A Method for Fault Detection in Wireless Sensor Network Based on Pearson’s Correlation Coefficient and Support Vector Machine Classification

Priyajit Biswas1 · Tuhina Samanta1

Accepted: 19 October 2021 / Published online: 1 November 2021
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract
Sensor nodes are tiny low-cost devices prone to various faults. So, it is imperative to detect those faults. This paper presents a sensor measurement fault detection algorithm based on Pearson’s correlation coefficient and the Support Vector Machine (SVM) algorithm. As environmental phenomena are spatially and temporally correlated but faults are somewhat uncorrelated, Pearson’s correlation coefficient is used to measure correlation. Then SVM was used to classify faulty readings from normal readings. After classification, faulty readings are discarded. Here each sensor nodes periodically collects environmental features and sends them to their associated cluster heads. Each cluster head analyze collected data using the classification algorithm to detect whether any fault is present or not. Network simulator NS-2.35 and Matlab are used for evaluation of our proposed method. The fault detection algorithm was evaluated using performance metrics, namely, Accuracy, Precision, Sensitivity, Specificity, Recall, $F_1$ Score, Geometric Mean (G_mean), Receiver Operating Characteristics (ROC), and Area Under Curve (AUC). Performance evaluation shows, the proposed method performs well for high fault percentages.

Keywords Wireless sensor network · Machine learning algorithm · Correlation coefficient · SVM · Measurement fault

1 Introduction

Wireless Sensor Network (WSN) consists of a large number of sensor nodes scattered over a large spacial region with one or more base stations (BS). The basic functionality of the WSN node is to collect information and send it to BS for analysis [10, 19]. These sensor nodes are tiny, low-cost devices equipped with one or more sensors. Sensor nodes are prone to various faults [8]. There are many factors involved that can cause a fault, such as...
hardware failure, software fault, communication error, etc. [21] These faults may persist for a long time or may be instantaneous. Since faults are unavoidable, discovering faulty and fault-free nodes is crucial in the field of WSN [4]. The fault detection algorithm needs to be designed in such a way that it can detect such inconsistent behavior that can disrupt the normal functionality of the WSN. Sensors are deployed to monitor the region, and if any suspicious event occurs, report the same to the base station. Moreover, these cheap nodes are prone to various faults, including measurement faults. This implies that sensor measurements are not reliable. So the event can’t be concluded based on a single measurement instance. Measurements need to be correlated spatially and temporally to decide whether a measurement is faulty, normal, or event. It is shown that environmental phenomena are spatially and temporally correlated [1]. In this work, this property was exploited to detect faulty reading. The Person’s correlation coefficient [20] between two-time slots was utilized to detect the presence of a fault. Then SVM [2] was used to classify between faulty reading and normal reading.

The contribution in this work is as follows,

- Pearson’s correlation coefficient ($\rho$) was used to measure the correlation between two-time slots. $\rho$ is invariant of scaling and self-normalizing, which makes it suitable for spatially correlated environmental features.
- SVM was used to classify faulty and normal reading. SVM performs well in higher dimensional feature space. Moreover, using the kernel trick of SVM, nonlinear data can also be solved. As the used dataset is nonlinear and has two features, namely humidity, and temperature, SVM is suitable for the proposed method.
- NS2 and Matlab are used to simulate and analyze our proposed method, respectively. The proposed method was also tested over a real-world dataset.

The rest of the paper is organized as follows: related works are discussed in Sect. 2. Problem definition is given in Sect. 3. Preliminaries are described in Sect. 4. Our proposed method is presented in Sect. 5. Simulation results and performance evaluation are shown in Sect. 6. The conclusion is drawn in Sect. 7.

### 2 Related Works

In [1], authors proposed General anomaly detection(GAD), a fully distributed scheme for practical large scale networked industrial sensing systems(NISSs). Real-time detection, Distributed solution, and General solution are three properties of GAD. Their method, distributed matching-based grouping algorithm (DMGA), divides all sensing components into small, strongly correlated groups in a fully distributed way. Spatial correlation is a natural property in various physical phenomena. “Since physical phenomena are continuous, these spatial correlations should be temporally correlated to previous mappings.” They assumed measurement errors follow Gaussian distributions. They evaluated their method using a successful detection rate (SDR) and false-positive detection rate (FPDR).

In [21], authors used SVM with Gaussian kernel to detect faults in WSN. They classified Data faults as Offset fault, Gain fault, Stuck-at fault, Out of bounds. In their proposed method, received sensor data is classified using SVM with Gaussian kernel function. Their learning phase is performed in BS. Then, the decision function was transmitted to each cluster head to classify new measured data. In their proposed method, every time a new
data is measured, an observation vector composed of last three data measurement of two sensors \((V_t, V_{t-1}, V_{t-2})\) is constructed by the data preparation block although SVM is capable of multidimensional classification. After that, this observation vector is classified by SVM using the decision function. If SVM output is positive new data is normal else data is faulty. Their labeled dataset is based on an existing dataset published by the researchers of the University of North Carolina at Greensboro [12].

