An Incremental Clustering-Based Fault Detection Algorithm for Class-Imbalanced Process Data

Jueun Kwak, Taehyung Lee, and Chang Ouk Kim

***Abstract*—Training fault detection model requires advanced data-mining algorithms when the growth rate of the process data is notably high and normal-class data overwhelm fault-class data in number. Most standard classification algorithms, such as support vector machines (SVMs), can handle moderate sizes of training data and assume balanced class distributions. When the class sizes are highly imbalanced, the standard algorithms tend to strongly favor the majority class and provide a notably low detection of the minority class as a result. In this paper, we propose an online fault detection algorithm based on incremental clustering. The algorithm accurately finds wafer faults even in severe class distribution skews and efficiently processes massive sensor data in terms of reductions in the required storage. We tested our algorithm on illustrative examples and an industrial example. The algorithm performed well with the illustrative examples that included imbalanced class distributions of Gaussian and non-Gaussian types and process drifts. In the industrial example, which simulated real data from a plasma etcher, we verified that the performance of the algorithm was better than that of the standard SVM, one-class SVM and three instance-based fault detection algorithms that are typically used in the literature.**

***Index Terms*—Fault detection, class imbalance data, data mining, incremental clustering, process drift.**

I. Introduction

SEMICONDUCTOR process faults are responsible for a large proportion of wafer defects. This paper deals with the detection of process faults at equipment level. Training fault detection model requires advanced data-mining algorithms when the growth rate of the process sensor data is high and normal-class data overwhelm fault-class data in number. In this paper, we propose an online fault detection algorithm based on incremental clustering. The algorithm accurately finds wafer faults even in severe class distribution skews and efficiently processes massive sensor data in terms of reductions in the required storage.

Most manufacturers prevent the occurrence of faults at the risk of high false-alarm rates. As a result, considerably less fault data are readily available compared to the amount of normal data available. A dataset is imbalanced if the classes are not approximately equally represented. Fault data may comprise 20-30% of all data if the equipment is in an unhealthy state, whereas this percentage decreases to 10% or even 1% if the equipment is under ideal operation conditions [2]. In these cases, the fault data represent only a small portion of ill process conditions, which prevents data-mining algorithms from providing generalized knowledge over the entire fault data space. In particular, when the class sizes are highly imbalanced, conventional classification algorithms tend to strongly favor the majority class and detect the minority class at extremely low rates [1]. Machine learning using such data sets is a relatively new research topic that must be further investigated in the research community.

In general, semiconductor process data are distributed in the feature space such that the normal data form clouds and fault data are located sporadically around the normal clouds. Traditional Schewhart-type control charts, such as the Hotelling's T^2 and squared-prediction-error (SPE) charts [7], [14] do not require fault data to determine a boundary to separate the normal operation area and faulty operation area. However, this type of control chart is based on the Gaussian assumption that the normal data are distributed in the feature space according to a multivariate Gaussian function. Recently, residual-based online fault detection approaches have been proposed [27], [28]. In these studies, data-driven system identification models were formulated, and fault detection was performed by locating abnormalities in the residual signals. The residual-based approaches are potentially applicable in classification problems without fault data.

From the data-mining perspective, the fault detection problem entails learning a binary classifier that outputs two class labels: normal and fault. The classification algorithms are either parametric or non-parametric. Parametric models assume an underlying functional form of the classifier and have some fit parameters. Non-parametric models have no explicit assumption about the form of the classifier. The support vector machine (SVM) is one of the most popular and promising parametric algorithms [4]; the SVM finds the separating hyper-plane in the feature space that can create maximum distance between the plane and the nearest data of different classes. The SVM has been proven to exhibit robust performance in process fault detection [11], [32]. The standard SVM algorithm has been extended to accommodate process drift [10], [13]. However, when an SVM is trained over high degrees of features with limited fault samples, the learning may progress toward high accuracy for massive normal datasets but low accuracy for small fault datasets. The one-class SVM is a variant of the standard SVM [23] that only

Manuscript received July 24, 2014; revised January 31, 2015 and May 3, 2015; accepted June 2, 2015. Date of publication June 15, 2015; date of current version July 31, 2015. This work was supported by the Technology Innovation Program funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea) under Grant 10045913.

The authors are with the Department of Information and Industrial Engineering, Yonsei University, Seoul 120-749, Korea (e-mail: jueuny7@yonsei.ac.kr; interact@yonsei.ac.kr; kimco@yonsei.ac.kr).

Digital Object Identifier 10.1109/TSM.2015.2445380

0894-6507 © 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

requires the normal operating regime data points to train the model, whereas the standard SVM requires samples from both the normal and faulty classes involved to build a classifier. In the one-class SVM, there is an inherent assumption that the origin in the feature space belongs to the negative or faulty class. Hence, the objective is to maximize the separation between the origin and the cluster of normal samples in the feature space. The one-class time-adaptive SVM [13] has been recently presented; this SVM can detect abrupt process changes with normal-class training data. In addition, the incremental one-class SVM [3] has been developed for cognitive fault diagnosis using a sliding window concept.

Recent research on process fault detection has focused on non-parametric approaches. The k-nearest-neighbor (kNN) approach is a simple type of instance-based learning in which the classification function performs locally in the feature space. To classify an unlabeled sample, the distance of the wafer to all labeled samples in the training set is computed, its k nearest samples are identified, and the class labels of the nearest neighbors are used to determine the class label of the unlabeled sample. The fault detection kNN (FD-kNN) [16] is a variant of the standard kNN rule. To classify a new sample, FD-kNN calculates the sum of the squared Euclidean distances of the k-nearest normal neighbors and compares the total distance with a threshold, which is determined using a non-central chi-squared distribution. Similarly, the adaptive Mahalanobis distance-based kNN (MD-kNN) [30] applies the modified k-nearest-neighbor rule for fault detection, but instead of the standard Euclidean distance, MD-kNN searches k neighbors of a new sample considering the density of the sample distribution using the Mahalanobis distance, which is calculated from the local covariance structure of the new sample. Both approaches are advantageous in that no fault data are required to perform classifications and they can adapt to process drifts. PC-kNN [17] is an extension of FD-kNN that performs fault detection in a reduced feature space, which is created via principal component analysis (PCA). In addition, in a recent study [19], high-dimensional features with a nonlinear structure were transformed into low-dimensional manifold features using a diffusion map, and the kNN rule was used to predict the potential faults. Two studies [17], [19] sought to accelerate the search for k neighbors and save data storage by reducing the number of features. In all kNN approaches, the classification accuracy becomes high when the sample space contains a sufficiently large number of labeled data. However, searching for the k nearest neighbors in a massive dataset requires a long computational time and a large storage space.

In this paper, we propose an incremental clustering-based fault detection algorithm that accurately finds wafer faults even in severe class distribution skews. The algorithm does not maintain all wafer data collected by the equipment monitoring system. Instead, it clusters normal data (i.e., majority-class data) to reduce the storage and computational requirements. For each cluster, the statistical summaries (called prototypes in this study) are only maintained to detect potential faulty wafers. The class label of a new wafer is predicted in a multi-dimensional feature space using the Mahalanobis distance between the new wafer and the center (mean) of the closest cluster. The Mahalanobis distance is a statistical distance measure that considers the correlations and differences among the data points. On the other hand, the Euclidean distance is a geometric distance measure that only considers the difference among the data points. When a new wafer is normal, the prototype of the closest cluster is updated with the new wafer data using recursive formulas. Because faults are rare, only normal wafer data are clustered in this study. If the classified result is faulty but the actual class is normal, the algorithm creates a new single-member cluster whose center is the sample. A merge operation starts when the clusters overlap in the feature space. Using illustrative examples, we verified that the performance of the proposed algorithm was excellent for imbalanced datasets that were generated from non-Gaussian distributions. In addition, the incremental learning could quickly recognize process drifts and shift the normal wafer regions accordingly. Finally, we tested our algorithm on an industrial example that simulated real data from a plasma etcher.

The remainder of this paper is organized as follows. Section II presents the proposed algorithm, and Section III introduces the illustrative examples. Section IV compares the performance of the algorithm with that of the standard SVM, one-class SVM, FD-kNN, PC-kNN, and MD-kNN algorithms. Section V concludes this research.

II. Proposed Algorithm

A. Notations

Let d be the number of features. The feature space is a multi-dimensional Euclidean space constructed with process variables. All features are assumed to be normalized into the hypercube $[0, 1]^d$. A wafer data point is represented by a vector $\boldsymbol{x}$ in the feature space. A cluster is a collection of similar data, and a cluster representative is called a prototype p. A prototype contains an estimated mean vector ($\boldsymbol{m}$), the data size (n), and the inverse of an estimated covariance matrix ($\hat{\boldsymbol{\Sigma}}^{-1}$). The mean vector determines the location of the cluster; the covariance matrix specifies the shape and size of the cluster. The inverted covariance matrix is required to efficiently calculate the Mahalanobis distance (MD).

