi}(\mathbf{x}_i) - \mathbf{\Phi}(\mathbf{x}_j)\|^2 = \frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{N}(\mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij}) \tag{4}$$

The isometry constraints in Eq. (2) can also be easily expressed in terms of these matrix elements. The local isometry constraints can be rewritten as

$$\|\mathbf{\Phi}(\mathbf{x}_i) - \mathbf{\Phi}(\mathbf{x}_j)\|^2 = \mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2 = D_{ij} \tag{5}$$

Likewise, the centering constraint in Eq. (3) can be conducted and expressed in terms of these inner products:

$$0 = \left\|\sum_n \mathbf{\Phi}(\mathbf{x}_n)\right\|^2 = \sum_{ij}\mathbf{\Phi}(\mathbf{x}_i)^T\mathbf{\Phi}(\mathbf{x}_j) = \sum_{ij}\mathbf{K}_{ij} \tag{6}$$

Substituting Eq. (6) into Eq. (4), the objective function can be finally expressed by:

$$\Gamma = \frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{N}(\mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij}) = \frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{N}(\mathbf{K}_{ii} + \mathbf{K}_{jj}) = Tr(\mathbf{K}) \tag{7}$$

Note that both Eqs. (5)–(6) are linear equality constraints on the elements of $\mathbf{K}_{ij}$. Also, one must further constrain the optimization to the cone of symmetric positive semi-definite matrices ($\mathbf{K} \geq 0$) [22]. Thus, combined with the positive semi-definite constraint on the matrix, the above optimization problem is formulated as the SDP problem [22,27–29],

$$\begin{aligned} &\max_{\mathbf{K}} Tr(\mathbf{K}) \\ &s.t. \\ &\mathbf{K} \geq 0 \\ &\textstyle\sum_{ij}\mathbf{K}_{ij} = 0 \\ &\mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij} = D_{ij} \text{ for all } (\mathbf{x}_i, \mathbf{x}_j) \text{ with } \mathbf{S}_{ij} = 1 \text{ or } [\mathbf{S}^T\mathbf{S}]_{ij} = 1 \end{aligned} \tag{8}$$

The problem is guaranteed to be feasible because the constraints are trivially satisfied by the inner product matrix of the inputs. There are many papers on efficiently solving SDPs and on a number of general-purpose toolboxes. The results in this work are obtained using CSDP toolbox [28].

As a special variation of KPCA, MVU is a remarkable dimension reduction method for the nonlinear data with numerous mutually correlated measured variables. To clarify the features of MVU, the scores of the sample data among PCA, KPCA and MVU in the low dimension space are explained. The example of a data set sampled from Swiss-roll manifold in Fig. 1(a) is given [14]. The samples are colored in a gradient ramp. The results of applying PCA, KPCA (based on the Gaussian kernel function) and MVU to the dataset are shown in Fig. 1(b), (c) and (d), respectively, where the samples in the original space and the corresponding scores of PCA, KPCA and MVU in the low dimension space are marked in the same colors. As the samples in the original space are colored in the gradient ramp, if the algorithm (PCA, KPCA or MVU) applied to unfolding the global manifold structure and scores in the low dimension space should also be colored in the same gradient ramp, the samples in the local manifold structure are

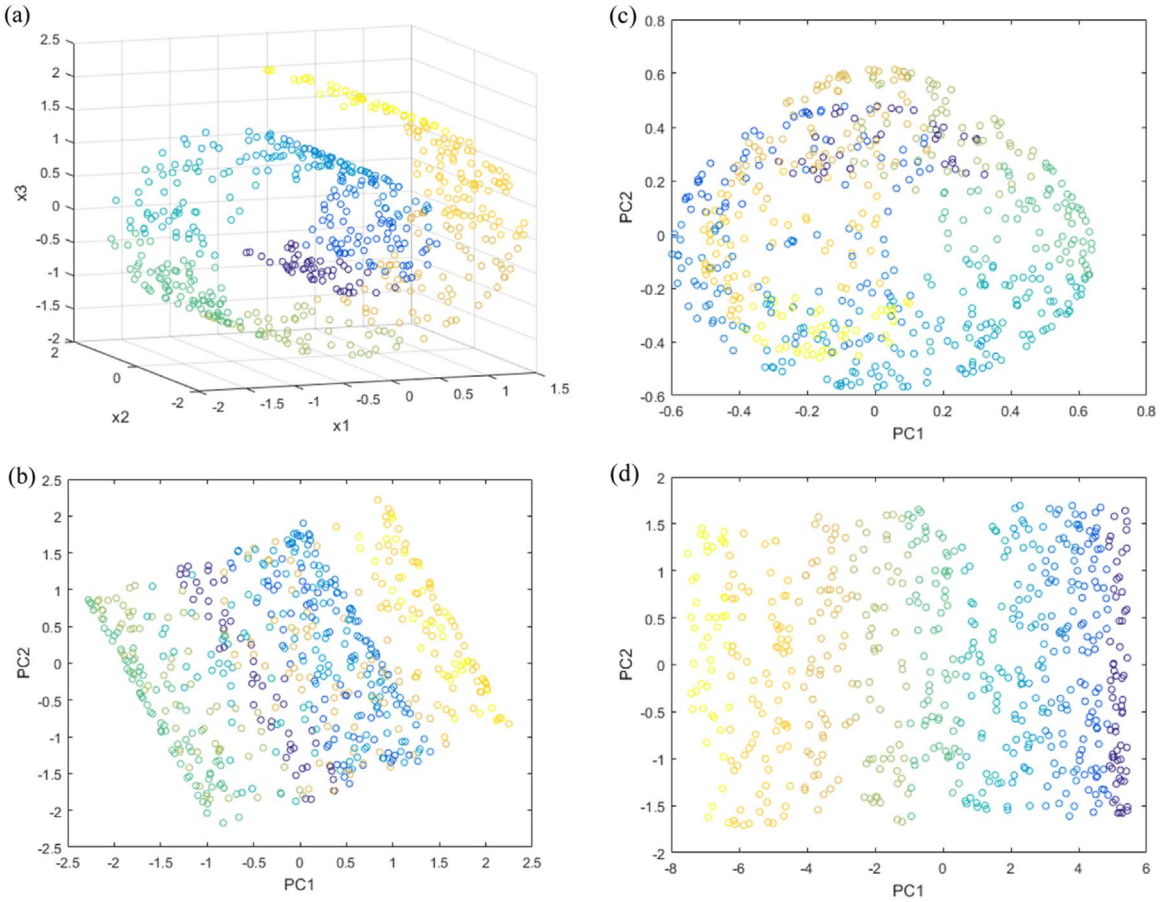

Fig. 1. (a) Swiss-roll manifold, colored in the gradient ramp, and the corresponding scores after applying (b) PCA, (c) KPCA, and (d) MVU.

preserved successfully and satisfactorily. Different colored samples in Fig. 1(b) and (c) are overlapped while the samples in Fig. 1(d) are still colored in the gradient ramp. Thus, compared with PCA and KPCA, MVU performs better at representing the intrinsic dimension of the data. The boundary of the distribution region of the training samples is faithfully preserved in the MVU kernel space. These features facilitate its applications in dimension reduction and the process monitoring [15–17].

MVU can learn the manifolds bottom up from the topology of the input data, but the data representatives on the learned manifold are not guaranteed to have desired properties such as classification. MVU only projects the raw data onto the manifold with the lowest dimension without preserving the data topology at all. Thus, when the training samples with several classes are applied, the unfolded manifold structure of each class may be mixed together. To further show the problem of MVU in the classification work, the example of Swiss-roll manifold [14] in Fig. 2(a) is allocated as there are two types of classes in the dataset. The results of applying MVU to the dataset are shown in Fig. 2(b). The underlying manifold structure is formed by connecting each input to its k nearest neighbors' graph (here $k = 4$). Obviously, given this manifold, the samples of different classes in Fig. 2(b) would be not easily separated as MVU is an unsupervised method [13,14]. These features limit the applications of MVU when it comes to classification. The MVU kernel can only be expected to perform well when the classes are far away from each other, the decision boundary on the unfolded manifold is approximately linear. There is no a priori reason, however, to expect this type of the linear separability for different classes.

3. Supervised maximum variance unfolding

Classification is a common goal for many machine learning and patter recognition problems. And a large number of algorithms are proposed to achieve better classification performance for various applications [7,18–21]. These algorithms can also be regarded as the manifold learning algorithms though their goal is to learn a new manifold for best separation between classes, the low dimension as well as the local geometry preservation.

When MVU is applied to the training samples of several classes, different classes in the unfolded manifold structure may be mixed together and be inseparable because MVU is an unsupervised method. Two new supervised MVU algorithms termed SMVU1 and SMVU2 are proposed in this work. The intuition concept of the algorithms is to discover the natural boundary from the given data with the label annotations. Both algorithms aim at obtaining the unfolding, which is separable, of low dimension and preserving the data structure at the same time. Specifically, the algorithms of SMVU1 and SMVU2 are quite different: in SMVU1, new constraints are introduced and they aim at pulling different class data apart with manifold unfolding still conducted in each class; in SMVU2, a new objective is to make data of different classes as far as possible and to minimize the unfolding manifold of the same classes while its constraints remain the same as MVU's.

