Sensors **2015**, *15*, 6066-6090; doi:10.3390/s150306066

OPEN ACCESS

sensors

ISSN 1424-8220

www.mdpi.com/journal/sensors

Article

Data Fault Detection in Medical Sensor Networks

Yang Yang *, Qian Liu, Zhipeng Gao, Xuesong Qiu and Luoming Meng

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, No.10 Xitucheng Road, Haidian District, Beijing 100876, China; E-Mails: lq_1990@bupt.edu.cn (Q.L.); gaozhipeng@bupt.edu.cn (Z.G.); xsqiu@bupt.edu.cn (X.Q.); lmmeng@bupt.edu.cn (L.M.)

* Author to whom correspondence should be addressed; E-Mail: yyang@bupt.edu.cn; Tel./Fax: +86-10-6119-8090.

Academic Editor: Leonhard M. Reindl

Received: 20 November 2014 / Accepted: 27 February 2015 / Published: 12 March 2015

Abstract: Medical body sensors can be implanted or attached to the human body to monitor the physiological parameters of patients all the time. Inaccurate data due to sensor faults or incorrect placement on the body will seriously influence clinicians' diagnosis, therefore detecting sensor data faults has been widely researched in recent years. Most of the typical approaches to sensor fault detection in the medical area ignore the fact that the physiological indexes of patients aren't changing synchronously at the same time, and fault values mixed with abnormal physiological data due to illness make it difficult to determine true faults. Based on these facts, we propose a Data Fault Detection mechanism in Medical sensor networks (DFD-M). Its mechanism includes: (1) use of a dynamic-local outlier factor (D-LOF) algorithm to identify outlying sensed data vectors; (2) use of a linear regression model based on trapezoidal fuzzy numbers to predict which readings in the outlying data vector are suspected to be faulty; (3) the proposal of a novel judgment criterion of fault state according to the prediction values. The simulation results demonstrate the efficiency and superiority of DFD-M.

Keywords: fault detection; medical sensor; local outlier factor; fuzzy number

1. Introduction

Wireless sensor networks (WSNs) have been widely used in medical applications. Body sensors are designed to be implanted in or adhere to the human body to monitor physiological parameters for a long term. For example, clinicians can monitor heart rate, blood glucose, or the body temperature of patients at any time [1,2]. Clinicians then diagnose the state of illness according to accurate sensed readings.

How to transmit and verify measurement data in a timely way has received more attention from researchers. The changing status of illness requires physiological readings to be reported frequently. What's more, data inaccuracy due to limited resources (e.g., fluorescent lighting may cause data errors in pulse oximeters), or incorrect placement on the patient's body will seriously influence the diagnosis of clinicians. e.g., piezoelectric sensors for snoring monitor the snore waveform of patients under sleeping, anaesthesized or sober conditions. The belt or medical adhesive tape may fall off and result in detection failure because patients always turn over.

Recently, some references have researched medical sensor detection problems [3–5]. They classify the states of patients as normal and abnormal, and then analyze readings' abnormity according to intervals of physiological parameters to detect and isolate erroneous nodes, but these approaches have some drawbacks: (1) they detect data abnormities caused by sensor faults only when peoples' states are stable, but it's difficult to determine and judge when fault readings are mixed with abnormal physiological data; (2) in fact, physiological indicators of patients may be not be synchronously changing at the same time. For example, the heart rate can increase rapidly, while temperatures may slowly increase after a few minutes. Out-of-sync changes of medical attributes will cause a decrease in detection accuracy. Faulty measurements from sensors negatively influence the measured results and lead to diagnosis errors. Furthermore, they may threaten the life of a patient after alerting emergency personnel. Therefore, an important task is to detect abnormal and faulty measurements that deviate from real observations, and to distinguish sensor faults from real emergency situations in order to reduce false alarms.

In the paper, we mainly focus on detecting faulty medical sensors which generate faulty measurements mixed with abnormal physiological data, and accordingly increase the detection accuracy rate. We propose a Data Fault Detection in Medical body monitoring networks (DFD-M) method. Our innovations are:

(1) We firstly use a dynamic-LOF algorithm to identify outlying data vector. We find the new added objects which only influence parts of objects' local outlier factors (LOF) in the original dataset. Finding these related objects whose LOF values have changed, the dynamic-LOF algorithm narrows down the scope of the detected objects' nearest neighborhood to be more sensitive to outliers. What's more, through finding three-levels of influenced objects, it can reduce the time complexity when the size of dataset is increasing dynamically.

(2) After locating an outlying sensed data vector, we use a fuzzy linear regression model based on trapezoidal fuzzy numbers to predict suspected readings in the data vector. In order to better fuzzify the prediction values, we construct an objective function of the sum of errors related to a fuzzy number expectation. These approaches help to embody a more precise relationship between prediction values and readings suspected to be faulty.

(3) Propose a novel judgment criterion for fault state according to the fuzzy prediction value. We summarize 15 relative position relationships between the fuzzy prediction results and the normal intervals of the corresponding physiological parameters, and accordingly conclude whether a sensor is at fault.

The rest of the paper is organized as follows: Section 2 describes some related works about fault detection in WSNs. Section 3 introduces the proposed DFD-M mechanism and involved algorithms, including dynamic-LOF, fuzzy linear regression process, and determination criterions. The simulation results in Section 4 demonstrate our algorithm's efficiency and superiority. In Section 5, we conclude the paper.

2. Related Works

Typical faults in WSNs are link faults and data faults. For faulty link detection, transmission and reception will be influenced by destructive nodes. Link faults will cause network bottlenecks and partitioning. Most approaches use probes to fetch feedback information about faulty paths. Researchers analyze probe feedback according to network symptoms, and infer network topology and link states. A link scanner (LS) [6] collects hop counts of probe messages. It provides a blacklist containing all possible faulty paths. In probabilistic reasoning, each link generates two probe records. A link is deemed to fail to send probes when the collection record is mismatched with the expectation.

In istributed data fault detection, a sensor can determine its fault state through its neighbors' monitoring values [7]. Sensors only know one-hop neighbors' states instead of global information. Sensors should send their readings and determined tendency states periodically. When more than half of one-hop good neighbors are deemed it to be possibly good, then it maybe fault-free. The detection accuracy lies in the fault probability of sensors and network topology. Ding *et al.* [8] proposed a faulty identification method with lower computational overhead compared with [7]. If the difference between readings is large or large but negative, then the sensor is deemed as faulty.

Krishnamachari and co-workers proposed in [9] a distributed solution for the canonical task of binary detection of interesting environmental events. They explicitly take into account the possibility of measurement faults and develop a distributed Bayesian scheme for detecting and correcting faults. Each sensor node identifies its own status based on local comparisons of sensed data with some thresholds and dissemination of the test results [10]. Time redundancy is used for tolerating transient sensing and communication faults.

In the centralized mode, MANNA architecture [11] adopts a manager to control and manage a global vision of the wireless sensor network. Every node will check its energy level and send a message to the manager/agent whenever there is a state change. The manager obtains the coverage map and energy level of all sensors based upon the collected information. To detect node failures, the manager sends GET operations to retrieve the node state. Without hearing from the nodes, the manager will consult the energy map to check its residual energy. However, this approach requires an external manager to perform the centralized diagnosis and the communication between nodes and the manager is too expensive for WSNs.

In medical sensor networks, readings are typically accumulated and transmitted to a central device [4]. The device stores and processes data and judges illnesses. Focusing on diagnosis errors

influenced by faulty measurements, [12] represents sensor fault and patient anomaly detection and classification. The approach classifies sensed readings into normal and abnormal, and then uses regression prediction to distinguish faulty readings from physiological abnormal readings. It can reduce the frequency of false alarm triggered by inconsistent monitoring.

Reliable and light weight are also objectives in the detection of faults caused by sensors. The paper [13] proposes an online detection algorithm for isolating incorrect readings. To reduce false alarms, it firstly uses a discrete Haar wavelet decomposition and Hampel filter for detecting spatial deviations. Then a boxplot is adopted for temporal analysis. Miao *et al.* [14] presented an online lightweight failure detection scheme named Agnostic Diagnosis (AD), which is motivated by the fact that the system metrics of sensors usually exhibit certain correlation patterns. In [3**]**, the authors present an online solution for fault detection and isolation of erroneous nodes in body sensor networks to provide sustained delivery of services despite encountered perturbations. It assumes that the sensors are either permanently faulty or fault free. Breunig *et al.* proposed an outlier detection method based on density and assign to each object a local outlier factor (LOF), which is the degree to which the object is outlying [15]. In [16], authors propose a fault detection and isolation algorithm in pervasive motion monitoring with two motion sensors, but in real life, sensors are not always perfectly synchronized. The poor constructions of sensors or monitoring program can generate missing data or unsynchronized data.

The dynamic changing illness of patients and out-of-sync variation of medical attributes influence the performance of data fault detection for medical sensors, so we further promote detection accuracy, reduce time complexity, and resolve the practical problem of out-of sync changing of sensed attributes.

