Supervised Anomaly Detection in Uncertain Pseudoperiodic Data Streams

JIANGANG MA, LE SUN, HUA WANG, and YANCHUN ZHANG, Victoria University
UWE AICKELIN, University of Nottingham

Uncertain data streams have been widely generated in many Web applications. The uncertainty in data streams makes anomaly detection from sensor data streams far more challenging. In this article, we present a novel framework that supports anomaly detection in uncertain data streams. The proposed framework adopts the wavelet soft-thresholding method to remove the noises or errors in data streams. Based on the refined data streams, we develop effective period pattern recognition and feature extraction techniques to improve the computational efficiency. We use classification methods for anomaly detection in the corrected data stream. We also empirically show that the proposed approach shows a high accuracy of anomaly detection on several real datasets.

Categories and Subject Descriptors: I.5.4 [**Pattern Recognition**]: Applications

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Anomaly detection, uncertain data stream, segmentation, classification

ACM Reference Format:
Jiangang Ma, Le Sun, Hua Wang, Yanchun Zhang, and Uwe Aickelin. 2016. Supervised anomaly detection in uncertain pseudoperiodic data streams. ACM Trans. Internet Technol. 16, 1, Article 4 (January 2016), 20 pages.
DOI: http://dx.doi.org/10.1145/2806890

4

1. INTRODUCTION

Data streams have been widely generated in many Web applications, such as monitoring click streams [Gündüz and Özsu 2003], stock tickers [Chen et al. 2000; Zhu and Shasha 2002], sensor data streams, and auction bidding patterns [Arasu et al. 2003]. For example, in the applications of Web tracking and personalization, Web log entries and click streams are typical data streams. Other traditional and emerging applications include wireless sensor networks (WSNs), in which data streams collected from sensor networks are being posted directly to the Web. Typical applications comprise environment monitoring (with static sensor nodes) [Akyildiz et al. 2005] and animal and object behavior monitoring (with mobile sensor nodes), such as water pollution detection [He et al. 2012] based on water sensor data, agricultural management and cattle moving

This work is supported by the National Natural Science Foundation of China (NSFC 61332013) and the Australian Research Council (ARC) Discovery Projects DP140100841 and DP130101327, and Linkage Project LP100200682.
Authors' address: J. Ma, L. Sun, and H. Wang, Centre for Applied Informatics, Victoria University, VIC, 3011, Australia; emails: Jiangang.Ma@vu.edu.au, le.sun1@live.vu.edu.au, hua.wang@vu.edu.au; Y. Zhang, School of Computer Science, Fudan University, Shanghai 200433, China, and Centre for Applied Informatics, Victoria University, VIC, 3011, Australia; email: Yanchun.Zhang@vu.edu.au; U. Aickelin, Computer Science, The University of Nottingham, NG8 1BB UK; email: uwe.aickelin@nottingham.ac.uk.

DOI: http://dx.doi.org/10.1145/2806890

habits [Swain et al. 2011], and analysis of trajectories of animals [Gudmundsson et al. 2007], vehicles [Zheng et al. 2010], and fleets [Lee et al. 2007].

Anomaly detection is a typical example of a data stream application. Here, anomalies or outliers or exceptions often refer to the patterns in data streams that deviate from expected normal behaviors. Thus, anomaly detection is a dynamic process of finding abnormal behaviors from given data streams. For example, in medical monitoring applications, a human electrocardiogram (ECG) (vital signs) and other treatments and measurements are typical data streams that appear in a form of periodic patterns. In other words, the data present a repetitive pattern within a certain time interval. Such data streams are called *pseudo periodic time series*. In such applications, data arrives continuously, and anomaly detection must detect suspicious behaviors from the streams, such as abnormal ECG values, abnormal shapes, or exceptional period changes.

Uncertainty in data streams makes the anomaly detection far more challenging than detecting anomalies from deterministic data. For example, uncertainties may result from missing points from a data stream, missing stream pieces, or measurement errors due to different reasons, such as sensor failures and measurement errors from different types of sensor devices. This uncertainty may cause serious problems in data stream mining. For example, in an ECG data stream, if a sensor error is classified as abnormal heartbeat signals, it may cause a serious misdiagnosis. Therefore, it is necessary to develop effective methods to distinguish uncertainties and anomalies, remove uncertainties, and finally find accurate anomalies.

There are several related research areas to sensor data stream mining, such as data streams compression, similarity measurement, indexing, and querying mechanisms [Esling and Agon 2012]. For example, to clean and remove uncertainty from data, a method for compressing data streams was presented in Douglas and Peucker [1973]. This method uses some critical points in a data stream to represent the original stream. However, this method cannot compress uncertain data streams efficiently, because such compression may result in an incorrect data stream approximation and may remove useful information that can correct the error data.

This article focuses on anomaly detection in uncertain pseudoperiodic time series. The uncertainty on which we focus in this work is the noisy signals coming from the error signal collection. A pseudoperiodic time series refers to a time-indexed data stream in which the data present a repetitive pattern within a certain time interval. However, the data may in fact show small changes between different time intervals. Although much work has been devoted to the analysis of pseudoperiodic time series [Keogh et al. 2005; Huang et al. 2014], few of them focus on the identification and correction of uncertainties in this kind of data stream.

We propose a supervised classification framework for detecting anomalies in uncertain pseudoperiodic time series, which consists of four components: a time series signal noise reduction component (SNRC), a time series compression component (TSCC), a period segmentation and summarization component (PSSC), and a anomaly detection and prediction component (ADPC). First, SNRC processes a time series to remove uncertainties from the time series. Then TSCC compresses the processed raw time series to an approximate time series. Afterward, the PSSC identifies the periodic patterns of the time series and extracts the most important features of each period, and finally the CADC detects anomalies based on the selected features. Our work has made the following distinctive contributions:

—We present a classification-based framework for anomaly detection in uncertain pseudoperiodic time series, together with a novel set of techniques for segmenting and extracting the main features of a time series. The procedure of preprocessing uncertainties can reduce the noise of anomalies and improve the accuracy of anomaly detection. The time series segmentation and feature extraction techniques can improve the performance and time efficiency of classification.

—We propose the novel concept of a feature vector (FV) to capture the features of the turning points in a time series and introduce a silhouette value-based approach to identify the periodic points that can effectively segment the time series into a set of consecutive periods with similar patterns.

—We conduct an extensive experimental evaluation over a set of real time series datasets. Our experimental results show that the techniques we have developed outperform previous approaches in terms of accuracy of anomaly detection. In the experiment part of this article, we evaluate the proposed anomaly detection framework on ECG time series. However, due to the generic nature of features of pseudoperiodic time series (e.g., similar shapes and intervals occur in a periodic manner), we believe that the proposed method can be widely applied to periodic timeseries mining in different areas.

The structure of this article is as follows. Section 2 introduces the related research work. Section 3 presents the problem definition and generally describes the proposed anomaly detection framework. Section 4 describes the anomaly detection framework in detail. Section 5 presents the experimental design and discusses the results. Finally, Section 6 concludes the article.

2. RELATED WORK

We analyze the related research work from two dimensions: anomaly detection and uncertainty processing.

Anomaly detection in data streams. Anomaly detection in time series has various applications in wide areas, such as intrusion detection [Tavallaee et al. 2010], disease detection in medical sensor streams [Manning and Hudgins 2010], and biosurveillance [Shmueli and Burkom 2010]. Ling Zhang et al. [2009] designed a Bayesian classifier model for identification of cerebral palsy by mining gait sensor data (stride length and cadence). In stock price time series, anomalies exist in a form of change points that reflect the abnormal behaviors in the stock market, and often repeating motifs are of interest [Wilson et al. 2008]. Detecting change points has significant implications for conducting intelligent trading [Jiang et al. 2011]. Liu et al. [2010] proposed an incremental algorithm that detects changes in streams of stock order numbers, in which a Poisson distribution is adopted to model the stock orders and a maximum likelihood (ML) method is used to detect the distribution changes.

The segmentation of a time series refers to the approximation of the time series, which aims to reduce the time series dimensions while keeping its representative features [Esling and Agon 2012]. One of the most popular segmentation techniques is the piecewise linear approximation (PLA)-based approach [Keogh et al. 2004; Qi et al. 2015], which splits a time series into segments and uses polynomial models to represent the segments. Xu et al. [2012] improved the traditional PLA-based techniques by guaranteeing an error bound on each data point to maximally compact time series. Lemire [2007] introduced an adaptive time series summarization method that models each segment with various polynomial degrees. To emphasize the significance of the newer information in a time series, Palpanas et al. [2008] defined user-oriented amnesic functions for decreasing the confidence of older information continuously.

