A Competitive Learning Algorithm for Checking Sensor Data Integrity in Unknown Environments

Tatiana Bokareva*
tbokareva@cse.unsw.edu.au
The University of NSW

Nirupama Bulusu
nbulusu@cs.pdx.edu
Portland State University

Sanjay Jha
sjha@cse.unsw.edu.au
The University of NSW

Technical Report: UNSW-CSE-TR-0516

October 18, 2005

*Part of this work was done while Tatiana Bokareva was visiting Portlan State University.

Abstract

Ad-Hoc wireless sensor networks derive much of their promise from their potential for autonomously monitoring remote or physically inaccessible locations. As we begin to deploy sensor networks in real world applications [17], ensuring the integrity of sensor data is of paramount importance.

In this paper, we motivate, propose, evaluate and analyze an online algorithm for modeling and validating sensor data in an unknown physical environment. Previous work on checking sensor data integrity developed within the context of process control systems uses a priori characterization of sensor data. In contrast, our approach leverages the concept of *competitive learning* for online characterization of a dynamic, unknown environment and the derivation of conditions for verifying sensor data integrity over time. Moreover, to scale to very large sensor networks, our algorithm leverages in-network processing a hierarchical, tiered sensor network by executing on the distributed cluster heads, rather than at a central base station.

We prove the convergence properties of our algorithm through theoretical analysis. Furthermore, we implement our algorithm on a real physical sensor network of motes and Stargates, and demonstrate that our algorithm successfully learns real-world environmental data characteristics and filters anomalous data in a sensor network.

1 Introduction

For widespread adoption of sensor technology, robust and high-integrity operation of the sensor network is of paramount importance. In previous work, we have made the case for a self-healing sensor network architecture called SASHA[2], that is inspired by the natural immune system. The goal of SASHA is to eventually provide automatic fault recognition and response over a wide variety of faults, and evolve its monitoring and inference capabilities with time.

A fundamental challenge in ensuring high-integrity operation of the sensor network as envisioned by SASHA is ensuring the sensor data integrity. This paper proposes an online algorithm for mapping an environment and detecting anomalies in the data gathered using a sensor network. This is important functionality for a sensor network because sensor data are prone to numerous faults, such as bio-fouling (e.g., leaves falling on sensors), adverse ambient (e.g., snow covering sensors), failures due to adverse ambient (e.g., heat,) sensor displacement due to the environment (e.g., due to wind), battery failures in event-oriented systems, and rarer failures due to transducer calibration.

There has been much work in calibrating and detecting individual sensor faults in industrial process control systems, such as [11], [18]. However, with the exception of [12], the problem of verifying the correctness of sensor data in the presence of faults and disruption of the sensing channel, which we refer to as *data integrity verification* in a sensor network has been largely unexplored.

This is a difficult problem due to several reasons. First, sensor networks are often ad hoc deployed in unknown environments with limited knowledge of the phenomena being observed. In this environment, distinguishing normal and anomalous behavior of sensor readings is difficult because of a lack of knowledge of normal data behavior. Second, individual sensor devices have very limited computational capacity which bounds the amount of computation that can be performed on these devices. The approach for checking sensor data integrity must scale to large networks, minimize the energy overhead required to communicate sensor data, and adapt to system and environmental dynamics, such as node failures and the natural changing environment. To address the first problem, we propose to leverage competitive learning neural networks (CLNN) for verifying sensor data integrity. To address the second problem, we leverage in-network processing in a hierarchical sensor network, where the data integrity verification is performed at cluster-heads or micro-servers (also known as monitoring nodes in SASHA) as shown in Figure 1, rather than a centralized base station.

We formulate the data integrity verification problem as follows:
Given a set of input sensor data X, CLNN must

- *Learn* the characteristics and frequency distribution of the environmental data.
- *Deduce* what input data are correct, and what may be anomalous based on the learned environment model.

Note that our objective is not to map the spatio-temporal characteristics of the environmental phenomenon, only to determine the operational range of sensor values.

CLNN uses an *unsupervised learning* procedure, which we define formally in Section 3, to group the input data into clusters. The strength of this approach is that it does not require us to collect a priori training data for the sensor network, making it suitable for large scale, remotely deployed sensor networks. Furthermore, the frequency and length of the training period can be adjusted based on robustness or performance requirements of the application. Finally, clustering techniques can reduce the amount of information need to be transmitted over the radio which has been identified as a major source of the energy consumption. However, the reduction in the information space comes at the price of the higher preprocessing of data. Another drawback of this approach

is that it may require a large amount of training time depending on the characteristics of the phenomena being observed. Also, if more than 50% of the nodes are faulty, we may not be able to deduce the correct sensor readings with a high degree of confidence.

1.1 Paper Contributions

This paper motivates and proposes a novel framework for data integrity verification. We leverage and motivate the use of *data clustering* as a building block for sensor data integrity verification, assigning an additional probability to each cluster. This approach filters the high confidence data from the low-confidence data, while preserving the fine-grained features within the data.

We have evaluated our approach on a physical sensor readings and empirically measured its accuracy and performance on a continuous sampling application. Our evaluation demonstrates that CLNN is computationally efficient and identifies the correct sensor readings even in the presence of environmental dynamics (such as a changing phenomenon) and system dynamics (such as faulty sensor nodes).

In the rest of this paper, we review related work in Section 2, we describe and formally define CLNN in Section 3. In Section 4, we provide a mathematical background of competitive learning and its complexity analysis. In Section 5, we describe the experimental evaluation of CLNN. Finally, we conclude in Section 6.

2 Related Work

Once deployed in the physical world, sensors are prone to numerous faults which corrupt sensor data. This motivates techniques for automatic sensor fault detection and calibration, which can identify these faults and correct them early.

The noise level of a physical sensor device is typically determined in the factory through calibration. However, even factory-calibrated sensors are prone to numerous faults which corrupt sensor data. For example, sensors deployed in the Columbia River Estuary gather information on physical dynamics and changes in estuary habitat [3]. Salinity sensors are particularly susceptible to bio-fouling, which gradually degrades sensor response and corrupts critical data. The authors propose a framework for detecting degradation that grows with time, based on the standard sequential likelihood method from classical pattern recognition. The limitation of the proposed approach in [3] is that it is applicable to a single sensor system and the training of sensors happens offline.

One of the the most interesting methodology for sensor fault-tolerance was developed by Marzullo [11]. This methodology was originally proposed for process control systems, but it can be applied broadly to wireless sensor networks. This work uses the notion of abstract sensor, which is a piecewise continuous function from a physical state variable to a dense interval of real numbers. An abstract sensor is correct if it is not too inaccurate, it always includes the value of the physical variable. How do we construct an abstract sensor that is tolerant of failures? The solution lies in replication. Suppose we are given n independent abstract sensors and an assumption that no more than f sensors can fail. Marzullo intuit that intervals containing the correct value must intersect, any point not contained in at least $(n - f)$ intervals must not be correct.

Marzullo's work shows that local sensing can be bound through sensor replication, even in the presence of sensor failures. However, his methodology does not give an indication of the confidence level of the fault-tolerant abstract sensor and it is largely centralized approach. It may also require a large amount of storage

for sensor readings and a large amount of computational power as the number of sensors and the amount of collected data becomes large.

Krishnamachari and Iyengar [12] proposed a solution to the recognition of faulty sensor readings and introduced self-organization algorithms which combine a shortest-path routing and the construction of a spanning tree as a clustering mechanism for nodes in a feature region.

To minimize the amount of forwarding data several aggregation options were presented and the total cost of each scheme in terms of the total number of transmitted bits were studied.

One important aspect in the feature recognition is the understanding of what represents a feature. In this work, the authors assumed that a simple threshold value is sufficient to determine the presence of an event. In other words a large value is considered to be an unusual reading, where the low value is assumed to be a normal sensor reading. The value of the threshold can be specified by a query or directly preloaded into nodes before deployment. The problem is reduced to the mapping of a sensor reading to a binary value S_i. S_i=0 if a sensor measures an usual value and S_i=1 if a sensor indicates an unusual reading. One of the examples given for the use of this algorithm is the monitoring of a chemical concentration, where a region of an unusually high concentration is of interest. It is also argued that the modularity in sensor readings can be introduced by the lowest/highest chemical concentration region.

Although it is a reasonable assumption for some sensor network applications, nevertheless it is not suitable for most of them. In many sensor network applications, events cannot be modeled by a simple threshold value. In contrast, our approach dynamically tries to learn what the operational range of the data is.

There is also a large body of related work that ensure data integrity from the security perspective in wireless sensor networks, such as "SIA:Secure Iformation Aggregation"[19], or to ensure the data is correctly aggregated, without counting duplicates, such as Synopsis Diffusion [16]. Their focus is in ensuring the integrity of the data collection process, rather than the data itself.

