Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 368190, 14 pages
http://dx.doi.org/10.1155/2015/368190

Research Article

Fault Diagnosis with Evolving Fuzzy Classifier Based on Clustering Algorithm and Drift Detection

Maurilio Inacio,[1,2] Andre Lemos,[3] and Walmir Caminhas[3]

[1] *Graduate Program in Electrical Engineering, Federal University of Minas Gerais, Avenue Antônio Carlos 6627, 31270-901 Belo Horizonte, MG, Brazil*
[2] *Department of Computer Engineering, Faculdade de Ciência e Tecnologia de Montes Claros, Avenue Deputado Esteves Rodrigues 1637, 39400-142 Montes Claros, MG, Brazil*
[3] *Department of Electronics Engineering, Federal University of Minas Gerais, Avenue Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brazil*

Correspondence should be addressed to Maurilio Inacio; maurilio.j.inacio@gmail.com

Received 16 April 2014; Accepted 23 July 2014

Academic Editor: Minping Jia

The emergence of complex machinery and equipment in several areas demands efficient fault diagnosis methods. Several fault diagnosis methods based on different theories and approaches have been proposed in the literature. According to the concept of intelligent maintenance, the application of intelligent systems to accomplish fault diagnosis from process historical data has been shown to be a promising approach. In problems involving complex nonstationary dynamic systems, an adaptive fault diagnosis system is required to cope with changes in the monitored process. In order to address fault diagnosis in this scenario, use of the so-called "evolving intelligent systems" is suggested. This paper proposes the application of an evolving fuzzy classifier for fault diagnosis based on a new approach that combines a recursive clustering algorithm and a drift detection method. In this approach, the clustering update depends not only on a similarity measure, but also on the monitoring changes in the input data flow. A merging cluster mechanism was incorporated into the algorithm to enable the removal of redundant clusters. Multivariate Gaussian memberships functions are employed in the fuzzy rules to avoid information loss if there is interaction between variables. The novel approach provides greater robustness to outliers and noise present in data from process sensors. The classifier is evaluated in fault diagnosis of a DC drive system. In the experiments, a DC drive system fault simulator was used to simulate normal operation and several faulty conditions. Outliers and noise were added to the simulated data to evaluate the robustness of the fault diagnosis model.

1. Introduction

The advance of technology has resulted in the emergence of machinery and complex equipment, which imposes great challenges for its management and maintenance. In industries, for instance, fault diagnosis in major processes is vitally important to assure normal operation of a plant. In these cases, due to the complexity of the systems, it is infeasible for human operators to diagnose abnormal situations (faults) in a timely manner, leading them to take wrong decisions. Statistical studies indicate that approximately 70% of the accidents in industries are caused by human error, which can account for economic losses, security reductions, and environmental damages [1].

This scenario led to the emergence of new concepts on management and maintenance of machinery and equipment, such as condition-based maintenance (CBM) [2]. CBM refers to the use of machine or equipment data obtained in real time to infer its working condition (or faulty condition), allowing maintenance scheduling and preventing equipment crashes. Based on CBM, the concept of intelligent maintenance has emerged [3]. It employs advanced fault diagnosis systems to achieve the desired goals. Thus, intelligent maintenance becomes necessary for current complex machinery and equipment.

Over the past decades, several intelligent fault diagnosis methods based on different theories and approaches have been proposed in the literature. In general, these methods use mathematical/statistical models, accumulated experience, or even process data to perform fault diagnosis [1]. Although methods based on models or experience have shown to be effective, they have the disadvantage of requiring previous knowledge of the dynamic system in question. On the contrary, methods based on process data do not require prior knowledge. They are based solely on data obtained directly from the system.

Recently, fault diagnosis methods based on process data have received great emphasis, since the acquisition of data through sensors is widely common in today's automation systems [4, 5]. Given this current scenario, many times it is easier to extract knowledge from data than developing a model or accumulating experience. In this type of diagnosis, several works have already proposed data based diagnostics methods employing so-called "intelligent systems," which are tools derived from computational intelligence, mainly artificial neural networks, fuzzy systems, and neurofuzzy networks, among others [2].

However, despite the good performance achieved by intelligent systems in fault diagnosis, they tend to face difficulties when the problem involves complex nonstationary dynamic systems, which represent the vast majority of the current real cases. In such systems, physical parameters, operating characteristics and fault behaviours change over time, requiring an adaptive fault diagnosis system, able to self-adapt in favor to cope with changes in the monitored system. In order to address fault diagnosis in this scenario, several works propose the use of the so-called "evolving intelligent systems" [6–10].

Evolving intelligent systems are systems based on fuzzy inference systems, artificial neural networks, or a combination of both, the neurofuzzy networks, whose main characteristic is the ability to gradually determine both its structure and parameters from input data acquired in online mode and often in real time [11, 12]. The application of evolving intelligent systems has been growing in recent years. Many works present successful applications in real world complex problems involving modeling, control, classification, or prediction [13]. An important aspect of evolving intelligent systems is that there are different theoretical and practical approaches which can be used for its implementation. Regardless of the approach to be used, the main features of evolving intelligent systems are as follows:

(i) its structure is not fixed and is not defined a priori: it grows (expands or shrinks) naturally as the system evolves;

(ii) its parameters are adjusted (adapted) as the system evolves;

(iii) the operation is continuous; that is, they are based on online learning algorithms and, if necessary, in real time.

One of the most used approaches to define the structure of an evolving intelligent system is unsupervised recursive clustering. Generally, the algorithm performs data clustering in the input or input-output data space in an incremental manner, defining the center of each cluster, and in some cases, the radius of the cluster (or zone of influence). During the evolving process, the algorithm can create new clusters, update existing clusters, or eliminate redundant ones. The models proposed in [14–21] are examples of intelligent systems based on evolving clustering algorithms.

Most evolving intelligent systems based on recursive clustering adopt a mechanism to update the structure and parameters of the system (creation/modification/removal of clusters) using some measure of similarity between input data samples and existing clusters. Although this mechanism is functional, it may lead to an erroneous definition of the structure, since outliers or noisy samples (as usually are the data acquired by sensors in industrial environments) which exceed the measure of similarity may generate clusters that do not effectively represent the data spacial structure [21]. Some evolving intelligent systems adopt more elaborated mechanisms to update the model structure and system parameters, such as the models proposed in [20, 21], using methods to ignore/filter outliers and noise.

Considering the fault diagnosis problem, the use of evolving intelligent systems based on recursive clustering algorithms robust to outliers and data noise is mandatory. In this problem, each new cluster created is usually associated with a new faulty condition. Thus, if the clustering procedure is not robust, the fault diagnosis model tends to have a high false alarm rate; that is, new faulty conditions are erroneously detected. In this context, this paper proposes a fault diagnosis approach based on an evolving fuzzy classifier which uses a new robust unsupervised recursive clustering algorithm. The proposed classifier uses a modified version of the Gustafson-Kessel (GK) clustering algorithm [22] with the incorporation of the drift detection method (DDM) [23].

GK is a powerful clustering algorithm. Unlike many others, it allows the identification of clusters with different shapes and orientations in space. The algorithm employs a technique to adapt the distance metric to the shape of each cluster using an estimation of the cluster covariance matrix. Furthermore, the GK algorithm has also the advantage of being relatively insensitive to data scale and initialization of the partition matrix [24]. Several applications have been proposed in the literature based on this clustering algorithm, such as time series prediction, dynamic systems modeling, fault diagnosis, and prognosis.

According to the literature, a drift detection is a method to detect gradual changes in the context of input data. By context, it is understood as a set of generated data when the process is stationary. Thus, a method for drift detection is able to detect time instants when changes occur in the context of the data. The detection of a new context suggests that the current model is outdated and needs to be updated using current relevant information. Drift detection methods are suitable for applications involving machine learning, where algorithms are applied to real world problems, in complex, nonstationary, and dynamic environments. In these appli- cations, large amounts of information are provided in a continuous flow of high-speed data presenting variations over time as, for example, real time monitoring of industrial plants [25]. The learning algorithms must be able to monitor the behavior of the dynamic system in question and adapt the model as changes occur. Among several methods proposed for drift detection, the DDM algorithm employs simple and computationally efficient method to detect moments when changes occur. It consists of an independent drift detection method, and it can be embedded into any learning algorithm, while increasing its efficiency in problems involving nonstationary dynamic models.