However, the approaches mentioned previously are not designed to process and adapt to the area of pseudoperiodic data streams. Detecting anomalies from periodic data streams has received considerable attention, and several techniques have been proposed recently [Folarin et al. 2001; Grinsted et al. 2004; Levy and Pappano 2007]. The existing techniques for anomaly detection adopt sliding windows [Keogh et al. 2005; Gu et al. 2005] to divide a time series into a set of equal-sized subsequences. However, this

Table I. Frequently Used Symbols

Symbols	Meaning
TS	A time series
p_i	The ith point in a TS
SS	A subsequence
PTS	A pseudo periodic time series
Q	A set of period points in a PTS
pd	A period in a PTS
CTS	A compressed PTS
vec_i	An FV of point p_i
$sil(p_i)$	Silhouette value of point p_i
$sim(p_i, p_j)$	Euclidean distance based similarity between points p_i and p_j
C	A set of clusters
$msil(C)$	Mean silhouette value of a cluster C
seg_i	A summary of a period
STS	A segmented CTS
A	A set of annotations
Lbs	A set of labels indicating the states
$lb_{(i)}$	The ith label in Lbs

type of method may be vulnerable to tiny difference in time series because it cannot well distinguish the abnormal period and a normal period having small noisy data. In addition, as the length of periods is varying, it is difficult to capture the periodicity by using a fixed-size window [Tang et al. 2007]. Other examples of segmenting pseudoperiods include a peak-point–based clustering method and valley-point–based method [Huang et al. 2014; Tang et al. 2007]. These two methods may have very low accuracy when the processed time series have noisy peak points or have irregularly changed subsequences. Our proposed approach falls into the category of classification-based anomaly detection, which is proposed to overcome the challenge of anomaly detection in periodic data streams. In addition, our method is able to identify qualified segmentation and assign annotation to each segment to effectively support the anomaly detection in a pseudoperiodic data streams.

Uncertainty processing in data streams. Most data streams coming from real-world sensor monitoring are inherently noisy and uncertain. A lot of work has concentrated on the modeling of uncertain data streams [Aggarwal and Yu 2008; Aggarwal 2009; Leung and Hao 2009]. Dallachiesa et al. [2012] surveyed recent similarity measurement techniques of uncertain time series and categorized these techniques into two groups: probability density function–based methods [Sarangi and Murthy 2010] and repeated measurement methods [Aßfalg et al. 2009]. Tran et al. [2012] focused on the problem of relational query processing on uncertain data streams. However, previous work rarely focused on the detection and correction of the missing critical points for a discrete time series.

3. PROBLEM SPECIFICATION AND PREREQUISITES

In this section, we first give a formal definition of the problems and then describe the proposed framework of detecting abnormal signals in uncertain time series with pseudoperiodic patterns. The symbols frequently used in this article are summarized in Table I.

3.1. Problem Definition

Definition 3.1. A *time series TS* is an ordered real sequence: $TS = (v_1, \ldots, v_n)$, where v_i, $i \in [1, n]$, is a point value on the time series at time t_i.

We use the form $|TS|$ to represent the number of points in time series TS (i.e., $|TS| = n$). Based on the preceding definition, we define subsequence of a TS as follows.

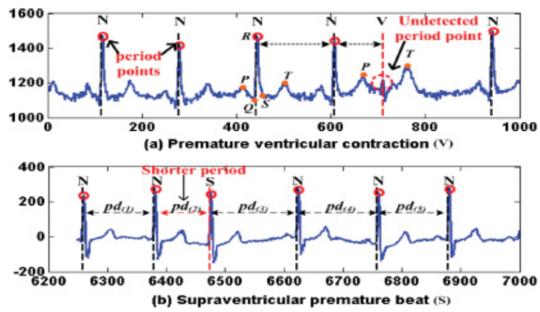

Fig. 1. Two examples of local anomaly in ECG time series.

Definition 3.2. For time series TS, if $SS(\subset TS)$ comprises m consecutive points: $SS = (v_{s_1}, \ldots, v_{s_m})$, we say that SS is a *subsequence* of TS with length m, represented as $SS \sqsubseteq TS$.

Definition 3.3. A *pseudoperiodic time series PTS* is a time series $PTS = (v_1, v_2, \ldots, v_n)$, $\exists Q = \{v_{p_1}, \ldots, v_{p_k} | v_{p_i} \in PTS, i \in [1, k]\}$, that regularly separates PTS on the following condition:

(1) $\forall i \in [1, k-2]$, if $\triangle_1 = |p_{i+1} - p_i|$, $\triangle_2 = |p_{i+2} - p_{i+1}|$, then $| \triangle_2 - \triangle_1 | \leq \xi_1$, where ξ_1 is a small value.
(2) Let $s_1 = (v_{p_i}, v_{(p_i)+1}, \ldots, v_{p_{i+1}}) \sqsubseteq PTS$, and $s_2 = (v_{p_{i+1}}, v_{(p_{i+1})+1}, \ldots, v_{p_{i+2}}) \sqsubseteq PTS$, then $dsim(s1, s2) \leq \xi_2$, where $dsim()$ calculates the dis-similarity between $s1$ and $s2$, and ξ_2 is a small value. $dsim()$ can be any dis-similarity measuring function between time series (e.g., Euclidean distance).

In particular, $v_{p_{i+1}} \in Q$ is called a *period point*.

An uncertain PTS is a PTS having error detected data or missing points.

Definition 3.4. If $pd \sqsubseteq PTS$, and $pd = (v_{p_i}, v_{(p_i)+1}, \ldots, v_{p_{i+1}})$, $\forall v_{p_i} \in Q$, then pd is called a *period* of the PTS.

Definition 3.5. A *normal pattern M* of a PTS is a model that uses a set of rules to describe a behavior of a subsequence SS, where $m = |SS|$ and $m \in [1, |PTS|/2]$. This behavior indicates the normal situation of an event.

Based on the preceding definitions, we describe types of anomalies that may occur in a PTS. There are two possible types of anomalies in a PTS: local anomalies and global anomalies. Given the PTS in Definition 3.3, and a normal pattern $N = (v_1, \ldots, v_m) \sqsubseteq PTS$, a local anomaly ($L$) is defined as follows.

Definition 3.6. Assume that $L = (v_{l_1}, \ldots, v_{l_n}) \sqsubseteq PTS$, L is a local anomaly if either of the two conditions in Definition 3.3 is broken (shown in (1)), and at the same time satisfies the other two conditions ((2) and (3)):

(1) $\triangle_N - \triangle_L > \xi_1$ or $dsim(N, L) > \xi_2$; and
(2) Frequency of L: $freq(L) \ll freq(N)$, and L does not happen in a regular sampling frequency.
(3) $|L| \ll |PTS|$.

Example 3.7. Figure 1 shows two examples of pseudoperiodic time series and their local anomalies. Figure 1(a) shows a premature ventricular contraction (PVC) signal in an ECG stream. A PVC [Levy and Pappano 2007] is perceived as a "skipped beat." It

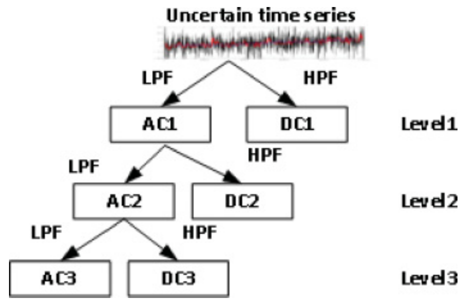

Fig. 2. Iterative wavelet decomposition.

can be easily distinguished from a normal heartbeat when detected by the ECG. From Figure 1(a), the QRS and T waves of a PVC (indicated by V) are very different from the normal QRS and T (indicated by N). Figure 1(b) presents an example of premature atrial contractions (PACs) [Folarin et al. 2001]. A PAC is a premature heartbeat that occurs earlier than the regular beat. If we use the highest peak points as the period points, then a segment between two peak points is a period. From Figure 1, the second period (a PAC) is clearly shorter than the other periods.

3.2. Wavelet-Based Error Reduction

In the proposed framework, we use the wavelet noise reduction method to reduce the white noise in a time series obtained from the signal collection stage [Agante and de Sa 1999]. We briefly introduce this denoising method in this section. The wavelet denoising process contains the following three steps.

Step 1: Wavelet signal decomposition. In this step, a time series is iteratively broken down to finer-resolution signals in terms of frequencies. This decomposition process depends on two symmetric filters: low-pass filter (LPF) and high-pass filter (HPF), which are both created from a *mother* wavelet. The LPF filters the low-frequency signals (approximate coefficient), whereas the HPF keeps the high-frequency signals (detailed coefficient). They are applied in a few iterative steps, which results in a tree structure with signals decomposed by different banks. The decomposition structure is shown in Figure 2, where *AC* represents the approximate coefficient and *DC* means the detailed coefficient.