3.1. SMVU1

Because MVU does not preserve the data topology at all, the supervised MVU algorithm is proposed to obtain an unfolding manifold

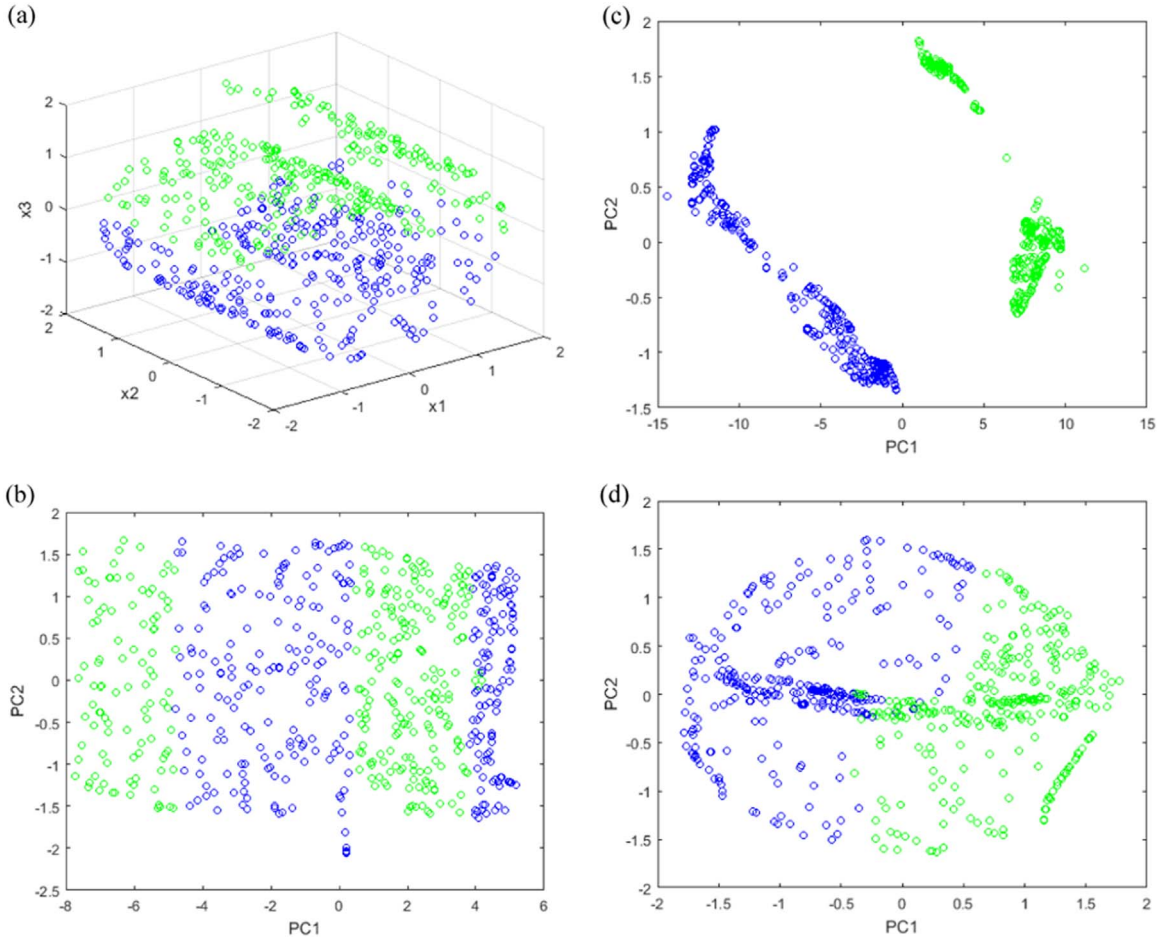

Fig. 2. (a) In Swiss-roll manifold, there are two classes labelled in green circle and in blue circle, and the learned manifolds of (b) MVU, (c) SMVU1, and (d) SMVU2. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$\{\mathbf{y}_n = \mathbf{\Phi}(\mathbf{x}_n)\}_{n=1}^N$, $\mathbf{y}_n \in R^d$ from $\{\mathbf{x}_n\}_{n=1}^N$, $\mathbf{x}_n \in R^D$, where their annotations of all the samples are $\mathbf{c} = \{c_n\}_{n=1}^N$, $c_n \in \{1 \ \cdots \ C\}$, and C is the number of classes. The algorithm is to preserve the input data structure while simultaneously allowing for classification with a high margin. The main goal is to find a low dimensional manifold that captures the geometrical constraints among the neighboring samples. The neighboring samples of the same class would be put together while those of different classes with annotations would be separated. To achieve such a goal, a simple way is to add new constraints to enlarge the distances between different classes and modify the objective to unfold manifold structure in each class in the kernel feature space.

To keep the manifold in the lowest dimension for the data with the same class, like Eq. (5) of MVU, the locally isometric constraints for the same class can be written as

$$\mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij} = D_{ij} \tag{9}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are themselves neighbors ($\mathbf{S}_{ij} = 1$) or common neighbors of another point in the data set ($[\mathbf{S}^T\mathbf{S}]_{ij} = 1$), and they are in the same class.

To pull class scatters, class locations should be constrained. Because each class is homogenous and compact with respect to certain characteristics and each class should be different from the other classes with respect to the same characteristics, the simple way is to select the representative point from each class and to add the constraints to the representative points among the classes. The representative point of each class is calculated by

$$\mathbf{x}_c = \min_{\mathbf{x}_n \in G_c} \|\mathbf{x}_n - \overline{\mathbf{x}}_c\|^2 \tag{10}$$

where $\overline{\mathbf{x}}_c = \frac{1}{N_c}\sum_{n\in G_c}\mathbf{x}_n$ is the mean of all the data which belong to the set G_c in class c. N_c is the number of observations in the class c. Note that $\overline{\mathbf{x}}_c$ does not belong to the training samples; as a result, $\mathbf{\Phi}(\overline{\mathbf{x}}_c)$ cannot be transformed to an element of kernel matrix $\mathbf{K}$. The data point $\mathbf{x}_c$ is then used as an approximate representative point of class c. Thus, the training data for each class have been stacked into the data set $\{\mathbf{x}_{c,n}\}_{n=1}^{N_c}$ before calculating $\overline{\mathbf{x}}_c$, where $\mathbf{x}_{c,n}$ is the vector of the measurement variables for the nth observation at class$c$. The distances between $\mathbf{\Phi}(\mathbf{x}_{c_i})$, $(c_i = 1, \cdots, C)$ and $\mathbf{\Phi}(\mathbf{x}_{c_j})$ $(c_j \neq c_i)$ are encouraged to be pulled away, which can be formally written in a mathematical formula as

$$\text{Class 1: } \begin{cases} \|\mathbf{\Phi}(\mathbf{x}_{c_1}) - \mathbf{\Phi}(\mathbf{x}_{c_2})\|^2 = \mathbf{K}_{c_1c_1} + \mathbf{K}_{c_2c_2} - 2\mathbf{K}_{c_1c_2} = \alpha^2\|\mathbf{x}_{c_1} - \mathbf{x}_{c_2}\|^2 \\ \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ \|\mathbf{\Phi}(\mathbf{x}_{c_1}) - \mathbf{\Phi}(\mathbf{x}_{c_C})\|^2 = \mathbf{K}_{c_1c_1} + \mathbf{K}_{c_Cc_C} - 2\mathbf{K}_{c_1c_C} = \alpha^2\|\mathbf{x}_{c_1} - \mathbf{x}_{c_C}\|^2 \end{cases} \tag{11}$$

$$\text{Class 2: } \begin{cases} \|\mathbf{\Phi}(\mathbf{x}_{c_2}) - \mathbf{\Phi}(\mathbf{x}_{c_3})\|^2 = \mathbf{K}_{c_2c_2} + \mathbf{K}_{c_3c_3} - 2\mathbf{K}_{c_2c_3} = \alpha^2\|\mathbf{x}_{c_2} - \mathbf{x}_{c_3}\|^2 \\ \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ \|\mathbf{\Phi}(\mathbf{x}_{c_2}) - \mathbf{\Phi}(\mathbf{x}_{c_C})\|^2 = \mathbf{K}_{c_2c_2} + \mathbf{K}_{c_Cc_C} - 2\mathbf{K}_{c_2c_C} = \alpha^2\|\mathbf{x}_{c_2} - \mathbf{x}_{c_C}\|^2 \end{cases} \tag{12}$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

$$\text{Class } C\text{: } \|\mathbf{\Phi}(\mathbf{x}_{c_{C-1}}) - \mathbf{\Phi}(\mathbf{x}_{c_C})\|^2 = \mathbf{K}_{c_{C-1}c_{C-1}} + \mathbf{K}_{c_Cc_C} - 2\mathbf{K}_{c_{C-1}c_C} = \alpha^2 \|\mathbf{x}_{c_{C-1}} - \mathbf{x}_{c_C}\|^2 \tag{13}$$

where $\alpha > 1$ can be adjusted to preserve the good data topology.

Considering different classes, the inner product matrix for each class is separately calculated,

$$\Gamma_c = \frac{1}{2}\sum_{i=1}^{N_c}\sum_{j=1}^{N_c}(\mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij}), \quad \mathbf{x}_i, \mathbf{x}_j \in G_c \tag{14}$$

Thus, the objective function which measures the weighted sum of the inner product matrix of each class is defined as

$$\Gamma = \sum_{c=1}^{C}\frac{\Gamma_c}{N_c} \tag{15}$$

where the weight $\frac{1}{N_c}$ is to balance different quantities of samples from different classes.

