Sliding Window-Based Fault Detection From High-Dimensional Data Streams

Liangwei Zhang, Jing Lin, *Member, IEEE*, and Ramin Karim

***Abstract*—High-dimensional data streams are becoming increasingly ubiquitous in industrial systems. Efficient detection of system faults from these data can ensure the reliability and safety of the system. The difficulties brought about by high dimensionality and data streams are mainly the "curse of dimensionality" and concept drifting, and one current challenge is to simultaneously address them. To this purpose, this paper presents an approach to fault detection from nonstationary high-dimensional data streams. An angle-based subspace anomaly detection approach is proposed to detect low-dimensional subspace faults from high-dimensional datasets. Specifically, it selects fault-relevant subspaces by evaluating vectorial angles and computes the local outlier-ness of an object in its subspace projection. Based on the sliding window strategy, the approach is further extended to an online mode that can continuously monitor system states. To validate the proposed algorithm, we compared it with the local outlier factor-based approaches on artificial datasets and found the algorithm displayed superior accuracy. The results of the experiment demonstrated the efficacy of the proposed algorithm. They also indicated that the algorithm has the ability to discriminate low-dimensional subspace faults from normal samples in high-dimensional spaces and can be adaptive to the time-varying behavior of the monitored system. The online subspace learning algorithm for fault detection would be the main contribution of this paper.**

***Index Terms*—Big data analytics, fault detection, high-dimensional data, stream data mining.**

NOMENCLATURE

Acronyms

ABSAD	Angle-based subspace anomaly detection.
AUC	Area under curve.
EWPCA	Exponentially weighted principal component analysis.
FNR	False negative rate.
FPR	False positive rate.
ICA	Independent component analysis.
KDE	Kernel density estimation.
LOF	Local outlier factor.
LOS	Local outlier score.
MSPC	Multivariate statistical process control.
PCA	Principal component analysis.
ROC	Receiver operating characteristic.
RPCA	Recursive PCA.
SNN	Shared nearest neighbors.
SOD	Subspace outlier detection.
SPE	Squared prediction error.
SWPCA	Sliding window PCA.
TPR	True positive rate.

$\boldsymbol{X}$	Design matrix.
m	Number of data points (rows) in $\boldsymbol{X}$.
n	Number of dimensions (columns) in $\boldsymbol{X}$.
N	Index set of feature space $\{1, \ldots, n\}$.
LOS	Vector of LOSs.
i	ith data point (row) in $\boldsymbol{X}$, or the ith window.
j	jth element of a vector, or the jth dimension (column) of a matrix.
v	Vector representation of a point.
p	Data point (outlier candidate).
RP	Set of reference points of a point.
q	Data point represents the geometric center of all the points in RP(p).
l	Line connected through two points (e.g., p and q).
$\mathrm{dist}(\cdot, \cdot)$	Metric for measuring the distance between two points.
$k\mathrm{NN}(i)$	k nearest neighbor list of the ith point.
$\mathrm{Sim}_{\mathrm{SNN}}$	Similarity value of two points derived by the SNN method.
SNN_s	s nearest neighbor list of a point derived by the SNN method.
$\mathrm{PCos}(\vec{l}, \vec{\mu}_n(j))$	Average absolute value of cosine between line l and the jth axis in all possible combinations of the 2-D spaces (j, j^-), where $j^- \in N \backslash \{j\}$.
d	Number of retained dimensions of a point.
G	Threshold for singling out large PCos values.
$\mathrm{K}(\cdot)$	Kernel function.
h	Smoothing parameter in the KDE.
$\boldsymbol{W}(i)$	All the samples preserved in the ith window profile.
kNN-list(i)	Set of sets, containing the k nearest neighbors of every sample in the ith window.
k-Dist(i)	Vector, recording the k-distance of each sample in the ith window.
CL(i)	Scalar, the control limit of the ith window.

Manuscript received May 16, 2015; revised September 2, 2015 and November 2, 2015; accepted June 19, 2016. This paper was recommended by Associate Editor G. Biswas. *(Corresponding author: Liangwei Zhang.)*

The authors are with the Division of Operation and Maintenance Engineering, Luleå University of Technology, Luleå SE-97187, Sweden (e-mail: liangwei.zhang@ltu.se).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2016.2585566

α	Acute angle between line l and axis x.
β	Acute angle between line l and axis y.
γ	Significance level.
σ	Row vector, containing the columnwise standard deviation of a matrix.
ε	Significantly small positive quantity.
μ	Axis-parallel unit vector.
θ	Input parameter of the ABSAD-based algorithm for selecting relevant subspace.
$\mathbf{\Sigma}$	Covariance matrix of a set of points.
#	Cardinality of a set.
$\overline{\square}$	Mean vector of a matrix ($\square$ denotes a placeholder).
$\vec{\square}$	Vector representation of a line.
$\hat{\square}$	Estimation of an underlying function.
$\square^*$	Normalized matrix (e.g., $\boldsymbol{X}^*$).
$\square^T$	Transpose of a vector or a matrix.
$\square^{-1}$	Inverse of a square matrix.
$\square^\#$	Non-zero scalar quantity obtained by zero-value replacement (e.g., $l_j^\# = 10^{-5}$, if $l_j = 0$).
$\square^-$	One of the remainder dimensions of the original feature space excluding a specific dimension (e.g., $j^- \in N\backslash\{j\}$).
$\square'$	Projection of point, set of points or line on the retained subspace (e.g., $\mathrm{RP}(p)'$).

I. Introduction

FAULT detection is one important task to identify defective states and conditions within industrial systems, subsystems, and components [1]. Early discovery of system faults can ensure the reliability and safety of the systems and reduce the risk of unplanned breakdowns [2], [3]. Primary fault detection techniques can be categorized into the classes of model-based, signal-based, knowledge-based (data-driven), and active ones [3], [4]. With the ever-increasing sensor data available, knowledge-based techniques are finding more chances in fault detection applications [5], [6]. Depending on whether the raw data are labeled or not, knowledge-based techniques can be further classified into supervised and unsupervised ones [3]. The former uses plentiful positive (faulty) and negative (normal) data to learn the underlying data generating mechanisms for both classes, as has been done in [7], whereas the later learns the normal system behavior only from normal samples, and faults are detected as deviations from the learned normality, as has been done in [8]. Although supervised algorithms can provide favorable results in detecting and even isolating faults, faulty data for training purpose are generally insufficient and expensive to acquire in real-world applications. This problem becomes even worse as dimensionality increases since the number of data for covering a fraction of the feature space grows exponentially with increasing dimensionality [9].

With the advance of sensor technology, industrial systems are increasingly equipped with a large number of sensors, such as thermometers, vibroscopes, displacement meters, flow meters, etc. Those sensors can continuously generate high-dimensional data at high speed, namely high-dimensional data streams. Recently, considerable interest has been focused on big data analytics for its attempts to extract information, knowledge and wisdom from these data, among which fault detection is one of the most promising applications wherein reliability meets big data [10]. However, it is challenging to utilize existing techniques to conduct fault detection on these high-dimensional data streams which partly share the characteristics of big data [11]. Current research on fault detection from high-dimensional data streams are mainly studied from two aspects separately: 1) high dimensionality and 2) data stream.

High dimensionality is one measure of the high volume of big data (the other measure being instance size) [12]. It has been recognized as the distinguishing feature of modern field reliability data [10]. The "curse of dimensionality" may cause the deterioration of many fault detection techniques because the degree of data abnormality in fault-relevant dimensions can be obscured or even masked by irrelevant attributes [9], [13], [14]. Moreover, notions like proximity, distance, or neighborhood become less meaningful as dimensionality increases [15]. Existing MSPC methods, including PCA and ICA, have been widely used in fault detection applications [16], [17]. However, PCA assumes multivariate normality of the measurements and ICA assumes the measurements to be non-Gaussian distributed [6], and thereby limiting their performance in real-world applications [18]. To improve this, several studies have integrated MSPC methods with the density-based LOF technique, which is free of distribution assumptions [18], [19]. Though better accuracy was reported, the performance of LOF implemented on full-dimensional spaces still degrades as dimensionality increases, as will be shown in Section IV-C. Theoretical studies on high-dimensional anomaly detection mainly focus on subspace anomaly detection, including, for example, by random projection or heuristic searches over subspaces [20], [21]. However, these methods are either arbitrary in selecting subspaces or computationally intensive. Although several studies have started to probe the above high dimensionality problem, research concerning high-dimensional fault detection remains under-explored.

Data stream refers to the data that are continuously generated at a high rate. It reflects the characteristics of big data in the aspects of both high volume and high velocity. Normally, data streams are also temporally ordered, fast-changing, and potentially infinite [22], [23]. Moreover, they tend to be high-dimensional in their nature in many cases, such as sensor networks, cybersurveillance, and so on. The difficulties raised by data streams in fault detection tasks can be summarized as follows.

1) To have a timely assessment on the system status, algorithms must have low-latency in responding to the fast-flowing data stream. Therefore, "on-the-fly" analysis is desired in this context [22].
2) It is impractical or even impossible to scan a potentially infinite data stream multiple times considering the finite memory resources [23]. Thus, algorithms that conduct one-pass scan over the data stream are imperative.
3) Data streams can evolve as time progresses. This is also known as concept drifting [7], [24]. In the context

of fault detection, the behavior of systems can vary over time—time-varying—due to many reasons, such as seasonal fluctuation, equipment aging, process drifting, and so forth. Fault detection algorithms need to be adaptive to this time-varying behavior of the monitored system [25]. A large portion of online fault detection algorithms were extended from existing ones for the purpose of monitoring data streams, such as the RPCA, EWPCA, and SWPCA [26]–[28]. The key to these algorithms is that the learning model should be refined, enhanced, and personalized while the stream evolves so as to accommodate the natural drift in the data stream. In spite of the extensive studies of fault detection techniques, fault detection applications which specifically address the challenges imposed by data stream properties are also limited [29].

Today, the capability of fault detection techniques in simultaneously addressing the challenges associated with high dimensionality and data streams remains limited. To solve the above problems, this paper proposes an unsupervised approach to fault detection from high-dimensional data streams with time-varying characteristics.

1) First, in considering the high-dimensional challenges in fault detection tasks, an ABSAD approach is proposed. The ABSAD approach selects fault-relevant subspaces by evaluating vectorial angles and computes the local outlier-ness of an object in its subspace projection by a normalized Mahalanobis distance measure.
2) Second, aiming to detect faults from data stream with time-varying characteristics, the ABSAD approach is extended to an online mode based on the sliding window strategy. According to the requirements of the ABSAD approach, several necessities (mean vector, KNN list, etc.) are identified and incorporated into the window profile of the sliding window ABSAD algorithm.

The updating mechanisms to these necessities are investigated and thoroughly described. To validate the proposed algorithm, we compared it with the LOF-based approaches on artificial datasets and found the algorithm displayed superior accuracy. The results of the experiment demonstrated the efficacy of the proposed algorithm. They also indicated that the algorithm is able to discriminate low-dimensional subspace faults from normal samples in high-dimensional spaces and can be adaptive to the time-varying behavior of the monitored system. This paper contributes to a new online subspace learning algorithm for detecting faults from nonstationary systems. To the best of our knowledge, no online fault detection algorithms utilizing angle evaluation to select fault-relevant subspaces have been reported to date.

