Chapter 4

Fault-Tolerant Early Classification Approach for Multivariate Time Series

The previous chapter presented an early classification approach for MTS which is generated by the sensors of different sampling rate. In some applications such as human activity monitoring, the sensors have equal sampling rate but they may be faulty or unreliable, thus generating the MTS with faulty data components. In this chapter, we focus on to solve the early classification problem for such MTS.

4.1 Introduction

With the ubiquity of cheap and small-size sensors, obtaining fine-grained information about any phenomenon became effortless. However, the correctness of the information is heavily rely on the reliability of the sensors. Such reliability can be validated by using the generated time series data [36]. A sensor is said to be faulty if it generates faulty data due to an error with microprocessor or low battery [77]. If a component of MTS is generated from such a faulty sensor then it may influence the accuracy of classifier.

In addition, one can get highly accurate classification results by increasing the number of sensors, especially in human activity classification system. However, in resource constrained environments such as smartphone and wearable (*e.g.,* Fitbit [78], Biostrap [79]), it is desirable to use fewer sensors for activity classification but with maintaining a desired level of accuracy. It suggests that some of the sensors can be omitted if they contribute marginally to the accuracy [37]. We also consider these sensors as faulty in this work, essentially they are irrelevant.

For example, the sensors in the human activity classification system generate an MTS corresponding to an activity. The early classification of such MTS refers to predicting a class label of activity before its complete execution, with a marginal compromise of accuracy. The faulty data components may be introduced in the MTS (corresponding to the ongoing activity) due to faulty sensors or unexpected motions. Such faulty components can mislead the classification process, which in turn will result in lower accuracy. In addition, the presence of faulty components is very common in IoT applications, but there exists no early classification approach that can handle these faulty data components of the MTS. Further, an MTS may also have some components which neither contribute to earliness nor to accuracy and therefore can also be removed during early classification.

In this chapter, we focus on early classification of MTS using a labeled MTS training dataset. We consider that a new MTS (to be classified) can have some of its components generated from faulty sensors. The problem is then two folds: first, identification of faulty components in the new MTS and second, prediction of the class label without waiting for complete MTS.

4.1.1 Motivation

There exists following major limitations in the existing work that motivated our work.

- *Fault-tolerant:* The existing approaches can not classify a new MTS correctly if any of its component is generated from a faulty sensor.

- *Correlation:* The existing work [23, 26, 32, 41] do not consider the correlation between the components of MTS while predicting the class label of a new MTS. This correlation can be quite informative for predicting the class label along the progress of MTS.
- *Earliness:* In real-time applications such as human activity classification, it is desirable to classify an MTS as early as possible with a desired level of accuracy. The existing work [8, 21] require complete MTS to classify the human activities and therefore do not provide any earliness.
- In addition, the existing work [21,29,80] use all components of MTS in classification even when some of them do not provide any identifiable information about a particular human activity. Such components increase the time complexity of the classification model and also degrade the user experience.

4.1.2 Major Contributions of the Work

This work addresses the early classification problem for MTS with faulty components. In particular, the major contributions of this chapter are as follows.

- This work proposes a **F**ault-tolerant **E**arly **C**lassification of **M**TS approach (FECM) to classify an incomplete MTS. By fault-tolerant we mean that FECM can correctly classify the MTS even if some of its components are generated from faulty sensors. FECM employs ARIMA model to identify the faulty component by learning error thresholds using training dataset.
- The FECM uses MTS that is generated by multiple sensors where the components have correlation among them. The FECM computes relevancy score of the component using k-means clustering to arrange them in non-increasing order of their relevancy scores. Such arrangement of components is used during estimation of MRL and classification of new incomplete MTS.

- Next, the FECM obtains a set of time series from a given MTS using Partial ordered set, which are sufficient for the desired level of accuracy of the classification. For providing the earliness during classification, FECM estimates the class discriminating MRLs using GP classifier and k-means clustering. The GP is probabilistic classifier which provides class probabilities against the MTS instances. We propose a method to classify a given MTS by using the estimated MRLs.
- Finally, we demonstrate an experiment for human activities classification and evaluate the performance of FECM using accuracy and earliness by using the created dataset and two publicly available datasets. We considered eight human activities during the experiment for creating the dataset. We also compared the FECM with three existing approaches [22, 25, 29].

The rest of the chapter is organized as follows. Next section defines the terminologies used in this work. Section 4.3 proposes the fault-tolerant early classification approach to classify the **MT**S with **F**aulty components (MTF). Section 4.4 evaluates the performance of the proposed approach on various human activity datasets and then compares the results with existing approaches. Finally, Section 4.5 concludes the chapter.

4.2 Preliminaries

This section defines the terminologies that are required to understand our work. Let $\mathbf{D} = \{\langle \mathbf{C}^1, \mathcal{L}^1\rangle, \langle \mathbf{C}^2, \mathcal{L}^2\rangle, \cdots, \langle \mathbf{C}^N, \mathcal{L}^N\rangle\}$ be an MTS training dataset, where $\mathbf{C}^j$ denotes j^{th} MTS and $\mathcal{L}^j$ denotes its class label, where $1 \leq j \leq N$. We assume that the dataset $\mathbf{D}$ has l different class labels, given as $\mathbf{L} = \{L_1, L_2, \cdots, L_l\}$. Each MTS in $\mathbf{D}$ has n time series and consists of M data points.

Definition 4.1 (Sub-dataset) *Let* $\mathbf{D}$ *be an MTS dataset which consists of* N *MTS and each MTS has* n *components. A sub-dataset of* $\mathbf{D}$*, denoted by* $\mathbf{D}_{\mathcal{C}}$*, is a dataset with all* N *MTS consisting of only* $\mathcal{C}$ *components, where* $\mathcal{C}$ *is a subset of components as*

$\mathcal{C} \subset \{\mathbf{C}_1, \mathbf{C}_2, \cdots, \mathbf{C}_n\}$. In addition, $\mathbf{D}_{-i}$ denotes a sub-dataset with all N MTS having n components excluding $\mathbf{C}_i$.

Definition 4.2 (Faulty component) *A component is said to be faulty if it is generated from a faulty sensor.*

Definition 4.3 (Partial ordered set) *A set $\mathcal{P}$ is said to be a Partial order set (Poset) [81] over an inclusion relation "$\subseteq$" if following axioms hold for any elements $x, y, z \in \mathcal{P}$:*

- ***reflexivity:*** *$x \subseteq x$.*
- ***antisymmetry:*** *if $x \subseteq y$ and $y \subseteq x$ then $x = y$.*
- ***transitivity:*** *if $x \subseteq y$ and $y \subseteq z$ then $x \subseteq z$.*

4.2.1 ARIMA model

ARIMA model has been widely used in the framework of time series forecasting [82,83]. We use it for identifying a faulty time series in the new MTS during testing. It is an Auto Regressive Moving Average (ARMA) model with d degree of differencing. Moreover, ARMA model itself is a combination of two models: AR(p) and MA(q). AR(p) model expresses an observation of a time series at time t as linear polynomial of past p observations and a random error ε_t. Unlike AR(p), MA(q) model considers past q prediction errors instead of the observations. The ARMA(p, q) model is mathematically expressed as

$$x_t = \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{j=1}^{q} \varphi_j \varepsilon_{t-j} + \varepsilon_t, \tag{4.1}$$

where, $\phi_1, \phi_2, \cdots, \phi_p$ and $\varphi_1, \varphi_2, \cdots, \varphi_q$ are the model parameters, which can be estimated from past data and $\varepsilon_t \in \mathcal{N}(\mu, \sigma^2)$. Equation 4.1 can also be written as $\phi(\mathbf{B})x_t = \varphi(\mathbf{B})\varepsilon_t$ for brevity, where $\mathbf{B}$ is a backshift operator as $\mathbf{B}x_t = x_{t-1}$. Adding differencing d into ARMA(p, q), we get the ARIMA(p, d, q) model as

$$\phi(\mathbf{B})(1-\mathbf{B})^d x_t = \varphi(\mathbf{B})\varepsilon_t, \tag{4.2}$$

where, $(1-\mathbf{B})^d$ denotes the d degree of differencing.

• **Parameter estimation:** ARIMA model uses Maximum Likelihood Estimator (MLE) to estimate the parameters from the past data (*i.e.*, training data). Let $\boldsymbol{x} = \{x_1, x_2, \cdots, x_T\}$ is a set of T observed data points and $\boldsymbol{\theta} = \{\phi_1, \phi_2, \cdots, \phi_p, \varphi_1, \varphi_2, \cdots, \varphi_q\}$ is a set of ARIMA model parameters to be estimated. The joint probability function of observing $\boldsymbol{x}$ is given as $f(x_T, x_{T-1}, \cdots, x_1; \boldsymbol{\theta})$. The MLE uses a log-likelihood function, which is defined as follows

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x}) = f(x_T, x_{T-1}, \cdots, x_1; \boldsymbol{\theta}). \tag{4.3}$$

To estimate $\boldsymbol{\theta}$ in parameter space $\boldsymbol{\Theta}$, the MLE is given as

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\operatorname{argmax}} \quad \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x}) \tag{4.4}$$

As $\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x})$ is continuous for all $\boldsymbol{\theta}$, we can obtain the maximum likelihood estimate $\hat{\boldsymbol{\theta}}$ by solving

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x})}{\partial \boldsymbol{\theta}} = 0. \tag{4.5}$$

Now, the parameters of ARIMA model can be estimated from the past data using Equation 4.5.

4.2.2 Kernel Density Estimation (KDE)

It is non-parametric method for data smoothing problem on a finite sample. KDE [65] estimates a probability density function of outcomes of a stochastic process. Let $X =$

$\{x_1, x_2, \cdots, x_n\}$ be random sample drawn from unknown probability density function $\hat{f}$. The kernel density of $\hat{f}$ at x^* can be estimated as

$$\hat{f}_h(X = x^*) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x^* - x_i}{h}\right), \tag{4.6}$$

where, $K(\cdot)$ is a kernel function and $h > 0$ is a smoothing parameter. We consider Gaussian kernel to approximate the sample X, where the optimal choice of h is $1.06\sigma n^{\frac{1}{5}}$ [84] and kernel function is given as

$$K\left(\frac{x^* - x_i}{h}\right) = \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-\frac{(x^* - x_i)^2}{2h^2}}. \tag{4.7}$$

Substituting kernel function from Equation 4.7 into Equation 4.6, we get x^* as a representative data point of X.

4.3 Fault-tolerant Early Classification of MTF

This section proposes a Fault-tolerant Early Classification of MTS approach to predict a class label of a new incomplete MTF . For a given MTS training dataset $\mathbf{D}$, we consider that all the MTS are complete and do not have any faulty component. However, a new MTS (*i.e.,* MTF), which is to be classified, can have faulty components.

4.3.1 Overview of FECM

The FECM consists of two major phases: learning and testing. In learning phase, FECM builds a set of classification models using the given training activity dataset in an offline manner. Each model learns class discriminating MRLs and error thresholds corresponding to each component of the MTS during this phase. Once the models are built, they utilize the learned MRLs to classify a new MTF (corresponding to an ongoing activity) before its complete execution in an online manner during testing phase. The testing phase first identifies the faulty components using learned error thresholds and then classifies the MTF by selecting a suitable classification model from the set. Figure 4.1 illustrates the block diagram of the FECM.

