Anomaly Detection in Wireless Sensor Networks using Self-Organizing Map and Wavelets

SUPAKIT SIRIPANADORN¹, WIPAWEE HATTAGAM², NEUNG TEAUMROONG³
¹,²School of Telecommunication Engineering, Institute of Engineering
³School of Biotechnology, Institute of Agriculture
Suananaree University of Technology
111 University Avenue, Muang, Nakhon Ratchasima 30000
THAILAND
¹architect_ton@hotmail.com ²wusaha@ieee.org ³neung@sut.ac.th

Abstract: - Wireless Sensor Networks (WSNs) have been applied in agriculture monitoring to monitor and collect various physical attributes within a specific area. It is important to detect data anomalies to determine a suitable course of action. The underlying aim of this paper is therefore to propose an anomaly detection scheme which is able to detect anomalies accurately by means of exploiting both time and frequency characteristics of the data signals. The contribution of this paper centers on anomaly detection by using Discrete Wavelet Transform (DWT) combined with a competitive learning neural network called self-organizing map (SOM) in order to accurately detect abnormal data readings. Experiment results from synthetic and real data collected from a WSN show that the proposed algorithm outperforms the SOM algorithm by up to 18% and DWT algorithm by up to 35% in presence of bursty faults with marginal increase of false alarm rate.

Key-Words: - Anomaly Detection, Wavelets, Discrete Wavelet Transform, Self-Organizing Map, Wireless Sensor Networks, Agriculture Monitoring

1 Introduction

Wireless sensor networks (WSNs) have been recently deployed in many areas of agriculture to increase yield and prevent outbreaks such as in hydroponics and paddy fields, fertilizer composting process, and livestock monitoring. However, these applications rely mainly on manually measuring and controlling the parameters such as moisture, temperature, pH, oxygen, soil nutrients, etc., which is both time consuming and laborious. Autonomous monitoring devices such as WSNs therefore warrant potential use in agriculture monitoring.

A WSN is a wireless network that consists of distributed autonomous devices using sensors to cooperatively monitor or collect environmental conditions at different locations. Several measurements can be collected from the WSN. The collected measurements from the WSN may be affected by anomalies in the sensor network such as faulty sensors, faulty communication between sensors or actual abnormal physical measurements. With the huge amount of data continually collected from the WSN, it becomes increasingly difficult to detect anomalies in the data measurements. Therefore, anomaly detection techniques are necessary to automatically detect faults and alert the system controller to take suitable action.

Research emphasizing on anomaly detection in communication networks has progressed in recent years, e.g. in IP networks [1], in cellular mobile networks [2]. There are also works on fault and anomaly detection in wireless sensor networks (WSNs). Ref.[3] classified anomalies in WSNs into 2 groups, i.e. 1) intrusion or security attacks, and 2) faults from unusual changes in a monitored environment. This paper focuses on the latter since such anomaly detection is more critical to agricultural use. To detect the latter type of anomalies, a dynamic model of WSNs based on recurrent neural networks (RNNs) was presented in [4], whereas a distributed kernel density estimator was proposed in [5] to approximate the density distribution of the data.

Another mechanism commonly used for anomaly detection is a competitive learning method called self-organizing map (SOM) [6], [7]. SOM has several beneficial features which make it a useful tool in data mining. It follows the probability density function of the data and is, thus, an efficient clustering and quantization algorithm. The most important feature of the SOM is the visualization property.

However, SOM has some weaknesses where it extracts relevant features of the data only in the time domain. In many scenarios, features of the data extracted from both time and frequency domain can be used to further enhance anomaly detection [8]. This can be achieved by the Discrete Wavelet Transform (DWT). Wavelets have been extensively employed for anomaly
and fault detection in many applications [9]. DWT has also been integrated with SOM to detect faults [10], [11]. In particular, feature vectors of the faults have been constructed using DWT, sliding window and a statistical analysis. Classification of the feature vectors was obtained by using SOM.

To the best of our knowledge, DWT and SOM have not yet been applied to anomaly detection in WSNs. Therefore, the underlying aim of this paper is to propose an anomaly detection algorithm which determines the wavelet transform, and detects the abnormality of the sensor readings by training the SOM using the wavelet coefficients.

2 Anomaly Detection

The first step of anomaly detection involves selecting the data parameters to be monitored and grouping them together in a pattern vector $x^\mu \in \mathbb{R}^n$, $\mu = 1, \ldots, N$,

$$
x^\mu = \begin{pmatrix}
x_1^\mu \\
x_2^\mu \\
\vdots \\
x_n^\mu
\end{pmatrix} = \begin{pmatrix}
\text{KPI}_1^\mu \\
\text{KPI}_2^\mu \\
\vdots \\
\text{KPI}_n^\mu
\end{pmatrix},
$$

where $\mu$ is the observation index, $n$ is the number of parameter types or key performance indices (KPIs) chosen to monitor the environmental condition.