After the new objective function and constraints of the above optimization are expressed in terms of the inner product matrix for different classes, SMVU1 optimization problem can be written as

$$\begin{aligned} &\max_{\mathbf{K}} \ \textstyle\sum_{c=1}^{C}\frac{\Gamma_c}{N_c} \\ &s.\ t. \\ &\mathbf{K} \geq 0 \\ &\textstyle\sum_{ij}\mathbf{K}_{ij} = 0 \\ &\begin{cases} \mathbf{K}_{c_ic_i} + \mathbf{K}_{c_{i+1}c_{i+1}} - 2\mathbf{K}_{c_ic_{i+1}} = \alpha^2\|\mathbf{x}_{c_i} - \mathbf{x}_{c_{i+1}}\|^2 \\ \mathbf{K}_{c_ic_i} + \mathbf{K}_{c_Cc_C} - 2\mathbf{K}_{c_ic_C} = \alpha^2\|\mathbf{x}_{c_i} - \mathbf{x}_{c_C}\|^2 \end{cases} \text{for all } c_i, \\ &c_{i+1}, \cdots, c_C \text{ and } i = 1, \cdots, C \\ &\mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij} = D_{ij} \text{ for all } (\mathbf{x}_i, \mathbf{x}_j) \text{ with } \mathbf{S}_{ij} = 1 \text{ or } [\mathbf{S}^T\mathbf{S}]_{ij} = 1, \text{ and} \\ &\mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are in the same class} \end{aligned} \tag{16}$$

3.2. SMVU2

The dimension reduction is especially important when the dimension of the observation space is large while the numbers of observations in the classes are relatively small. The MVU approach to dimension reduction was discussed in the Section 2. Although MVU contains certain optimality properties which find unfolding manifold, it is not well suited for classification because it does not take into account the information between the classes when determining the lower-dimensional representation. A good subspace for classification is the well separated class-means being measured related to the sum of the variances of the data assigned to a particular class. Thus, the goal of SMVU2 is to find a kernel matrix to maximize the "between-class" scatter while minimizing the "within-class" scatter for similar samples in the unfolding manifold.

To accomplish SMVU2, various indices that quantify the scatter within classes and the scatter between classes should be defined first. With class c defined as the set of vectors $\mathbf{\Phi}(\mathbf{x}_n)$ which belong to the class c, the within scatter σ_c^w for class c is

$$\sigma_c^W = \frac{1}{N_c}\sum_{\mathbf{x}_n \in G_c}\|\mathbf{\Phi}(\mathbf{x}_n) - \mathbf{\Phi}(\mathbf{x}_c)\|^2 \tag{17}$$

Like SMVU1, the representative point from class c is used here instead of the mean $\overline{\mathbf{x}}_c$. Then, the measurement of the overall within-class scatter can be calculated by

$$\sigma^W = \sum_{c=1}^{C}\sigma_c^W \tag{18}$$

σ_c^W can be easily expressed in terms of the combination of elements of $\mathbf{K}$ and be obtained by

$$\sigma_c^W = \frac{1}{N_c}\sum_{\mathbf{x}_n \in G_c}(\mathbf{K}_{nn} + \mathbf{K}_{cc} - 2\mathbf{K}_{cn}) \tag{19}$$

where the weight $\frac{1}{N_c}$ is used to balance different quantities of samples from different classes, $\mathbf{K}_{cc} = \mathbf{\Phi}(\mathbf{x}_c)^T\mathbf{\Phi}(\mathbf{x}_c)$ and $\mathbf{K}_{cn} = \mathbf{\Phi}(\mathbf{x}_c)^T\mathbf{\Phi}(\mathbf{x}_n)$.

The measurement of between-class scatter in unfolding manifold is calculated as

$$\sigma^B = \frac{1}{C}\sum_{c_i, c_j}\|\mathbf{\Phi}(\mathbf{x}_{c_i}) - \mathbf{\Phi}(\mathbf{x}_{c_j})\|^2 \tag{20}$$

The conventional FDA is based on the sum of squares of the differences between the mean of each class and the total mean [20].

Unlike the conventional FDA, the sum of squares of the differences among the representative points is used here, because the mean of each class and the total mean do not belong to the original training samples. The between-class scatter would not be expressed in terms of the combination of elements of $\mathbf{K}$. Now the between-class scatter in Eq. (20) can be expressed in terms of these inner products as

$$\sigma^B = \sum_{c_i,c_j} \mathbf{K}_{c_i c_i} + \mathbf{K}_{c_j c_j} - 2\mathbf{K}_{c_i c_j} \tag{21}$$

The objective of SMVU2 is to maximize the scatter between classes while minimizing the scatter within classes,

$$\Gamma = C(\sigma^B - \sigma^W) \tag{22}$$

Instead of $\Gamma = \sigma^B/\sigma^W$ used in FDA, the difference of the scatter between classes and the scatter within classes is used in SMVU2. Because σ^B/σ^W cannot be transformed to linear combinations of elements of $\mathbf{K}$, the objective function based on σ^B/σ^W is not a convex function of $\mathbf{K}$. It is a quasi-convex function of $\mathbf{K}$. Its corresponding optimization problem is more computationally complex than Eq. (22).

After the new objective function and constraints of the above optimization are expressed in terms of the inner product matrix for different classes, SMVU2 optimization problem can be written as

$$\begin{aligned} &\max_{\mathbf{K}} C(\sigma^B - \sigma^W) \\ &s.t. \\ &\mathbf{K} \geq 0 \\ &\textstyle\sum_{ij} \mathbf{K}_{ij} = 0 \\ &\mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij} = D_{ij} \text{ for all } (\mathbf{x}_i, \mathbf{x}_j) \text{ with } \mathbf{S}_{ij} = 1 \text{ or } [\mathbf{S}^T\mathbf{S}]_{ij} = 1 \end{aligned} \tag{23}$$

Fig. 2(c) and (d) also show the results of applying the proposed SMVU1 and SMVU2 to the 3D Swiss-roll dataset. Unlink unsupervised MVU, SMVU1 can find a manifold where the classes are linearly separable; however, the data structure is a little destroyed. SMVU2 reveals a low dimensional space with the preserved local distance structure, but some data among the classes are a little overlapped. As shown in Fig. 2(c) and (d), the proposed SMVU1 and SMVU2 algorithms have the best classification performance.

[NOTE 1] The concepts of SMVU1 and SMVU2 are the same; that is, they are used to pull different class data apart for classification and to unfold the manifold for the dimension reduction. The conditions of SMVU1 and SMVU2 are functionally different. They are summarized as follows

1. Pulling different class data apart: Class separation constraints (Eqs. (11)–(13)) in SMVU1 are used while the objective with the unfolding manifold (Eq. (17)) in SMVU2 is applied to maximizing the between-class variances. The former is the hard constraints which can absolutely separate different classes; the latter, soft constraints whose separability counts on the optimal objective function.
2. Isometry constraints: In SMVU1, isometry constraints (Eq. (9)) act only on the neighboring samples from the same class; Like MVU, isometry constraints (Eq. (23)) in SMVU2 act on the neighboring samples from the whole data.
3. Between-class and within-class manifold structures: In SMVU1, between-class manifold structures are defined by the class separation constraints (Eqs. (11)–(13)) and within-class manifold structures to be unfolded are defined by the objective function (Eq. (15)); in SMVU2, between-class manifold structures to be unfolded and within-class manifold structures optimized to be folded are put together in the objective function (Eq. (22)).

Thus, SMVU1 has better performance in the dimension reduction as the hard constraints in specific locations are used and the objective function aims at unfolding the manifold, but its generalization ability is unwarrantable when the margins of different classes are close; on the other hand, SMVU2 is developed based on the soft constraints. It has the ability to avoid the problem of close margins of different classes, but the performance of the dimension reduction is worse because the objective function aims at reducing dimension, unfolding between-class manifold structures and folding within-class manifold structures. Therefore, SMVU1 is good for the classification with highly certain data while SMVU2 is able to avoid high uncertainty. To measure how noise affects classification, the uncertainty index (UI) which is the ratio of noise level to system level is defined as

$$\text{UI}=\max_i \left[\frac{\text{Noise}_i}{\text{Std}_i} \right] \tag{24}$$

where $i = 1, \cdots, D$ and D is the number of variables; the numerator, Noise_i, and the denominator, Std_i, represent the estimated noise and the standard deviation of variable i. The noise level can be estimated from the collected measured variables [30,31].

[NOTE 2] All the constraints of SMVU1 are rigorous equality constraints. In practice, the measurements with noise are unavoidable. The noise seems to be modeled in many different ways, but the modeling is not easy because the system is unknown with bad signals and it varies rapidly. It is meant to obtain the coarseness, or lack of smoothness, of the estimated manifold structure. It is often desirable to relax the rigorous equality constraints, so the distance between samples only serves as a rough estimate of neighborhood relationship and the local manifold structure is often approximately preserved instead of being strictly preserved. Therefore, the resulting inner product kernel matrix tends to have fewer numbers of principal eigenvalues and the resulting dimension-reduction outputs tend to have equal or even greater variance. Hence, the optimization with relaxed constraints can be seen as a variant for more aggressive forms of classification and dimension reduction [13,14]. One way to relax constraints is to use inequalities (Eq. (9)) instead of rigorous equalities, i.e.

$$\mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij} \leq D_{ij} \text{ for all } (\mathbf{x}_i, \mathbf{x}_j) \text{ with } \mathbf{S}_{ij} = 1 \text{ or } [\mathbf{S}^T\mathbf{S}]_{ij} = 1 \tag{25}$$

and this allows distances between samples to shrink, but not to grow. Please note Eq. (25) should not be applied to SMVU2 problem to prevent the concentration of the data with the same class in a narrow region without unfolding the data. Another way to relax constraints (Eq. (9)) is to allow the distance to both shrink and grow slightly, but to add a term to the objective function to penalize this change of distance. In this paper, the former one to relax constraints is selected even though the later one is more flexible and supposed to obtain better results, because some extra computations are required using the later one.

The proposed SMVU1 or SMVU2, unlike the conventional Gaussian based latent variable models, are manifold learning procedures, casted on semi-definite programming (SDP) problems. They only require a topological space that locally resembles the Euclidean space near each point. (More precisely, each point of a D-dimensional manifold has a neighbourhood that is homeomorphic to the Euclidean space of dimension D.) [32,33]. This condition is weaker than the normal distribution needed in the conventional Gaussian based latent variable models. Thus, one does not need to test the normality of the applied data.

4. Kernel function approximation

The unfolding manifold is based on the kernel-based algorithms. Kernel methods rely solely on similarity measures between pairs of data points, namely inner products. Although the kernel methods allow us to create an unfolding based on the local geometric information of the original data, these approaches do not know how to treat new data points which do not come from the original samples. Thus, it is desirable to obtain the data-dependent kernel functions. They can be applied to new data points without repeating the whole training algorithms. Although the power of some algorithms stems from the

ability to replace the standard inner product with some other kernel functions, the kernel function parameters cannot be easily determined.

