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 paper, we present a novel framework that supports anomaly detection in uncertain data streams. The proposed framework adopts an efficient uncertainty pre-processing procedure to identify and eliminate uncertainties in data streams. Based on the corrected 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 a number of 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

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 or click-streams are typical data streams. Other traditional and emerging applications include wireless sensor networks (WSN) 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 behaviour monitoring (with mobile sensor nodes), such as water pollution detection [He et al. 2012] based on water sensor data, agricultural management and cattle moving habits [CSIRO 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 streams application. Here, anomalies or outliers or exceptions often refer to the patterns in data streams that deviate expected normal behaviours. Thus, anomaly detection is a dynamic process of finding abnormal behaviours from given data streams. For example, in medical monitoring applications, a human electrocardiogram (ECG) (vital signs) and other treatments and

Correspondence author’s addresses: Yanchun Zhang, School of Computer Science, Fudan University, Shanghai 200433, China; and Centre for Applied Informatics, Victoria University, VIC, 3012, Australia

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© YYYY ACM 1533-5399/YYYY/01-ARTA $15.00
DOI: http://dx.doi.org/10.1145/000000.0000000

ACM Transactions on Internet Technology, Vol. V, No. N, Article A, Publication date: January YYYY.
measurements are typical data streams that appear in a form of periodic patterns. That is, 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 behaviours 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 heart beat 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 a number of 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 it may remove useful information that can correct the error data.

This paper focuses on anomaly detection in uncertain pseudo periodic time series. A pseudo periodic 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 pseudo periodic 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.

In order to deal with the issue of anomaly detection in uncertain data streams, we propose a supervised classification framework for detecting anomalies in uncertain pseudo periodic time series, which comprises four components: a uncertainty identification and correction component (UICC), a time series compression component (TSCC), a period segmentation and summarization component (PSSC), and a classification and anomaly detection component (CADC). First, UICC processes a time series to remove uncertainties from the time series. Then TSCC compresses the processed raw time series to an approximate time series. Afterwards 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 pseudo periodic time series, together with a novel set of techniques for segmenting and extracting the main features of a time series. The procedure of pre-processing 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 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 data sets. 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 paper, we evaluate the proposed anomaly detection framework on ECG time series. However, due to the generic nature of features of pseudo periodic 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 time series mining in different areas.

The structure of this paper 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 this paper.

2. RELATED WORK

We analyse 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 area, such as intrusion detection [Tavallaee et al. 2010], disease detection in medical sensor streams [Manning and Hudgins 2010], and biosurveillance [Shmueli and Burkom 2010]. Zhang et al. [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. [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. [Xu et al. 2012] improved the traditional PLA based techniques by guaranteeing an error bound on each data point to maximally compact time series. Daniel [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. [Palpanas et al. 2008] defined user-oriented amnesic functions for decreasing the confidence of older information continuously.

However, the approaches mentioned above are not designed to process and adapt to the area of pseudo periodic 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 sub-sequences. However, this 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 [an Tang et al. 2007]. Other examples of
Table I. Frequently Used Symbols

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TS$</td>
<td>A time series</td>
</tr>
<tr>
<td>$p_i$</td>
<td>The $i$th point in a $TS$</td>
</tr>
<tr>
<td>$SS$</td>
<td>A subsequence</td>
</tr>
<tr>
<td>$PTS$</td>
<td>A pseudo periodic time series</td>
</tr>
<tr>
<td>$Q$</td>
<td>A set of period points in a $PTS$</td>
</tr>
<tr>
<td>$pd$</td>
<td>A period in a $PTS$</td>
</tr>
<tr>
<td>$CTS$</td>
<td>A compressed $PTS$</td>
</tr>
<tr>
<td>diff$i$</td>
<td>$diff_1 = t_i - t_{i-1}$, $diff_2 = t_{i+1} - t_i$</td>
</tr>
<tr>
<td>vec$i$</td>
<td>A feature vector of point $p_i$</td>
</tr>
<tr>
<td>sil$(p_i)$</td>
<td>Silhouette value of point $p_i$</td>
</tr>
<tr>
<td>sim$(p_i, p_j)$</td>
<td>Euclidean distance based similarity between points $p_i$ and $p_j$</td>
</tr>
<tr>
<td>$C$</td>
<td>A set of clusters</td>
</tr>
<tr>
<td>m$sil(C)$</td>
<td>Mean silhouette value of a cluster $C$</td>
</tr>
<tr>
<td>seg$i$</td>
<td>A summary of a period</td>
</tr>
<tr>
<td>STS</td>
<td>A segmented $CTS$</td>
</tr>
<tr>
<td>ASTS</td>
<td>A set of annotations</td>
</tr>
<tr>
<td>Lbs</td>
<td>A set of labels indicating the states</td>
</tr>
<tr>
<td>lb$(i)$</td>
<td>The $i$th label in Lbs$_{PTS}$</td>
</tr>
</tbody>
</table>