Our proposal could be extended to work over aggregated sensor data rather the raw sensor readings, however this is not the focus of our design and evaluation. Rather, our research is motivated by a question likely to be increasingly important as we begin to deploy sensor networks — Can a sensor network learn what the properties of its measured environmental sensor data are, and use this to check what input data is likely to be correct and what is anomalous? In the next two sections, we describe our approach toward addressing this question, and its evaluation.

3 Approach

Sensor networks are expected to be ad hoc deployed to study an unknown physical phenomenon, sometimes in remote or hostile environments. In this case, verification of the quality of collected sensor data becomes a challenging problem. This motivates autonomous verification of collected data without human intervention. Moreover, the environment may be dynamic and vary with time. Therefore, our primary design goal was to come up with a solution that *did not require a priori characterization of the sensor data*.

Network Model: In recent years, the research community has recognized the need for a hierarchical organization of the sensor network [9]. As it was mentioned earlier and very well documented, the currently available sensors such as the mica mote family [10] have very limited computational capacity. Moreover, studies show that the energy expended for performing a computational task on these processors is surprisingly much greater than a PDA sized device[14]. Therefore, they are reserved for simple data collection and most of the demand-

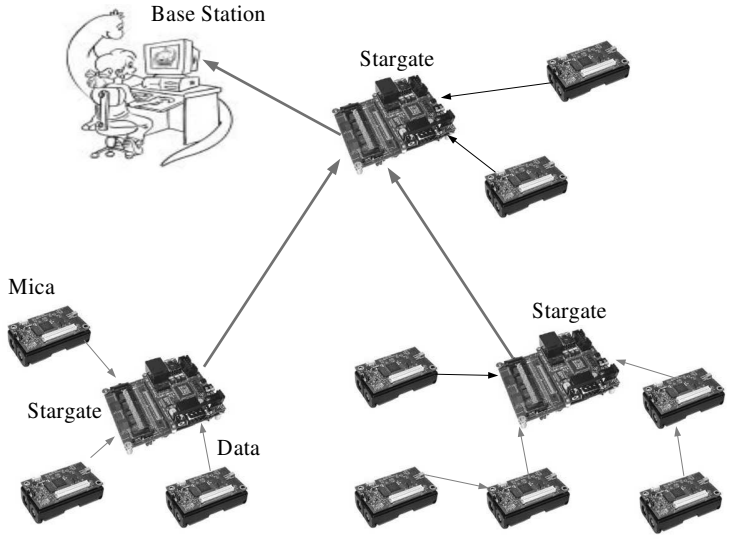

Figure 1: An example of a hybrid sensor network.

ing computational tasks are shifted to devices such as Stargate or Infrinsys Cube. Figure 1 shows the generic architecture of a hybrid sensor network where each node communicates its sensor data to a closest cluster head. The cluster head classifies the input data and performs most of the signal processing before forwarding the data to a base station.

In the rest of this section, we describe our approach for verifying sensor data integrity. The algorithm can be broken into two phases.

- *Phase 1: Learning*
  In first phase it will learn physical attributes of a measured real world phenomenon. We acquire such knowledge by means of the Competitive Learning Neural Network (CLNN) [13], which divides the input data into clusters.
- *Phase 2: Deduction*
  In the second phase, based on the acquired knowledge the algorithm will filter out noisy or faulty sensor readings. We filter out noisy measurements by assigns the level of confidence to each data cluster.

3.1 Competitive Learning Neural Network (CLNN)

Multi-Layer Feed-Forward Neural Networks (FFNN) have been traditionally used to identify faulty sensor readings in process control systems [18], [5] and [6]. And it was shown that neural networks in general are good candidates for such tasks, especially when the sensor data contains a large amount of noise. However, Multi-Layer Feed-Forward Neural Networks with Back Propagation requires a set of input vectors $X = [x_1, ..x_n]$ and a set of corresponding output vectors $O = [o_1, ..o_n]$ for its training.

The major limitation of such an approach is that the corresponding output set O may not be available, unless some a priori measurements of exactly the same environment were taken. This raises the question — to what degree can we trust a priori measurements? Real world phenomenon are rarely stable which makes the training of FFNNs a challenging proposition for wireless sensor networks. We require what is known as

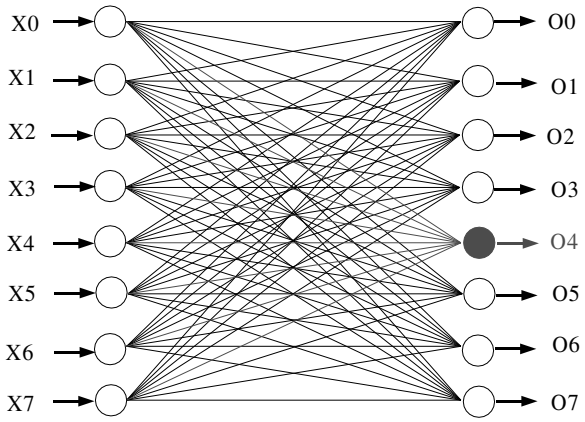

Figure 2: Architecture of the Competitive Learning Neural Network (CLNN). The shaded color indicates the winning neuron.

an *unsupervised learning* procedure, where we let the neural network deduce the appropriate behavior of a measured phenomenon online, rather than using a priori measurements.

Instead of mapping a set of inputs to the corresponding set of output vectors, CLNN tries to find the relevant information within a set of input vectors X by dividing its elements into clusters, where similar elements are grouped into one cluster. If the phenomenon can be categorized by several distinct features then separate clusters will learn these features. Figure 2 shows the architecture of the CLNN used in this study. There are 8 output units $[o_0, ..., o_7]$ connected to all 8 input units $[x_0, ..., x_7]$ by means of the weight matrix W. The rationale for using 8 units is due to the hardware/software characteristics of the embedded device and it will be explained in Section 5.

$$W = \begin{matrix} w_{0,0} & \dots & w_{7,0} \\ \vdots & \ddots & \vdots \\ w_{0,7} & \dots & w_{7,7} \end{matrix}$$

At a time t an input vector $x(t)$ is presented to CLNN and only one output unit o_i will be selected as *the winner* for this input. The winner unit is chosen based on the smallest Euclidean distance between a row vector of weighted matrix W and an input vector $x(t)$

$$o_i :\| w_o(t) - x(t) \| \leq \| w_i(t) - x(t) \|, \forall i \tag{1}$$

where $w_o(t)$ is a vector of weights that is attached to the winning neuron o_i, $x(t)$ is a input vector of sensors readings at a time t and $w_i(t)$ is a weighted vector attached to losing neurons. Note that $x(t)$, $w_o(t)$ and $w_i(t)$ are all vectors of equal dimensions. Once a winner is selected, the corresponding vector of weights $w_o(t)$ is shifted toward the input vector $x(t)$.

$$w_o(t+1) = w_o(t) + \lambda \times (x(t) - w_o(t)) \tag{2}$$

where λ is a constant learning rate and $w_o(t+1)$ is an updated weight vector of a winning neuron o_i. As a result, vector $w_o(t+1)$ is shifted closer to the input vector $x(t)$. Therefore, when a similar input vector is presented to CLNN, the same winning neuron o_i will have a greater chance to win the competition. In the absence of noisy data, it is easy to see that the same neuron may win all the time and the rest of the weights will never learn the phenomenon. To avoid such a problem, we apply a simple rule known as *leaky learning*. The weights of losing neurons are updated in a similar manner to the winning one.

$$w_i(t+1) = w_i(t) + \lambda^{"} \times (x(t) - w_i(t)), \forall i \neq o \tag{3}$$

where $\lambda^{"} << \lambda$. In this manner, even the losing clusters have a chance to learn and capture more fine grained variations in a phenomenon.