A fault diagnosis protocol based on gradient descent and evolutionary approach was proposed by Swain et al. in [13]. Their method detects the faulty nodes and isolates them from the network. Their proposed protocol comprises four phases, namely clustering phase, communication phase, fault detection and classification phase, and isolation phase. They used a genetic algorithm and neural network for fault detection. Their method classifies faults into four types according to the rate of faults. Then faults are isolated. Their comparison showed that evolutionary approach outperforms gradient descent for their sensor data.

In [14], the authors proposed a fault diagnosis protocol that detects heterogeneous faults in WSN. Their protocol is capable of detecting soft permanent, hard permanent, transient, and intermittent faults, as claimed by authors. In their method, hard permanent faults are identified by a time out status register mechanism. Soft permanent, intermittent, and transient faults are detected by a statistical test, namely analysis of variance (ANOVA) test. In this phase, \(m\) number of measurements of \(n\) nodes in a cluster was used for the ANOVA test. This test was repeated for \(r\) times to detect faults. They also used the feed-forward probabilistic neural network (PNN) to classify these heterogeneous faults. However, as the protocol relies on the coordinator node, any inconsistent behavior such as node failure or erroneous results of the coordinate node will lead to the degradation of the performance of the protocol.

In [11], Saeedi et al. proposed a density-based spatial clustering of applications with noise (DBSCAN) algorithm for detecting anomalies. Their proposed method used three features from eight features of the Intel Berkeley Research lab (IRLB) dataset. A significant portion of the data is normal, and they are stored in CH with high Density. Authors utilized this idea in their work. DBSCAN algorithm identifies low-density regions and labels them as an anomaly. Then they trained the SVM classifier using these labeled data.

In [5], authors objective was to design a fault detection and diagnosis system using existing ML techniques. They used three machine-learning algorithms, namely, Fuzzy Deep Neural Network (FDNN), Support vector machine (SVM), and Neural Network (NN), for fault detection in their work.

In [15], authors proposed a complete fault diagnosis methodology to detect faulty sensors in WSN. Their proposed method contains four phases, namely, initialization phase, fault detection phase, fault classification phase, and fault tolerance phase. They used checksum and Fletcher’s checksum method to detect hard fault and link fault. They also used Mann-Whitney U statistical test for soft fault detection. Then they utilized Gaussian transformation function to classify the soft faults. They also used a stepwise regresional method to tolerate fault in their method. The Detection phase is performed on each cluster head with its own and member sensor measurements. In Mann-Whitney U statistical test, \(P\)-value decides the status of the sensor measurement value. They used a threshold value \(\theta\) based on application and situation of the sensor network. If \(P\)-value is less than the threshold, their method declares it as soft fault.

In [17], the authors proposed DODS (Distributed Outlier Detection Scheme), where outliers are detected locally by each node. They used four data types, i.e., temperature, voltage, humidity, and light. The main idea is to clean sensed data (measurements) from outlier (incorrect data). The scheme operates in nodes that made the sensing operation
and does not require any neighbor’s communication. The solution exploits the temporal correlations existing in the sensed data (current and history sensed data) of the same node and its remaining energy level. Outlier detection is performed using Bayes’ classifier for each type of data. Only nodes belong to an interesting region (IR) participate in the outlier detection process. To learn the prior probability and to compute all conditional probability, they used a supervised off-line method. They considered a different set of classes (small, medium, large) for different data types and used the maximum a posteriori (MAP) concept in order to determine optimal class. But their method does not differentiate between faults and events. No spatial correlation is considered in their method.

3 Problem Description

Let, \( N \) number of nodes are deployed in a region. Each node is equipped with \( k \) sensors to sense \( k(k > 1) \) number of environmental features. These nodes periodically sense environmental features and send them to their associated Cluster Head (CH). Upon receiving sensed data, CH starts analyzing them to detect if there is any fault.

Suppose, sensor nodes periodically sense environmental data with time interval \( T \). Then total time is divide into time slots \( t (t = 0, 1, 2 \ldots) \) depending on \( T \) as illustrated in Fig. 1.