In this work, the out-of-sample extrapolation method which has been used in MVU [23–25] is also used to construct the data-dependent kernels. Specifically, the method approximates the kernel eigenfunction using Gaussian basis functions via generalized Nyström, and then it obtains a data-dependent kernel function. The Nyström formula [25] is a general method for learning kernel eigenfunctions based on the prediction of the eigenvector value of a new data point and on the convergence of the eigenvalues of a matrix. Using the generalized Nyström formula, the approximated pth scaled eigenfunction $f_{p,N}(\mathbf{x})$ of $\mathbf{K}$ is

$$f_{p,N}(\mathbf{x}) = \sum_{n=1}^{N} b_{pn} r(\mathbf{x}, \mathbf{x}_n) \tag{26}$$

with weights $\mathbf{b}_p = [b_{p1}, \ldots, b_{pN}]^T = (\mathbf{R} + \lambda \mathbf{I})^{-1} \boldsymbol{\nu}_p$, where $\boldsymbol{\nu}_p$ denotes the pth eigenvector of $\mathbf{K}$. Matrix $\mathbf{R}$ denotes the Gram matrix for $r(\cdot,\cdot)$ on all the training samples with elements $\mathbf{R}_{ij} = r(\mathbf{x}_i, \mathbf{x}_j)$, and $\lambda \mathbf{I}$ is the regularization term that handles well the ill-conditioned matrix. Then, the corresponding approximated data-dependent kernel function is expressed as

$$\begin{aligned} \kappa(\mathbf{x}_i, \mathbf{x}_j) &= \textstyle\sum_p l_p f_{p,N}(\mathbf{x}_i) f_{p,N}(\mathbf{x}_j) \\ &= \mathbf{r}(\mathbf{x}_i)^T (\mathbf{R} + \lambda \mathbf{I})^{-1} \mathbf{K} (\mathbf{R} + \lambda \mathbf{I})^{-1} \mathbf{r}(\mathbf{x}_j) \end{aligned} \tag{27}$$

where $\mathbf{r}(\mathbf{x}_i)$ denotes a column vector $\mathbf{r}(\mathbf{x}_i) = [r(\mathbf{x}_i, \mathbf{x}_1), \cdots, r(\mathbf{x}_i, \mathbf{x}_N)]^T$, and $r(\mathbf{x}_i, \mathbf{x}_j)$ is a traditional kernel function. Usually, $r(\mathbf{x}_i, \mathbf{x}_j)$ is determined to be Gaussian RBF-kernel function$r(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\rho)$ as it is robust to parameter variations and it has infinite degrees of freedom.

Given the kernel matrix obtained from SMVU1 and SMVU2, a quadratic error criterion is used to determine a suitable approximated data-dependent kernel function,

$$\{\rho^*, \lambda^*\} = \arg\min_{\rho,\lambda} \sum_{i=1}^{N} \sum_{j=1}^{N} (\mathbf{K}_{ij} - \kappa(\mathbf{x}_i, \mathbf{x}_j))^2 \tag{28}$$

In this case, the accuracy of approximation uniquely depends on the training samples. However, samples with noise represent a highly undesirable and dreaded part of any data. In Eq. (28), when ρ and λ are close to zero, it is implied that $r(\mathbf{x}_i, \mathbf{x}_j)$ tends to be one at 1, $\mathbf{x}_i = \mathbf{x}_j$ or zero at $\mathbf{x}_i \neq \mathbf{x}_j$, and $\mathbf{R}$ becomes an identity matrix; as a result, $\sum_{i=1}^{N} \sum_{j=1}^{N} (\mathbf{K}_{ij} - \kappa(\mathbf{x}_i, \mathbf{x}_j))^2$in Eq. (28) is zero. In this case, when a new testing sample which is similar to one of the training samples but noise-contaminated is applied, $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ would always be equal to zero because the training sample is not exactly the same as the testing sample ($\mathbf{x}_i \neq \mathbf{x}_j$). This implies that the trained $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ is only good for training samples but not for testing samples and the trained kernel matrix is severe overfitting. A strategy must be taken to insure that any meaningful result from the analysis should not be contaminated by noise. Although the traditional method is to remove noise, there are, however, cases when noise is added in order to help data analysis, to assist the detection of weak signals, and to delineate the underlying processes. The known utilization of noise in aiding data analysis has been developed in the field of signal processing [34,35]. As adding noise to the input of models would be beneficial to detecting weak signals, the noise-assistant data approach is selected in this paper. The objective function (Eq. (28)) is modified as

$$\begin{aligned} &\{\rho^*, \\ &\lambda^*\} = \arg\min_{\rho,\lambda} \left[\sum_{i=1}^{N} \sum_{j=1}^{N} (\mathbf{K}_{ij} - \kappa(\mathbf{x}_i, \mathbf{x}_j))^2 + \sum_{i=1}^{N} \sum_{j=1}^{N} (\mathbf{K}_{ij} - \kappa(\mathbf{x}_i^{off}, \mathbf{x}_j^{off}))^2 \right] \end{aligned} \tag{29}$$

where $\mathbf{x}_i^{off}$ represents adding the noise vector to the measurement vector $\mathbf{x}_i$. As the reduced dimension of the underlying manifold has been obtained from SMVU1 and SMVU2, the sample with noise $\mathbf{x}_i^{off}$ is the off-manifold sample which can be easily obtained by biasing $\mathbf{x}_i$ slightly away from the manifold along the directions in the residual space. Because $\mathbf{x}_i^{off}$ are similar to the original ones $\mathbf{x}_i$, the approximate $\kappa(\mathbf{x}_i^{off}, \mathbf{x}_j^{off})$ from the noise added measurements $\mathbf{x}_i^{off}$ and the approximate $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ from the original measurements $\mathbf{x}_i$ would be the same. Eq. (29) has two parts: minimizing $\sum_{i=1}^{N} \sum_{j=1}^{N} (\mathbf{K}_{ij} - \kappa(\mathbf{x}_i, \mathbf{x}_j))^2$ which guarantees the approximation accuracy from the original samples and minimizing $\sum_{i=1}^{N} \sum_{j=1}^{N} (\mathbf{K}_{ij} - \kappa(\mathbf{x}_i^{off}, \mathbf{x}_j^{off}))^2$which guarantees that the slightly off-manifold samples can still be close to the corresponding elements of $\mathbf{K}$. Then the over-fitting data can be avoided and authentic information of data is ensured.

SMVU1 and SMVU2 are extended from MVU, which is a special variation of KPCA. Once the kernel matrix is obtained, the dimension-reduction results of SMVU1 and SMVU2 can be obtained in a similar way of KPCA. Apply the eigenvalue decomposition onto $\mathbf{K}$,

$$\lambda \boldsymbol{\nu} = \frac{1}{N} \mathbf{K} \boldsymbol{\nu} \tag{30}$$

Eigenvalue λ and eigenvector $\boldsymbol{\nu}$ can be solved by Eq. (30), yielding the orthonormal eigenvectors,$\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \cdots, \boldsymbol{\nu}_N$, and the associated corresponding eigenvalues, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$ [11,12,16]. The solutions $\boldsymbol{\nu}$ can be expanded into $\boldsymbol{\nu} = \sum_{n=1}^{N} \alpha_n \boldsymbol{\Phi}(\mathbf{x}_n)$, where N is the total number of samples. The lth principal component of an input sample $\mathbf{x}_n$ can be extracted by projecting $\boldsymbol{\Phi}(\mathbf{x}_n)$ onto the eigenvector $\mathbf{v}_l$ in F

$$y_{l,n} = \langle \mathbf{v}_l, \Phi(\mathbf{x}_n) \rangle = \frac{1}{\sqrt{\lambda_l}} \sum_{n=1}^{N} \alpha_n^l k(\mathbf{x}_n, \mathbf{x}) \tag{31}$$

where $k(\cdot,\cdot)$ is the approximated data-dependent kernel function. To determine the number of orthonormal eigenvectors (d), there are several ways, like choosing the number of orthonormal eigenvectors for the variance to achieve a predetermined percentage, say 95%, choosing the number of orthonormal eigenvectors where there is a large gap in the eigenvalue spectrum between the dth and the $d+1$th eigenvalues, and the k-fold cross-validation. For more convincing results, in this paper, the leave-one-out cross-validation is utilized to determine d, which is an effective, attractive and sensible criteria [36]. More detailed descriptions can be referred to [36–38]. The d-dimensional embedding or the reduce-dimension output ($\mathbf{y}_n$) of $\mathbf{x}_n \in R^D$ is denoted as $\mathbf{y}_n = [y_{1,n} \ \cdots \ y_{d,n}] \in R^d$ and $d < D$. Only the first d eigenvectors of the kernel matrix which capture the majority of the system variance are kept for classification. If the rest of the eigenvectors are included for classification, the noise is likely to undermine the useful information, resulting in poor classification performance.

5. Bayesian inference classification

Once the kernel matrices are obtained through SMVU1 or SMVU2 algorithms and then the lower dimensional manifold as well as data-dependent kernel functions are constructed, it is possible to catenate the given labelled classes to build up a discriminant system in the manifold. The traditional Bayesian classifier assumes that the samples for each class are normally distributed. As the learned manifolds of MVU, SMVU1 and SMVU2 are obtained from complex transformations of a higher dimensional space, it is likely that the samples cannot satisfy the normal assumption. (There are several methods given for testing normality [39–41]). In this paper, the flexible Bayesian classifier [42,43] based on kernel density estimation is used. It does not require a strong assumption, such as normal distribution with more costs of computing time and memory [42–44].