3. Data Fault Detection of Body Sensors

We firstly assume physiological parameters have correlations between them. For example, heart rate (HR) is measured as the number of intervals in an electrocardiogram (ECG) signal. The respiration rate (RR) is proportional to the heart rate, but varies for different individuals. An additional one degree centigrade body temperature adds around 10–15 beats per minute to the heart rate [2]. In addition, episodes of low blood sugar (hypoglycemia) can trigger an arrhythmia, which often behaves like a racing heart [5]. In conclusion, respiration rate and body temperature are all positively correlated with heart rate.

Physiological parameters are correlated in time and space, and the correlation must be exploited to identify and isolate faulty measurements, in order to ensure reliable operation and accuracy diagnosis results. Usually, there is no spatial and temporal correlation among monitored attributes for faulty measurements. Based on the above theory, we introduce the data fault detection of body sensors, and focus on detecting faulty medical sensors so as to determine and judge the fault readings.

We firstly give some definitions. Physiological readings are described as a matrix $X = (X_{ji})$, in which j represents measuring time, and i is sensor i. The sequence $X_i = (X_{1i}, X_{2i}, X_{3i}, \ldots, X_{ti})$ represents measured values of sensor i from time T_1 to T_t. The vector $X_j = (X_{j1}, X_{j2}, \ldots, X_{jn})$ represents all of physiological parameters on time T_j.

The DFD-M mechanism is described as follows:To For reduce computational complexity and enhance detection average accuracy, a dynamic-LOF algorithm is applied in the increasing datasets.

Based on this, it narrows down the scope of the detected objects' nearest neighborhood to be more sensitive to outliers. Then, it finds out three levels of influenced objects and narrows the range of updating LOF values so as to improve the outlier detection efficiency (see Section 3.1).

Step 2: For an outlying vector, it needs to determine how many values are abnormal. Firstly, we define physiological normal and abnormal intervals of patients. For example, physiological normal HR is in the range (50, 130). Next, if all of readings are all in their normal or abnormal intervals and the corresponding vector is determined to be an outlier, no sensor is faulty, so this condition must be caused by a patient's changing illness state. Otherwise, classify all of readings in the outlying vector into physiological normal and abnormal sets. The set which has more elements (supposed to be set *A*) will be regarded as reliable inputs of a fuzzy linear regression model. The outputs are the prediction values corresponding to elements in another set (supposed to be set *B*). As physiological indicators of patients may not synchronously changing at the same time, the prediction results only deduce that some sensors in set *B* are suspected to be faulty (see Section 3.2).

Step 3: For any reading y to be suspected, it must be a real number. While in the fuzzy regression process, the output is a fuzzy prediction value $\widetilde{Y_p} = (y_a, y_b, y_l, y_r)$. Its lower bound is $(y_a - y_l)$, and upper bound is $(y_b + y_r)$. We need to analyze the difference between $\widetilde{Y_p}$ and y according to the corresponding normal intervals. If y is closer to $\widetilde{Y_p}$ according to the determination criterion (see Section 3.3), it means the sensor corresponding with the reading y is good, otherwise, it is faulty.

3.1. Dynamic-LOF Algorithm

An exact definition of an outlier depends on the data structure and detection methods. A data object is either an outlier or not. Breunig *et al.* proposed an outlier detection method based on density and assign to each object a local outlier factor (LOF), which is the degree to which the object is outlying [15]. It is local in that the degree depends on how isolated the object is with respect to the surrounding neighborhood, so the key idea of LOF is to compare the local density of an object's neighborhood with that of its neighbors.

Our detection targets are dynamic time-series readings which are dynamic and constantly updated. Since the density-based LOF algorithm cannot detect contextual anomalies, the LOF values of all objects will be recalculated once the dataset changes. When the size of dataset increases, it will take a lot of time to frequently update the LOF values of all the objects in dataset, so the density-based LOF algorithm may not be suitable for the dynamic increment dataset. We find that newly added objects can only influence parts of objects' LOF values in the original dataset. We only need to find out these related objects whose LOF values have changed, and recalculate their LOF values. Thus the efforts in recalculating LOF values of all the objects can be reduced, so in this paper, we propose a dynamic-LOF algorithm to identify outlying data vectors. We find out the three levels of influenced objects and recalculate these nodes' new LOF values. Besides, a small modification is made to narrow down the scope of the detected objects' nearest neighborhood, which can increase the detection accuracy, and then more outliers are detected.

The core idea of dynamic-LOF algorithm is, on the one hand, a small modification in k-distance, which makes our algorithm achieve higher detection average accuracy. On the other hand, finding out three levels of influenced objects and narrowing the range of updating LOF values to improve the

outlier detection efficiency. The vector $X_j = (X_{j1}, X_{j2}, \ldots, X_{jn})$ are all of sensed physiological readings at time T_j is regarded as an object in multidimensional space. The size of dataset D continually increases over time. Firstly we obtain LOF values of all objects in dataset D, and then establish an initial knowledge base. All the objects in the initial knowledge base are considered normal. The dynamic-LOF algorithm only needs to update the LOF values of the newly added object and other objects which are influenced by the new one.

Assume o, o', p, q, s, x, y, z to denote objects in a dataset D. Each object in dataset is assigned a local outlier factor. The larger the LOF is, the greater the chance is of an object being an outlier. We use the notation $d(s, o)$ to denote the distance between objects s and o. s is an object in D. We take mean distance of object s, denoted as *mk-distance* to replace *k-distance* in the original LOF algorithm. It indicates the mean distance from s to its k-nearest objects.

Definition 1. k-distance and k-distance neighborhood of an object s.

For any positive integer k, the *k-distance* of object s, denoted as *k-distance*(s) is defined as the distance $d(s, o)$ between s and an object $o \in D$ such that:

(1) For at least k objects $o' \in D\backslash\{s\}$ it holds that $d(s, o') \leq d(s, o)$, and

(2) For at most $k - 1$ objects $o' \in D\backslash\{s\}$ it holds that $d(s, o') < d(s, o)$.

Then the k-distance neighborhood of s is $N_k(s) = \{q \in D\backslash\{s\} | d(s, q) \leq k\text{-}distance(s)\}$. These objects q are called the k-nearest neighbors of s.

Definition 2. mk-distance of an object s.

For any positive integer k, the *mk-distance* of object s is:

$$mk - distance(s) = \frac{\sum_{o \in N_k(s)} d(s, o)}{|N_k(s)|} \tag{1}$$

Definition 3. mk-distance neighborhood of an object s.

Given *mk*-distance of s, the *mk*-distance neighborhood of s contains every object whose distance from s is not greater than *mk-distance*, *i.e.*, $N_{mk}(s) = \{q \in D\backslash\{s\} | d(s, q) \leq mk\text{-}distance(s)\}$. Each object q in $N_{mk}(s)$ is also in $N_k(s)$.

Definition 4. Reachability distance of an object s with respect to object o is:

$$r - dist_{mk}(s, o) = max\{mk - distance(o), d(s, o)\} \tag{2}$$

Definition 5. Local reachability density of an object s is:

$$lrd_{mk}(s) = 1 \Big/ \left(\frac{\sum_{o \in N_{mk}(s)} r - dist_{mk}(s, o)}{|N_{mk}(s)|}\right) \tag{3}$$

Definition 6. Local outlier factor of an object s is:

$$LOF_{mk}(s) = \sum_{o \in N_{mk}(s)} \frac{lrd_{mk}(o)}{lrd_{mk}(s)} \Big/ |N_{mk}(s)| \tag{4}$$

The outlier factor of object s captures the degree to which we call s an outlier. It is the average of the ratio of the local reachability density of s and those of its *mk*-distance neighbors. It is easy to see that the lower local reachability density is, and the higher the local reachability densities of its *mk*-distance neighbors are, the higher is the LOF value of s.

By using the above formulas to calculate the LOF_{mk} values of all objects in the dataset, the scope of nearest neighborhood of each object can be narrowed down, so our improved algorithm is more sensitive to outliers, and can achieve higher detection average accuracy.

According to the steps above, it's clear that the LOF value of each object depends on the local reachability density and its *k*-nearest neighbors. When there are any newly added, deleted, or updated objects, the LOF values of partially related objects would be influenced. In the environment of dynamic increment dataset, updating the LOF values of all the objects in dataset frequently will cost a great deal of temporal and spatial resources, but we note that only part of the related objects will be influenced by the changes of dataset, so we only need to find out these related objects whose LOF values have changed, and recalculate LOF values of these objects. Assume that an added object is *p*. According to Definition 1 to Definition 6, we also have the following definitions to find the three levels of influenced objects:

Definition 7. The first-level influenced objects. Given a new added object *p*, the first-level influenced objects of *p* contains every object *x* whose distance from *p* is not greater than *k-distance*(*x*), and the first-level influenced objects can be defined as:

$$F_1(p) = \{x | (x \in D \backslash \{p\}) \wedge d(x,p) \leq k - distance(x)\} \quad (5)$$

The new object *p* makes $N_k(x)$ change, and then leads to the subsequent change of $N_{mk}(x)$, $lrd_{mk}(x)$ and $LOF_{mk}(x)$.