segmenting pseudo periods include an peak-point-based clustering method and valley-point-based method [Huang et al. 2014] [an 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 sub-sequences. 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 pseudo periodic data streams.

**Uncertainty processing in data streams:** Most data streams coming from real-world sensor monitoring are inherently noisy and uncertainties. A lot of work has concentrated on the modelling of uncertain data streams [Aggarwal and Yu 2008] [Aggarwal 2009] [Leung and Hao 2009]. Dallachiesa et al. [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 [Affalag et al. 2009]. Tran et al. [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. In this work, we model a continuous time series as a discrete time series by identifying the critical points in a time series, and introduce a novel method of detecting and correcting the missing inflexions based on the angles between points.

3. PROBLEM SPECIFICATION AND FRAMEWORK DESCRIPTION

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 pseudo periodic patterns. The symbols frequently used in this paper are summarized in Table I.

3.1. Problem definition

**Definition 3.1.** A **time-series** $TS$ is an ordered real sequence: $TS = (v_1, \cdots, v_n)$, 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 above definition, we define subsequence of a $TS$ as below.
Definition 3.2. For time series $TS$, if $SS \subset TS$ comprises $m$ consecutive points: $SS = \{v_1, \ldots, v_m\}$, we say that $SS$ is a subsequence of $TS$ with length $m$, represented as $SS \subset TS$.

Definition 3.3. A pseudo periodic 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 condition that

(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 $\xi_1$ is a small value.

(2) let $s_1 = (v_{p_1}, v_{(p_1)+1}, \ldots, v_{p_{i+1}}) \subset PTS$, and $s_2 = (v_{p_{i+1}}, v_{(p_{i+1})+1}, \ldots, v_{p_{i+2}}) \subset PTS$, then $d_{sim}(s_1, s_2) \leq \xi_2$, where $d_{sim}()$ calculates the dis-similarity between $s_1$ and $s_2$, and $\xi_2$ is a small value. $d_{sim}()$ 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 $pd \subset PTS$, and $pd = (v_{p_1}, v_{(p_1)+1}, \ldots, v_{p_{i+1}}) \forall v_{p_i} \in Q$, 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 behaviour of a subsequence $SS$, where $m = |SS|$ and $m \in [1, |PTS|/2]$. This behaviour indicates the normal situation of an event.

Based on the above 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 $N = (v_1, \ldots, v_m) \subset PTS$, a local anomaly ($L$) is defined as:

Definition 3.6. Assume $L = (v_{l_1}, \ldots, v_{l_{n}}) \subset PTS$, $L$ is a local anomaly if either of the two conditions in Definition 3.3 is broken (shown as below (1)), and at the same time satisfies the following two conditions (below (3)):

(1) $\triangle_N - \triangle_L > \xi_1$ or $d_{sim}(N, L) > \xi_2$;
(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. Fig 1(a) shows two examples of pseudo periodic time series and their local anomalies. Fig 1(b) shows a premature ventricular contraction signal in an ECG stream. A premature ventricular contraction (PVC) [Levy and Pappano 2007] is perceived as a "skipped beat". It can be easily distinguished from a normal heart beat when detected by the electrocardiogram. From Fig 1(a), the QRS and T waves of a PVC (indicated by $V$) are very different from the normal QRS and T (indicated by $N$). Fig 1(b) presents an example of premature atrial contractions (PACs) [Folarin et al. 2001]. A PAC is a premature heart beat 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 Fig 1 the second period (a PAC) is clearly shorter than the other periods.