With G_c being the sample of class c, it can be classified by using the discriminant function [20,26,42,44],

$$g_c(\mathbf{y}) = P(G_c|\mathbf{y}) \tag{32}$$

where $g_c(\mathbf{y})$ is the discriminant function for class c given a data vector $\mathbf{y} \in R^d$ and $P(G_c|\mathbf{y})$ is the posteriori probability of belong to class c.

Using Bayes' rule,

$$P(G_c|\mathbf{y}) = \frac{P(\mathbf{y}|G_c)P(G_c)}{P(\mathbf{y})} \tag{33}$$

$P(G_c)$ is the priori probability of the occurrence of class c, $P(\mathbf{y})$ is the probability density function for $\mathbf{y}$, and $P(\mathbf{y}|G_c)$ is the probability density function for the particular $\mathbf{y}$ conditioned on G_c. The identical classification occurs when Eq. (33) is replaced by the log likelihood of the sample set,

$$g_c(\mathbf{y}) \propto \ln P(\mathbf{y}|G_c) + \ln P(G_c) \tag{34}$$

Instead of assuming that the samples for each class in the manifold is normally distributed, $P(\mathbf{y}|G_c)$ can be represented in kernel estimation form

$$P(\mathbf{y}|G_c) = \frac{1}{n_c h^{(c)}} \sum_{n_c} \left[\Omega\left(\frac{\mathbf{y} - \boldsymbol{\mu}_c}{h^{(c)}} \right) \right] \tag{35}$$

where $h^{(c)} = 1/\sqrt{n_k}$ is the bandwidth and

$$\Omega(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}[\det \boldsymbol{\Sigma}_c]^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{y} - \boldsymbol{\mu}_c)\boldsymbol{\Sigma}_c^{-1}(\mathbf{y} - \boldsymbol{\mu}_c)^T \right] \tag{36}$$

n_c, $\boldsymbol{\Sigma}_c$ and $\boldsymbol{\mu}_c$ are the number of samples, mean vector and the covariance matrix for class c, respectively.

As a result, the decision of the classification can be made by

$$c^{opt}(\mathbf{y}) = arg \min_{1 \le c \le C} g_c(\mathbf{y}) \tag{37}$$

where $c^{opt}(\mathbf{y})$ denotes the classification result of $\mathbf{y}$.

To sum up, the procedure of the proposed SMVU-based process classification is depicted through the following steps.

Manifold Embedding Procedures:

(1) Normalize the training sample $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$ with different classes to zero mean and unit variance.
(2) Apply SMVU (of Eq. (16) for SMVU1 or of Eq. (23) for SMVU2) to training samples to obtain the kernel matrix $\mathbf{K}$.
(3) Perform eigen-decomposition of $\mathbf{K}$to obtain d-dimensional embedding (output) of training samples.
(4) Construct the data-dependent kernel function using Eq. (29).

On-Line Classification Procedures.

(1) Pick up the incoming data $\mathbf{x}_w$. Apply the same scaling as was used in the manifold embedding.
(2) Calculate the kernel function of the new data and extract d-dimensional embedding $\mathbf{y}_w$.
(3) Calculate posteriori probability of $\mathbf{x}_w$ belonging to each class using Eq. (34).
(4) Classify a new set of observations that describe the status of the current operation using Eq. (37).

6. Case studies

Two case studies are investigated to evaluate the performances of the two proposed supervised maximum variance unfolding methods (SMVU1 and SMVU2). The benchmark datasets of the first case are created artificially. They are presented in a three-dimensional system, as visualization of the advantage of the two proposed methods (SMVU1 and SMVU2) over MVU is very easy. Additionally, the comparisons between SMVU1 and SMVU2 are included. The second case focuses on a real industrial problem with more input variables, a more challenging testbed for process diagnosis. The simulation environment for both case studies is Matlab V2015a with CPU main frequency 3.4 GHz and 8 GB flash memory.

6.1. Numerical case

The numerical nonlinear system is based on the previous investigation problem [11,15,16]. The system with three variables is expressed as

$$\begin{aligned} x_1 &= t + e_1 \\ x_2 &= t^2 - 3t + e_2 \\ x_3 &= -t^3 + 3t^2 + e_3 \end{aligned} \tag{38}$$

where e_i, $i = 1, 2, 3$ are independent random noises. To make easy explanations, assume all e_i, $i = 1, 2, 3$ follow the same Gaussian distribution $N(0, \sigma^2)$. Also, three different variations of the measurements σ^2, 0.03^2, 0.05^2 and 0.07^2, respectively, are used. t is the input parameter which follows the uniform distribution $(0.01, 2)$. Only x_i, $i = 1, 2, 3$ are measured but e_i, $i = 1, 2, 3$ and t are not. To some extent, the noise here is larger than that in most real processes. The purpose in the case is to test the classification abilities of MVU, SMVU1 and SMVU2 for handling uncertainty of data. Besides, another two process models made by changes of the following two conditions respectively are treated as two fault conditions of the system:

Fault 1: A step change of x_2 by 0.6 is introduced.

Fault 2: x_3 is linearly increased by adding $0.005j$ in its range, where j is the sample number.

For easy visualization, three sets of observations in the normal condition represented by open blue circle points are shown in Figs. 3(a), 4(a) and 5(a). The samples represented by the green circle points and the yellow circle points are the abnormal data from fault 1 and 2 respectively. It seems to be not easy to distinguish the normal events from the abnormal ones in the joint 3D plot.

In this study, a total of 600 samples, including 200 samples from the normal operation, 200 samples from fault 1 and 200 samples from fault 2 are collected as training data. The parameters validated for each algorithm are listed:

1. Observations with noise $(\sigma^2 = 0.03^2)$:$(k = 7, d = 2, \rho = 5)$ for MVU, $(k = 7, \alpha = 400, d = 2, \rho = 6)$ for SMVU1 and $(k = 7, d = 2, \rho = 5)$ for SMVU2.
2. Observations with noise $(\sigma^2 = 0.05^2)$:$(k = 6, d = 2, \rho = 5)$ for MVU, $(k = 7, \alpha = 400, d = 2, \rho = 3)$ for SMVU1 and $(k = 6, d = 2, \rho = 10)$ for SMVU2.
3. Observations with noise $(\sigma^2 = 0.07^2)$:$(k = 7, d = 2, \rho = 3)$ for MVU, $(k = 10, \alpha = 400, d = 2, \rho = 1.5)$ for SMVU1 and $(k = 7, d = 2, \rho = 10)$ for SMVU2.

All the algorithms require finding the k-nearest neighbors for the data. The parameter k is validated within [13,14]. Also, the dimension d of the learned manifolds is determined by the leave-one-out cross-validation method.

Once the manifolds are learned by MVU, SMVU1 and SMVU2 algorithms, the effectiveness of the learned manifolds is evaluated using the three testing datasets. Each set contains two fault data and a normal one, all of which have never been used to construct the manifolds. The unsupervised MVU is able to unfold the data in a 2D manifold and the results are shown in Figs. 3(f), 4(f) and 5(f); however, all the dataset are not linearly separable. Note that in Figs. 3–5, the left column is for the training sets and the right column, for the testing sets. Like the previous study on Swiss-roll manifold, SMVU1 and SMVU2 for the training sets in Figs. 3(c)(d), 4(c)(d) and 5(c)(d) can reveal a low dimensional space where the classes are linearly separable. Figs. 3(g)(h), 4(g)(h) and 5(g)(h) show the manifolds of the testing sets with three different measurement noises. In order to determine the classification performance, the correct classification rate (CCA) is used to indicate the accuracy of the learned manifolds in classification problems. Additionally, to make fair comparisons of the classification results among MVU, SMVU1 and SMVU2, the simulation is conducted