B. Procedure Overview

The flow diagram of the proposed algorithm is described in Fig. 1, where the algorithm involves the following four phases:

1) Phase 0 (Initialization): The algorithm assumes that there is a training sample $\boldsymbol{x}$ in the feature space. Then, the algorithm begins the fault detection task for a single-member cluster p_0 with $n_{p_0} = 1$, $\boldsymbol{m}_{p_0} = \boldsymbol{x}$, and $\hat{\boldsymbol{\Sigma}}_{p_0}^{-1} = [1/t_{ij}]$, $(i, j = 1, \ldots, d)$, where $t_{ij} =$ fraction if $i = j$, and $1/t_{ij} = 0$ otherwise. Here, fraction denotes a small fraction of the range [0, 1], i.e., we restrict the range of the single-member cluster to a small area in the feature space.

2) Phase 1 (Classification): When a new sample is collected, its class label is assigned using the distance between the new sample and the center (mean) of the closest normal cluster.

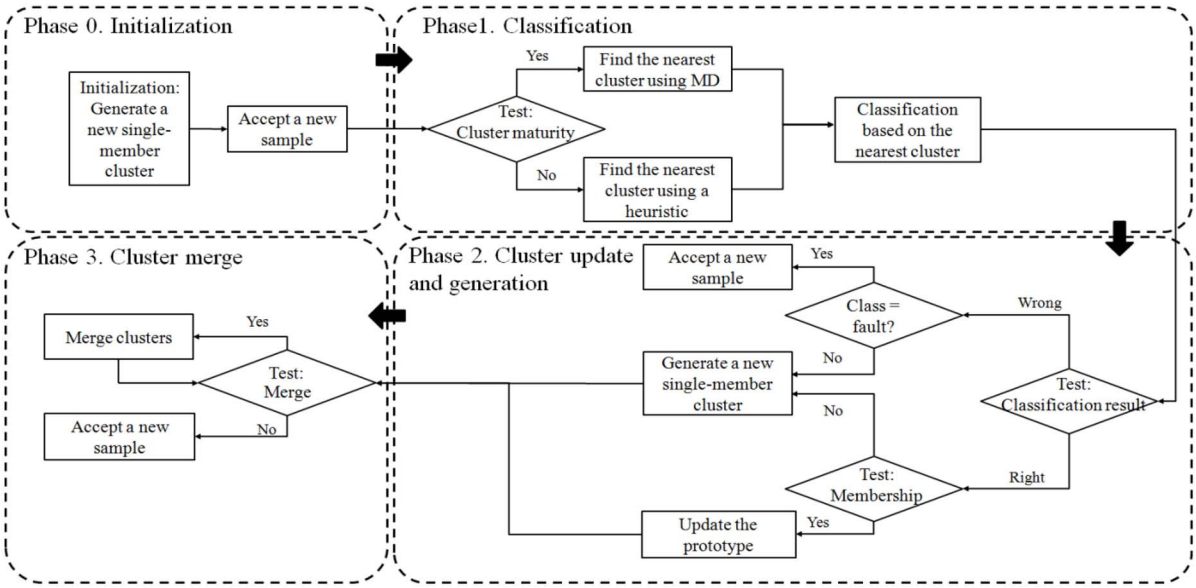

Fig. 1. Flow diagram of the proposed algorithm.

The distance is calculated using the Mahalanobis metric. If the calculated distance is less than a threshold, the new sample is considered a normal wafer; otherwise, it is considered a fault. We impose the threshold using the probability distribution of the squared distance.

3) Phase 2 (Cluster Update and Generation): If the new sample is normal, the prototype of the closest cluster is updated with consideration of the new sample. As a result, the cluster grows incrementally. Because faults are rare in manufacturing data, only normal samples are clustered in this study. If the classification considers the cluster faulty but the actual class is normal, the algorithm creates a new single-member cluster whose center point is the sample.

4) Phase 3 (Cluster Merge): Overlapped clusters in the feature space are sequentially merged as long as a distance-based merge condition is satisfied. This operation creates large clusters, which reduces the computation time to find the nearest cluster in the classification phase.

C. Phase 1: Classification

When the algorithm accepts a new sample $\boldsymbol{x}$, the class label of the instance is assigned according to the following classification steps.

(Step 1) Find the winner cluster p that is the nearest to $\boldsymbol{x}$ using the squared Mahalanobis distance.

$$MD_p(\boldsymbol{x})^2 = \left(\boldsymbol{x} - \boldsymbol{m}_p\right)^T \hat{\boldsymbol{\Sigma}}_p^{-1} \left(\boldsymbol{x} - \boldsymbol{m}_p\right). \tag{1}$$

The Mahalanobis distance is an effective metric for multivariate problems with interaction effects among large numbers of variables because it can consider the correlation of the dataset using the covariance matrix [6].

(Step 2) Test the membership of the new sample $\boldsymbol{x}$ using the distribution of the squared Mahalanobis distance in (1). If the sample $\boldsymbol{x}$ is sufficiently close to the winner cluster p, the algorithm classifies it as normal; otherwise, the algorithm declares it as faulty. Under the Gaussian assumption, the squared Mahalanobis distance follows a chi-squared distribution with d degrees of freedom. However, it is related to a Fisher distribution with d and $(n_p - d)$ degrees of freedom if the mean vector and the covariance matrix of cluster p are estimated from the existing data [31].

$$\frac{n_p\left(n_p - d\right)}{d\left(n_p^2 - 1\right)} \left(MD_p\right)^2 \sim F\left(d, n_p - d\right). \tag{2}$$

Suppose the new sample $\boldsymbol{x}$ belongs to cluster p. Then, it is possible to find the probability of the squared distance from the center of cluster p using (2). A small probability implies that $\boldsymbol{x}$ has a small likelihood of membership in cluster p. Under the significance level α, the corresponding threshold θ_p is given as

$$\theta_p = \frac{d\left(n_p^2 - 1\right)}{n_p\left(n_p - d\right)} F_\alpha\left(d, n_p - d\right). \tag{3}$$

If the squared distance does not exceed the threshold in (3), the new sample $\boldsymbol{x}$ is considered a member of cluster p and the class label of the cluster is assigned to the sample. The significance level α controls the tradeoff between stability (update of an old cluster) and flexibility (creation of a new cluster). An inappropriate setting will be partially compensated by the merging option, as explained below.

Remark 1: When the number of members, n_p, in the winner cluster does not exceed the number of features d, i.e., $(n_p - d)$ is non-positive, the Fisher-like distribution cannot be defined. The creation of the un-matured cluster cannot be avoided at the early stage of model training. When $n_p \leq d$, we suggest another threshold formula defined in (4), which is based on a heuristic method [20]

$$\theta_p = (d \ / \ fraction) \cdot r, \tag{4}$$

where d represents the maximum squared Euclidean distance in the feature space $[0, 1]^d$, *(d / fraction)* is the maximum

squared Mahalanobis distance in the feature space when the distance is calculated with the inverse covariance matrix of a single-member cluster, $\hat{\boldsymbol{\Sigma}}_{p_0}^{-1} = [1/t_{ij}]$, where t_{ij} = fraction, and $r(r < 1)$ is a threshold parameter that adjusts the squared distance boundary of un-matured clusters.

D. Phase 2: Prototype Update and Generation

(Step 1) If the new sample $\boldsymbol{x}$ is correctly classified and its true class is normal, then we update the mean vector, covariance matrix and data size of the corresponding cluster as follows. First, the data size is increased by one, and the mean vector is updated using a recursive average method.

$$n_p^{new} = n_p^{old} + 1, \tag{5}$$

$$\boldsymbol{m}_p^{new} = \boldsymbol{m}_p^{old} + \frac{1}{n_p^{new}}\left(\boldsymbol{x} - \boldsymbol{m}_p^{old}\right). \tag{6}$$

Second, the algorithm updates the covariance matrix. By using the rank-one modification method [24], the sample covariance matrix $\boldsymbol{S}_p$ can be recursively estimated as

$$\boldsymbol{S}_p^{new} = \frac{n_p^{old} - 1}{n_p^{old}}\boldsymbol{S}_p^{old} + \frac{1}{n_p^{old}}\left(\boldsymbol{x} - \boldsymbol{m}_p^{new}\right)\left(\boldsymbol{x} - \boldsymbol{m}_p^{new}\right)^T. \tag{7}$$