Step 2: Noise reducing through soft thresholding. The key step of noise reducing is to find a noise threshold that is used to distinguish the normal and noise signals. The soft-thresholding method proposed by Donoho [1995] is applied to filter the noises in the high-frequency signals (i.e., *DC* in Figure 2), which is processed as follows: if the amplitude of a signal point (i.e., wavelet coefficient) is smaller than a threshold value, the signal point is seen as a noise and it is removed (i.e., its coefficient is set to 0); otherwise, this point is treated as a normal waveform signal and its value is subtracted by the threshold. In this work, we mainly deal with the white Gaussian noise whose threshold value is determined by Formula (1).

$$t_n = \sigma\sqrt{2\log n}, \tag{1}$$

where σ is a noise standard deviation estimated based on the first-level signals with highest frequency (i.e., $DC1$ in Figure 2) and n is the length of the time series.

Step 3: Signal reconstructing. After noise reduction is done on each level, the remaining signal points are combined together in a bottom-up manner (from level 3 to the root in Figure 2) to obtain a filtered time series.

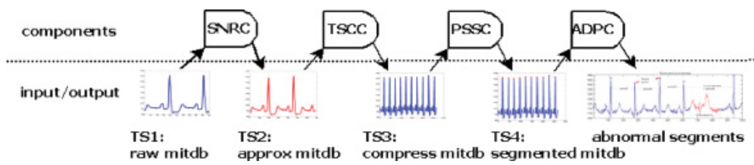

Fig. 3. Workflow of the *mitdb* processing based on the proposed framework.

4. ANOMALY DETECTION IN UNCERTAIN PERIODIC TIME SERIES

The proposed framework consists of four main components: an SNRC), a TSCC, a PSSC, and an anomaly detection and prediction component (ADPC). We explain the process of anomaly detection of the proposed framework using an example of the dataset *mitdb*. Figure 3 shows the processing progress of *mitdb*. First, the uncertain *mitdb* time series is an input to the SNRC component. The TS1 in Figure 3 shows a subsequence of the raw *mitdb*. The SNRC removes the errors in *mitdb*, then the uncertain *mitdb* is transformed into a refined time series (TS2 in Figure 3). The TSCC component then further compresses the approximated *mitdb*. The TS3 in Figure 3 shows the compressed time series (*CTS*) that is a compression of the subsequence in TS2. The PSSC component segments the time series and assigns annotations to each segment. TS4 in Figure 3 shows the segmented and annotated *CTS* corresponding to the *CTS* in TS3. Finally, the ADPC component learns a classification model based on the segmented *CTS* to detect abnormal subsequences in similar time series.

In the sequel of this section, we introduce the anomaly detection framework in detail.

4.1. Anomaly Detection in Refined Time Series

The first step is to remove the noise in the uncertain time series (SNRC). We use the wavelet-based approach introduced in Section 3.2 to filter the errors obtained in the signal collection process. The refined time series is then processed for anomaly detection and normal pattern identification, which is based on the unit of *period*. Therefore, we need to identify period points Q that separate *PTS* into a set of periods. We use a clustering method to categorize the inflexions of a *PTS* into several clusters. Then a cluster quality validation mechanism is applied to validate the quality of each cluster. The cluster with the highest quality will be adopted as the period cluster—that is, the points in the period cluster will be the period points for the time series. The period points are the points that can regularly and consistently separate the *PTS* better than the points in the other clusters.

The cluster quality validation mechanism is a silhouette-value–based method, in which the cluster that has the highest mean silhouette value will be assumed to have the best clustering pattern. To accurately conduct clustering, we introduce an FV for each inflexion of *PTS*, with the optimal intention that each point can be distinguished with others efficiently.

4.1.1. Time Series Compression. To save the storage space and improve the calculation efficiency, the raw *PTS* will first be compressed. In this work, we use the Douglas–Peucker (DP) [Hershberger and Snoeyink 1994] algorithm to compress a *PTS*, which is defined as follows: (1) use line segment $\overline{p_1 p_n}$ to simplify the *PTS*; (2) find the farthest point p_f from $\overline{p_1 p_n}$; (3) if distance $d(p_f, \overline{p_1 p_n}) \leq \lambda$, where λ is a small value, and $\lambda \geq 0$, then the *PTS* can be simplified by $\overline{p_1 p_n}$, and this procedure is stopped; (4) otherwise, recursively simplify the subsequences $\{p_1, \ldots, p_f\}$ and $\{p_f, \ldots, p_n\}$ using steps 1 through 3.

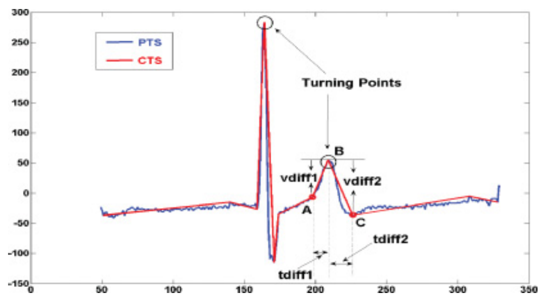

Fig. 4. A PTS and one of its CTSs.

Definition 4.1. Given a $PTS = (v_1, \ldots, v_n)$, a *compressed time series CTS* of PTS is represented as $CTS = (v_{c_1}, \ldots, v_{c_n}) \subseteq PTS$, where $\forall p_{c_i} \in CTS$ is an inflexion, and $|CTS| \ll |PTS|$.

The FV of an inflexion is defined as follows.

Definition 4.2. A *feature vector* for a point $p_i \in CTS$ is a four-value vector $vec_i = (vdiff1_i, vdiff2_i, tdiff1_i, tdiff2_i)$, where $vdiff1_i = v_i - v_{i-1}$, $vdiff2_i = v_{i+1} - v_i$, $tdiff1_i = t_i - t_{i-1}$, and $tdiff2_i = t_{i+1} - t_i$.

Example 4.3. Figure 4 shows an example of a PTS and one of its compressed time series CTS. The value differences $vdiff1$ and $vdiff2$, and the time differences $wdiff1$ and $wdiff2$, are shown in Figure 4.

4.1.2. Period Segmentation and Summarization. PSSC identifies period points that separate the CTS into a series of periods, which is implemented by three steps: cluster points of CTS, evaluate the quality of clusters based on silhouette value, and segment and annotate periods. Details of these steps are given in the following.

Step 1: Cluster points of CTS. Points are clustered into several clusters based on their FVs. In this work, we use k-means++ [Arthur and Vassilvitskii 2007] clustering method to cluster points. It has been validated that based on the proposed FV, the k-means++ is more accurate and less time consumed than other clustering tools (e.g., k-means [Hartigan and Wong 1979], Gaussian mixture models [Reynolds 2009], and spectral clustering [Ng et al. 2001]). We give a brief introduction of the k-means++ in this section.

k-means++ is an improvement of k-means by first determining the initial clustering centers before conducting the k-means iteration process. k-means is a classical NP-hard clustering method. One of its drawbacks is the low clustering accuracy caused by randomly choosing the k starting points. The arbitrarily chosen initial clusters cannot guarantee a result converging to the global optimum all the time. k-means++ is proposed to solve this problem. k-means++ chooses its first cluster center randomly, and each of the remaining ones is selected according to the probability of the point's squared distance to its closest center point being proportional to the squared distances of the other points. The k-means++ algorithm has been proved to have a time complexity of $O(logk)$, and it is of high time efficiency by determining the initial seeding. For more details of k-means++, readers can refer to Arthur and Vassilvitskii [2007].

Step 2: Evaluate the quality of clusters based on silhouette value. We use the mean silhouette value [Rousseeuw 1987] of a cluster to evaluate the quality of a cluster. The silhouette value can interpret the overall efficiency of the applied clustering method and the quality of each cluster, such as the tightness of a cluster and the similarity of the elements in a cluster. The silhouette value of a point belonging to a cluster is defined as follows.

Definition 4.4. Let points in PTS be clustered into k clusters: $C_{CTS} = \{C_1, \ldots, C_m, \ldots, C_k\}, k \leq |CTS|$. For any point $p_i = v_i \in C_m$, the silhouette value of p_i is

$$sil(p_i) = \frac{b(p_i) - a(p_i)}{max\{a(p_i), b(p_i)\}}, \tag{2}$$

where $a(p_i) = \frac{1}{M-1} \sum_{p_i, p_j \in C_m, i \neq j} sim(p_i, p_j)$, $M = |C_m|$ is the number of elements in cluster m, and $b(p_i) = min(\frac{1}{M-1} \sum_{p_i \in C_m, p_j \in C_h, h \neq m} sim(p_i, p_j))$. $sim(p_i, p_j)$ represents the similarity between p_i and p_j.