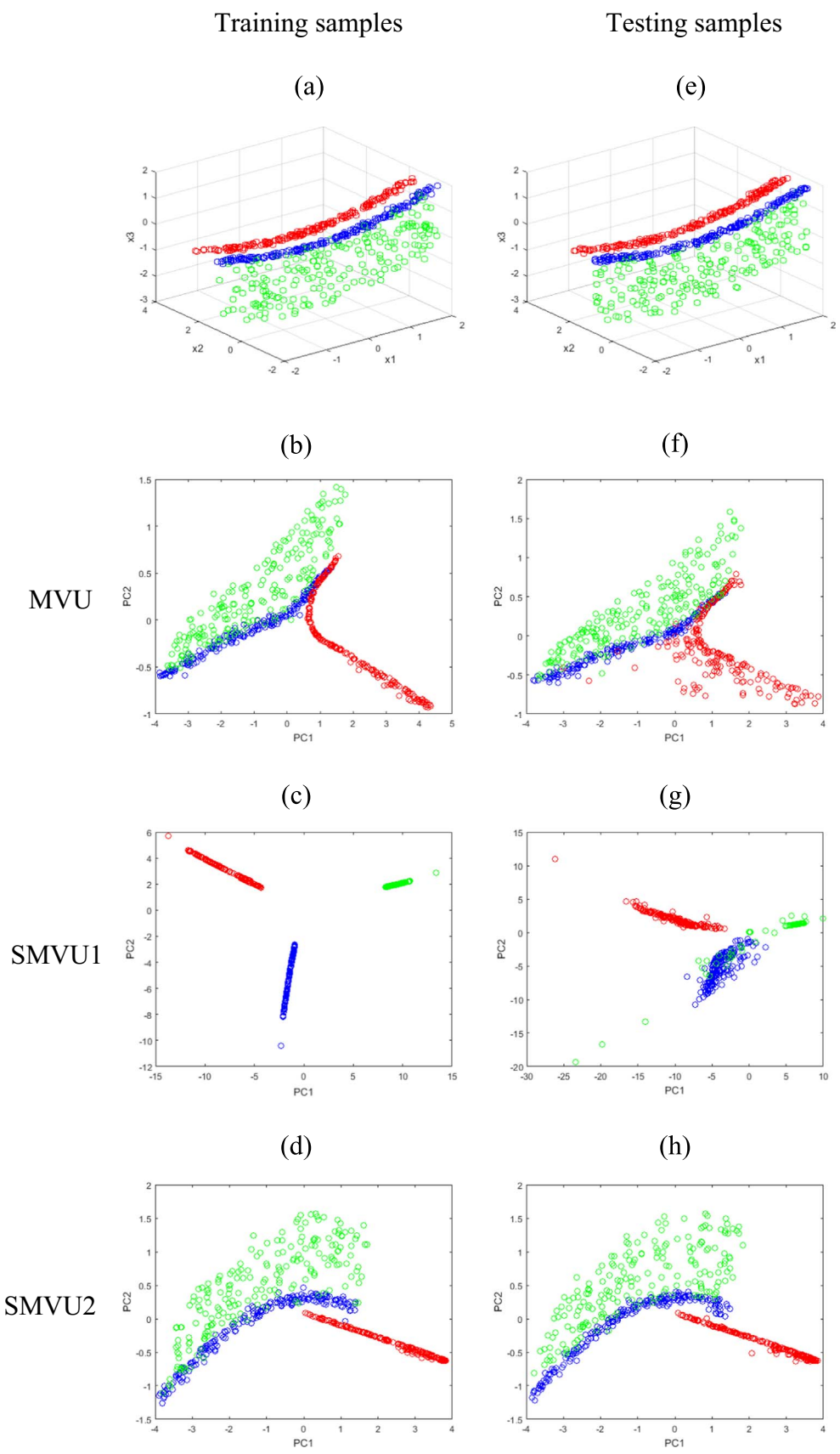

Fig. 3. Numerical case of (a) training samples and (e) testing samples where there are normal conditions labelled in blue circles, fault 1 in red circles, and fault 2 in green circles. The variation of the measurements is 0.03^2 with zero mean. The learned manifolds of the training samples are (b) MVU, (c) SMVU1, and (d) SMVU2. The learned manifolds of the testing samples (f) MVU, (g) SMVU1, and (h) SMVU2. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

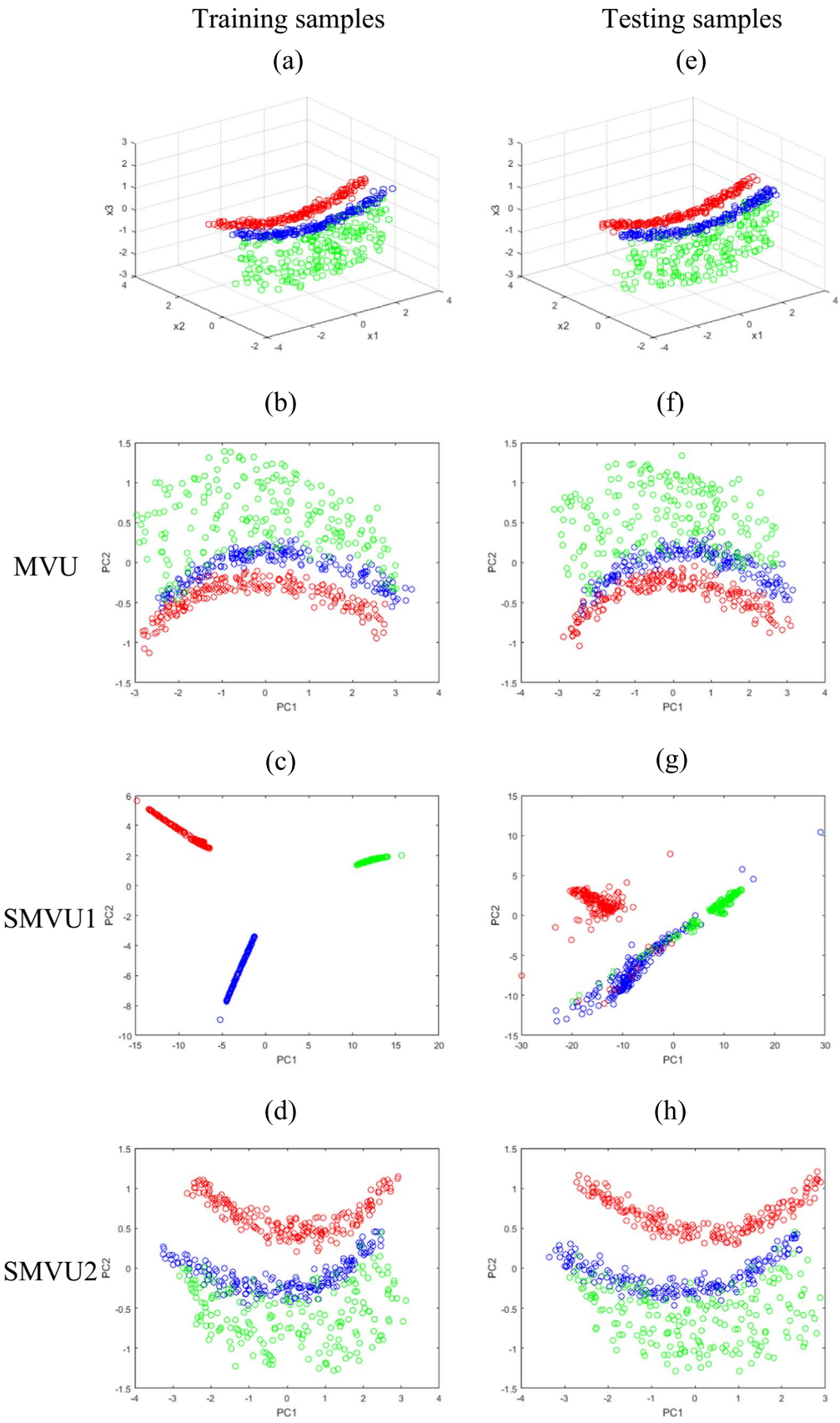

Fig. 4. Numerical case of (a) training samples and (e) testing samples where there are normal conditions labelled in blue circles, fault 1 in red circles, and fault 2 in green circles. The variation of the measurements is 0.05^2 with zero mean. The learned manifolds of the training samples are (b) MVU, (c) SMVU1, and (d) SMVU2. The learned manifolds of the testing samples (f) MVU, (g) SMVU1, and (h) SMVU2. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

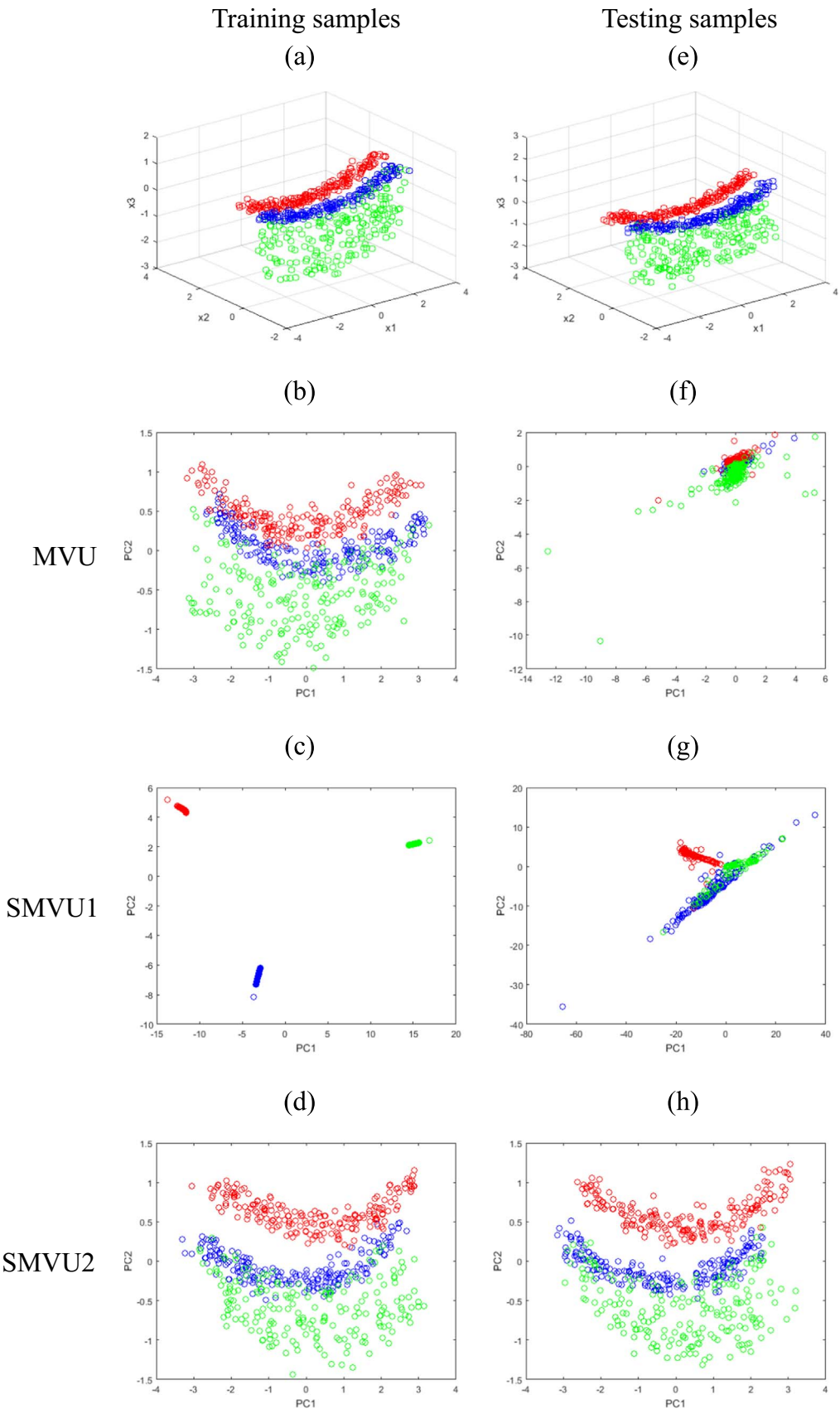

Fig. 5. Numerical case of (a) training samples and (e) testing samples where there are normal conditions labelled in blue circles, fault 1 in red circles, and fault 2 in green circles. The variation of the measurements is 0.07^2 with zero mean. The learned manifolds of the training samples are (b) MVU, (c) SMVU1, and (d) SMVU2. The learned manifolds of the testing samples (f) MVU, (g) SMVU1, and (h) SMVU2. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Numerical case of the overall correct classification rate for the three sets of testing samples (a) $\sigma^2 = 0.03^2$, (b) $\sigma^2 = 0.05^2$and (c) $\sigma^2 = 0.07^2$in MVU, SMVU1 and SMVU2 algorithms.