The output of a winning unit o_i is set to a vector $[min, max]$, where min and max are the lowest and highest value of sensor readings seen by the winning cluster during the training period. Lets assume that during the training period unit o_i won the competition for the set of vectors $S = [x_1, ..., x_n]$, each element in this set is a vector in a eight dimensional space, then we can define l to be the smallest and h to be be the highest sensor reading contained in S.

$$o_i = \begin{cases} [l, h] & \| w_o(t) - x(t) \| \leq \| w_i(t) - x(t) \| \ \forall i \\ 0 & \text{otherwise} \end{cases}$$

We also calculate the probability of every winning cluster to win the competition in a future as follows:

$$P(o_i) = \frac{n_i}{N} \tag{4}$$

where n_i is the number of times cluster o_i won the competition out of N times. Note that

$$\sum_{i=0}^{n} P(o_i) = 1 \tag{5}$$

Therefore, at the end of a training period, each cluster will learn some aspects of a measured phenomena which are defined by the learned intervals and the probability of their future occurrences. Lets assume that every neuron won a competition at least once, then the output of the CLNN at the end of the training period is a collection of learned intervals and the probabilities of sensor reading to fall within these intervals:

$$CLNN = \begin{array}{ll} o_0 = [l_0, h_0] & P(o_0) \\ o_1 = [l_1, h_1] & P(o_1) \\ o_2 = [l_2, h_2] & P(o_2) \\ o_3 = [l_3, h_3] & P(o_3) \\ o_4 = [l_4, h_4] & P(o_4) \\ o_5 = [l_5, h_5] & P(o_5) \\ o_6 = [l_6, h_6] & P(o_6) \\ o_7 = [l_7, h_7] & P(o_7) \end{array}$$

4 Analysis

In this section, we present theoretical analysis of the proposed algorithm. As described in the previous section, learning part of what constitutes the correct sensor readings is achieved by means of the *Competitive Learning Neural Network*. References [8, 4, 15, 13] and [7] provide a good introduction to competitive learning. The actual process of learning in neural networks is usually associated with a search in multidimensional error-surface for an optimal state that minimizes the error function. It was shown that the corresponding error function for the learning rule in equation (2) corresponds to minimizing the error function $E(w_o(t))$ by following its negative gradient [13]:

$$E(w_o(t)) = \frac{1}{2} \times \sum_{i=1}^{i=n} ||(w_o(t) - x_i(t))||^2 \tag{6}$$

where $x_i(t)$ is an input vector at a particular instance of time t, $w_o(t)$ is a weight vector of equal dimensions and n is a number of training examples for which o_i was a winner. The analysis of this learning rule can be divided onto two parts. Firstly it can be shown that the $E(w_o(t))$ indeed seeks to find a minimum for the equation (2) [13]. Secondly it can be demonstrated that under certain assumptions $E(w_o(t))$ converges to the equilibrium state which is proportional to the conditional probability of a neuron o_i winning when the input vector $x(t)$ is present [4, 7, 1].

Theorem 1 *The error function:*

$$E(w_o(t)) = \frac{1}{2} \times \sum_{i=1}^{i=n} ||(w_o(t) - x_i(t))||^2 \tag{7}$$

minimizes the weight updates of a learning rule

$$w_o(t+1) = w_o(t) + \lambda \times (x(t) - w_o(t)) \tag{8}$$

where $w_o(t)$ is the weight vector at the time t, $w_o(t+1)$ is an updated weight vector, $x(t)$ is the current input vector of sensor readings and λ is a learning rate.

Proof 1 *In order to achieve the equilibrium state we need to move weighted vector $w_o(t)$ from its current position toward the gradient decent direction of $E(w_o(t))$, such that:*

$$\triangle w = -\lambda \times \frac{\partial E(w_o(t))}{\partial w_o(t)}$$

and

$$w_o(t+1) = w_o(t) + \triangle w$$

$$\frac{\partial E(w_o)}{\partial w_o(t)} = (w_o(t) - x(t)) \frac{\partial (w_o(t) - x(t))}{\partial w_o(t)}$$

where

$$\frac{\partial (w_o(t) - x(t))}{\partial w_o(t)} =$$

$$[\frac{\partial(w_{o1}(t) - x_1(t))}{\partial w_{o1}(t)}, ..., \frac{\partial(w_{on}(t) - x_n(t))}{\partial w_{on}(t)}] =$$

$$[1, ..., 1]$$

and hence,

$$\frac{\partial E(w_o)}{\partial w_o(t)} = w_o(t) - x(t)$$

and

$$\triangle w = -\lambda \times (w_o(t) - x(t)) = \lambda \times (x(t) - w_o(t))$$

therefor

$$w_o(t+1) = w_o(t) + \triangle w = w_o(t) + \lambda \times (x(t) - w_o(t))$$

which is equivalent to equation (8).

The learning rule in equation (8) brings winning neuron's weight vector closer to the training example $x(t)$. Now lets consider the normalized learning rule[7, 1] by setting

$$\triangle w = \lambda \times (\frac{\sum_{i=1}^{n} x_i(t) k_i}{\sum_{i=1}^{n} k_i} - w_o(t))$$

Let o_i be the winning neuron when vector $x(t)$ is present and

$$k_i = \begin{cases} 1 & \text{if the } o_i \text{ is the winner for } x(t) \\ 0 & \text{otherwise} \end{cases}$$

Let $P(x_i(t))$ be the probability that $x_i(t)$ is presented and $P(o_i)$ be the probability of o_i winning the competition for this input vector. Then the *equilibrium state* μ for the cluster o_i can be expressed as:

$$\mu = \sum_{i=1}^{i=n} \triangle w P(x_i(t)) P(o_i) \tag{9}$$

hence

$$\begin{aligned} &\mu = \textstyle\sum_{i=1}^{i=n} \lambda(\frac{x(t)}{k} - w_o(t)) P(x_i(t)) P(o_i) = \\ &\lambda \textstyle\sum_{i=1}^{i=n} \frac{x(t)}{k} P(x_i(t)) P(o_i) - \lambda \sum_{i=1}^{i=n} w_o(t) P(x_i(t)) P(o_i) \end{aligned} \tag{10}$$

where

$$\frac{x(t)}{k} = \frac{\sum_{i=1}^{n} x_i(t) k_i}{\sum_{i=1}^{n} k_i} \tag{11}$$

At the equilibrium state $\mu = 0$ and if all clusters contain the same number of vectors then

$$w_o(t) = \frac{\sum_{i=1}^{i=n} x_i(t) P(x_i(t)) P(o_i)}{k \sum_{i=1}^{i=n} P(x_i(t)) P(o_i)} \tag{12}$$

Graphically, $w_o(t)$ at the equilibrium state corresponds to the center of the cluster o_i.

Earlier we mentioned that the competitive learning converges under certain assumptions. One such assumption is that the input sensor readings can be grouped into disjoint clusters. If this assumption does not hold, the system may have many of such equilibrium states, some of which may be more stable than others. In our study such variation occurs when there is a large amount of noise present in sensor readings or there is a high variability in the phenomenon's behaviors.

Finally, we present the complexity analysis of the learning phase. Note that the computational complexity is very much implementation dependent and in this section we present the analysis of a simple implementation.

Algorithm 1 the implementation of simple competitive learning algorithm for a single input vector $x(t)$.

1. For the input vector $x(t)$ calculate the Euclidean distance from all the weights vector to $x(t)$: $O(n^2)$
2. For the winning neuron o_i update l, h and $P(o_i)$: $O(n)$
3. Do While ((number of epochs $>$ 0) or (the distance between $x(t)$ and $w_o(t)$ $>$ the minimum error tolerance))
   - update weights of a winning unit: $O(n)$
   - calculate the new distance between w_o and $x(t)$: $O(n)$
   - update weights of losing neurons: $O(n)$
4. Update the final $P(o_i)$: $O(n)$

Algorithm 1 shows the implementation of simple competitive learning for a single input vector of sensor readings. Lets assume that we train the CLNN on N number of input vectors, each vector of size n, then the complexity Cm of CLNN is defined as follows:

$$Cm = \\ O(N) * (O(n^2) + O(n) + 3 * epochs * O(n) + O(n)) = \\ O(N) * (O(n^2) + (2 + 3 * epochs) * O(n))$$

Because the input vector size n is a constant parameter (for example, 8 in our implementation), the complexity of the algorithm is effectively $O(N)$

5 Experimental Results

In this section, we describe the evaluation of the proposed algorithm on a physical sensor readings and the experimental results obtained. We also explain some of the choices made for the implementation of CLNN.

The objective of our evaluation were three-fold:

- Is the algorithm effective? Can it learn the data characteristics in the presence of noise and natural changes in the phenomena?
- How much noise in sensor readings can be present or how frequently the phenomenon can vary before the algorithm losses its ability to deduce the correct sensor readings?
- Is the algorithm efficient? How much time should we train it to correctly infer sensor data?

Figure 3: The physical network topology of the motes. Motes are laid out in an 8 feet by 10 feet area. Each mote measures its light sensor reading every minute and sends it to the gateway device.

5.1 Implementation

For the evaluation of the proposed algorithm we set up a sensor network of 30 micaZ motes in a grid topology, as shown in Figure 3.

Each mote collects eight samples of light sensor readings per minute and sends these samples to a clustering node. We choose light as the sensing modality for our experiments because it is easy to introduce temporal and permanent noise into sensors readings by casting a shadow over a sensor or by covering its light sensor with a paper cup, as shown in Figure 4. Moreover, it is relatively straightforward to control the light intensity, in an indoor setting, allowing for repeatable experiments.

We limit the number of data samples to 8 because TinyOS packet size does not support more than 29 bytes of data payload, each reading occupies 2 bytes and we reserved 4 bytes for nodes Cartesian coordinates. This is also why the number of input and output units of CLNN was set to eight. During the training period CLNN was trained on every received packet.