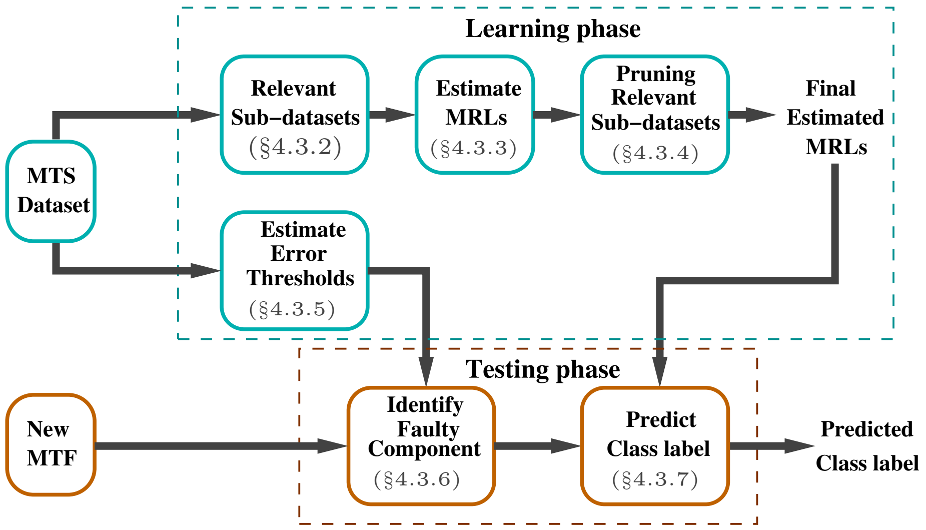

Learning phase
MTS Dataset
Relevant Sub–datasets (§4.3.2)
Estimate MRLs (§4.3.3)
Pruning Relevant Sub–datasets (§4.3.4)
Final Estimated MRLs
Estimate Error Thresholds (§4.3.5)
Testing phase
New MTF
Identify Faulty Component (§4.3.6)
Predict Class label (§4.3.7)
Predicted Class label

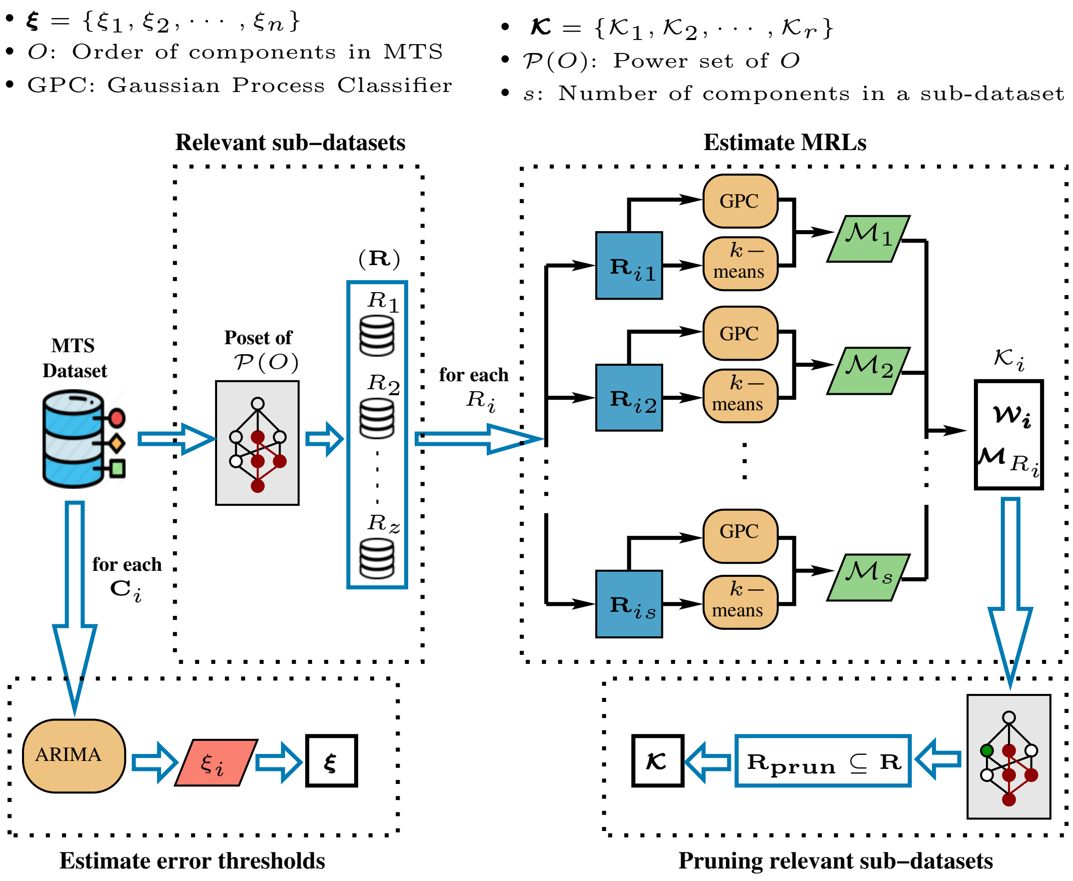
Figure 4.1: Block diagram of the FECM.

4.3.1.1 Learning phase

The main objective of this phase is to construct a set of classification models by estimating the class discriminating MRLs using GP classifier and k-means clustering. FECM first finds the relevant sub-datasets using a combination of components and then builds a separate classification model for each relevant sub-dataset. Learning phase consists of four major steps: obtaining relevant sub-datasets, estimating MRLs, pruning of relevant sub-datasets, and estimating error thresholds. The steps of learning phase are discussed in Sections 4.3.2, 4.3.3, 4.3.4, and 4.3.5 in detail. Figure 4.2 illustrates an overview of learning phase of FECM. Further, Algorithm 4.1 illustrates all steps of learning phase of the FECM. The input and output of this phase are as follows:

Input: *An MTS training dataset* $\mathbf{D} = \{\langle \mathbf{C}^j, \mathcal{L}^j \rangle : 1 \leq j \leq N\}$ *that consists of* N *labeled MTS with a class label* $\mathcal{L}^j \in \mathbf{L}$, *where* $\mathbf{L}$ *is a set of all class labels. Each MTS has* n *components, i.e.,* $\mathbf{C}^j = \{\mathbf{C}^j_1, \mathbf{C}^j_2, \cdots, \mathbf{C}^j_n\}$.

Output: *A set of classification models* $\boldsymbol{\mathcal{K}} = \{\mathcal{K}_1, \mathcal{K}_2, \cdots, \mathcal{K}_r\}$*, where each model* $\mathcal{K}_i$ *is trained using a relevant sub-dataset for* $1 \leq i \leq r$ *and a set of error thresholds corresponding to* n *components, i.e.,* $\boldsymbol{\xi} = \{\xi_1, \xi_2, \cdots, \xi_n\}$.



• $\boldsymbol{\xi} = \{\xi_1, \xi_2, \cdots, \xi_n\}$
• O: Order of components in MTS
• GPC: Gaussian Process Classifier
• $\boldsymbol{\mathcal{K}} = \{\mathcal{K}_1, \mathcal{K}_2, \cdots, \mathcal{K}_r\}$
• $\mathcal{P}(O)$: Power set of O
• s: Number of components in a sub-dataset
Relevant sub–datasets
Estimate MRLs
MTS Dataset
Poset of $\mathcal{P}(O)$
$(\mathbf{R})$
R_1
R_2
R_z
for each R_i
$\mathbf{R}_{i1}$
$\mathbf{R}_{i2}$
$\mathbf{R}_{is}$
GPC
$k-$ means
$\mathcal{M}_1$
$\mathcal{M}_2$
$\mathcal{M}_s$
$\mathcal{K}_i$
$\boldsymbol{w_i}$
$\boldsymbol{\mathcal{M}}_{R_i}$
for each $\mathbf{C}_i$
ARIMA
ξ_i
$\boldsymbol{\xi}$
$\boldsymbol{\mathcal{K}}$
$\mathbf{R_{prun}} \subseteq \mathbf{R}$
Estimate error thresholds
Pruning relevant sub–datasets

Figure 4.2: Overview of learning phase of the FECM.

4.3.1.2 Testing phase

This phase predicts a class label of a new MTF while maintaining a desired level of accuracy ($\boldsymbol{\alpha}$). It consists of two steps: identifying faulty components in the MTF and class label prediction. The FECM first identifies the faulty components in the MTF and removes them from the MTF. It then selects one of the constructed model from $\boldsymbol{\mathcal{K}}$ to classify the new MTF using the estimated MRLs. The selection of an appropriate classifier depends on the number of relevant components present in the new MTF.

As FECM uses only one classification model at a time, it can not take advantage of ensemble learning where multiple models are combined to improve the performance. The steps of testing phase are described in Sections 4.3.6 and 4.3.7. Algorithm 4.2 illustrates all steps of testing phase of the FECM.

4.3.2 Obtaining relevant sub-datasets using Poset

This section obtains the relevant sub-datasets of a given dataset $\mathbf{D}$ using Poset. For the dataset $\mathbf{D}$, there exists total 2^n possible sub-datasets. If n is large, there will be large number of sub-datasets. Not all of these sub-datasets can provide the desired level of accuracy $\boldsymbol{\alpha}$ even by using complete MTS. FECM first computes the relevancy score of the components of the MTS and then arrange them in non-increasing order of their relevancy scores. The FECM employs k-means clustering to compute relevancy score of the components. The relevancy score of a component $\mathbf{C}_i$ can be computed as

$$\mathcal{S}(\mathbf{C}_i) = \mathcal{A}_{\mathbf{D}} - \mathcal{A}_{\mathbf{D}_{-i}}, \tag{4.8}$$

where, $\mathcal{A}_{\mathbf{D}}$ and $\mathcal{A}_{\mathbf{D}_{-i}}$ are the accuracy that can be obtained by applying k-means clustering on $\mathbf{D}$ and $\mathbf{D}_{-i}$ datasets, respectively, and k is equal to the total number of class labels in the respective dataset. The FECM uses correlation based similarity measure to find the nearest cluster. The similarity between any two MTS $X, Y \in \mathbf{D}$, is given as

$$Sim(X,Y) = \frac{1}{n}\sum_{i=1}^{n} \frac{\sum\limits_{k=1}^{M}(x_i^{(k)} - \mu_{X_i})(y_i^{(k)} - \mu_{Y_i})}{\sqrt{\sum\limits_{k=1}^{M}(x_i^{(k)} - \mu_{X_i})^2 \sum\limits_{k=1}^{M}(y_i^{(k)} - \mu_{Y_i})^2}},$$

where, μ and M denote expected mean and length of MTS, respectively. Now, the order of components in the MTS is given as $O = \{1, 2, \cdots, n\}$. For any two elements $a, b \in O$

corresponding to components $\mathbf{C}_a$ and $\mathbf{C}_b$, respectively, if $a > b$ then $\mathcal{S}(\mathbf{C}_a) \leq \mathcal{S}(\mathbf{C}_b)$.

With the help of relevancy scores, the FECM constructs a Hasse diagram of a Poset of all possible subsets (*i.e.,* 2^n), where each node corresponds to a sub-dataset (*i.e.,* a combination of components). In the Hasse diagram, FECM evaluates each node for its relevancy and assign a color accordingly, *i.e.,* 'white' if node is relevant and 'red' otherwise. A node is said to be relevant with $\mathcal{C}$ components, if $\mathcal{A}_{\mathbf{D}_\mathcal{C}} \geq \boldsymbol{\alpha}\mathcal{A}_{\mathbf{D}}$ holds, where $\boldsymbol{\alpha}$ is desired level of accuracy. Procedure 1 illustrates the steps for assigning a color to the nodes in the Hasse diagram (from top to bottom).

Procedure 1: Color_node()

```
1.  Initially assign 'white' color to all nodes.
2.  Compute accuracy $\mathcal{A}_{\mathbf{D}}$.
3.  For each node (set of $\mathcal{C}$ components):
4.      If node is 'white'
5.          Compute accuracy $\mathcal{A}_{\mathbf{D}_\mathcal{C}}$.
6.          If $\mathcal{A}_{\mathbf{D}_\mathcal{C}} \geq \boldsymbol{\alpha}\mathcal{A}_{\mathbf{D}}$:
7.              No color change.
8.          Else:
9.              Assign 'red' to node and its descendants.
```

After coloring, the sub-datasets that correspond to 'white' nodes, are called as relevant sub-datasets. Let $\mathbf{R}$ denotes a set of all relevant sub-datasets obtained after coloring, *i.e.,* $\mathbf{R} = \{R_1, R_2, \cdots, R_z\}$, where $z \ll 2^n$. The FECM builds a classification model for each of the z relevant sub-datasets.