(a)

Algorithms	Class 1	Class 2	Class 3	Total
MVU	0.785	0.820	0.295	0.633
SMVU1	0.895	0.850	0.855	0.867
SMVU2	0.990	0.945	0.535	0.823

(b)

Algorithms	Class 1	Class 2	Class 3	Total
MVU	0.685	0.850	0.705	0.747
SMVU1	0.950	0.950	0.740	0.880
SMVU2	0.935	1.000	0.710	0.882

(c)

Algorithms	Class 1	Class 2	Class 3	Total
MVU	0.780	0.810	0.640	0.743
SMVU1	0.780	0.890	0.735	0.802
SMVU2	0.895	0.960	0.695	0.850

repeatedly for 20 times and the average of CCA for each algorithm is shown in Table 1. Among the three sets of data with different noises, the bigger the value of the measurement noise is, the larger the overlapped data regions become. When the variance of noise is small (0.03^2), which means uncertainty is inconspicuous, SMVU1 performs better than SMVU2; as the noise becomes larger (the variance is 0.05^2), the two algorithms performs quite the same; and when the noise becomes even larger (the variance is 0.07^2) with high uncertainty, SMVU2 outperforms SMVU1. Because the hard constraints in SMVU1 are used in specific locations and the objective function aims at unfolding the manifold, its generalization ability is unwarrantable when the margins of different classes are close; that is, SMVU1 is good for the classification with highly certain data. On the other hand, SMVU2 is able to avoid the problem of close margins of different classes with high uncertainty based on the soft constraints.

6.2. Industrial case

In this section, the proposed methods are further validated by the classification of the melting index (MI) of an industrial polyethylene process in Taiwan. In this study, all the data samples have been collected from the daily process records and the corresponding laboratory analysis. The grade of the product quality of the polyethylene production is sampled and analyzed in the lab once a day, always in the morning. As MI cannot be analyzed directly online, the operating variables have to be manipulated until the assay results are available. In this section, the proposed classification algorithms are applied to online classification MI of polyethylene to validate its performance. The process variables correlated with the product quality have been selected. As prior requirements, the reliable sensor measurements and data collections play crucial roles in process classification. In order to enhance the performance of the classification, the prior knowledge of the MI production process has been taken into account. According to the engineering experience, there are 3 grades of prior knowledge available. Among the 3 grades, ten variables are selected from the sixteen variables for modeling and testing. In this study, 638 samples of three steady-state grades collected in a product line from 2008 to 2011 are investigated. The data are divided into training and testing samples. The numbers of training samples for the steady-state grades (S1, S2 and S3) are 150, 176 and 162, respectively. The numbers of testing samples for the steady-state grades (S1, S2 and S3) are 50, 50 and 50, respectively.

It is not possible to plot ten variables in a plot. Three out of ten variables of the training samples and the testing samples are shown in Fig. 6(a) and (e). The proposed SMVU1 algorithm (Fig. 6(c) and (g)) and SMVU2 algorithm (Fig. 6(d) and (h)) have been well separable when compared with MVU (Fig. 6(b) and (f)) even though some data in SMVU1 and SMVU2 are overlapped. The classification accuracy performance for all the algorithms on all the datasets is summarized in Table 2. It is found that the proposed algorithm performs better than the other algorithms when there is a hidden "wrapped" manifold in the original data set, such as the changes in Fig. 6(a) and (e). They show better classification accuracy as the conventional MVU.

In the three numerical cases and one industrial case, four uncertain indices in Eq. (24) are 1.0668, 1.0883, 1.1361 and 1.0463, respectively. It is found that there is less noise in this industrial case than in the numerical case. As uncertainty is inconspicuous, SMVU1 outperforms SMVU2 in this industrial case. In a word, when noise is not significant, SMVU1 is recommended. In the numerical cases, as shown in Figs. 3–5 and Table 1, when the variation of the measurement is 0.03^2 with zero mean, uncertainty is quite inconspicuous. In this situation, SMVU1 outperforms SMVU2; when the variation of the measurement is 0.05^2 with zero mean, SMVU1 and SMVU2 perform quite the same; when the variation of the measurement is 0.07^2 with zero mean, which means uncertainty is quite conspicuous, SMVU2 outperforms SMVU1.

7. Conclusion

The MVU algorithm finds a low dimensional representation of high dimensional input data. The Euclidean metric in the low dimensional space is often much more meaningful than that in the original input space, but it does not preserve the data topology at all. In contrast to the previous MVU work, in this work, the concepts of the supervised manifold embedding approaches are extended via exploiting two different types of supervision objectives, SMVU1 and SMVU2. The features of the proposed methods are listed as follows:

- Like the unsupervised scenario of MVU, the proposed SMVU1 and SMVU2 can map the manifold data sets with fewer intrinsic degrees of freedom than their ambient vector space via the non-linear dimension reduction algorithm. However, unlike MVU, the supervised SMVU1 and SMVU2 algorithm can unfold data and pull different class data apart simultaneously to keep the data topology.
- Like the core of MVU algorithm involving a semi-definite program, SMVU1 and SMVU2 algorithms that can learn a kernel matrix are systematically derived. Because the two supervised manifold learning algorithms are all formulated with the kernel matrix. Each entry of the matrix is the inner product of the embeddings, the computation complexity of the two algorithms is irrelevant to the dimension of the data. The kernel matrix of SMVU1 and SMVU2 can be learned by convex optimization; thus the local manifold structure is preserved. This makes the algorithms very powerful and reliable.
- To treat new data points which do not come from the original training samples, the data-dependent kernel functions for SMVU1 and SMVU2 are developed. The kernel function approximation of the actual kernel based SMVU1 and SMVU2 is provided based on Nyström method as it is good for major applications of other sampling techniques, like SVM, SVR and KPCA.
- In SMVU1, hard constraints are used in specific locations and the objective function aims at unfolding the manifold. Its generalization ability is unwarrantable when the margins of different classes are close; in other words, SMVU1 is good for the classification with highly certain data. On the other hand, SMVU2 has the ability to avoid the problem of close margins of different classes with high uncertainty based on the soft constraints.

The case studies show the capability of the proposed methods and its applicability to a chemical process. In the future work, the proposed methods will be applied to the applications of different data types in other fields, like the UCI machine learning datasets. Also, for large problems such as the number of data set exceeding thousands, the

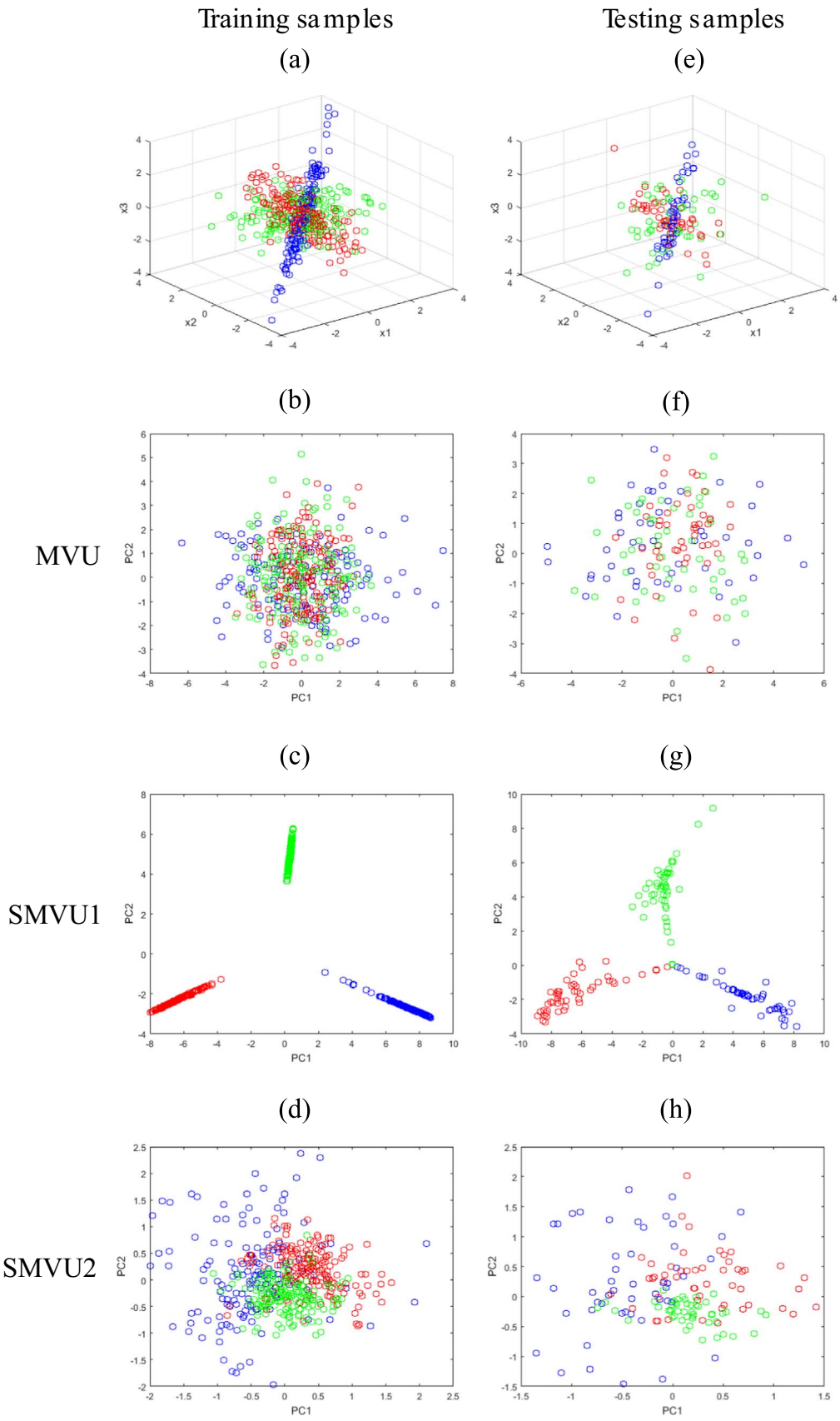

Fig. 6. Industrial case of (a) training samples and (e) testing samples where there are grade 1 labelled in blue circle, grade 2 in red circle, and grade 3 in green circle. The learned manifolds of training samples (b) MVU, (c) SMVU1, and (d) SMVU2. The learned manifolds of testing samples (f) MVU, (g) SMVU1, and (h) SMVU2. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Industrial case of the overall correct classification rate for testing samples in MVU, SMVU1 and SMVU2 algorithms.