There are several important parameters that affect CLNN performance. First of all, one of the main problem of convergence to a locally optimal solution that is associated with the neural networks is related to initialization of its initial weight values. In this study we divided the entire spectrum of possible sensor readings into 8 zones and each vector of weights was initialized to values from a particular zone.

For example, the light sensor can not return a negative value or a value above 1000 engineering units. Note, that this is not strictly true for all cases. If nodes are completely faulty, then they may try to return these values, but such readings can easily be checked and eliminated by introducing guard conditions. Weights of the neuron o_0 were initialized to 0.0, weights for the output neuron o_1 were initialized to 125, weights of o_2 were initialized to 250 engineering units, etc and so on. This permits our CLNN to learn an entire spectrum of possible sensors readings.

Secondly, there are two related parameters that affect the rate of CLNN learning. Namely, number of epochs that we train CLNN for each input vector $x(t)$ and the amount of time we allocate for the training period. With a longer training period, more input vectors in a CLNN will be learned, and larger number of clusters will learn

Figure 4: To simulate faulty sensors, motes are covered with cups to affect their direct observation of light.

these input vectors. With a larger number of iterations over the same input sample and smaller training period, clusters will learn more about each input vector but the CLNN will learn over a lower number of samples.

The trade off between these choices is largely dependent on the measured characteristics of the phenomenon. If we expect our environment to have a lot of noisy data then the training period should be longer. This will allow CLNN to capture more of the non-noisy data and therefore it would predict the correct sensor readings with a higher level of confidence. Subsection 5.2 contains the evaluation of the number of training epochs in the stable environment.

We run our algorithm on three different types of phenomenon's behavior. Firstly, we kept the light level in our office on during the training period and introduced faulty nodes by covering them with a white paper cup. Subsection 5.3 contains the results from this experiment.

Secondly, we introduced dynamics into phenomenon's behavior by casting a time varying shadow over sensors and by switching the light off for an approximately half of the training period. Subsections 5.4 and 5.5 contain the results of these experiments.

5.2 Varying the Training Period and Epoch Frequency

This subsection presents the results for evaluation of the trade off between the number of epochs and the duration of the training period. Figure 5 shows the convergence of weight vectors with 10 epochs and the training period was set to two hours. As can be seen from this figure even the furthest weights vector w_0 learned the phenomenon in over an hour. Figure 6 shows the distribution of light reading across nodes. Remarkably, at approximately 15:20, a node at the position $(0, 10)$ sent a noisy data and as can be seen from Figure 5 CLNN immediately learns these noisy values. Nevertheless, the level of light in a computer lab is a reasonably stable environment. Therefore, we do not need all output units to learn the input vectors and hence, we do not need such large training period.

To study the impact of number of epochs on the learning rate of CLNN, we kept the training period constant at 5 minutes and we vary the number of interactions for each input vector. Figures 7 shows the learning rates

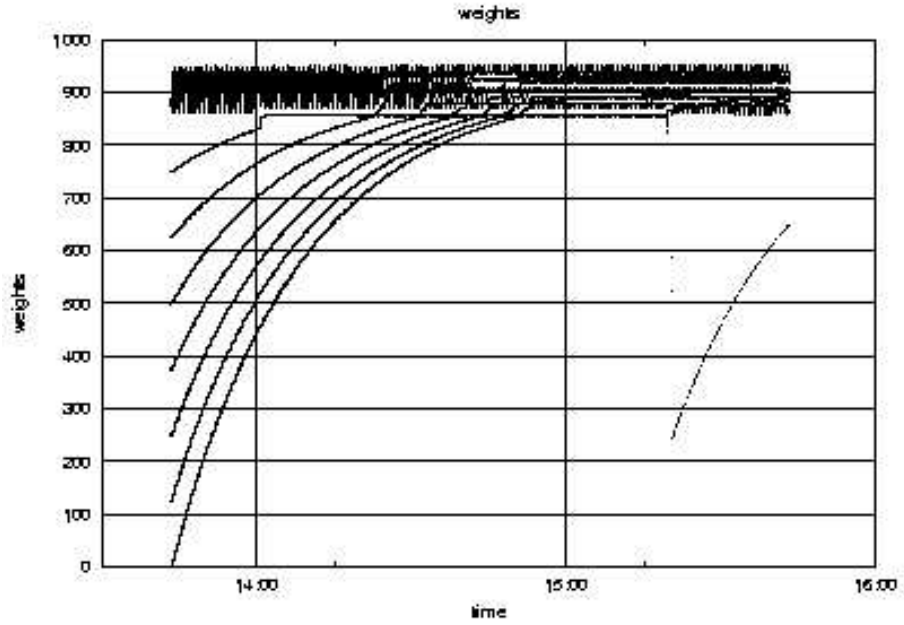

Figure 5: The convergence of weights using a 2 hour training period, with 10 epochs. Nearly all the weights converge within a 2 hour period. The non-converging weight at the right corner indicates a random source of noise.

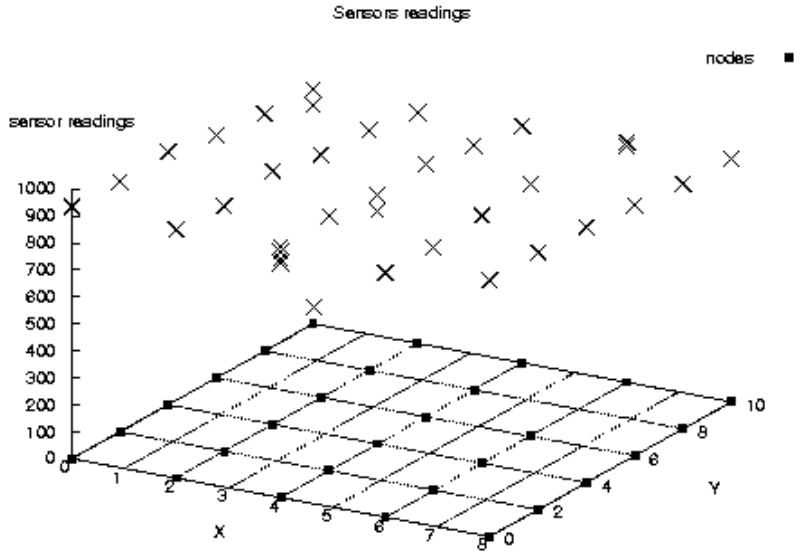

Figure 6: The readings of the sensors after 2 hours. No sensor reading is shown at location (6,8) because the battery of that mote failed during the experiment.

of weights for 10, 150, 500 and 1000 epochs respectively. As expected the fastest learning rate occurred at the 1000 epochs. However, surprisingly the rate of learning did not increase proportionally to the number of epochs. We can see that the weights learned the characteristics of a phenomenon twice as fast if the number of epochs is increased from 10 to 150. But when the number of epochs is increased from 150 to 1000 we did not observe any such significant increase in the learning rate. This suggests that a trade off can be made between the computational capacity of the available hardware and the number of iterations for each input vector.

5.3 Varying the Number of Faulty Nodes

This subsection studies the number of faulty nodes that can be present during the training period before the algorithm losses its ability to distinguish between the correct and faulty readings. As can be deduced from Figure 5, the normal light readings in our lab should lie between approximately 850 and 950 engineering units.

Figure 5.3 and 5.3 shows the actual sensor values learned by different clusters and the level of confidence in the correctness of these data values. As can be seen, cluster number 7 and cluster number 6 learned the correct characteristics of a phenomenon in most cases. The rest of the clusters either never won a competition or captured the noise in the environment. As expected, the confidence level in the correct data decreases as the number of faulty nodes increases and in order to deduce the correct readings, a confidence level should be ≥ 0.5. As can be seen from Figure 5.3, the algorithm correctly learned and predicted sensor readings when up to 60% of nodes were faulty. However, with 60% of nodes being faulty, the confidence level for the correct data was lower than 0.5. Hence, in the presence of the constant noise the algorithm can deduce the correct readings when up to 50% of nodes are working correctly. Another interesting aspect of learning is the distribution of data values and the trajectory of the weights for each cluster during the learning period. Figures 10 and 11 show the trajectory of average weight values for each cluster for the scenarios with the percentage of faulty nodes varying form 0% to 100%. The crosses in the graphs represent the average of vectors containing the sensor readings and the small dots represent the average of weights matrices. As can be seen form Figure 10 when there is no faulty node present in the environment, data falls within a single cluster and all the weights are moving toward this data cluster. The top two clusters (weight 6 and 7) are following the non-noisy data.
As the number of faulty nodes increases, we can see the emergence of new data clusters. When only a single faulty node is present, an additional data cluster is formed at about 600 engineering units and there are 4 different clusters that learned the noise. However, their probability of winning is much lower than that of the cluster 7. As more faulty nodes are introduced we can see that more and more of the data values are falling into clusters that are lying approximately between 590 and 720 engineering units. As more of the data is getting shifted away form the top clusters, the probability of winning of the correct clusters is reduced. However, as there are multiple clusters that learn the noisy readings, the probability of winning for these clusters is less than 0.5. Even when all the nodes are faulty, the probability of winning for the cluster 5 which learned most of the noisy readings is below 0.4.