In the preceding definition, $sim(p_i, p_j)$ can be calculated by any similarity calculation formula. In this work, we adopt the Euclidean distance as a similarity measure—that is, $sim(p_i, p_j) = \sqrt{(v_i - v_j)^2 + (t_i - t_j)^2}$, where t_i and t_j are the time indexes of the points p_i and p_j. From the definition, $a(p_i)$ measures the dissimilarity degree between point p_i and the points in the same cluster, whereas $b(p_i)$ refers to the dissimilarity between p_i and the points in the other clusters. Therefore, a small $a(p_i)$ and a large $b(p_i)$ indicate a good clustering. As $-1 \leq sil(p_i) \leq 1$, a $sil(p_i) \rightarrow 1$ means that a point p_i is well clustered, whereas $sil(p_i) \rightarrow_+ 0$ represents that the point is close to the boundary between clusters M and H, and $sil(p_i) < 0$ indicates that point p_i is close to the points in the neighboring clusters rather than the points in cluster M.

The mean value of the silhouette values of points is used to evaluate the quality of the overall clustering result: $msil(C_{CTS}) = \frac{1}{|CTS|} \sum_{p_i \in CTS} sil(p_i)$. Similar to the silhouette value of a point, the $msil \rightarrow 1$ represents a better clustering.

After clustering, we need to choose a cluster in which the points will be used as period points for the CTS. The chosen cluster is called *period cluster*. The points in the period cluster are the most stable points that can regularly and consistently separate CTS. We use the mean silhouette value of each cluster to evaluate the efficiency of a single cluster, represented as $msil(C_m) = \sum_{p_i \in C_m} sil(p_i)$, where $-1 \leq msil(C_m) \leq 1$, and $msil(C_m) \rightarrow 1$ means the high quality of the cluster m. Based on the definition of silhouette values, we give Algorithm 1 of the choosing period cluster from a clustering result. Algorithm 1 shows that if the mean silhouette value of the overall clustering result is less than a predefined threshold value η, then the clustering result is unqualified. FVs of points need to be reclustered with adjusted parameters (e.g., change the number of clusters). The last line indicates that the chosen period cluster is the one with highest mean silhouette values that is higher than a threshold ξ.

ALGORITHM 1: Cluster quality validation

Input: (1) $V = \{vec_i | 1 \leq i \leq |CTS|\}$, where $vec_i = (vdiff1_i, vdiff2_i, tdiff1_i, tdiff2_i)$
(2) A set of point clusters: $C_{CTS} = \{C_m | 1 \leq m \leq k\}$
(3) Threshold values η and ξ, $0 \leq \eta, \xi \leq 1$
Output: Period cluster C_{perid}
Calculate $sil(p_i)$ for $\forall p_i \in CTS$;
Calculate mean silhouette value: $msil(C_{CTS})$;
if $msil(C_{CTS}) < \eta$ **then**
    $C_{perid} = NULL$;
    return;
end
$C_{perid} = max(msil(C_m))$ & $msil(C_m) > \xi$ for $\forall C_m \in C_{CTS}$.

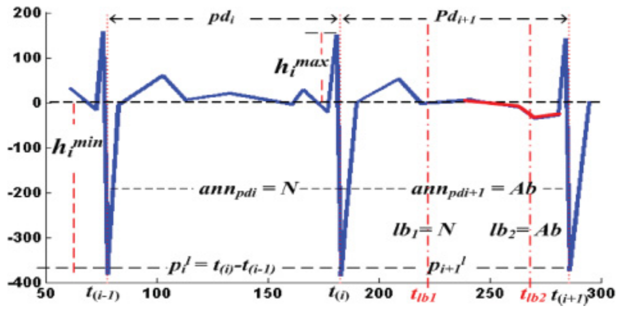

Fig. 5. Segmentation and annotation of two periods.

Step 3: Segmentation and annotation of periods. As mentioned in the previous section, a *CTS* can be divided into a series of periods by using the period points. Thus, detecting a local anomaly in *CTS* means identifying an abnormal period or periods. In this section, we introduce a segmenting approach to extract the main and common features of each period. The extracted information will be used as classification features that are used for model learning and anomaly detection. In addition, signal annotations (e.g., *Normal* and *Abnormal*) are attached to each period based on the original labels of the corresponding *PTS*. We will first give the concept of a summary of a period.

Definition 4.5. Given a CTS that has been separated into D periods, a *summary* of a period $pd_i = (v_{i_1}, \ldots, v_{i_m}), 1 \le i \le D$ is a vector $seg_i = (h_i^{min}, t_i^{min}, h_i^{max}, t_i^{max}, h_i^{mea}, p_i^{minmax}, p_i^l)$, where h_i^{min} is the amplitude value of the point having minimum amplitude in period i: $h_i^{min} = min\{v_{i_k}; 1 \le k \le m\}$, and t_i^{min} is the time index of the point with minimum amplitude. If there are two points having the minimum amplitude, t_i^{min} is the time index of the first point. $h_i^{max} = max\{v_{i_k}\}$; t_i^{max} is the first point with maximum amplitude, $h_i^{mea} = \frac{1}{m}(\sum v_{i_k})$, $p_i^{minmax} = |t_i^{max} - t_i^{min}|$, and $p_i^l = t_{i_m} - t_{i_1}$.

We represent the segmented *CTS* as $STS = \{seg_1, \ldots, seg_n\}$. Each period corresponds to an annotation *ann* indicating the state of the period. In this article, we will only consider two states: *normal* and *abnormal*. Therefore, a *STS* is always associated with a series of annotations $A_{STS} = \{ann_1, \ldots, ann_n\}$.

For the supervised pattern recognition model, the original *PTS* has a set of labels to indicate the states of the disjoint subsequences of *PTS*, which are represented as $Lbs = \{lb_{(1)}, \ldots, lb_{(w)}\}, \forall lb_{(r)} = \{'N'(Normal),' Ab'(Abnormal)\}, 1 \le r \le w$. However, *Lbs* cannot be attached to the segmentations of the *PTS* directly because the periodic separation is independent from the labeling process. To determine the state of a segmentation, we introduce a logical-multiplying relation of two signals.

Rule 1. $ann = \otimes('Ab',' N') =' Ab'$ *and* $ann = \otimes('N',' N') =' N'$.

Assume that a period covers a subsequence that is labeled by two signals; if there exists an abnormal behavior in the subsequence, then based on rule 1, the behavior of the segmentation of the period is abnormal, and otherwise the period is a normal series. This label assignment rule can be extended to multiple labels: given a set of labels $Lbs = \{lb_1, \ldots, lb_r\}$, if $\exists lb_j =' Ab', 1 \le j \le r$, the value of *Lbs* is $'Ab'$, represented as $lbs = \otimes(lb_1, \ldots, lb_r) =' Ab'$, and if $\forall lb_j =' N'$, $lbs =' N'$.

According to the preceding discussion, the annotation of a period pd_i is determined by Algorithm 2.

Example 4.6. We present the segmentation and annotation of a period in Figure 5 to explain their processes more clearly. Figure 5 shows that pd_i does not involve any

ALGORITHM 2: Period annotation

Input: Period $pd_i = (v_{i1}, \ldots, v_{im})$, $1 \leq i \leq n$;
A series of labels $Lbs = (lb_1, \ldots, lb_r)$;
Output: An annotated $pd_i^{'}$;
$t_i^1 = NULL$: the time of the 1^{st} annotation in the period;
$t_i^{end} = NULL$: the time of the last annotation in the period;
if $\exists lb_j$ *that* $t_{(i-1)1} \leq t_{j-1} \leq t_{(i-1)m} < t_{i1} \leq t_j \leq t_{im}$ **then**
    $t_i^1 = t_j$;
end
if $\exists lb_k$ & $t_{i1} \leq t_k \leq t_{im}$ & $t_{(i+1)1} \leq t_{k+1} \leq t_{(i+1)m}$ **then**
    $t_i^{end} = t_k$;
end
if $t_i^1 \neq NULL \parallel t_i^{end} \neq NULL$ **then**
    if $t_i^1 = NULL$ **then**
        $t_i^1 =' N'$
    end
    if $t_i^{end} = NULL$ **then**
        $t_i^{end} =' N'$
    end
    $Lbs = Lbs\{t_i^1, \ldots, t_i^{end}\}$;
    $lbs = \otimes(Lbs)$;
else
    $lbs = Lbs\{t_{i+1}^1\}$;
end

label and the first label in $pd_{i+1}$is $lb_1 = N$, so $lb_{pd_i} =' N'$. lb_2 is "*Ab*," and hence pd_{i+1} is annotated as *Ab*.

4.1.3. Classification-Based Anomaly Detection and Prediction: ADPC. From Example 4.6, each period of a *PTS* is summarized by seven features of the period: (h_i^{min}, t_i^{min}, h_i^{max}, t_i^{max}, h_i^{mea}, p_i^{minmax}, p_i^l). Using these seven features to abstract a period can significantly reduce the computational complexity in a classification process. In the next section, we validate the proposed anomaly detection framework with various classification methods on the basis of different ECG datasets.