The rest of this paper proceeds as follows. In Section II, we propose an ABSAD approach with the aim of handling high dimensionality challenges in anomaly detection tasks. To address the challenges associated with data stream mining, Section III extends the ABSAD approach to an online mode based on the sliding window strategy. Section IV evaluates the proposed algorithm on synthetic datasets and compares it with other alternatives. Finally, the study is concluded in Section V.

II. Angle-Based Subspace Anomaly Detection Approach

To mitigate the impact exerted by anomaly-irrelevant attributes, the degree of deviation of a data point from its neighboring points (i.e., local outlier-ness) should be computed in a meaningful subspace instead of the full-dimensional space. The subspace is said to be meaningful in the sense that it should capture most information with regard to the discordance of an object to its adjacent ones. To this end, we propose an ABSAD approach to exploring and selecting low-dimensional, axis-parallel subspaces that can retain a large portion of points' local outlier-ness. For each data instance, the degree of deviation from its neighborhood is evaluated on the obtained subspace. A local outlier score is then computed for these points indicating whether it is abnormal or not. More theoretical discussions to this approach can be referred to [30].

In the following, we describe the ABSAD approach. We first elucidate the model assumption, and then introduce the structure of the ABSAD approach. Subsequently, we elaborate the main steps of the approach respectively and integrate them into a single algorithm.

A. Model Assumption

The separability of different data generation mechanisms may not necessarily depend on the amount of data dimensionality, but instead on the ratio of relevant versus irrelevant attributes [13]. In cases where the relevant attributes account for a large proportion of the whole dimensions, the separability among different mechanisms tends to increase, which means traditional techniques are still valid and may work even better in high-dimensional spaces. Conversely, when relevant attributes are in a minority of the whole dimensions, the curse of dimensionality would hamper anomaly detection tasks. This paper attempts to address the problem of the latter case. Hereinafter, we assume the number of anomaly-relevant attributes is in a minority of all the attributes in the feature space.

B. Computational Procedure

The computational procedure of the ABSAD approach is presented in Fig. 1. The first step, data preparation, usually comprises data acquisition, data cleaning, feature selection, and other preprocessing processes. The complexity of this step mainly depends on the quality of the collected raw data. Since this step is highly dependent on various applications and plentiful studies have been conducted specifically on these topics, the remainder of this section will instead focus on the core part of the approach (enclosed by the outer box in Fig. 1).

In the following, we define $\boldsymbol{X}$ ($\boldsymbol{X} \subseteq R^{m\times n}$) as the design matrix. Each row of the matrix represents a data point (also known as data instance, object or observation) in a n-dimensional feature space N, where $N = \{1, \ldots, n\}$ and $n \geq 2$. The objective of this approach is to define a function which maps $\boldsymbol{X}$ to a real-valued vector LOS, i.e., $f : \boldsymbol{X} \rightarrow$LOS, where LOS($i$) is the ith point's local outlier score. To evaluate the local outlier-ness of a particular data point p, a set of reference points RP(p) of p needs to be specified beforehand. The set RP(p) reflects the notion of locality. Additionally, a distance metric dist(p, o) (e.g., one of the L_p norms) measuring

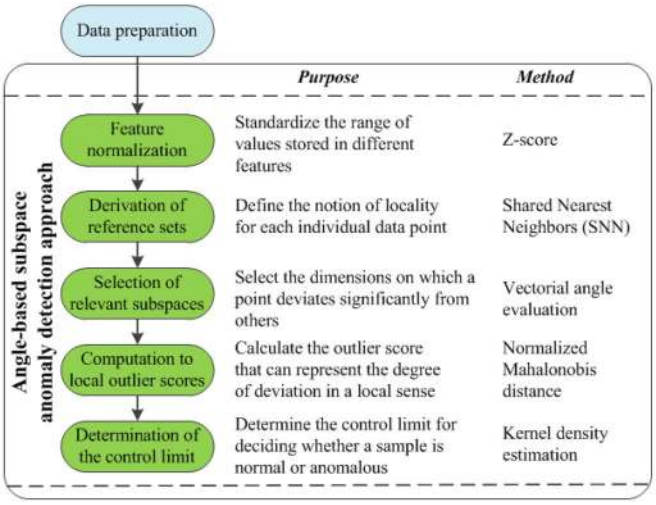

Fig. 1. Computational procedure of the ABSAD approach.

the distance between any two points p and o is required when deriving the set RP(p).

C. Feature Normalization

The feature normalization step is to standardize the range of values in different features. It is imperative to conduct this step because those features with mean or variance that are orders of magnitude larger than others are likely to dominate succeeding computations. In anomaly detection applications, we recommend the use of the Z-score normalization instead of the min–max scaling considering the latter may suppress the effect of abnormality which might deviate from our intention. The Z-score method normalizes the design matrix $\boldsymbol{X}$ to a dimensionless matrix $\boldsymbol{X}^*$ with zero mean and unit variance. The ith row of $\boldsymbol{X}^*$ can be calculated as follows:

$$x_i^* = \frac{x_i - \overline{x}}{\sigma}, \text{ for all } i \in \{1, 2, \ldots, m\} \tag{1}$$

where $\overline{x}$ is the columnwise mean vector of the design matrix and σ is the columnwise standard deviation vector.

D. Derivation of Reference Sets

The implication of locality needs to be defined in local outlier detection approaches, i.e., to determine the set of reference points. In low-dimensional spaces, distance-based measures are frequently used to explore the vicinity of a point. However, as stated before, notions like proximity, distance, or neighborhood become less meaningful in high-dimensional spaces. To cope with this problem, an alternative series of methods, which introduce a secondary measure based on the rankings of data instances produced by a primary similarity measure, were proposed. Among these methods, the SNN approach is the most common one. The applicability of SNN in high-dimensional spaces has been empirically justified in [13] and it was adopted in several other related research projects [21], [31].

The main idea of the SNN method is that two points generated by the same mechanism should have more overlap in their nearest neighbor list, and vice versa. Specifically, SNN measures the similarity of two points as the number of common nearest neighbors which are derived from a primary measure. Prior to calculating the SNN similarity, a primary measure is needed to specify the nearest neighbors for all the points. The primary measure can be of any traditional similarity measure (such as L_p norm or the cosine measure). Notably, the ranking of data instances derived by the primary measure is typically still meaningful in high-dimensional spaces even though the contrast of distance measure has deteriorated. Suppose the k nearest neighbor set of point p is denoted as kNN(p). Then, the SNN similarity between points p and q can be represented as

$$\text{Sim}_{\text{SNN}}(p, q) = \#\{k\text{NN}(p) \cap k\text{NN}(q)\}. \tag{2}$$

Here the sign # returns the cardinality of the intersection between sets kNN(p) and kNN(q). Notably, the above equation only computes the similarity between pairwise points. To derive a secondary nearest neighbor list SNN(p), we need to sort all the SNN similarity values of point p with respect to other remaining points in $\boldsymbol{X}$ in descending order. The first s elements with largest SNN similarity values in the set SNN(p), i.e., SNN$_s$(p), constitute the reference set RP(p).

E. Selection of Relevant Subspaces

The general idea as to which dimensions should be retained to constitute the anomaly-relevant subspaces is elaborated as below. The example shown in Fig. 2 gives us an intuition on selecting relevant subspaces. In a 2-D case as shown in Fig. 2(a), the set RP(p) (enclosed by an ellipse) contains the nearest neighbors of an outlier candidate p (black cross). In Fig. 2(b), the geometrical center of RP(p) is calculated and represented by the point q (red circle). Points p and q are connected to form the line l (red solid line). In considering which of the 2-D (x and y) p deviates significantly from its reference points, we can evaluate the angle α between line l and the x-axis, and β between line l and the y-axis (both α and β are acute angle). Intuitively, the dimension which has a fairly small angle with line l should be retained in the relevant subspace. In this case, angle α is small indicating that line l is nearly parallel to the x-axis, whereas β is markedly larger implying that line l is almost perpendicular to the y-axis. Consequently, the dimension on the x-axis is retained and the dimension on the y-axis is excluded. Now, as shown in Fig. 2(c), we can project the original data points onto the x-axis and compute the local outlier-ness of p in this subspace.

Before generalizing the selection of relevant subspaces in high-dimensional spaces, let us define some more notations. Formally, let $\vec{\mu}_n(j)$, $j \in N$ denote the jth axis-parallel unit vector in a n-dimensional space. Concretely, $\vec{\mu}_n(j)$ is a $n \times 1$ vector with the jth element one and others zero. Further, let v_p and v_q be the vector representation of point p and q, respectively, and v_q is the mean of all the points in RP(p). Correspondingly, the vector representation of line l can be written as $\vec{l}$, and $\vec{l} = v_p - v_q$. Here we define the jth element of vector $\vec{l}$ as l_j, i.e., $\vec{l} = [l_1, l_2, \ldots, l_n]^T$.

An alternative way to measure the angle between two lines is by the absolute value of the cosine between the two corresponding vectors. Let $|\cos(\vec{l}, \vec{\mu}_n(j))|$ denote the absolute

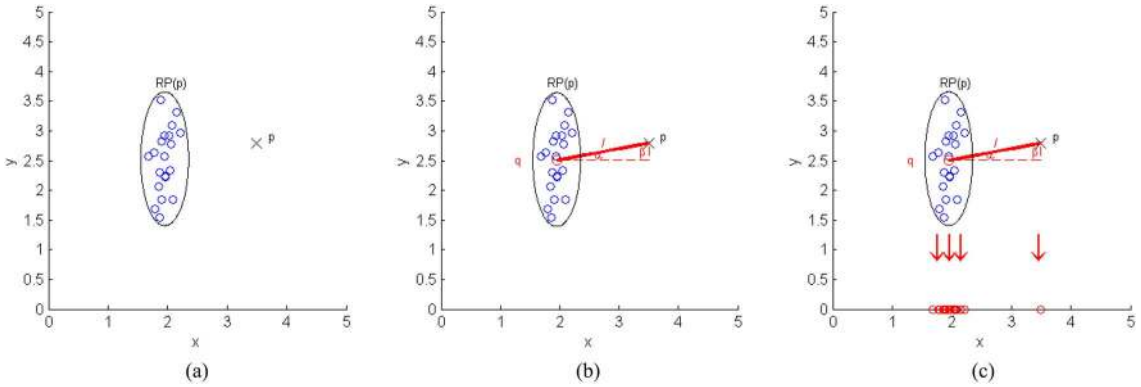

Fig. 2. Intuition of finding relevant subspace and subspace projection. (a) Problem setting. (b) Finding relevant subspace. (c) Subspace projecion.

value of cosine between vector $\vec{l}$ and the jth axis-parallel unit vector $\vec{\mu}_n(j)$

$$\left|\cos\left(\vec{l}, \vec{\mu}_n(j)\right)\right| = \frac{\left|<\vec{l},\ \vec{\mu}_n(j)>\right|}{\left\|\vec{l}\right\| \cdot \|\vec{\mu}_n(j)\|} \tag{3}$$

where $|\cdot|$ is the absolute value sign, $<\cdot,\cdot>$ represents inner product, and $\|\cdot\|$ calculates the norm of the embedded vector. The absolute value of a cosine lies in the range [0, 1]. Similar to the intuitive example, if the metric is close to one, the jth axis tends to be parallel to line l and hence should be retained in the subspace. Contrarily, if the metric is approaching zero, the jth axis is prone to be perpendicular to line l and instead should be excluded.