2.1 Self-Organizing Map

Competitive neural models such as the self-organizing map (SOM) [6], are able to extract statistical regularities from the input data vectors and encode them in the weights without supervision. It maps a high-dimensional data manifold onto a low-dimensional, usually two-dimensional, grid or display.

The basic SOM consists of a regular grid of map units or neurons as shown in Fig 1(a). Each neuron, denoted by $i$ (depicted by the black dot), has a set of layered neighboring neurons (depicted by the white dots) as shown in Fig 1(a).

Neuron $i$ maintains a weight vector $m_i$. In order to follow the properties of the input data, such vector is updated during the training process. For example, Fig 1(b) shows a SOM represented by a 2-dimensional grid of 4×4 neurons. The dimension of each vector is equal to the dimension of the input data. In the figure, a vector of input data (marked by $x$) is used to train the SOM weight vectors (the black dots). The winning neuron (marked by BMU) as well as its 1-neighborhood neurons, adjust their corresponding vectors to the new values (marked by the gray dots).

The SOM is trained iteratively. In each training step, one sample vector $x$ from the input data set is chosen.

![Fig. 1 An illustration of the SOM (a) with rectangular lattice neighbors belonging to the innermost neuron (black dot) corresponding to 1, 2 and 3- neighborhoods, (b) SOM updates the BMU with 1-neighborhood.](image)

The distances between the sample data and all of weight vectors in the SOM are calculated using some distance measure. Suppose that at iteration $t$, neuron $i$ whose weight vector $m_i(t)$ is the closest to the input vector $x(t)$. We denote such weight vector by $m_c(t)$ and refer to it as the Best-Matching Unit (BMU), that is

$$
\|x(t) - m_c(t)\| = \arg \min_{i} \|x(t) - m_i(t)\|
$$

where $\|$ is the Euclidian distance.

Suppose neuron $i$ is to be updated, the SOM updating rule for the weight vector of neuron $i$ is given by

$$
m_i(t + 1) = m_i(t) + \eta_c h_c(i,t)(x(t) - m_c(t))
$$

where $t$ is the iteration index, $x(t)$ is an input vector, $\eta_c$ is the learning rate, $h_c(i,t)$ is the neighborhood function of the algorithm. The Gaussian neighborhood function may be used, that is

$$
h_c(i,t) = \exp \left(-\frac{\|r_c(t) - r_i(t)\|^2}{2\sigma^2(t)}\right)
$$

where $r_c(t)$ and $r_i(t)$ are the positions of neurons $i$ and the BMU $c$ respectively, and $\sigma(t)$ is the radius of the neighborhood function at time $t$. Note that $h_c(i,t)$ defines the width of the neighborhood. It is necessary that $\lim_{t \to \infty} h_c(i,t) = 0$ and $\lim_{t \to \infty} \eta_c = 0$ for the algorithm to converge [6].

2.2 Discrete Wavelet Transform

DWT is a mathematical transform that separates the data signal into fine-scale information known as detail coefficients, and rough-scale information known as approximate coefficients. Its major advantage is the multi-resolution representation and time-frequency localization property for signals. Usually, the sketch of the original time series can be recovered using only the low-pass-cut off decomposition coefficients; the details can be modeled from the middle-level decomposition.
coefficients; the rest is usually regarded as noises or irregularities. The following equations describe the computation of the DWT decomposition process:

\[ a_{j+1}^{DWT}(k) = \sum_n h_0(n-2k)a_j^{DWT}(k) \]  

(5)

\[ d_{j+1}^{DWT}(k) = \sum_n g_0(n-2k)a_j^{DWT}(k) \]

(6)

where the rough-scale (or approximation) coefficients \( a_j^{DWT} \) are convolved separately with \( h_0 \) and \( g_0 \), the wavelet function and scaling function, respectively, \( n \) is the time scaling index, \( k \) is the frequency translation index for wavelet level \( j \). The resulting coefficient is down-sampled by 2. This process splits \( a_j^{DWT} \) roughly in half, partitioning it into a set of fine-scale (or detail) coefficients \( d_{j+1}^{DWT} \) and a coarser set of approximation coefficients \( a_{j+1}^{DWT} \) [12].