Let at time slot \( t \) sensed data is stored in CH in vector \( Z_t = [Z_{1,t}, Z_{2,t}, \ldots, Z_{m,t}] \), where \( m \) is member nodes of the cluster head. As sensor nodes are faulty, this \( Z_t \) is composed of both faulty and normal reading. CH detects these faults and eliminates them. In this work, CH computes Pearson’s correlation coefficient\((\rho)\) between \( Z_t \) and \( Z_{t-1} \) i.e., \( \rho(Z_{t-1}, Z_t) \). Then, CH classifies \( \rho(Z_{t-1}, Z_t) \) as faulty or normal using Support Vector Machine Classifier.

4 Preliminaries

Spatiotemporal correlation is the nature of various physical phenomena such as temperature, humidity, illumination, and many more. Spatial correlation implies a correlation mapping of measures between two neighboring nodes at time \( t \). As physical phenomena are continuous, there should also be a mapping between current measure \( (t) \) and previous measure \( (t - 1) \). This is called temporal correlation. Several possible correlation measures are there, among which Pearson’s correlation coefficient is most popular.

**Definition 1** The (Pearson) correlation coefficient between two random variable \( X \) and \( Y \) is defined as

\[
\text{corr}[X, Y] = \frac{\text{Cov}[X, Y]}{\sqrt{\text{var}[X], \text{var}[Y]}}
\]

(1)
where \( \text{cov}[X,Y] \) is the covariance of \( X \) and \( Y \). Whereas \( \text{var}[X] \) is the variance of random variable \( X \). The correlation coefficient can be viewed as a degree of linearity between \( X \) and \( Y \). \[20\]

### 4.1 Support Vector Machine

Let a training sample set of length of length \( k \) is given with two separable classes \( P \) and \( N \):

\[
\{(X_k, y_k), k = 1, \ldots, K\}
\]  

where \( y_k \in \{1, -1\} \) labels \( X_k \) to belong to either of the two classes. It is required to find a hyper-plane in terms of weight vector\( (w) \) and bias term\( (b) \), that separates the two classes.

In case of linear classification, SVM classifier computes a decision hyper-plane i.e. \( x^T w + b = 0 \) to separate the two classes, \( P = \{(x_i, 1)\} \) and \( N = \{(x_i, -1)\} \). This implies for both \( x_i \in P \) and \( x_i \in N \) has to satisfy (3).

\[
y_i(x_i^T w + b) \geq 0
\]  

From all possible hyperplanes, the objective is to find out the optimal hyperplane that satisfies the above condition. It is needed need to place the optimal hyperplane in a position such that distance from the hyperplane to the closest point of either side is the same. Now the problem of finding the optimal decision plane in terms of \( w \) and \( b \) can be formulated as:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} w^T w = \frac{1}{2} ||w||^2 \\
\text{subject to} & \quad y_i(x_i^T w + b) \geq 1
\end{align*}
\]  

Its solution gives us the optimal margin classifier. The solution can be obtained using the Lagrange multipliers method, as shown in (4).

\[
w = \sum_{i=1}^{m} \alpha_i y_i x_i = \sum_{i\in sv} \alpha_i y_i x_i
\]  

where \( \alpha_i \) is Lagrange multiplier and \( x_i \) is support vector.

**Definition 2** The vectors \( x_i \) residing on either side of separation hyperplane for which \( y_i(w^T x_i + b) = 1 \) holds are called support vector\( (sv) \). SVM only depends on the support vectors. Other sample vectors are not important. \[2\]

Substituting \( w \) from (3) we get,

\[
y_i(\sum_{j\in sv} \alpha_j y_j x_j^T x_i + b) = 1
\]

For the optimal weight vector \( w \) and optimal bias \( b \), we have:
4.1.1 SVM for nonlinear classification/Kernel Mapping

The method mentioned above is called linear SVM, which converges in case of linearly separable data. However, using kernel trick, SVM also works with nonlinear data set. With the help of Kernel trick, a sample $x$ is mapped into a higher dimensional feature space where sample $x$ is linearly separable as:

$$x \rightarrow \phi(x)$$

Moreover, the decision function can be rewritten for the new space as:

$$f(x) = \phi(x)^T w + b = \sum_{j=1}^{m} \alpha_j y_j \phi(x)^T \phi(x_j) + b \tag{5}$$

where

$$w = \sum_{j=1}^{m} \alpha_j y_j \phi(x_j)$$

and $b$ are the parameters of the decision plane in the new space.

Furthermore, the classification function in new space becomes:

$$y_j (\phi(x)^T w + b) \geq 0 \tag{6}$$

From (5) and (6), we can see that vector $x_j$ appears only in inner products of both decision function and learning law. This implies that it is not needed to explicitly specify mapping function $\phi(X)$. Just need the inner product of the vectors in new space $\phi$.