Example: Let a dataset has four components ($\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3$, and $\mathbf{C}_4$) represented as $O = \{1, 2, 3, 4\}$. Hasse diagram for all possible subsets of O over an inclusion relation "$\subseteq$" is shown in part (a) of Figure 4.3. Initially, all nodes are of 'white' color. After coloring, the nodes (representing relevant sub-datasets) remain 'white' and others become 'red' as shown in Part (b) of Figure 4.3. This step is shown at Lines $1 - 5$ in Algorithm 4.1.

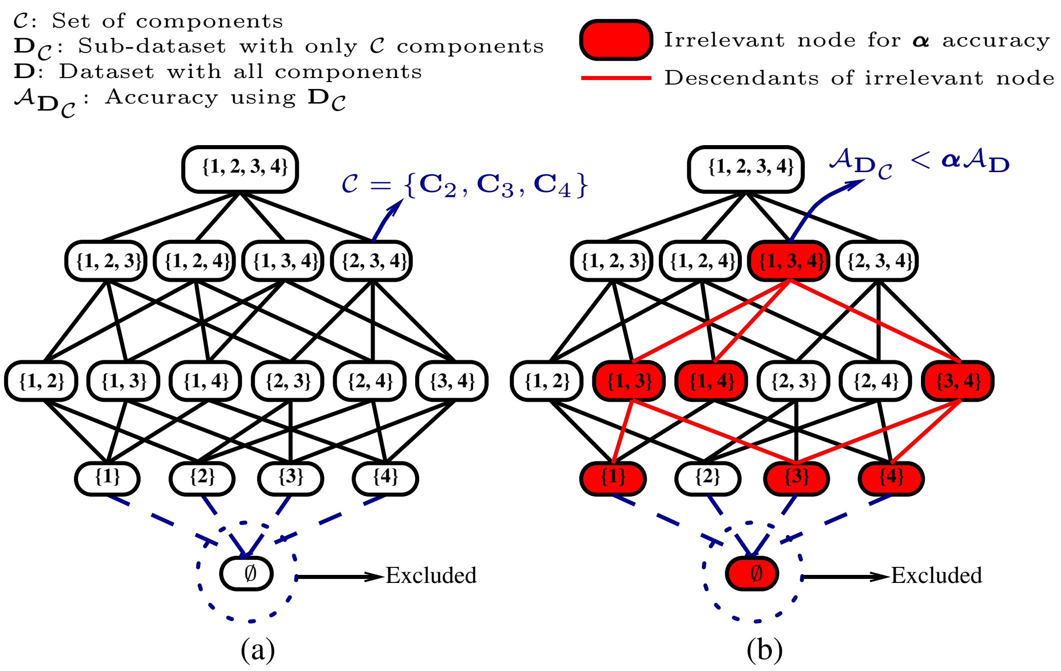

$\mathcal{C}$: Set of components
$\mathbf{D}_{\mathcal{C}}$: Sub-dataset with only $\mathcal{C}$ components
$\mathbf{D}$: Dataset with all components
$\mathcal{A}_{\mathbf{D}_{\mathcal{C}}}$: Accuracy using $\mathbf{D}_{\mathcal{C}}$
Irrelevant node for $\boldsymbol{\alpha}$ accuracy
Descendants of irrelevant node
$\mathcal{C} = \{\mathbf{C}_2, \mathbf{C}_3, \mathbf{C}_4\}$
$\mathcal{A}_{\mathbf{D}_{\mathcal{C}}} < \boldsymbol{\alpha}\mathcal{A}_{\mathbf{D}}$
Excluded
(a)
(b)

Figure 4.3: Illustration of Hasse diagram for a power set. Part (a) illustrates an example of Hasse diagram for a power set of $O = \{1, 2, 3, 4\}$. Part (b) shows the coloring of the Hasse diagram based on relevancy of nodes.

4.3.3 Estimating class discriminating MRL

This step uses GP [42] classifier and k-means clustering to compute class discriminating MRLs for the given dataset. Let $\mathbf{S} \in \mathbf{R}$ be a relevant dataset with s components and $\mathbf{S}_i$ is sub-dataset of $\mathbf{S}$ with only $\mathbf{C}_i$ component, where $1 \leq i \leq s$. The FECM first computes the class discriminating MRLs for each component separately and later it computes MRLs for the MTS by utilizing the correlation between the components.

4.3.3.1 Computation of MRL for an individual component

Let $X_f \in \mathbf{S}_i$ denotes a time series with f data points and f equals to M for a complete time series. The objective of an early classifier is to learn a mapping $X_f \rightarrow \mathcal{L}$ using given dataset, where $\mathcal{L}$ is ground truth class label and $f(\leq M)$ is the MRL of time series X. For each time series $X \in \mathbf{S}_i$, we perform following steps to obtain the MRL:
(a) The FECM uses k-means clustering to obtain the compact groups of similar time

series using full length (M), where k denotes the number of classes in the dataset, *i.e.,* $k = l$. Let a cluster corresponding to L_q class label is denoted by $\mathcal{G}_q$ and the confusion matrix of $l \times l$, obtained after clustering, is denoted by $\mathcal{B}$. A prior probability of class label L_q by using $\mathcal{B}$ is given as

$$p(\mathcal{G}_q) = \mathcal{B}[q][q]. \tag{4.9}$$

The probability that the time series X_f belongs to cluster $\mathcal{G}_q$, is given as

$$p(\mathcal{G}_q|X_f) = \frac{\delta_q}{\sum_{k=1}^{l} \delta_k}, \tag{4.10}$$

where, $\delta_q = \bar{\mathcal{G}} - d_q$ and $\bar{\mathcal{G}}$ is an average of distances from X_f to all the l clusters and d_q is the distance from X_f to cluster $\mathcal{G}_q$, and is given as

$$\bar{\mathcal{G}} = \frac{1}{l}\sum_{q=1}^{l} \text{dist}(X_f, \mathcal{G}_l) \quad \text{and} \quad d_q = \text{dist}(X_f, \mathcal{G}_q).$$

The function $\text{dist}(X_f, \mathcal{G}_q)$ uses only first f data points to compute the similarity. We consider euclidean distance as similarity measure but any other distance metric such as dynamic time warping [85], can be easily incorporated here.

(b) In this step, FECM uses GP classifier to model a time series as outcome of stochastic process and to obtain class posterior probabilities using given dataset $\mathbf{S}_i$. Let h_f is a GP classifier such that $h_f : \mathbb{R}^f \rightarrow \mathcal{L}$. The posterior probability of class L_q for a time series X_f can be obtained using Baye's rule as follows

$$p(L_q|X_f) = \prod_{k=1}^{f} \frac{p(L_q) \cdot p(x_f^{(k)}|L_q)}{p(x_f^{(k)})}, \tag{4.11}$$

where, $p(x_f^{(k)})$ and $p(L_q)$ are marginal and prior probabilities that can be estimated using $\mathbf{S}_i$, respectively. Further, the likelihood term $p(x_f^{(k)}|L_q)$ is given as

$$p(x_f^{(k)}|L_q) = \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-\frac{(x_f^{(k)}-\mu)^2}{2\sigma^2}}. \tag{4.12}$$

(c) We define a measure $\mathcal{F}_f$ to estimate the relative accuracy that can be obtained using f data points of the time series instead of using all M data points. Using Equations 4.9, 4.10, and 4.11, we define the relative accuracy of a time series X_f belonging to ground truth class label L_q as

$$\mathcal{F}_f = \frac{p(L_q|X_f) \cdot p(\mathcal{G}_q|X_f)}{\boldsymbol{\alpha} \cdot p(\mathcal{G}_q)}, \tag{4.13}$$

where, $\boldsymbol{\alpha}$ is a desired level of accuracy given by user. The utility function $\mathcal{U}(\cdot)$ that optimizes a tradeoff between relative accuracy $\mathcal{F}_f$ and earliness $\mathcal{E}$, is given by

$$\mathcal{U}(X_f) = \frac{2 \times \mathcal{F}_f \times \mathcal{E}}{\mathcal{F}_f + \mathcal{E}}, \qquad \text{where } \mathcal{E} = \frac{M-f}{M}. \tag{4.14}$$

At this point, we can compute the MRL of a time series $X \in \mathbf{S}_i$ as given in following expression

$$\mathrm{MRL} = \underset{f}{\operatorname{argmax}} \; \{\mathcal{U}(X_f)\}. \tag{4.15}$$

The FECM computes a representative MRL for each class label L_q using obtained MRLs (using Equation 4.15), where $1 \leq q \leq l$. For a dataset $\mathbf{S}_i$, let N_q is number of time series that has class label L_q. Using Equation 4.15, the vector of learned MRLs for the class label L_q is given by $\mathcal{V}_q = \{v_1, v_2, \cdots, v_{N_q}\}$. The FECM uses KDE [65] to estimate a class representative MRL by considering $\mathcal{V}_q$ as a random sample. Using Equation 4.6, we get v^* as a representative MRL for L_q class label. Next, FECM stores the representative MRLs for $\mathbf{S}_i$ dataset into a vector $\mathcal{M}_i = \{f_{i,1}, f_{i,2}, \cdots, f_{i,l}\}$. Finally, the class discriminating MRLs for the dataset $\mathbf{S}$, are given as $\boldsymbol{\mathcal{W}} = \{\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_s\}$. This step is illustrated at Lines $6 - 24$ in Algorithm 4.1.

4.3.3.2 Computation of MRL for MTS with all components

In this step, FECM computes the MRL for MTS considering all the components at once by utilizing the correlation between the components. Let $\mathbf{C}$ be an MTS with s components, *i.e.*, $\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \cdots, \mathbf{C}_s\}$. For each MTS $\mathbf{C} \in \mathbf{S}$ with true class label $\mathcal{L}$, FECM performs following steps to compute MRL:

(a) FECM starts prediction from component $\mathbf{C}_1$ by considering f data points (*i.e.*, estimated MRL obtained using $\mathcal{M}_1$ for $\mathcal{L}$ class label).

(b) For $i = 1$ to s:
   - GP classifier h_f, which is trained over dataset $\mathbf{S}_i$ using first f data points, predicts the class label $\hat{L}_i$.
   - Forward $\hat{L}_i$ to next component $\mathbf{C}_{i+1}$.
   - Compute minimum required data points to predict class label $\hat{L}_i$ using $\mathbf{C}_{i+1}$ as $f = \mathcal{M}_{i+1}[\hat{L}_i]$.

(c) Repeat Step (b) till predicted class label $\hat{L}_i$ is not obtained for all the components, *i.e.*, $1 \leq i \leq s$.

(d) The MRL of $\mathbf{C}$ is obtained using only those components whose predicted class label is $\mathcal{L}$ and number of data points used to predict $\mathcal{L}$ becomes the MRL of $\mathbf{C}$.

Figure 4.4 illustrates the computation of class discriminating MRLs for the MTS using all components at once. Such MRLs for the dataset $\mathbf{S}$ are obtained as $\boldsymbol{\mathcal{M}}_{\mathbf{S}} = \{f_1, f_2, \cdots, f_l\}$ using Equation 4.6 and accuracy $\mathcal{A}_{\mathbf{S}}$.