For calculating the Mahalanobis distance, the inversion of the sample covariance matrix in (7) is required. However, the matrix inversion is slow and inherently often ill-posed. To avoid this problem, we apply the recursive update formula for the matrix inversion in (9) [5], [22], which can be derived using the *Neumann series* (refer to Appendix A for the derivation):

$$\begin{aligned}
&\left(\boldsymbol{S}_p^{new}\right)^{-1} \\
&= \left[\frac{n_p^{old} - 1}{n_p^{old}}\boldsymbol{S}_p^{old} + \frac{1}{n_p^{old}}\left(\boldsymbol{x} - \boldsymbol{m}_p^{new}\right)\left(\boldsymbol{x} - \boldsymbol{m}_p^{new}\right)^T\right]^{-1}, \quad (8)\\
&= \frac{n_p^{old}}{n_p^{old} - 1}\boldsymbol{S}_p^{old-1} \\
&\quad - \frac{n_p^{old}}{n_p^{old} - 1}\frac{\left(\boldsymbol{S}_p^{old-1}\left(\boldsymbol{x} - \boldsymbol{m}_p^{new}\right)\right)\left(\boldsymbol{S}_p^{old-1}\left(\boldsymbol{x} - \boldsymbol{m}_p^{new}\right)\right)^T}{n_p^{old} + \left\{\left(\boldsymbol{x} - \boldsymbol{m}_p^{new}\right)^T\boldsymbol{S}_p^{old-1}\left(\boldsymbol{x} - \boldsymbol{m}_p^{new}\right) - 1\right\}}.
\end{aligned} \tag{9}$$

Note that the inversion formula in (9) cannot be applied when $n_p^{old} = 1$ (i.e., a single-member cluster). For single-member clusters, the inverse matrix is directly obtained from $\boldsymbol{S}_p^{new}$. When the amount of data in a cluster is large, the sample covariance matrix produces an unbiased estimate for the correlation of members in the cluster. However, the estimator is not robust when the cluster has a small amount of data. In particular, a distorted estimation result may be obtained when the amount of data is less than the feature dimension. The shrinkage method [25] is an efficient statistical technique for estimating a large-scale covariance matrix. The underlying idea lies in the weighted average of the sample covariance matrix $\boldsymbol{S}_p^{new}$ and target matrix $\boldsymbol{T}_p$ where $\boldsymbol{T}_p$ is the diagonal matrix of $\boldsymbol{S}_p^{new}$:

$$\hat{\boldsymbol{\Sigma}}_p = \lambda\boldsymbol{T}_p + (1 - \lambda)\boldsymbol{S}_p^{new}, \tag{10}$$

where

$$\lambda = \frac{1}{n_p^{new}}. \tag{11}$$

The shrinkage intensity $\lambda \in [0, 1]$ in (11) approaches zero as $n_p \to \infty$. We modify the shrinkage correction to incrementally estimate the inverse covariance matrix as follows (refer to Appendix B for the derivation):

$$\begin{aligned}
\left(\hat{\boldsymbol{\Sigma}}_p\right)^{-1} &= \frac{1}{1 - \lambda}\left(\boldsymbol{S}_p^{new}\right)^{-1} - \frac{1}{(1 - \lambda)^2}\left(\boldsymbol{S}_p^{new}\right)^{-1} \\
&\quad \times\left[\frac{1}{1 - \lambda}\left(\boldsymbol{S}_p^{new}\right)^{-1} + \frac{1}{\lambda}\left(\boldsymbol{T}_p\right)^{-1}\right]\left(\boldsymbol{S}_p^{new}\right)^{-1}.
\end{aligned} \tag{12}$$

Using (9)-(12), the inverse covariance matrix can be efficiently updated when a new sample is obtained.

(Step 2) If the new sample $\boldsymbol{x}$ is misclassified as normal, the algorithm returns to the classification phase and waits for the next sample. However, if the new sample is misclassified as faulty, the algorithm generates a new single-member cluster, p_0, and sets the inverse of the initial covariance matrix to $\hat{\boldsymbol{\Sigma}}_{p_0}^{-1} = [1/t_{ij}]$, $(i, j = 1, \ldots, d)$, where t_{ij} = fraction if $i = j$, and $1/t_{ij} = 0$ otherwise.

E. Phase 3: Merge

The generation of many single-member clusters may result in a swarm of clusters, which increases the computational burden required to find the nearest cluster (similar to the standard kNN-based learning). The merge phase maintains a small number of clusters by repeating the merge of two adjacent clusters until a merge condition is satisfied. The algorithm starts the merge with a cluster that is either updated or newly created in the update phase. Let the output cluster from the update phase be the target cluster p_m. The merge phase involves the following steps.

(Step 1) The algorithm finds the nearest cluster p' of the target cluster p_m using the squared Mahalanobis distance, which is calculated using the covariance matrix of p_m. If the squared distance between p' and p_m is greater than the threshold θ_{p_m}, the merge is terminated and the algorithm returns to the classification phase. The formula in (3) is used for the threshold θ_{p_m} when the number of members in p_m exceeds the feature dimension d. Otherwise, the threshold in (4) is used.

(Step 2) Clusters p' and p_m are combined into a new cluster p''. The number of members in the new cluster p'' is $n_{p''} = n_{p_m} + n_{p'}$, and the mean vector $\boldsymbol{m}_{p''}$ is defined as in (13). The estimated covariance matrix $\hat{\boldsymbol{\Sigma}}_{p''}$ is updated as in (14).

$$\boldsymbol{m}_{p''} = \frac{n_{p_m}}{n_{p_m} + n_{p'}}\boldsymbol{m}_{p_m} + \frac{n_{p'}^{'}}{n_{p_m} + n_{p'}}\boldsymbol{m}_{p'}, \tag{13}$$

$$\hat{\boldsymbol{\Sigma}}_{p''} = \frac{n_{p_m}}{n_{p_m}+n_{p'}}\hat{\boldsymbol{\Sigma}}_{p_m} + \frac{n_{p'}}{n_{p_m}+n_{p'}}\hat{\boldsymbol{\Sigma}}_{p'} + \frac{n_{p_m}n_{p'}}{\left(n_{p_m}+n_{p'}\right)^2}\left(\boldsymbol{m}_{p_m}-\boldsymbol{m}_{p'}\right)\left(\boldsymbol{m}_{p_m}-\boldsymbol{m}_{p'}\right)^T. \quad (14)$$

Analyzing cluster shapes with covariance matrices is advantageous because spatial relationships can be encoded in the covariance matrices. When there are overlapped or adjacent clusters, it would be beneficial to merge them and replace them with an elongated cluster. The merging of two covariance matrices is conducted to specify the shape and size of the combined cluster.

Let us denote $\left(\frac{n_{p_m}}{n_{p_m}+n_{p'}}\hat{\boldsymbol{\Sigma}}_{p_m} + \frac{n_{p'}}{n_{p_m}+n_{p'}}\hat{\boldsymbol{\Sigma}}_{p'}\right)$ in (14) as $\boldsymbol{\Delta}$. Then, the inversion of the merged cluster's covariance matrix can be efficiently calculated as shown below in (15) and (16) because the inverse matrices of $\hat{\boldsymbol{\Sigma}}_{p_m}$ and $\hat{\boldsymbol{\Sigma}}_{p'}$ are maintained in the clusters p_m and p', respectively (the derivation of $\boldsymbol{\Delta}^{-1}$ is described in Appendix B):

$$\left(\hat{\boldsymbol{\Sigma}}_{p''}\right)^{-1} = \boldsymbol{\Delta}^{-1} - n_{p_m}n_{p'} \times \frac{\left(\boldsymbol{\Delta}^{-1}\left(\boldsymbol{m}_{p_m}-\boldsymbol{m}_{p'}\right)\right)\left(\boldsymbol{\Delta}^{-1}\left(\boldsymbol{m}_{p_m}-\boldsymbol{m}_{p'}\right)\right)^T}{\left(n_{p_m}+n_{p'}\right)^2 + n_{p_m}n_{p'}\left(\boldsymbol{m}_{p_m}-\boldsymbol{m}_{p'}\right)^T\boldsymbol{\Delta}^{-1}\left(\boldsymbol{m}_{p_m}-\boldsymbol{m}_{p'}\right)}, \quad (15)$$

$$\boldsymbol{\Delta}^{-1} = \frac{n_{p_m}+n_{p'}}{n_{p_m}}\hat{\boldsymbol{\Sigma}}_{p_m}^{-1} - \left(\frac{n_{p_m}+n_{p'}}{n_{p_m}}\right)^2\hat{\boldsymbol{\Sigma}}_{p_m}^{-1} \times \left(\frac{n_{p_m}+n_{p'}}{n_{p_m}}\hat{\boldsymbol{\Sigma}}_{p_m}^{-1} + \frac{n_{p_m}+n_{p'}}{n_{p'}}\hat{\boldsymbol{\Sigma}}_{p'}^{-1}\right)^{-1}\hat{\boldsymbol{\Sigma}}_{p_m}^{-1}. \quad (16)$$

(Step 3) The algorithm deletes the clusters p' and pm that are combined in the merge operations. Let p' and $p_m \leftarrow p''$ and repeat steps (1) and (2).