Unfortunately, pairs of random vectors in high-dimensional spaces are typically perpendicular to each other [32], [33]. Specifically, all the axis-parallel unit vectors tend to be orthogonal to vector $\vec{l}$ as dimensionality increases, i.e., $\lim_{n\to\infty} \cos(\vec{l}, \vec{\mu}_n(j)) = 0$ for all $j \in N$. Instead of measuring the cosine value of two vectors directly in a n-dimensional space, we can calculate the average absolute value of cosine between vectors $\vec{l}$ and $\vec{\mu}_n(j)$ in all possible combinations of 2-D spaces. Here the 2-D spaces comprise the jth dimension and the j^-th dimension ($j^- \in N\backslash\{j\}$), which is selected from all the remaining dimensions in N. Obviously, when examining the jth axis with line l, there are totally $n-1$ pairs of 2-D spaces (j, j^-). Further, we define a metric PCos (let us call it "pairwise cosine" in the sense that it is derived from 2-D spaces) to measure the relationship between a line and an axis in all possible 2-D spaces. To maintain a uniform notation, let $\mathrm{PCos}(\vec{l}, \vec{\mu}_n(j))$ denote the pairwise cosine between vector $\vec{l}$ and the jth dimension

$$\begin{aligned}\mathrm{PCos}\left(\vec{l}, \vec{\mu}_n(j)\right) &= \frac{1}{(n-1)} \sum_{j^- \in N\backslash\{j\}} \frac{\left|<\left[l_j^{\#}\ l_{j^-}^{\#}\right]^T, [1\ 0]^T>\right|}{\left\|\left[l_j^{\#}\ l_{j^-}^{\#}\right]^T\right\| \cdot \left\|[1\ 0]^T\right\|} \\ &= \frac{1}{(n-1)} \sum_{j^- \in N\backslash\{j\}} \frac{\left|l_j^{\#}\right|}{\sqrt{{l_j^{\#}}^2 + {l_{j^-}^{\#}}^2}}. \end{aligned} \tag{4}$$

In order to avoid a zero denominator, elements in $\vec{l}$ that are equal to zero should be substituted by a significantly small positive quantity ε (e.g., 10^{-5}), that is

$$l_j^{\#} = \begin{cases} l_j, & \text{if } l_j \neq 0 \\ \varepsilon, & \text{otherwise} \end{cases} \quad \text{for all } j \in N.$$

As with the metric defined in (3), the larger the metric PCos is, the more we should include the corresponding dimension in the subspace, and vice versa. Although this rarely happens in high-dimensional spaces, an exceptional case arises when vector $\vec{l}$ is a zero vector. This implies that point p is overlapping with the geometric center of its adjacent points. Intuitively, it should be considered normal in a local sense. Thus, its outlier score can be simply set to zero.

Now we will discuss the expectation and variance of the metric pairwise cosine. If $\mathrm{PCos}(\vec{l}, \vec{\mu}_n(j))$, $j \in N$ is regarded as a random variable, its expectation will be as follows (derivation is provided in the Appendix):

$$\mathrm{E}\left[\mathrm{PCos}\left(\vec{l}, \vec{\mu}_n(j)\right)\right] = \frac{1}{n \cdot (n-1)} \sum_{\substack{j, j^- \in N \\ j^- \neq j}} \frac{\left|l_j^{\#}\right| + \left|l_{j^-}^{\#}\right|}{\sqrt{{l_j^{\#}}^2 + {l_{j^-}^{\#}}^2}}. \tag{5}$$

Notice that PCos is basically the average absolute value of cosine and it naturally lies in the range [0, 1]. Therefore, we have the following proposition (proof can be referred to [30]).

Proposition: The expectation in (5) lies in the interval $(1/2, \sqrt{2}/2]$ and does not depend on the magnitude of dimensionality.

Besides, the expectation and variance of PCos tend to be asymptotically stable as dimensionality increases. As shown in Fig. 3(a) and (b), the mean of PCos that is derived from a uniformly distributed dataset and a normally distributed dataset, both with 10^5 samples, are plotted against increasing dimensionalities and gradually converge to a value around 0.65 (not exactly). Even though the variance of this metric is analytically intractable from our knowledge, as demonstrated in Fig. 3(c) and (d), it again tends to level off and be rather stable as dimensionality increases. Notably, the asymptotic property of the expectation and variance of the metric PCos holds even for samples with large order of magnitude based on our experiments.

As elaborated before, the dimensions with large PCos values should be incorporated into the subspace, whereas the dimensions with small PCos values should be excluded. For each data point in the dataset $\boldsymbol{X}$, there exists one particular line l

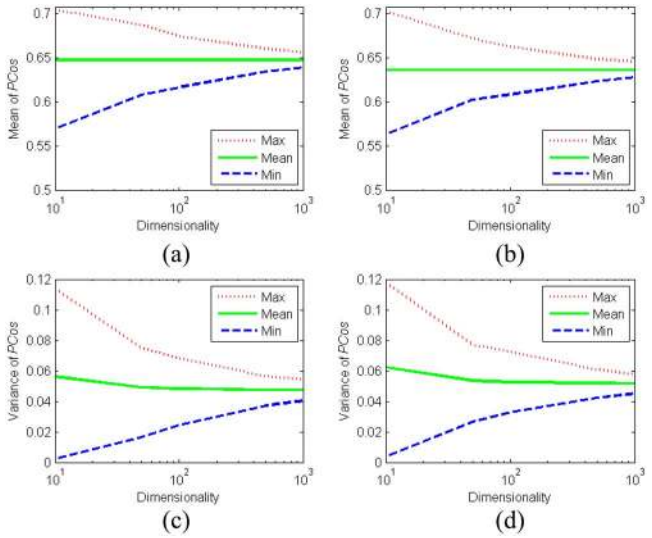

Fig. 3. Asymptotic property of expectation and variance of the metric PCos. Asymptotic property of (a) expectation: uniform, (b) expectation: Gaussian, (c) variance: uniform, and (d) variance: Gaussian.

and we can obtain n different values of PCos by applying (4) iteratively. It has been justified that PCos is a relatively robust metric, so for a specific data point, we can set a threshold G to single out those large PCos values as

$$G = (1+\theta)\cdot\frac{1}{n}\sum_{j\in N}\mathrm{PCos}\left(\vec{l},\vec{\mu}_n(j)\right). \tag{6}$$

Here, θ is an input parameter lying in [0, 1), and n is the number of dimensions of the feature space N. The right part of the multiplier in (6) is essentially the average PCos over all the dimensions. Those dimensions with a PCos value greater than G are retained to constitute the relevant subspace for a specific data point.

F. Computation to Local Outlier Scores

After going through the process of selecting relevant subspaces, we might find some of the points do not have any relevant attributes being retained as a part of the subspace. This circumstance simply indicates that those points do not significantly deviate from their neighbors in any subsets of all the dimensions. For those points with no subspace to project on, we simply set the outlier score to zero.

In the following, we describe how to measure the local outlier-less of a data point in its subspace. Generally, some state-of-the-art anomaly detection techniques (e.g., distance-based, density-based, and statistical models) which perform well in low-dimensional spaces can be employed here. For example, in the SOD algorithm [21], Euclidean distance was used to calculate the local outlier-ness of a point in its subspace projection. However, in fault detection applications, different dimensions tend to be correlated to each other due to the interaction between subsystems and components. This correlation may lead to a failure of some distance metrics in defining the extent of outlier-ness. For this reason, we introduce a normalized Mahalanobis distance to measure the LOS of a specific data point. First, let $x_i^{*'}$ be the projection of the ith normalized point on the retained subspace, and $\mathrm{RP}(i)'$ denotes the subspace projection of the original reference points $\mathrm{RP}(i)$. Second, the mean vector of the reference point's projection is denoted as $\overline{\mathrm{RP}(i)'}$, and the inverse of the covariance matrix of $\mathrm{RP}(i)'$ is $\Sigma^{-1}_{\mathrm{RP}(i)'}$. Further, let $d(i)$ denote the number of retained dimensions for the ith data point. Then the LOS for the ith point $\mathrm{LOS}(i)$ is defined as follows:

$$\mathrm{LOS}(i) = \frac{1}{d(i)}\cdot\sqrt{\left(x_i^{*'}-\overline{\mathrm{RP}(i)'}\right)^T\Sigma^{-1}_{\mathrm{RP}(i)'}\left(x_i^{*'}-\overline{\mathrm{RP}(i)'}\right)}. \tag{7}$$

In (7), the right side of the multiplier is basically the Mahalanobis distance from the normalized point i to its reference points in the subspace projection. Essentially, the overall LOS for the ith point $\mathrm{LOS}(i)$ is the Mahalanobis distance in the retained subspace normalized by the number of retained dimensions.

Notably, for the covariance matrix of the projected reference points $\Sigma_{\mathrm{RP}(i)'}$ to be invertible, the covariance matrix should be nonsingular. The nonsingularity of the covariance matrix relies on the following three prerequisites.

1) In the data preparation step, feature selection has eliminated redundant attributes resulting in the retained dimensions in the subspace not being highly correlated.
2) We assumed that the anomaly-relevant attributes are in a minority of all the dimensions in Section II-A and the process of selecting relevant subspaces described in Section II-E should filter out large amount of irrelevant attributes, and hence the number of retained dimensions is small.
3) The choice of the number of reference points s (as will be discussed in Section IV-B) should be set large enough.

The above three conditions can basically suffice for the nonsingularity of the covariance matrix.

G. Determination of the Control Limit for Reporting Faults

In general, anomaly detection can be considered as a skewed binary classification problem. Unfortunately, there is no deterministically accurate criterion to convert the continuous LOS to a binary label, namely normal or faulty. One way is to set a control limit and those data points with an LOS exceeding the control limit should be regarded as anomalies. The objective of introducing one additional control limit is to reduce the probability of committing both type I (FPR) and type II (FNR) error.