4.3.4 Pruning of relevant sub-datasets using $\boldsymbol{\mathcal{M}}_{\mathbf{S}}$ and $\mathcal{A}_{\mathbf{S}}$

There may exist multiple suitable groups of components (*i.e.,* relevant sub-datasets) that can provide the desired level of accuracy $\boldsymbol{\alpha}$. The FECM therefore builds a Hasse

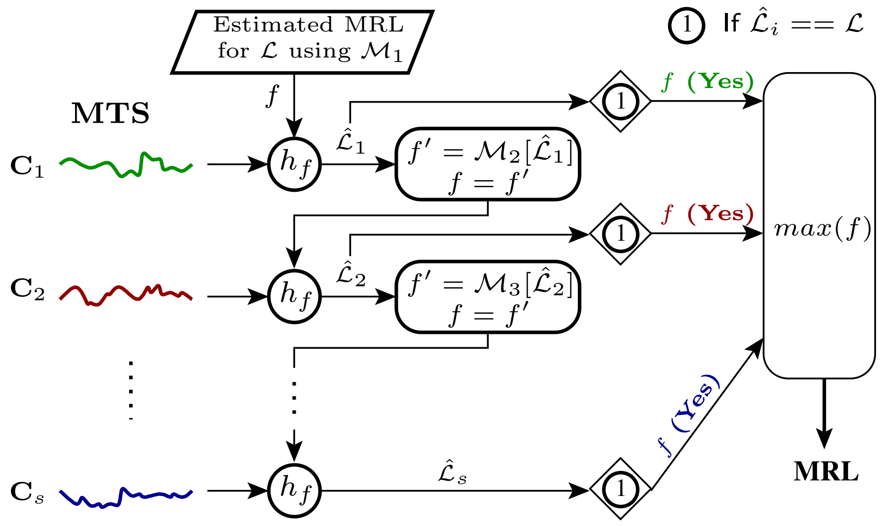

Estimated MRL for $\mathcal{L}$ using $\mathcal{M}_1$
1 If $\hat{\mathcal{L}}_i == \mathcal{L}$
MTS
f
C_1
h_f
$\hat{\mathcal{L}}_1$
$f' = \mathcal{M}_2[\hat{\mathcal{L}}_1]$ $f = f'$
f (Yes)
C_2
$\hat{\mathcal{L}}_2$
$f' = \mathcal{M}_3[\hat{\mathcal{L}}_2]$ $f = f'$
f (Yes)
$max(f)$
C_s
$\hat{\mathcal{L}}_s$
f (Yes)
MRL

Figure 4.4: Computation of class discriminating MRLs for the MTS.

diagram of those suitable groups and selects the most suitable group of components using the Hasse diagram. The most relevant sub-dataset is obtained by pruning the other sub-datasets based on accuracy and earliness. In part (a) of Figure 4.5, let $\mathbf{S}$ and $\mathbf{S}'$ are two relevant sub-datasets corresponding to nodes in Hasse diagram and $\mathbf{S}$ is parent node of $\mathbf{S}'$. A sub-dataset $\mathbf{S}$ can be pruned if avg($\boldsymbol{\mathcal{M}}_{\mathbf{S}}{}'$) $<$ avg($\boldsymbol{\mathcal{M}}_{\mathbf{S}}$) and $\mathcal{A}_{\mathbf{S}}{}' \geq \mathcal{A}_{\mathbf{S}}$, as depicted by 'green' color in Hasse diagram in part (b) of Figure 4.5. As the number of components in $\mathbf{S}$ is more than $\mathbf{S}'$, pruning $\mathbf{S}$ can save many computations during the classification of a new MTS. This pruning process is carried out for all the nodes, which eventually provides the most relevant sub-dataset.

Let $\mathbf{R}_{\text{prun}}(\subseteq \mathbf{R})$ denotes the set of remaining relevant sub-datasets, given as $\mathbf{R}_{\text{prun}} = \{R_1, R_2, \cdots, R_r\}$, where $r(\leq z)$ denotes the number of relevant sub-datasets in $\mathbf{R}_{\text{prun}}$. For each relevant sub-dataset $R_i \in \mathbf{R}_{\text{prun}}$, FECM builds a classification model $\mathcal{K}_i = \{\boldsymbol{\mathcal{W}}_i, \mathcal{M}_{R_i}\}$. Further, for whole training dataset $\mathbf{D}$, the set of classification models is given as $\boldsymbol{\mathcal{K}} = \{\mathcal{K}_1, \mathcal{K}_2, \cdots, \mathcal{K}_r\}$. This step is shown in Algorithm 4.1.

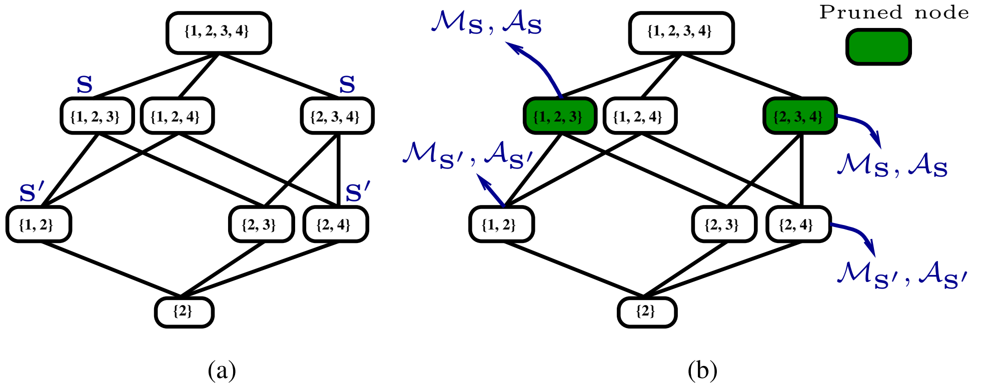

{1, 2, 3, 4}
S
S
{1, 2, 3}
{1, 2, 4}
{2, 3, 4}
S′
S′
{1, 2}
{2, 3}
{2, 4}
{2}
(a)
$\mathcal{M}_{\mathbf{S}}, \mathcal{A}_{\mathbf{S}}$
Pruned node
$\mathcal{M}_{\mathbf{S}'}, \mathcal{A}_{\mathbf{S}'}$
$\mathcal{M}_{\mathbf{S}}, \mathcal{A}_{\mathbf{S}}$
$\mathcal{M}_{\mathbf{S}'}, \mathcal{A}_{\mathbf{S}'}$
(b)

Figure 4.5: Pruning of relevant sub-datasets using accuracy and earliness. Part (a) shows a Hasse diagram obtained from part (b) of Figure 4.3 after removing 'red' color nodes. Part (b) illustrates pruning of nodes using accuracy and earliness.

4.3.5 Estimating error threshold

The objective of this step is to estimate an error threshold ξ corresponding to each component of MTS using training dataset $\mathbf{D}$. Such error threshold can be estimated by modeling the time series using ARIMA model [83]. The maximum permissible error threshold ξ_i in component $\mathbf{C}_i$ is given as

$$\xi_i = \max_{1 \leq j \leq N, 1 \leq k \leq M} \{\epsilon_{jk}\}, \tag{4.16}$$

where, ϵ_{jk} denotes an error that is the difference between k^{th} actual and predicted observations of j^{th} time series using ARIMA model. In this way, the FECM can compute the error thresholds for whole dataset $\mathbf{D}$ as $\boldsymbol{\xi} = \{\xi_1, \xi_2, \cdots, \xi_n\}$. This step is shown at Lines $29 - 31$ in Algorithm 4.1.

4.3.6 Identifying faulty component

In this step, FECM uses the estimated error thresholds $\boldsymbol{\xi}$ to identify the faulty components of a new MTF. Let $\mathbf{C}^p = \{\mathbf{C}_1^p, \mathbf{C}_2^p, \cdots, \mathbf{C}_n^p\}$ be a new MTF. The FECM considers the learned MRL $\mathcal{M}_1$ to know the right time (*i.e.,* sufficient data points) to start the prediction of class label. In addition, it also helps to avoid premature or needless predic-

Algorithm 4.1: Learning phase

```
Input: A labeled dataset $\mathbf{D}$ of $N$ MTS with $l$ labels. Each MTS has $n$ complete components
       of length $M$ and a class label $\mathcal{L} \in \mathbf{L}$;
Output: A set of classification models $\boldsymbol{\mathcal{K}} = \{\mathcal{K}_1, \mathcal{K}_2, \cdots, \mathcal{K}_r\}$;
/* Obtaining relevant sub-datasets */
1  for $i \leftarrow 1$ to $n$ do
2      Compute relevancy score $\mathcal{S}(\mathbf{C}_i)$ using Equation 4.8.
3  Arrange components in non-increasing order of relevancy score.
4  Construct a Hasse diagram of set of components using $\subseteq$.
5  Obtain relevant sub-datasets using Procedure 1 as $\mathbf{R} = \{R_1, R_2, \cdots, R_z\}$
/* Estimating MRL and pruning of relevant sub-datasets */
6  for $i \leftarrow 1$ to $z$ do
       /* Number of components in $R_i$ is $s$ */
7      for $j \leftarrow 1$ to $s$ do
8          Obtain confusion matrix $\mathcal{B}$ using $k$-means clustering.
           /* $\mathcal{G}_q$ is a group corresponding to class $L_q \in \mathbf{L}$ */
9          Compute $p(\mathcal{G}_q)$ using Equation 4.9.
           /* $U$ is matrix of $N \times M$ */
10         for $f \leftarrow 1$ to $M$ do
11             Construct GP classifier $h_f : \mathbb{R}^f \rightarrow \mathcal{L}$.
12             for $k \leftarrow 1$ to $N$ do
                   /* $X_{k,f}$ is $k^{th}$ time series of length $f$ */
13                 Compute $p(\mathcal{G}_q|X_{k,f})$ using Equation 4.10.
14                 Compute $p(L_q|X_{k,f})$ using $h_f$ using Equation 4.11.
15                 Compute utility $\mathcal{U}(X_{k,f})$ using Equation 4.14.
16                 Append $U[k,f] \leftarrow \mathcal{U}(X_{k,f})$.
17         for $k \leftarrow 1$ to $N$ do
18             $\text{MRL}(X_k) = \underset{f}{\text{argmax}} \{U[k,f]\}$.
19             Append $\mathcal{V} \leftarrow \text{MRL}(X_k)$.
           /* $\mathcal{V}_q$ is a set MRLs for $L_q$ class */
20         $f_{j,q} = \hat{f}_h(\mathcal{V}_q = v^*)$, using Equation 4.6.
21         $\mathcal{M}_j = [f_{j,q}]$ for $1 \leq q \leq l$.
22         Append $\boldsymbol{\mathcal{W}}_i \leftarrow \mathcal{M}_j$.
       /* Consider all components of $R_i$ at once to compute MRLs */
23     Compute $\boldsymbol{\mathcal{M}}_{R_i}$ using steps given in Section 4.3.3.
24     Compute accuracy $\mathcal{A}_{R_i}$.
/* Pruning a sub-dataset from $\mathbf{R}$ using Hasse diagram */
/* Consider a node $\mathbf{S}' \in \mathbf{R}$ with parent node $\mathbf{S} \in \mathbf{R}$ */
25 if $(avg(\boldsymbol{\mathcal{M}}_{\mathbf{S}'}) < avg(\boldsymbol{\mathcal{M}}_{\mathbf{S}})$ & $\mathcal{A}_{\mathbf{S}'} \geq \mathcal{A}_{\mathbf{S}})$ then
26     Remove sub-dataset corresponding to node $\mathbf{S}$ from $\mathbf{R}$.
/* Consider $\mathbf{R}_{\text{prun}}(\subseteq \mathbf{R})$ is a set of remaining sub-dataset */
27 $\mathbf{R}_{\text{prun}} = \{R_1, R_2, \cdots, R_r\}$, where $r \leq z$.
28 $\boldsymbol{\mathcal{K}} = \{\mathcal{K}_1, \mathcal{K}_2, \cdots, \mathcal{K}_r\}$, where $\mathcal{K}_i = \{\boldsymbol{\mathcal{W}}_i, \mathcal{M}_{R_i}\}$ and $1 \leq i \leq r$.
29 for $i \leftarrow 1$ to $n$ do
30     Compute error threshold using ARIMA as given in 4.3.5 $\xi_i = \max_{1\leq j\leq N, 1\leq k\leq M} \{\epsilon_{jk}\}$.
31 $\boldsymbol{\xi} = \{\xi_1, \xi_2, \cdots, \xi_n\}$.
32 return $\boldsymbol{\mathcal{K}}$ and $\boldsymbol{\xi}$.
```

tions. As soon as $\mathbf{C}_1^p$ gets $min\{\mathcal{M}_1\}$ data points, FECM checks for faulty components before starting the actual classification. FECM first fits ARIMA(p, d, q) model to each time series of the new MTF and then uses the fitted ARIMA(p, d, q) model to predict the next w data points, where w is a window of minimum number of data points that are to be considered for error checking. A suitable window size can be learned from the training dataset using a method given in [86]. Later, if the difference between the observed and predicted data points is greater than the estimated error threshold then the respective time series can be marked as faulty component of the MTF.