DWT has the capability to encode the finer resolution of the original time series with its hierarchical coefficients. Furthermore, DWT can be computed efficiently in linear time, which is important while dealing with large datasets.

**2.3 Integration of SOM and DWT**

In the integration of SOM and DWT algorithm, the DWT algorithm is used as an input data preprocessor of the SOM algorithm in order to reduce the size of data without losing any significant feature of the data. This enables the implementation of in-network processing which helps to reduce the radio communication energy and eventually prolong the lifetime of the WSN [3]. The input data will be padded with zero if its length is odd data. After obtaining the wavelet coefficients, these coefficients will be fed to the SOM algorithm which can be divided into 2 sets. Each set contains both approximate and detail coefficients. The first set which is obtained from noiseless data, will be used to train the SOM algorithm. The second set which is obtained from the faulty data will be used to test the SOM algorithm. Then to reduce the false alarms the detected results will be double checked by using the univariate method [6], [7].

**2.4 Anomaly Detection**

A new observation data set can be considered abnormal if the distance between the weight vector of the winning neuron and the new state vector, given by

\[ e^\mu = \left\| x^new - m^\mu \right\| \]  

(7)

is greater than a certain percentage \( p = 1 - \alpha \) of the distances in the distance distribution profile. That is,

\[
\text{IF } e^\mu \in \left[ e^\mu_{\mu}, e^\mu_p \right],
\]

\[
\text{THEN } x^new \text{ is NORMAL}
\]

\[
\text{ELSE } x^new \text{ is ABNORMAL.} 
\]

(8)

Equation (8) is referred to as the global decision. In [6], an addition of local decisions of each KPIs is presented. Suppose that a data vector \( x^new \) is considered abnormal by the global decision. Then in the local anomaly detection, the absolute value of error in each component of the error vector is then computed by

\[
\left| E^new \right| = \begin{bmatrix}
| x_{\mu_{c,1}}^\mu - m_{\mu_{c,1}} | \\
| x_{\mu_{c,2}}^\mu - m_{\mu_{c,2}} | \\
\vdots \\
| x_{\mu_{c,o}}^\mu - m_{\mu_{c,o}} |
\end{bmatrix}.
\]

(9)

The error in each KPI is then compared to the interval of normality component-by-component, and the anomaly decision is carried out as in (8).
3 Experiment Results

In this section, we evaluated the performance of the proposed integration of SOM and DWT algorithm by detecting anomalies in series of synthetic data and actual data collected from a wireless sensor network.

In the experiment, we generated the synthetic input data from a normal distribution $N(0,1)$ and synthetic faults by additive white Gaussian noise (AWGN) with power 25 dBW generated from MATLAB. We used such fault because its statistical similarity to the synthetic input data thus, it is more difficult to be detected. Therefore, we can evaluate the performance of the algorithms under ambiguous faults. The amount of faults can be represented by the notation $n/s$, where “$n$” is the amount of faults per series and “$s$” is the amount of series of faults, resulting in the total amount of $n \times s$ faults. The generated faults added to the input data ranged from bursty (20/10) to sparse (1/10). The exact positions of the faults incurred in the input data were predetermined and was later used to detect true and false alarms. In the experiment using real data, we have chosen 2 parameters, namely temperature and moisture, as KPIs collected from samples of compost in a bioorganic fertilizer plant. In this paper, the data of the 2 KPIs at the WSNs were collected every 5 minutes for 3 days.

We compared 3 anomaly detection methods:
1. SOM algorithm
2. DWT algorithm
3. Integration of SOM and DWT algorithm

We measured 2 performance metrics: 1) the true alarm rate which is defined by the number of detected true anomalies over the total number of true anomalies in the data set; and 2) the false alarm rate which is defined by the number of detected false alarms over the total number of detected anomalies.

In the DWT algorithm, we used the threshold in (11) in order to decide whether the data is normal or abnormal.

$$\sigma_w = \text{median}(d_i - \bar{d})$$

$$T_w = \sigma_w \sqrt{2 \log_2(N)},$$

(10)

where $N$ is the size of data and $\bar{d}$ is the sample mean of the level 1 detail coefficients [12].

This threshold was calculated from the low pass and high pass coefficients from the assumed normal data by using Haar and Daubechies4 mother wavelets. The Haar and Daubechies4 wavelets were used because they are relatively easy to cross-check by hand with computed coefficients from MATLAB program. Hence, we can compare the position of each coefficient with the actual fault position. After the threshold calculation, the set of coefficients which are obtained from the DWT of the noisy data will be compared with the threshold, coefficient by coefficient. For the real data scenario, the data was normalized by equation (12) before being processed by the DWT to eliminate potential outliers.

$$\text{Norm}(\text{Data}) = \frac{(\text{Data} - \text{mean}(\text{Data}))}{\text{variance}(\text{Data})}$$

(12)

If the absolute value of the coefficient is greater than the computed threshold, an anomaly is said to be detected.