**Definition 3** In Machine Learning, a kernel refers to kernel trick, which is used to solve a non-linear problem using a linear classifier. A kernel function takes $x_i$ and $x_j$ vectors as arguments and returns the inner product of their images $\phi(x_i)$ and $\phi(x_j)$: [9]

$$K(x_1, x_2) = \phi(x_1)^T \phi(x_2)$$

A kernel function only returns the inner product of two vectors. So the dimension of kernel space is not so important. Kernel $K(x_1, x_2)$ needs to be positive semidefinite to fulfill the criteria of Reproducing Kernel Hilbert Space(RKHS) where optimization problem has a finite-dimensional solution that converges to an optimal one.

Now by replacing $x^T$ with $\phi(x)^T$ of (3), the new separation hyperplane in kernel space is obtained as shown in (7). We used this equation for classification of non linear problem.
The bias term \( b \) can be computed from any of the support vectors \( x_i \) as shown in (8).

\[
y_i(\phi(x)^T w + b) = y_i(\sum_{j=1}^{m} \alpha_j y_j K(x, x_j) + b) \geq 0
\]  

The bias term \( b \) can be computed from any of the support vectors \( x_i \) as shown in (8).

\[
b = y_i - \phi(x_i)^T w = y_i - \sum_{j=1}^{m} \alpha_j y_j \phi(x_i)^T \phi(x_j)
\]

\[
y_i - \sum_{j=1}^{m} \alpha_j y_j K(x_i, x_j)
\]

\[
y_i(\phi(x)^T w + b) = y_i(\sum_{j=1}^{m} \alpha_j y_j K(x, x_j) + b) \geq 0
\]

\[
b = y_i - \phi(x_i)^T w = y_i - \sum_{j=1}^{m} \alpha_j y_j \phi(x_i)^T \phi(x_j)
\]

\[
y_i - \sum_{j=1}^{m} \alpha_j y_j K(x_i, x_j)
\]

5 Proposed Scheme

In this work, Pearson’s correlation coefficient(\( \rho \)) and Support Vector Machine(SVM) was applied to determine faults. Pearson’s \( \rho \) is suitable for fault detection because of some of its intrinsic properties such as \( \rho \) is invariant to scaling i.e. \( \rho(x, y) = \rho(x, ax + b) \), where \( a \) and \( b \) are constants. This implies normal reading, as well as event reading, will show a high correlation, whereas the presence of faulty reading will show a lesser correlation [7]. SVM is capable of classifying higher dimensional data, and using kernel trick SVM also can classify nonlinear data. As our data set in nonlinear and has two features, SVM classifier was used in this work. Figure 2 shows the learning process(training and Classification).

5.1 Training Phase

Support Vector Machine(SVM) was used to classify between faulty reading and normal reading. SVM is trained at Base Station with labeled data \((Z_{i,t}, y)\), where \( y \in \{-1, 1\} \). Here label \( y = 1 \) represents not faulty and \( y = -1 \) represents faulty class.

Our labeled data \( Z_{i,t} \) is composed of humidity and temperature. At first, \( Z_{i,t} \) is separated in two vectors \( HZ_{i,t} \) and \( TZ_{i,t} \) for humidity and temperature respectively. Then two new vector is constructed as \( HZ_t = (HZ_{1,t}, HZ_{2,t}, \ldots, HZ_{m,t}) \) and \( TZ_t = (TZ_{1,t}, TZ_{2,t}, \ldots, TZ_{m,t}) \). These are done in data preparation phase.

Then Pearson’s correlation coefficient(\( \rho \)) is computed between two time slot \( t-1 \) and \( t \) for all \( HZ_t \) and \( TZ_t \) as \( \rho_H \) and \( \rho_T \) respectively. After that vector \( \rho \) is set as \( \rho = [ \rho_H, \rho_T ] \). Then SVM is trained with \( (\rho, y) \) using procedure given in Section (4.1). Then \( w \) and \( b \) of classification hyperplane is send to all cluster head nodes.

![Training Phase](image)

Fig. 2 Learning Process
In this method, Gaussian kernel function (\(K\)) was used as (10).

\[
K(x, x_j) = \exp\left(-\frac{||x-x_j||^2}{2\sigma^2}\right) \tag{10}
\]

where \(x\) and \(x_j\) are two vectors and \(\sigma\) is a free parameter.

### 5.2 Classification Phase

At each time slot \(t\), every sensor node collects environmental phenomena (e.g., humidity and temperature) and sends them to its associated cluster-head. Cluster head (CH) collects the reading of the neighboring node. CH computes the correlation between time-slot \(t\) and \(t-1\) of each feature of all the neighboring nodes.