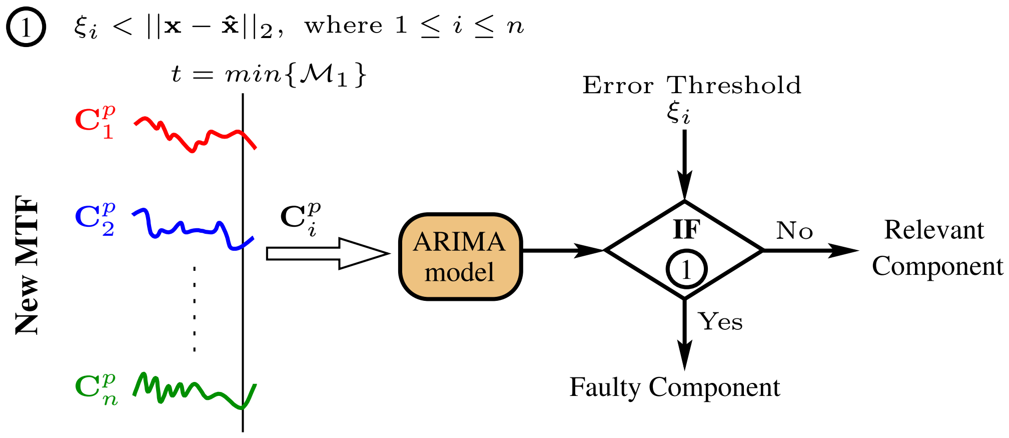


Figure 4.6: Identification of faulty components in a new MTF.

Let m_i denotes number of data points at time t in $\mathbf{C}_i^p \in \mathbf{C}^p$. Let $\mathbf{x}$ denotes a vector of next w observed data points after time t as $\mathbf{x} = \{x^{(t+j)} : 1 \leq j \leq w\}$. Similarly, let $\hat{\mathbf{x}}$ denotes a vector of w predicted data points after time t, using fitted ARIMA(p, d, q) model, as $\hat{\mathbf{x}} = \{\hat{x}^{(t+j)} : 1 \leq j \leq w\}$. Now, if $\xi_i < \|\mathbf{x} - \hat{\mathbf{x}}\|_2$ holds then $\mathbf{C}_i^p$ is marked as faulty component. Figure 4.6 illustrates the process of faulty component identification using ARIMA model. This step is shown at Lines $3 - 7$ in Algorithm 4.2.

4.3.7 Class label prediction

FECM predicts the class label of the MTF without considering the faulty components. Let $\mathbf{C}^{p,s}$ denotes the new MTS $\mathbf{C}^p$ (corresponds to an ongoing activity) with s relevant

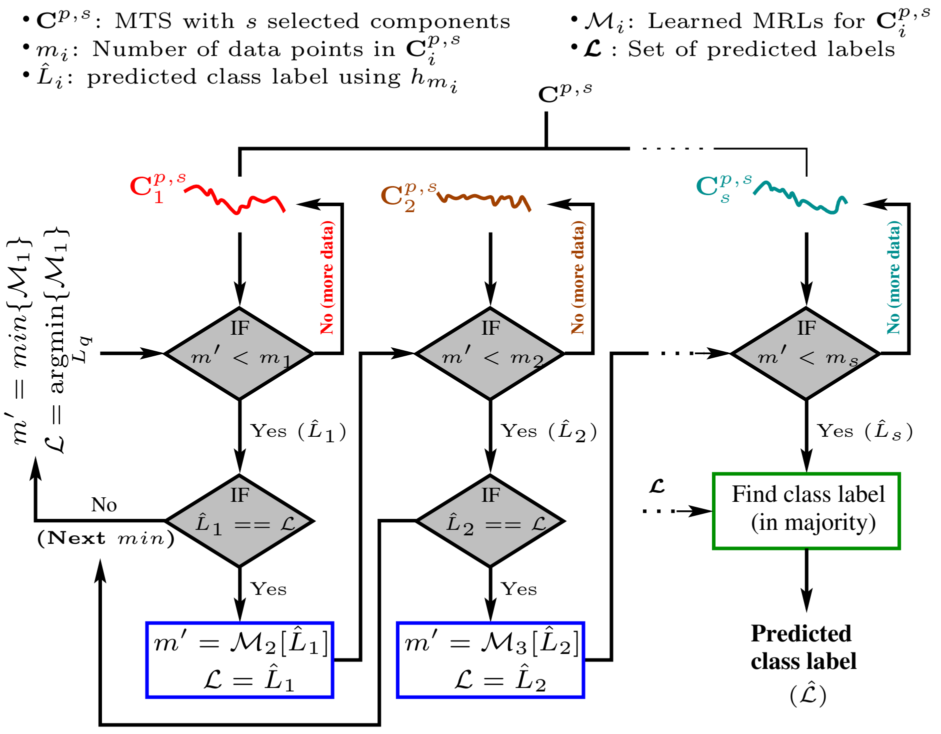
Algorithm 4.2: Testing phase

```
Input: A set built classification models $\boldsymbol{\mathcal{K}}$ and error thresholds $\boldsymbol{\xi}$ using
        Algorithm 4.1 and a new MTF $\mathbf{C}^p = \{\mathbf{C}_1^p, \mathbf{C}_2^p, \cdots, \mathbf{C}_n^p\}$ with $n$ components;
Output: Predicted class label $\hat{\mathcal{L}}$;
/* Number of arrived data points in $\mathbf{C}_1^p$ at $t$ is $m_1 (\leq M)$ */
/* Obtain learned MRLs $\mathcal{M}_1$ for $\mathbf{C}_1^p$ using $\boldsymbol{\mathcal{K}}$ */
1  if $min\{\mathcal{M}_1\} = m_1$ then
2      while $\mathbf{C}^p$ is not complete do
           /* Identifying faulty component */
3          for $i \leftarrow 1$ to $n$ do
4              Learn parameters $p, d, q$ of ARIMA for $\mathbf{C}_i^p$ at $t$.
5              Predict next data point $\hat{x}^{(t+1)}$.
               /* Observed data point at $t+1$ is $x^{(t+1)}$ */
6              if $\xi_i < (x^{(t+1)} - \hat{x}^{(t+1)})$ then
7                  $\mathbf{C}_i^p$ is faulty component and remove it from $\mathbf{C}^p$.
           /* Class label prediction */
           /* New MTS after removing faulty components is $\mathbf{C}^{p,s}$ */
8          Select suitable classification model $\mathcal{K}_s \in \boldsymbol{\mathcal{K}}$ for $\boldsymbol{\alpha}$ accuracy.
9          Select components from $\mathbf{C}^{p,s}$ required for $\mathcal{K}_s$.
           /* Obtain $\mathcal{M}_i$ from $\mathcal{K}_s$, $1 \leq i \leq s$ */
10         Compute $m' = min\{\mathcal{M}_1\}$ and $\mathcal{L} = \underset{L_q}{\text{argmin}}\{\mathcal{M}_1\}$.
11         for $i \leftarrow 1$ to $s$ do
12             if $m' < m_i$ then
13                 Predict label $(\hat{L}_i)$ of $\mathbf{C}_i^{p,s}$ using GP classifier.
14             else
15                 Wait for more data points and go to Line 2.
16             if $\hat{L}_i == \mathcal{L}$ then
17                 Compute $m' = \mathcal{M}_{i+1}[\hat{L}_i]$ and $\mathcal{L} = \hat{L}_i$.
18                 Append $\boldsymbol{\mathcal{L}} \leftarrow \mathcal{L}$.
19             else
                   /* Find next minimum MRL of $\{\mathcal{M}_1\}$ */
20                 Go to Line 10.
21         Find $\hat{\mathcal{L}} = \underset{\mathcal{L}}{\text{argmax}}\{\boldsymbol{\mathcal{L}}\}$ and break.
22 return $\hat{\mathcal{L}}$.
```

components, where $s \leq n$. Let $\mathcal{K}_s$ is the selected classification model from $\boldsymbol{\mathcal{K}}$ based on the number of selected components in $\mathbf{C}^{p,s}$. We propose a method to utilize the correlation between the components by forwarding a class label between the components. In this method, the classification model $\mathcal{K}_s$ uses GP classifier (h_{m_i}) to estimate a class label $(\hat{L}_i)$ using $\mathbf{C}_i^{p,s}$, where m_i denotes the number of data points in $\mathbf{C}_i^{p,s}$ at the time of prediction. Let m' is the estimated MRL which is required to predict a class label $\mathcal{L}$. The method consists of following steps:

1. Compute $m' = min\{\mathcal{M}_1\}$ and $\mathcal{L} = \underset{L_q}{\text{argmin}}\{\mathcal{M}_1\}$.
2. For $i = 1$ to s:
   - If $m' < m_i$ then predict the class label of $\mathbf{C}_i^{p,s}$ otherwise wait for more data points. Let $\hat{L}_i$ denotes a predicted class label.
   - If $\hat{L}_i == \mathcal{L}$ then update $m' = \mathcal{M}_{i+1}[\hat{L}_i]$ for next component and $\mathcal{L} = \hat{L}_i$. Otherwise find the next minimum MRL to avoid the cumulative effect of wrong prediction and go to step 1.
3. Find a class label $(\hat{\mathcal{L}})$ that is predicted by majority of components and assign it to $\mathbf{C}^{p,s}$. Next, the earliness can be obtained using the number of data points used for prediction.

Figure 4.7 shows the method to predict the class label of an MTS. Further, this step is illustrated at Lines $8 - 21$ in Algorithm 4.2. After the prediction, the model waits for full length of the MTS and again predicts the class label. If the predicted class labels using MRL and full length are same then the MTS is segmented from ongoing activity and assigned with the predicted class label. On the contrary, if the predicted class labels are not same then discard collected data from ongoing activity and restart the prediction of class label with forthcoming sensory values.



• $\mathbf{C}^{p,s}$: MTS with s selected components
• m_i: Number of data points in $\mathbf{C}_i^{p,s}$
• $\hat{L}_i$: predicted class label using h_{m_i}
• $\mathcal{M}_i$: Learned MRLs for $\mathbf{C}_i^{p,s}$
• $\mathcal{L}$: Set of predicted labels
$\mathbf{C}^{p,s}$
$\mathbf{C}_1^{p,s}$
$\mathbf{C}_2^{p,s}$
$\mathbf{C}_s^{p,s}$
No (more data)
IF $m' < m_1$
IF $m' < m_2$
IF $m' < m_s$
Yes ($\hat{L}_1$)
Yes ($\hat{L}_2$)
Yes ($\hat{L}_s$)
$m' = min\{\mathcal{M}_1\}$
$\mathcal{L} = \underset{L_q}{\text{argmin}}\{\mathcal{M}_1\}$
No
(Next min)
IF $\hat{L}_1 == \mathcal{L}$
IF $\hat{L}_2 == \mathcal{L}$
Yes
$m' = \mathcal{M}_2[\hat{L}_1]$ $\mathcal{L} = \hat{L}_1$
$m' = \mathcal{M}_3[\hat{L}_2]$ $\mathcal{L} = \hat{L}_2$
$\mathcal{L}$
Find class label (in majority)
Predicted class label ($\hat{\mathcal{L}}$)

Figure 4.7: Overview of prediction of class label of an MTS $\mathbf{C}^{p,s}$.