There is a volume-based merge condition that combines two clusters when they touch each other in the feature space and are similar in shape and orientation [21]. Our merge condition combines two clusters when their centers are sufficiently close such that the Mahalanobis distance between the centers is less than the membership threshold, i.e., a merge operation is executed when the clusters exhibit substantial overlap. Throughout the merge phase, the storage requirement for finding the nearest neighbor can be considerably reduced.

Remark 2: Classification based on the Mahalanobis distance is known to exhibit high-level performance when correlated features form ellipsoidal data regions, such as the multivariate Gaussian distribution. However, the Mahalanobis distance is appropriate when the covariance matrix is calculated using only the local data points near the new samples [30]. The proposed algorithm forms small ellipsoidal clusters to approximate the global data distributions of non-ellipsoidal shapes.

III. Illustrative Examples

In this section, we introduce four illustrative examples to demonstrate the ability of the proposed algorithm to detect faults in different process scenarios. Hereafter, we will refer to the proposed algorithm as the incremental clustering-based fault detection method (IC-FDM). The IC-FDM requires four parameters, i.e., the range parameter *fraction* that initializes the covariance matrix of single-member cluster, the significant level α that controls the tradeoff between the stability and flexibility of cluster, the threshold parameter r that handles the early cluster's volume size, and the smoothing parameter β that is a weight between 0 and 1 given to the most recent sample in the update of the mean vector of cluster. In this study, we fix *fraction* to 1/400 and r to 1/80 in all of the examples because these parameters are used at the early stage of the clustering process. In all of the examples, we found that the performance of the IC-FDM is not sensitive to the two parameters. For process drift, the mean vector of the cluster is updated using an exponentially weighted moving average formula, $\boldsymbol{m}_p^{new} = \boldsymbol{m}_p^{old} + \beta(\boldsymbol{x} - \boldsymbol{m}_p^{old})$, that is dependent on the smoothing parameter $\beta(\beta > 0)$. This smoothing parameter controls the weight of recent samples to calculate the center of a moving cluster. We conducted a sensitivity analysis for different combinations of α and β.

A. Performance Evaluation Measures

Conventionally, the total accuracy rate is used to evaluate the performance of fault detection models. However, it is not a proper measure for datasets with unequal class sizes: if the normal wafers are perfectly classified, the total accuracy will be high even though the faults are largely misclassified. Therefore, in this paper, we use the G-mean [15] to evaluate the performance of the IC-FDM. The G-mean is known as a more suitable measure than the total accuracy, particularly for classifier learning with imbalanced class distributions.

Let the majority class (normal wafer) be a negative class and the minority class (faulty wafer) be a positive class. There are four combinations of true classes and predicted results: true positive (TP), false negative (FN), false positive (FP), and true negative (TN). FP and FN are also called type I error and type II error, respectively. Define the FP rate (FPR) and the FN rate (FNR) as

$$\text{FPR} = \frac{\text{the number of misclassified as faulty wafer}}{\text{total number of normal wafers}} = \frac{\text{FP}}{\text{FP}+\text{TN}}, \quad (17)$$

$$\text{FNR} = \frac{\text{the number of misclassified as normal wafer}}{\text{total number of faulty wafers}} = \frac{\text{FN}}{\text{TP}+\text{FN}}. \quad (18)$$

Then, the G-mean is

$$\text{G}-\text{mean} = \sqrt{(1-\text{FPR})\times(1-\text{FNR})}. \quad (19)$$

The G-mean produces a score between zero and one, where one represents a perfect prediction and zero represents a random prediction. Unlike total accuracy, the G-mean for the classification results in which normal wafers are perfectly classified and faults are largely misclassified is close to zero because the corresponding FNR is extremely high.

TABLE I
CLASSIFICATION RESULTS OF THE ILLUSTRATIVE EXAMPLES

Parameter		Process scenario			
α	β	Gaussian	Frank-Copula	t-Copula	Clayton-Copula
0.01	0	0.925 (5)	0.497 (1)	0.770 (3)	0.797 (3)
	0.005	0.924 (6)	0.519 (1)	0.776 (3)	0.797 (2)
	0.01	0.931 (6)	0.561 (1)	0.782 (3)	0.812 (2)
0.05	0	0.896 (12)	0.867 (3)	0.950 (5)	0.906 (4)
	0.005	0.896 (12)	0.847 (3)	0.950 (5)	0.906 (4)
	0.01	0.888 (11)	0.873 (3)	0.918 (6)	0.906 (4)
0.1	0	0.879 (17)	0.952 (9)	0.954 (11)	0.965 (13)
	0.005	0.879 (17)	0.952 (9)	0.954 (11)	0.965 (13)
	0.01	0.879 (17)	0.951 (9)	0.954 (11)	0.965 (13)
0.2	0	0.865 (23)	0.921 (22)	0.881 (21)	0.952 (17)
	0.005	0.865 (23)	0.921 (22)	0.881 (21)	0.952 (17)
	0.01	0.865 (23)	0.921 (22)	0.881 (21)	0.952 (17)
0.3	0	0.831 (24)	0.932 (23)	0.878 (20)	0.935 (18)
	0.005	0.831 (24)	0.932 (23)	0.878 (20)	0.935 (18)
	0.01	0.830 (24)	0.932 (23)	0.878 (20)	0.935 (18)

* The values in parentheses refer to the numbers of clusters created.

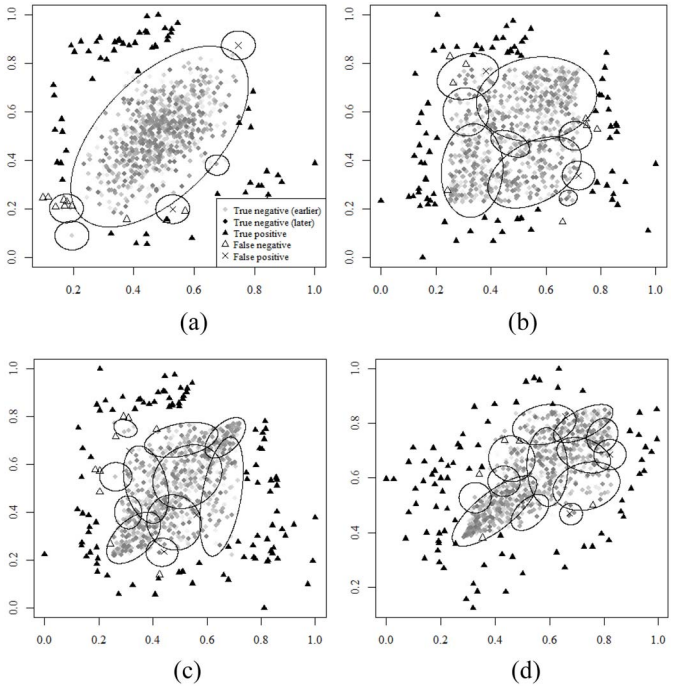

Fig. 2. Clustering results. (a) Gaussian ($\alpha = 0.01, \beta = 0.01$, G-mean = 0.931). (b) Frank-Copula ($\alpha = 0.1, \beta = 0$, G-mean = 0.952). (c) t-Copula ($\alpha = 0.1, \beta = 0$, G-mean = 0.954). (d) Clayton-Copula ($\alpha = 0.1, \beta = 0$, G-mean = 0.965).

TABLE II
CLASSIFICATION RESULTS FOR DRIFT CASES

	FPR	FNR	G-mean	Accuracy
Slow drift (3σ)	0.006	0.059	0.967(5)	0.985
Fast drift (10σ)	0.009	0.057	0.967(4)	0.983

B. Process Scenarios

In semiconductor manufacturing, previous studies have shown that wafers processed in normal equipment conditions are located near each other in the feature space [16]. However, the sample distribution might not follow an elliptical shape. Faults are distributed near the boundaries of normal wafers. Based on observations, we consider four illustrative examples that address two input features. In the first scenario, normal-class samples are generated using a bivariate Gaussian distribution. The remaining three scenarios use non-elliptical distributions: a t-Copula distribution, a Frank-Copula distribution, and a Clayton-Copula distribution [26]. We used the nacopula packages programmed in R to generate the three Copula distributions [18]. The total number of samples generated by each distribution is 1,000. The percentage of fault samples is set to approximately 20% of the total number of samples. The IC-FDM uses half of the samples to establish the cluster prototypes and the remaining half to test its performance.

We also test the performance of the IC-FDM in two process drift scenarios. Process drift is the change of the normal process behavior caused by the aging of equipment over time. We create two scenarios as follows. In the training stage, 500 wafer samples are generated according to a bivariate Gaussian distribution. In the following test stage, we assume that a slow drift or rapid drift occurs. In the slow drift case, the mean of each feature is set to change three times the standard deviation of the feature (3σ) during the production of 1,000 wafers. In the rapid drift case, the mean change is set to ten times the standard deviation (10σ). In each stage, the number of fault samples is set to 20% of the total number of samples.