In the SOM algorithm and the proposed integration of SOM and DWT algorithm, the initial value for learning rate in the SOM part was set to $\eta_0 = 0.9$, and gradually reduced to $\eta_T = 10^{-5}$, in order to guarantee convergence [6]. The number of training epochs was set to 50 because longer training epochs tend to over train the SOM [6]. The confidence interval was set to 99% ($K=2.57$). We used a Gaussian neighborhood function because the distribution of the collected data after the normalization fits well to the Gaussian distribution. The 30x30 size of neurons was used. Figures 3 and 4 show that the anomaly detection in SOM algorithm and the integrated SOM and DWT algorithms improve as the number of neurons is increased. This suggests that the more neurons used, the “finer” SOM’s classification becomes resulting in enhanced detection performance. However, at neuron size 50x50, the SOM requires much longer training time with a marginal improvement in the detection performance. Therefore, The 30x30 size of neurons was selected to train and test the SOM. We also improved the SOM algorithm by double checking with the univariate method in order to reduce the false alarm rate [6], [7]. To obtain accurate results, each algorithm was repeated for 70 runs which gave the best accuracy as shown in Table 1.

Table 1. Accuracy results obtained by feeding synthetic input data to the 30x30 neuron SOM algorithm.

<table>
<thead>
<tr>
<th>Runs</th>
<th>True Alarm rate</th>
<th>Deviation from previous iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>62.00</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>59.50</td>
<td>0.040</td>
</tr>
<tr>
<td>20</td>
<td>57.65</td>
<td>0.031</td>
</tr>
<tr>
<td>30</td>
<td>58.17</td>
<td>0.009</td>
</tr>
<tr>
<td>40</td>
<td>57.68</td>
<td>0.008</td>
</tr>
<tr>
<td>50</td>
<td>57.82</td>
<td>0.002</td>
</tr>
<tr>
<td>60</td>
<td>57.88</td>
<td>0.004</td>
</tr>
<tr>
<td>70</td>
<td>58.14</td>
<td>0.001</td>
</tr>
<tr>
<td>80</td>
<td>58.06</td>
<td>0.001</td>
</tr>
<tr>
<td>90</td>
<td>58.17</td>
<td>0.001</td>
</tr>
<tr>
<td>100</td>
<td>58.27</td>
<td>0.001</td>
</tr>
</tbody>
</table>
Fig. 3 True alarm rates with different size of neurons in the sparse faults case.

Fig. 4 True alarm rates with different size of neurons in the bursty faults case.

Fig. 5 True alarm rates with synthetic data.

Fig. 6 True alarm rates with real data.

Fig. 7 False alarm rates with synthetic data.

Fig. 8 False alarm rates with real data.

Fig. 9 True alarm rate with 10 dBW AWGN faults.

Fig. 10 False alarm rate with 10 dBW AWGN faults.
To evaluate the performance of all algorithms, the results of each algorithm were compared to the (known) fault positions which were added into the input data. In particular, when an anomaly was detected then its position was compared with the (known) fault position. If this position existed, then the anomaly detected was a true alarm; otherwise, it was a miss. On the other hand, if an anomaly was detected but the (known) fault position did not exist, then the anomaly was a false alarm.

Fig. 5 and Fig. 6 show the percentage of true alarm rate averaged over 70 runs, as the function of the amount of faults added into the input data. Note that the proposed integrated SOM and DWT algorithm which used Haar as a mother wavelet gives the best performance over other algorithms. This is because the DWT with Haar wavelet can detect changing points. In particular, the Haar wavelet uses 2 adjacent input data to compute a coefficient whereas the Daubechies4 uses 4 adjacent input data to compute a coefficient. However, Daubechies4 gave a lower performance than Haar because each coefficient was computed from an average over 4 input data. If a fault occurred in 1 of these 4 data, such fault will be averaged with the remaining 3 normal data resulting in a coefficient with an absolute value possibly lower than the decision threshold. Consequently, the true alarm rate is reduced. On the other hand, the Haar wavelet only uses 2 adjacent data to compute 1 coefficient. Thus, the true alarm rate is significantly higher than that of Daubechies4. The integrated SOM and DWT algorithm using Haar also outperforms the SOM algorithm. This is because in the Haar case, the coefficients obtained were transformed from two adjacent data. Therefore, if some data was faulty or differed greatly from the data nearby, this coefficient can detect such anomaly. On the other hand, the SOM algorithm directly checked the data one by one to detect an anomaly. If the data was faulty but had a small magnitude, then this fault may not be detected, and consequently the true alarm rate was reduced. Note that the DWT algorithm has the lowest performance because the decision threshold in (11) is rather conservative. Furthermore, the threshold is fixed throughout the detection and the algorithm does not have any double checking method.