At each time slot, Cluster Head (CH) stores measurements of its member nodes in a vector \(Z_t\). Then CH computes correlation coefficient \(\rho\) between \(Z_t\) and \(Z_{t-1}\) using (1). After that, decision function (5) is applied to this \(\rho\). If \(\rho\) belongs to positive class then \(Z_t\) is not faulty otherwise \(Z_t\) is faulty.

The fault detection algorithm was presented in Algorithm 1. As described in the algorithm, like any other supervised learning, this method is also divided into two phases. First, the training phase, which is performed at BS using training data. The training phase produces a decision function in terms of \(w\) and \(b\). Second, Classification phase, which is performed at each CH using the decision function.

### Algorithm 1: Training and classification.

```plaintext
input : Labeled data \(Z_{i,t}\)
output: \(w\) and \(b\)

Function TrainingPhase() is
  /* On Base-station. */
  Compute \(\rho\) between \(Z_t\) and \(Z_{t-1}\) using (1);
  Compute \(w\) and \(b\) using (5) and (8);
  Send \(w\) and \(b\) to all CH node;
end

input : \(w\) and \(b\)
output: Fault/no-Fault

Function ClassificationPhase() is
  /* On each CH. */
  foreach Timeslot \(T(1,2,3,\ldots)\) do
    foreach Member Node \(m(1,2,3,\ldots)\) do
      Collect data from each member node;
      \(Z_t \leftarrow [Z_{1,t}, Z_{2,t}, \ldots, Z_{m,t}]\);
    end
    Compute \(\rho\) between \(Z_t\) and \(Z_{t-1}\) using (1);
    Classify \(\rho\) using (7);
  end
```

### 6 Simulation Scenario and Performance Evaluation

To evaluate the proposed method, NS2.35 [3] and Matlab is used. NS2 is used to simulate the network scenario and generate measurement data. Then, this generated data is analyzed using Matlab. Performance evaluation is also done using Matlab.
The simulation data is generated using NS2.35. Here Gaussian distribution (N) with mean (μ) and variance (σ) is used of normal and event reading of the dataset [12]. This dataset contains only measurements of 4 sensor nodes. However, actual WSN consists of hundreds of sensor nodes. In the simulation for simplicity, only one cluster head with 100 member nodes is simulated. Results for actual dataset by computing temporal correlation also shown in Section 6.4.

In the network scenario, 100 sensors are randomly deployed in a 300m × 300m with CH at the center. The transmission range of each node is 60m.

Normal sensor readings for temperature are drawn from $N(\mu_{1t}, \sigma_{1t}^2)$ and event readings from $N(\mu_{2t}, \sigma_{2t}^2)$. Where $\mu_{1t} = 28.1273$, $\mu_{2t} = 29.3112$ and $\sigma_{1t} = 1.0952$, $\sigma_{2t} = 4.5588$.

Normal sensor readings for humidity are drawn from $N(\mu_{1h}, \sigma_{1h}^2)$ and event readings from $N(\mu_{2h}, \sigma_{2h}^2)$. Where $\mu_{1h} = 59.6504$, $\mu_{2h} = 78.4943$ and $\sigma_{1h} = 9.7391$, $\sigma_{2h} = 11.3831$.

All the faulty readings for temperature is also drawn from $N(\mu_{2t}, \sigma_{2t}^2)$. Where $\mu_{2t} = 29.3112$ and $\sigma_{2t} = 4.5588$. Moreover all the faulty readings for humidity is also drawn from $N(\mu_{2h}, \sigma_{2h}^2)$. Where $\mu_{2h} = 78.4943$ and $\sigma_{2h} = 11.3831$.

In the simulation, for analyzing the proposed method, the first 20% data are drawn from normal reading, and the last 20% data are drawn from event reading. Rest is a mix of normal and fault reading. Fault readings are mixed according to fault percentage. For example, in the case of 10% fault, 10% data is faulty, and the rest 90% data is normal. We run our simulation for 4690 time slots for 100 nodes for fault percentages 50%, 40%, 30%, 20%, 10%, and 5%. So for each fault percentage length of total data is 469000, 60% of which is used for training purpose and the rest 40% is used for testing. Simulation parameters are given in Table 1.

| Table 1 | Simulation Parameters |
|---------|-----------------------|
| Parameters | Values |
| Network size | 300 m × 300 m |
| Number of nodes | 100 |
| Mac protocol | 802.15.4 |
| Routing Protocol | AODV |
| Transport Protocol | UDP |
| Node’s transmission range | 60 m |
| Cluster Head location | (150,150)m |
| Simulation time | 06h:30m:50s |
| Time slot width | 5 sec |
| Total time slot | 4690 |
| Normal sensor reading (Temperature) | $N(28.1273, 1.0952)$ |
| Normal sensor reading (Humidity) | $N(59.6504, 9.7391)$ |
| Event sensor reading (Temperature) | $N(29.3112, 4.5588)$ |
| Event sensor reading (Humidity) | $N(78.4943, 11.3831)$ |
| Faulty sensor reading (Temperature) | $N(29.3112, 4.5588)$ |
| Faulty sensor reading (Humidity) | $N(78.4943, 11.3831)$ |
| Percentage of measurement faulty nodes | 50%, 40%, 30%, 20%, 10%, and 5% |
6.1 Computational Complexity Analysis