C. Results and Discussion

Table I shows the performance results (G-means and the final cluster amounts) of the IC-FDM for five significance levels and three smoothing parameter values. The significance level α in (3), which refers to the probability of the right-tail area of the Fisher distribution, controls the number of clusters. When α increases, the membership threshold θ_p in (3) decreases. Then, test samples are likely to create independent single-member clusters instead of belonging to an existing cluster. Four examples in Table I show that larger values of α yield more clusters. However, even for an inappropriately large α, the IC-FDM prevents an excessive number of clusters by merging the clusters. As shown in the results, the smoothing parameter β only slightly affected the performance because the examples did not consider process drift (when $\beta = 0$, we used the mean vector update formula in (6)).

For the Gaussian case in Table I, the IC-FDM exhibited the best G-mean (0.931 for $\alpha = 0.01$). Because the IC-FDM represents a cluster using the covariance matrix, it generates an elliptical cluster. Therefore, the normal data region is well represented by a small number of clusters when the distribution of the normal data in the feature space resembles an ellipse. As shown in Fig. 2(a), it was possible to approximate the entire normal region containing 400 test samples using six overlapped clusters; a large cluster represented most of the data, while relatively small clusters were generated at the boundary of the distribution to manage the outliers of the Gaussian distribution. This result implies that an optimistic approach – increasing the membership threshold using a small $\alpha-$ enabled the IC-FDM

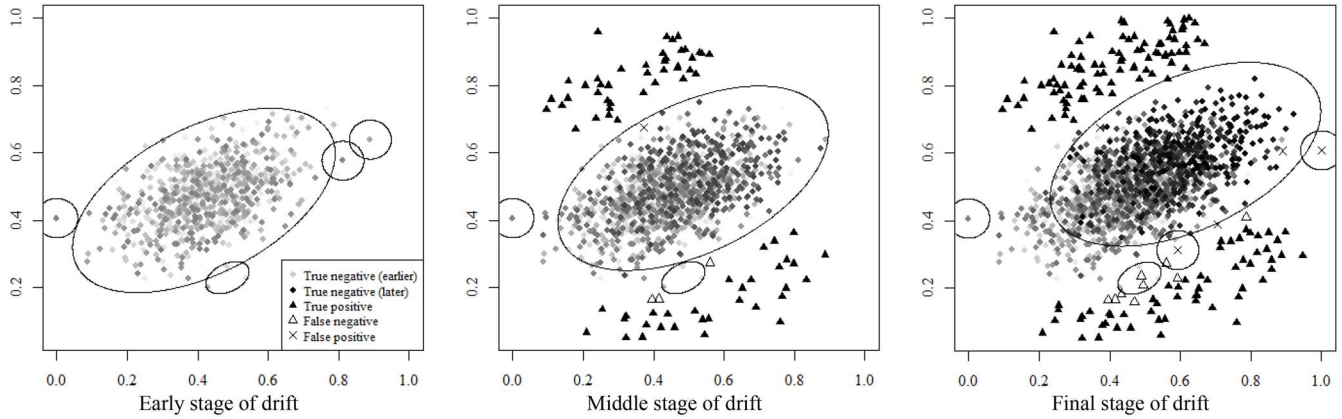

Fig. 3. Clustering process for slow drift (3σ) ($\alpha = 0.01, \beta = 0.01$).

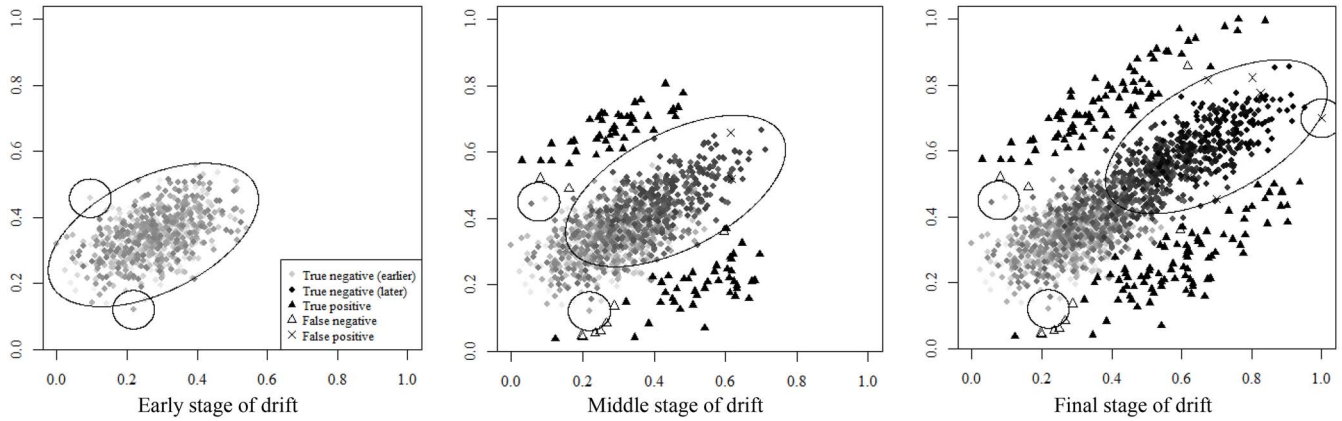

Fig. 4. Clustering process for fast drift (10σ) ($\alpha = 0.01, \beta = 0.01$).

to actively update and merge clusters to approximate the overall data region using a small number of clusters. The clusters in the figure were plotted using the chemometrics package [9].

Fig. 2(b) shows the Frank-Copula distribution, where the generated samples form a rectangular normal region. The IC-FDM precisely represents the region formed with 400 test samples using nine elliptical clusters. Fig. 2(c) and (d) shows the classification results of the IC-FDM for the samples generated from a t-Copula distribution and Clayton-Copula distribution. The normal data regions in these two figures appear difficult to approximate with few clusters. The IC-FDM requires more small clusters than the Gaussian and Frank-Copula cases to distinguish between normal data and fault data regions. Fig. 2(c) depicts the t-Copula distribution where the normal data region formed a rhombus shape. The IC-FDM generated 11 elliptical clusters to approximate the region. Fig. 2(d) shows a Clayton-Copula distribution, where the lower left end of the normal region is narrow and the upper right side of the normal region has a circular, water droplet shape. In this case, the normal region was represented by a combination of 13 clusters. For the three examples, conservative cluster update and merge were required to obtain high G-means by setting higher α of 0.1 than the value of 0.01 in the Gaussian case. Fig. 2 demonstrates that the IC-FDM creates several local overlapping ellipsoids as a chain to represent non-elliptical global data regions and undertakes classification based on this chain.

Table II shows the performance results when the IC-FDM was tested under two drift scenarios at $\alpha = 0.01$ and $\beta = 0.01$. The IC-FDM successfully represents the normal region using no more than five clusters, which resulted in a high G-mean of 0.967 and a high average total accuracy of 0.984. Additionally, the FPR (type I error) and FNR (type II error) were very low, i.e., averages of 0.0075 for FPR and 0.058 for FNR, respectively. Figs. 3 and 4 show how the IC-FDM moved the clusters during process drift. In the figures, the early, middle, and final stages depict the samples generated before the drift event, the samples during the progress of drift, and all samples at the end of the drift, respectively. The two figures show that the clusters were managed differently depending on the drift speed. The IC-FDM prefers to update existing clusters to stay current with the changing normal region when the drift speed is slow (see Fig. 3), while it favors chasing new normal samples from stage to stage when the drift speed is fast (see Fig. 4).

IV. Industrial Example

A. Set Up

This section compares the performance of the IC-FDM with the FD-kNN [16] ($k = 10, \alpha = 0.05$), PC-kNN [17] ($k = 10$, $\alpha = 0.05$), MD-kNN [30] ($K = 100, k = 10, \alpha = 0.05$),

where k is the number of neighbors used for determining the membership of a new sample; K is the number of neighbors used for estimating the local covariance matrix of a new sample; α is the significance level for the membership criterion. Based on the experiment descriptions in the references, we tested several parameter value combinations and selected the best one for each algorithm. For the PC-kNN, the number of principal components (PCs) was selected such that more than 90% of the total variance of training samples are explained. The FD-kNN and PC-kNN use the Euclidean distance and the MD-kNN uses the Mahalanobis distance. In addition, standard SVM [4], and one-class SVM [29] were introduced as comparison models. The 'linear (L)', 'polynomial (P)', 'radial-basis (R)' and 'sigmoid (S)' SVM kernel functions were tested in the experiment. The trade-off parameter of the one-class SVM, which corresponds to an expected fraction of outliers within the feature vectors, was optimized using a particle swarm optimization method because the one-class SVM did not perform well without optimally tuning the parameter [29].

This experiment was conducted using a dataset obtained by simulating real wafer data from a plasma etcher [32]. Each wafer data record consisted of 18 process variables. The standard recipe for the etching process consists of six steps. Because steps 4 and 5 represent the main process steps, the sensor records collected from steps 4 and 5 for each variable were used in the experiment. Traces of the all variables were individually summarized into single values by averaging the time-indexed trace values. The comparison algorithms were applied to the summary data, which were normalized into the hypercube $[0, 1]^{18}$. The total number of normal wafers was 5000, in which 18 data fields were collected for each wafer. Half of these wafers were reserved for training, and the remaining wafers were used for testing. We considered four imbalanced ratios between normal data and fault data for the training and testing procedures: 10:1 (500 fault records), 50:1 (100 fault records), and 100:1 (50 fault records). The dataset used in the experiment is accessible online[1].

B. Results and Discussion

As shown in Table III, the IC-FDM performed better than the other algorithms. Specifically, when the class imbalance ratio was 10:1, the SVM with a radial basis kernel function (SVM-R) resulted in the best performance in terms of the G-mean (0.895). The IC-FDM.3 ($\alpha = 0.1, \beta = 0.005$) ranked second (G-mean of 0.859). In particular, the performance was achieved by abstracting 5,000 wafer data records into 70 clusters. From the perspective of the FNR, the first-ranked classifier was the one-class SVM with a radial basis kernel function; the second was the IC-FDM.4 ($\alpha = 0.1, \beta = 0.01$). The FNRs were 0.108 and 0.151, respectively. For the FPR, the IC-FDM.1 ($\alpha = 0.01, \beta = 0.01$) was ranked first (FPR = 0.010), while the SVM-P was second (FPR = 0.011). When the class imbalance ratio was 50:1, the IC-FDM.2 ($\alpha = 0.1, \beta = 0$) exhibited the best G-mean (0.847) and the lowest FNR (0.144). For the FPR, the SVM-P resulted in the best performance (0.001), although the FNR was very high (0.602). When the class imbalance ratio was 100:1, the IC-FDM.3 ($\alpha = 0.1, \beta = 0.005$) was the best in terms of the G-mean. Moreover, its FNR and FPR were not high.

The results in Table III signify that the four IC-FDMs with different combinations of the parameters have similarly high performance levels except the IC-FDM.1 ($\alpha = 0.01$, $\beta = 0.01$), which has relatively high FNRs compared to the other three IC-FDMs. Moreover, the performance of the IC-FDMs did not decrease when the class imbalance was worsened. On the contrary, the SVMs, which exhibited good G-means exceeding 0.7 for an imbalance ratio of 10: 1, exhibited a rapid performance drop as the ratio increased to 50:1 and 100:1 – the FNRs of the SVMs increased substantially with the imbalance ratio. The SVM models are well-known fault detection algorithms in the semiconducting manufacturing field. However, our observations indicate that the models are difficult to apply for datasets in which there are only a few faults. Unlike our expectation, the one-class SVMs, which were developed for extremely imbalanced problems (e.g., outlier detection), did not exhibit excellent performance, although parameter optimization was conducted. This result demonstrates that the dataset used in this experiment is not readily classified using outlier detection algorithms because faults (outliers) cannot be clearly separated from normal data in the feature space. The FD-kNN, PC-kNN, and MD-kNN exhibited stable G-mean scores for the three class imbalance scenarios. However, the performance of the IC-FDMS was superior to that of the three algorithms. In particular, the FNRs of the three algorithms exceeded 0.4.

Phase 2 (prototype update and generation) of the IC-FDM algorithm requires feedback on the actual class labels of test wafers to update the cluster prototypes. Three feedback scenarios have been considered in this paper. The first scenario assumes that the IC-FDM receives no class information about the test wafers (no inspection). In this scenario, the class labels for test wafers predicted by the IC-FDM are only used to update clusters. The second scenario assumes that the operator inspects all test wafers and provides actual class information to the IC-FDM (full inspection). This scenario is ideal, although it is costly and time-consuming. In the work-site, random sampling is a standard policy. In the third scenario, the class labels of sampled wafers are only supplied to the IC-FDM (sample inspection).

Table IV shows the classification results for the three scenarios. In this experiment, we used the IC-FDM with $\alpha = 0.1$ and $\beta = 0.005$, which demonstrated high overall performance in the previous experiment. As expected, the 'full inspection' was the best in terms of the G-mean, followed by the 'sample inspection' and the 'no inspection'. However, the performance gap was very small, i.e., the maximum gap in the G-mean among the three scenarios in each class imbalance case was 0.011. Accordingly, the three scenarios resulted in very similar levels of FNR and FPR. The results demonstrate that the performance of the IC-FDM in work-site process environments is robust to wafer inspection policies. Note that the 'no inspection' case produced the least number of clusters because the IC-FDM did not generate single-member clusters. In phase 2 of the algorithm, a single-member cluster is created when

[1]http://isd.yonsei.ac.kr/data

TABLE III
COMPARISON RESULTS

Imbalance ratio	Classifier	FPR	FNR	G-mean	Accuracy
10:1	FD-*k*NN	0.050	0.475	0.705	0.786
	PC-*k*NN	0.046	0.554	0.650	0.758
	MD-*k*NN	0.051	0.418	0.742	0.808
	SVM-L	0.136	0.302	0.776	0.800
	SVM-P	0.011	0.381	0.782	0.847
	SVM-R	0.046	0.160	0.895	0.910
	SVM-S	0.180	0.395	0.704	0.738
	One-class SVM-L	0.258	0.668	0.482	0.584
	One-class SVM-P	0.829	0.373	0.327	0.346
	One-class SVM-R	0.198	0.108	0.845	0.836
	One-class SVM-S	0.014	0.926	0.259	0.635
	IC-FDM.1 ($\alpha = 0.01, \beta$=0.01)	0.010	0.502	0.701 (5)	0.801
	IC-FDM.2 ($\alpha = 0.1, \beta$ =0)	0.134	0.159	0.853 (74)	0.857
	IC-FDM.3 ($\alpha = 0.1, \beta$ =0.005)	0.124	0.156	0.859 (70)	0.863
	IC-FDM.4 ($\alpha = 0.1, \beta$ =0.01)	0.141	0.151	0.854 (74)	0.855
50:1	FD-*k*NN	0.050	0.484	0.698	0.902
	PC-*k*NN	0.050	0.610	0.604	0.888
	MD-*k*NN	0.050	0.422	0.739	0.909
	SVM-L	0.018	0.704	0.523	0.906
	SVM-P	0.001	0.602	0.629	0.932
	SVM-R	0.002	0.507	0.697	0.941
	SVM-S	0.006	0.961	0.137	0.888
	One-class SVM-L	0.265	0.670	0.483	0.690
	One-class SVM-P	0.820	0.345	0.343	0.233
	One-class SVM-R	0.189	0.114	0.847	0.819
	One-class SVM-S	0.013	0.935	0.220	0.885
	IC-FDM.1 ($\alpha = 0.01, \beta$=0.01)	0.012	0.513	0.692 (5)	0.933
	IC-FDM.2 ($\alpha = 0.1, \beta$ =0)	0.123	0.144	0.866 (71)	0.875
	IC-FDM.3 ($\alpha = 0.1, \beta$ =0.005)	0.130	0.159	0.854 (73)	0.867
	IC-FDM.4 ($\alpha = 0.1, \beta$ =0.01)	0.138	0.153	0.854 (72)	0.860
100:1	FD-*k*NN	0.051	0.451	0.714	0.926
	PC-*k*NN	0.047	0.647	0.576	0.919
	MD-*k*NN	0.052	0.394	0.753	0.929
	SVM-L	0.008	0.819	0.365	0.946
	SVM-P	0.001	0.739	0.504	0.958
	SVM-R	0.000	0.928	0.234	0.947
	SVM-S	0.000	0.997	0.014	0.944
	One-class SVM-L	0.244	0.679	0.481	0.731
	One-class SVM-P	0.832	0.319	0.336	0.197
	One-class SVM-R	0.188	0.107	0.851	0.817
	One-class SVM-S	0.013	0.919	0.239	0.936
	IC-FDM.1 ($\alpha = 0.01, \beta$=0.01)	0.010	0.471	0.720 (5)	0.964
	IC-FDM.2 ($\alpha = 0.1, \beta$ =0)	0.136	0.143	0.860 (78)	0.864
	IC-FDM.3 ($\alpha = 0.1, \beta$ =0.005)	0.120	0.132	0.873 (68)	0.880
	IC-FDM.4 ($\alpha = 0.1, \beta$ =0.01)	0.136	0.139	0.862 (70)	0.864

TABLE IV
RESULTS OF THREE SCENARIOS

Imbalance ratio	Scenario	FPR	FNR	G-mean	Accuracy
10:1	Full inspection	0.141	0.156	0.859 (74)	0.863
	Sample inspection	0.141	0.156	0.858 (54)	0.861
	No inspection	0.129	0.162	0.848 (28)	0.851
50:1	Full inspection	0.143	0.159	0.854 (73)	0.867
	Sample inspection	0.125	0.159	0.853 (54)	0.863
	No inspection	0.134	0.152	0.852 (29)	0.856
100:1	Full inspection	0.135	0.132	0.873 (68)	0.880
	Sample inspection	0.128	0.146	0.863 (49)	0.873
	No inspection	0.126	0.140	0.862 (29)	0.865

a normal wafer is misclassified as a fault. The 'no inspection' case does not provide the true class of a wafer; therefore, the IC-FDM does not know whether it correctly classified a wafer. The algorithm only expands the normal clusters created during the training period. On the contrary, the 'full inspection' case actively creates single-member clusters during the testing period using the true class information of the test wafers. The number of clusters generated in the 'random inspection' case was between the other two scenarios.

V. CONCLUSION

In this study, we proposed a new fault detection algorithm based on incremental clustering. Each time a new wafer is

accepted, the algorithm classifies the wafer as a member of the closest cluster using the Mahalanobis distance and probability-based membership tests. The algorithm is incremental because it immediately reflects new wafer information in the prototype of the cluster and merges clusters if necessary. These characteristics are highly advantageous when performing fault detection in stream data environments.

The proposed algorithm exhibits good performance in our experiments even with imbalanced class distributions and under process drifts. In particular, in the drift case, where the normal data region moves as the number of wafers being processed increases, the proposed algorithm can reflect the change in the data by updating the existing cluster regions. In the industrial example, where four class imbalance cases are considered, the algorithm can represent 5,000 normal data records using less than 80 clusters and obtain a higher G-mean than the comparison algorithms.

This study focused on reducing the number of data records. Future research should consider reducing the number of variables in fault detection algorithms. PC-kNN reduces the variables using a PCA [17]. However, because PCA is a type of unsupervised learning, some interference variables with large variances but small contributions to wafer faults may be introduced in the principal components. To solve this problem, less important variables for fault detection should be removed before applying the IC-FDM.

APPENDIX

A. Inversion of the Sum of Two Matrices With Rank-One Matrix

$$\left(\boldsymbol{S}^{new}\right)^{-1} = \left[\frac{n_p^{old}-1}{n_p^{old}}\boldsymbol{S}^{old} + \frac{1}{n_p^{old}}\left(\boldsymbol{x}^{new}-\boldsymbol{m}^{new}\right)\left(\boldsymbol{x}^{new}-\boldsymbol{m}^{new}\right)^T\right]^{-1}, \tag{A.1}$$

Let a represents $1/n^{old}$. Additionally, $\boldsymbol{x}^{\text{new}}$ and $\boldsymbol{m}^{\text{new}}$ are replaced with $\boldsymbol{x}$ and $\boldsymbol{m}$. Then,

$$\left(\boldsymbol{S}^{new}\right)^{-1} = \left[(1-a)\boldsymbol{S}^{old} + a(\boldsymbol{x}-\boldsymbol{m})(\boldsymbol{x}-\boldsymbol{m})^T\right]^{-1} \tag{A.2}$$

$$= \left[(1-a)\boldsymbol{S}^{old}\left\{\boldsymbol{I} + \frac{a}{1-a}\left(\boldsymbol{S}^{old}\right)^{-1}(\boldsymbol{x}-\boldsymbol{m})(\boldsymbol{x}-\boldsymbol{m})^T\right\}\right]^{-1}, \tag{A.3}$$

$$= \left[\boldsymbol{I} + \frac{a}{1-a}\left(\boldsymbol{S}^{old}\right)^{-1}(\boldsymbol{x}-\boldsymbol{m})(\boldsymbol{x}-\boldsymbol{m})^T\right]^{-1}\frac{\left(\boldsymbol{S}^{old}\right)^{-1}}{1-a}. \tag{A.4}$$

We apply the Neumann series to invert the sum of matrices in the parenthesis [22]:

$$\left(\boldsymbol{S}^{new}\right)^{-1} = \left\{\mathbf{I} - \left(\frac{a}{1-a}\right)\left(\boldsymbol{S}^{old}\right)^{-1}(\boldsymbol{x}-\boldsymbol{m})(\boldsymbol{x}-\boldsymbol{m})^T + \left(\frac{a}{1-a}\right)^2\left(\boldsymbol{S}^{old}\right)^{-1}(\boldsymbol{x}-\boldsymbol{m})(\boldsymbol{x}-\boldsymbol{m})^T\left(\boldsymbol{S}^{old}\right)^{-1}(\boldsymbol{x}-\boldsymbol{m})(\boldsymbol{x}-\boldsymbol{m})^T - \ldots\right\}\frac{\left(\boldsymbol{S}^{old}\right)^{-1}}{1-a}. \tag{A.5}$$

Let b and λ denote $\frac{a}{1-a}$ and $(\boldsymbol{x}-\boldsymbol{m})^T(\boldsymbol{S}^{old})^{-1}(\boldsymbol{x}-\boldsymbol{m})$, respectively. Then,

$$\left(\boldsymbol{S}^{new}\right)^{-1} = \frac{\left(\boldsymbol{S}^{old}\right)^{-1}}{1-a} - \frac{b}{1-a}\left(\boldsymbol{S}^{old}\right)^{-1}(\boldsymbol{x}-\boldsymbol{m})(\boldsymbol{x}-\boldsymbol{m})^T \times\left\{1-\lambda b+\lambda^2 b^2-\ldots\right\}, \tag{A.6}$$

$$= \frac{\left(\boldsymbol{S}^{old}\right)^{-1}}{1-a} - b\cdot\frac{\left(\boldsymbol{S}^{old}\right)^{-1}(\boldsymbol{x}-\boldsymbol{m})\left\{\left(\boldsymbol{S}^{old}\right)^{-1}(\boldsymbol{x}-\boldsymbol{m})\right\}^T}{1+\left\{a(\boldsymbol{x}-\boldsymbol{m})^T\left(\boldsymbol{S}^{old}\right)^{-1}(\boldsymbol{x}-\boldsymbol{m})-1\right\}}, \tag{A.7}$$

$$= \frac{n^{old}}{n^{old}-1}\left(\boldsymbol{S}^{old}\right)^{-1} - \frac{n^{old}}{n^{old}-1}\cdot\frac{\left(\boldsymbol{S}^{old}\right)^{-1}(\boldsymbol{x}-\boldsymbol{m})\left\{\left(\boldsymbol{S}^{old}\right)^{-1}(\boldsymbol{x}-\boldsymbol{m})\right\}^T}{n^{old}+\left\{(\boldsymbol{x}-\boldsymbol{m})^T\left(\boldsymbol{S}^{old}\right)^{-1}(\boldsymbol{x}-\boldsymbol{m})-1\right\}}. \tag{A.8}$$

B. Inversion of the Sum of Two Invertible Matrices

When two matrices A and B are both invertible, then $(\boldsymbol{A}+\boldsymbol{B})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}(\boldsymbol{A}^{-1}+\boldsymbol{B}^{-1})^{-1}\boldsymbol{A}^{-1}$ holds [8].

1) Shrinkage method

Let $\boldsymbol{A} = (1-\lambda)\boldsymbol{S}_p^{new}$ and $\boldsymbol{B} = \lambda\boldsymbol{T}_p$. Then,

$$\left(\hat{\boldsymbol{\Sigma}}_{\mathbf{p}}\right)^{-1} = \left(\lambda\boldsymbol{T}_p + (1-\lambda)\boldsymbol{S}_p^{new}\right)^{-1}, \tag{B.1}$$

$$= \frac{1}{1-\lambda}\left(\boldsymbol{S}_p^{new}\right)^{-1} - \frac{1}{(1-\lambda)^2} \times\left(\boldsymbol{S}_p^{new}\right)^{-1}\left(\frac{1}{1-\lambda}\left(\boldsymbol{S}_p^{new}\right)^{-1} + \frac{1}{\lambda}\left(\boldsymbol{T}_p\right)^{-1}\right)\left(\boldsymbol{S}_p^{new}\right)^{-1}. \tag{B.2}$$

2) Merged clusters

Let $\mathrm{A} = \frac{n_{p_m}}{n_{p_m}+n_{p'}}\hat{\boldsymbol{\Sigma}}_{p_m}$ and $\mathrm{B} = \frac{n_{p'}}{n_{p_m}+n_{p'}}\hat{\boldsymbol{\Sigma}}_{p'}$. Then,

$$\boldsymbol{\Delta}^{-1} = \left(\frac{n_{p_m}}{n_{p_m}+n_{p'}}\hat{\boldsymbol{\Sigma}}_{p_m} + \frac{n_{p'}}{n_{p_m}+n_{p'}}\hat{\boldsymbol{\Sigma}}_{p'}\right)^{-1}, \tag{B.3}$$

$$= \frac{n_{p_m}+n_{p'}}{n_{p_m}n_{p'}}\hat{\boldsymbol{\Sigma}}_{p_m}^{-1} - \left(\frac{n_{p_m}+n_{p'}}{n_{p_m}}\right)^2\hat{\boldsymbol{\Sigma}}_{p_m}^{-1} \times\left(\frac{n_{p_m}+n_{p'}}{n_{p_m}}\hat{\boldsymbol{\Sigma}}_{p_m}^{-1} + \frac{n_{p_m}+n_{p'}}{n_{p'}}\hat{\boldsymbol{\Sigma}}_{p'}^{-1}\right)^{-1}\hat{\boldsymbol{\Sigma}}_{p_m}^{-1}. \tag{B.4}$$

REFERENCES

[1] E. Byon, A. K. Shrivastava, and Y. Ding, "A classification procedure for highly imbalanced class sizes," *IIE Trans.*, vol. 42, no. 4, pp. 288–308, 2010.