4.3.8 Complexity Analysis

Time complexity: In Algorithm 4.1, as classification models can be built for each sub-dataset parallelly, the **for** loop at Line 6 can be removed. The time complexity is mainly depend on Lines 11, 13, and 30. At Line 11, the computational complexity of GP classifier is $O(N^3)$, which can be approximated to $O(p^2N)$ where $p(\ll N)$ is a subset of N MTS [42]. As Line 11 is inside two **for** loops (excluding loop at Line 6), its time complexity is $O(s \times M \times p^2N) = O(MN)$ as $s \ll \{M, N\}$. Similarly, the time complexity of Lines 13 and 30 can also be obtained as $O(MN)$. In sum, the time complexity of Algorithm 4.1 can be given as $O(MN)$. The time complexity of Algorithm 4.2 depends on mainly ARIMA model and a **for** loop at Line 11. As ARIMA takes $O(M)$ time, the Line 4 runs $O(M \times n \times M) = O(M^2)$ times, as $n \ll M$. Similarly, Line 11 runs $O(M) = M \times s \times l$ times, as $\{s, l\} \ll M$. Now, time complexity of Algorithm 4.2 can be given as $O(M^2)$. The total time complexity of FECM is therefore $O(MN + M^2)$.

Space complexity: In Algorithm 4.1, if all the classification models are constructed parallelly then the space required to store all relevant sub-datasets is $O(d \times N \times M)$, where $d(\ll 2^n)$ denotes the number of relevant sub-datasets out of 2^n possible sub-datasets. Now, since all the classification models are given as input to Algorithm 4.2, its space complexity will also become $O(dNM)$. The total space complexity of FECM is therefore $O(dNM)$.

4.4 Experimental Evaluation

In this section, we carry out the experimental evaluation to validate the performance of FECM by giving answer to following questions:

- What are the selected components of MTS for different datasets using FECM? (Section 4.4.2.1)
- What is the activity wise performance of FECM using accuracy and earliness metrics? (Section 4.4.2.2)
- What is the impact of faulty components on the performance of FECM (Sections 4.4.2.3 and 4.4.2.4)
- How efficiently can FECM classify the activities compared to existing approaches? (Section 4.4.3.1)
- What is the execution time difference between FECM and the existing approaches? (Section 4.4.3.2)

4.4.1 Datasets

The FECM uses following datasets to evaluate and compare the performance of FECM.

- Human Activity Classification (HAC) dataset: An experiment is conducted to collect human activities using smartphones.
- Existing datasets: Daily and Sports Activities (DSA) [13], HHAR [13], and NTU RGB+D [87].

4.4.1.1 HAC Dataset

We conduct an experiment to create a dataset using onboard sensors of a smartphone. An Android application is developed to record the activities for 30 participants (10 female and 20 male) of age between 20 and 35. There are total eight activities considered in this experiment including standing while talking (**A1**), sitting on sofa (**A2**), sitting on floor (**A3**), lying on bed (**A4**), lying on floor (**A5**), walking downstairs (**A6**), walking upstairs (**A7**), and eating (**A8**). The participants perform each of these activities continuously in a predefined manner at sampling rate 100 Hz. We have used the onboard sensors of Motorola Moto X smartphone to collect activity data of participants. The activity data is collected using three smartphones worn on the wrist, waist, and thigh. Table 4.1 illustrates the type of onboard sensors of smartphones that are selected to place on different body parts. In this experiment, the participants are directed to perform 100 repetitions of each activity. As result, a dataset is created that consists of total 24000 (*i.e.,* $30 \times 8 \times 100$) MTS with their ground truth labels. a dataset

Table 4.1: Type of onboard sensors of smartphone on different body parts.

Position on body	Onboard Sensors	Abbreviation	No. of components
Waist	3-Axis Accelerometer	Acc_wax, Acc_way, Acc_waz	3
Wrist	3-Axis Gyroscope	Gyr_wix, Gyr_wiy, Gyr_wiz	3
Thigh	3-Axis Accelerometer, Pressure, Temperature	Acc_thx, Acc_thy, Acc_thz, Press, Temp	5

• **Preprocessing:** As user is continuously performing the different activities, it becomes extremely important to spot the beginning point of an activity. Such spotting is done by utilizing the fact that the fluctuation in sensory values is higher when user is performing the activity. The fluctuation in the sensory values is measured by computing the variance of a sliding window of last 10 data points (obtained from the ongoing activity). If the variances of any two subsequent windows differ more than a user-defined value then start point of that window indicates the beginning of the activity. The FECM can also identify the end point of the activity using this sliding window. As each activity lasts for different duration, the MTS corresponding to it consists of different number of data points. This work therefore uses least and most significant values of the shorter activities for making the length of all the activities equal [86]. We first determine the minimum length of MTS that can provide nearly full accuracy (*i.e.,* 100%) for all the activities. k-means ($k = 8$) clustering is used to determine such length. In our experiment, full accuracy is achieved using first 900 data points of the MTS. We, therefore, keep only first 900 data points of the MTS in the created dataset and discard the remaining data points.

4.4.1.2 Existing Datasets

DSA dataset consists of MTS data of 19 different human activities with their ground truth class labels. As multiple sensors are used to record the activity, an MTS is generated corresponding to the activities. Five units of motion sensors are used on five different body parts. FECM uses MTS of five motion sensors as follows: gyroscopes (*i.e.,* Gyr1, Gyr2, and Gyr3) on torso, left arm, and right arm, respectively and accelerometers (*i.e.,* Acc1 and Acc2) on left and right legs, respectively. FECM includes following activities from DSA dataset: sitting (**B1**), standing (**B2**), lying on back (**B3**), lying on right side (**B4**), rowing (**B5**), jumping (**B6**), and playing basketball (**B7**).

Next, HHAR dataset was created to understand the impact of heterogeneous nature of mobile sensing devices for human activity classification [88]. Each activity was recorded by using two sensors (*i.e.,* accelerometer and gyroscope) from many devices including eight smartphones and four smartwatches from different manufacturers. The dataset consists of MTS of six different activities including sitting (**C1**), standing (**C2**), walking (**C3**), stair up (**C4**), stair down (**C5**), and biking (**C6**). As the dataset contains the activities that are performed in many different orientations of smartphones and smartwatches, FECM includes the MTS of accelerometer and gyroscope both from smartphone (*i.e.,* Gyr1 and Acc1) and smartwatch (*i.e.,* Gyr2 and Acc2) for performance evaluation.

Further, NTU RGB+D dataset contains video data of various human activities which was collected by using the depth cameras. This dataset also contains MTS observations on motion angles of different human skeleton joints. As three dimensional coordinates are recorded for the skeleton joint by three different cameras, the MTS consists of nine components as $\mathbf{C}_1, \mathbf{C}_2, \cdots \mathbf{C}_9$. In our experiments, we consider following 10 different activities from this dataset: drink water (**D1**), eat meal (**D2**), brush teeth (**D3**), brush hair (**D4**), drop (**D5**), pick up (**D6**), throw (**D7**), sit down (**D8**), stand up (**D9**), and clapping (**D10**).

4.4.2 Results

This section presents the experimental results of the proposed approach using different datasets. The datasets are divided into two parts with balanced class distribution: training with 70% MTS and testing with 30% MTS. FECM uses standard 10-fold cross-validation method in learning phase. Input parameters for different datasets are given in Table 4.2.

Table 4.2: Input parameters of FECM for different datasets.

Dataset	N	N_{train}	N_{test}	n	M	l
HAC	24000	16800	7200	11	900	8
DSA [13]	3360	2352	1008	15	125	7
HHAR [13]	20000	14000	6000	12	1200	6
NTU RGB+D [87]	8000	5600	2400	9	1000	10

4.4.2.1 Selection of components

The FECM uses Hasse diagram (as shown in Figures 4.3 and 4.5) to obtain the most suitable group of components that can provide $\boldsymbol{\alpha}$ accuracy. Figure 4.8 shows the group of components that are selected from MTS for learning MRLs for different datasets at $\boldsymbol{\alpha} = \{0.5, 0.6, 0.7, 0.8, 0.9\}$. At a particular value of $\boldsymbol{\alpha}$, the components that are not selected from the MTS, are either faulty or not required for $\boldsymbol{\alpha}$ accuracy. For example, for HAC dataset at $\boldsymbol{\alpha} = 0.9$, the components $\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3, \mathbf{C}_4, \mathbf{C}_5, \mathbf{C}_7, \mathbf{C}_9$, and $\mathbf{C}_{10}$ are sufficient to provide the $\boldsymbol{\alpha}$ accuracy. The other components $\mathbf{C}_6, \mathbf{C}_8$, and $\mathbf{C}_{11}$ are not required for $\boldsymbol{\alpha} = 0.9$. We observe from the Figure 4.8 that $\mathbf{C}_6$ and $\mathbf{C}_8$ are not selected for any $\boldsymbol{\alpha}$, which indicates that they are the faulty components. We consider $\boldsymbol{\alpha} = 0.9$ for the further results. Further, as the components are arranged in increasing order (starting from $\mathbf{C}_1$) of their relevancy score, the same set of components from different datasets may be used to achieve the desired level of accuracy.

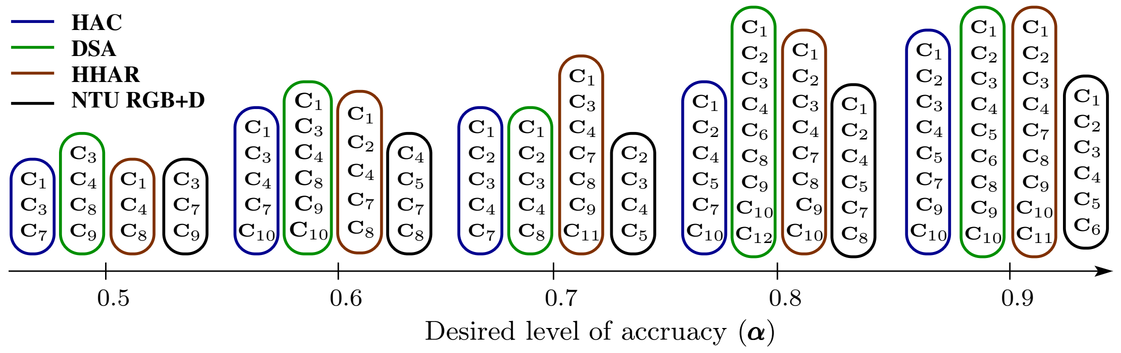
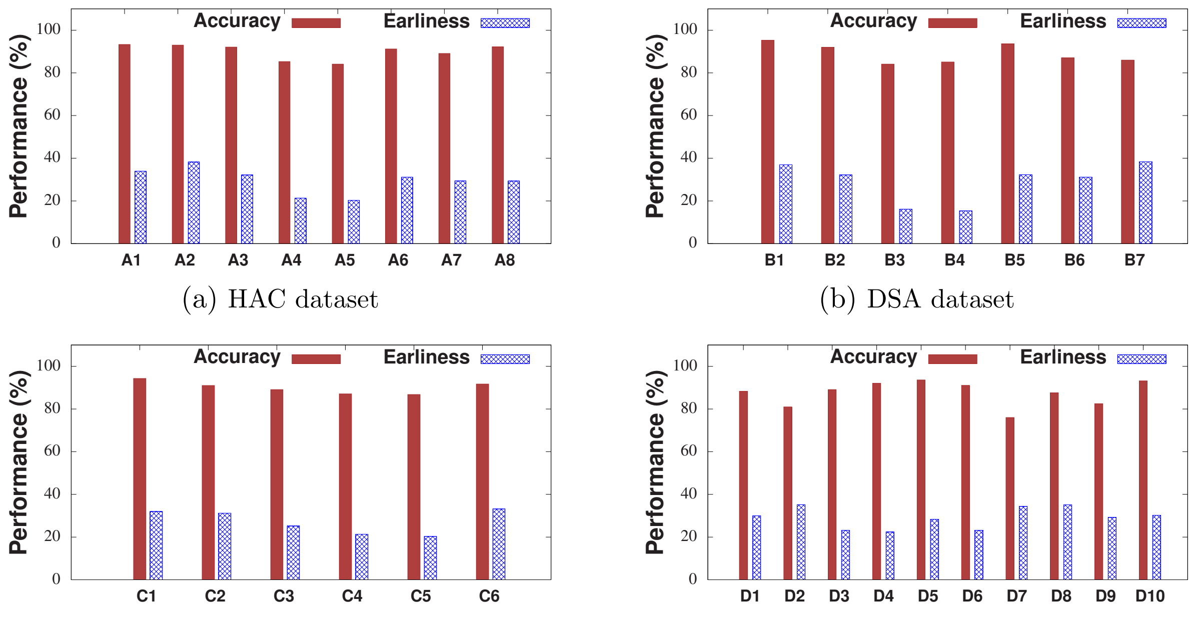


Figure 4.8: Illustration of selected components for different datasets in FECM.