3.2. Overview of the Anomaly Detection Framework for Uncertain Time Series Data

As mentioned previously, the proposed framework comprises four main components: an uncertainty identification and correction component (UICC), a time series compression component (TSCC), a period segmentation and summarization component (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 \textit{mitdb}. Fig[2] shows the processing progress of \textit{mitdb}. First, the raw \textit{mitdb} time series is an input to the UICC component. The TS1 in Fig[2] shows a subsequence of the raw \textit{mitdb}. The UICC identifies the inflexions (including missing inflexions) of \textit{mitdb}, and the raw \textit{mitdb} is transformed into an approximated time series that only consists of the identified inflexions (TS2 in Fig[2]). The TSCC component then further compresses the approximated \textit{mitdb}. The TS3 in Fig[2] 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 Fig[2] 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 next section, we introduce the framework and its four components in detail.

4. ANOMALY DETECTION IN UNCERTAIN PERIODIC TIME SERIES

4.1. Uncertainty Identification and Correction: UICC

In this section, we introduce the procedure of eliminating uncertainties of a \textit{PTS} caused by non-captured key-points of a \textit{PTS}, based on our previous work \cite{He et al. 2013}. We first introduce the definition of key-points of a time series.

\textbf{Definition 4.1.} Given a \textit{PTS} = (v_1, \ldots, v_n), if a point, p_i = v_i or v_n, is a turning point, then p_i is a key-point; or else, if \( \angle p_i = \pi - \angle p_j p_k \) and \( \angle p_i > \epsilon \), where \( \angle p_i \) is the angle between vectors \( \overrightarrow{p_j p_i} \) and \( \overrightarrow{p_k p_i} \), \( 2 \leq j < i < k \leq n \), \( \epsilon \) is a threshold, p_j and p_k are key-points, and for any point \( p_r, j < r < k \), \( \angle p_r \leq \epsilon \), then \( p_i \) is a key-point.

From the above definition, the core procedure to determine a point \( p_k \) as a key-point is based on the angles between \( p_{k-1} p_k \) and \( p_k p_{k+1} \) (i.e., \( \angle p_k = \pi - \angle p_{k-1} p_k p_{k+1} \)), given that \( p_{k-1} \) and \( p_{k+1} \) are both key-points. If \( \angle p_k \) is larger than a threshold value, and
the angles of all the other points between \( k - 1 \) and \( k + 1 \) are not larger than the threshold, then \( p_k \) is a key-point. However, if \( p_k \) is missing, we need to check at least four points: two key-points before and two key-points after \( p_k \) respectively. Therefore, we generally check four consecutive points at the same time. Combined with Fig.3, the detailed process is described below:

Given four consecutive points \( p_1 = v_1, p_2 = v_2, p_3 = v_3, \) and \( p_4 = v_4 \), where \( p_1 \) and \( p_4 \) are key-points, and a small value \( \epsilon \) → 0, let \( \angle p_2 = \pi - \angle p_1 p_2 p_3 \) and \( \angle p_3 = \pi - \angle p_2 p_3 p_4 \),

- If \( \angle p_2 > \epsilon \), \( \angle p_3 < \epsilon \), and there is no other point between \( p_1 \) and \( p_4 \), then \( p_2 \) is a key-point (see Fig.3(a));
- If \( \angle p_2 < \epsilon \), and \( \angle p_3 > \epsilon \), then \( p_3 \) is a key-point;
- If \( \angle p_2 > \epsilon \), \( \angle p_3 > \epsilon \), and \( \angle p_2 + \angle p_3 < \pi \), then there may be a missing key-point. In this case, it is also possible that both of \( p_2 \) and \( p_3 \) are key-points. If we can find a missing point \( p = v \) at time \( t \), that \( \angle p = \angle p_2 + \angle p_3 \geq 2+ \epsilon \), then the point \( p \) is more likely to be a key-point between \( p_2 \) and \( p_3 \), as the larger \( \angle p \) indicates the larger turning degree of the time series at point \( p \). We deduce missing key-points by solving the equation

\[
Q = \frac{|p_2p_3|}{\sin(\angle p_2)} = \frac{|p_3p_4|}{\sin(\angle p_3)}, \quad \text{where} \quad Q = \frac{|p_2p_3|}{\sin(\pi - \angle p_2 - \angle p_3)},
\]

which can be written as:

\[
\begin{align*}
Q^2 \sin \angle p_3 &= (v - v_2)^2 + (t - t_2)^2 \\
Q^2 \sin \angle p_2 &= (v - v_3)^2 + (t - t_3)^2
\end{align*}
\]

If Equation (1) only has one solution, this solution is a key-point; if it has two solutions, we adopt the one on the line of \( p_2p_3 \), i.e., \( p \) in Fig.3(b) as a key-point; if it does not have solution, point \( p_2 \) and \( p_3 \) are key-points.

- If \( \angle p_2 > \epsilon \), \( \angle p_3 > \epsilon \), and \( \angle p_2 + \angle p_3 > \pi \), then \( p_2 \) and \( p_3 \) are both key-points (Fig.3(c)). In addition, it is impossible that there are other missing points, say \( p \), between \( p_2 \) and \( p_3 \), that \( \angle p > \epsilon \).

- If more than one consecutive key-points are missing, the above method will only detect one missing point as an representation of all the missing key-points. For example, Fig.3(c) shows \( p_{k12} \) and \( p_{k22} \) are two missing key-points, however, one virtual key-point \( p_2 \) based on the existing points \( p_1, p_{k11}, p_{k21} \), and \( p_3 \) are deduced.

Key-points capture the critical information and fill the missing information of a \( PTS \), hence, the detected key-points can be used to represent the raw \( PTS \). In the sequel sections, a \( PTS \) typically refers to a series of key-points of the original \( PTS \).

4.2. Anomaly Detection in Corrected Time Series

Anomaly detection and normal pattern identification are both processed based on the unit of period. The first step is to identify period points \( Q \) that separate \( PTS \) into a set of periods. We use clustering method to categorize the inflexions of a \( PTS \) into a number of 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 have highest mean silhouette value will be assumed to have the best clustering pattern. To accurately conduct clustering, we introduce a feature vector for each inflexion of $PTS$, with the optimal intention that each point can be distinguished with others efficiently.

4.2.1. Time Series Compression: TSCC. 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: (1) use line segment $p_1p_n$ to simplify the $PTS$; (2) find the farthest point $p_f$ from $p_1p_n$; (3) if distance $d(p_f, p_1p_n) \leq \lambda$, where $\lambda$ is a small value, and $\lambda \geq 0$, then the $PTS$ can be simplified by $p_1p_n$, and this procedure is stopped; (4) otherwise, recursively simplify the subsequences $\{p_1, \cdots, p_f\}$ and $\{p_f, \cdots, p_n\}$ using steps (1−3).

Definition 4.2. Given a $PTS = (v_1, \cdots, v_n)$, a compressed time series $CTS$ of $PTS$ is represented as $CTS = (v_{c_1}, \cdots, v_{c_n}) \subseteq PTS$, where $\forall p_{c_i} \in CTS$ is an inflexion, and $|CTS| \ll |PTS|$.

The feature vector of an inflexion is defined as:

Definition 4.3. A feature vector for a point $p_i \in PTS$ is a four-value vector $vec_i = (vdiff_1i, vdiff_2i, tdiff_1i, tdiff_2i)$, where $vdiff_1i = v_i - v_{i-1}$, $vdiff_2i = v_{i+1} - v_i$, $tdiff_1i = t_i - t_{i-1}$, and $tdiff_2i = t_{i+1} - t_i$.

Example 4.4. Fig.4 shows an example of a $PTS$ and one of its compressed time series $CTS$. The value differences $vdiff_1$ and $vdiff_2$, and the time differences $tdiff_1$ and $tdiff_2$ are shown in Fig.4.

4.2.2. Period Segmentation and Summarization: PSSC. PSSC component 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 below.

Step 1: Cluster Points of CTS Points are clustered into a number of clusters based on their feature vectors. 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 feature vector, 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 an brief introduction of the $k$-means++ in this section.
**k-means++** is an improvement of **k-means** by first determining the initial clustering centres 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-mean++** 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 centre 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(log_k)** 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:

**Definition 4.5.** 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

\[
\text{sil}(p_i) = \frac{b(p_i) - a(p_i)}{\max\{a(p_i), b(p_i)\}}
\]

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

In the above definition, \( \text{sim}(p_i, p_j) \) can be calculated by any similarity calculation formula. In this work, we adopt the Euclidean Distance as similarity measure, i.e.,

\[
\text{sim}(p_i, p_j) = \sqrt{(v_i - v_j)^2 + (t_i - t_j)^2},
\]

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, while \( b(p_i) \) 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 \( -1 \leq \text{sil}(p_i) \leq 1 \), a \( \text{sil}(p_i) \rightarrow 1 \) means that a point \( p_i \) is well clustered, while \( \text{sil}(p_i) \rightarrow 0 \) represents the point is close to the boundary between clusters \( M \) and \( H \), and \( \text{sil}(p_i) < 0 \) indicates that point \( p_i \) is close to the points in the neighbouring 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:

\[
\text{msil}(C_{CTS}) = \frac{1}{|CTS|} \sum_{p_i \in CTS} \text{sil}(p_i).
\]

Similar to the silhouette value of a point, the \( \text{msil} \rightarrow 1 \) 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 \( \text{msil}(C_m) = \sum_{p_i \in C_m} \text{sil}(p_i) \), where \( -1 \leq \text{msil}(C_m) \leq 1 \), and \( \text{msil}(C_m) \rightarrow 1 \) means the high quality of the cluster \( m \). Based on the definition of silhouette values, we give Algorithm [1] of 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 pre-defined threshold value \( \eta \), then the clustering result is unqualified. Feature vectors of points need to be re-clustered with adjusted parameters, e.g.,
Algorithm 1: Cluster quality validation

Input: (1) \( V = \{ v_{ci} | 1 \leq i \leq |CTS| \} \), where \( v_{ci} = (\alpha^i, \text{diff}_1^i, \text{diff}_2^i) \)
(2) A set of point clusters: \( C_{CTS} = \{ C_m | 1 \leq m \leq k \} \)
(3) Threshold values \( \eta \) and \( \xi \), \( 0 \leq \eta, \xi \leq 1 \)

Output: Period cluster \( C_{\text{perid}} \)
Calculate \( \text{sil}(p_i) \) for \( \forall p_i \in CTS \);
Calculate mean silhouette value: \( \text{msil}(C_{CTS}) \);
if \( \text{msil}(C_{CTS}) < \eta \) then
\( C_{\text{perid}} = \text{NULL}; \)
return;
\( C_{\text{perid}} = \max(\text{msil}(C_m)) \) \& \( \text{msil}(C_m) > \xi \) \( \forall C_m \in C_{CTS} \).

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 \( \xi \).

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 to identify 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 4.6. Given a \( CTS \) that has been separated into \( D \) periods, a summary of a period \( \text{seg}_i = (v_{i1}, \ldots, v_{in}) \), \( 1 \leq i \leq D \) is a vector \( \text{seg}_i = (h_i^\text{min}, t_i^\text{min}, h_i^\text{max}, t_i^\text{max}, h_i^\text{mea}, p_i^\text{minmax}, p_i^\text{max}) \), where \( h_i^\text{min} \) is the amplitude value of the point having minimum amplitude in period \( i; h_i^\text{min} = \min\{v_{ik} | 1 \leq k \leq m\}; t_i^\text{min} \) is the time index of the point with minimum amplitude. If there are two points having the minimum amplitude, \( t_i^\text{min} \) is the time index of the first point, \( h_i^\text{max} = \max\{v_{ik} | 1 \leq k \leq m\}; t_i^\text{max} \) is the first point with maximum amplitude; \( h_i^\text{mea} = \frac{1}{m} \sum v_{ik}; p_i^\text{minmax} = |t_i^\text{max} - t_i^\text{min}|; p_i^\text{max} = t_{ik} - t_{i1} \).

We represent the segmented \( CTS \) as \( STS = \{ \text{seg}_1, \ldots, \text{seg}_n \} \). Each period corresponds to an annotation \( \text{ann} \) indicating the state of the period. In this paper, we will only consider two states: normal and abnormal. Therefore, a \( STS \) is always associated with a series of annotations \( A_{STS} = \{ \text{ann}_1, \ldots, \text{ann}_n \} \).

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

Rule 1. \( \text{ann} = \otimes ('Ab', 'N') = 'Ab' \text{ and } \text{ann} = \otimes ('N', 'N') = 'N' \).

Assume a period covers a subsequence that is labelled by two signals, if there exists an abnormal behaviour in the subsequence, then based on rule 1, the behaviour of the segmentation of the period is abnormal; otherwise the period is a normal series. This label assignment rule can be extended to multiple labels: given a set of labels \( Lbs = \{ l_{b(1)}, \ldots, l_{b_r} \} \), if \( \exists l_{b_j} = 'Ab', 1 \leq j \leq r \), the value of \( Lbs \) is 'Ab', represented as \( lbs = \otimes (l_{b(1)}, \ldots, l_{b_r}) = 'Ab'; \text{if } \forall l_{b_j} = 'N', lbs = 'N' \).
Algorithm 2: Period annotation

Input: Period \( pd_i = (v_1, \cdots, v_m), 1 \leq i \leq n \); A series of labels \( Lbs = (lb_1, \cdots, lb_r) \);

Output: An annotated \( pd_i' \);

\[ t^1_i = NULL; \] the time of the 1st annotation in the period;
\[ t'^{end}_i = NULL; \] the time of the last annotation in the period;

if \( \exists lb_j \) that \( t(i-1)1 \leq t_{j-1} \leq t(i-1)m < t_i \leq t_{im} \) then
\[ t^1_i = t_j; \]
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
\[ t'^{end}_i = t_k; \]
end

if \( t^1_i \neq NULL \& t'^{end}_i \neq NULL \) then
  if \( t^1_i = NULL \) then
  \[ t^1_i = 'N' \]
  end
  if \( t'^{end}_i = NULL \) then
  \[ t'^{end}_i = 'N' \]
  end
  \( Lbs = Lbs\{t^1_i, \cdots, t'^{end}_i\}; \)
  \( lbs = \otimes(Lbs); \)
else
  \( lbs = Lbs\{t^1_{i+1}\}; \)
end

Fig. 5. Segmentation and annotation of two periods

According to the above discussion, the annotation of a period \( pd_i \) is determined by Algorithm 2.

Example 4.7. We present the segmentation and annotation of a period in Fig 5 to explain their processes more clearly. Fig 5 shows that \( pd_i \) does not involve any label and the first label in \( pd_{i+1} \) is \( lb_1 = N \), so \( lb_{pd_i} = 'N' \). \( lb_2 \) is 'Ab', hence \( pd_{i+1} \) is annotated as 'Ab'.

4.2.3. Classification-based Anomaly Detection and Prediction: ADPC. From Definition 4.6, each period of a PTS is summarised by seven features of the period: \((h^m_{min}, t^m_{min}, h^m_{max}, t^m_{max}, h^m_{mea}, p^m_{minmax}, p^m_{l})\). Using these seven features to abstract a period can significantly reduce the computational complexity in a classification process.
Table II. ECG Datasets used in experiments

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Abbr.</th>
<th>#ofSamples</th>
<th>AnomalyTypes</th>
<th>#ofAbnor</th>
<th>#ofNor</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHA0001</td>
<td>ahadb</td>
<td>899750</td>
<td>V</td>
<td>115</td>
<td>2162</td>
</tr>
<tr>
<td>SupraventricularArrhythmia800</td>
<td>svdb</td>
<td>230400</td>
<td>S &amp; V</td>
<td>75</td>
<td>1846</td>
</tr>
<tr>
<td>SuddenCardiacDeathHolter30</td>
<td>sddb</td>
<td>22099250</td>
<td>V</td>
<td>38</td>
<td>5743</td>
</tr>
<tr>
<td>MIT-BIH Arrhythmia100</td>
<td>mitdb</td>
<td>650000</td>
<td>A &amp; V</td>
<td>164</td>
<td>2526</td>
</tr>
<tr>
<td>MIT-BIH Arrhythmia106</td>
<td>mitdb06</td>
<td>650000</td>
<td>A &amp; V</td>
<td>34</td>
<td>2239</td>
</tr>
<tr>
<td>MGH/MF Waveform001</td>
<td>mgh</td>
<td>403560</td>
<td>S &amp; V</td>
<td>23</td>
<td>776</td>
</tr>
<tr>
<td>MIT-BIH LongTerm14046</td>
<td>ltdb</td>
<td>10828800</td>
<td>V</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>AF TerminationN04</td>
<td>aftdb</td>
<td>7680</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

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 compress the raw ECG time series by utilizing the DP algorithm, and to represent each inflexion in the perceived CTS as a feature vector (see Definition 4.4). Secondly, the $K$-means++ clustering algorithm is applied to the series of feature vectors 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. Thirdly, each segment is summarised by the seven features (see Definition 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. 2000a], which are summarised in Table II where 'V' represents Premature ventricular contraction, 'A': Atrial premature ventricular, and 'S': Supraventricular premature beat. Apart from the aftdb dataset, each time series is separated into a series of subsequences that are labelled 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.

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. 2000b] 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:

1. Accuracy (acc): $(TP + TN) / \text{Number of all classified samples}$;
2. Sensitivity (sen): $TP / (TP + FN)$;
3. Specificity (spe): $TN / (FP + TN)$;
4. Prevalence (pre): $TP / \text{Number of all samples}$.
5. Fmeasure (fmea): $2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$, where recall = sen, precision = $\frac{TP}{TP + FP}$.

Details of the experiments are illustrated in the following sections.

5.1. Inflexion Detection and Time Series Compression

At first, we design an experiment to detect the inflexions in a time series. The detected inflexions will be used as an approximation of the raw time series, and will be compressed by DP algorithm. We design this experiment based on the work of [Rosin 2003]. We assess the stability of the uncertainty detection and DP compression algorithms under the variations of the change of scale parameters and the perturbation of data. The former is measured by using a monotonicity index and the latter is quantified by a break-point stability index.
Table III. Decreasing monotonicity degree of six datasets in terms of the value of $\epsilon$ and $\lambda$

<table>
<thead>
<tr>
<th></th>
<th>ahadb</th>
<th>svdb</th>
<th>sddb</th>
<th>mitdb</th>
<th>mgh</th>
<th>aftdb</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon$</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Fig. 6. Monotonically decreasing number of breakpoints in terms of $\epsilon$ for the inflexion detection procedure and $\lambda$ for the DP algorithm

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 inflexion detection algorithm and the DP algorithm, if the values of the scale parameters $\epsilon$ and $\lambda$ are 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 of $M_D$ 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 $\epsilon$ for inflexion detection procedure and $\lambda$ for DP algorithm. For the inflexion detection procedure, we set $\epsilon = 1, 2, 3, 4, 5$. Table III shows that the breakpoint numbers for the six datasets are perfectly decreasing in terms of the increasing $\epsilon$, which can also be seen in Fig.6(a). For DP algorithm, we first fix $\epsilon = 1$, and detect inflexions of the six time series. Based on the detected inflexions, we set $\lambda = 5, 10, 15, 20, 25, 30, 35, 40, 45, 50$ to conduct DP compression. From Table III and Fig.6(b), we can see that the numbers of breakpoints are also 100% decreasing in terms of the increasing $\lambda$.

The break-point 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: $\epsilon = 1$ for the inflexion detection and $\lambda = 10$ for the DP algorithm. The endpoint stability measurement is defined as $S = (1 - \frac{1}{m} \sum_d \sum_b \frac{s_d^d}{n_d^d})$, where $m$ is the level number of deletion, $d$ is the $d$th level, $s_d^d$ 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 $d$th deletion. Table IV shows the deletion length of each running circle and the stability degree of each time series. For example, after inflexion detection, the sample number of ahadb is 307350. We iteratively delete 10000 samples from the beginning of the remaining ahadb time series, and
Table IV. Endpoint stability of six datasets and pertubations

<table>
<thead>
<tr>
<th></th>
<th>ahadb</th>
<th>svdb</th>
<th>sddb</th>
<th>mtdb</th>
<th>mgh</th>
<th>aftdb</th>
</tr>
</thead>
</table>

Conduct the DP algorithm based on the new time series. The positions of the identified breakpoints in each running circle are compared with 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

From the above testing (see Fig 6), we can see that when $\epsilon \geq 4$, the number of detected inflexions of each time series is going to be 0. Based on Fig 6, we set $\epsilon = 1$ and $\lambda = 10$ for inflexion detection and time series compression. We then compare three methods of period point representation: (1) inflexions in CTS are represented by feature vectors (FV); (2) inflexions are represented by angles (Angle) of peak points [Huang et al. 2014]; (3) inflexions are represented by valley points (Valley) [Tan 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 and $\alpha$ is an outlier control factor that is determined and adjusted by experts. As stated by Tan et al. [Tan et al. 2007], the best values of initial upper bound and $\alpha$ in ECG are 50mmHg and 1.1. The perceived feature vector 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 Definition 4.6). Finally, Linear Discriminant Analysis (LDA) and Naive Bayes (NB) classifiers are applied for sample classification and anomaly detection. Fig 7 shows the identified period points using the FV-based method for four datasets: ltdb, sddb, svdb and ahadb. From Fig 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 column ‘mean’ refers to the mean silhouette value of a dataset clustering, and the values in columns c(luster)1-6 are the mean silhouette values of each cluster after clustering a dataset. ‘NAS’ in the sixth column means that the inflexions in the corresponding datasets are clustered into five groups, which 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 or higher than 0.9 ($\xi = 0.8$ in Algorithm 1). In addition, for each dataset, we select the points in the cluster with highest silhouette value as the period points. For example, for dataset ahadb, points in cluster 4 are selected as period points.

Fig 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 clus-
Fig. 7. Period point identification of four datasets based on feature vectors

Table V. Silhouette values of six datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Silhouette values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
</tr>
<tr>
<td>ahdab</td>
<td>0.8253</td>
</tr>
<tr>
<td>svdb</td>
<td>0.6941</td>
</tr>
<tr>
<td>sddb</td>
<td>0.772</td>
</tr>
<tr>
<td>mitdb</td>
<td>0.9373</td>
</tr>
<tr>
<td>mitdb06</td>
<td>0.7339</td>
</tr>
<tr>
<td>ltdb</td>
<td>0.9149</td>
</tr>
<tr>
<td>mgh</td>
<td>0.8253</td>
</tr>
</tbody>
</table>

ters, and he values in each cluster are more similar to each other compared with 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 Fig 8(a)) is 0.9373, while the angle-based clustering (Fig 8(b)) is 0.7461; and the mean values for ltdb are 0.9149 and 0.8155 (Fig 8(c) and Fig 8(d)) respectively.

Fig 8 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 Fig 8, 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 while 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. This experiment is conducted on seven datasets: ahadb, sddb, sddb, mitdb, mitdb06, mgh, and ltdb. From the previous subsections, 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 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 Fig. 10 which compares the performance of classification methods LDA, NB and RF, based on datasets (a) ahadb, (b) sddb, (c) mitdb, (d) mgh, (e) sddb, 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 over 90% for the first five datasets (a-e). However, we can also see that the feature values of LDA are always higher than the feature values of the other two methods.
Fig. 10. Classification performance of six datasets based on the summarized features using classification methods of LDA, RF, and NB

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 ((a) ahadb, (b) ltdb, (c) mitdb, (d) sddb, and (e) svdb)

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 DP algorithm and periodically separated by feature vector 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 Section 4.3). 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 five benchmarks: acc, sen, fmea, and prev.

Fig.11 shows the evaluated results of the classifier performance based on the proposed period identification and segmentation method. From Fig.11 we can see that the accuracy values of classification based on the 5 datasets are over 90%, except the cases of LPB with mitdb, LDA with sddb, LDA with svdb, and RUS with svdb. Some of them are of more than 98% accuracy. The sensitivity of classification based on the datasets of abadb, ltdb, and mitdb are closing to 100%. The sensitivity based on the datasets of sddb and svdb are over 85%. The f-measure rates of classification based on abadb, 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 over 90%.

6. CONCLUSIONS
In this paper, we have introduced a framework of detecting anomalies in uncertain pseudo periodic time series. We formally define pseudo periodic 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 pre-processed 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. Though 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. Correcting false-detected inflexions and detecting global anomalies in an uncertain PTS will be the main target of our next research work.

Acknowledgement
This work is supported by the National Natural Science Foundation of China (NSFC 61332013) and the Australian Research Council (ARC) Discovery Projects DP140100841, DP130101327, and Linkage Project LP100200682.

REFERENCES
Charu C. Aggarwal. 2009. On high dimensional projected clustering of uncertain data streams. In IEEE 25th International Conference on Data Engineering (ICDE'09). IEEE, Shanghai, China, 1152–T154. 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 IEEE 24th International Conference on Data Engineering (ICDE'08). IEEE, Cancun, Mexico, 150–159. DOI: http://dx.doi.org/10.1109/ICDE.2008.4497423


Jing He, Yanchun Zhang, Guangyan Huang, and Paulo de Souza. 2013. CIRCE: correcting imprecise readings and compressing excrescent points for querying common patterns in uncertain sensor streams. Inform. Syst. 38, 8 (2013), 1234–1251. doi: http://dx.doi.org/10.1016/j.ins.2012.10.003


Thanh T. Tran, Liping Peng, Yanlei Diao, Andrew Ng, and Anna Liu. 2012. CLARO: modelling and processing uncertain data streams. VLDB J. 21, 5 (October 2012), 651–676. doi:10.1007/s00778-011-0261-7