5. EXPERIMENTAL EVALUATION

Our experiments are conducted in four steps. The first step is to remove the noises and compress the raw ECG time series by utilizing the DP algorithm, and to represent each inflexion in the perceived *CTS* as an FV (see Definition 4.4). Second, the *k*-means++ clustering algorithm is applied to the series of FVs of the *CTS*, and the clustering result is validated by silhouette values. Based on the mean silhouette value of each cluster, a period cluster is chosen and the *CTS* is periodically separated to a set of consistent segments. Third, each segment is summarized by the seven features (see Example 4.6). Finally, a normal pattern of the time series is constructed, and anomalies are detected by utilizing classification tools on the basis of the seven features.

We validate the proposed framework on the basis of eight ECG datasets [Goldberger et al. 2000], which are summarized in Table II, where "V" represents PVC, "A" represents atrial premature ventricular, and "S" represents supraventricular premature beat. Apart from the *aftdb* dataset, each time series is separated into a series of subsequences that are labeled by the dataset provider. We give the number of abnormal subsequences (#ofAbnor) and the number of normal subsequences (#ofNor) of each time series in Table II.

Table II. ECG Datasets Used in Experiments

Datasets	Abbr.	#ofSamples	AnomalyTypes	#ofAbnor	#ofNor
AHA0001	ahadb	899,750	V	115	2,162
SupraventricularArrhythmia800	svdb	230,400	S & V	75	1,846
SuddenCardiacDeathHolter30	sddb	22,099,250	V	38	5,743
MIT-BIH Arrhythmia100	mitdb	650,000	A & V	164	2,526
MIT-BIH Arrhythmia106	mitdb06	650,000	A & V	34	2,239
MGH/MF Waveform001	mgh	403,560	S & V	23	776
MIT-BIH LongTerm14046	ltdb	10,828,800	V	000	000
AF TerminationN04	aftdb	7,680	NA	NA	NA

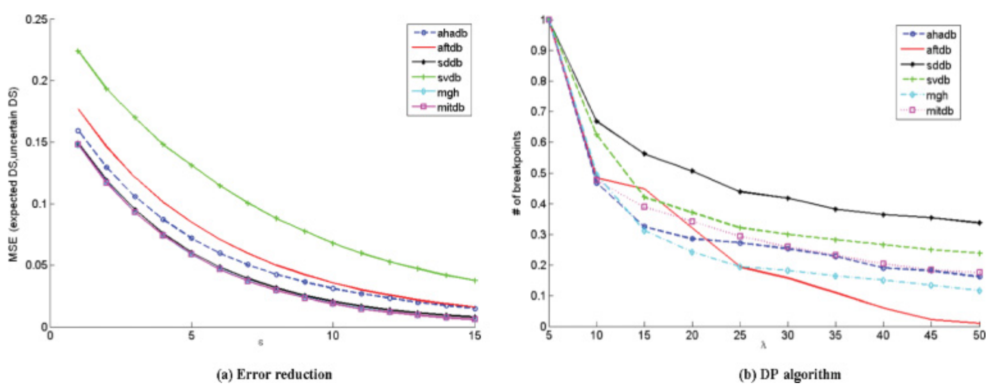

Fig. 6. MSE of noise reduction and monotonically decreasing number of breakpoints in terms of λ of the DP algorithm.

Our experiment is conducted on a 32-bit Windows system, with 3.2GHz CPU and 4GB RAM. The ECG datasets are downloaded to a local machine using the WFDB toolbox [Silva and Moody 2014; Goldberger et al. 2000] for 32-bit MATLAB. We use the 10-fold cross-validation method to process the datasets.

The metrics used for evaluating the final anomaly classification results include the following:

(1) *Accuracy (acc)*: $(TP + TN)$ / Number of all classified samples;
(2) *Sensitivity (sen)*: TP / $(TP + FN)$;
(3) *Specificity (spe)*: TN / $(FP + TN)$;
(4) *Prevalence (pre)*: TP / Number of all samples.
(5) *Fmeasure (fmea)*: $2 * \frac{precision * recall}{precision + recall}$, where $recall = sen$, $precision = \frac{TP}{TP+FP}$
(TP, true positive; TN, true negative; FP, false positive; and FN, false negative).
Details of the experiments are illustrated in the following sections.

5.1. Error Detection and Time Series Compression

At first, we design an experiment for noisy reduction in an uncertain time series. We use the synthetic uncertain data: we plant the additive Gaussian white noise to six time series in Table II: ahadb, aftdb, sddb, svdb, mgh, and mitdb. The performance of the error reduction is evaluated by the mean squared error (MSN) between the six real time series and the synthetic uncertain time series. We use different signal-to-noise ratio ϵ (from 1 to 15) to see the change of the MSE value based on the wavelet denoising approach. The experiment result is shown in Figure 6(a). We can see that the MSEvalues of the six time series are decreasing from 0.25 to 0 when the value of the signal-to-noise ratio is increasing from 0 to 15.

Table III. Decreasing Monotonicity Degree of Six Datasets in Terms of the Value of λ

	ahadb	svdb	sddb	mitdb	mgh	aftdb
λ	100	100	100	100	100	100

Table IV. Endpoint Stability of Six Datasets and Perturbations

	ahadb	svdb	sddb	mitdb	mgh	aftdb
Shifting length	10,000	10,000	10,000	10,000	10,000	100
S	100	99.8988	99.9955	99.9725	99.9348	99.9351

The refined time series (whose errors have been reduced) will be compressed by the DP algorithm. We use the approach proposed by the work of Rosin [2003] to assess the stability of the DP compression algorithm under the variations of the change of the scale parameter and the perturbation of data. The former is measured by using a monotonicity index, and the latter is quantified by a breakpoint stability index.

The monotonicity index is used to measure the monotonically decreasing or increasing trend of the number of breakpoints when the values of scale parameters of a polygonal approximation algorithm are changed. For the DP algorithm, if the value of the scale parameter λ is increasing, the number of the produced breakpoints of the time series will be decreasing, and vice versa. The decreasing monotonicity index is defined as $M_D = (1 - \frac{T_+}{T_-}) \times 100$, and the increasing monotonicity index is $M_I = (1 - \frac{T_-}{T_+}) \times 100$, where $T_- = -\sum_{\forall \Delta v_i < 0} \Delta v_i / h_i$, $T_+ = \sum_{\forall \Delta v_i > 0} \Delta v_i / h_i$, and $h_i = \frac{v_i + v_{i-1}}{2}$. Both M_D and M_I are in the range [0, 100], and their perfect scores are 100. We test the decreasing monotonicity degrees for the datasets *ahadb*, *svdb*, *sddb*, *mitdb*, *mgh*, and *aftdb* in terms of different values of λ for the DP algorithm. We set $\lambda = 5, 10, 15, 20, 25, 30, 35, 40, 45, 50$ to conduct DP compression. From Table III and Figure 6(b), we can see that the numbers of breakpoints are also 100% decreasing in terms of the increasing λ.

The breakpoint stability index is defined as the shifting degree of breakpoints when deleting increasing amounts from the beginning of a time series. We use the endpoint stability to test the breakpoint stability for fixed parameter settings : $\lambda = 10$ for the DP algorithm. The endpoint stability measurement is defined as $S = (1 - \frac{1}{m} \sum_d \sum_b \frac{s_b^d}{n_d l_d})$, where m is the level number of deletion, d is the dth level, s_b^d is the shifting pixels at breakpoint b, l_d is the length of the remaining time series, and n_d is the number of breakpoints after the dth deletion. Table IV shows the deletion length of each running circle and the stability degree of each time series. We iteratively delete $10,000$ samples from the beginning of the remaining *ahadb* time series and conduct the DP algorithm based on the new time series. The positions of the identified breakpoints in each running circle are compared to the positions of the breakpoints identified in the whole *ahadb*. From Table IV, we can see that each time series is of high stability (i.e., values of S) when conducting the uncertainty detection procedure and the DP algorithm with fixed scale parameters.