Definition 8. The second-level influenced objects. The second-level influenced objects of *p* contains every object *y* whose distance from *x* (object in $F_1(p)$) is not greater than *mk-distance*(*y*), and the second-level influenced objects can be defined as:

$$F_2(p) = \{y | (y \in D \backslash F_1(p)) \wedge (x \in F_1(p)) \wedge d(y,x) \leq mk - distance(y)\} \quad (6)$$

These second-level influenced objects remain $N_{mk}(y)$ unchanged, but should recalculate $lrd_{mk}(y)$ and $LOF_{mk}(y)$ due to the change of *mk-distance*(*x*).

Definition 9. The third-level influenced objects. The third-level influenced objects of *p* contains every object *z* whose distance from *y* (object in $F_2(p)$) is not greater than *mk-distance*(*z*), and the third-level influenced objects is:

$$F_3(p) = \{z | (z \in D \backslash \{F_1(p) \cup F_2(p)\}) \wedge (y \in F_2(p)) \wedge d(z,y) \leq mk - distance(z)\} \quad (7)$$

The third-level influenced objects only $LOF_{mk}(z)$ changed. Based on the abovementioned analysis, we find that because of the addition of object *p*, there are only three levels influenced objects need to recalculate their LOF_{mk} values. Other objects' LOF_{mk} values remain unchanged.

Definition 10. The set of influenced objects whose LOF_{mk} values to be recalculated is *F*.

$$F = F_1(p) \cup F_2(p) \cup F_3(p) \quad (8)$$

In the Dynamic LOF, we firstly obtain the *k*-distance neighborhood N_k, mk-distance neighborhood N_{mk}, local reachability density lrd_{mk} and local outlier factor LOF_{mk} of all the objects in dataset *D*. We put the new objects into the knowledge base, and find in turn the three level influenced objects based on the new knowledge base. Finally, we update N_k, N_{mk}, lrd_{mk} and LOF_{mk} of these objects, while the LOF_{mk} values of other objects remain unchanged. Finally, the LOF_{mk} value of each incoming new object will be calculated according to the unceasing updating of the knowledge base. If the LOF_{mk}

value of a new object is smaller than a given threshold (an empirical value, equal to 2.0), it is normal. Otherwise it is outlying. Algorithm 1 describes the process of finding the first-level influenced objects after adding a new object p into dataset D.

Algorithm 1. Find First Level Objects (D, $F_1(p)$, p).

0: Initialize $F_1(p)$
1: **for** $d(x,p) \leq k-distance(x)$ **do** // The new object p is in the k-distance neighborhood of object x
2: input x into $F_1(p)$; // Construct the set of the first-level influenced objects
3: **end for**
4: input p into D
5: input p into $F_1(p)$ // p is also contained in $F_1(p)$

$F_1(p)$ indicates the set of the first-level influenced objects (including p). If p is in the k-distance neighborhood of object x, then the object x is a first-level influenced object, and should be put into $F_1(p)$. In the end, all of objects in $F_1(p)$ should be recalculated the values of N_k, N_{mk}, lrd_{mk} and LOF_{mk}, so does object p, so we also put object p into $F_1(p)$. Algorithms 2 and 3 describe the process of constructing the sets of $F_2(p)$ and $F_3(p)$.

Algorithm 2. Find Second Level Objects (D, $F_2(p)$, $F_1(p)$)

0: Initialize $F_2(p)$
1: **for** all objects x in $F_1(p)\backslash\{p\}$ **do**
2: **for** all objects y in $D\backslash F_1(p)$ && x is in $N_{mk}(y)$ **do**
3: input y into $F_2(p)$; // Construct the set of the second-level influenced objects
4: **end for**
5: **end for**

Algorithm 3. Find Third Level Objects (D, $F_3(p)$, $F_2(p)$).

0: Initialize $F_3(p)$
1: **for** all objects y in $F_2(p)$ **do**
2: **for** all objects z in $D\backslash\{F_1(p) \cup F_2(p)\}$ && y is in $N_{mk}(z)$ **do**
3: input z into $F_3(p)$; // Construct the set of the third-level influenced objects
4: **end for**
5: **end for**

Algorithm 4 describes the update process of $N_k(x)$ and $N_{mk}(x)$, where x is a first-level influenced object.

Algorithm 4. Update K-Distance ($F_1(p)$).

0: **for** all objects x in $F_1(p)\backslash\{p\}$ **do**
1: **if** $d(x,p) = k - distance(x)$ **then**
2: input p into $N_k(x)$;
3: **else if** ($d(x,p) < k - distance(x)$ && there are less than (k-1) objects in $N_k(x)$ **then**
4: input p into $N_k(x)$;
5: **else if** $d(x,p) < k - distance(x)$ && there are (k-1) objects in $N_k(x)$ **then**
6: remove the farthest neighbor in $N_k(x)$;
7: input p into $N_k(x)$;
8: recalculate $k - distance(x)$;
9: recalculate $mk - distance(x)$;
10: recalculate $N_{mk}(x)$;
11: **else** break;
12: **end if**
13: **end for**

According to the different positions which p inserts into, the influence of $N_k(x)$ differs. There are three different cases as follows:

Situation 1. In Figure 1a, $d(x, p) = k\text{-}distance(x)$, that is p falls on the circle of the k-distance neighborhood of object x. Put p into $N_k(x)$ directly.

Situation 2. In Figure 1b, $d(x, p) < k\text{-}distance(x)$, that is p falls within the circle of the k-distance neighborhood of x. If there are less than (k-1) objects within the circle, then put p into $N_k(x)$.

Situation 3. In Figure 1c, $d(x, p) < k\text{-}distance(x)$, p falls within the circle of the k-distance neighborhood of x. If there are exactly $(k - 1)$ objects within the circle, firstly remove the farthest neighbor in $N_k(x)$, then put p into $N_k(x)$ and recalculate $k\text{-}distance(x)$.

After updating of $N_k(x)$ and $k\text{-}distance(x)$, $N_{mk}(x)$ and $mk\text{-}distance(x)$ can also be recalculated.

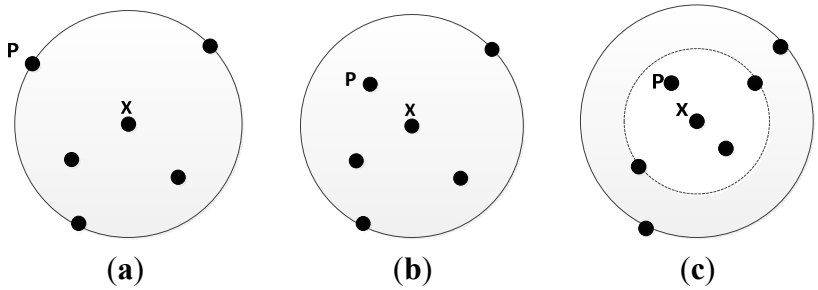


Figure 1. (**a**) Situation 1; (**b**) Situation 2; (**c**) Situation 3.

Algorithm 5 describes the updating process of $lrd_{mk}(x)$ and $lrd_{mk}(y)$, where x is a first-level influenced object and y is a second-level influenced object. Algorithm 6 describes the updating process of $LOF_{mk}(x)$, $LOF_{mk}(y)$, and $LOF_{mk}(z)$, where x is a first-level influenced object, y is the second-level influenced object and z is the third-level influenced object.

Algorithm 5. Update LRD ($F_1(p)$, $F_2(p)$)

0: **for** all objects x in $F_1(p)$ **do**
1: recalculate $lrd_{mk}(x)$;
2: **end for**
3: **for** all objects y in $F_2(p)$ **do**
4: recalculate $lrd_{mk}(y)$;
5: **end for**

Algorithm 6. Update LOF ($F_1(p)$, $F_2(p)$, $F_3(p)$)

0: **for** all objects x in $F_1(p)$ **do**
1: recalculate $LOF_{mk}(x)$;
2: **end for**
3: **for** all objects y in $F_2(p)$ **do**
4: recalculate $LOF_{mk}(y)$;
5: **end for**
6: **for** all objects z in $F_3(p)$ **do**
7: recalculate $LOF_{mk}(z)$;
8: **end for**

3.2. The Fuzzy Linear Regression Process

Considering the possible anomalies of sensed readings, and the uncertain relationships among them, it's quite hard to reflect the fuzzy relationships by the simple linear regression which may cause prediction errors between the regression values and the actual sensed values. It's better to characterize the output variable and regression coefficients by fuzzy numbers.

For medical sensor readings, we consider that the normal sensor readings from several days ago always show similarities, whereas abnormal sensor readings of adjacent times may deviate from each other, so the physiological parameters of a patient at a given time are closely associated with the historical data of adjacent times (which may be within several hours) instead of readings from several days ago. It also means that the impacts of historical data on the estimated outputs are more important at nearer monitoring times.

Based on the limitation mentioned above, we propose a linear regression model based on trapezoidal fuzzy number to predict a more appropriate fuzzy value for the suspected reading. In this regression model, we regard the minimum sum of regression error as a new objective function, and propose a method to obscure the sensor data using the expected value of trapezoidal fuzzy number. In addition, our proposed regression model has given adequate consideration to the different impacts of historical sensor data. By constructing the minimum sum of regression error and fuzzifying readings, we achieve more precise estimated outputs.

Construct the following fuzzy linear function:

$$\widetilde{Y_j} = \widetilde{A_0} + \widetilde{A_1}x_{j1} + \widetilde{A_2}x_{j2} + \cdots + \widetilde{A_n}x_{jn} \quad (9)$$

where n is the number of independent variables, j is j-th data vector on time T_j which is involved in regression modeling. In Equation (9), the prediction value $\widetilde{Y_j}$ and regression coefficient $\widetilde{A_i}(i = 0,1,\cdots,n)$ are fuzzy values, and x_{ji} is i-th measured real number of j-th vector. Define $\widetilde{Y_j}$ and $\widetilde{A_i}$ are trapezoidal fuzzy numbers, so $\widetilde{Y_j} = (y_{aj}, y_{bj}, y_{lj}, y_{rj})$, $\widetilde{A_i} = (a_i, b_i, l_i, r_i)$, the membership function of $\widetilde{Y_j}$ is defined as follows:

$$\mu_y(y_j) = \begin{cases} \dfrac{y_j - y_{aj} + y_{lj}}{y_{lj}}, y_{aj} - y_{lj} < y_j < y_{aj} \\ 1, y_{aj} \le y_j \le y_{bj} \\ \dfrac{y_{bj} + y_{rj} - y_j}{y_{rj}}, y_{bj} < y_j \le y_{bj} + y_{rj} \\ 0, otherwise \end{cases} \tag{10}$$

where:

$$y_{aj} = \sum_{i=1}^{n} a_i x_{ji} + a_0 \tag{11}$$

$$y_{bj} = \sum_{i=1}^{n} b_i x_{ji} + b_0 \tag{12}$$

$$y_{lj} = \sum_{i=1}^{n} l_i |x_{ji}| + l_0 \tag{13}$$

$$y_{rj} = \sum_{i=1}^{n} r_i |x_{ji}| + r_0 \tag{14}$$

These historical sensed readings are real numbers, and must be fuzzified for calculating the corresponding fuzzy regression coefficients $\widetilde{A_i}(i = 0,1,\cdots,n)$. We construct an optimized prediction model of trapezoidal fuzzy numbers so the prediction value is closer to the true value. To resolve the problem of fuzzifying historical values, we introduce a fuzzy number expectation which is a real number with reflecting the values of fuzzy number in average meaning. The expectation of trapezoidal fuzzy number [16] is:

$$\mathrm{E}_{\widetilde{\mathrm{A}}} = \frac{2\mathrm{a} + 2\mathrm{b} - \mathrm{l} + \mathrm{r}}{4} \tag{15}$$

We construct u sets of data to calculate the fuzzy coefficients using measured data. As physiological readings are cyclical, the value of u is the total number of readings that pick out faulty data in a period. Assume that there are totally m groups of measured samples. The following formula represents the m-th sample group in one period:

$$\begin{cases} \mathrm{Y_{1m}}, \mathrm{x_{11}}, \mathrm{x_{12}}, \cdots, \mathrm{x_{1n}} \\ \mathrm{Y_{2m}}, \mathrm{x_{21}}, \mathrm{x_{22}}, \cdots, \mathrm{x_{2n}} \\ \vdots \\ \mathrm{Y_{um}}, \mathrm{x_{u1}}, \mathrm{x_{u2}}, \cdots, \mathrm{x_{un}} \end{cases} \tag{16}$$

where u is the number of samples in one group, n is the number of independent variables. For independent variables $x_{ji}(i = 1,2,\cdots,n, j = 1,2,\cdots,u)$ in the m-th group, we keep the precise real

values unchanged and use them to calculate the fuzzy coefficients. For output variables $Y_{jh}(j = 1,2,\cdots,u, h = 1,2,\cdots,m)$, regard the historical measured data at the same instant but in different groups as multi-measured results. Let y_{bj} and y_{aj} be the maximum and minimum of $Y_{jh}(h = 1,2,\cdots,m)$ respectively:

$$\mathrm{y_{bj}} = \max\{\mathrm{Y_{j1}}, \mathrm{Y_{j2}}, \mathrm{Y_{j3}}, \cdots, \mathrm{Y_{jm}}\} \tag{17}$$

$$\mathrm{y_{aj}} = \min\{\mathrm{Y_{j1}}, \mathrm{Y_{j2}}, \mathrm{Y_{j3}}, \cdots, \mathrm{Y_{jm}}\} \tag{18}$$

The highest and lowest dependent variables in the samples are viewed as parameters of *a* and *b*, so only parameters of y_{lj} and y_{rj} corresponding to $\widetilde{Y_{jm}}$ are changing, and thus, the output variables $\widetilde{Y_{jm}}(j = 1,2,\cdots,u)$ have been fuzzified:

$$\begin{cases} \widetilde{\mathrm{Y_{1m}}}, \mathrm{x_{11}}, \mathrm{x_{12}}, \cdots, \mathrm{x_{1n}} \\ \widetilde{\mathrm{Y_{2m}}}, \mathrm{x_{21}}, \mathrm{x_{22}}, \cdots, \mathrm{x_{2n}} \\ \vdots \\ \widetilde{\mathrm{Y_{um}}}, \mathrm{x_{u1}}, \mathrm{x_{u2}}, \cdots, \mathrm{x_{un}} \end{cases} \tag{19}$$

The least square method is a mathematical procedure for finding the best-fitting curve to a given set of points by minimizing the sum of the squares of the residuals of the points from the curve, so we borrow the ideas from the least square method to ensure the minimal sum of differences of evaluation precision R, it can be defined as follows:

$$R = \sum_{j=1}^{u} \left(\frac{E_{\widetilde{Y_{jm}}} - Y_{jm}}{Y_{jm}} \right)^2 \tag{20}$$

where Y_{jm} is a precise prediction value on *m*-th sample group. Then $E_{\widetilde{Y_{jm}}}$ is calculated as follows:

$$E_{\widetilde{Y_{jm}}} = \frac{2y_{bj} + 2y_{aj} - y_{lj} + y_{rj}}{4} \tag{21}$$

Besides, for each tuple of sensor data, the fuzzy linear regression requires the estimated fuzzy number to contain the observed data with more than the degree of fitting $\lambda_j (0 \le \lambda_j \le 1)$ which is a constant chosen by the decision-maker:

$$\mu_y(y_j) \ge \lambda_j \tag{22}$$

Namely:

$$\begin{cases} \dfrac{y_j - y_{aj} + y_{lj}}{y_{lj}} \ge \lambda_j, y_{aj} - y_{lj} < y_j < y_{aj} \\ \dfrac{y_{bj} + y_{rj} - y_j}{y_{rj}} \ge \lambda_j, y_{bj} < y_j \le y_{bj} + y_{rj} \end{cases} \tag{23}$$

For different readings, λ_j is designed as different membership. In general, the closer to the prediction instant, the more important the readings are, and the higher the corresponding membership is. The parameters to be solved are:

$$Min\ R = \sum_{j=1}^{u} \left(\frac{E_{\widetilde{Y_{jm}}} - Y_{jm}}{Y_{jm}} \right)^2 \tag{24}$$

$$s.t. Y_{jm} = \left(y_{aj}, y_{bj}, y_{lj}, y_{rj}\right) \tag{25}$$

$$y_{aj} = min\{Y_{j1}, Y_{j2}, Y_{j3}, \cdots, Y_{jm}\} \tag{26}$$

$$y_{bj} = max\{Y_{j1}, Y_{j2}, Y_{j3}, \cdots, Y_{jm}\} \tag{27}$$

$$y_{aj} = \sum_{i=1}^{n} a_i x_{ji} + a_0 \tag{28}$$

$$y_{bj} = \sum_{i=1}^{n} b_i x_{ji} + b_0 \tag{29}$$

$$y_{lj} = \sum_{i=1}^{n} l_i \left| x_{ji} \right| + l_0 \tag{30}$$

$$y_{rj} = \sum_{i=1}^{n} r_i \left| x_{ji} \right| + r_0 \tag{31}$$

$$(\sum_{i=1}^{n} a_i x_{ji} + a_0) - (1 - \lambda_j)(\sum_{i=1}^{n} l_i \left| x_{ji} \right| + l_0) \le y_j \tag{32}$$

$$\begin{gathered} (\textstyle\sum_{i=1}^{n} b_i x_{ji} + b_0) + (1 - \lambda_j)(\textstyle\sum_{i=1}^{n} r_i \left| x_{ji} \right| + r_0) \ge y_j \\ y_{lj} > 0, y_{rj} > 0, j = 1,2, \cdots, u, 0 \le \lambda_j \le 1 \end{gathered} \tag{33}$$

Expectation $E_{\widetilde{Y_{jm}}}$ can be linearly described by y_{lj} and y_{rj}. The object function R is a quadratic function and all of restriction conditions are linear. Obviously, this is a nonlinear programming problem with one single objective function and several linear restrictions. In order to find the optimal solution to this problem and reduce the computation cost, we firstly transfer this constrained nonlinear programming problem into unconstrained nonlinear programming problem.

Consider the restrictions in this nonlinear programming model, Equations (28)–(33) can be rewritten as follows:

$$h_1 = y_{aj} - \sum_{i=1}^{n} a_i x_{ji} + a_0 \tag{34}$$

$$h_2 = y_{bj} - \sum_{i=1}^{n} b_i x_{ji} + b_0 \tag{35}$$

$$h_3 = y_{lj} - \sum_{i=1}^{n} l_i \left| x_{ji} \right| + l_0 \tag{36}$$

$$h_4 = y_{rj} - \sum_{i=1}^{n} r_i \left| x_{ji} \right| + r_0 \tag{37}$$

$$h_i = 0, i = 1,2,3,4 \tag{38}$$

$$g_1 = y_j - ((\sum_{i=1}^{n} a_i x_{ji} + a_0) - (1 - \lambda_j)(\sum_{i=1}^{n} l_i \left| x_{ji} \right| + l_0)) \tag{39}$$

$$g_2 = (\sum_{i=1}^{n} b_i x_{ji} + b_0) + (1 - \lambda_j)(\sum_{i=1}^{n} r_i |x_{ji}| + r_0) - y_j \quad (40)$$

$$g_i = 0, i = 1,2 \quad (41)$$

Then we construct the following unconstrained nonlinear programming model which is equivalent to the original constrained nonlinear programming model:

$$\sum_{j=1}^{u} \left(\frac{E_{\widetilde{Y_{jm}}} - Y_{jm}}{Y_{jm}} \right)^2 + \sum_{i=1}^{4} |h_i| + \sum_{i=1}^{2} |min(0, g_i)| \quad (42)$$

To find the optimal solution for this model, many traditional solutions have been proposed, such as the Lagrange multiplier method or Conjugate Gradient method. Meanwhile, several heuristic algorithms such as Genetic Algorithms (GA) [17–19] also play an important role in solving this problem, because it is an efficient method to deal with the nonlinear programming problem with high-complexity and multi-parameters.

We use a genetic algorithm to get the optimal solution to Equation (42), so as to ensure the minimal sum of differences R when each membership of prediction variable is not lower than λ_j. Then we get the fuzzy representation of historical sensed readings and the least square estimation $\widetilde{\widehat{A}_\iota} = (\widehat{a_\iota}, \widehat{b_\iota}, \widehat{l_\iota}, \widehat{r_\iota})$ that is related to the fuzzy regression coefficients $\widetilde{A_\iota}(i = 0,1, \cdots, n)$. The normal data will be involved in the prediction after getting the fuzzy coefficients. Finally, we get the fuzzy prediction value of the given sensor.

3.3. Fault Judgment Criterion

We define the following fault judgment criterion. As shown in Table 1, the black trapezoid indicates the prediction result of the fuzzy number $\widetilde{Y_p}$. λ is the similarity membership degree and $0.5 < \lambda < 1$. $(a - l)$ and $(b + r)$ are the lower bound and upper bound of the fuzzy number, a and b are corresponding values of the membership degree λ. The red line indicates the normal interval of the corresponding physiological parameter. low and $high$ are the lower bound and upper bound of the normal interval. y indicates the actual sensed readings.

As shown in Table 1, whether a sensor is good or faulty depends on different relative position relationships between the prediction result of the fuzzy number and the normal intervals of the corresponding physiological parameter. In addition, it depends on which interval the actual sensed readings lie in. The undecided state indicates that the state of the corresponding sensor cannot be determined in this detection round because physiological indicators of patients may not synchronously change at the same time. It may be caused by a detection delay or a very slow change of the physiological parameter. We continue to detect the undecided faulty nodes in successive detection rounds. If the undecided state of the same sensor remains unchanged after the following several rounds of detection, then it will be viewed as faulty. The undecided state in judgment rule is suitable for various readings' out-of-sync status.

Table 1. The relationships between $\widetilde{Y_p}$ and y.

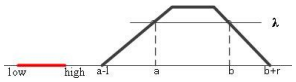
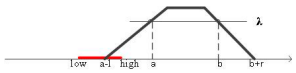
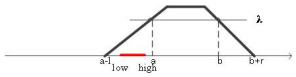
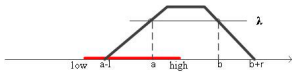
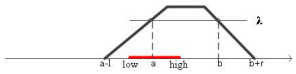
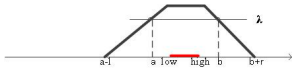
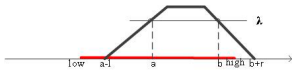
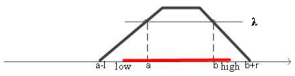
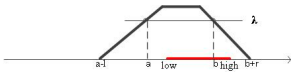
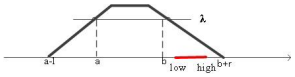
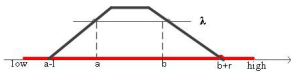
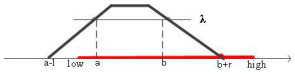
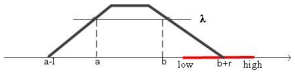
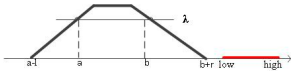
The Relationships between $\widetilde{Y_p}$ and Y	Different Intervals Which the Actual Sensed Readings Lie in	
	Normal Interval	***Anomalous Interval***
	1. $low \le y \le high$, fault	1. $y < low$, fault 2. $high < y < a$ undecided 3. $a \le y \le b$ good 4. $b < y \le b + r$, undecided 5. $b + r < y$, fault
	1. $low \le y < a - l$, fault 2. $a - l \le y \le high$, undecided	1. $y < low$, fault 2. $high < y < a$, undecided 3. $a \le y \le b$, good 4. $b < y \le b + r$, undecided 5. $b + r < y$, fault
	1. $low \le y \le high$, undecided	1. $y < a - l$, fault 2. $a - l \le y < low$, undecided 3. $high < y < a$, undecided 4. $a \le y \le b$, good 5. $b < y \le b + r$, undecided 6. $b + r < y$, fault
	1. $low \le y < a - l$, fault 2. $a - l \le y < a$, undecided 3. $a \le y \le high$, good	1. $y < low$, fault 2. $high < y \le b$, good 3. $b < y \le b + r$, undecided 4. $b + r < y$, fault
	1. $low \le y < a$, undecided 2. $a \le y \le high$, good	1. $y < a - l$, fault 2. $a - l \le y < low$, undecided 3. $high < y \le b$, good 4. $b < y \le b + r$, undecided 5. $b + r < y$, fault
	1. $low \le y \le high$, good	1. $y < a - l$, fault 2. $a - l \le y < a$, undecided 3. $a \le y < low$, good 4. $high < y \le b$, good 5. $b < y \le b + r$, undecided 6. $b + r < y$, fault
	1. $low \le y < a - l$, fault 2. $a - l \le y < a$, undecided 3. $a \le y \le b$, good 4. $b < y \le high$, undecided	1. $y < low$, fault 2. $high < y \le b + r$, undecided 3. $b + r < y$, fault

Table 1. *Cont.*

The Relationships between $\widetilde{Y_p}$ and Y	Different Intervals which the Actual Sensed Readings Lie in	
	Normal Interval	***Anomalous Interval***
	1. $low \le y < a$, undecided 2. $a \le y \le b$, good 3. $b < y \le high$, undecided	1. $y < a - l$, fault 2. $a - l \le y < low$, undecided 3. $high < y \le b + r$, undecided 4. $b + r < y$, fault
	1. $low \le y \le b$, good 2. $b < y \le high$, undecided	1. $y < a - l$, fault 2. $a - l \le y < a$, undecided 3. $a \le y < low$, good 4. $high < y \le b + r$, undecided 5. $b + r < y$, fault
	1. $low \le y \le high$, undecided	1. $y < a - l$, fault 2. $a - l \le y < a$, undecided 3. $a \le y \le b$, good 4. $b < y < low$, undecided 5. $high < y \le b + r$, undecided 6. $b + r < y$, fault
	1. $low \le y < a - l$, fault 2. $a - l \le y < a$, undecided 3. $a \le y \le b$, good 4. $b < y \le b + r$, undecided 5. $b + r < y \le high$, fault	1. $y < low$, fault 2. $high < y$, fault
	1. $low \le y < a$, undecided 2. $a \le y \le b$, good 3. $b < y \le b + r$, undecided 4. $b + r < y \le high$, fault	1. $y < a - l$, fault 2. $a - l \le y < low$, undecided 3. $high < y$, fault
	1. $low \le y \le b$, good 2. $b < y \le b + r$, undecided 3. $b + r < y \le high$, fault	1. $y < a - l$, fault 2. $a - l \le y < a$, undecided 3. $a \le y < low$, good 4. $high < y$, fault
	1. $low \le y \le b + r$, undecided 2. $b + r < y \le high$, fault	1. $y < a - l$, fault 2. $a - l \le y < a$, undecided 3. $a \le y \le b$, good 4. $b < y < low$, undecided 5. $high < y$, fault
	1. $low \le y \le high$, fault	1. $y < a - l$, fault 2. $a - l \le y < a$, undecided 3. $a \le y \le b$, good 4. $b < y < low$, undecided 5. $high < y$, fault

We also find that if the actual sensed readings lie in the interval that corresponds to similarity membership degree λ, and then the sensor is regarded as good. If the actual output sensor data is greater than the larger one of $(b + r)$ and $high$, or less than the smaller one of $(a - l)$ and low, then the sensor is faulty. Otherwise, the sensor is considered as undecided.

4. Simulation Results

In this section, we analyze the performance results of DFD-M and typical algorithms. We obtain the variations of physiological readings including Heart Rate (HR), Respiration Rate (RR), Body Temperature (BT) and Oxygen Saturation (SpO2) from medical sensors. The HR fluctuates from 40 to 140 bpm, which may be caused by movements or physiological abnormalities. RR fluctuates around 20 bpm, and is approximately proportional to HR. SpO2 fluctuates from 90 to 100, and only in a rare case that SpO2 is lower than 90. With a few exceptions, BT basically remains stable around 37 °C. To simulate sensor failure, we artificially insert some failure data into the monitoring data.

All the algorithms are implemented in Matlab R2010a. Our simulations are divided into three parts: (1) simulations for the *D*-LOF algorithm compared with the Kernel-LOF (*Ker*-LOF) [20] and the original LOF [15] to verify the detection accuracy rate and time complexity; (2) simulations for fuzzy linear regression based on trapezoidal fuzzy numbers (*FLR-Tra*), simple linear regression (*SLR*) and fuzzy linear regression based on triangle fuzzy numbers (*FLR-Tri*) to demonstrate the evaluation error; (3) simulations for the whole detection performances of *DFD-M,* the fault detection algorithm (*DA-J48*) proposed in [11], and using *FLR-Tri* and *SLR* to replace the regression model in our proposed algorithm.

4.1. Simulations for LOF Values

In this part, we show the results of detection accuracy rate and time complexity. To simulate the dynamic increment of dataset, 83% instances are in the initial knowledge base, and 17% are viewed as new updated objects. In Figure 2, the simulation settings are k = 10 (in the denotation *k-distance*), and the maximal size of dataset is 5×10^4 with 15 outliers. As a given threshold of LOF_{mk} is 2.0, LOF values of most of instances fluctuate around 1.0. The 15 instances whose LOF values are greater than 2.0 can be completely detected.

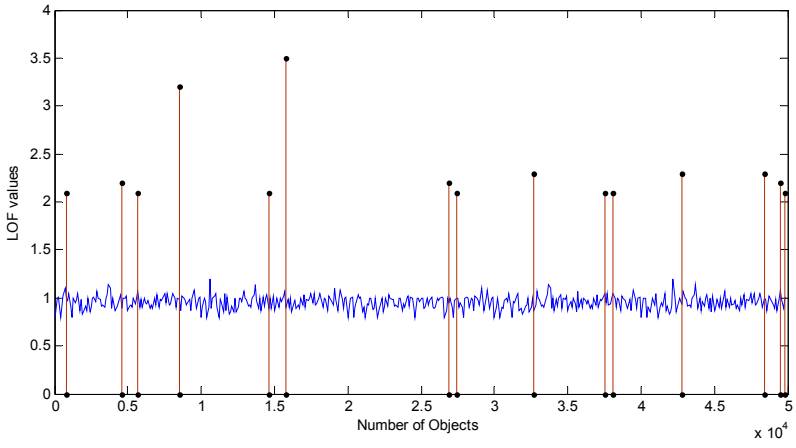


Figure 2. LOF values of objects with 15 outliers (k = 10).

As parameter k (in the notation *k-distance*) influenced the performance of algorithms, the Figure 3 shows the detection rate for outliers with different k when size of dataset is 2×10^4 and the number of outliers is 40. When setting $k = 20$, *D*-LOF has a perfect detection rate with 99.1%, while *Ker*-LOF is 98.7% and original LOF is 95.5%. Changing values of k, the detection rates for outliers accordingly decline. But our algorithm still remains a higher performance compared with others.

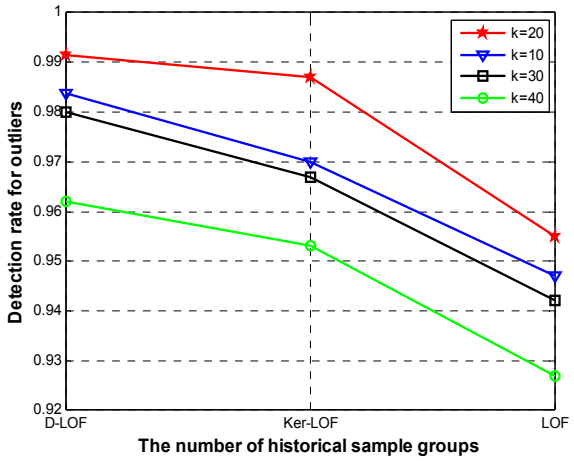


Figure 3. Detection rate for outliers with different k values.

To further analyze the influences caused by k value and the sizes of datasets, we set datasets be two-dimensional instances whose sizes are respectively 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5 with $\times 10^4$, while there are 25, 35, 45, 55, 65, 75, 85, 95, 105, 115 outliers randomly distributed in these datasets respectively. Figure 4 shows the detection rates for outliers with different k when increasing the sizes of datasets in our algorithms. The optimized relations with the highest detection rates are respectively: 1×10^4 with $k = 10$, 2×10^4 with $k = 20$, 2×10^4 with $k = 20$, 4×10^4 with $k = 30$, and 5×10^4 with $k = 40$. And the detection rates are respectively 99.8%, 99.1%, 97.7%, 96.3% and 94.2% in these situations. For example, the best rates are respectively 2.3% and 1.8% higher than the worst rates in situations of 4×10^4 with $k = 30$ and 5×10^4 with $k = 40$.

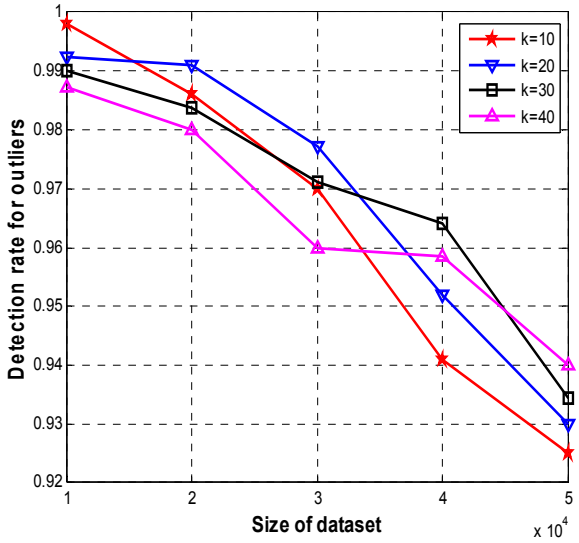


Figure 4. Detection rate for outliers with different k.

Figure 5 shows the running time of *D*-LOF, *Ker*-LOF and original LOF. We vary the values of parameter k related to the number of neighbors in the domain of an object. The time taken by these algorithms increases as the size of dataset increases, while *D*-LOF has a much lower running time than others. When $k = 10$ and the size of dataset is 5×10^4 with 105 outliers, run time of *D*-LOF is 56 s, which is 53.3% lower than that of original LOF, and 49.2% lower than that of *Ker*-LOF. With the same dataset size, when $k = 40$, running time of *D*-LOF is 81 s. It is 61.9% lower than that of original LOF, and 56.7% lower than *Ker*-LOF. The time complexity is sharply reduced because only three levels of influenced objects update their LOF values. We also find that when the size of dataset is unchanged, the smaller the value of k is, the faster the outliers are detected out.

Figure 6 shows the detection rate of *D*-LOF compared with *Ker*-LOF and original LOF when the values of k are set as the best optimized according to Figures 3 and 4. When the size of the dataset is smaller, they almost achieve 100% detection rate. As the size of dataset is increasing, all of detection rates slowly decline. Because *D*-LOF is more sensitive to outliers, it is clear that of *D*-LOF performs better than the other two and achieves a higher detection rate for different dataset sizes. Even though the size of the dataset is 5×10^4 with 115 outliers, 109 outliers can be detected by *D*-LOF with a detection rate 94.8%, while that of the original LOF is 87.8%, and for *Ker*-LOF it is 92.4%.

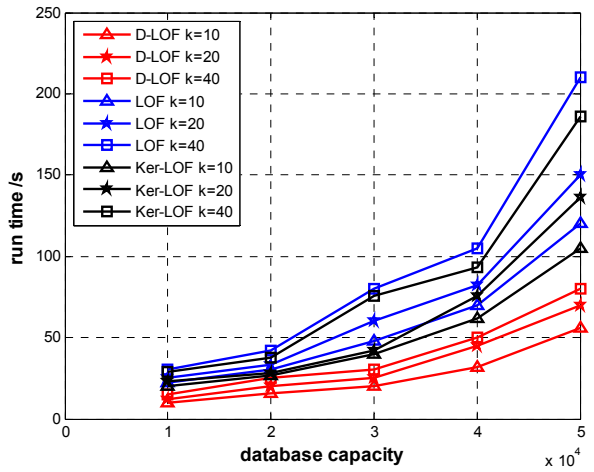


Figure 5. Run time with different k.