The new unsupervised recursive clustering algorithm proposed in this paper combines the advantages of the GK algorithm, especially the ability to identify clusters with different shapes and orientations in an online mode, with the DDM algorithm. The DDM algorithm is used to detect changes in the input stream triggering updates in the cluster structure. In the proposed algorithm, any clustering update depends not only on the similarity measure, but also on monitoring changes in the input data flow, which gives the algorithm a greater robustness to the presence of outliers and noise. A merging cluster mechanism was also incorporated into the algorithm to enable the removal of redundant clusters. The fuzzy rule base of the proposed classifier is updated whenever the cluster structure is modified. The clusters centers and covariance matrices are used as parameters of fuzzy rules. Multivariate Gaussian memberships functions are employed in the rules, characterized by a central vector and a dispersion matrix, which represents the current dispersion of the input variables, as well as the interactions between them [21].

In accordance with the characteristics of the proposed recursive clustering algorithm, the main benefits achieved by the classifier used in this work are

(i) the ability to learn faults of the dynamic system in online mode and, if necessary, in real time, eliminating the need for prior knowledge of the system;

(ii) the ability to adapt whenever changes are detected in the monitored system, allowing the application to real problems;

(iii) low false alarm rate and high fault isolation rate due to the robustness to outliers and noise, increasing the reliability of diagnosis.

To evaluate the performance of the proposed approach in fault diagnosis, a DC drive system fault simulator was used to simulate normal operation and several faulty conditions. Outliers and noise were added to the simulated data to evaluate the robustness of the fault diagnosis model.

This paper is organized as follows. Section 2 presents the theoretical concepts regarding recursive clustering algorithm, drift detection method, and presents the proposed recursive clustering algorithm. Next, Section 3 presents the proposed classifier and its application in fault diagnosis. Section 4 presents the experiments and results. Finally, Section 5 presents the conclusion and suggestions for future works.

2. Recursive Clustering Algorithm and Drift Detection

2.1. Recursive Gustaffson-Kessel Algorithm. In pattern recognition, clustering algorithms are among the most useful tools to solve problems that involve analysis of nonlabeled data, or unsupervised learning [26]. Over the past decades, thousands of clustering algorithms have been proposed [27], but most of them are based on the offline learning concept or batch learning; it is assumed that the entire dataset is previously available. However, for many applications, data is acquired in real time, requiring online learning.

In contrast to clustering algorithms for offline learning which find clusters employing an iterative strategy, such as K-means and Fuzzy C-Means (FCM) [27], clustering algorithms for online learning are based on recursive strategies, which allow the algorithm to find clusters processing each input data sample only once. Several algorithms have been proposed in the last years based on this approach, such as evolving clustering method (ECM) [14], evolving vector quantization (eVQ) [18], and eClustering [28]. A common feature of these algorithms is that they assume that the form of the clusters is spherical, which can be a limiting factor in real applications, where the clusters may have different shapes and orientation in space.

Unlike many clustering algorithms that employ Euclidian distance as measure of similarity, GK algorithm employs Mahalanobis distance, which allows the identification of clusters with ellipsoidal shapes. In this algorithm, the distance is defined as follows:

$$d_{ik}^2 = (x_k - v_i) A_i (x_k - v_i)^T, \quad (1)$$

where d_{ik}^2 represents the distance between an input data sample $x_k = [x_{k1}, \ldots, x_{kn}]$, $k = 1, \ldots, N$, and the cluster center v_i, $i = 1, \ldots, c$, where N is the number of data samples, n is the number of data dimensions, and c is the number of clusters. The norm-inducing matrix A_i, $i = 1, \ldots, c$, defines the shape and orientation of each cluster in space, which depends on a fuzzy covariance matrix F_i, $i = 1, \ldots, c$, and of the membership degree of the input data sample u_{ik}, $i = 1, \ldots, c$, $k = 1, \ldots, N$. The GK algorithm uses an iterative process to estimate the parameters of the clusters (the cluster center and fuzzy covariance matrix), which are used to define the distance d_{ik}^2 and membership degree u_{ik}. This process is finished when a certain convergence criterion is reached. But, as discussed at the beginning of this section, when the application requires the definition of clustering in online mode, a recursive procedure is required. More details about the GK algorithm can be found in [22].

In [24], an extended version of the GK algorithm named evolving GK-like algorithm (eGKL) is proposed. This approach estimates the number of clusters and performs the adaptation of its parameters recursively, maintaining the advantages of the GK algorithm, such as the ability to identify clusters with generic shapes and orientations. The eGKL algorithm does not demand any a priori information regarding the number of clusters. In order to estimate the number of clusters, a strategy to evaluate each new input data sample is used. The strategy checks if each sample belongs to an existing cluster. If the current data sample belongs to a cluster already set, the parameters of the cluster (center and covariance matrix) are updated. If the data sample does not belong to any of the existing clusters, it is used to define a new one. To evaluate the similarity between a new sample data and one of the existing clusters, the eGKL algorithm employs the Mahalanobis distance, defined as follows:

$$D_{ik}^2 = (x_k - v_i) F_i^{-1} (x_k - v_i)^T. \tag{2}$$

In this strategy, the current data sample belongs to an existing cluster if the distance to the cluster center is smaller than the cluster radius. The eGKL algorithm uses an approach inspired in concepts of statistical process control to estimate the radius of each cluster. In this approach, it is assumed that a sample belongs to a cluster if the following relationship holds:

$$D_{ik}^2 < \chi_{n,\beta}^2, \tag{3}$$

where $\chi_{n,\beta}^2$ is the value of a Chi-squared distribution with n degrees of freedom and a confidence interval β. The degrees of freedom n correspond to the input space dimension. This approach has the advantage of avoiding the problem called "curse of dimensionality" [29], that is, the problem of increasing the distance between two adjacent points with the increase in the input space dimensionality, since $\chi_{n,\beta}^2$ is proportional to the dimension of the input data.

In eGLK algorithm, if condition (3) is satisfied, it means that the current data sample belongs to a cluster, so the cluster parameters are updated. Otherwise, it is assumed that the current data sample does not belong to any one of the existing clusters, and a new cluster is created. The complete procedures of the eGLK algorithm can be seen in [24].

To increase the eGKL algorithm robustness to outliers, the authors propose a mechanism based on the number of data samples that belong to a cluster. In this mechanism, if the number of data samples M_i, $i = 1, \ldots, c$, already assigned to an existing cluster is less than $M_{\min}$ (a minimum number initially chosen), even if the new data sample does not belong to that cluster, the cluster parameters are updated. Although it is functional, this mechanism depends on the proper choice of parameters to the problem at hand, which can be difficult for problems where a priori information is not available.

2.2. Drift Detection Method. Several drift detection methods have been proposed. In general, they can be classified into two categories: methods that perform adaptive learning at regular intervals regardless of the occurrence of changes and methods that detect changes first and subsequently adapt the learning to these changes [25]. Considering the first category, methods can use time windows of fixed size or weight the data according to their age or utility [30–32]. When the time windows of a fixed size are used, at each time frame, learning is performed only with data samples included in the window. An inherent difficulty with methods using fixed-size windows is choosing the appropriate window size for each problem. In the second category, methods use some indicators monitored over time to detect changes, such as performance measures, data distribution, or data properties [23, 33, 34]. If during the monitoring process a drift is detect, actions are taken to adapt the model to the change that has occurred, as in the case of using adaptive size time window, where the actions are to adjust the window according to the extent of the context change.

The DDM algorithm, which belongs to the second category, employs a simple method with direct application. This method is based on monitoring the number of errors produced by a learning model during prediction. The method uses the Binomial distribution to determine the general form of the probability for the random variable that represents the number of prediction errors into a sequence of n input data samples. For each k data sample sequences, the error rate is the probability of the prediction error p_k with standard deviation $s_k = \sqrt{p_k(1-p_k)/k}$. According to the probability approximately correct (PAC) learning model [35], the error rate of the learning algorithm decreases with the increase of input data samples, and if the distribution is stationary, a significant increase in the error rate suggests context changes. In this case, it is assumed that the current model is inappropriate and should be updated.

In this method, while monitoring the error, it defines a warning and a drift level. When $p_k + s_k$ exceeds the warning level, the data samples are stored in memory. However, if $p_k + s_k$ exceeds the drift level, it is considered that there is a context change. In this situation, the model induced by the learning algorithm should be updated with the data samples stored since the time that the warning level has been reached. It is possible that the error increases and, after reaching the warning level, it decreases to lower levels. This situation corresponds to a false alarm, where there is no change of context and, therefore, no action is required and the data samples stored in the memory are no longer needed. More details about the DDM method can be found in [23].