5.2. Compressed Time Series Representation

Based on Figure 6, we set $\lambda = 10$ for time series compression. We then compare three methods of period point representation: (1) inflexions in *CTS* are represented by FVs, (2) inflexions are represented by angles (Angle) of peak points [Huang et al. 2014], and (3) inflexions are represented by valley points (Valley) [Tang et al. 2007]. Valley points are points in a *PTS*, which have values less than an upper bound value (represented as U). U is initially specified by users and will be updated as time evolves. The update procedure is defined as $U_b = \alpha(\sum_{i=1}^{N} V_i)/N$, where N is the number of past valley points

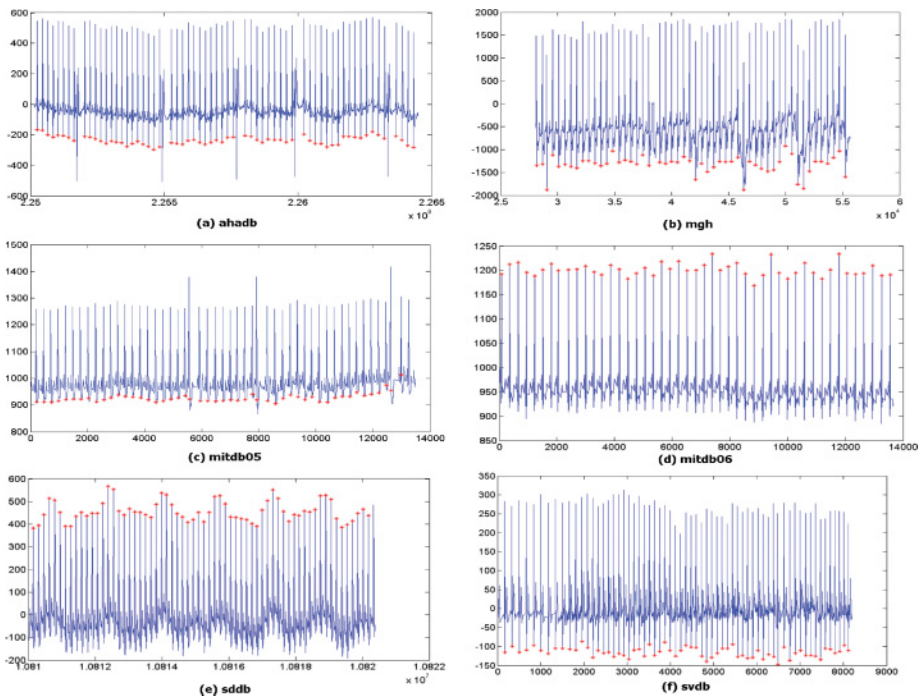

Fig. 7. Period point identification of four datasets based on FVs.

Table V. Silhouette Values of Six Datasets

Dataset	Silhouette Values						
	Mean	Cluster1 (C1)	C2	C3	C4	C5	C6
ahadb	0.8253	0.4479	0.8502	0.9824	0.9891	0.9381	NA
svdb	0.6941	0.9792	0.6551	0.9703	0.5463	0.5729	0.959
sddb	0.772	0.6888	0.5787	0.965	0.9727	0.6971	0.7529
mitdb	0.9373	0.9877	0.7442	0.9898	0.9711	0.5854	0.3754
mitdb06	0.7339	0.7317	0.8998	0.609	0.8577	0.8669	NA
ltdb	0.9149	0.9164	0.8381	0.9739	0.9079	0.8975	NA
mgh	0.8253	0.4479	0.8502	0.9824	0.9891	0.9381	NA

and α is an outlier control factor that is determined and adjusted by experts. As stated by Tang et al. [2007], the best values of initial upper bound and α in ECGare 50mmHg and 1.1. The perceived FV sets, angle sets, and valley point sets are passed to the next step, in which points are clustered and the period points of the *CTS* are identified. Each period is then segmented using the proposed segmentation method (see Example 4.6). Finally, linear discriminant analysis (LDA) and naive Bayes (NB) classifiers are applied for sample classification and anomaly detection. Figure 7 shows the identified period points using the FV-based method for four datasets: *ltdb*, *sddb*, *svdb*, and *ahadb*. From Figure 7, we can see that for each dataset, the FV-based method successfully identifies a set of periodic points that can separate the *CTS* in a stable and consistent manner.

Table V presents the silhouette values of clustering the inflexions in the *CTSs* of seven time series, where the *Mean* column refers to the mean silhouette value of a dataset clustering, and the values in columns C1 through C6 are the mean silhouette values of each cluster after clustering a dataset. An *NA* in the sixth column means that the inflexions in the corresponding datasets are clustered into five groups, which

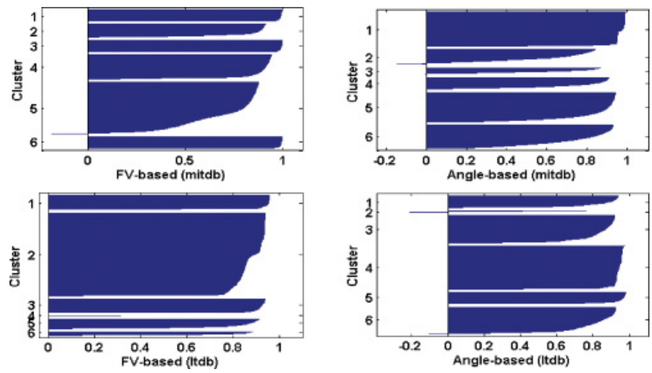

Fig. 8. Silhouette value comparison between the FV-based clustering method and the angle-based clustering method for the *mitdb* and *ltdb* datasets.

present the best clustering performance in this dataset. From Definition 4.5, we know that if the silhouette values in a cluster is close to 1, the cluster includes a set of points having similar patterns. On the other hand, if the silhouette values in a cluster are significantly different from each other or have negative values, the points in the cluster have very different patterns with each other or they are more close to the points in other clusters. Table V shows that for each of the seven datasets, the mean silhouette values of the overall clustering result and each of the individual clusters are higher than 0.4 ($\eta = 0.4$ in Algorithm 1). The best silhouette value of an individual cluster in each dataset is close to or higher than 0.9 ($\xi = 0.8$ in Algorithm 1). In addition, for each dataset, we select the points in the cluster with the highest silhouette value as the period points. For example, for dataset *ahadb*, points in cluster 4 are selected as period points.

Figure 8 presents the silhouette values of clustering the inflexions in the *CTSs* of *mitdb* and *ltdb* time series. From this figure, we can see that for both the *mitdb* and *ltdb* datasets, FV-based clustering results in fewer negative silhouette values in all clusters, and the values in each cluster are more similar to each other compared to the angle-based clustering. We also come to a similar conclusion by examining their mean silhouette values. The mean silhouette values of FV-based clustering for *mitdb* (corresponding to Figure 8(a)) is 0.9373, whereas the angle-based clustering (Figure 8(b)) is 0.7461; the mean values for *ltdb* are 0.9149 and 0.8155 (Figure 8(c) (d), respectively.

Figure 9 compares the average classification performance on the basis of four datasets using four classifiers: LDA, NB, decision tree (DT), and AdaBoost (Ada) with 100 ensemble members. From Figure 9, we can see that the classifiers based on the FV periodic separating method have the best performance in terms of the four datasets (i.e., the highest accuracy, sensitivity, f-measure, and prevalence). In the case of LDA and DT, the valley-based periodic separating method has the worst performance, whereas in the cases of NB and Ada, valley-based methods perform better than angle-based methods.

5.3. Evaluation of Classification Based on Summarized Features

This section describes the experimental design and the performance evaluation of classification based on the summarized features. The experiment is conducted on seven datasets: *ahadb*, *svdb*, *sddb*, *mitdb*, *mitdb*06, *mgh*, and *ltdb*. From the previous sections, we know that the seven time series have been compressed and the period segmenting points have been identified (see Table V). The segments of each of the time series

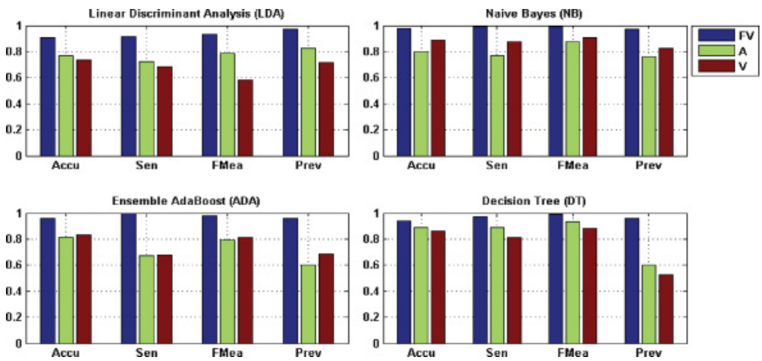

Fig. 9. Average performance comparison of four classifiers (LDA, NB, ADA, DT) based on FV, angle-based (A) and valley point–based (V) periodic separating methods.

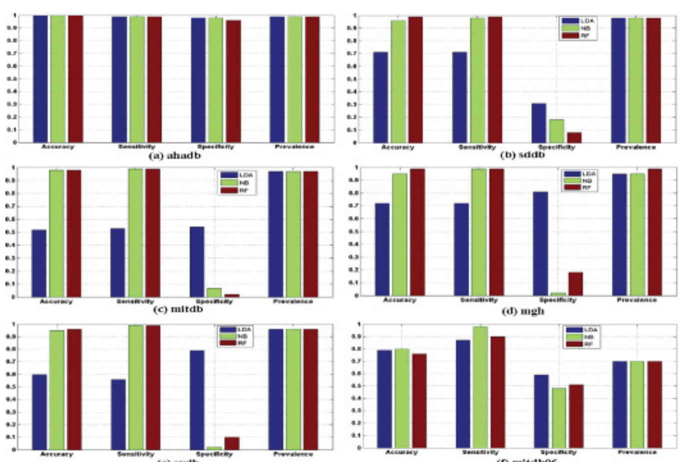

Fig. 10. Classification performance of six datasets based on the summarized features using classification methods of LDA, RF, and NB.

are classified by using three classification tools: random forest with 100 trees (RF), LDA, and NB. We use matrices of *acc*, *sen*, *spe*, and *pre* to validate the classification performance.

The classification performance is shown in Figure 10, which compares the performance of classification methods LDA, NB, and RF based on datasets (a) *ahadb*, (b) *sddb*, (c) *mitdb*, (d) *mgh*, (e) *svdb*, and (f) *mitdb06*. From the figure, we can see that for all six datasets, the performances of NB and RF are better than the performance of LDA based on the selected features. The accuracy and sensitivity of NB and RF are higher than 80% for each of the datasets. Their prevalence values are greater than 90% for the first five datasets (a through e). However, we can also see that the feature values of LDA are always higher than the feature values of the other two methods.

5.4. Performance Evaluation of Other Classification Methods Based on Summarized Features

In this section, we design an experiment to evaluate the performance of the proposed time series segmentation method. Experimental results on the basis of five datasets (i.e., *mitdb*, *ltdb*, *ahadb*, *sddb*, and *svdb*) are presented in this section. We carry out the experiment by the following steps. First, the raw time series are compressed by the

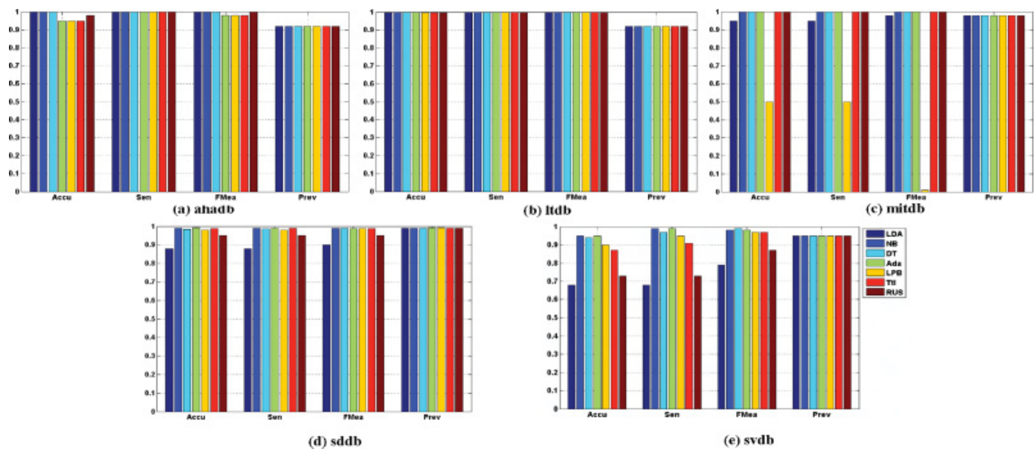

Fig. 11. Performance of seven classifiers (LDA, NB, DT, Ada, LPB, Ttl, and RUS) based on the proposed period identification and segmentation methods on five datasets: ahadb (a), ltdb (b), mitdb (c), sddb (d), and svdb (e).

DP algorithm and periodically separated by the FV-based period identification method. Second, each period is summarized by the proposed period summary method (see Definition 4.7) and is annotated by the annotation process (see Step 3 in Section 4.1.2). The classification methods used in this experiment include LDA, NB, DT, and a set of ensemble methods: AdaBoost (Ada), LPBoost (LPB), TotalBoost (Ttl), and RUSBoost (RUS). The classification performance is validated by four benchmarks: acc, sen, $fmea$, and $prev$.

Figure 11 shows the evaluated results of the classifier performance based on the proposed period identification and segmentation method. From this figure, we can see that the accuracy values of classification based on the five datasets are greater than 90%, except the cases of LPB with *mitdb*, LDA with *sddb*, LDA with *svdb*, and RUS with *svdb*. Some of them are of greater than 98% accuracy. The sensitivity of classification based on the datasets of *ahadb*, *ltdb*, and *mitdb* are closing to 100%. The sensitivity based on the datasets of *sddb* and *svdb* are greater than85%. The f-measure rates of classification based on *ahadb*, *ltdb*, *mitdb*, and *sddb* are higher than 95%. The f-measure rates of RUS and LDA based on *mitdb* and *svdb* are less than 80%, but the f-measure of other classifiers based on these two datasets are all higher than 80%, and some of them are closing to 100%. The prevalence rates of classification on the basis of the five datasets are greater than 90%.

6. CONCLUSIONS

In this article, we have introduced a framework of detecting anomalies in uncertain pseudoperiodic time series. We formally defined pseudoperiodic time series (*PTS*) and identified three types of anomalies that may occur in a *PTS*. We focused on local anomaly detection in *PTS* by using classification tools. The uncertainties in a *PTS* are preprocessed by an inflexion detecting procedure. By conducting DP-based time series compression and feature summarization of each segment, the proposed approach significantly improves the time efficiency of time series processing and reduces the storage space of the data streams. One problem of the proposed framework is that the silhouette coefficient–based clustering evaluation is a time-consuming process. Although the compressed time series contains much fewer data points than the raw time series, it is necessary to develop a more efficient evaluation approach to find the optimal clusters of data stream inflexions. In the future, we are going to find a more time-efficient way to recognize the patterns of a *PTS*. In addition, we will do more testing based on other datasets to further validate the performance of the method. Correctingfalse-detected inflexions and detecting global anomalies in an uncertain *PTS* will be the main target of our next research work.

REFERENCES

P. M. Agante and J. P. M. de Sa. 1999. ECG noise filtering using wavelets with soft-thresholding methods. In *Proceedings of the 1999 Computers in Cardiology Conference*. 535–538. DOI:http://dx.doi.org/10.1109/CIC.1999.826026

Charu C. Aggarwal. 2009. On high dimensional projected clustering of uncertain data streams. In *Proceedings of the IEEE 25th International Conference on Data Engineering (ICDE'09)*. IEEE, Los Alamitos, CA, 1152–1154. DOI:http://dx.doi.org/10.1109/ICDE.2009.188

Charu C. Aggarwal and Philip S. Yu. 2008. A framework for clustering uncertain data streams. In *Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE'08)*. IEEE, Los Alamitos, CA, 150–159. DOI:http://dx.doi.org/10.1109/ICDE.2008.4497423

Ian F. Akyildiz, Dario Pompili, and Tommaso Melodia. 2005. Underwater acoustic sensor networks: Research challenges. *Ad Hoc Networks* 3, 3, 257–279. DOI:http://dx.doi.org/10.1016/j.adhoc.2005.01.004

Arvind Arasu, Shivnath Babu, and Jennifer Widom. 2003. *The CQL Continuous Query Language: Semantic Foundations and Query Execution*. Technical Report 2003-67. Stanford InfoLab. http://ilpubs.stanford.edu:8090/758/

David Arthur and Sergei Vassilvitskii. 2007. K-means++: The advantages of careful seeding. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'07)*. 1027–1035. http://dl.acm.org/citation.cfm?id=1283383.1283494.

Johannes Aßfalg, Hans-Peter Kriegel, Peer Kröger, and Matthias Renz. 2009. Probabilistic similarity search for uncertain time series. In *Scientific and Statistical Database Management.*Lecture Notes in Computer Science, Vol. 5566. Springer, 435–443. DOI:http://dx.doi.org/10.1007/978-3-642-02279-1_31

Jianjun Chen, David J. DeWitt, Feng Tian, and Yuan Wang. 2000. NiagaraCQ: A scalable continuous query system for Internet databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*. 379–390. http://doi.acm.org/10.1145/342009.335432.

D. L. Swain, M. A. Friend, G. J. Bishop-Hurley, R. N. Handcock, and T. Wark. 2011. Tracking livestock using global positioning systems – are we still lost? *Animal Production Science* 51, 167–175.

Michele Dallachiesa, Besmira Nushi, Katsiaryna Mirylenka, and Themis Palpanas. 2012. Uncertain time-series similarity: Return to the basics. *Proceedings of the VLDB Endowment* 5, 11, 1662–1673. DOI:http://dx.doi.org/10.14778/2350229.2350278

David Leigh Donoho. 1995. De-noising by soft-thresholding. *IEEE Transactions on Information Theory* 41, 3, 613–627. DOI:http://dx.doi.org/10.1109/18.382009

David H. Douglas and Thomas K. Peucker. 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica* 10, 2, 112–122.