In this work, Pearson’s correlation coefficient and SVM are used. The computational complexity of SVM training and testing is $O(n^3)$ and $O(n)$ [18]. The complexity of Pearson’s correlation coefficient is $O(n)$. So in this work, computational complexity for training phase is $O(n^3)$, and for testing phase, it is $O(n)$.

6.2 Performance Evaluation

6.2.1 Confusion Matrix

In the case of binary classification, only four possible outcomes may occur. They are, Positive samples tested as positive (TP), Negative sample tested as negative (TN), Positive sample tested as negative (FN) and, Negative sample tested as positive (FP).

In binary classification, data samples labeled as positive or negative can be predicted as positive or negative or vice versa. So, only four possible Test outcomes can occur, and they can be represented in the form of a confusion matrix [16], as shown in Table 2.

The proposed method is evaluated using performance matrices namely accuracy, precision, recall, $F_1$ Score, Gmean, sensitivity, and specificity [6]. These matrices are defined as follows.

**Accuracy Measure:** “Accuracy is ratio of observations that are correctly predicted to total observations.”

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$  \hspace{1cm} (11)

**Precision:** “Precision is ratio of observations that are true positive to sum of true positives and false positives.”

$$\text{Precision} = \frac{TP}{TP + FP}$$  \hspace{1cm} (12)

**Recall:** “Recall is the ratio of observations predicted as positive to observations that are actually positive.”

$$\text{Recall} = \frac{TP}{TP + FN}$$  \hspace{1cm} (13)

**$F_1$ Score:** “$F_1$ Score is a weighted harmonic mean of Precision and Recall.”

| Table 2 Confusion matrix |
|-------------------------|
|                         |
|                         |
|                         |
| Actual                  |
|                         |
| True                    |
| False Positive (FP)     |
| False Negatives (FN)    |
| True Positive (TP)      |
| False Negative (TN)     |
|                         |

& Springer
A Method for Fault Detection in Wireless Sensor Network Based…

Gmean: “Geometric mean (Gmean) is the square root of true positive and true negative.”

\[ Gmean = \sqrt{TPR \times TNR} = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{FP + TN}} \]  

(15)

Sensitivity: “Sensitivity is the ratio between all the positive samples to true positive (TP) samples.”

\[ Sensitivity = \frac{TP}{TP + FN} \]  

(16)

Specificity: “Specificity is the ratio between actual negative samples to true negative (TN) samples.”

\[ Specificity = \frac{TN}{FP + TN} \]  

(17)

Accuracy, Precision, Sensitivity, Specificity, Recall, \( F_1 \) Score, G_mean for faults 50%, 40%, 30%, 20%, 10%, and 5% is shown in Table 3. From the results presented in Table 3, it can be seen clearly that with increasing fault percentages, performances increases. This is because, with increasing fault percentages, the correlation between faulty and normal data decreases. However, the normal reading correlation remains high, making classification hyperplane more and more precise.

ROC Analysis: Receiver Operating Characteristics (ROC) analysis studies the sensitivity and the Specificity of the classifier. A ROC curve is a plot in which x-axis is the Specificity, and y-axis is the sensitivity of the classifier.

Area Under curve: The total area under the ROC curve is abbreviated as AUC.

ROC curves are generally used to evaluate the performance of various machine learning algorithms as it gives a comprehensive and visual method of summing the accuracy of an algorithm [6]. Therefore, in this paper, ROC and AUC analysis is used as one of the performance metrics to evaluate our proposed method. From the ROC curve, AUC is calculated by calculating the size of the area under curve. The higher the area, the Better the performance of the method. Figure 3 shows ROC for various fault percentages. AUC is given in Table 3.