The use of the DDM algorithm embedded in a model learning algorithm can keep the dynamic system model continuously updated to the current context. For instance, DDM can be used embedded in a recursive clustering algorithm. In this case, the definition of the clusters are adjusted whenever a context change is detected. DDM is used to avoid the nonrobust approach of creating new clusters whenever a similarity measure threshold is violated. This mechanism gives the recursive clustering algorithm a greater robustness to outliers and noise in applications where online learning of nonstationary dynamic models is necessary.

2.3. Proposed Algorithm. This section describes the proposed unsupervised recursive clustering algorithm with a new mechanism of clustering update. The algorithm is a recursive version of the GK algorithm, inspired by the eGKL algorithm, incorporating the DDM algorithm. In the proposed algorithm, clustering is performed in online mode and, if necessary, in real time.

Assuming that there is no a priori information about the clustering structure nor a initial set of input data samples, the proposed algorithm starts by associating the center of the first cluster c_1 to the first data sample x_1. The corresponding covariance matrix F_1, the learning rate α_1, and the number of samples associated with the first cluster M_1 are defined as follows:

$$\begin{aligned} c_1 &= x_1; \qquad F_1 = F_{\text{init}}; \\ \alpha_1 &= \alpha_{\text{init}}; \qquad M_1 = 1, \end{aligned} \tag{4}$$

where $F_{\text{init}} = \gamma I$; I is an identity matrix of n size, γ is a small positive number (default value: $\gamma = 10^{-2}$), and $\alpha_{\text{init}} \in [0,1]$ is the initial learning rate (default value: $\alpha_{\text{init}} = 0.5$).

The algorithm stops when all data samples are processed; otherwise, a new data sample x_k is obtained and the distance between the data sample and the centers of the existing clusters is computed:

$$D_{ik}^2 = (x_k - v_i) F_i^{-1} (x_k - v_i)^T, \quad i = 1,\ldots,c. \tag{5}$$

The similarity between the current data sample and the existing clusters is verified by the similarity condition

$$D_{ik}^2 < \chi_{n,\beta}^2, \quad i = 1,\ldots,c. \tag{6}$$

If similarity condition (6) is met for a given cluster, it is assumed that the current sample belongs to this cluster. The cluster parameters (center, covariance matrix, learning rate, and number of samples in the cluster) are then updated as follows:

$$\begin{gathered} v_q = v_q + \alpha_q \left(x_k - v_q\right), \\ F_q = F_q + \alpha_q \left(\left(x_k - v_q\right)^T \left(x_k - v_q\right) - F_q\right), \\ \alpha_q = \frac{\alpha_{\text{init}}}{M_q}, \\ M_q = M_q + 1, \end{gathered} \tag{7}$$

where $q = \arg\min_{i=1,\ldots,c}(D_{ik}^2)$.

If similarity condition (6) is not met, it is assumed that the current sample does not belong to any existing cluster. The algorithm increments a variable that represents the number of dissimilarities, $M_{\text{dis}} = M_{\text{dis}} + 1$; then, the error probability and the standard deviation are computed as

$$\begin{gathered} p = \frac{M_{\text{dis}}}{k}, \\ s = \sqrt{\frac{p(1-p)}{k}}. \end{gathered} \tag{8}$$

In this algorithm, the p and s values are stored whenever $p+s$ reach the lowest value during the process, obtaining $p_{\min}$ and $s_{\min}$. If the following condition is met,

$$p + s < p_{\min} + s_{\min}, \tag{9}$$

then $p_{\min} = p$ and $s_{\min} = s$. Note that, when algorithm starts, the p and s values must be initialized as a positive number, is suggested set at one for each value.

To decide whether the current data sample x_k represents a new cluster or it is just an outlier, warning and drift conditions are evaluated. The warning condition is verified as

$$p + s > p_{\min} + z_1 \cdot s_{\min}, \tag{10}$$

where z_1 is the warning level (default value: $z_1 = 2$). If the warning level is reached, then the current data sample is stored in a window of samples $W(\text{data})_j$, $j = 1,\ldots,m$ (where m is the current size of the window) and then the drift condition is evaluated. Otherwise, the algorithm processes the next input data sample. Drift condition is verified as

$$p + s > p_{\min} + z_2 \cdot s_{\min}, \tag{11}$$

where z_2 is the drift level (default value: $z_2 = 3$). If the drift level is reached, a new cluster is created and the center and the covariance matrix of the new cluster are determined by the samples stored in the data window as follows:

$$\begin{gathered} c = c + 1, \\ v_c = \frac{1}{m}\sum_{j=1}^{m} W(\text{data})_j, \\ F_c = \operatorname{cov}\left(W(\text{data})_j\right), \quad j = 1,\ldots,m. \end{gathered} \tag{12}$$

The remaining parameters of the new cluster (learning rate and number of samples in the cluster) are initialized as

$$\alpha_c = \alpha_{\text{init}}; \qquad M_c = 1. \tag{13}$$

In order to avoid redundant cluster formation, during the update, the similarity between clusters is checked. To achieve this, distances between the centers of the clusters are computed as follows:

$$\begin{gathered} D_{ij}^2 = \left(v_i - v_j\right) F_i^{-1} \left(v_i - v_j\right)^T, \quad i = 1,\ldots,c,\ j = 1,\ldots,c. \\ D_{ji}^2 = \left(v_j - v_i\right) F_j^{-1} \left(v_j - v_i\right)^T, \quad j = 1,\ldots,c,\ i = 1,\ldots,c. \end{gathered} \tag{14}$$

If one of the following similarity conditions is met for two existing clusters i and j,

$$\begin{gathered} D_{ij}^2 < \chi_{n,\beta}^2, \quad i = 1,\ldots,c,\ j = 1,\ldots,c, \\ D_{ji}^2 < \chi_{n,\beta}^2, \quad j = 1,\ldots,c,\ i = 1,\ldots,c, \end{gathered} \tag{15}$$

the clusters are merged. These clusters have a hyper ellipsoidal shape, defined by a mean vector, a covariance matrix, and a number of samples associated with each one. The combination of these two clusters produces a new one with parameters computed as follows [36]:

$$\begin{gathered} M_i = M_i + M_j, \\ v_i = \frac{M_i}{M_i + M_j} v_i + \frac{M_j}{M_i + M_j} v_j, \\ F_i = \frac{M_i - 1}{M_i + M_j + 1} F_i + \frac{M_j - 1}{M_i + M_j + 1} F_j \\ + \frac{M_i M_j}{M_i + M_j\left(M_i + M_j - 1\right)} \left(v_i - v_j\right)^T \left(v_i - v_j\right). \end{gathered} \tag{16}$$

Algorithm 1 summarizes the proposed recursive clustering algorithm.

```
Input: $x_k$, $\chi^2_{n,\beta}$, $F_{\text{init}}$, $\alpha_{\text{init}}$, $z_1$, $z_2$;
Output: $v_i$, $F_i$;
Read the first data sample $x_1$;
Initialize the first cluster;
for $k = 2, 3, \ldots$ do
    Read $x_k$;
    Compute $D^2_{ik}$ for all clusters;
    Identify the closest cluster;
    if $D^2_{qk} < \chi^2_{n,\beta}$ then
        Update the closest cluster;
    else
        Update the dissimilarity number $M_{\text{dis}}$;
        Compute $p$ and $s$;
        if $p + s < p_{\min} + s_{\min}$ then
            Update $p_{\min}$ and $s_{\min}$;
        end if
        if $p + s > p_{\min} + z_1 \cdot s_{\min}$ then
            Store $x_k$ in the data window $W(\text{data})_j$;
        end if
        if $p + s > p_{\min} + z_2 \cdot s_{\min}$ then
            Create new cluster;
        end if
    end if
    Compute $D^2_{ij}$ and $D^2_{ji}$ for all clusters;
    if $D^2_{ij} < \chi^2_{n,\beta}$ or $D^2_{ji} < \chi^2_{n,\beta}$ then
        Merge redundant clusters;
    end if
end for
```

Algorithm 1: Recursive clustering algorithm with drift detection.

3. Evolving Fuzzy Classifier for Fault Diagnosis

The use of algorithms for pattern classification is present in many current applications, such as fingerprint recognition for security systems, handwriting recognition on touch screen computers, DNA sequences identification in medical diagnostic softwares, and fault diagnosis in industrial equipment. In this context, the problem of pattern classification consists in assigning a class or a category for each data sample from a set of "raw" data [26].