Philippe Esling and Carlos Agon. 2012. Time-series data mining. *ACM Computing Surveys* 45, 1, Article No. 12. DOI:http://dx.doi.org/10.1145/2379776.2379788

Victor A. Folarin, Patrick J. Fitzsimmons, and William B. Kruyer. 2001. Holter monitor findings in asymptomatic male military aviators without structural heart disease. *Aviation, Space, and Environmental Medicine* 72, 9, 836–838. http://www.ncbi.nlm.nih.gov/pubmed/11565820.

Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. 2000. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation* 101, 23, e215–e220.

Aslak Grinsted, John C. Moore, and Svetlana Jevrejeva. 2004. Application of the cross wavelet transform and wavelet coherence to geophysical time series. *Nonlinear Processes in Geophysics* 11, 5-6, 561–566. DOI:http://dx.doi.org/10.5194/npg-11-561-2004

Yu Gu, Andrew McCallum, and Don Towsley. 2005. Detecting anomalies in network traffic using maximum entropy estimation. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC'05)*. 32. http://dl.acm.org/citation.cfm?id=1251086.1251118.

Joachim Gudmundsson, Marc van Kreveld, and Bettina Speckmann. 2007. Efficient detection of patterns in 2D trajectories of moving points. *GeoInformatica* 11, 2, 195–215. DOI:http://dx.doi.org/10.1007/s10707-006-0002-z

Şule Gündüz and M. Tamer Özsu. 2003. A Web page prediction model based on click-stream tree representation of user behavior. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 535–540.

John A. Hartigan and Manchek A. Wong. 1979. Algorithm AS 136: A K-means clustering algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 28, 1, 100–108. http://www.jstor.org/stable/2346830.

Jing He, Yanchun Zhang, and Guangyan Huang. 2012. Exceptional object analysis for finding rare environmental events from water quality datasets. *Neurocomputing* 92, 0, 69–77. DOI:http://dx.doi.org/10.1016/j.neucom.2011.08.036 Data Mining Applications and Case Study.

John Hershberger and Jack Snoeyink. 1994. An O(Nlogn) implementation of the Douglas-Peucker algorithm for line simplification. In *Proceedings of the 10th Annual Symposium on Computational Geometry (SCG'94)*. ACM, New York, NY, 383–384. DOI:http://dx.doi.org/10.1145/177424.178097

Guangyan Huang, Yanchun Zhang, Jie Cao, Michael Steyn, and Kersi Taraporewalla. 2014. Online mining abnormal period patterns from multiple medical sensor data streams. *World Wide Web* 17, 4, 569–587. DOI:http://dx.doi.org/10.1007/s11280-013-0203-y

Ruoyi Jiang, Hongliang Fei, and Jun Huan. 2011. Anomaly localization for network data streams with graph joint sparse PCA. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11)*. ACM, New York, NY, 886–894. DOI:http://dx.doi.org/10.1145/2020408.2020557

Eamonn Keogh, Jessica Lin, and Ada Fu. 2005. HOT SAX: Efficiently finding the most unusual time series subsequence. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM'05)*. IEEE, Los Alamitos, CA, 226–233. DOI:http://dx.doi.org/10.1109/ICDM.2005.79

Eamonn J. Keogh, Selina Chu, David Hart, and Michael Pazzani. 2004. Segmenting time series: A survey and novel approach. In *Data Mining in Time Series Databases*, M. Last, A. Kandel, and H. Bunke (Eds.). Series in Machine Perception and Artificial Intelligence, Vol. 57. World Scientific Publishing Company, 1–22.

Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. 2007. Trajectory clustering: A partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD'07)*. ACM, New York, NY, 593–604. DOI:http://dx.doi.org/10.1145/1247480.1247546

Daniel Lemire. 2007. A better alternative to piecewise linear time series segmentation. In *Proceedings of the 7th SIAM International Conference on Data Mining (SDM'07)*. 545–550.

Carson Kai-Sang Leung and Boyu Hao. 2009. Mining of frequent item-sets from streams of uncertain data. In *Proceedings of the IEEE 25th International Conference on Data Engineering (ICDE'09)*. IEEE, Los Alamitos, CA, 1663–1670. DOI:http://dx.doi.org/10.1109/ICDE.2009.157

Matthew N. Levy and Achilles J. Pappano. 2007. *Cardiovascular Physiology*. Mosby Elsevier.

Bai Ling Zhang, Yanchun Zhang, and Rezaul K. Begg. 2009. Gait classification in children with cerebral palsy by Bayesian approach. *Pattern Recognition* 42, 4, 581–586. DOI:http://dx.doi.org/10.1016/j.patcog.2008.09.025

X. Liu, X. Wu, H. Wang, R. Zhang, J. Bailey, and K. Ramamohanarao. 2010. Mining distribution change in stock order streams. In *Proceedings of the IEEE 26th International Conference on Data Engineering (VLDB'04)*. 105–108. DOI:http://dx.doi.org/10.1109/ICDE.2010.5447901

Melanie Manning and Louanne Hudgins. 2010. Array-based technology and recommendations for utilization in medical genetics practice for detection of chromosomal abnormalities. *Genetics in Medicine* 12, 11, 742–745.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.). Vol. 14. MIT Press, Cambridge, MA, 849–856.

Themis Palpanas, Michail Vlachos, Eamonn Keogh, and Dimitrios Gunopulos. 2008. Streaming time series summarization using user-defined amnesic functions. *IEEE Transactions on Knowledge and Data Engineering* 20, 7, 992–1006. DOI:http://dx.doi.org/10.1109/TKDE.2007.190737

Jianzhong Qi, Rui Zhang, Kotagiri Ramamohanarao, Hongzhi Wang, Zeyi Wen, and Dan Wu. 2015. Indexable online time series segmentation with error bound guarantee. *World Wide Web* 18, 2, 359–401. DOI:http://dx.doi.org/10.1007/s11280-013-0256-y

Douglas Reynolds. 2009. Gaussian mixture models. In *Encyclopedia of Biometrics*, S. Z. Li and A. Jain (Eds.). Springer, 659–663. DOI:http://dx.doi.org/10.1007/978-0-387-73003-5_196

Paul L. Rosin. 2003. Assessing the behaviour of polygonal approximation algorithms. *Pattern Recognition* 36, 2, 505–518. DOI:http://dx.doi.org/10.1016/S0031-3203(02)00076-6 Biometrics.

Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics* 20, 0, 53–65. DOI:http://dx.doi.org/10.1016/0377-0427(87)90125-7

Smruti R. Sarangi and Karin Murthy. 2010. DUST: A generalized notion of similarity between uncertain time series. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'10)*. ACM, New York, NY, 383–392. DOI:http://dx.doi.org/10.1145/1835804.1835854

Galit Shmueli and Howard Burkom. 2010. Statistical challenges facing early outbreak detection in biosurveillance. *Technometrics* 52, 1, 39–51.

Ikaro Silva and George Moody. 2014. An open-source toolbox for analysing and processing physionet databases in MATLAB and octave. *Journal of Open Research Software* 2, 1, e27.

Lv-an Tang, Bin Cui, Hongyan Li, Gaoshan Miao, Dongqing Yang, and Xinbiao Zhou. 2007. Effective variation management for pseudo periodical streams. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD'07)*. ACM, New York, NY, 257–268. DOI:http://dx.doi.org/10.1145/1247480.1247511

Mahbod Tavallaee, Natalia Stakhanova, and Ali Akbar Ghorbani. 2010. Toward credible evaluation of anomaly-based intrusion-detection methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 40, 5, 516–524. DOI:http://dx.doi.org/10.1109/TSMCC.2010.2048428

Thanh T. Tran, Liping Peng, Yanlei Diao, Andrew Mcgregor, and Anna Liu. 2012. CLARO: Modelling and processing uncertain data streams. *VLDB Journal* 21, 5, 651–676. DOI:http://dx.doi.org/10.1007/s00778-011-0261-7

William Wilson, Phil Birkin, and Uwe Aickelin. 2008. The motif tracking algorithm. *International Journal of Automation and Computing* 5, 1, 32–44.

Zhenghua Xu, Rui Zhang, Ramamohanarao Kotagiri, and Udaya Parampalli. 2012. An adaptive algorithm for online time series segmentation with error bound guarantee. In *Proceedings of the 15th International Conference on Extending Database Technology (EDBT'12)*. ACM, New York, NY, 192–203. DOI:http://dx.doi.org/10.1145/2247596.2247620

Yu Zheng, Xing Xie, and Wei-Ying Ma. 2010. GeoLife: A collaborative social networking service among user, location and trajectory. *IEEE Data Engineering Bulletin* 33, 2, 32–39. http://sites.computer.org/debull/A10june/geolife.pdf.

Yunyue Zhu and Dennis Shasha. 2002. StatStream: Statistical monitoring of thousands of data streams in real time. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02)*. 358–369. http://dl.acm.org/citation.cfm?id=1287369.1287401.

Received November 2014; revised July 2015; accepted July 2015