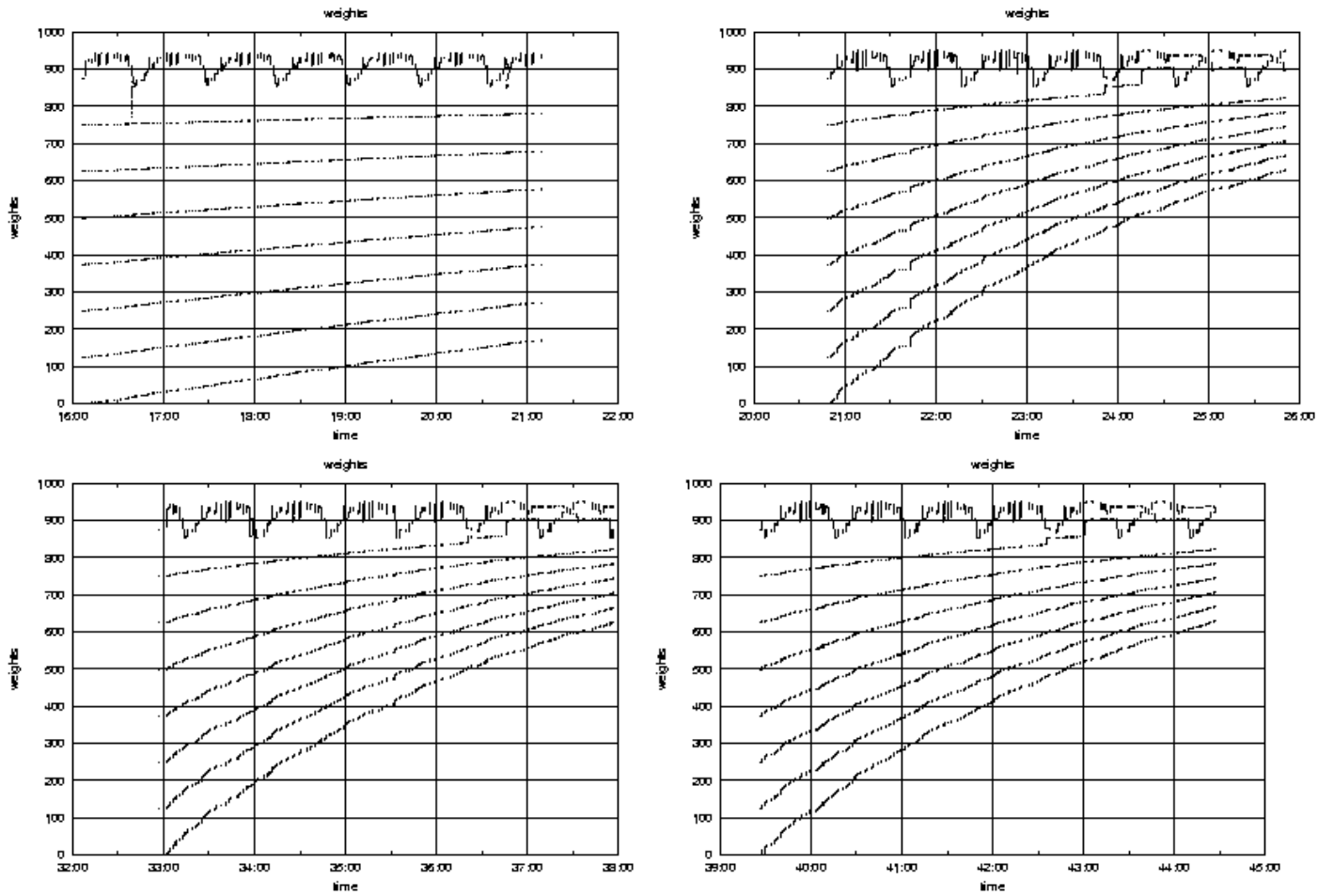

Figure 7: Convergence of weights with 10 , 150, 500 and 1000 epochs, after a 5 minute training period. Only the closest winning weight converges to the correct sensor readings after 10 epochs. However, two closest winning weights, converged after 100 epochs. No significant difference is noticed in the convergence of the weights to the correct sensor readings between 150, 500 and 1000 epochs. This indicates that 150 epochs may be more than sufficient for the algorithm.

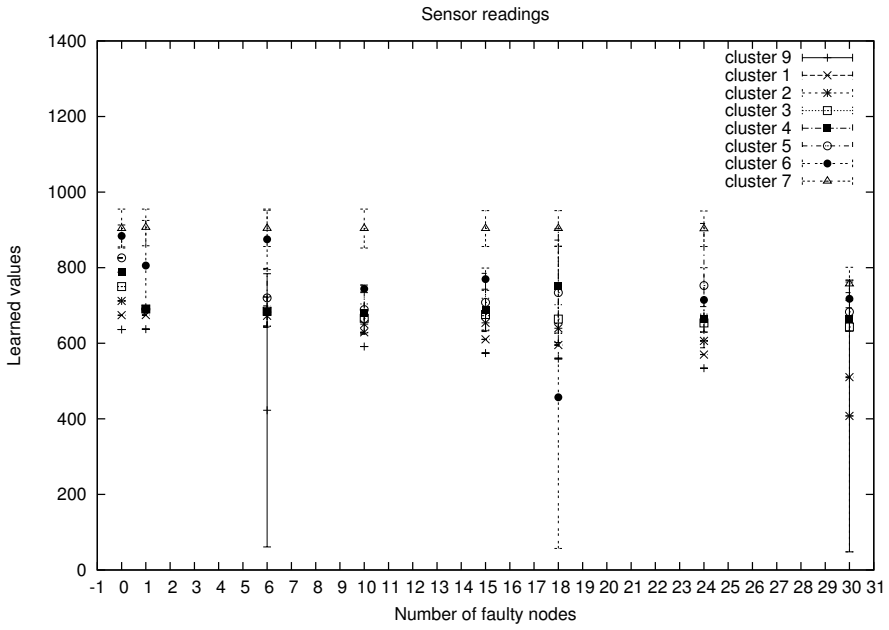

Figure 8: The range of learned values to be contained in a cluster as a function of the number of faulty nodes after the training period.

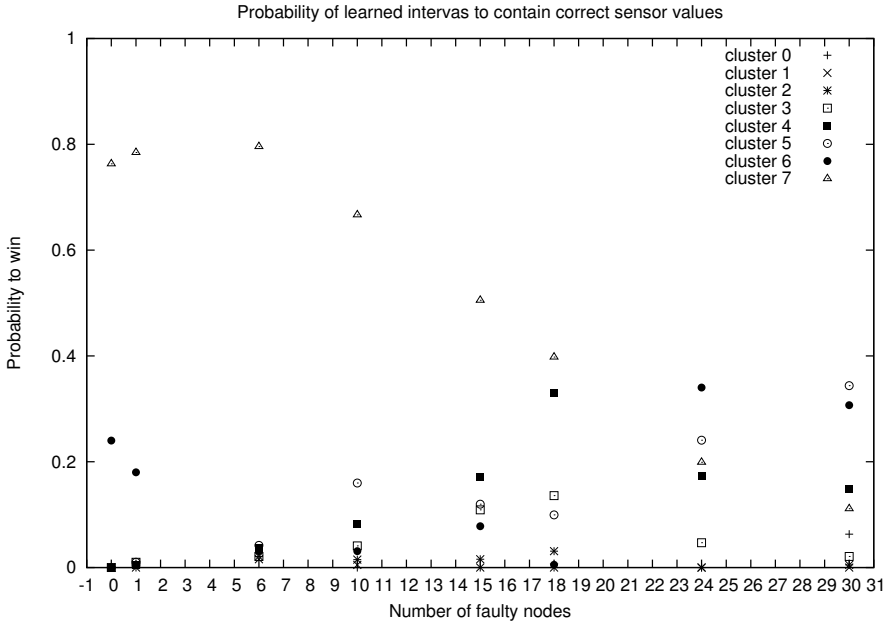

Figure 9: The range of learned values to be contained in a cluster as a function of the number of faulty nodes after the training period.