| Table 3 Performance evaluation | Metrics       | Fault percentage |
|-------------------------------|---------------|-----------------|
|                               | 50% | 40% | 30% | 20% | 10% | 5% |
| Accuracy                      | 0.9981 | 0.9968 | 0.9927 | 0.9833 | 0.9377 | 0.8096 |
| Precision                     | 0.9991 | 0.9982 | 0.9951 | 0.9881 | 0.9509 | 0.8069 |
| Sensitivity                   | 0.9978 | 0.9965 | 0.9928 | 0.9844 | 0.9460 | 0.8071 |
| Specificity                   | 0.9986 | 0.9971 | 0.9925 | 0.9813 | 0.9241 | 0.7953 |
| Recall                        | 0.9978 | 0.9965 | 0.9928 | 0.9844 | 0.9460 | 0.8071 |
| \( F_1 \) Score               | 0.9984 | 0.9973 | 0.9940 | 0.9863 | 0.9484 | 0.8053 |
| G_mean                        | 0.9982 | 0.9968 | 0.9927 | 0.9829 | 0.9349 | 0.7975 |
| AUC                            | 0.9996 | 0.9996 | 0.9993 | 0.9979 | 0.9823 | 0.8575 |
Comparison with existing work

Simulation results is also compared with the existing works of Zidi et al. [21] and Jan et al. [5]. Figure 4a shows comparative results of Detection Accuracy for our method and

Table 4 Detection accuracy

| Metrics  | Fault percentage |
|----------|------------------|
|          | 50%   | 40%   | 30%   | 20%   | 10%   |
| Proposed | 0.9377 | 0.9833 | 0.9927 | 0.9968 | 0.9981 |
| FDNN     | 0.8678 | 0.8816 | 0.8959 | 0.9041 | 0.9178 |
| SVM      | 0.8245 | 0.8498 | 0.8825 | 0.8960 | 0.9070 |
| NN       | 0.7807 | 0.7823 | 0.7854 | 0.8101 | 0.8197 |
| Zidi et al. | 0.9990 | 0.9989 | 0.9970 | 0.9965 | 0.9950 |

Fig. 4 Detection Accuracy and False Positive Rate

6.3 Comparison with existing work

Simuation results is also compared with the existing works of Zidi et al. [21] and Jan et al. [5]. Figure 4a shows comparative results of Detection Accuracy for our method and
the method of Zidi et al. and Jan et al. for fault percentages 5% to 50% (already shown in Table 4). Figure 4a shows that our method works better with higher fault percentages. Our approach is based on the correlation coefficient. As we know, if the fault percentages increase, the correlation coefficient among data measurements decreases. For this reason, our method works better in high fault percentages.

Figure 4b shows comparative results of Average False Positive Rate (FPR) for our method and the method of Zidi et al. Figure 4b shows that our method shows 14% improvement in terms of Average False Positive Rate compared to Zidi et al. Correlation among measurements of non-faulty node is high. So, a non-faulty node measurement detected as faulty is less likely to happen. That is why our method has better FPR.

6.4 Performance Evaluation For dataset

The proposed method also evaluated using a real-world environmental dataset of [12]. From this dataset, data of temperature and humidity for the indoor sensor node of the multi-hop scenario with anomalies is taken. In this dataset, the anomaly is achieved on a sensor node by a hot water kettle, which increases the temperature and the humidity simultaneously.

Accuracy, Precision, Sensitivity, Specificity, Recall, $F_1$ Score, G_mean for faults 40%, 30%, and 20% is shown in Table 5.

AUC and ROC are given in Fig. 5.

7 Conclusion

In this work, a fault detection algorithm based on Pearson’s correlation coefficient and Support Vector Machine classification is presented. Simulation results show that the method works better in high fault percentages. This is because, with increasing fault percentage, correlation decreases. The proposed method is also evaluated using a dataset [12]. Simulation results, as well as results from real-world data, show that the proposed method successfully classifies fault and normal sensor reading. A possible future directive could be working with more than one machine learning algorithm to detect faults. Applicability of the machine learning methods over some complex networks and data with more event parameters may also be studied in the future.
Declarations

Conflict of interest The authors declare that they do not have any conflict of interest.