4.4.2.2 Performance on different human activities

We evaluate the activity wise performance of FECM at $\boldsymbol{\alpha} = 0.9$, using the selected components (shown in Figure 4.8). Figure 4.9 illustrates the obtained accuracy and earliness results for various considered activities using different datasets. Part (a) of Figure 4.9 shows that FECM is able to obtain the desired level of accuracy (*i.e.*, 90%)

for all activities except **A4** and **A5**. It indicates that these activities have less distinguishable patterns in their MTS. FECM achieves highest accuracy (*i.e.,* 93.3%) for **A1** activity with 33.9% earliness and the highest earliness (*i.e.,* 38.2%) is obtained for **A2** activity. On the other hand, FECM provides both lowest accuracy (*i.e.,* 84.3%) and lowest earliness (*i.e.,* 20.3%) for **A5** activity. We can make Similar observations about DSA, HHAR, and NTU RGB+D datasets, as shown in parts (b), (c) and (d) of Figure 4.9, respectively.



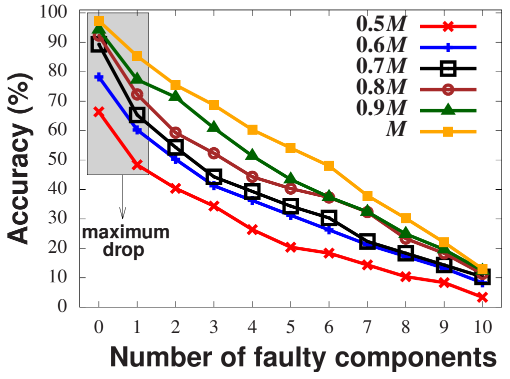
(a) HAC dataset (b) DSA dataset

(c) HHAR dataset (d) NTU RGB+D dataset

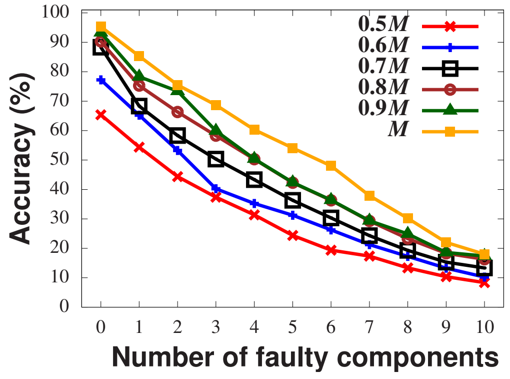
Figure 4.9: Performance evaluation of FECM on different activities at $\boldsymbol{\alpha} = 0.9$.

4.4.2.3 Fault tolerance capability of FECM during training

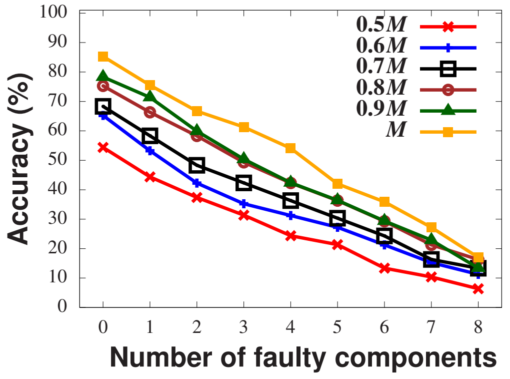
Figure 4.10 illustrates the fault tolerance capability of the FECM with varying number of faulty components in the training data. The performance of FECM is evaluated on four datasets (*i.e.,* HAC, DSA, and NTU RGB+D in parts (a), (b), and (c), respectively). We considered six variations of the each dataset based on the different length of MTS, which are given as $0.5M, 0.6M, 0.7M, 0.8M, 0.9M$, and M, where M denotes the full length of MTS and $0.7M$ denotes 70% length of MTS (for instance). Faulty



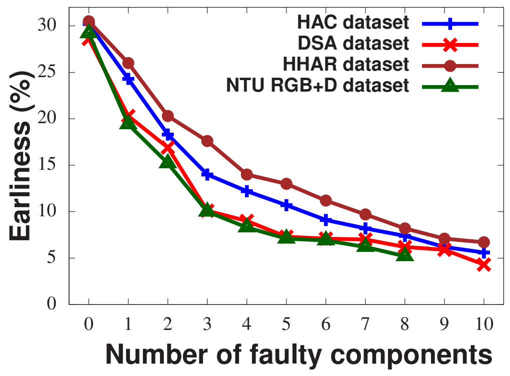
(a) HAC dataset



(b) DSA dataset



(c) NTU RGB+D dataset



(d) Impact on earliness

Figure 4.10: Fault tolerance capability of the FECM with varying number of faulty components in the training data using different datasets.

components are taken from high end of relevancy order (*i.e.,* $\mathbf{C}_1, \mathbf{C}_2, \cdots$). For example, if number of faulty components is four then it means $\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3$, and $\mathbf{C}_4$ are faulty in MTS. We observe following points about Figure 4.10:

- As first component $\mathbf{C}_1$ is most relevant in the MTS, it causes maximum drop in the accuracy of FECM for all the variations if $\mathbf{C}_1$ is faulty (as shown in part (a)).
- NTD RGB+D dataset shows lower accuracy compared to other datasets for all the variations as it has lesser number of components (*i.e.,* 9).
- The accuracy of FECM is worst when it is trained using the dataset with $0.5M$ length of MTS. It indicates that higher the incompleteness lower the accuracy, irrespective of number of faulty components.

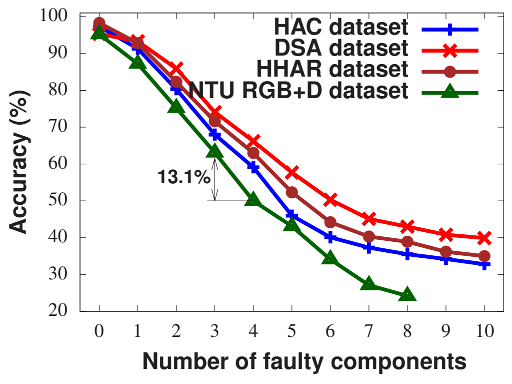
Part (d) of Figure 4.10 shows the impact of faulty components on the earliness of the

FECM when training dataset consists faulty components in the MTS. It is clear from the figure that earliness drops very fast when first few components are faulty in the MTS. For example, earliness drops around 18% when only first three components are faulty for DSA and NTU RGB+D datasets and after that it decreases gradually. It indicates that the first three components are very informative for learning class discriminating MRLs. Further, the number of data points in the training MTS with faulty components also affects the earliness and thus FECM achieves better earliness for HHAR than others.

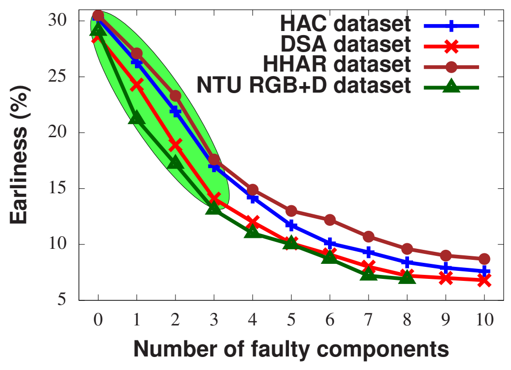
4.4.2.4 Fault tolerance capability of FECM during testing

We conducted an experiment to evaluate the performance of FECM with varying number faulty components in the testing data. The experiment is carried out using the training dataset with complete MTS. Based on the relevancy order of components, the FECM is evaluated for following cases:

• **Case 1 (faulty components are taken from high end of relevancy order):** Figure 4.11 demonstrates the impact of faulty components on the performance of FECM when only testing dataset consists of the faulty components in the MTS. Parts (a) and (b) show that the FECM obtains $(95 \pm 2)\%$ accuracy with $(29 \pm 1.5)\%$ earliness for all the datasets if there is no faulty component in the MTS. In part (a), as faulty components only exist in the testing MTS, the FECM is able to obtain more than 33% accuracy with even ten faulty components which is $(11 \pm 5)\%$ if training MTS also have these faulty components (as shown in Figure 4.10). Similar observations can be made about part (b) of Figure 4.11. We observe that NTU RGB+D dataset shows a 13.1% drop in the accuracy when fourth component also becomes faulty along with previous three, as illustrated in part (a) of Figure 4.11. It indicates that the fourth component has significant identifiable information. The proposed approach can handle up to $n - 1$ faulty sensors. However, the accuracy and earliness both start decreasing rapidly when more than 3 most relevant sensors are faulty.



(a) Accuracy



(b) Earliness

Figure 4.11: Impact on the accuracy and earliness of the FECM with varying number of faulty components in the testing data.

• **Case 2 (faulty components are taken from low end of relevancy order):** In this case, the performance of FECM is evaluated using different datasets by taking faulty components from low end of relevancy order ($i.e.$, $\mathbf{C}_n, \mathbf{C}_{n-1}, \cdots$). For $n = 15$, if number of faulty components is four then it means components $\mathbf{C}_{15}, \mathbf{C}_{14}, \mathbf{C}_{13}$, and $\mathbf{C}_{12}$ are faulty. Table 4.3 illustrates the obtained results using different datasets with increasing number of faulty components. It is interesting to observe that the FECM is able to maintain an acceptable level of accuracy $(90.5 \pm 2.2)\%$ and earliness $(29.2 \pm 1.1)\%$ even when two components are faulty in the testing MTS. Another interesting observation is about NTU RGD+D dataset with eight faulty components, where the FECM is able to achieve 35.7% accuracy using only first component $\mathbf{C}_1$ (as faulty components are removed from the testing MTS). We can make similar observations about other datasets also.

4.4.3 Comparison with existing approaches

In this section, the FECM is compared with four existing early classification approaches for MTS including MCFEC [49], MD-MPP [22], OAE [25], and DMP+PPM [29]. We considered these approaches for performance comparison as they are widely used for early classification of MTS data and have shown good performance. In [22] and [29],

Table 4.3: Performance of FECM on increasing the number of faulty components in the testing MTS. [$\mathcal{A}$: Accuracy and $\mathcal{E}$: Earliness]

Number of faulty components	**HAC dataset**		**DSA dataset**		**HHAR dataset**		**NTU RGB+D dataset**	
	$\mathcal{A}$	$\mathcal{E}$	$\mathcal{A}$	$\mathcal{E}$	$\mathcal{A}$	$\mathcal{E}$	$\mathcal{A}$	$\mathcal{E}$
2	91.3%	29.3%	90.7%	28.1%	92.7%	30.4%	88.3%	30.2%
4	84.1%	26.5%	85.1%	24.2%	85.2%	27.4%	75.6%	25.4%
6	69.4%	18.4%	74.3%	17.3%	71.7%	20.3%	55.7%	18.2%
8	54.9%	12.6%	59.6%	10.2%	62.4%	14.1%	35.7%	8.4%
10	37.2%	9.1%	43.3%	8.4%	48.3%	11.2%	-	-
12	-	-	38.4%	7.7%	32.2%	8.2%	-	-
14	-	-	32.8%	6.6%	-	-	-	-

although MD-MPP and DMP+PPM are evaluated on MTS of video data but these approaches can equally work for other domain data also.