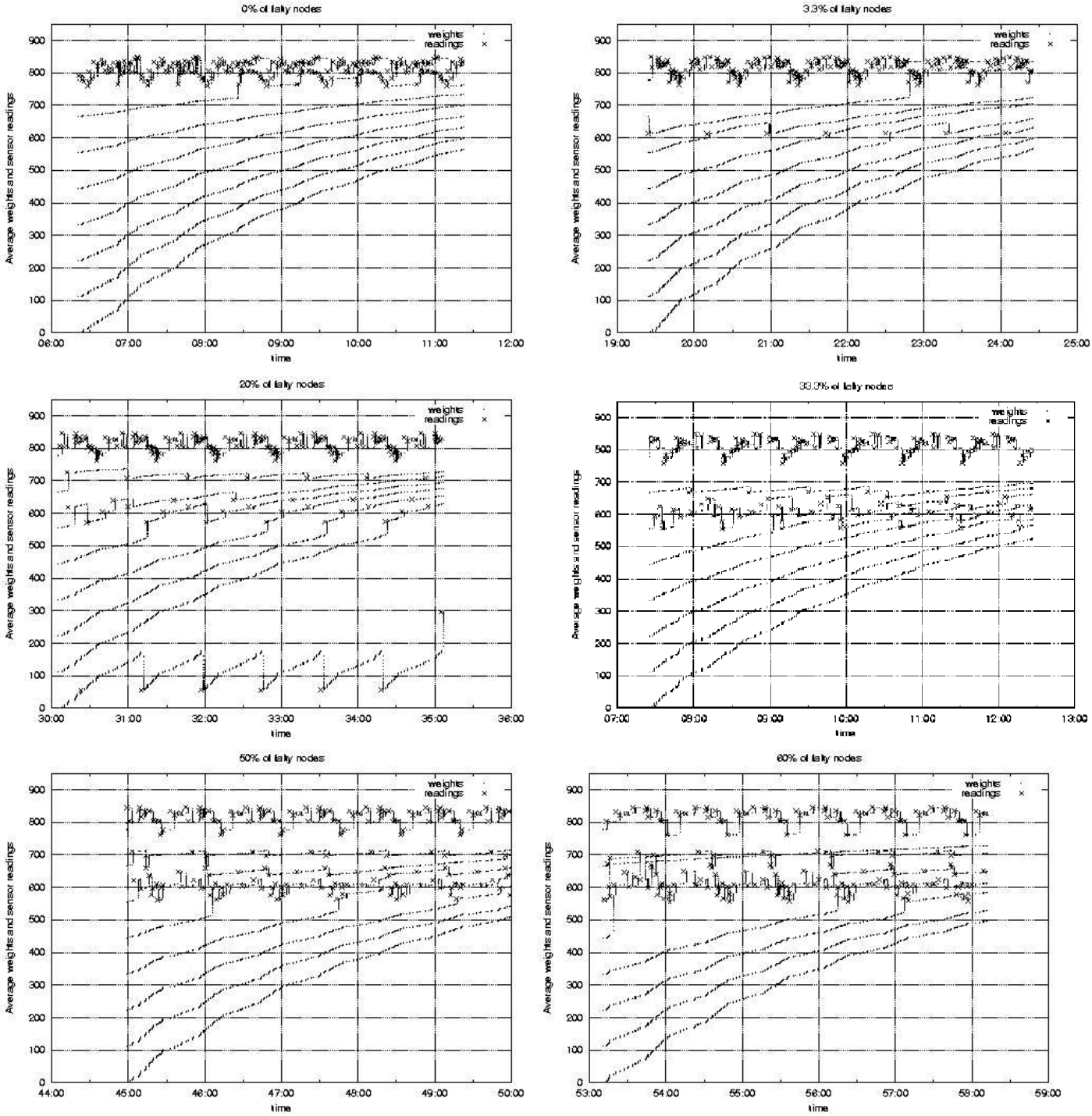

Figure 10: The trajectory of weights for the scenarios with the number of faulty nodes varying from 0%, to 60% of faulty nodes.

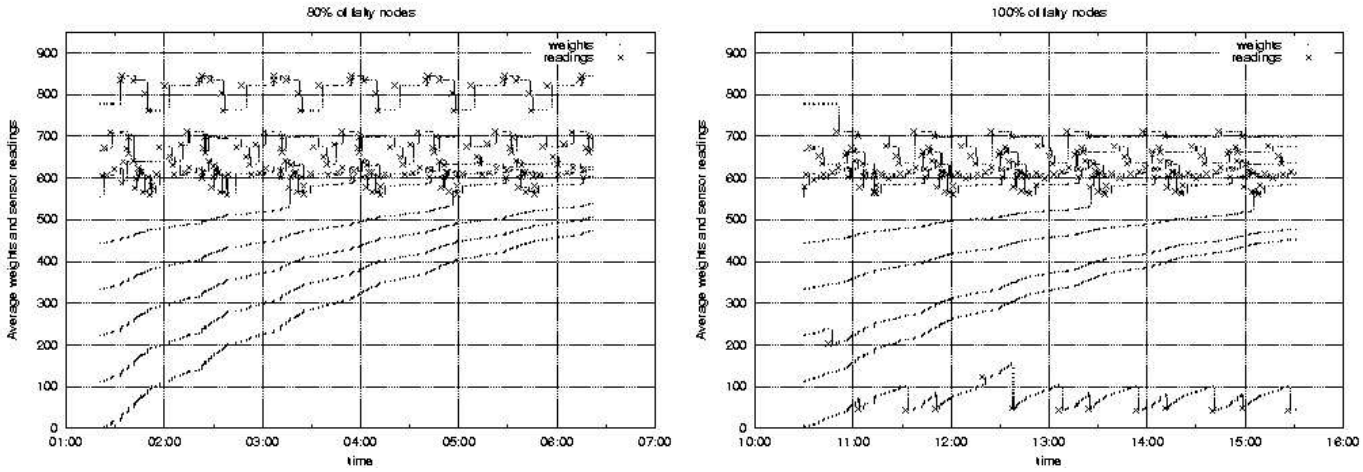

Figure 11: The trajectory of weights for the scenarios with the number of faulty nodes varying from 80%, to 100% of faulty nodes.

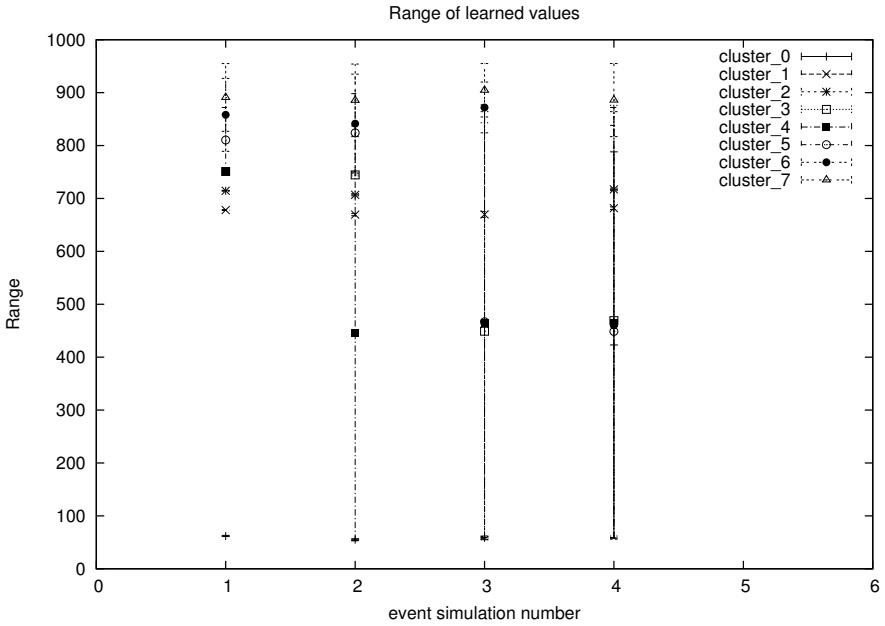

Figure 12: Distributed of learned data values in clusters for the four experiments with a moving shadow.

5.4 Dynamics: Time Varying Shadows

This experiment studies whether our algorithm is robust and can adapt to dynamics of the environment and in the presence of a temporal noise in the sensor readings. Next, we study the performance of the algorithm on the scenario of a moving shadow. A person slowly moves from the position $(2, 10)$ to the position $(2, 0)$ carrying a large box for the duration of the training period, taking a 2-meter step per minute (Figure 6). We repeated the experiment four times. Figures 5.4 and 5.4 show the learned values and the corresponding levels of confidence for all four experiments. It shows that the algorithm identifies the correct sensors readings (cluster 7), with a high level of confidence ($\geq 0.7\%$). We consider the correct sensor readings to be the constant level of light in the office and noise to be the temporal shadow readings.

Figure 14 contains the distribution of sensor readings and the trajectory of weights during the training period for all four experiments. As in previous experiment most of the average sensor readings falls in the interval learned by the cluster 7 and rest of the neurons either never won the competition or learned the temporal shadow readings.