References

1. Chen, P. Y., Yang, S., & McCann, J. A. (2015). Distributed real-time anomaly detection in networked industrial sensing systems. *IEEE Transactions on Industrial Electronics*, 62(6), 3832–3842.
2. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3), 273–297.
3. Fall, K., Varadhan, K., et al. (2005). The ns manual (formerly ns notes and documentation). *The VINT project*, 47, 19–231.
4. Guesmi, H., Ben Salem, S., & Bacha, K. (2015). Smart wireless sensor networks for online faults diagnosis in induction machine. *Computers & Electrical Engineering*, 41, 226–239. https://doi.org/10.1016/j.compeleceng.2014.10.015.
5. Jan, S. U., Lee, Y. D., & Koo, I. S. (2021). A distributed sensor-fault detection and diagnosis framework using machine learning. *Information Sciences*, 547, 777–796. https://doi.org/10.1016/j.ins.2020.08.068.
6. Japkowicz, N., & Shah, M. (2011). *Evaluating learning algorithms: A classification perspective*. Cambridge: Cambridge University Press.
7. Krishnamachari, B., & Iyengar, S. (2004). Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. *IEEE Transactions on Computers*, 53(3), 241–250.
8. Mohammed, T., & Shaikh, R. A. (2017). An analysis of fault detection strategies in wireless sensor networks. *Journal of Network and Computer Applications*, 78, 267–287. https://doi.org/10.1016/j.jnca.2016.10.019.
9. Ng, A. (2000). Cs229 lecture notes. *CS229 Lecture notes, I*(1), 1–3.
10. Priya, K.C.K., & Terence, S. (2013). Retp: Reliable event transmission protocol in a wireless sensor network. In *2013 IEEE International Conference ON Emerging Trends in Computing, Communication and Nanotechnology (ICECCN)* (pp. 181–188), https://doi.org/10.1109/ICE-CCN.2013.6528489
11. Saedie Emadi, H., & Mazinani, S. M. (2018). A novel anomaly detection algorithm using dbscan and svm in wireless sensor networks. *Wireless Personal Communications*, 98(2), 2025–2035. https://doi.org/10.1007/s11277-017-4961-1
12. Suthaharan, S., Alzahrani, M., Rajasegarar, S., Leckie, C., & Palaniswami, M. (2010). Labelled data collection for anomaly detection in wireless sensor networks. In *2010 Sixth International
A Method for Fault Detection in Wireless Sensor Network Based...

13. Swain, R. R., & Khilar, P. M. (2017). Composite fault diagnosis in wireless sensor networks using neural networks. *Wireless Personal Communications, 95*(3), 2507–2548. https://doi.org/10.1007/s11277-016-3931-3

14. Swain, R. R., Khilar, P. M., & Bhoi, S. K. (2018). Heterogeneous fault diagnosis for wireless sensor networks. *Ad Hoc Networks, 69*, 15–37. https://doi.org/10.1016/j.adhoc.2017.10.012

15. Swain, R. R., Dash, T., & Khilar, P. M. (2019). A complete diagnosis of faulty sensor modules in a wireless sensor network. *Ad Hoc Networks, 93*, 101924. https://doi.org/10.1016/j.adhoc.2019.101924

16. Ting, K.M. (2010). Confusion Matrix, Springer US, Boston, MA, pp 209–209. https://doi.org/10.1007/978-0-387-30164-8_157

17. Titouna, C., Naït-Abdesselam, F., & Khokhar, A. (2019). Dods: A distributed outlier detection scheme for wireless sensor networks. *Computer Networks, 161*, 93–101. https://doi.org/10.1016/j.comnet.2019.06.014

18. Tsang, I. W., Kwok, J. T., & Cheung, P. M. (2005). Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research, 6*(13), 363–392. http://jmlr.org/papers/v6/tsang05a.html

19. Ul Islam, R., Hossain, M. S., & Andersson, K. (2018). A novel anomaly detection algorithm for sensor data under uncertainty. *Soft Computing, 22*(5), 1623–1639. https://doi.org/10.1007/s00500-016-2425-2

20. Wright, S. (1921). Correlation and causation. *Journal of Agricultural Research, 20*, 557–580.

21. Zidi, S., Moulahi, T., & Alaya, B. (2018). Fault detection in wireless sensor networks through svm classifier. *IEEE Sensors Journal, 18*(1), 340–347. https://doi.org/10.1109/JSEN.2017.2771226

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Priyajit Biswas has done his B.Tech. Degree in Information Technology from Haldia Institute of Technology, West Bengal. And, M.Tech in Information Technology from Indian Institute of Engineering Science And Technology, Shibpur. Currently, he is pursuing PhD in Information Technology from Indian Institute of Engineering Science And Technology, Shibpur. His research interests are Wireless Sensor Networks, Machine Learning, and Information Security.
Tuhina Samanta is presently an assistant professor in Indian Institute of Engineering Science and Technology, Shibpur, formerly known as Bengal Engineering and Science University, Shibpur. She received her B.Sc. degree in Physics honours from University of Calcutta in the year 2000. She completed her B.Tech. and M.Tech. from the Institute of Radiophysics and Electronics, University of Calcutta in 2003 and 2005 respectively. She was awarded Canada Research Scholarship during her M.Tech. She received her doctoral degree from Bengal Engineering and Science University, Shibpur, in 2010. She was awarded Raman Post Doctoral fellowship to pursue her research work for one year at University of Central Florida, Florida in the year 2014–15. Her major research areas are study and design of algorithms for performance analysis of sub-micron VLSI systems, physical design for Digital Microfluidic Biochip and CAD, and development of algorithm in energy efficient wireless sensor network. She received couple of best paper awards in IEEE International Conferences. She performed as a program committee member of several IEEE/ACM conferences.