For the purpose of fault detection, the control limit of LOSs may vary among different applications. In case of no sufficient labeled data, some probabilistic methods can be used to set the control limit. To automate the process of setting control limits, we regard $\mathrm{LOS}(i),\ i\in\{1,2,\ldots,m\}$ as the observations of a random variable s and apply the KDE method to estimate its probability density function $\hat{f}(s)$, as has been done in [17] and [18]. Then the control limit can be set according to $\hat{f}(s)$ and a confidence level $(1-\gamma)$ given by the user. In our univariate case, the KDE method places a kernel at the location of each observation from the sample set and sums up these kernels to get the estimation of the probability density

Algorithm 1: ABSAD$(\boldsymbol{X}, k, s, \theta, \gamma)$

BEGIN
Initialize LOS;
Conduct feature normalization on $\boldsymbol{X}$, and save it to matrix $\boldsymbol{X}^*$;
Derive k nearest neighbors using dist$(\cdot,\cdot)$ on $\boldsymbol{X}^*$, and save it to matrix $\mathbf{NN}_k$;
Derive s nearest neighbors based on $\mathbf{NN}_k$ and the SNN similarity measure, and then save it to matrix **SNN**;
FOREACH $v_i \in \boldsymbol{X}^*$
Calculate the mean vector of RP(i), save it to vector v_q;
Connect point i with q to form a line vector $\vec{l}$;
FOREACH $j \in N$
Compute PCos$\left(\vec{l}, \vec{\mu}_n(j)\right)$ and save it to the jth element of vector PCos;
END
Determine the threshold G based on PCos and θ;
Select relevant subspace, then project v_i and RP(i) on the retained subspace, and then compute LOS(i);
END
Calculate the control limit CL based upon LOS and the significance level γ;
RETURN (LOS, CL);
END

Fig. 4. ABSAD algorithm in a batch mode.

function over the entire sample set. The kernel density estimator of the function $\hat{f}(s)$ is defined in (8), where s is the concerned random variable, s_i is the value of the ith sample, m is the sample size, h is the smoothing parameter named the bandwidth, and $\mathrm{K}(\cdot)$ is the kernel function, of which the Gaussian kernel function is the most widely used one

$$\hat{f}(s) = \frac{1}{mh}\sum_{i=1}^{m} \mathrm{K}\left(\frac{s - s_i}{h}\right). \tag{8}$$

As described previously, the LOS of those points with no subspace to project on will be set to zero. A mass of these LOSs at the location zero may dominate the probability density of s. To reduce the bias in the estimation of the underlying probability density function, we simply substitute these zero values with the half of the minimum nonzero value prior to the estimation of $\hat{f}(s)$

$$s_i = \begin{cases} \mathrm{LOS}(i), & \text{if } \mathrm{LOS}(i) \neq 0 \\ \min\limits_{i\in\{1,\ldots,m\}} \{\mathrm{LOS}(i) \big|\ \mathrm{LOS}(i) \neq 0\}/2, & \text{otherwise.} \end{cases} \tag{9}$$

Now, given a finite set of LOSs and a predefined confidence level, we should be able to determine a control limit for deciding whether an observation is normal or anomalous.

H. Model Integration

In keeping with the computational procedure of the ABSAD approach as described in Section II-B, we define an integrated algorithm in Fig. 4. The algorithm takes in a finite number of data points and some user-defined parameters, and outputs a vector of LOSs and a control limit. In contrast to some online fault detection algorithms where samples are evaluated one at a time upon their arrival, the ABSAD algorithm shown in Fig. 4 can be viewed as an algorithm running in a batch mode.

The algorithm shown in Fig. 4 can be easily adapted to an online mode to monitor system states. Let us call this online mode of the ABSAD approach "primitive ABSAD." The primitive ABSAD approach first conducts offline training on a finite size of normal data points (a fixed window) and obtains the control limit. For any new observation from the data stream, we calculate its LOS with respect to the original training set following the same procedure listed in Fig. 1. If the LOS exceeds the control limit, a fault is detected. The crucial problem with the primitive ABSAD is that the control limit and the data points in the window are fixed. They are not changing along with the system's normal time-varying behavior. Consequently, the type I error of the fault detection task may be high (as will be shown in Section IV-C). Of course, the batch mode of ABSAD approach can also be run regularly to absorb the normal change of the system. But it requires intensive computation, which is unsuitable for online fault detection that demands timely response.

III. Sliding Window ABSAD-Based Fault Detection Scheme

To deal with the above problems, we adapt the ABSAD approach to another online mode based on the sliding window strategy in this section. The sliding window strategy is frequently used in data stream mining and it assumes that recent data bear greater significance than historical data. It discards old samples from the window, inserts new samples into the window, and updates the parameters of the model iteratively. Since the "sliding window ABSAD" algorithm is adaptive to the dynamic change of the system, it can reduce the type I error significantly compare to the primitive ABSAD algorithm (as will be shown in Section IV-C). At the end of this section, we analyze the computational complexity of the sliding window ABSAD algorithm.

A. Structure of the Sliding Window ABSAD Algorithm

The structure of the sliding window ABSAD algorithm is shown in Fig. 5. It comprises two stages: 1) offline model training and 2) online fault detection. The first stage, offline model training, is a one-off task followed by the online fault detection routine. The second stage continuously processes each new observation from the data stream upon its arrival. To enhance the computational efficiency of the algorithm, the current window profile needs to be stored and maintained continuously. The different stages of the algorithm are explained below.

1) Offline Model Training: The first stage basically follows the same procedure of the ABSAD approach listed in Fig. 1. Notably, it is worth meticulously selecting fault-free samples to construct the training set since the existence of faulty samples in the training set may potentially reduce the LOS of a new observation and augment the control limit, and hence increase the risk of committing the type II error. After the completion of the first stage, the output is used to initialize

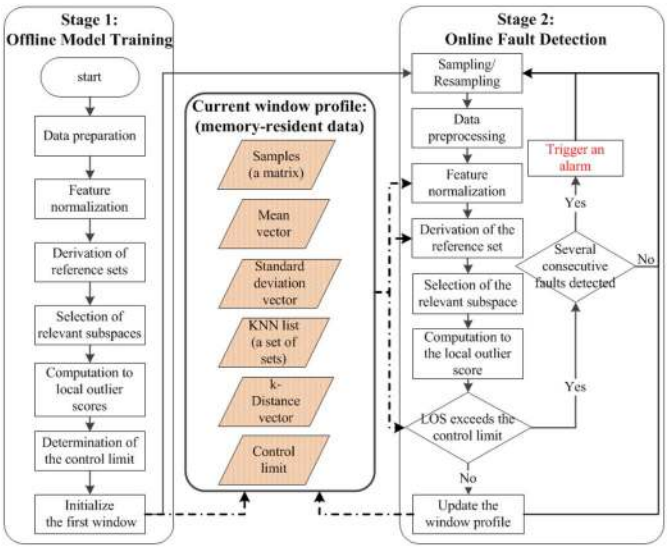

Fig. 5. Structure of the sliding window ABSAD algorithm.

the profile of the first window. Then, the algorithm is prepared for the succeeding online fault detection routine.

2) Online Fault Detection: The second stage continuously processes the data stream and monitors the states of the system. In accordance with the flow chart of the online fault detection stage shown in Fig. 5, the concrete steps are explained as follows. First, the first two steps, sampling and data preprocessing, collect the raw data and transform it into a required form that is appropriate for mining. Specifically, the sampling step acquires real-time measurements in a raw format. Then the data preprocessing step transforms the raw data into a suitable format which is in line with the output format of the data preparation step in the first stage. Second, the subsequent four steps calculate the local outlier-ness of a new sample. They again align with the steps listed in Fig. 1. The difference is that there is only one observation going through these steps at a time in order to be processed. In addition, the information stored in the current window profile should be utilized in these two steps: 1) feature normalization and 2) derivation of the reference set, as indicated by the dashed arrow in Fig. 5. Concretely, according to (1), the feature normalization step standardizes the new sample based on the mean vector and the standard deviation vector stored in the current window profile. In addition, the reference set of the new sample originates from the most recent normal samples maintained in the current window profile. Calculating the SNNs of a coming online sample with respect to all the samples in the current window profile yields its reference points. According to (2), the KNN list in the window profile can also speed up the derivation of the reference set of the new sample. After going through the same process described in Sections II-E and II-F, the LOS of the new sample can be calculated. Lastly, we may regard the remaining parts of the second stage as the post-processing steps. On the one hand, if the obtained LOS exceeds the control limit in the current window profile, a possible fault is then detected and the process starts over from the resampling step. Meanwhile, if several consecutive faults are detected, an alarm should be triggered.

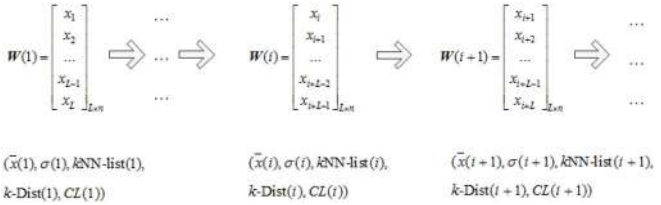

Fig. 6. Transition of the window profile.

In this case, the window profile should not be updated. On the other hand, if the LOS is less or equal than the control limit, the current window profile should be updated to incorporate the normal change of the system and then the process goes back to the resampling step.

In the above algorithm, the first stage learns the normal behavior of the monitored system and stores the key information into the first window. The second stage continuously detects whether a new sample is normal or faulty based on the information preserved in the current window profile. If a new sample is judged to be normal, the information contained in this sample will be absorbed into the new window. Correspondingly, the information of the oldest sample will be discarded. By doing so, the latest normal behavior of the system can always be incorporated into the current window and serves as the basis for dynamic fault detection. Among all the steps in the sliding window ABSAD algorithm, updating the window profile is the most critical and complex one. The updating mechanism will be elaborated in the next section.

B. Update to the Current Window Profile

Through updating the current window profile regularly, the sliding window strategy enables the online fault detection to adapt to the time-varying behavior of the system. Based on the requirements of the ABSAD approach, six items are identified to be preserved and maintained in the window profile, i.e., samples, mean vector, standard deviation vector, KNN list, k-distance vector, and the control limit, as shown in Fig. 6. The following contents will mainly discuss how to update these items.

Before looking at the details, let us make some more notations. We define $\boldsymbol{W}(i)$ as all the samples in the ith window with a window size L. As shown in Fig. 6, $\boldsymbol{W}(i)$ is a L by n matrix, in which each row represents one sample (e.g., x_i is the first sample in the ith window) and n is the number of dimensions in the feature space. Notably, the window profile is updated only under the condition that a new sample is judged to be normal. Therefore, the samples preserved in a window may not be consecutive in the time scale even though the samples in a window are sequentially indexed. Further, let $\bar{x}(i)$, $\sigma(i)$, kNN-list(i), k-Dist(i), and CL(i) be the columnwise mean vector, columnwise standard deviation vector, KNN list (a set of sets, containing the k nearest neighbors of every sample in the ith window), k-distance vector, and the control limit of the ith window correspondingly. The initialization to the first window after the offline model training stage is rather straightforward. Now we elaborate the updating mechanism to the six items of the window profile from the ith window to the $(i + 1)$th window as below.

1) Update to the Samples: Obviously, in the case of updating samples from the ith window to the $(i+1)$th window, x_i is the oldest sample, and x_{i+L} is the latest normal sample that should be absorbed into the new window. Hence, $\boldsymbol{W}(i+1)$ is simply obtained by deleting x_i from $\boldsymbol{W}(i)$ and adding x_{i+L} to the end of $\boldsymbol{W}(i)$.