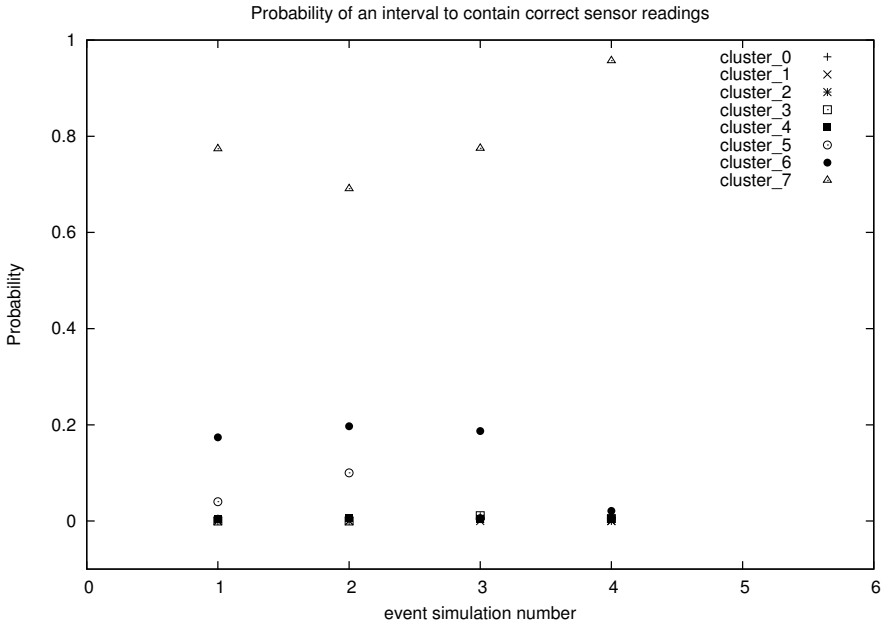

Figure 13: The confidence level associated with each cluster after the training period for the four experiments with a moving shadow.

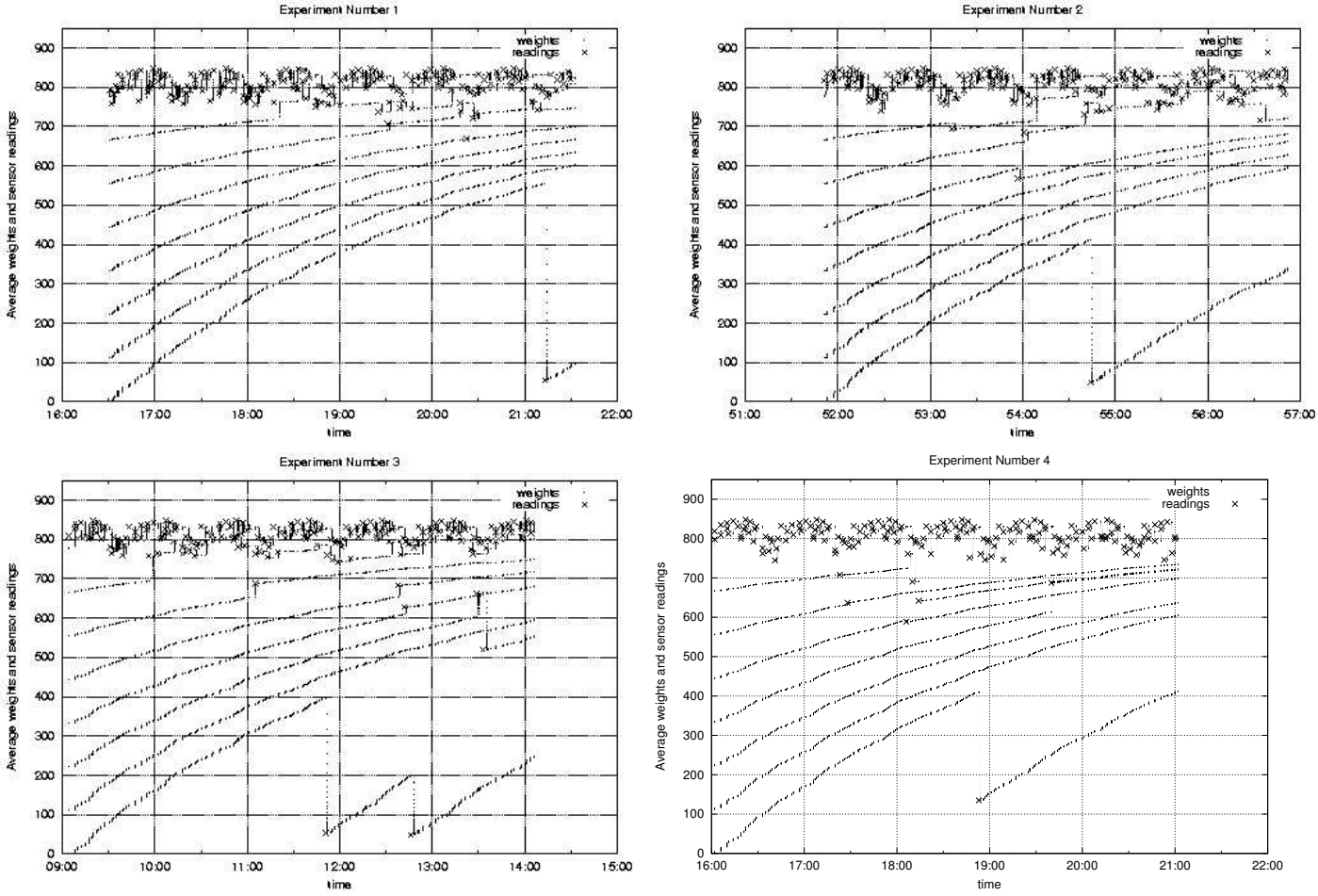

Figure 14: Trajectory of weights and the sensor readings for the four experiments with the moving shadow

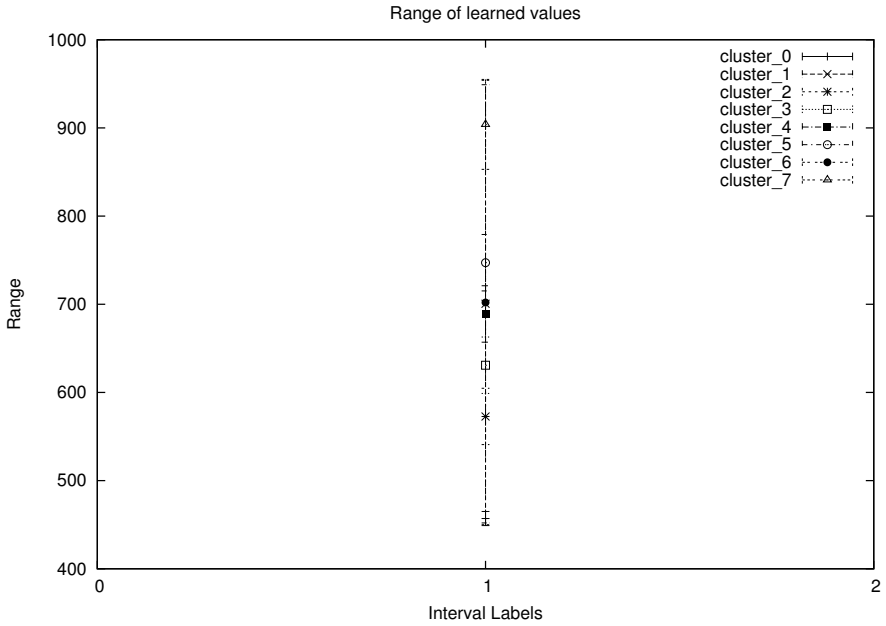

Figure 15: Distribution of learned data values in clusters for a phenomenon with distinct features.

5.5 Dynamics: Light and Dark

This experiment studies whether the algorithm can adapt to dynamics of the phenomenon where we have multiple distinct, and time-varying features of a phenomenon. We kept the lights on in our lab for the approximately half of the training period. We then turned the light off for the remaining time of the training period. Figures 5.5 and 5.5 show the learned values and the corresponding levels of confidence. It shows that the algorithm identifies the correct bright light sensors readings (cluster 7), with a level of confidence of approximately 0.5. Cluster 1, contained dark light readings and it has a confidence levels of approximately 0.4. Clusters 2, 3, 4, and 5 capture the transitional characteristics of the phenomenon.

We intended for our phenomenon to have 2 equally distinct characteristics. However, it is not possible to switch light at precisely half way through the experiment, as there are always few seconds of a human error. As you can see from Figure 5.5 there was a transitional period of approximately 30 seconds for the lights level to go down to approximately 400 engineering units. The duration of the dark training period therefore was slightly shorter than that of the light period. This is why the confidence levels of the dark feature are slightly lower than one for the bright ones. Table 1 summarizes the results obtained for the degree of accuracy on predicted values.

6 Conclusion

In this paper, we motivated and demonstrated the use of a competitive learning algorithm that successfully learns the environmental data characteristics and uses it to filter anomalous data in a sensor network, without requiring *a priori* knowledge of the environment.