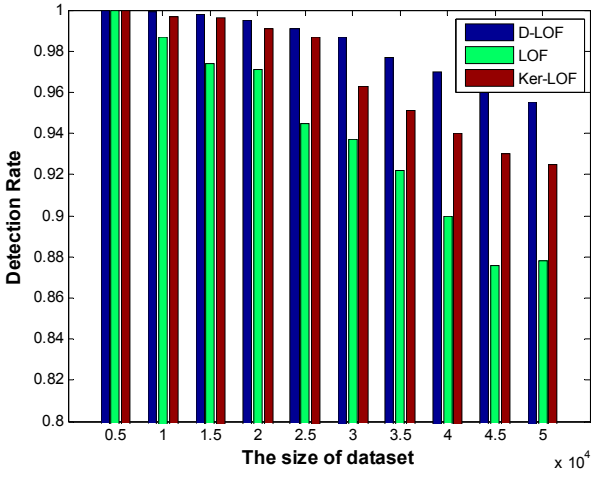


Figure 6. Detection rate with different optimized k.

4.2. Simulations for Evaluation Error R

In this part, we calculate the differences of evaluation precision for *FLR-Tra* in our algorithm compared with *SLR* and *FLR-Tri.* We represent the sum of differences evaluates precision *R* (defined in III. B section) of the three regression models in Figure 7. It shows the trends from rising to decline with the increasing number of historical sample groups in the linear regression model. That is because, in the initial stage, an increase in the number of historical samples leads to a significant increase of *R*. When the number of historical sample is 20, the three regression models achieve similar evaluation precision differences, that is 0.0175, 0.0168, and 0.0162 of *SLR*, *FLR-Tri* and *FLR-Tra.* The *FLR-Tra* achieves the top point when the number of samples is 80. When the number of historical samples is 100, the sum of differences precision evaluation of *FLR-Tra* reduces to 0.0143, which is 39.1% lower than that of *FLR-Tri* and 70.8% lower than that of *SLR*. That is, as the number of historical samples becomes even larger, the fuzzified historical values get closer to their actual values, make our regression more accurate, and then *R* declines.

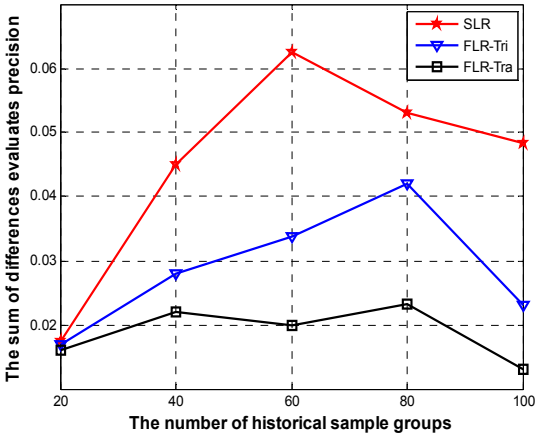


Figure 7. The sum of differences of evaluation precision R.

In Figure 8, the mean of the sum of differences evaluates precision $\bar{R}$ has an obvious decrease for *FLR-Tra* and *FLR-Tri*, while that of *SLR* rises at the initial stage, and then declines sharply. *FLR-Tra* achieves the best performance on $\bar{R}$.

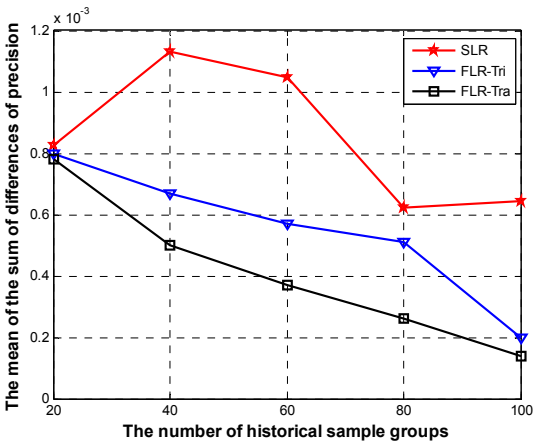


Figure 8. The mean of the sum of difference of evaluation precision $\bar{R}$.

4.3. Simulations for Whole Detection Performances of DFD-M

Alarms are reported when readings exceed the normal range. The main task of *DFD-M* is to distinguish data failure from alarms. To simplify for visualization, we only adopt readings of RR and HR in the results of Figure 9, which shows the alarm reporting scene without fault detection. There are in total 19 alarms raised presented in vertical red lines. It reports exceptions for HR and RR. Figure 10 shows the alarms reported by *DFD-M*. Only nine alarms remain in this chart, which excludes the other 10 alarms caused by sensor data faults.

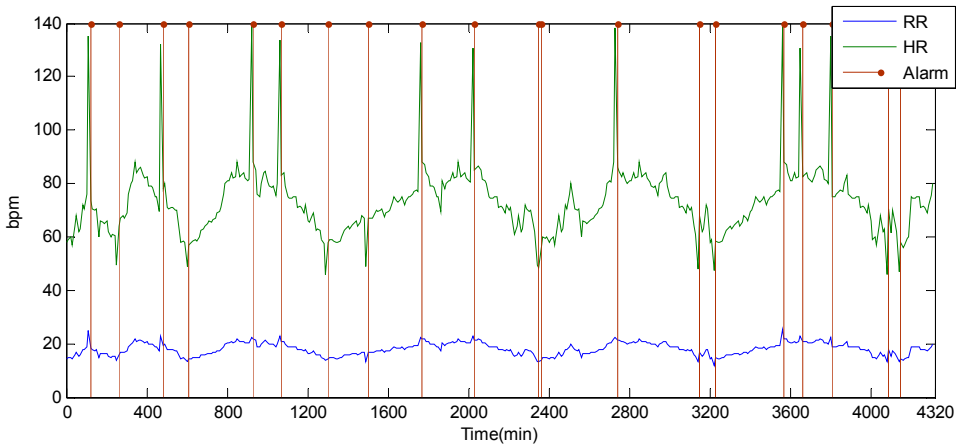


Figure 9. Undetected alarms.

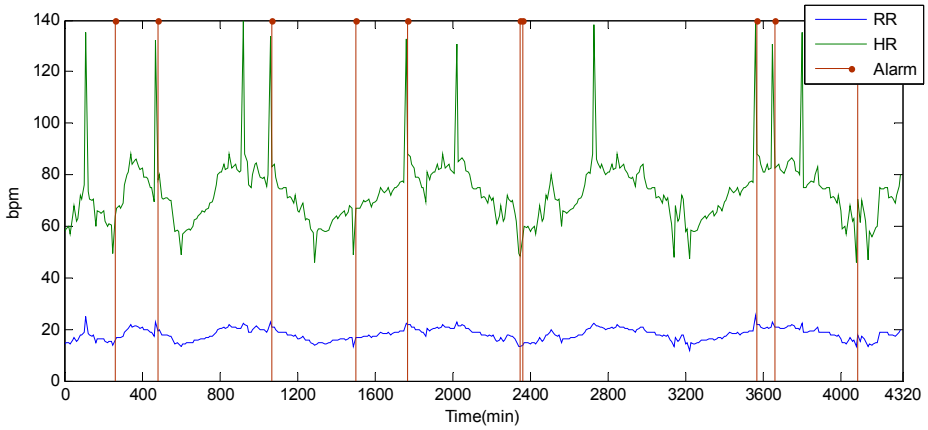


Figure 10. Alarms by DFD-M.

To further evaluate the performance of *DFD-M*, we add new two types of readings: blood glucose and blood pressure. When the size of the data is increasing, false alarms inevitably happen. In this part, we mainly analyze the detection accuracy rate and false alarm rate. The latter is the ratio of number of real readings that are misjudged as false to the sum of readings. Figure 11 shows the detection accuracy rate for different data sizes when varying the data dimensions. The smaller the dataset size and dimension are, the higher the detection rate is. The lowest accuracy is 84.6% with 6 dimensions and data size of 5×10^4. Figure 12 shows the average of detection accuracy rate with increasing dimensions. *DFD-M* achieves almost 100% with two data attributes. With the increasing data dimensions, all of the accuracy rates of these algorithms decline slightly. When adding up to six dimensions, the

average accuracy rate of *DFD-M* reaches a high level with 97.2%. We use *FLR-Tri* and *SLR* to replace the regression model in our proposed algorithm. The detection accuracy using *FLR-Tri* is 95.1% and only 91.3% in *SLR* with six dimensions. While *DA-J48* has the lowest detection rate 90.5%.

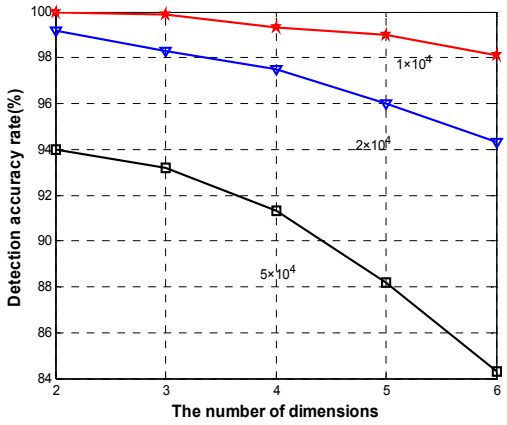


Figure 11. Detection accuracy rate for different data sizes of DFD-M.

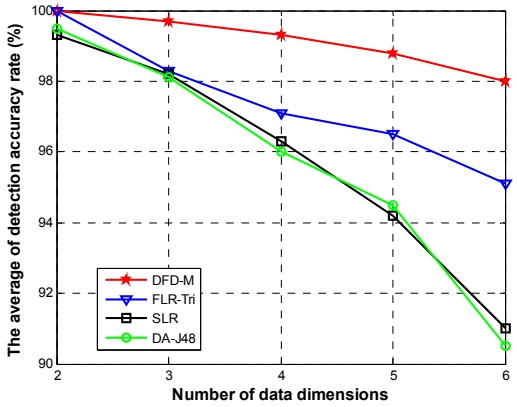


Figure 12. Detection accuracy rate.

Figure 13 shows the false alarm rate for different data sizes when varying data dimensions. The averages of false alarm rates of *DFD-M* are respectively 2.33%, 3.06%, and 5.58% for different data sizes, while that of *DAJ-48* are respectively 4.3%, 6.26% and 9.8%.

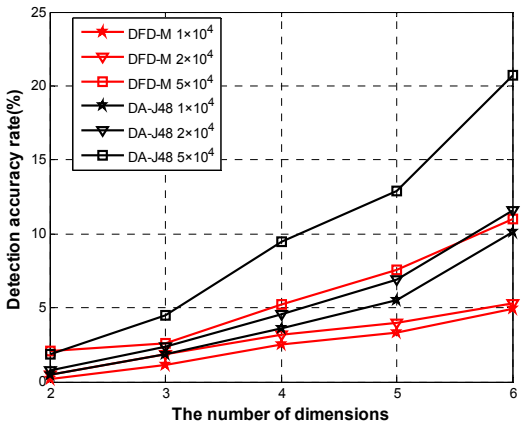


Figure 13. False alarm rate for different data sizes.

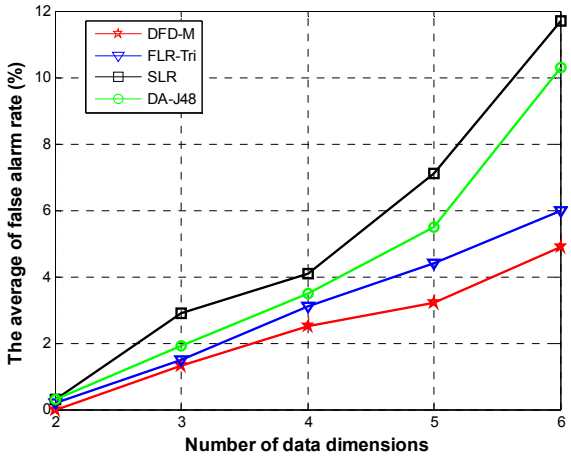


Figure 14. False alarm rate.

Even though the dimensions increase to six and the size of dataset is 5×10^4, *DFD-M* still has a lower false alarm rate of 11.2%, 45.3% lower than that of *DAJ-48*. Figure 14 shows the average of false alarm rates for the abovementioned four algorithms. When adding dimensions, the false alarm rates of these four algorithms increase slowly. When adding up to five dimensions, the false alarm rate of *DFD-M* is only 3.2%, while that of *FLR-Tri* is 4.4%, *SLR* is 7.1% and *DA-J48* is 5.5%. When adding up to six dimensions, the false alarm rate of *DFD-M* is 4.9%, which is 18.3% lower than that of *FLR-Tri*, 55.5% lower than that of *SLR*, and 50.4% lower than that of *DA-J48*. The above experimental results show that *DFD-M* achieves more superior performance than the others.

5. Conclusions

The paper proposes a medical sensor fault detection mechanism for data failure called DFD-M. It firstly identifies outlying data vectors, and then uses an improved fuzzy linear regression model to predict the reasonable range for outlying data, and finally it analyzes the relationships between the fuzzy prediction results and the normal intervals by using a novel fault state judgment criterion. The simulations demonstrate that DFD-M has a higher detection accuracy rate and lower false alarm rate that other similar algorithms.

Acknowledgments

This work was partly supported by NSFC (61401033), Ph.D. Programs Foundation of Ministry of Education of China (No. 20110005110011), Fundamental Research Funds for the Central Universities (No. 2014RC1102), and Beijing Higher Education Young Elite Teacher Project (YETP0474).

Author Contributions

Yang Yang and Zhipeng Gao were responsible for researching and designing the mechanisms presented in the paper; Qian Liu was involved in simulating and testing the algorithms; Xuesong Qiu and Luoming Meng edited the manuscript.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Berset, T.; Romero, I.; Young, A.; Penders, J. Robust heart rhythm calculation and respiration rate estimation in ambulatory ECG monitoring. In Proceedings of the IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI), Hong Kong, China, 5–7 January 2012; pp. 400–403.
2. Rotariu, C.; Manta, V. Wireless system for remote monitoring of oxygen saturation and heart rate. In Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS), Wroclaw, Poland, 9–12 September 2012; pp. 193–196.
3. Mahapatro, A.; Khilar, P.M. Online Fault Detection and Recovery in Body Sensor Networks. In Proceedings of the World Congress on Information and Communication Technologies (WICT), Mumbai, India, 11 December 2011; pp. 407–412.
4. Kim, D.J.; Suk, M.H.; Prabhakaran, B. Fault Detection and Isolation in Motion Monitoring System. In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), San Diego, CA, USA, 28 August 2012; pp. 5234–5237.
5. Nguyen, L.; Su, S.; Nguyen, H.T. Effects of hyperglycemia on variability of RR, QT and corrected QT intervals in Type 1 diabetic patients. In Proceedings of the 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Osaka, Janpa, 3–7 July 2013; pp. 1819–1822.
6. Ma, Q.; Liu, K.; Xiao, X.; Cao, Z.; Liu, Y. Link Scanner: Faulty Link Detection for Wireless Sensor Networks. In Proceedings of the INFOCOM, Turin, Italy, 14–19 April 2013; pp. 2688–2696.
7. Jiang, P. A new method for node fault detection in wireless sensor networks. *Sensors* **2009**, *9*, 1282–1294.
8. Ding, M.; Chen, D.; Xing, K.; Cheng, X. Localized fault-tolerant event boundary detection in sensor networks. In Proceedings of the INFOCOM, Miami, FL, USA, 13–17 March 2005; pp. 902–913.
9. Krishnamachari, B.; Iyengar, S. Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. *IEEE Trans. Comput.* **2004**, *53*, 241–250.
10. Lee, M.H.; Choi, Y.H. Fault detection of wireless sensor networks. *Comput. Commun.* **2008**, *31*, 3469–3475.
11. Ruiz, L.B.; SiqueiraI, G.; Oliveira, L.B.; Wong, H.C.; Nogueira, J.M.S.; Loureiro, A.A.F. Fault management in event-driven wireless sensor networks. In Proceedings of the ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Venice, Italy, 4–6 October 2004; pp. 149–156.

12. Salem, O.; Guerassimov, A.; Mehaoua, A.; Marcus, A.; Furht, B. Sensors Fault and Patient Anomaly Detection and Classification in Medical Wireless Sensor Networks. In Proceedings of the International Conference on Communications (ICC), Budapest, Hungary, 9–13 June 2013; pp. 4373–4378.
13. Salem, O.; Yaning, L.; Mehaoua, A. A Lightweight Anomaly Detection Framework for Medical Wireless Sensor Networks. In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), Shanghai, China, 7–10 April 2013; pp. 4358– 4363.
14. Miao, X.; Liu, K.; He, Y.; Liu, Y.; Papadias, D. Agnostic diagnosis: Discovering silent failures in wireless sensor networks. In Proceedings of the INFOCOM, Shanghai, China, 10–15 April 2011; pp. 1548–1556.
15. Markus, M.B.; Hans, P.K.; Raymond, T.NG.; Jorg, S. LOF: Identifying density-based local outliers. In Proceedings of the ACM SIGMOD Internatioanl Conference on Management of Data, Dallas, TX, USA, 16 May 2000; pp. 93–104.
16. Zeynep, S.E.; Etrugrul, K. A fuzzy regression and optimization approach for setting target levels in software quality function deployment. *Softw. Qual. Control* **2010**, *18*, 323–339.
17. Krishnakumar, K. Micro-genetic algorithms for stationary and non-stationary function optimization. In Proceedings of the International Society for Optics and Photonics, Los Angeles, CA, USA, 14 January 1990; pp. 289–296.
18. Goldberg, D.E.; Richardson, J. Genetic algorithms with sharing for multimodal function optimization. In Proceedings of the International Conference on Genetic Algorithms, Hillsdale, NJ, USA, 28–31 July 1987; pp. 41–49.
19. Abd-El-Wahed, W.F.; Mousa, A.A.; El-Shorbagy, M.A. Integrating particle swarm optimization with genetic algorithms for solving nonlinear optimization problems. *J. Comput. Appl. Math.* **2011**, *235*, 1446–1453.
20. Bao, L.; Xiao, Y.; Yu, P.S.; Hao, Z.; Cao L. An efficient approach for outlier detection with imperferct data labels. *IEEE Trans. Knowl. Eng.* **2014**, *26*, 1602–1616.