2) Update to the Mean Vector and the Standard Deviation Vector: The mean vector $\bar{x}(i+1)$ and the standard deviation vector $\sigma(i+1)$ can be updated from the previous window profile by applying (10) and (11). Notably, all of the operations in these two equations should be conducted in an elementwise manner

$$\bar{x}(i+1) = \bar{x}(i) + \frac{1}{L} \cdot (x_{i+L} - x_i) \tag{10}$$

$$\sigma(i+1) = \left[\sigma^2(i) - \frac{L+1}{L \cdot (L-1)} \cdot x_i^2 + \frac{1}{L} \cdot x_{i+L}^2 + \frac{2}{L-1} \cdot \left(x_i \cdot \bar{x}(i) - x_{i+L} \cdot \bar{x}(i) + \frac{1}{L} \cdot x_{i+L} \cdot x_i\right)\right]^{1/2}. \tag{11}$$

3) Update to the KNN List and the k-Distance Vector: The set kNN-list(i) records the k nearest neighbors of every sample in the ith window. As mentioned in Section III-A, it is used when deriving the SNNs of the new sample, i.e., the reference set. In the ith window, the vector k-Dist(i) stores the k-distance of each sample, i.e., the distance to the kth nearest neighbor from each sample. Even though k-Dist(i) does not directly contribute to the calculation of the new sample's LOS, it facilitates the updating to the KNN list.

For a sample in the ith window with the index j, where $j \in \{i+1,\ i+2, \ldots i+L-1\}$, we define kNN-list$(i)_j$ to be the k nearest neighbor list of this sample x_j. Correspondingly, let k-Dist$(i)_j$ be the distance to the kth nearest neighbor from x_j in the ith window. Note that, kNN-list$(i)_j$ is a set containing the index of k samples that originates from the ith window and k-Dist$(i)_j$ is a scalar.

Moving from the ith window to the $(i+1)$th window, the whole neighborhood relationship is updated by removing the information about x_i away and adding the information about x_{i+L}. In the following, we consider these two steps sequentially.

1) *Remove the Information of the Oldest Sample:* If the first sample x_i is among the k nearest neighbor list of x_j, we should remove it and then add the $(k+1)$th nearest neighbor to its k nearest neighbor list. Correspondingly, the k-distance should be updated with the distance to the $(k+1)$th nearest neighbor from sample x_j. Formally, the KNN list and the k-distance vector of the ith window should be updated as follows:

$$k\text{NN-list}(i)_j = \left(k\text{NN-list}(i)_j \backslash \{x_i\}\right) \cup \left\{x_{j(k+1)}\right\} \quad \text{if } x_i \in k\text{NN-list}(i)_j \text{ and } j \in \{i+1, i+2, \ldots, i+L-1\} \tag{12}$$

$$k\text{-Dist}(i)_j = (k+1)\text{-Dist}(i)_j \quad \text{if } x_i \in k\text{NN-list}(i)_j \text{ and } j \in \{i+1, i+2, \ldots, i+L-1\} \tag{13}$$

where $x_{j(k+1)}$ represents the $(k+1)$th nearest neighbor of the sample x_j in the ith window, and $(k+1)$-Dist$(i)_j$ denotes the distance to the $(k+1)$th nearest neighbor from sample x_j in the ith window.

2) *Add the Information of the New Sample:* In this step, the distance from the new sample x_{i+L} to the samples in the ith window (except for the first sample) should be evaluated first. We define dist(x_{i+L}, x_j), where $j \in \{i+1, i+2, \ldots, i+L-1\}$, as the distance from the new sample x_{i+L} to the sample x_j. By sorting these distances in ascending order, we get the k nearest neighbor list and the k-distance of the new sample in the next window, i.e., kNN-list$(i+1)_{i+L}$ and k-Dist$(i+1)_{i+L}$. Intuitively, if the k-distance of the sample x_j is greater than the distance from x_j to x_{i+L}, the k nearest neighbor list and the k-distance of x_j should be updated. Formally, we update the KNN list and the k-distance vector as follows:

$$k\text{NN-list}(i)_j = \left(k\text{NN-list}(i)_j \backslash \left\{x_{jk}\right\}\right) \cup \{x_{i+L}\} \quad \text{If } \text{dist}\left(x_{i+L}, x_j\right) < k\text{-Dist}(i)_j \text{ and } j \in \{i+1, i+2, \ldots, i+L-1\} \tag{14}$$

$$k\text{-Dist}(i)_j = \max\left\{(k-1)\text{-Dist}(i)_j, \text{dist}\left(x_{i+L}, x_j\right)\right\} \quad \text{If } \text{dist}\left(x_{i+L}, x_j\right) < k\text{-Dist}(i)_j \text{ and } j \in \{i+1, i+2, \ldots, i+L-1\} \tag{15}$$

where x_{jk} represents the kth nearest neighbor of x_j in the ith window and $(k-1)$-Dist$(i)_j$ denotes the distance to the $(k-1)$th nearest neighbor from sample x_j in the ith window.

Finally, the KNN list and the k-distance vector of the $(i+1)$th window can be obtained by removing the first element and adding the corresponding element of the new sample x_{i+L} to the end of kNN-list(i) and k-Dist(i). For example, the KNN list of the $(i+1)$th window can be set as follows:

$$k\text{NN-list}(i+1) = \left(k\text{NN-list}(i) \backslash \left\{k\text{NN-list}(i)_j\right\}\right) \cup \left\{k\text{NN-list}(i+1)_{i+L}\right\}.$$

4) Update to the Control Limit: The control limit is used for judging whether a new sample is faulty or normal. When a normal sample with a high LOS is absorbed into the new window, the tolerance of the new window to a high LOS on the same level should be augmented slightly, and vice versa. Hence, the control limit should also be renewed each time after a normal sample is added to the current window. The control limit can be reset based on (8) and (9) together with the confidence level as described in Section II-G. To ensure computational efficiency, we only update the control limit when the LOS is greater than zero.

C. Computational Complexity Analysis

Computational complexity is one key factor in assessing the merit of an algorithm for data stream mining. In addition, it is crucial in fault detection applications which require timely

response from the monitoring algorithm to ensure system safety. In the following, we discuss the time complexity and space complexity of the sliding window ABSAD algorithm.

For the first stage of the sliding window ABSAD algorithm (the batch mode of the ABSAD approach), the time complexity and space complexity are $O(L^2 \cdot \max(n,k))$ and $O(L \cdot \max(n,k))$, respectively, considering L is typically much larger than n and k. If some indexing structures like k-d tree or R^* tree are employed here, the algorithm's time complexity can be reduced to $O(L\log L \cdot \max(n,k))$. Given such demanding computations, it is unadvisable to repeatedly run the batch mode of the ABSAD approach to absorb information from the latest normal sample in online fault detection applications. This is precisely the reason why we extend the original ABSAD approach to the sliding window-based ABSAD.

The second stage of the sliding window ABSAD algorithm continuously processes new samples upon their arrival. The computational complexity of this stage is more important in the sense that it decides whether the algorithm can isochronously monitor the state of the system. By analyzing each step of the second stage, the time complexity of this stage for processing a single sample is $O(L\cdot\max(n,k))$ and the space complexity is $O(L\cdot\max(n,k))$. Although the above time complexity is not linear, it is still rather attractive in the context of dealing with high-dimensional data streams. Notably, to accelerate the processing speed of online fault detection, the sliding window ABSAD algorithm designates a space for storing the window profile and continuously maintains it. The window profile does not only contain the critical parameters for calculating the LOS of a new sample and detecting whether it is faulty or not (such as the mean vector), but also includes those parameters for maintaining the window profile itself (such as the k-distance vector). This is where the concept of trading space for time applies.

IV. Numerical Illustration

This section validates the efficacy of the above-described sliding window ABSAD algorithm on synthetic datasets. To do so, we contrast it with the primitive ABSAD, "primitive LOF," and "sliding window LOF" algorithms. Here the LOF-based algorithms are selected for comparison for the reason that the LOF approach is one of the most well-known density-based unsupervised anomaly detection techniques. The LOF approach computes the average ratio of the local reachability density of a point and those of the point's nearest neighbors [34]. In the literature, several studies have also chosen the LOF approach as an alternative to compare with their methods. Some of these examples can be referred to [19] and [21]. Similar to the primitive ABSAD, the primitive LOF approach applies the original LOF algorithm to calculate the local outlier-ness of a new sample over a fixed set of samples. By adopting the sliding window strategy, the sliding window LOF approach with a dynamically updated window was proposed and applied in process fault detection applications [19]. The sliding window LOF approach was reported to exhibit a superior accuracy compared to PCA-based models and can be adaptive to the time-varying characteristics of the monitored system. It does, however, suffer from the curse of dimensionality which leads to the degradation of its accuracy as dimensionality increases, as will be shown in Section IV-C.

A. Data Generation

Consider the following system which is modeled on an input-output form, similar to the example used in [19]:

$$\begin{aligned}\boldsymbol{O}(t) &= \boldsymbol{A}\cdot\boldsymbol{I}(t)+\boldsymbol{E}(t)\\ &= \begin{bmatrix} 0.86 & 0.79 & 0.67 & 0.81\\ -0.55 & 0.65 & 0.46 & 0.51\\ 0.17 & 0.32 & -0.28 & 0.13\\ -0.33 & 0.12 & 0.27 & 0.16\\ 0.89 & -0.97 & -0.74 & 0.82\end{bmatrix}\cdot\begin{bmatrix} I_1(t)\\ I_2(t)\\ I_3(t)\\ I_4(t)\end{bmatrix}\\ &\quad+\begin{bmatrix} e_1(t)\\ e_2(t)\\ e_3(t)\\ e_4(t)\\ e_5(t)\end{bmatrix}\end{aligned} \tag{16}$$

where t is the temporally ordered sampling index and $t \in \{1, 2, \ldots 2000\}$, $\boldsymbol{I}(t)\epsilon R^{4\times 2000}$ is the input matrix which consists of four input variables $I_1(t),\ I_2(t),\ I_3(t),\ I_4(t)$, $\boldsymbol{O}(t) \in R^{5\times 2000}$ is the output matrix that comprises five monitored variables $O_1(t),\ O_2(t),\ O_3(t),\ O_4(t),\ O_5(t)$, $\boldsymbol{A}$ is the parameter matrix with appropriate dimensions, and $\boldsymbol{E}(t) \in R^{5\times 2000}$ encompasses five random noise variables, each of which is normally distributed with a zero mean and a variance equal to 0.02.

Further, the data of the four input variables are generated as follows:

$$I_1(t) = 2\cdot\cos(0.08t)\ \cdot\sin(0.006t) \tag{17}$$

$$I_2(t) = \text{sign}[\sin(0.03t)\ + 9\cdot\cos(0.01t)] \tag{18}$$

$$I_3(t) \sim U(-1,\ 1) \tag{19}$$

$$I_4(t) \sim N(2,\ 0.1). \tag{20}$$

Based on the above data generating mechanism, we construct four datasets (2000 samples and five dimensions in each) with different types of faults all induced starting from the 1501st sample. The faults are as follows.

1) *Scenario 1 (Fault 1):* An abrupt drift is placed on the fifth output variable $O_5(t)$ with a magnitude of 5.
2) *Scenario 2 (Fault 2):* An abrupt drift is injected into the fourth input variable $I_4(t)$ with a magnitude of -5.
3) *Scenario 3 (Fault 3):* An abrupt drift is injected into the second input variable $I_2(t)$ with a magnitude of -3.
4) *Scenario 4 (Fault 4):* A ramp change $-0.1\times(t-1500)$ is added to the first input variable $I_1(t)$.

To simulate the time-varying behavior of the system, we add a gradually slow drift $0.003(t-1000)$ to two entries of the parameter matrix $\boldsymbol{A}(1,2)$ and $\boldsymbol{A}(2,2)$ starting from the 1001st sample. In the end, we append another 95 fault-irrelevant dimensions to each of these four datasets to create a high-dimensional setting. All the fault-irrelevant dimensions are distributed uniformly on [0, 1].

Finally, four datasets with 2000 samples and 100-D in each are constructed. For all of these datasets, the first 1500 samples are normal and the last 500 samples are faulty. Among these normal samples, a slight change has been gradually placed

on the ones with sample index from 1001 to 1500. A decent online fault detection algorithm should not only be able to detect the faulty samples but also be adaptive to the time-varying behavior of the system. In other words, the algorithm should reduce both type I error and type II error as much as possible.

B. Parameter Analysis and Tuning

In sliding window-based algorithms, it is crucial to choose an appropriate window size L. A large window size may endow high model accuracy but result in intensively computational load. By contrast, a small window size indicates low complexity in computation but may lead to low model accuracy. An exploratory test was performed to probe the effect of different window sizes on the two types of error of the sliding window ABSAD algorithm, and the results are shown in Fig. 7(a). In this test, the dataset associated with the second fault was used. Additionally, parameter k and s for deriving the reference set were set to be one fourth of the window size, i.e., $k = s = L/4$, for simplicity. Parameter θ for selecting relevant subspace was set at 0.4 and the confidence level $1-\gamma$ for deciding the control limit was set at 99%. From the results shown in Fig. 7(a), the window size primarily affects the type I error in this example. Further, a small window size may lead to a higher type I error, which is mainly because of the lack of representative neighboring points in the window to support the normality of a normal sample. On the other hand, the type I error tends to increase slightly as the window size goes above 900, and that may be caused by the inadequacy of the model to adapt to the time-varying characteristics of the system. Thus, an ideal range of the window size for this case may be chosen from 600 to 900.

Similarly, parameters k and s also matter to the model accuracy and the computational burden. First, parameter k specifies the number of nearest neighbors for computing SNN similarity. As with some other algorithms related to the SNN method, k should be set large enough so as to capture sufficient points from the same generating mechanism. As reported in [13], if k is chosen roughly in the range of cluster size then a considerably satisfactory performance in terms of defining the notion of locality can be achieved. Second, parameter s defines the size of the reference sets. For the same reason, it should be chosen large enough but not greater than k. In [13], it was shown that the performance of the SNN method does not degrade until the size of reference points approaches the full dataset size. To investigate the effect of these two parameters on the two types of errors, a similar test on the dataset containing the second fault was conducted and the results are shown in Fig. 7(b). Again in this test, for simplicity, parameters k and s were set to be equal. Other parameters were set as follows: $L = 750$, $\theta = 0.4$, and $1-\gamma = 99\%$. As shown in Fig. 7(b), parameters k and s primarily affect the type II error in this case. A small value of k and s may lead to a high type II error which is mainly because of insufficient neighboring points in the window to discriminate a faulty sample from normal ones. In accordance with the above analysis to parameter k and s, Fig. 7(b) indicates that satisfactory model accuracy can be obtained as long as k and s are set large enough. From the perspective of model accuracy, k and s should be larger than 40 based on the results shown in Fig. 7(b). However, k and s should not be set too large to lose the meaning of defining the notion of locality. In addition, they should not be set too large in considering the computational efficiency.

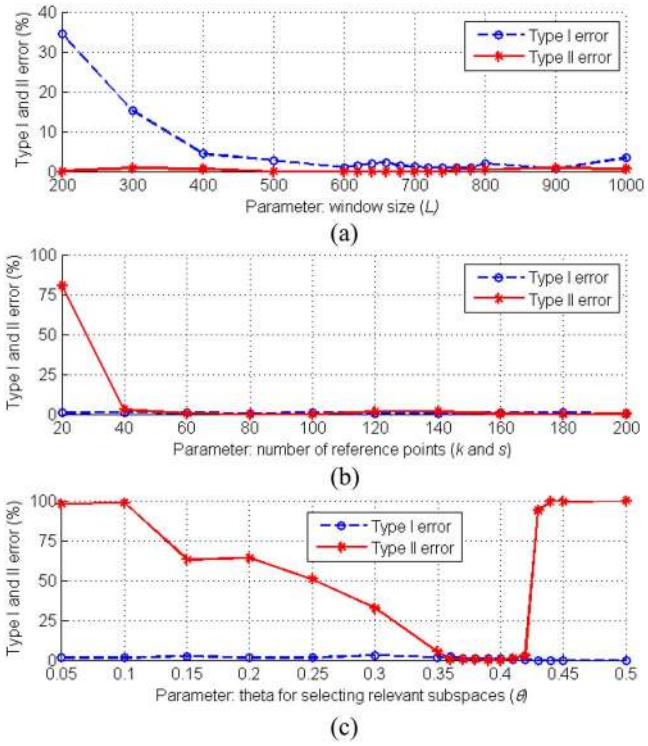

Fig. 7. Effect of different parameters on the two types of error. Effect of (a) window size on the two types of error, (b) number of reference points on the two types of error, and (c) theta on the two types of error.

The last parameter θ decides which dimensions should be kept as a part of the relevant subspace. It may have a great influence on selecting relevant subspace and hence affect the subsequent calculation of the LOS. Generally, the lower the value θ, the more dimensions will be included in the subspace, and vice versa. As with the above two tests, the dataset containing the second fault was selected to explore the effect of θ on the two types of error and the results are shown in Fig. 7(c). Other parameters were set as follows: $L = 750$, $k = s = 100$, and $1-\gamma = 99\%$. As demonstrated by Fig. 7(c), parameter θ primarily affects the type II error in this example. If θ is set too small, a large share of dimensions which have less significance in defining the local outlier-ness of a point will be retained, hence reduce the LOS of a faulty sample. Conversely, if θ is set too large, the algorithm can capture very few fault-relevant dimensions or even no dimensions to construct the subspace, and hence malfunction in detecting faulty samples. According to the results shown in Fig. 7(c), the acceptable range of parameter θ is from 0.36 to 0.42.

Based on the above three tests regarding tuning parameters and the tradeoff between complexity of computation and model accuracy, the window size is set at 750, k and s are set at 100, and θ is chosen to be 0.4 for the sliding window ABSAD algorithm in the following simulation. Moreover, the parameters of the algorithm primitive LOF and sliding window

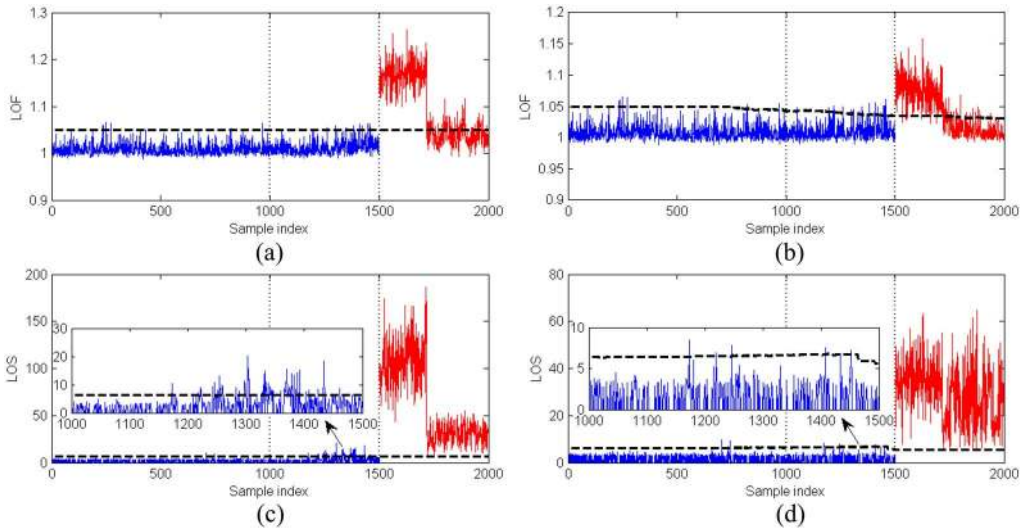

Fig. 8. Fault detection results of (a) primitive LOF, (b) sliding window LOF, (c) primitive ABSAD, and (d) sliding window ABSAD in scenario 2 of the numerical example.

TABLE I
FAULT DETECTION RESULTS OF THE NUMERICAL EXAMPLE

Dataset and error type (Units of numbers are in percentage: %)		Primitive LOF	Sliding window LOF	Primitive ABSAD	Sliding window ABSAD
Fault 1	Type I error	1.73	1.73	8.4	0.67
	Type II error	32.2	91.8	0.2	4.4
Fault 2	Type I error	2.4	3.73	8.4	0.8
	Type II error	38.8	51	0	0
Fault 3	Type I error	2.8	2.27	8.13	1.2
	Type II error	0	36.4	0	0
Fault 4	Type I error	2.13	1.87	8.8	0.67
	Type II error	4.8	6.8	3.8	4.2

LOF for comparison are set exactly the same as the settings in [19], i.e., $L = 750$ and $k = 30$. For all of these methods, the confidence level $1 - \gamma$ will be 99%.

C. Results and Discussions

The results of the four fault detection algorithms on the four datasets (associated with the four different faults) are summarized in Table I. Although the type I errors of LOF-related algorithms are rather low in all of the four scenarios, this is mainly caused by the insensitivity of LOF to faults that exist only in small subsets of high-dimensional spaces. As a result of this, the type II errors of LOF-related algorithms are significantly high in the first two scenarios. A further explanation is that LOF-related algorithms are implemented in full-dimensional spaces and those signals relevant to the faults can be easily concealed by the massive fault-irrelevant dimensions (the 95 uniformly distributed dimensions in this example). Fig. 8(a) and (b) gives a graphical description of the above result. To alleviate the impact exerted by irrelevant dimensions, the proposed ABSAD approach finds fault-relevant dimensions first and then measures the local outlier-ness of a point in its retained subspace. By doing so, the power of discriminating low-dimensional subspace faults from normal samples in high-dimensional spaces can be greatly enhanced. Consequently, the type II errors produced by ABSAD-related algorithms are relatively low as shown in Table I.