Our experiments show the relationship between the noise in the sensor data, and the ability of the network

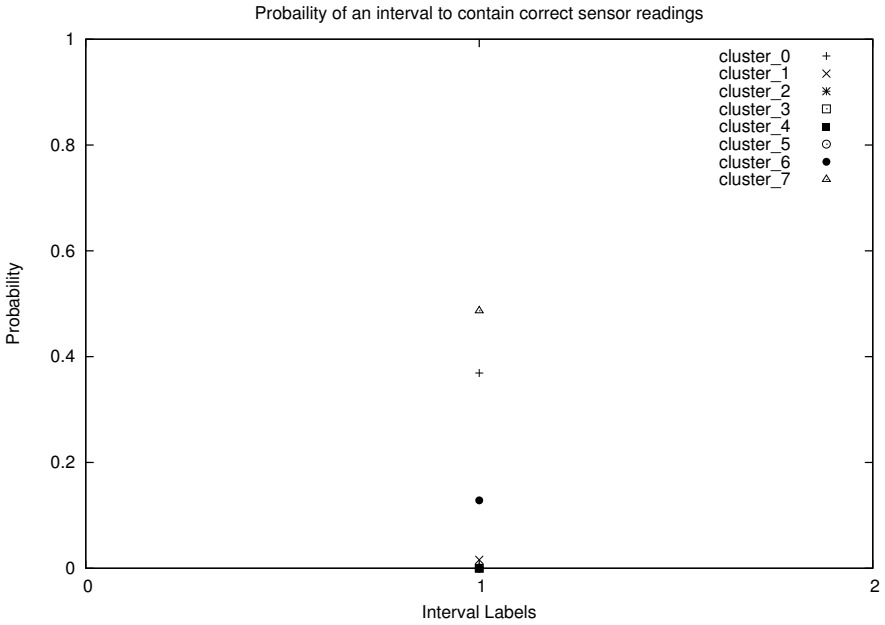

Figure 16: Distribution of learned data values in clusters for a phenomenon with distinct features and associated probabilities of clusters to win the competition.

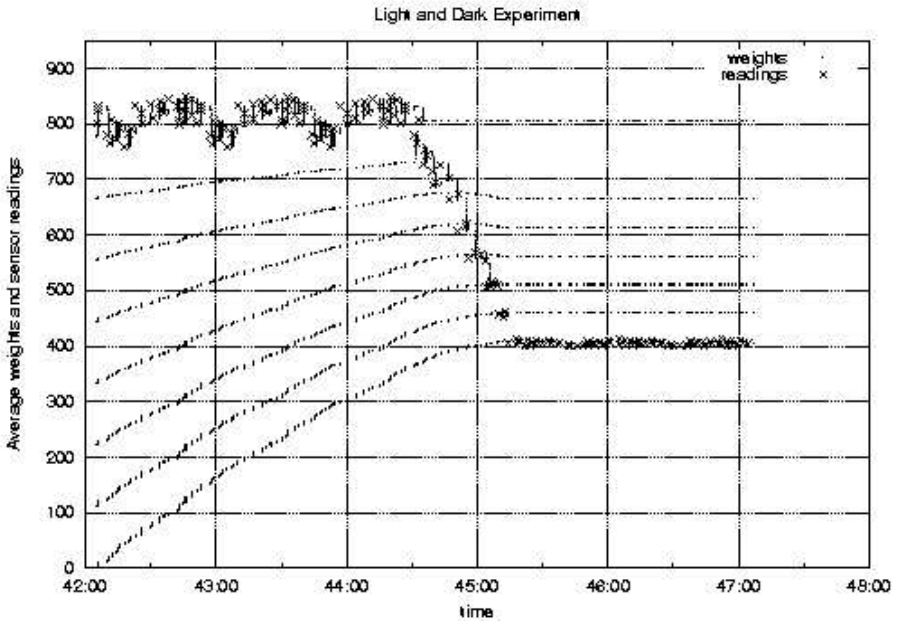

Figure 17: Trajectory of the weights and sensor readings for the Light and Dark experiment

Table 1: Summary of Results

Studied Scenarios	Correct readings of a phenomenon	Learned readings	Clusters that learned correct readings with the confidence levels
Varying the number of faulty nodes	850-950	Cluster 7 (850-950) Cluster 6 (850-950)	Cluster 7, highest confidence 80%. Clusters can learned these values when up to 60% of nodes are faulty.
Dynamics: Time Varying Shadows	850-950	Cluster 7 (850-950)	Cluster 7, highest confidence 99%.
Dynamics: Light and Dark	850-950 400-500	Cluster 7 (850-950) Cluster 1 (450)	Cluster 7, confidence 50%. Cluster 1, confidence 40%.

to learn its environment and detect them. For a periodic light data collection application, the algorithm can be trained with a small number of epochs and a small training period.

The algorithm is robust under sensor noise, it identifies the correct sensor data, even with 50% faulty sensor nodes. As the number of faulty nodes increases, the level of confidence in data gradually decreases.

Finally, the algorithm works correctly even in the presence of event dynamics. When a small, moving shadow is introduced in the environment, the algorithm still infers the bright light sensor readings with a high degree of confidence, while preserving the finer-grained data corresponding to the presence of a shadow. Because the algorithm leverages in-network processing and executes at the cluster heads of a hierarchical sensor network instead of a central base station, it can scale to potentially very large sensor networks.

For future work, we are interested in exploring further performance optimizations in the algorithm. Also, we would like to evaluate it over a wider range of sensing applications, including those featuring multiple sensing modalities.

References

[1] Yuichiro Anzai. *Pattern Recoqnition and Machine Learning*. Academic Press Inc, Iwanami Shoten Publishers, 1989.

[2] Tatiana Bokareva, Nirupama Bulusu, and Sanjay Jha. Sasha: Towards a self-healing hybrid sensor network architecture. In *Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors (EmNetS-II)*, Sydney, Australia, May 2005.

[3] Archer C. and Leen T. K.and Baptista A. Parameterized novelty detection for environmental sensor monitoring. 2000.

[4] Rumelhart D. E and Zipser D. Feature discovery by competitive learning. In *Cognetive Science.*, volume 9, pages 75–112, 1985.

[5] K.P Ferentinos, L.D. Albright, and B. Selman. Neural network-based detection of mechanical sensor and biological faults in deep-trough hydroponics. *Computer and Electronics in Agriculture*, 40:65–85, 2003.

[6] T.-H. Guo and J. Nurre. Sensor failure detection and recovery by neural networks. In *Proceedings of IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume 1, pages 221–226, Seattle, WA, USA, 1991.

[7] Mohamad H. Hassoun. *Fundamentals of Artificila Neural Networks*. MIT Press, 1995.

[8] Simon Haykin. *Neural Networks. A Comprehensive Foundation*. Prentice Hall International, Inc, 1999.

[9] Wen Hu, Nirupama Bulusu, and Sanjay Jha. A communication paradigm for hybrid sensor/actuator networks. In *Proceedings of the 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communication (PIMRC 2004)*, Barcelona, Spain, 5-8 September 2004.

[10] Crossbow Technologies Incorporated. Crossbow technologies. *http://www.xbow.com*.

[11] K.Marzullo. Tolerating failures of continuous-valued sensors. In *Proceedings of ACM Transactions on Computer Systems*, volume 8, no. 4, pages 284–304, November 1990.

[12] Bhaskar Krishnamachari and Sitharama Iyengar. Efficient and fault-tolerant feature extraction in sensor networks. In *Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03)*, Palo Alto, California, April 2003.

[13] Ben Krose and Patrick van der Smagt. An introduction to neural networks, November 1996.

[14] Ram Kumar, Vlasios Tsiatsis, and Mani B. Srivastava. Computation hierarchy for in-network processing. In *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 68–77, New York, NY, USA, 2003. ACM Press.

[15] Kishnan Mehrotra, Chilukuri Mohan, and Sanjay Ranka. *Elements of Artificial Neural Networks*. A Bradford Book, The MIT Press, Cambridge, Massachusetts, 1996.

[16] Suman Nath, Phillip B. Gibbons, Srinivasan Seshan, and Zachary R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 250–262, New York, NY, USA, 2004. ACM Press.

[17] University of California at Berkeley. Habitat monitoring on great duck island. *http://www.greatduckisland.net/*.

[18] M. Pardo, G.Faglia, G.Sberveglieri, M.Corte, F.Masulli, and M.Riani. Monitoring reliability of sensors in an array by neural networks. *Sensors and Actuators*, B 67:128–133, 2000.

[19] B. Przydatek, D. Song, and A. Perrig. Sia: Secure information aggregation in sensor networks. In *Proceedings of the ACM SenSys 2003*, Los Angeles, CA, November 2003.