[2] C. M. Chandler, "Formulation of lean six sigma critical business processes for manufacturing facilities," Ph.D. dissertation, Dept. Ind. Eng., Univ. Texas Arlington, Arlington, TX, USA, 2007.

[3] H. Chen, P. Tino, X. Yao, and A. Rodan, "Learning in the model space for fault diagnosis," *IEEE Trans. Neural Netw.,* vol. 25, no. 1, pp. 124–136, Jan. 2014.

[4] C. Cortes and V. Vapnik, "Support vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[5] S. De Backer and P. Scheunders, "Texture segmentation by frequency-sensitive elliptical competitive learning," *Image Vis. Comput.*, vol. 19, nos. 9–10, pp. 639–648, 2001.
[6] R. De Maesschalck and D. Jouan-Rimbaud, "The Mahalanobis distance," *Chemomet. Intell. Lab. Syst.*, vol. 50, no. 1, pp. 1–18, 2000.
[7] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. New York, NY, USA: Wiley, 2000.
[8] C. A. Felippa, *Introduction to Finite Element Methods*, Univ. Colorado Boulder, Boulder, CO, USA, 2004. [Online]. Available: http://www.colorado.edu/engineering/CAS/courses.d/IFEM.d
[9] P. Filzmoser and K. Varmuza. (2012). *Chemometrics: Multivariate Statistical Analysis in Chemometrics. R Package Version 1.3.8.* [Online]. Available: http://artax.karlin.mff.cuni.cz/r-help/library/chemometrics/html/drawMahal.html
[10] K. Fu, M. Zhu, P. Liu, and G. Wang, "Incremental fault diagnosis for nonlinear processes," *Adv. Mater. Res*., vols. 433–440, pp. 6430–6436, Jan. 2012.
[11] Z. Ge and Z. Song, "Semiconductor manufacturing process monitoring based on adaptive substatistical PCA," *IEEE Trans. Semicond. Manuf.,* vol. 23, no. 1, pp. 99–108, Feb. 2010.
[12] G. L. Grinblat, L. C. Uzal, H. A. Ceccatto, and P. M. Granitto, "Solving nonstationary classification problems with coupled support vector machines," *IEEE Trans. Neural Netw.,* vol. 22, no. 1, pp. 37–51, Jan. 2011.
[13] G. L. Grinblat, L. C. Uzal, and P. M. Granitto, "Abrupt change detection with one-class time-adaptive support vector machines," *Expert Syst. Appl.,* vol. 40, no. 18, pp. 7242–7249, 2013.
[14] H.-F. Guo, C. J. Spanos, and A. J. Miller, "Real time statistical process control for plasma etching," in *Proc. IEEE/SEMI Int. Semicond. Manuf. Sci. Symp.,* Burlingame, CA, USA, 1991, pp. 113–118.
[15] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
[16] Q. He and J. Wang, "Fault detection using the k-nearest neighbor rule for semiconductor manufacturing processes," *IEEE Trans. Semicond. Manuf.,* vol. 20, no. 4, pp. 345–354, Nov. 2007.
[17] Q. He and J. Wang, "Large-scale semiconductor process fault detection using a fast pattern recognition-based method," *IEEE Trans. Semicond. Manuf.*, vol. 23, no. 2, pp. 194–200, May 2010.
[18] M. Hofert and M. Mächler, "Nested Archimedean copulas meet R: The nacopula package," *J. Statist. Softw.*, vol. 39, no. 9, pp. 1–20, 2011.
[19] Y. Li and X. Zhang, "Diffusion maps based k-nearest-neighbor rule technique for semiconductor manufacturing process fault detection," *Chemometr. Intell. Lab. Syst.,* vol. 136, pp. 47–57, Aug. 2014.
[20] E. Lughofer, "Extensions of vector quantization for incremental clustering," *Pattern Recognit.*, vol. 41, no. 3, pp. 995–1011, 2008.
[21] E. Lughofer, "A dynamic split-and-merge approach for evolving cluster models," *Evol. Syst.*, vol. 3, no. 3, pp.135–151, 2012.
[22] E. Lughofer and M. Sayed-Mouchaweh, "Autonomous data stream clustering implementing incremental split-and-merge techniques—Towards a plug-and-play approach," *Inf. Sci*., vol. 304, pp. 54–79, May 2015.
[23] S. Mahadevan and S. Shah, "Fault detection and diagnosis in process data using one-class support vector machines," *J. Process Control,* vol. 19, no. 10, pp. 1627–1639, 2009.
[24] S. J. Qin, W. Li, and H. H. Yue, "Recursive PCA for adaptive process monitoring," *J. Process Control*, vol. 10, no. 5, pp. 471–486, 2000.
[25] J. Schäfer and K. Strimmer, "A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics," *Statist. Appl. Genet. Mol. Biol.*, vol. 4, no. 1, pp. 1–30, 2005.
[26] C. Schöelzel and P. Friederichs, "Multivariate non-normally distributed random variables in climate research—Introduction to the copula approach," *Nonlin. Process. Geophys.,* vol. 15, no. 5, pp. 761–772, 2008.
[27] F. Serdio *et al.*, "Fault detection in multi-sensor networks based on multivariate time-series models and orthogonal transformations," *Inf. Fus.,* vol. 20, pp. 272–291, Nov. 2014.
[28] F. Serdio, E. Lughofer, K. Pichler, T. Buchegger, and H. Efendic, "Residual-based fault detection using soft computing techniques for condition monitoring at rolling mills," *Inf. Sci.,* vol. 259, pp. 304–320, Feb. 2014.
[29] J. Tian and H. Gu, "Anomaly detection combining one-class SVMs and particle swarm optimization algorithms," *Nonlin. Dyn.*, vol. 61, nos. 1–2, pp. 303–310, 2010.
[30] G. Verdier and A. Ferreria, "Adaptive Mahalanobis distance and k-nearest neighbor rule for fault detection in semiconductor manufacturing," *IEEE Trans. Semicond. Manuf.,* vol. 24, no. 1, pp. 59–68, Feb. 2011.
[31] D. Ververidis and C. Kotropoulos, "Information loss of the Mahalanobis distance in high dimensions: Application to feature selection," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 31, no. 12, pp. 2275–2281, Dec. 2009.
[32] B. M. Wise, N. B. Gallagher, S. W. Butler, D. D. White, Jr., and G. G. Barna, "A comparison of principal component analysis, multiway principal component analysis, trilinear decomposition and parallel factor analysis for fault detection in a semiconductor etch process," *J. Chemomet.,* vol. 13, nos. 3–4, pp. 379–396, 1999.

Jueun Kwak received the B.S. degree in information and industrial engineering from Yonsei University, Seoul, Korea, in 2011, where she is currently pursuing the Ph.D. degree.

Her current research interests include fault detection and classification models, image processing, and deep learning.

Taehyung Lee received the B.S. degree in industrial engineering from Hansung University, Seoul, Korea, in 2004, and the M.S. and Ph.D. degrees in information and industrial engineering from Yonsei University, Seoul, in 2008 and 2015, respectively.

His current research interests include time-series data mining, fault detection and classification models, and machine learning-based process control.

Chang Ouk Kim received the B.S. and M.S. degrees from Korea University, Seoul, Korea, in 1988 and 1990, respectively, and the Ph.D. degree from Purdue University, West Lafayette, IN, USA, in 1996, all in industrial engineering.

From 1998 to 2001, he was an Assistant Professor with the Department of Industrial Systems Engineering, Myongji University, Yongin, Korea. In 2002, he joined the Department of Information and Industrial Engineering, Yonsei University, Seoul, where he is a Professor and currently serves as an Associate Dean for Research in the College of Engineering. His current research interests include data mining models for semiconductor manufacturing, fault detection and classification models, image processing, and deep learning. He has published over 100 papers in journals and conference proceedings.