The results in Table I also indicate that the primitive ABSAD has a higher level of type I errors in contrast to the sliding window ABSAD. As shown in the partially enlarged inset of Fig. 8(c), we can precisely locate the position of false alarms, where the blue line (LOS) exceeds the black dashed line (control limit). The reason for these false alarms is that the primitive ABSAD always holds the same window after the offline model training stage. The parameters of the model are invariant and thus cannot be adaptive to the time-varying behavior of the system. Instead of keeping a constantly unchanged window, the sliding window ABSAD absorbs new samples and discards old samples regularly and changes the window profile dynamically. As demonstrated by the partially enlarged inset in Fig. 8(d), the sliding window ABSAD algorithm adapts to the time-varying behavior of the system very well and very few false alarms are generated on the samples where the slow drift was added.

In the dataset containing fault 4, the degree of deviation of the fault from normal behavior of the system is remarkably higher than the other three faults. Therefore, LOF-related algorithms can still produce a desirable accuracy in terms of low type I and type II errors, as shown in Table I and Fig. 9. It is worthy to note that, according to Fig. 9, there is a huge difference between the scale of the values of LOS and LOF. Specifically, the LOS values are orders of magnitude higher than the LOF values. This difference can also be found in other scenarios. The leading cause of this phenomenon lies in the fact that the deviation on fault-relevant dimensions was considerably compensated by the normal behavior on massive fault-irrelevant dimensions. As a consequence, the obtained LOF values are vastly reduced even in the case of the faults being very evident, such as in scenario 4 as shown in Fig. 9.

To further evaluate the proposed approach, we compare the four algorithms under different scenarios using the ROC curve. The ROC curve is a well-established graphical plot that displays the accuracy of a binary classifier. It plots the

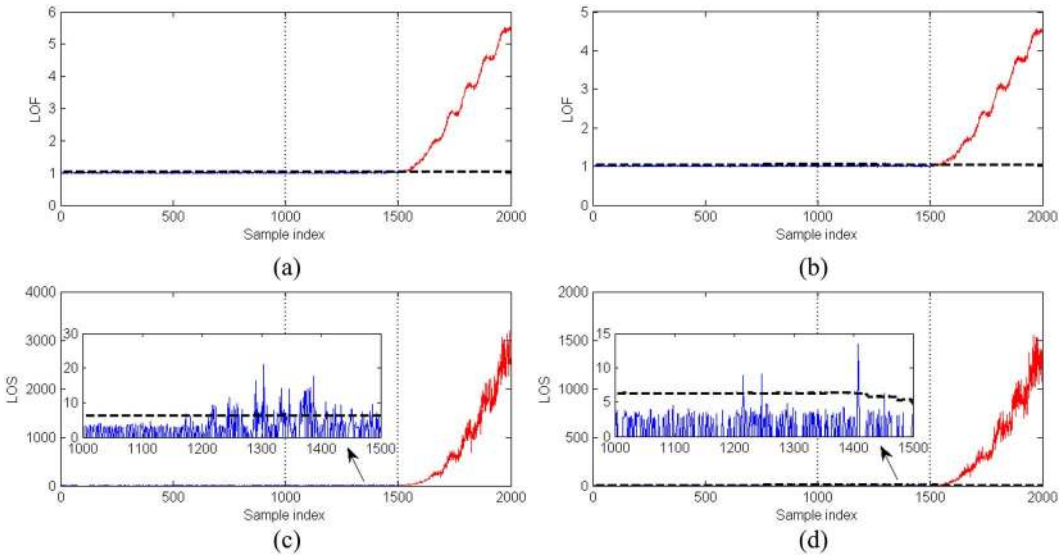

Fig. 9. Fault detection results of (a) primitive LOF, (b) sliding window LOF, (c) primitive ABSAD, and (d) sliding window ABSAD in scenario 4 of the numerical example.

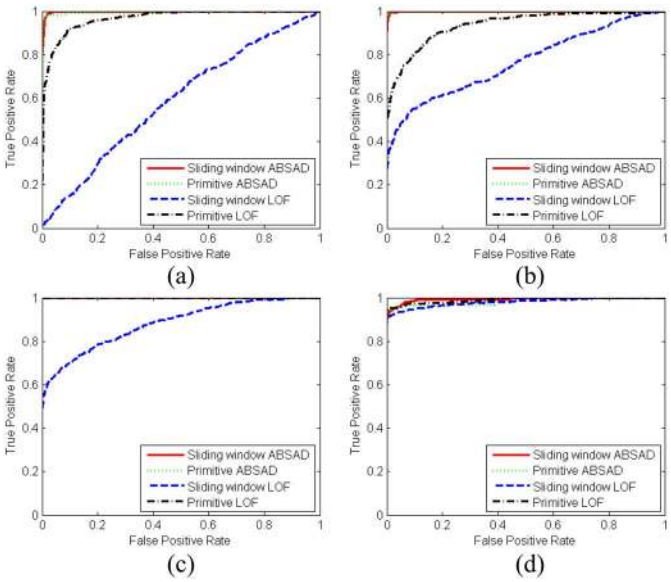

Fig. 10. ROC curve of the four algorithms under different faulty scenarios. (a) Scenario 1. (b) Scenario 2. (c) Scenario 3. (d) Scenario 4.

TPR against the FPR at various threshold settings, i.e., threshold independent. The larger the AUC, the better accuracy a binary classifier can achieve. As expected, the results shown in Fig. 10 again demonstrate the superior accuracy of ABSAD-related algorithms over LOF-related algorithms. Notably, it is difficult to tell the difference between the ROC curve of the sliding window ABSAD and the primitive ABSAD in these plots. We would consider this as another evidence to show the necessity of introducing the updating mechanism to the control limit as suggested in the sliding window ABSAD algorithm.

The presence of measurement noises or disturbances may lead to either false alarms or missed detections. As discussed in [4], a reasonable fault detection algorithm should be sensitive to faults, as well as robust against measurement noises or disturbances. The proposed ABSAD approach defines a threshold using (6) to single out those dimensions in which faulty signals may exist. On the other hand, it filters those dimensions that are less likely to be faulty, and thereby dimension-wisely attenuating the effect of measurement noises and of the disturbances. Moreover, the control limit defined in the sliding window ABSAD algorithm serves as a way to sample-wisely attenuate disturbance and noise signals. By adaptively updating the control limit, the online algorithm provides a better noise and disturbance attenuation ability over nonstationary systems.

The accuracy of those algorithms implemented on full-dimensional spaces, such as LOF, degrades significantly as dimensionality increases. To mitigate the influence exerted by irrelevant dimensions, the ABSAD approach computes the degree of deviation of a data point on its derived subspace projection. As we claimed in Section II, the retained subspace should be meaningful in the sense that it should be able to capture most of the information with regard to the discordance of an object to its adjacent data instances. By examining the retained subspace of the faults in all the four scenarios, we found that the dimensions in the subspace are exactly the same position where the faults were induced on.

Execution speed is another significant performance indicator of an algorithm, especially to those online algorithms dealing with high-speed data streams. Although the sliding window ABSAD algorithm is more computationally demanding than the LOF-based algorithms as shown by our experiments, it is still attractive in the context of high-dimensional data streams based on our computational complexity analysis in Section III-C. As mentioned earlier, trading space for time is one strategy for our online algorithm to speed up the computation and the other one is the use of some indexing structures. Notably, the computation of the ABSAD approach within each step does not need to be sequentially executed. For example, when deriving the reference set of a new sample, the distance from this sample to other samples in the current window profile can be calculated concurrently. This gives us the

possibility of utilizing parallel computing techniques to further improve the real-time performance.

V. Conclusion

The curse of dimensionality may lead to the deterioration of many fault detection techniques. Concept drifting in a data stream may further complicate online fault detection tasks because it requires the algorithm to be adaptive to the time-varying characteristics of the system. To simultaneously address these problems associated with high dimensionality and data streams, this paper proposes an unsupervised, online subspace learning approach to fault detection from non-stationary high-dimensional data streams. In considering the high-dimensional challenges in fault detection tasks, an ABSAD approach is proposed. Aiming to detect faults from data streams with time-varying characteristics, the ABSAD approach is extended to an online mode based on the sliding window strategy.

Based on the analytical study and numerical illustration, we can conclude that the proposed sliding window ABSAD algorithm can simultaneously tackle challenges associated with high dimensionality and data streams in fault detection tasks.

1) In the ABSAD approach, the proposed criterion pairwise cosine for measuring vectorial angles in high-dimensional spaces is a bounded metric and it becomes asymptotically stable as dimensionality increases.
2) The experiments on synthetic datasets indicate that the ABSAD approach has the ability to discriminate low-dimensional subspace faults from normal samples in high-dimensional spaces. Moreover, it outperforms the LOF approach in the context of high-dimensional fault detection.
3) The experiments on synthetic datasets further demonstrate that the sliding window ABSAD algorithm can be adaptive to the time-varying behavior of the monitored system and produce better accuracy than the primitive ABSAD algorithm even when the monitored system has time-varying characteristics.
4) By applying the concept of trading space for time, the sliding window ABSAD algorithm can perform an isochronously online fault detection.

High-dimensional data streams with time-varying characteristics are now emerging in various fields, such as cyber intrusion detection, financial fraud detection, etc. Since the fundamental assumption of this paper applies in many cases within these fields, we can expand the application of the proposed sliding window ABSAD algorithm to other anomaly detection problems.

Appendix

This section presents the proof to (5). It is to derive the expectation of the metric $\mathrm{PCos}(\vec{l}, \vec{\mu}_n(j))$, $j \in N$ and $N = \{1, 2, \ldots, n\}$, that is

$$\mathrm{E}\Big[\mathrm{PCos}\Big(\vec{l}, \vec{\mu}_n(j)\Big)\Big] = \frac{1}{n \cdot (n-1)} \sum_{\substack{j, j^- \in N \\ j^- \neq j}} \frac{\left|l_j^{\#}\right| + \left|l_{j^-}^{\#}\right|}{\sqrt{{l_j^{\#}}^2 + {l_{j^-}^{\#}}^2}}.$$

Proof: It is straightforward to prove the validity of the above equation when $n = 2$. For clarity reason, here in the following we assume $n \geq 3$. Let us substitute $j = 1,\ j = 2, \ldots,\ j = n$ sequentially into the metric PCos, that is

$$\mathrm{PCos}\Big(\vec{l}, \vec{\mu}_n(j)\Big) = \frac{1}{(n-1)} \sum_{j^- \in N \setminus \{j\}} \frac{\left|l_j^{\#}\right|}{\sqrt{{l_j^{\#}}^2 + {l_{j^-}^{\#}}^2}}.$$