Table 4.4: Comparison of FECM with existing approaches on accuracy and earliness metrics using different datasets at $\boldsymbol{\alpha} = 0.9$.

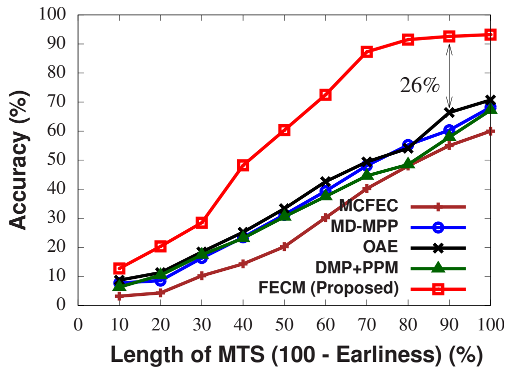
Dataset	**Approach**	**Using selected components** (faulty components excluded)		**Using all components** (faulty components included)	
		Accuracy	Earliness (MRL)	Accuracy	Earliness (MRL)
HAC	**MCFEC** [49]	61.8%	6.2% (844)	53.7%	4.7% (857)
	MD-MPP [22]	72.5%	8.3% (825)	60.3%	6.3% (843)
	OAE [25]	73.3%	10.5% (805)	64.1%	7.5% (832)
	DMP+PPM [29]	68.1%	13.2% (781)	57.9%	8.9% (820)
	FECM (Proposed)	**94.2%**	**24.2% (682)**	**90.1%**	**17.1% (746)**
DSA	**MCFEC** [49]	57.6%	5.6% (118)	51.2%	4.2% (120)
	MD-MPP [22]	69.1%	6.2% (117)	61.1%	4.3% (120)
	OAE [25]	72.7%	6.2% (117)	59.7%	5.5% (118)
	DMP+PPM [29]	66.2%	7.2% (116)	59.2%	6.8% (116)
	FECM (Proposed)	**91.7%**	**19.1% (101)**	**89.3%**	**13.2% (108)**
HHAR	**MCFEC** [49]	60.3%	9.5% (1086)	54.7%	5.5% (1134)
	MD-MPP [22]	73.2%	11.3% (1064)	63.2%	7.1% (1115)
	OAE [25]	76.5%	13.3% (1041)	67.5%	9.6% (1085)
	DMP+PPM [29]	69.1%	14.7% (1024)	58.2%	10.3% (1076)
	FECM (Proposed)	**93.7%**	**25.2% (898)**	**89.9%**	**18.3% (980)**
NTU RGB+D	**MCFEC** [49]	56.2%	7.3% (927)	52.9%	5.1% (949)
	MD-MPP [22]	67.9%	7.4% (926)	60.5%	5.7% (943)
	OAE [25]	69.7%	8.1% (919)	56.7%	6.2% (938)
	DMP+PPM [29]	65.2%	8.4% (916)	60.2%	7.8% (922)
	FECM (Proposed)	**90.5%**	**19.9% (801)**	**88.3%**	**14.2% (858)**

Table 4.4 shows comparison of FECM with existing approaches on the accuracy and earliness metrics at $\boldsymbol{\alpha} = 0.9$. The comparison is carried out for following cases: 1) *Using selected components* of MTS (*i.e.,* excluding faulty components) and 2) *Using all components* of MTS (*i.e.,* including faulty components). The number of faulty (or not

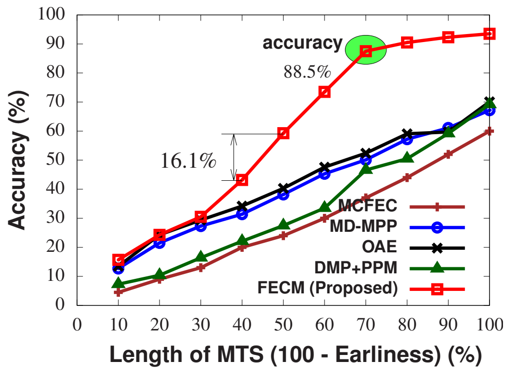
selected) components are varying in different datasets as shown in Figure 4.8. It is clear from the table that FECM outperforms the existing approaches on both accuracy and earliness metrics in both the cases for all the datasets. That means FECM can better classify an MTS with $\boldsymbol{\alpha}$ accuracy even when MTS have faulty components. We observe that the performance of MCFEC approach is worst on both evaluation metrics (*i.e.,* accuracy and earliness) for all the datasets. An interesting observation in the table is that there exists a substantial degradation (*i.e.,* $12 \pm 5\%$) in the accuracy of existing approaches if faulty components are included. For HAC dataset, OAE provides better accuracy than MD-MPP and DMP+PPM, *i.e.,* 73.3% when only selected components are used and 64.1% when all components are used. On the other hand, DMP+PPM gains better earliness than MD-MPP and OAE in both the cases. The FECM shows better earliness for NTU RGB+D dataset but worse accuracy when compared with DSA dataset.

4.4.3.1 Accuracy comparison along the progress of MTS

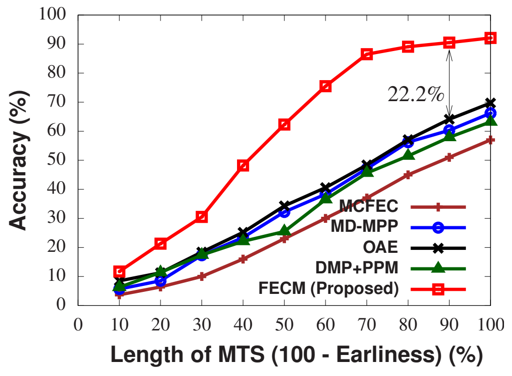
We compare the accuracy results along the progress of the MTS using selected components only, as shown in Figure 4.12. FECM beats all the existing approaches on accuracy as time elapses. FECM shows a substantial increment in the accuracy when compared with existing approaches. In part (a) of Figure 4.12, the increment in the accuracy of FECM is from 4% to 26% approximately while the existing approaches show nearly same accuracy along the progress of MTS. In part (b) of Figure 4.12, FECM performs approximately similar to OAE and MD-MPP upto the length of 30% of MTS but accuracy increases rapidly thereafter. Maximum increment in the accuracy (*i.e.,* 16.1%) occurs from 40% to 50% length of the MTS. FECM is able to achieve an accuracy of 88.5% using only 70% data points of the MTS. For DSA dataset, MCFEC performs worst among all the approaches while MD-MPP and OAE achieve nearly same accuracy along the progress of MTS. Further, similar observations can also be made about HHAR and NTU RGB+D datasets.



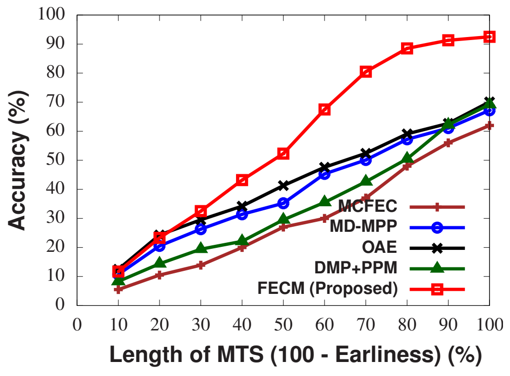
(a) HAC dataset.



(b) DSA dataset.



(c) HHAR dataset.

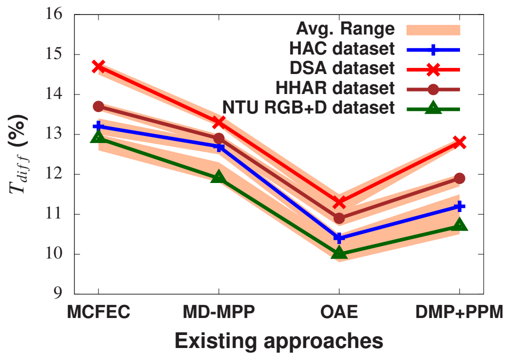


(d) NTU RGB+D dataset.

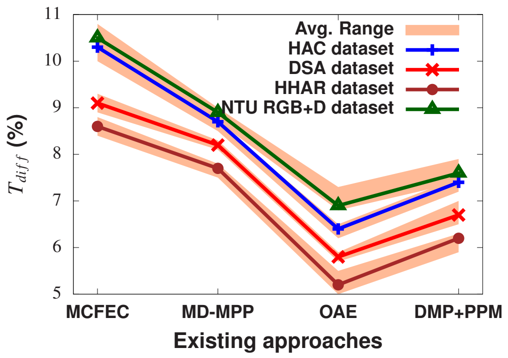
Figure 4.12: Accuracy comparison between FECM and existing approaches along the progress of MTS.

4.4.3.2 Execution time

In this result, we compare the execution time of FECM with existing approaches. Let execution time of any X approach is denoted as T_{X}. The execution time difference (T_{diff}) is computed as $\frac{T_{\mathrm{Y}}-T_{\mathrm{FECM}}}{T_{\mathrm{Y}}}$(%), where Y denotes an existing approach. Figure 4.13 illustrates the average T_{diff} (100 executions) between FECM and existing approaches for the different datasets at $\boldsymbol{\alpha} = 0.9$. Part (a) of Figure 4.13 shows the comparison of execution time difference using selected components and part (b) of Figure 4.13 shows the comparison using all components. It can be clearly observed from the figure that execution time of all existing approaches is more than the FECM, as T_{diff} is positive in all the cases. Moreover, FECM is significantly faster than the existing approaches when only selected components are used, as shown in part (b).



(a) Using selected components.



(b) Using all components.

Figure 4.13: Illustration of execution time difference for the different datasets.

4.5 Conclusion

In this chapter, we proposed a fault-tolerant early classification approach to classify the MTS. Unlike existing work, the proposed approach is able to classify an MTS even when it has some faulty components. FECM constructs a set classification models by estimating the class discriminating MRLs using GP classifier and k-means clustering. The models use ARIMA model to identify the faulty components in the new incomplete MTS. The FECM also selects minimum number of components from the MTS, which can provide the desired level of accuracy. Our experimental results showed that the FECM is able to achieve the desired level of accuracy while providing significant earliness, especially in the human activity classification system. FECM outperformed the existing approaches on all the used datasets. We believe that our research can be used in health care for early prediction of diseases and fall detection using activity data. This work motivates further research to develop a feature based early classification approach for time series data.

Here, the MTS is allowed to have some of its components as faulty and we mainly focused on handling the faulty components. The previous and current chapters assumed that new MTS (to be classified) should belong to one of the seen class label. It essentially means that the classification model can classifies the new MTS correctly only if it belongs to seen classes for which training instances were available. However, having the MTS instances of all possible class labels in the training dataset may not be feasible in some applications. For example, in industrial and domestic monitoring, the machines or appliances can undergo several faults during its lifetime but having the information about all types of faults is not practically possible. In the next chapter, we focus on to develop an early classification approach that can classify an MTS even if it belongs to an unseen class label. It improves the adaptability of the approach for industrial applications where unseen faults may occur in the appliances at any time.

Publication

- **Ashish Gupta**, H. P. Gupta, B. Biswas, and T. Dutta, “A fault-tolerant early classification approach for human activities using multivariate time series,” *IEEE Transactions on Mobile Computing*, pp. 1-14, 2020.