In many applications, pattern classification algorithms based on fuzzy rules have been used due to their advantages in relation to classic algorithms for pattern classification [26], especially by the good prediction performance in real problems and good transparency in linguistic rules [37], which allows an easy comprehension of the dependence between pattern characteristics. The typical architecture of a fuzzy classifier consists of a set of IF THEN fuzzy rules, defined as

$$\begin{aligned}\text{RULE}_i : \text{IF } x_1 \text{ IS } \mu_{i1} \text{ AND} \ldots \text{AND } x_n \text{ IS } \mu_{in} \\ \text{THEN } y_i = L_i,\end{aligned} \tag{17}$$

where $[x_{k1}, \ldots, x_{kn}]$ are the input variables or input patterns of n dimensionality; $[\mu_{i1}, \ldots, \mu_{in}]$ are antecedent fuzzy sets of the ith fuzzy rule; y_i is the output; L_i is the crisp output corresponding to the class label from the set $[1, \ldots, K]$, where K is the number of classes.

The classification of each new input data sample x_k is obtained by assigning to it the label of the class associated with the rule having the highest activation degree. The class is determined as follows:

$$y_i = L_{i^*}, \tag{18}$$

where $i^* = \arg\max_{1<i<R}(\tau_i)$, R is the number of fuzzy rules, and τ_i is the activation degree of the ith fuzzy rule, defined by a t-norm, usually expressed as a product operator:

$$\tau_i = \mathbf{T}^n_{j=1}\mu_{ij}\left(x_j\right), \tag{19}$$

where μ_{ij} are the membership functions of fuzzy sets defined by Gaussians:

$$\mu_{ij} = e^{-(1/2)((x_j - v_{ij})^2/\sigma^2_{ij})}, \tag{20}$$

where v_{ij} and σ^2_{ij} represent, respectively, the membership functions center and variance.

To implement this fuzzy classifier architecture, clustering is usually performed in the input or input-output data space. Then, rules are created using one-dimensional (or univariate) fuzzy sets, generated from the projection of the clusters in the axis of each variable. According to [21], this approach can lead to information loss if there is interaction between variables, and, to avoid this, the authors propose the use of multivariate Gaussian membership functions to represent antecedent fuzzy sets of each rule. These membership functions are described as

$$H(x) = e^{-(1/2)((x-v)\Sigma^{-1}(x-v)^T)}, \tag{21}$$

where v is a $1 \times n$ central vector and Σ is a $n \times n$ symmetric positive definite matrix. The central vector is defined as the modal value and represents $H(x)$ typical value and the Σ matrix denotes the dispersion and represents $H(x)$ spreading.

In this case, each cluster found by the clustering algorithm is associated with a fuzzy rule and the multivariate Gaussian membership function parameters is defined as the parameters of the corresponding cluster. If multivariate Gaussian membership functions are used, the fuzzy classifier will have a rule set defined as

$$\text{RULE}_i : \text{IF } x_k \text{ IS } A_i \text{ THEN } y_i = L_i, \tag{22}$$

where A_i is the fuzzy set with multivariate Gaussian membership function (21) of the ith fuzzy rule, with parameters extracted from the corresponding cluster.

Usually, more than one rule can be used to describe a class; for example, the class can be multimodal. In this case, only one rule cannot be sufficient to describe all possible variations of the same class. Thus, the fuzzy classifier aggregates rules outputs associated with the same class using a s-norm. The result of the aggregation can be interpreted like rules as follows:

$$\begin{aligned}&\text{IF } x_k \text{ IS } A_i \text{ OR IF } x_k \text{ IS } A_j \text{ OR} \ldots \text{IF } x_k \text{ IS } A_k \\ &\text{THEN } y_i = L_i.\end{aligned} \tag{23}$$

```
Input: $x_k$;
Output: $y_k$;
Initialize the classifier;
for $k = 1, 2, \ldots$ do
    Read $x_k$;
    Execute the recursive clustering algorithm;
    if new cluster is created then
        Create new fuzzy rule;
        Define the new class elicited by expert/system operator;
        $y_k$ = label of the new class;
    end if
    if cluster is updated then
        Update the corresponding fuzzy rule;
        Find the most active rule;
        $y_k$ = label of the most active rule;
    end if
    if clusters are merged then
        Merge the corresponding fuzzy rules;
    end if
end for
```

Algorithm 2: Evolving fuzzy classifier.

The result of this aggregation is the degree of relevance of each known class. The classification of each new sample x_k is defined by the class with the highest relevance degree.

In some pattern classification applications data samples classes are not known a priori. In these situations, it is required the use of an unsupervised learning process for classifier implementation. Moreover, in applications where the pattern classification should be performed in real time, the learning should be performed using incremental algorithms, processing each data sample once as a data stream. To solve these problems, the solution is to use a recursive clustering algorithm.

In this paper, we propose an evolving fuzzy classifier based on recursive clustering algorithm with drift detection presented in Section 2.3, which allows the creation of a fuzzy rule base in online mode and, if necessary, in real time from input data samples. This approach is different from the ones employed in traditional fuzzy classifiers, which require some training (usually supervised) conducted in offline mode.

The proposed classifier updates the rule base using the output of the recursive clustering algorithm described in the previous section. For each new input data sample, if a new cluster is created, a new fuzzy rule (22) is added to the rule base, where the cluster center and the covariance matrix are used as parameters of the multivariable Gaussian membership function of the antecedents. The rule consequent (the crisp output corresponding to the class label) must be defined by experts or system operators, since in unsupervised learning processes incoming online samples usually are not prelabelled. If a cluster is updated, the corresponding fuzzy rules are updated, the class label is determined as the consequent of the fuzzy rule with the highest activation degree, and the user intervention is not necessary. If two clusters are merged by the recursive clustering algorithm, the corresponding fuzzy rules are also merged to represent a unique class. It should be noted that both the number of rules and the number of classes are determined during the evolving process, and it is not necessary to set these parameters a priori. Algorithm 2 summarizes the procedures of the classifier.

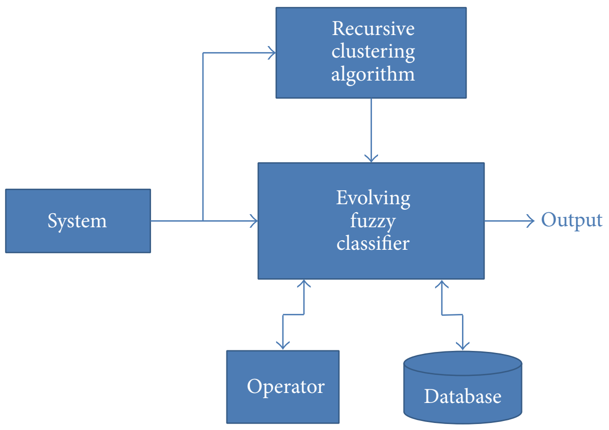

Figure 1: Fault diagnosis with an evolving fuzzy classifier.

The application of the proposed classifier for fault diagnosis is illustrated in Figure 1. Data samples are obtained from a dynamic system in a continuous stream, usually provided by sensors that monitor the process. These data might require the use of preprocessing techniques for feature extraction.

The classifier starts with an empty rule set. Rules are created as the recursive clustering algorithm creates clusters to represent the data stream. Each rule will be related to a class, and each class will be related to a dynamic system condition, representing a normal operation or a faulty condition. When a new rule is created, the system operator is notified and informs the label of the class that defines it as a normal operation or as a specific fault. All of the necessary diagnostic information, the fuzzy rules, and classes labels are stored in a unified database and updated while the system is used.

After an initial period of operation, the database will contain a set of fuzzy rules and classes labels defined so far. When a new data sample is associated with an existing cluster, the classifier updates the corresponding fuzzy rule and classifies the dynamic system condition as the label present in the consequent of the fuzzy rule with the highest activation degree. In this situation, system operator intervention is not required, and the classification of the dynamic system condition is performed automatically.

The classifier proposed in this work has as main characteristic the ability to diagnose faults in a complex nonstationary dynamic system. The classifier does not require any a priori information about the dynamic model neither process system historical data. This allows the classifier to construct a rule base in an evolving way and, with the aid of the operator, to learn to diagnose faults as they occur. Thus, the proposed classifier is able to adapt to the dynamic system, making it possible to diagnose faults not previously known.

4. Experiments and Results

The proposed classifier was evaluated for fault diagnosis in a DC drive system. A fault simulator was used in this evaluation, from which normal operation data and fault data were generated and organized in random sequences of different operation modes. The output of the classifier was compared with the provided sequence to prove its efficiency in detecting and classifying faults.

4.1. DC Drive System. The DC drive system model employed was proposed by [38] and consists of a benchmark for fault detection and diagnosis. As illustrated in Figure 2, the system comprises of two power supplies, two controlled static converters, a direct current machine and a mechanical load. The variables definitions shown in the representation of the system are as follows:

(i) v_a: voltage of the armature circuit;

(ii) v_{fd}: voltage of the field circuit;

(iii) i_a: current of the armature circuit;

(iv) i_{fd}: current of the field circuit;

(v) r_a, L_a: resistance/inductance of the armature circuit;

(vi) r_{fd}, L_{fd}: resistance/inductance of the field circuit;

(vii) e_a: counter-electromotive force of the armature;

(viii) T_{em}: electromagnetic torque;

(ix) T_L: torque required by the mechanical load.

Using this benchmark is possible to perform fault simulation on the actuators (armature and field converters), at the plant or process (machine and mechanical load), and on sensors (current and speed meters), as detailed in Table 1. To simulate a fault, was employed 750 V power supplies, constant speed and overload at 25% of nominal torque set at half of the simulation interval. A sampling period of 2 ms for the monitored variables was used.

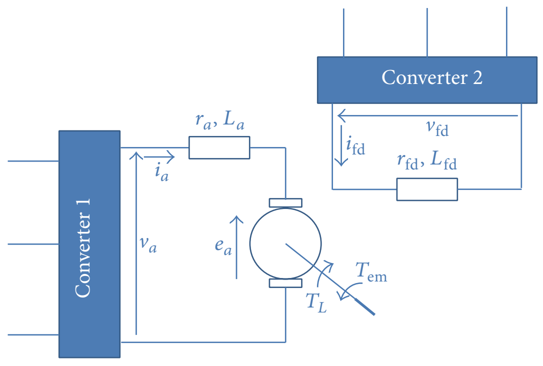

FIGURE 2: Representation of the DC drive system.

TABLE 1: Types of faults on DC drive system.

Index	Description
0	Normal operation
1	Armature converter disconnection
2	Field converter disconnection
3	Armature converter short circuit
4	Field converter short circuit
5	Armature turns short-circuit
6	Field turns short-circuit
7	Ventilation system fault
8	Bearing lubrication fault
9	Armature current sensor fault
10	Field current sensor fault
11	Machine speed sensor fault

Figures 3, 4, 5, 6, and 7 show as an example the curves of the armature current (I_a), field current (I_{fd}), and speed machine (V) in fault simulation. In this case, the following faults were simulated: armature converter disconnection, field converter short-circuit, armature turns short-circuit, bearing lubrication fault, and field current sensor fault. At the beginning of each simulation, the system is working under normal operation.

In Figure 8, the same faults presented in the previous figures are showed in three-dimensional space, where it is possible to observe that while some faults have abrupt behavior, others have an incipient behavior.

4.2. Fault Diagnosis. The fault diagnosis experiments were performed considering different scenarios. Each scenario consists in the simulation of sequences from 3 to 11 randomly selected fault types within a set of faults with periods of normal operation between faults. In order to assess the robustness of the proposed classifier to the presence of noise in the data, for each monitored variable random Gaussian noise was added with a zero mean and standard deviation equal to 2% of the variable nominal value, considering normal operation of the system.

Data samples related to monitored variables of the DC drive system, armature current (I_a), field current (I_{fd}), and speed machine (V) were provided as inputs of the classifier

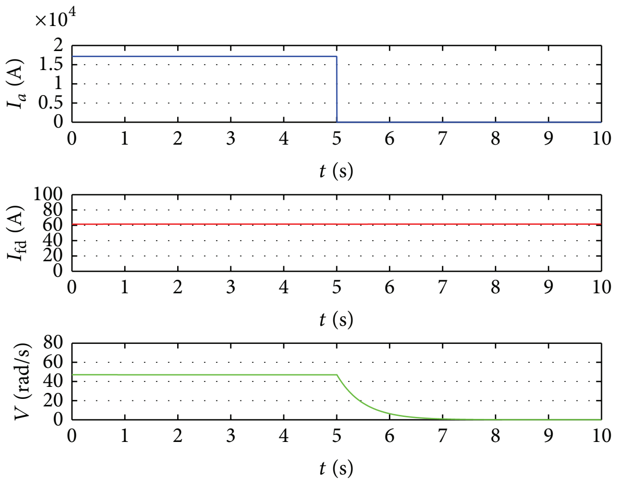

FIGURE 3: Fault simulation: armature converter disconnection.

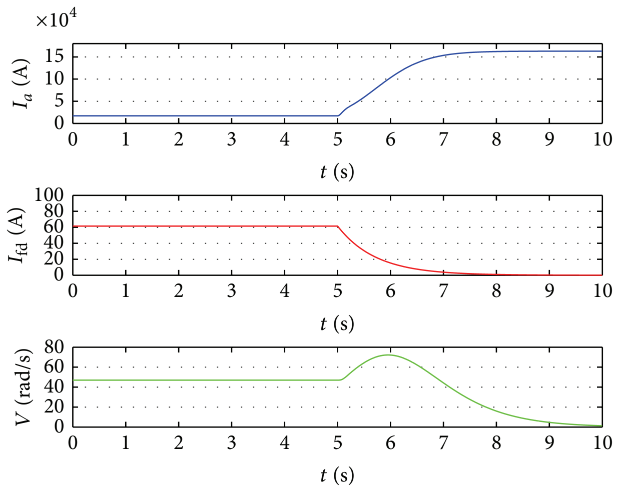

FIGURE 4: Fault simulation: field converter short-circuit.

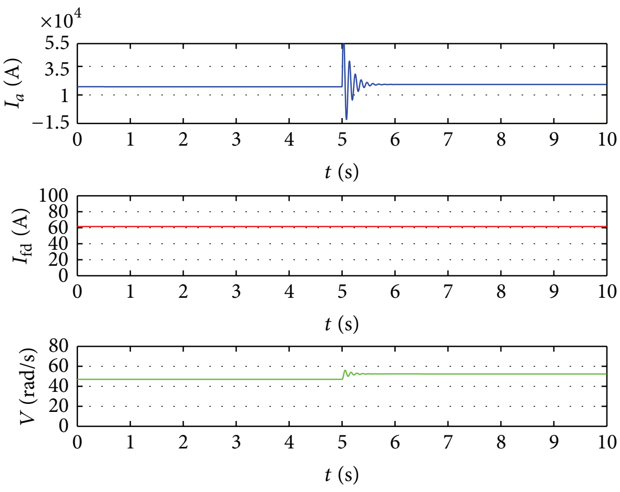

FIGURE 5: Fault simulation: armature turns short-circuit.

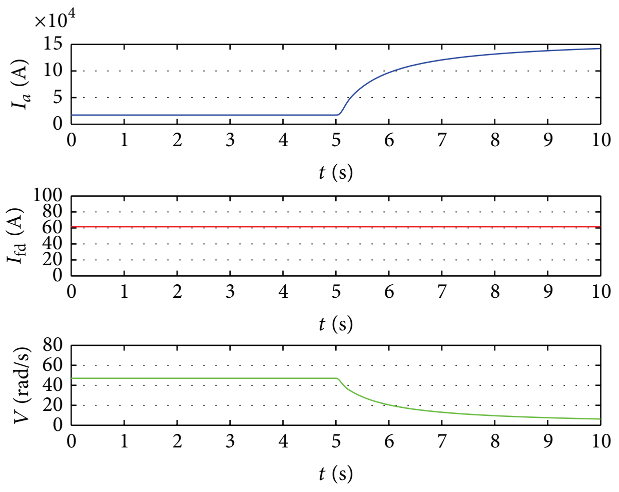

FIGURE 6: Fault simulation: bearing lubrication fault.

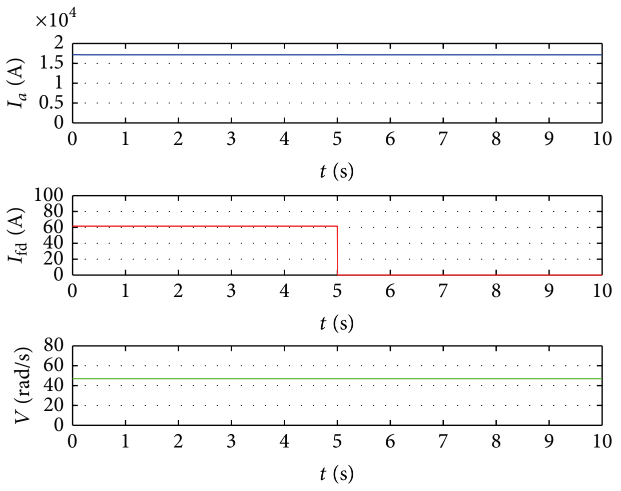

FIGURE 7: Fault simulation: field current sensor fault.

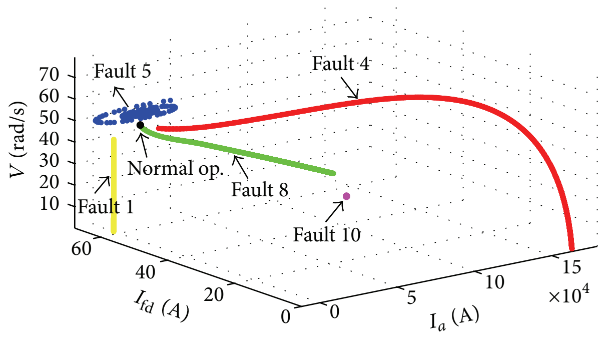

FIGURE 8: Fault simulation in 3D space.

in an online mode, and, in each sequence, the output classifier was compared to the sequence provided. Whereas the classifier starts with no fuzzy rule set, the first samples of data should match the normal operation of the system, that is, the first rule created to describe the normal operation. After that, during the diagnosis, faults are detected and new

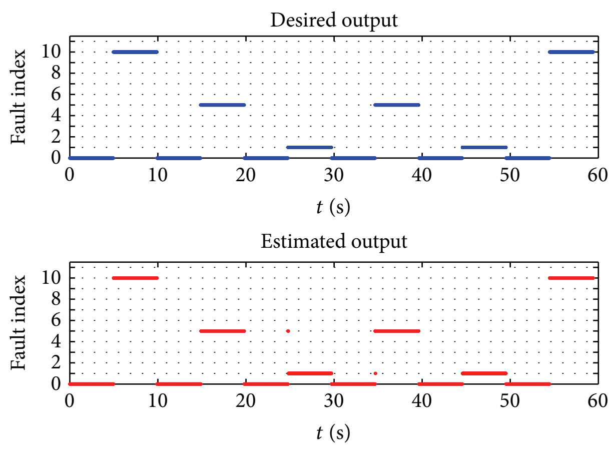

Figure 9: Desired output and estimated output by proposed classifier in a scenario of 3 faults.

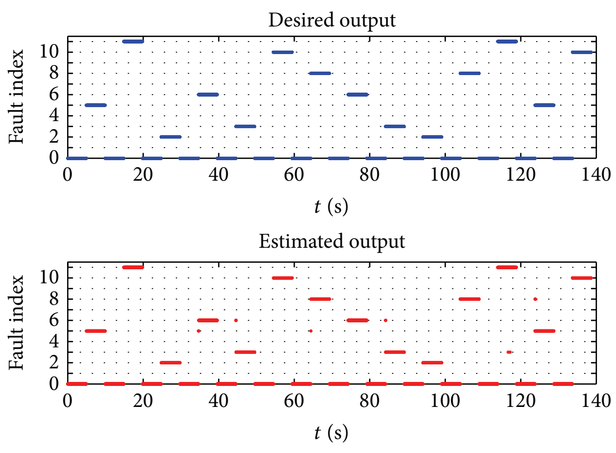

Figure 11: Desired output and estimated output by proposed classifier in a scenario of 7 faults.

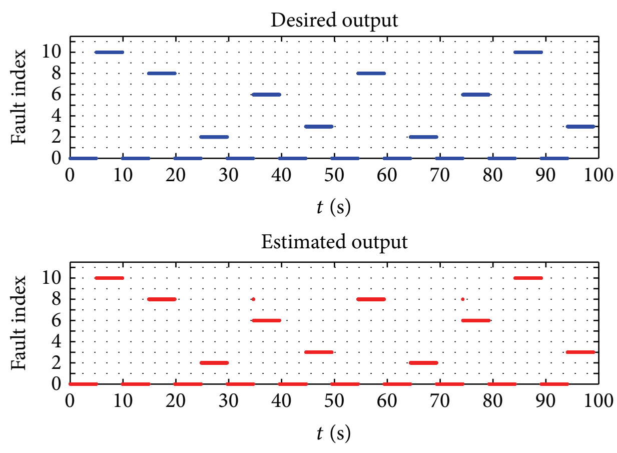

Figure 10: Desired output and estimated output by proposed classifier in a scenario of 5 faults.

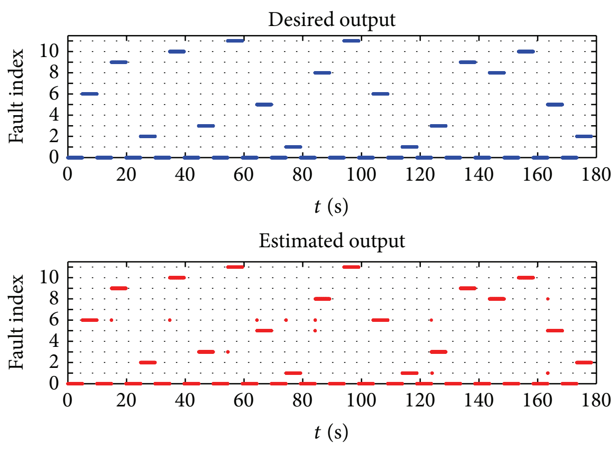

Figure 12: Desired output and estimated output by proposed classifier in a scenario of 9 faults.

rules that describe each type of fault are created. The label of each class, which defines it as normal operation or fault, is provided by the system operator at the time that each new rule is created. Following the first fault occurrence, the operator intervention is not needed anymore. For the experiments, the parameters of the recursive clustering algorithm were defined as follows: $\chi^2_{3,0.95} = 7,8147$; $F_{\text{init}} = 10^{-2}I$; $\alpha_{\text{init}} = 0.5$; $z_1 = 2$; $z_2 = 3$.

Figures 9, 10, 11, 12, and 13 show the results of fault diagnosis in each of the simulated scenarios, where we can compare the estimated output (classified faults sequence) of the proposed classifier with the desired output (selected faults sequence) from input data samples. Results show that the classifier was able to correctly diagnose all the DC drive system faults. Whereas the presence of noise in the data samples, the occurrence of false alarms or misclassification (represented by isolated points on the graphs) is significantly low, even in the scenario with the highest number of possible faults.

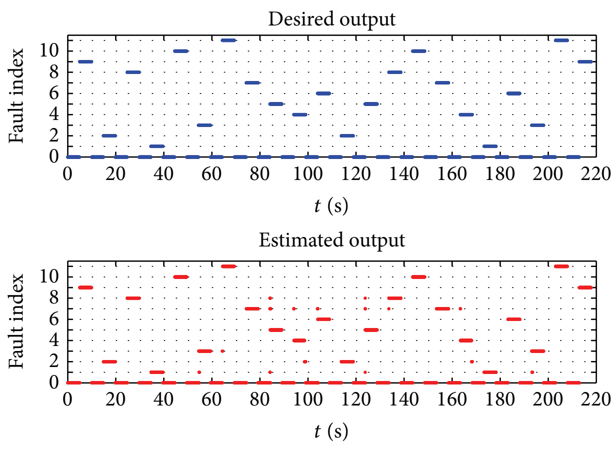

Figure 13: Desired output and estimated output by proposed classifier in a scenario of 11 faults.

Table 2: Fault detection performance.

Scenario	Proposed			Lemos et al. [10]		
	POD (%)	POFA (%)	ACC (%)	POD (%)	POFA (%)	ACC (%)
3 faults	99.85	0.00	99.89	99.79	0.00	99.85
5 faults	99.68	0.00	99.72	98.39	0.00	98.66
7 faults	99.79	0.00	99.89	98.50	0.00	98.68
9 faults	99.82	0.03	99.93	99.77	0.00	99.79
11 faults	99.33	0.29	99.32	93.73	0.00	94.26

Table 3: Fault classification performance.

Scenario	Proposed		Lemos et al. [10]	
	FIR (%)	OIR (%)	FIR (%)	OIR (%)
3 faults	99.75	0.05	99.77	0.11
5 faults	99.71	0.05	98.65	0.23
7 faults	99.76	0.03	98.39	0.19
9 faults	95.12	0.05	94.39	0.14
11 faults	94.39	0.05	90.85	0.38

In this work, the classifier performance evaluation was held in terms of fault detection and fault classification, as suggested in [3]. Three metrics were calculated in fault detection evaluation.

(i) Probability of detection (POD): it assesses the detected faults over all potential fault cases (sensitivity). Consider

$$\text{POD} = \frac{a}{a+c}. \tag{24}$$

(ii) Probability of false alarm (POFA): it considers the proportion of all fault-free cases that trigger a fault detection alarm. Consider

$$\text{POFA} = \frac{b}{b+d}. \tag{25}$$

(iii) Accuracy (ACC): it measures the effectiveness of the algorithm in correctly distinguishing between a fault-present condition and fault-free condition. Consider

$$\text{ACC} = \frac{a+d}{a+b+c+d}, \tag{26}$$

where a represents the number of detected faults; b represents the number of false alarms; c represents the number of missed faults; and d represents the number of correct rejections.

Regarding fault classification evaluation, the metric fault isolation rate (FIR) was used, expressing the percentage of all faults that the classifier is able to isolate unambiguously. This metric is computed as

$$\text{FIR} = \frac{A}{A+C}, \tag{27}$$

where A represents the total of detected and correctly classified faults and C represents the total of detected and incorrectly classified faults.

Other metrics that were used to assess the performance of the classifier are as follows.

(i) Detection delay time (DDT): it represents the time lag between the first occurrence of a given fault and its detection by the algorithm.

(ii) Isolation delay time (IDT): it represents the time lag between the second occurrence of a given fault and its classification by the algorithm.

(iii) Operator intervention rate (OIR): it is the percentage of faults classified with the intervention of the operator.

All results of fault diagnosis experiments with DC drive system obtained by classifier proposed in this work were compared to the results obtained using the classifier proposed by [10]. For the experiments, the parameters of this alternative classifier were set to $w = 100$, $\lambda = 0.001$, $\alpha = 0.01$, $T_{\mu_y} = 0.01$.

Table 2 summarizes the results for both classifiers using the fault detection metrics described. The results show that the classifier proposed in this work has higher levels of fault detection rates and accuracy in all scenarios, with values above 99% and low false alarm rates, with values below 0.3%. These results prove the efficiency of the algorithm in detecting simulated faults in the DC drive system. Despite its lower fault detection rates and lower accuracy, the classifier proposed by [10] did not show any false alarms.

Table 3 summarizes the results for both classifiers using the fault classification metrics described. The results show that the classifier proposed in this work presented higher fault isolation rate in all scenarios, with an average value of approximately 97%. In all scenarios, the operator intervention on fault classification was less than or equal to 0.05%. These results show the ability of the classifier to automatically diagnose almost all faults after the first occurrence, and it also reveals their ability to learn. Note that the classifier proposed by [10] in general had a lower performance in fault classification than the proposed classifier and it needed more operator interventions.

Table 4 summarizes the results for both classifiers using the time metrics in fault detection and classification. The average fault detection time found in the experiments with the classifier proposed in this work was approximately 0.060 s, which is primarily determined by the amount of data samples required to the recursive clustering algorithm to detect a context change. The average time to isolate faults found in the experiments was approximately 0.009 s. A comparison between the average values for fault detection

Table 4: Fault detection and classification time.

Scenario	Proposed		Lemos et al. [10]	
	DDT (s)	IDT (s)	DDT (s)	IDT (s)
3 faults	0.038	0.004	0.012	0.002
5 faults	0.047	0.003	0.047	0.005
7 faults	0.060	0.005	0.070	0.005
9 faults	0.068	0.008	0.012	0.006
11 faults	0.086	0.024	0.072	0.016

and fault isolation time demonstrates that fault classification is faster after the first occurrence of each type of fault, since the classifier database already has the fuzzy rules and labels for all types of detected faults, not requiring an operator intervention. The results of the experiments with the classifier proposed by [10] resulted in average values for fault detection and fault isolation time of 0.040 s and 0.007 s, respectively, demonstrating that their classifier has quicker response than the classifier proposed in this work, according to the different update mechanisms in the clustering algorithms used in each one of the classifiers.

Another experiment was conducted to evaluate the robustness of the proposed classifier to the presence of outliers in the data. In this experiment, a scenario of 5 faults was simulated. Outliers were inserted in the data samples; that is, some samples were corrupted with high variance noise. Figure 14 shows the fault simulation results in the presence of outliers in the three dimensions space. Figure 15 shows the results of fault diagnosis in this scenario.

The fault diagnosis results for this experiment shows that even in the presence of outliers the proposed classifier was able to correctly detect and diagnose all faults considered. This result shows that the classifier was able to correctly distinguish between outliers and valid data samples. The results of this experiment are presented in Tables 5 and 6. Analysing these tables, one can note that the proposed classifier has virtually the same performance in fault diagnosis with absence or presence of outliers, although we note an increase in false alarm rate. This experiment showed the greater robustness of the classifier proposed in this work when compared with the classifier proposed by [10], since the latter showed major differences in false alarm and fault isolation rates in scenarios with and without outliers.

5. Conclusions

In this work, we presented an evolving fuzzy classifier for fault diagnosis of complex nonstationary dynamic systems. The proposed classifier is composed by a set of fuzzy rules created and updated based on recursive unsupervised clustering algorithm. In this algorithm, a new mechanism for cluster updating based on a drift detection method is employed. With this mechanism, the update of the cluster depends not only on the similarity measure between data samples and clusters, but also on the data context monitoring. This feature gives the proposed classifier robustness to outliers and noise, as suggested by the experiment results. Multivariate

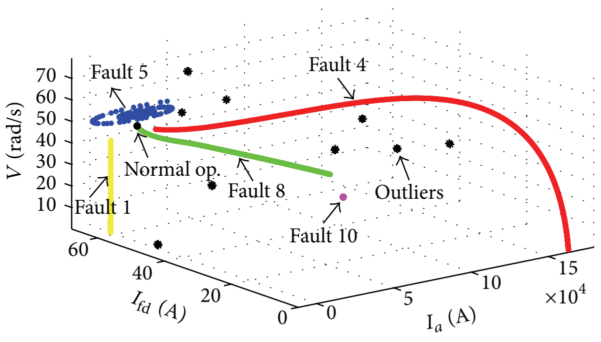

Figure 14: Fault simulation with outliers presence.

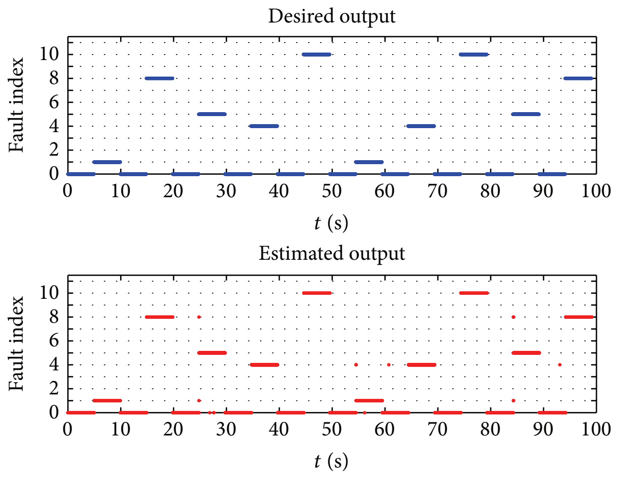

Figure 15: Desired output and estimated output by proposed classifier in a scenario of 5 faults with outliers presence.

Gaussian membership functions are used in fuzzy rules antecedents, whose parameters are extracted directly from clusters. This multivariate approach is used to eliminate the loss of information due to possible interactions between the input variables.

The classifier proposed in this work was evaluated in fault diagnosis experiments performed with a DC drive system model. The experiments showed that the classifier was able to detect and classify all faults with a high performance, even in the presence of outliers and noise. The low false alarm rate and high fault isolation rate obtained in all experiments showed that the recursive clustering algorithm with drift detection method was able to efficiently distinguish data samples representing clusters of invalid data. Moreover, the proposed classifier was able to automatically diagnose almost all faults, requiring operator intervention on a small percentage of cases. This demonstrates the advantage of the continuous and incremental learning of the classifier over other classifiers that require retraining whenever an unknown type of fault is found.

Considering the presented features, the classifier proposed in this work has as advantages the ability to learn from faults in online mode and in real time, the ability to adapt to cope with changes in the dynamic system, and

Table 5: Fault detection performance with outliers presence.

Scenario	Proposed			Lemos et al. [10]		
	POD (%)	POFA (%)	ACC (%)	POD (%)	POFA (%)	ACC (%)
Without outliers	99.69	0.00	99.85	99.70	0.08	99.74
With outliers	99.68	0.02	99.83	99.79	0.96	99.66

Table 6: Fault classification performance with outliers presence.

Scenario	Proposed		Lemos et al. [10]	
	FIR (%)	OIR (%)	FIR (%)	OIR (%)
Without outliers	99.75	0.03	99.32	0.21
With outliers	99.73	0.03	98.80	0.22

robustness to the presence of outliers and noise in the input data. Summarizing, the proposed classifier has showed to be a promising alternative for application in fault diagnosis in complex nonstationary dynamic systems, where other methods prove to be inefficient or less advantageous, because of the characteristics of such systems. In a future work, we will investigate the application of the proposed algorithm in a real time diagnosis and prognosis system of equipment.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the Brazilian National Research Council (CNPq) and the Research Foundation of the State of Minas Gerais (FAPEMIG).

References

[1] V. Venkatasubramanian, “Prognostic and diagnostic monitoring of complex systems for product lifecycle management: challenges and opportunities,” *Computers and Chemical Engineering*, vol. 29, no. 6, pp. 1253–1263, 2005.

[2] A. K. S. Jardine, D. Lin, and D. Banjevic, “A review on machinery diagnostics and prognostics implementing condition-based maintenance,” *Mechanical Systems and Signal Processing*, vol. 20, no. 7, pp. 1483–1510, 2006.

[3] G. Vachtsevanos, F. Lewis, M. Roeme, A. Hess, and B. Wu, *Intelligent Fault Diagnosis and Prognosis for Engineering Systems*, John Wiley & Sons, New York, NY, USA, 2006.

[4] J. V. Abellan-Nebot and F. Romero Subirón, “A review of machining monitoring systems based on artificial intelligence process models,” *International Journal of Advanced Manufacturing Technology*, vol. 47, no. 1–4, pp. 237–257, 2010.

[5] Y. Zhang, L. Zhang, and H. Zhang, “Fault detection for industrial processes,” *Mathematical Problems in Engineering*, vol. 2012, Article ID 757828, 18 pages, 2012.

[6] E. Lughofer and C. Guardiola, “Applying evolving fuzzy models with adaptive local error bars to on-line fault detection,” in *Proceedings of the 3rd International Workshop on Genetic and Evolving Systems (GEFS ’08)*, pp. 35–40, March 2008.

[7] D. Chivala, L. F. Mendonca, J. M. C. Sousa, and J. M. G. Sá da Costa, “Application of evolving fuzzy modeling to fault tolerant control,” *Evolving Systems*, vol. 1, no. 4, pp. 209–223, 2010.

[8] D. P. Filev, R. B. Chinnam, F. Tseng, and P. Baruah, “An industrial strength novelty detection framework for autonomous equipment monitoring and diagnostics,” *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 767–779, 2010.

[9] M. Petković, M. R. Rapaić, Z. D. Jeličić, and A. Pisano, “On-line adaptive clustering for process monitoring and fault detection,” *Expert Systems with Applications*, vol. 39, no. 11, pp. 10226–10235, 2012.

[10] A. Lemos, W. Caminhas, and F. Gomide, “Adaptive fault detection and diagnosis using an evolving fuzzy classifier,” *Information Sciences*, vol. 220, pp. 64–85, 2013.

[11] P. Angelov and N. Kasabov, “Evolving intelligent systems—eIS,” *IEEE SMC eNewsLetter*, vol. 15, pp. 1–13, 2006.

[12] N. Kasabov and D. Filev, “Evolving intelligent systems: methods, learning, & applications,” in *Proceeding of the International Symposium on Evolving Fuzzy Systems (EFS '06)*, pp. 8–18, September 2006.

[13] P. Angelov, D. Filev, and N. Kasabov, *Evolving Intelligent Systems: Methodology and Applications*, John Wiley & Sons, New York, NY, USA, 2010.

[14] N. K. Kasabov and Q. Song, “DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction,” *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 144–154, 2002.

[15] P. Angelov and D. Filev, “On-line design of Takagi-Sugeno models,” in *Proceedings of the 10th International Fuzzy Systems Association World Congress*, pp. 576–584, tur, July 2003.

[16] G. Leng, T. M. McGinnity, and G. Prasad, “An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network,” *Fuzzy Sets and Systems*, vol. 150, no. 2, pp. 211–243, 2005.

[17] H. Rong, N. Sundararajan, G. Huang, and P. Saratchandran, “Sequential Adaptive Fuzzy Inference System (SAFIS) for nonlinear system identification and prediction,” *Fuzzy Sets and Systems*, vol. 157, no. 9, pp. 1260–1275, 2006.

[18] E. D. Lughofer, “FLEXFIS: A robust incremental learning approach for evolving Takagi-Sugeno fuzzy models,” *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 6, pp. 1393–1410, 2008.

[19] H. Soleimani-B, C. Lucas, and B. N. Araabi, “Recursive Gath-Geva clustering as a basis for evolving neuro-fuzzy modeling,” *Evolving Systems*, vol. 1, no. 1, pp. 59–71, 2010.

[20] E. Lima, M. Hell, F. Gomide, and R. Ballini, *Evolving Fuzzy Modeling Using Participatory Learning*, John Wiley & Sons, New York, NY, USA, 2010.

[21] A. Lemos, W. Caminhas, and F. Gomide, “Multivariable gaussian evolving fuzzy modeling system,” *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 1, pp. 91–104, 2011.

[22] D. E. Gustafson and W. C. Kessel, “Fuzzy clustering with fuzzy covariance,” in *Proceedings of IEEE Conference on Decision and Control*, pp. 761–766, January 1979.

[23] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Lea rning with drift detection," in *Proceedings of 17th Brazilian Symposium on Artificial Intelligence*, pp. 286–295, 2004.

[24] D. Filev and O. Georgieva, "12. An extended version of the Gustafson-Kessel algorithm for evolving data stream clustering," in *Evolving Intelligent Systems: Methodology and Applications*, pp. 273–300, John Wiley and Sons, New York, NY, USA, 2010.

[25] R. Sebastiao and J. Gama, "A study on change detection methods," in *Proceedings of the 14th Portuguese Conference on Artificial Intelligence*, pp. 353–364, 2009.

[26] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, Wiley-Interscience, New York, NY, USA, 2nd edition, 2000.

[27] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[28] P. Angelov, "An approach for fuzzy rule-base adaptation using on-line clustering," *International Journal of Approximate Reasoning*, vol. 35, no. 3, pp. 275–289, 2004.

[29] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer, New York, NY, USA, 2nd edition, 2001.

[30] M. A. Maloof and R. S. Michalski, "Selecting examples for partial memory learning," *Machine Learning*, vol. 41, no. 1, pp. 27–52, 2000.

[31] R. Klinkenberg and T. Joachims, "Detecting concept drift with support vector machines," in *Proceedings of the 17th International Conference on Machine Learning (ICML '00)*, pp. 487–494, 2000.

[32] R. Klinkenberg, "Learning drifting concepts: example selection vs. example weighting," *Intelligent Data Analysis*, vol. 8, no. 3, pp. 281–300, 2004.

[33] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting change in data streams," in *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB '04)*, pp. 180–191, 2004.

[34] R. Sebastiao, J. Gama, P. P. Rodrigues, and J. Bernardes, "Monitoring incremental histogram distribution for change detection in data streams," in *Proceedings of the Workshop on Knowledge Discovery from Sensor Data (KDD '08)*, pp. 25–42, Las Vegas, NV, USA, August 2008.

[35] T. M. Mitchell, *Machine Learning*, McGraw-Hill, New York, NY, USA, 1997.

[36] P. M. Kelly, "An algorithm for merging hyperellipsoidal clusters," Tech. Rep. LA-UR-94-3306, Los Alamos National Laboratory, Los Alamos, NM, USA, 1994.

[37] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice-Hall, Upper Saddle River, NJ, USA, 1997.

[38] W. M. Caminhas and R. H. C. Takahashi, "Dynamic system failure detection and diagnosis employing sliding mode observers and fuzzy neural networks," in *Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, pp. 304–309, Vancouver, Canada, July 2001.