Algorithms	Class 1	Class 2	Class 3	Total
MVU	0.380	0.580	0.560	0.507
SMVU1	1.000	0.940	0.800	0.913
SMVU2	0.680	0.720	0.880	0.760

memory associated with storage and the computation time can be prohibitive even on modern computers. The sampling-based low-rank approximation of SMVU1 and SMVU2 will be extended to large-scale applications in the future.

Acknowledgments

This work is supported by National Nature Science Foundation of China under Grant 61573308, Research Fund for the Doctoral Program of Higher Education under Grant 20130101110138, and Ministry of Science and Technology, Taiwan, R.O.C. under Grant MOST 103-2221-E-033-068-MY3.

References

[1] V. Venkatasubramanian, R. Rengaswamy, K. Yin, S.N. Kavuri, A review of process fault detection and diagnosis: part I: quantitative model-based methods, Comput. Chem. Eng. 27 (2003) 293–311.
[2] L.H. Chiang, R.D. Braatz, E.L. Russell, Fault Detection and Diagnosis in Industrial Systems, Springer Science & Business Media, Springer-Verlag, London, 2001.
[3] S. Joe Qin, Statistical process monitoring: basics and beyond, J. Chemom. 17 (2003) 480–502.
[4] S. Yin, S.X. Ding, X. Xie, H. Luo, A review on basic data-driven approaches for industrial process monitoring, IEEE Trans. Ind. Electron. 61 (2014) 6418–6428.
[5] Z. Ge, Z. Song, F. Gao, Review of recent research on data-based process monitoring, Ind. Eng. Chem. Res. 52 (2013) 3543–3562.
[6] Y. Zhang, S.J. Qin, Fault detection of nonlinear processes using multiway kernel independent component analysis, Ind. Eng. Chem. Res. 46 (2007) 7780–7787.
[7] X. Chen, Z. Ge, Switching LDS-based approach for process fault detection and classification, Chemom. Intell. Lab. Syst. 146 (2015) 169–178.
[8] S. Gajjar, A. Palazoglu, A data-driven multidimensional visualization technique for process fault detection and diagnosis, Chemom. Intell. Lab. Syst. 154 (2016) 122–136.
[9] D. Slišković, R. Grbić, Ž. Hocenski, Multivariate statistical process monitoring, Teh. Vjesn.Tech. Gaz. 19 (2012) 33–41.
[10] I. Jolliffe, Principal component analysis, Wiley Online Library, 2002.
[11] J.-M. Lee, C. Yoo, S.W. Choi, P.A. Vanrolleghem, I.-B. Lee, Nonlinear process monitoring using kernel principal component analysis, Chem. Eng. Sci. 59 (2004) 223–234.
[12] B. Schölkopf, A. Smola, K.-R. Müller, Kernel principal component analysis, in: International Conference on Artificial Neural Networks, Springer, 1997, pp. 583–588.
[13] K.Q. Weinberger, L.K. Saul, Unsupervised learning of image manifolds by semidefinite programming, Int. J. Comput. Vis. 70 (2006) 77–90.
[14] K.Q. Weinberger, F. Sha, L.K. Saul, Learning a kernel matrix for nonlinear dimensionality reduction, in: Proceedings of the twenty-first international conference on Machine learning, ACM, 2004, pp. 106.
[15] J.-D. Shao, G. Rong, Nonlinear process monitoring based on maximum variance unfolding projections, Expert Syst. Appl. 36 (2009) 11332–11340.
[16] J.-D. Shao, G. Rong, J.M. Lee, Learning a data-dependent kernel function for KPCA-based nonlinear process monitoring, Chem. Eng. Res. Des. 87 (2009) 1471–1480.
[17] Y.-J. Liu, T. Chen, Y. Yao, Nonlinear process monitoring and fault isolation using extended maximum variance unfolding, J. Process Control 24 (2014) 880–891.
[18] J. Yu, Multiway discrete hidden Markov model-based approach for dynamic batch process monitoring and fault classification, AIChE J. 58 (2012) 2714–2725.
[19] J. Yu, A support vector clustering-based probabilistic method for unsupervised fault detection and classification of complex chemical processes using unlabeled data, AIChE J. 59 (2013) 407–419.
[20] S. Zhong, Q. Wen, Z. Ge, Semi-supervised Fisher discriminant analysis model for fault classification in industrial processes, Chemom. Intell. Lab. Syst. 138 (2014) 203–211.
[21] X. Wang, H. Feng, Y. Fan, Fault detection and classification for complex processes using semi-supervised learning algorithm, Chemom. Intell. Lab. Syst. 149 (2015) 24–32.
[22] L. Vandenberghe, S. Boyd, Semidefinite programming, SIAM Rev. 38 (1996) 49–95.
[23] A. Schwaighofer, V. Tresp, K. Yu, Learning Gaussian process kernels via hierarchical Bayes, Adv. Neural Inf. Process. Syst. (2004) 1209–1216.
[24] T.-J. Chin, D. Suter, Out-of-sample extrapolation of learned manifolds, IEEE Trans. Pattern Anal. Mach. Intell. 30 (2008) 1547–1556.
[25] C. Williams, M. Seeger, Using the Nyström method to speed up kernel machines, in: Proceedings of the 14th annual conference on neural information processing systems, 2001, pp. 682–688.
[26] K.P. Murphy, Naive bayes classifiers, University of British Columbia, (2006).
[27] H. Wolkowicz, R. Saigal, L. Vandenberghe, Handbook of Semidefinite Programming: Theory, Algorithms, and Applications, Springer Science & Business Media, Springer, US, 2012.
[28] B. Borchers, CSDP, AC library for semidefinite programming, Optim. Methods Softw. 11 (1999) 613–623.
[29] P.A. Parrilo, Semidefinite programming relaxations for semialgebraic problems, Math. Program. 96 (2003) 293–320.
[30] A. Zjajo, J.P. de Gyvez, Low-power high-resolution analog to digital converters: design, test and calibration, Springer Science & Business Media, 2010.
[31] B.P. Bezruchko, D.A. Smirnov, Extracting Knowledge from Time Series: An Introduction to Nonlinear Empirical Modeling, Springer Science & Business Media, Springer-Verlag, Berlin Heidelberg, 2010.
[32] H.S. Seung, D.D. Lee, The manifold ways of perception, Science 290 (2000) 2268–2269.
[33] L.K. Saul, S.T. Roweis, Think globally, fit locally: unsupervised learning of low dimensional manifolds, J. Mach. Learn. Res. 4 (2003) 119–155.
[34] S.C. Douglas, A. Cichocki, S.-i Amari, Self-whitening algorithms for adaptive equalization and deconvolution, IEEE Trans. Signal Process. 47 (1999) 1161–1165.
[35] A. Trucco, Experimental results on the detection of embedded objects by a prewhitening filter, IEEE J. Ocean. Eng. 26 (2001) 783–794.
[36] G.C. Cawley, N.L. Talbot, Efficient leave-one-out cross-validation of kernel fisher discriminant classifiers, Pattern Recognit. 36 (2003) 2585–2592.
[37] E. Saccenti, J. Camacho, Determining the number of components in principal components analysis: a comparison of statistical, crossvalidation and approximated methods, Chemom. Intell. Lab. Syst. 149 (2015) 99–116.
[38] S.J. Qin, R. Dunia, Determining the number of principal components for best reconstruction, J. Process Control 10 (2000) 245–250.
[39] M. Stehlík, L. Střelec, M. Thulin, On robust testing for normality in chemometrics, Chemom. Intell. Lab. Syst. 130 (2014) 98–108.
[40] N.M. Razali, Y.B. Wah, Power comparisons of shapiro-wilk, Kolmogorov–Smirnov, lilliefors and anderson-darling tests, J. Stat. Model. Anal. 2 (2011) 21–33.
[41] H.W. Lilliefors, On the Kolmogorov–Smirnov test for normality with mean and variance unknown, J. Am. Stat. Assoc. 62 (1967) 399–402.
[42] G.H. John, P. Langley, Estimating continuous distributions in Bayesian classifiers, in: Proceedings of the Eleventh conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., 1995, pp. 338-345.
[43] J. Friedman, T. Hastie, R. Tibshirani, The Elements of Statistical Learning, Springer Series in Statistics Springer, Berlin, 2001.
[44] Y.-L. He, R. Wang, S. Kwong, X.-Z. Wang, Bayesian classifiers based on probability density estimation and their applications to simultaneous fault diagnosis, Inf. Sci. 259 (2014) 252–268.