We then have the following set of equations:

$$\begin{aligned}
&\mathrm{PCos}\Big(\vec{l}, \vec{\mu}_n(1)\Big) \\
&= \frac{1}{n-1}\left(\frac{|l_1^{\#}|}{\sqrt{{l_1^{\#}}^2 + {l_2^{\#}}^2}} + \frac{|l_1^{\#}|}{\sqrt{{l_1^{\#}}^2 + {l_3^{\#}}^2}} + \cdots + \frac{|l_1^{\#}|}{\sqrt{{l_1^{\#}}^2 + {l_n^{\#}}^2}}\right) \\
&\mathrm{PCos}\Big(\vec{l}, \vec{\mu}_n(2)\Big) \\
&= \frac{1}{n-1}\left(\frac{|l_2^{\#}|}{\sqrt{{l_2^{\#}}^2 + {l_1^{\#}}^2}} + \frac{|l_2^{\#}|}{\sqrt{{l_2^{\#}}^2 + {l_3^{\#}}^2}} + \cdots + \frac{|l_2^{\#}|}{\sqrt{{l_2^{\#}}^2 + {l_n^{\#}}^2}}\right) \\
&\vdots \qquad\qquad\qquad \vdots \qquad\qquad\qquad \vdots \\
&\mathrm{PCos}\Big(\vec{l}, \vec{\mu}_n(n)\Big) \\
&= \frac{1}{n-1}\left(\frac{|l_n^{\#}|}{\sqrt{{l_n^{\#}}^2 + {l_1^{\#}}^2}} + \frac{|l_n^{\#}|}{\sqrt{{l_n^{\#}}^2 + {l_2^{\#}}^2}} + \cdots + \frac{|l_n^{\#}|}{\sqrt{{l_n^{\#}}^2 + {l_{n-1}^{\#}}^2}}\right).
\end{aligned}$$

Summing up both sides of the above equations yields

$$\sum_{j=1}^{n} \mathrm{PCos}\Big(\vec{l}, \vec{\mu}_n(j)\Big) = \frac{1}{n-1} \sum_{\substack{j, j^- \in N \\ j^- \neq j}} \frac{\left|l_j^{\#}\right| + \left|l_{j^-}^{\#}\right|}{\sqrt{{l_j^{\#}}^2 + {l_{j^-}^{\#}}^2}}.$$

Notably, the establishment of the above equation lies in the following fact: for any item $(|l_j^{\#}|/\sqrt{{l_j^{\#}}^2 + {l_{j^-}^{\#}}^2})$ in the right side of the previous set of equations, there exists another corresponding item $(|l_{j^-}^{\#}|/\sqrt{{l_{j^-}^{\#}}^2 + {l_j^{\#}}^2})$ (where $j, j^- \in N$ and $j^- \neq j$) that is additive to the former one. Hence

$$\begin{aligned}
\mathrm{E}\Big[\mathrm{PCos}\Big(\vec{l}, \vec{\mu}_n(j)\Big)\Big] &= \frac{1}{n} \sum_{j=1}^{n} \mathrm{PCos}\Big(\vec{l}, \vec{\mu}_n(j)\Big) \\
&= \frac{1}{n \cdot (n-1)} \sum_{\substack{j, j^- \in N \\ j^- \neq j}} \frac{\left|l_j^{\#}\right| + \left|l_{j^-}^{\#}\right|}{\sqrt{{l_j^{\#}}^2 + {l_{j^-}^{\#}}^2}}.
\end{aligned}$$

The proof is complete. ■

Acknowledgment

The authors would like to thank the editor, the Associate Editors, and the referees for their constructive comments and suggestions that greatly improved the content of this paper.

References

[1] R. Kothamasu, S. H. Huang, and W. H. VerDuin, "System health monitoring and prognostics—A review of current paradigms and practices," in *Handbook of Maintenance Management and Engineering*. London, U.K.: Springer, 2009, pp. 337–362.

[2] S. Zhong, H. Langseth, and T. D. Nielsen, "A classification-based approach to monitoring the safety of dynamic systems," *Rel. Eng. Syst. Safety*, vol. 121, pp. 61–71, Jan. 2014.

[3] X. Dai and Z. Gao, "From model, signal to knowledge: A data-driven perspective of fault detection and diagnosis," *IEEE Trans. Ind. Inform.*, vol. 9, no. 4, pp. 2226–2238, Nov. 2013.

[4] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault-tolerant techniques—Part I: Fault diagnosis with model-based and signal-based approaches," *IEEE Trans. Ind. Electron.*, vol. 62, no. 6, pp. 3757–3767, Jun. 2015.

[5] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault-tolerant techniques—Part II: Fault diagnosis with knowledge-based and hybrid/active approaches," *IEEE Trans. Ind. Electron.*, vol. 62, no. 6, pp. 3768–3774, Jun. 2015.

[6] S. Yin, S. X. Ding, X. Xie, and H. Luo, "A review on basic data-driven approaches for industrial process monitoring," *IEEE Trans. Ind. Electron.*, vol. 61, no. 11, pp. 6418–6428, Nov. 2014.

[7] C. Alippi, D. Liu, D. Zhao, and L. Bu, "Detecting and reacting to changes in sensing units: The active classifier case," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 3, pp. 353–362, Mar. 2014.

[8] Q. Zhao and Z. Xu, "Design of a novel knowledge-based fault detection and isolation scheme," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 2, pp. 1089–1095, Apr. 2004.

[9] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, pp. 78–87, Oct. 2012.

[10] W. Q. Meeker and Y. Hong, "Reliability meets big data: Opportunities and challenges," *Qual. Eng.*, vol. 26, no. 1, pp. 102–116, 2014.

[11] V. T. Sribar, D. Feinberg, N. Gall, A. Lapkin, and M. A. Beyer, *'Big Data' is Only the Beginning of Extreme Information Management*, Gartner, Stamford, CT, USA, 2011.

[12] Y. Zhai, Y.-S. Ong, and I. W. Tsang, "The emerging 'big dimensionality,"' *IEEE Comput. Intell. Mag.*, vol. 9, no. 3, pp. 14–26, Aug. 2014.

[13] M. E. Houle, H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Can shared-neighbor distances defeat the curse of dimensionality?" in *Scientific and Statistical Database Management*. Berlin Heidelberg, Germany: Springer, 2010, pp. 482–500.

[14] A. Zimek, E. Schubert, and H.-P. Kriegel, "A survey on unsupervised outlier detection in high-dimensional numerical data," *Stat. Anal. Data Min. ASA Data Sci. J.*, vol. 5, no. 5, pp. 363–387, 2012.

[15] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is 'nearest neighbor' meaningful?" in *Database Theory—ICDT'99*. Berlin Heidelberg, Germany: Springer, 1999, pp. 217–235.

[16] G.-J. Chen, J. Liang, and J.-X. Qian, "Process monitoring and fault detection based on multivariate statistical projection analysis," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, vol. 3. The Hague, The Netherlands, 2004, pp. 2719–2723.

[17] Z. Ge and Z. Song, "Process monitoring based on independent component analysis-principal component analysis (ICA–PCA) and similarity factors," *Ind. Eng. Chem. Res.*, vol. 46, no. 7, pp. 2054–2063, 2007.

[18] J. Lee, B. Kang, and S.-H. Kang, "Integrating independent component analysis and local outlier factor for plant-wide process monitoring," *J. Process. Control*, vol. 21, no. 7, pp. 1011–1021, Aug. 2011.

[19] Y. Ma, H. Shi, H. Ma, and M. Wang, "Dynamic process monitoring using adaptive local outlier factor," *Chemometr. Intell. Lab. Syst.*, vol. 127, pp. 89–101, Aug. 2013.

[20] C. C. Aggarwal and P. S. Yu, "Outlier detection for high dimensional data," *ACM SIGMOD Rec.*, vol. 30, no. 2, pp. 37–46, Jun. 2001.

[21] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Outlier detection in axis-parallel subspaces of high dimensional data," in *Advances in Knowledge Discovery and Data Mining*. Berlin Heidelberg, Germany: Springer, 2009, pp. 831–838.

[22] B. Krawczyk, J. Stefanowski, and M. Wozniak, "Data stream classification and big data analytics," *Neurocomputing*, vol. 150, pp. 238–239, May 2013.

[23] F. Olken and L. Gruenwald, "Data stream management: Aggregation, classification, modeling, and operator placement," *IEEE Internet Comput.*, vol. 12, no. 6, pp. 9–12, Nov. 2008.

[24] X. Zhu, P. Zhang, X. Lin, and Y. Shi, "Active learning from stream data using optimal weight classifier ensemble," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 6, pp. 1607–1621, Dec. 2010.

[25] J. Gao, B. Ding, W. Fan, J. Han, and P. S. Yu, "Classifying data streams with skewed class distributions and concept drifts," *IEEE Internet Comput.*, vol. 12, no. 6, pp. 37–49, Nov./Dec. 2008.

[26] G. A. Cherry and S. J. Qin, "Multiblock principal component analysis based on a combined index for semiconductor fault detection and diagnosis," *IEEE Trans. Semicond. Manuf.*, vol. 19, no. 2, pp. 159–172, May 2006.

[27] H. Chen, G. Jiang, C. Ungureanu, and K. Yoshihira, "Online tracking of component interactions for failure detection and localization in distributed systems," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 37, no. 4, pp. 644–651, Jul. 2007.

[28] J.-C. Jeng, "Adaptive process monitoring using efficient recursive PCA and moving window PCA algorithms," *J. Taiwan Inst. Chem. Eng.*, vol. 41, no. 4, pp. 475–481, 2010.

[29] A. Alzghoul and M. Löfstrand, "Increasing availability of industrial systems through data stream mining," *Comput. Ind. Eng.*, vol. 60, no. 2, pp. 195–205, 2011.

[30] L. Zhang, J. Lin, and R. Karim, "An angle-based subspace anomaly detection approach to high-dimensional data: With an application to industrial fault detection," *Rel. Eng. Syst. Safety*, vol. 142, pp. 482–497, Oct. 2015.

[31] L. Ertöz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *Proc. SDM*, San Francisco, CA, USA, 2003, pp. 47–58.

[32] L. O. Jimenez and D. A. Landgrebe, "Supervised classification in high-dimensional space: Geometrical, statistical, and asymptotical properties of multivariate data," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 28, no. 1, pp. 39–54, Feb. 1998.

[33] T. Cai, J. Fan, and T. Jiang, "Distributions of angles in random packing on spheres," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 1837–1864, 2013.

[34] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 93–104, Jun. 2000.

Liangwei Zhang received the M.S. degree in management science and engineering from the Nanjing University of Science and Technology, Nanjing, China, in 2009. He is currently pursuing the Ph.D. degree in operation and maintenance engineering with the Luleå University of Technology, Luleå, Sweden.

From 2009 to 2013, he was a Consultant of Reliability Engineering with SKF, Beijing, China. His current research interests include machine learning, fault detection, eMaintenance, and big data analytics.

Jing Lin (M'15) received the Ph.D. degree in management science and engineering from the Nanjing University of Science and Technology, Nanjing, China, in 2008.

She is an Associate Professor with the Division of Operation and Maintenance Engineering, Luleå University of Technology, Luleå, Sweden. She has authored above 60 peer-reviewed journal and conference papers, and one monograph in related topics. Her current research interests include reliability and maintenance engineering, big data analytics, eMaintenance, and asset management.

Ramin Karim received the Ph.D. degree in operation and maintenance engineering from the Luleå University of Technology, Luleå, Sweden, in 2008.

He researched in the area of information and communications technology for over 20 years, as an Architect, a Project Manager, a Software Designer, a Product Owner, and a Developer. He is currently a Professor, responsible for the research area of eMaintenance, with the Division of Operation and Maintenance Engineering, Luleå University of Technology. He has published over 100 refereed journal and conference papers. His current research interests include robotics, feedback control systems, and control theory.