Fig.7 and Fig. 8 show the false alarm rate results in the synthetic and real data experiments, respectively. Note that most results have low false alarm rates, i.e., less than 1 % except in the case of sparse faults due to the increased detection difficulty.

The integration of SOM and Daubechies4 DWT also gave a weak performance due to the reasons previously explained. All these results show that the integration of SOM and DWT with Haar as a mother wavelet outperforms the SOM algorithm and DWT method.

Fig.5 and 6 show that the proposed algorithm can achieve up to 65% and 67% of true alarm rates in case of bursty faults for synthetic and real data, respectively. The proposed algorithm achieved a true alarm rate of up to 18% higher than the SOM algorithm alone in presence of bursty faults. Compared to the DWT alone, the proposed algorithm can attain a true alarm rate of up to 35% more in the bursty faults case.

As for sparse faults, the proposed algorithm can achieve up to 69% and 80% true alarm rates for synthetic and real data, respectively. The integrated SOM and DWT also gave true alarm rates of up to 10% higher than the SOM algorithm alone whereas DWT performed the weakest, in presence of sparse faults.

Fig. 7 and 8 show that the false alarm rate of the proposed algorithm is 0.11% and 0.13% in presence of bursty faults and 0.91% and 1% in presence of sparse faults with synthetic and real data, respectively. Note that the false alarm rate of the propose algorithm is slightly higher than the other two algorithms. Since the gain in the true alarm rate is more significant, such tradeoff is therefore considered acceptable.

**DWT to reduce transmitted data:** The proposed integration of SOM and DWT algorithm with Haar wavelet outperformed the SOM algorithm and the DWT algorithm alone. Our results suggest that the proposed integrated SOM and DWT anomaly detection scheme can be deployed in a resource-constrained network such as a WSN. In particular, the DWT using Haar wavelet can be implemented at the sensor nodes as a data preprocessor to reduce the amount of data to be transmitted by at least half (for one-level DWT). Since energy consumption is critical in WSNs, such distributed in-network processing can reduce transmission energy and eventually help prolong the overall network lifetime of the WSN [3] while still maintaining acceptable anomaly detection accuracy.

Fig. 9 and Fig. 10 show the effect of the decreasing of AWGN noise power from 25dBW to 10 dBW in both synthetic and real data scenarios. Though the anomaly detection is more difficult, the proposed integrated SOM and DWT still consistently outperforms the other two methods in terms of true alarm rate but with marginal increase in the false alarm rate as tradeoff.

**4 Conclusion**

This paper proposed an integration of a competitive learning method called the self-organizing map (SOM) and the discrete wavelet transform (DWT), to detect anomalies in synthetic and actual data collected from a WSN.
The results show that the integration of SOM and DWT with Haar as a mother wavelet can attain 65% and 67% of true alarm rates in the case of bursty faults, and 69% and 80% of true alarm rates in case of sparse faults for synthetic and real data, respectively. In terms of the true alarm rate, the proposed algorithm outperforms the SOM algorithm by up to 18% and DWT algorithm by up to 35% in presence of bursty faults. With sparse faults, the proposed algorithm can gain a true alarm rate up to 10% above the SOM algorithm alone and entirely outperforms the DWT algorithm alone. Such gain in true alarm rates come with a marginal increase of false alarm rate. These results show that the integration of SOM and DWT with Haar wavelet can lead to a more effective anomaly detection which reduces human operator’s troubleshooting efforts.

In the future, we plan to extend our work to investigate anomaly detection with actual faults obtained from the bioorganic fertilizer plant environment, and study its performance by increasing the DWT level and considering other different types of wavelets. Furthermore, we also plan to investigate ways to identify and eliminate erroneous sensor readings at the sensor nodes, which could help reduce wasted energy from transmitting unwanted erroneous measurements to the base station.

5 Acknowledgment
The authors would like to thank Assoc. Prof. Dr. Boon Hee Soong and Mr. Edwin Chan of Intelligent Systems Laboratory, Nanyang Technological University, Singapore for their assistance and suggestions on the sensor development and implementation.

References: