INTERNATIONAL JOURNAL ON SMART SENSING AND INTELLIGENT SYSTEMS, VOL. 1, NO. 4, DECEMBER 2008

Training Data Compression Algorithms and Reliability in Large Wireless Sensor Networks

Vasanth Iyer, Garimella Ram Murthy, M.B. Srinivas
Computer Science and Engineering, International Institute of Information Technology,
Gachibowli, Hyderabad, India 500032
Email: vasanth@research.iiit.ac.in, rammurthy@iiit.ac.in, srinivas@iiit.net

Abstract

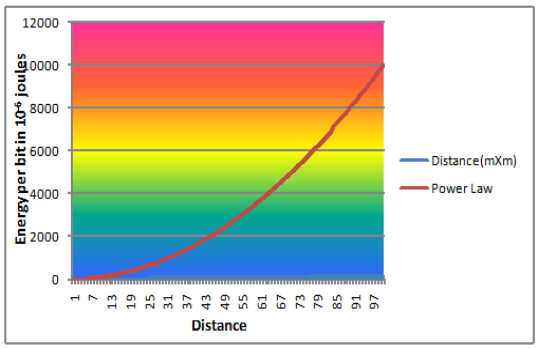
With the availability of low-cost sensor nodes there have been many standards developed to integrate and network these nodes to form a reliable network allowing many different types of hardware vendors to coexist. Most of these solutions however have aimed at industry-specific interoperability but not the size of the sensor network and the large amount of data which is collected in course of its lifetime. In this paper we use well studied data compression algorithms which optimize on bringing down the data redundancy which is related to correlated sensor readings and using a probability model to efficiently compress data at the cluster heads. As in the case of sensor networks the data reliability goes down as the network resource depletes and these types of networks lacks any central synchronization making it even more a global problem to compare different reading at the central coordinator. The complexity of calibrating each sensor and using an adaptable measured threshold to correct the reading from sensors is a severe drain in terms of network resources and energy consumption. In this paper we separate the task of comparative global analysis to a central coordinator and use a reference P_{Max} which is a normalized probability of individual source which reflects the current lifetime reliability of the sensors calculated at the cluster heads which then is compared with the current global reliability index based on all the P_{Max} of cluster heads. As this implementation does not need any synchronization at the local nodes it uses compress once and stamp locally without any threshold such as application specific calibration values (30^oC) and the summarization can be application independent making it more a sensor network reliability index and using it independent of the actual measured values.

I. Introduction

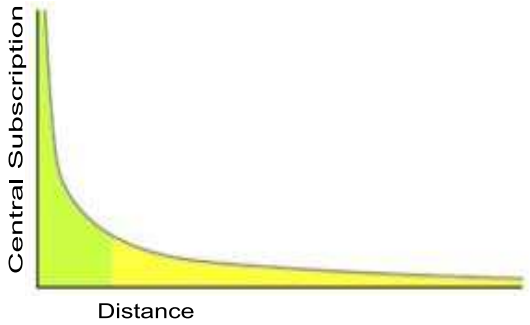
The lifetime of sensor networks is typically factored into the resources it is deployed with, as by design it is unattended (i.e. no replacement of batteries) it coexists for many months to some years. The numbers of sensor nodes are typically run into hundreds to thousands in a large environmental monitoring application. As the number of nodes in such applications are enormous than typical networks it uses a clustering algorithms in which typically 20%-30% [5] of the nodes aggregate the data of the remaining 70%-80% [5] of the connected nodes. These cluster heads are data concentrators which can be modeled as a device CODEC, compressor/decompressor. The sensors which are attached to the nodes typically sense temperature, humidity and light. It is true, however, that the sensor measurements in the operation region are spatially correlated (since many environmental phenomena are) they tend to be very similar. In a CODEC a probability model is used which gives the highest probability to the most frequently occurred values reported by the sensors within the same cluster. This allows transmitting peak values with least amount of bits as the underlying compression algorithm assigns least number of bit for frequently occurring values. This probability distribution is send with the data values to the central coordinator. So each cluster head has a unique P_{Max} [1] but not all cluster heads have the same measured value. As in recent development of VLSI and MEMS technologies have made it possible to package self-powered sensors and wireless radio components which together is capable of collecting and processing new sensor data for a period of many months to few years without replacing the internal batteries. The miniaturized sensors are sensitive to the available effective range to the energy consumed per bit. The **instantaneous drain** on the internal batteries is evident and the study shows that

$$\text{Energy consumed per bit to transmit} = \frac{100pj}{bit} m^2$$
$$\text{Energy consumed to receive a bit} = \frac{50nj}{bit}$$
$$\text{Transmit Energy} = E_{amp} + d_{i,j}^2 \tag{1}$$

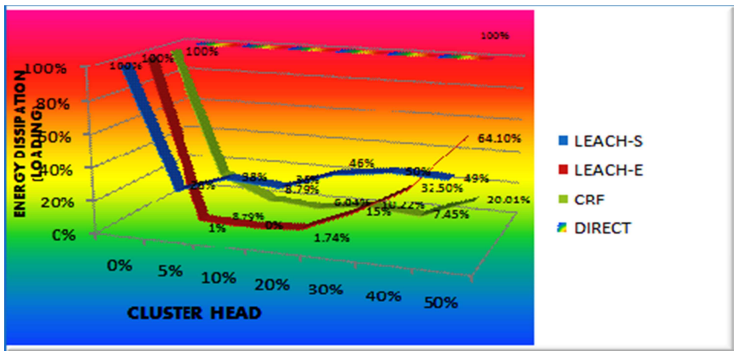
Where **d** is the distance to transmit between sensors **i** to sensor **j**, from this we get the Power rule based on the distance d of nearest sensor to the farthest away sensor, substituting in the above equation (1) and summing up the total energy required for all transmissions within one meter, two meters, three meters, four meters and extending up to (d-1) meters to a progressive sequence in equation (1.1) (as shown in Figure 1(a)).



(a)



(b)



(c)

Fig. 1. (a) Shows fixed energy overhead with distance in transmission, (b) Plot of the theoretical expected lifetime using Power Law, being used to demonstrate ranking of popularity. To the right is the long tail, to the left are the few that dominate (also known as the 80-20 rule) and (c) Simulation results with sensor nodes up to 100 using routing algorithms during lifetime calculation.

$$PowerLaw = 1^2 + 2^2 + 3^2 + 4^2 + ... + (d-1)^2 + d^2 \tag{1.1}$$

To sum up the total energy consumption we can write it in the form of Power Law equation (1.1)

$$PowerLaw = f(x) = ax^2 + o(x)^2 \tag{1.2}$$

Substituting d-distance for x and k number of bits transmitted, we equate as in equation (1.2).

$$PowerLaw = f(d) = kd^2 + o(d)^2 \tag{1.3}$$

Taking Log both sides of equation (1.2),

$$\log(f(d)) = 2\log d + \log k \tag{1.4}$$

Notice that the expression in equation (1.4) has the form of a linear relationship with slope k, and scaling the argument induces a linear shift of the function, and leaves both the form and slope k unchanged. Plotting to the log scale as shown in Figure 1(b)

we get a long tail showing a few nodes dominate the transmission power compared to the majority, similar to the Wikipedia reference 80-20 rule of Power Law [4].

A. Scale invariance property in clustering for energy dissipation in RF based applications.

As novel sensor applications are deployed to provide reliable data over the life-time [2] of the sensor network, with current routing algorithms [5] which are dependent to communicate with a central coordinator the instantaneous drain on the sensors are very demanding. A typical 9V battery communication for an RF sensor to transmit over 10 meters range will drain out as per the capacity table [3]. As shown in the previous equation in logarithmic scale for point to point transmission, we can extend this by clustering C nodes in the same range as shown in equation (1.6).

$$f(d) = kd^2 + o(d^2) \tag{1.5}$$

$$f(cd) = k(cd^2) = c^k f(d) \alpha f(d) \tag{1.6}$$

From the equation (1.6) we can infer that the property is scale invariant even with clustering c nodes in a given radius k. This is validated from the simulation results [5] obtained in Fig 1 (c) which show optimal results(minimum loading per node[7]) when clustering is $\leq 20\%$ as expected in theory(80-20 rule) from Fig 1 (b). It is true, however, that the sensor measurements in the operation region are spatially correlated, to be efficient in a large sensor network partitioning the network into special clusters in done periodically and data needs to be aggregated locally by fusing all sensor reading at the cluster head. This data is periodically routed to a central coordinator which is a collaborative effort of all the active nodes in the sensor network.

II. TRAINING DATA COMPRESSION ALGORITHMS

A. Probability Model

Most of the compression algorithms use a probability model based on the entropy of the source. Entropy of general source is given by

$$H(S) = \lim_{n \to \infty} \frac{1}{n} G_n, where \tag{2}$$

$$G_n = -\sum\sum...\sum P(X_1 = i_1, X_2 = i_2 ... X_n = i_n) \log P(X_1 = i_1, X_2 = i_2 ... X_n = i_n)$$

And is a sequence of length n from the source. In sensor each element in the sequence is independent and identically distributed (i.i.d.), then we can modify the entropy to the first order to equation (2)

$$H(S) = -\sum P(X_1) \log P(X_1) \tag{2.1}$$

B. Aggregation Model

If the cluster size in n, from the cluster equation (8) then the entropy of data aggregation [2.1] is

$$H(S) = -\sum_{i=0}^{n} P(X_1) \log P(X_1) \tag{2.2}$$

In a lossless mode if there are no faults in the sensor network then we can show that the highest probability given by P_{Max} is ambiguous if its frequency is $\leq \frac{n}{2}$ otherwise it can be determined by a local function.

C. Local P_{max} functions

$$|Pmax| = \begin{cases} local, & \text{for Pmax} \geq \frac{n}{2} \quad (2.3a) \\ global, & \text{for Pmax} < \frac{n}{2} \quad (2.3b) \end{cases}$$

Where n is total number of sensors placed in a cluster head. Here the probability of sampling similar values are highly correlated as in the case of environmental sensing the $P_{max} \geq 0.5$ then the entropy can be re-calculated as

$$H_{good} = -0.6 \log_2 0.6 - 0.4 \log_2 0.4 = 0.956 \tag{2.4}$$

per cluster head. For a good distributed clustering algorithm it uses 20% cluster heads [5] then the total entropy of the network will be $0.958 \times 20 = 19.16$ per round. To further calculate the algorithm efficiency the most popular being Huffman coding [1] which has a lower and upper bound for a given P_{max}. The Kraft-McMillan inequality there exist a uniquely decodable code with code word l_i. The average length of the code can be upper-bounded by using the right inequality.

$$l_{avg} = \sum_{i=0}^{n} P(a_i) l_i \leq \sum P(a_i) \left[\log_2 \frac{1}{P(a_i)} + 1 \right] \tag{2.5}$$

In fact it can be shown that if P_{max} is the largest probability in the probability model then for $P_{max} < 0.5$, the upper bound is

$$H(S) = P_{max} \tag{2.6}$$

While for $P_{max} < 0.5$ then the upper bound is

$$H(S) = P_{max} + 0.086 \tag{2.7}$$

Now to calculate the average numbers needed for both P_{max} using Huffman coding, when $P_{max} > 0.5$ then using the above equations we get

$$H(S) = P_{max} + 0.086 = 0.958 + 0.4 + 0.086 = 1.44bps \tag{2.8}$$

If the symbol distribution is highly skewed then it takes few extra bits which are certainly the case in sensor networks with no faults. To find the efficiency of the coding we use

$$Efficiency = \frac{lavg}{H(S)} = \frac{1.6}{0.78} \times 100 = 45\% \tag{2.9}$$

more than the source entropy using Huffman coding. The data payload which is aggregated for each round with source entropy which needs a minimum of 0.78 bits at the cluster heads, actually it represents 1.6 bits still reducing the total number of bits to be transmitted to the coordinator after coding.

III. P_{MAX}, Local value-independent redundancy filter for data aggregations at the cluster heads

Sensors networks when deployed has a predictable energy resource and uses a well distributed routing algorithm to aggregate its data to periodically send the sensed data to a central coordinator for further processing by using minimum resources. The goal of all the aggregation algorithms it to maximize the network reliability index which is a global threshold and reflects the health of the network. In the central coordinator mode we like to implement a classifier which allows maximizing on the redundancy of correlated data from each node as it learns during the lifetime of the network and maximizes on the fault by uniformly distributing the load on the node. This extends the useful lifetime of the sensor network by decreasing the number of energy holes in the network and corrupting good sensors readings.

A. Localized Classifier - Fault = 0

In the life-time of sensor networks when it has no faults then the case (fault=0) the classifier's view will be as shown in Figure 2. This given the training model of the classifier a good partition of the life-time of the sensor network. As this information is needed latter when faults happen in specific areas the cluster head transmit this data periodically to the central coordinator so it can send it to the host for latter comparisons.

B. Localized Classifier - Fault <= n

From the figure 3 it is clear that the localized aggregation function P_{max} is effective and the classifier rule will be able to differentiate the good and bad readings efficiently. As the fault rate increases (fault $< n$) then we have differentiation inside

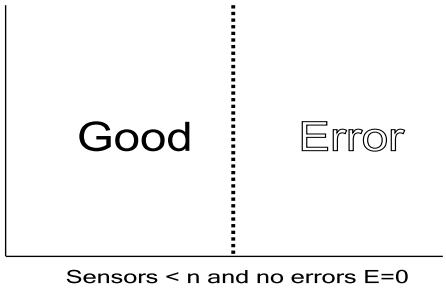


Fig. 2. Simple classification of faulty sensors.

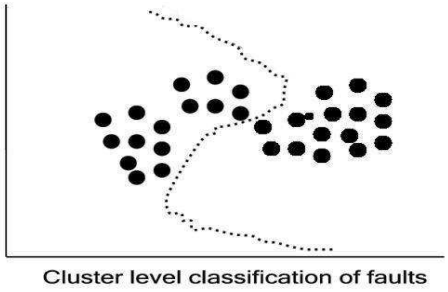


Fig. 3. Cluster level classification of faults.

one cluster as the sensor reading are correlated the cluster head is able to differentiate within the cluster boundaries. The classifier uses a local rule for this case.

C. Localized Classifier - Fault >= n

Now considering cases (fault $> n$) it uses sampled values from border nodes as well as some distributed nodes to compare and obtains a new global fault function. It is a significant task as shown in figure 4 to compare and correct the values to an expected value. The classifier uses a *Bayesian approach* in which it has to maximize on the posterior probability with existing prior probability. The classifier uses P_{max} as a reference if it is not able to resolve then, it takes all the faulty nodes and uses the highest P_{max} of the classified nodes and extracts the value as the best approximation for all the correlated sensors. We will say that we are trying to find the correction c, out of all possible corrections, that maximizes the probability of c given the original measurement M:

$$P_{MAX} = P\left(\frac{C}{M}\right) \tag{3}$$

By Bayes Theorem this is equivalent to equation below

$$P_{MAXC} = P\left(\frac{M}{C}\right) P(C) \tag{3.1}$$

P(c) the probability that a proposed correction c stands on its own. This is called the correlated cluster model.

$$P_{MAXC} = P\left(\frac{M}{C}\right) P(C) \tag{3.2}$$

This is called the correlated cluster model. $P\left(\frac{M}{C}\right) P(C)$ the probability that M would be measured by itself when the network meant c. This is the error model which is given by equation (3.2). P_{maxc},the control mechanism, which says to enumerate all feasible values of c, and then choose the one that gives the best combined probability score.

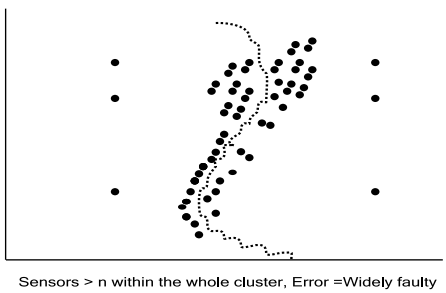


Fig. 4. Classification boundary of Faulty sensors.

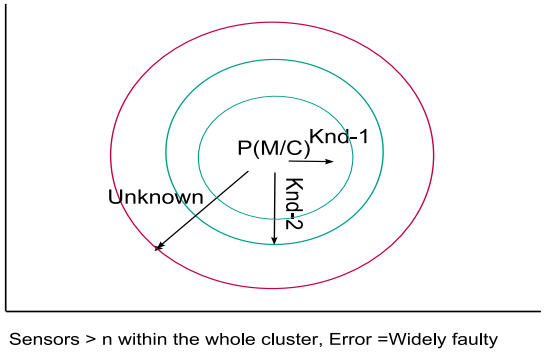


Fig. 5. Simulation setup of K-neighborhood algorithms.

IV. P_{MAX} GLOBAL, FEATURE EXTRACTION USING PAST VALUE FILTER OF ALL THE DATA AGGREGATED BY THE CLUSTER HEADS AT THE CENTRAL CO-ORDINATOR

The off-line process helps to train the model with the frequency of P_{max} generated by the sensors as in equation (2.3a) and (2.3b) which are in the fault mode but could have good readings. Below is the pseudo-code for the training the features.

Input: Measured Values from Sensors

Output: Weighted Measured Values in 'snapshot-sensor.db'

Model = collection.defaultTopologyID() **foreach** *f in features* **do**
    Model[f] += 1

end

Return model NSENSORS=train(Pmax(file('snapshot-sensor.db').read())

At this point, NSENSORS [M] holds a count of how many times the measured value M has been seen. Now let's look at the problem of enumerating the possible correction c of a seen measured value M. It is common to talk of k-neighborhood distance between two sensors, this is shown in figure 5, the number of comparisons it would take to confirm its relative measurement. If P_{max} does not get an instant match then it has to find it in K-neighborhood distance-2. In the expanded search if found then it can approximate to measured value which closely matches with an existing sensor of the same P_{max}. The process stops if there was no match for the seen frequency which is termed unrecoverable fault(unknown) as shown in figure 5. The next section deals with such cases using training database. Here is the pseudo code to return all measured corrections c that are K-neighbor distance away from sensor with a measure M:

Input: Measured Value P_{MAX} from Sensors

Output: Corrected Value (C) using K-Neighborhood distance vector N1,N2...

foreach *i in range(n) for c in the sampled live measurement* **do**
    KND1(Pmax) N+ (Pmax) return [Pmax [0:i] + c + Pmax [i+1]

end

With the implementation of this classifier we get an fault rate of $\leq 10\%$. The input and outputs corrections are shown in TABLE I.

V. TRAIN CLUSTERING FEATURES

In simple cases the sensor network uses local cluster head functions to predict the best correction but as the fault rate increases which is the case in large sensor network deployments. The central coordinator needs a way to cross validate the measured data to accept it or to make possible corrections by using the nearest correction found by using a global function. As this performed at the coordinator is not limited to any energy constraints and can use sophisticated methods such as training and feature classifiers. The main goal of the cross-validation logic as shown in figure 6 is to compare with live values as the primary factor, the more connected the network the better is its reliability. This factor would correct most of the anomalies which could occur due to bad calibration or external noise. Even the secondary factor which is to find an equivalent connected path further away from the cluster head and use its measured value to correct the currently seen value at the faulty sensor. This correction process is used even more as the sensor network becomes widely faulty [8] where the existing of full clusters are minimum or none.

VI. MULTI-FEATURE TRAINING

We like to train the feature vectors in a way that it can handle most of the aggregation locally and minimally globally. The two features are P_{MAX} which takes off the redundancy based on value locally equation (2.3a, 2.3b) and P_{MAXR} which is based on the relevance of the current measurement when exceeding a given threshold based on the trained data. Table II shows example local temperature reading in this case the P_{MAX}= 0.40, hence a high measured value of 75.0 can be safely ignored. This can be adapted efficiently using a linear discriminate analysis (LDA) as shown in Figure 7, 8. As this needs a lot of computing resources we differ such machine learning techniques to be implemented globally. In table III the trained values are corrected and are shown in 'C' and the new measured data set is shown as 'M'. Here again the sensor measures a high value

TABLE I
CLASSIFIER PERFORMANCE COMPARISON

Sample	K-Neighbor Distance- 1	K-Neighbor Distance- 2	Fault Rate
Sim Run-1	98%	1%	1%
Sim Run-1	91%	1%	8%

TABLE II
CLASSIFIER PERFORMANCE USING P_{MAX} LOCAL

Classification	Temperature Value	P_{MAX}	P_{MAXC}
M	40.00	0.10	
M	32.00	0.20	
M	44.00	0.30	0.40
(M)	75.00	0.10	
Aggregated	35.00		0.40

at 75.0 with a corresponding of P_{MAX}= 0.10 the classifier's other feature vector is matching similar readings reported earlier or newly available fused values from boarder nodes, in this case there is a such high relevant values in the training data set so comparing the measured data set the training data set we have equation (24). Substituting $P(B) = 0.30$ and $P(A) = 0.1$, the conditional probability of $P\left(\frac{A}{B}\right) = 0.30$ which is relevant to being a reportable event with high probability.

$$P_{MAXC} = 0.04, P_{MAXR} = 0.30 \tag{6}$$

$$P\left(\frac{A}{B}\right) = P\left(\frac{A \cap B}{P(B)}\right) \tag{6.1}$$

The global classifier updates the new measured value over P_{maxc}, as it is highly a probable event to be reported at the central coordinator. In the faulty case which is given in table IV we similarly shown as before. $P_{maxc} = 0.40, P_maxr = 0.30$. Substituting the values of P(A)= 0.30 and P(B)=0.0 in equation (6. 1) we get a faulty case as $P_{maxr} = not$ effected. The high value measured is treated as faulty as in this case the training data set it cannot find any matching value. Figure 10, shows LDA based classifier using dot product.

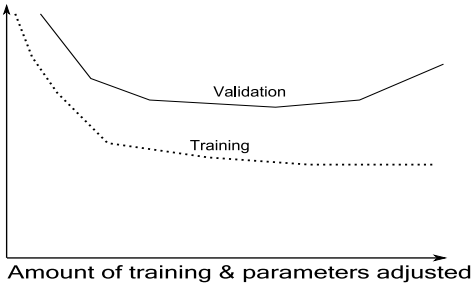


Fig. 6. Learning and training during correction as a global function.

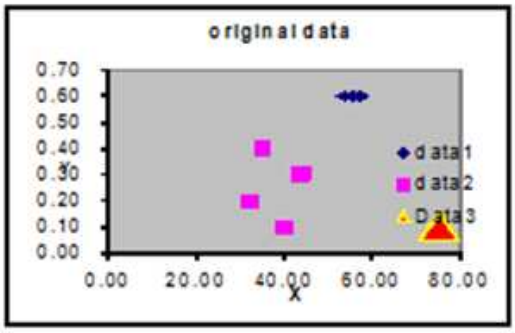


Fig. 7. Measured values using local P_{MAX} local LDA, with $\omega 1, \omega 2$.

VII. SORTING OUT FAULTY EVENTS

The above methods allow dividing good measured value and the faulty values measured data locally and globally. As the network becomes large there is a lot of redundant data and it hard to scale which makes it harder to provide a good solution. One of the alternate methods [7] used to have a better scale is using fuzzy logic. But here we like to define an event. An event can be defined in terms of entropy or self information or surprisal of message m. Where $p(m) = Pr(M = m)$ is the probability that message m is chosen from all possible choices in the message space M. From equation (7) we can define an

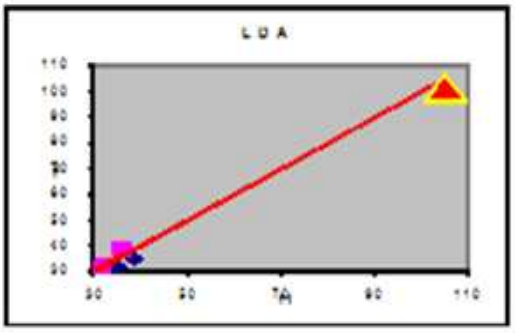


Fig. 8. Classification aggregated values using $P_{MAX}LDA\ \omega 1, \omega 2$.

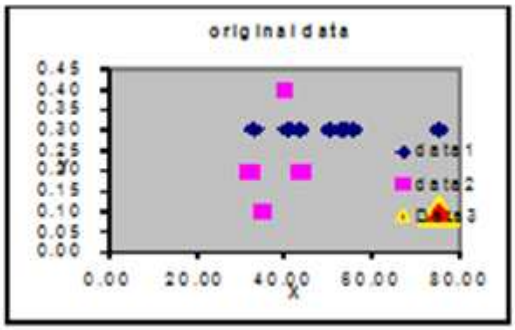


Fig. 9. Measured data set across sensor clusters.

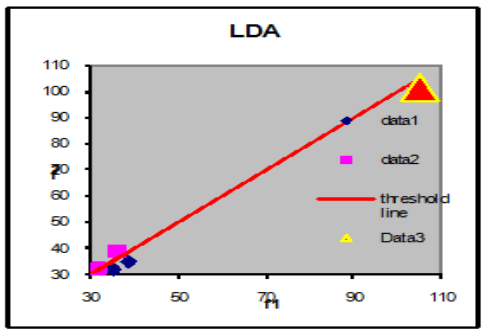


Fig. 10. Classifying local measured data with P_{MAXR} global thresholds at the central coordinator which is based on feature extraction and has unlimited processing power.

event as the lowest probability reported by a sensor in a clustered environment. In our data aggregation process we use P_{max} which typically has a higher probability of 0.3. If the sensor measured probability has a lower probability such as 0.1 then it is a highly probable event from equation (7). Now that we can have such events reported across many clusters we need to determine if it is a faulty event or an actual true event. To make it fault-tolerant we extend the same Bayesian learning method such as equations (3.1,3.2) to come with a fault scoring rule. This allows the algorithm to more fault-tolerant in a distributed network. So for this we assume that the sensors have fixed register based accuracy(8-16 bits) and we can assign a weighted score based on a probability that a expected bit is set or not set for an measurement to be accurate compared to P_{max} mask (measured value) bits. From equation (7.1) we have that given an event has been reported in a cluster, i.e. P(B).

$$I(m) = \log_2 \frac{1}{p(m)} = -\log_2(p(m)) \tag{7}$$

Then we can rewrite this as

$$P\left(\frac{A}{B}\right) = P\left(\frac{AggrValue}{Event}\right) = P\left(\frac{Value_{Aggr} \cap Event}{Event}\right) \tag{7.1}$$

From the previous definition of an event we know $P_{event} = 0.1$,so we need to find out the numerator $PValue_{aggr} \cap Event$. As the measured values are internally stored in bits, let us assume for a register with m bits, then it is equiprobable for any bit to set giving a probability of $\frac{1}{m_1}$ for every position of the bits in the register. Let us say the pattern of expected bits from a measurement has a probability p, and also an event has been reported then the fault probability of the event can be calculated by $\oplus$ each bit with the faulty bit at the positions $\frac{1}{m_1}, \frac{1}{m_2}, \ldots \frac{1}{m_3}$ the mismatched bits have a (1-p) probability. This is the intermediate fault score for each and every mis-matched bit present in the current set of measured value. Then the fault conditional probability given the expected pattern is calculated by summing up the score of the faulty bits as shown in equation

(7.2) which is then ranked to get a final fault score.

$$\begin{aligned}\text{Fault Score} = &\tfrac{1}{m_1}(1-p)faultybit(m_1)+\\ &\tfrac{1}{m_1}(1-p)faultybit(m_2)+\\ &\tfrac{1}{m_1}(1-p)faultybit(m_3)+\\ &...\tfrac{1}{m_1}(1-p)faultybit(m_n)\end{aligned} \tag{7.2}$$

In a clustering environment $P_{max} = p$, $P_{(event-local)} = 0.1$, also the average self-information that is an event is equiprobable in the whole network configured into c clusters is given by $P_{(event-global)} = \frac{1}{c}$

This allows to evenly scoring fault bits which have high significance such as MSB compared to least significant bits which have least significance. If a fault score is > 0.5 then the event is ignored otherwise it is reported even though is carries a self-information of as low as 0.1. This event detection and reporting algorithm is adaptive as it is based on, if the number of clusters in the sensor network is large then an event detected is more precise at the central coordinator as shown in equation (7) by substituting the $P_{(event-global)}value$.

TABLE III
CLASSIFIER PERFORMANCE USING P_{MAX}- P_{MAXR}-RELEVANT

Classification	Temperature Value	P_{MAX}	P_{MAXC}	P_{MAXR}
C	57.20	0.10		
C	53.60	0.10		
C	55.60	0.20	0.30	75.00
C	75.00	0.30	0.30	75.00
C	55.40	0.20		
M	40.00	0.10		
M	32.00	0.20		
M	44.00	0.30		
M	35.00	0.40	0.40	
(M)	75.00←	0.10		
Relevant	75.00		0.10	0.30

TABLE IV
CLASSIFIER PERFORMANCE USING P_{MAX}- P_{MAXR} -FAULTY

Classification	Temperature Value	P_{MAX}	P_{MAXC}	P_{MAXR}
C	57.20	0.10		
C	53.60	0.10		
C	55.40	0.20		
C	58.40	0.30	0.30	58.00
C	55.00	0.20		
M	40.00	0.10		
M	32.00	0.20		
M	44.00	0.30		
M	35.00	0.40	0.40	
(M)	75.00 ←	0.10		
Fault	35.00		0.40	0.0

VIII. SUMMARY

Theoretically we show that study of wireless sensor network energy management is a significant part of solving the reliability issue. Also we show that the energy constraint is network size invariant and converges to an optimal cluster size. To achieve a

balance we use compression algorithms to bring down the transmitted bits to known entropy at the cluster head as in equation (2.9). The same model serves as the reliability index for a central classifier which uses multi-feature and brings down the fault rate to a minimum in our case around 10%. This technique can also be shown that in training data, as the data is measured with respect to a given source entropy at each cluster head then the training data set over time for a large sensor network has the self mutual-information of all the nodes in the network with respect to the new measured values. This helps to further classify the current data set by reducing any faults due to measurements locally by using conditional probability of the new measured value given a similar measured value is reported by adjacent cluster heads. To scale a machine learning system to adapt to faulty sensors we have designed a rank based fault scoring system to classify good and bad events in a widely faulty environment and also has a precession factor calculated by the number of clusters. The classifier's design is based on the probability model and is independent of the actual measured value, some of the future work [7] extends this work into a fault-tolerant classifier for the class of distributed WSN clustering algorithms.

ACKNOWLEDGMENT

The authors would like to thank for the tools made available in the area of Data Compression and Pattern Recognition at International Institute of Information Technology, Hyderabad, India. We like to thanks Prof. Subash Mukhopadhyay from University of Massey, New Zealand for this constructive comments [9] in time that improved the quality of the paper for submission to the new sensor journal.

REFERENCES

[1] Introduction to Data compression. Khalid Sayood. 2nd Edition, Morgan Kaufmann Series in Multimedia Information and Systems (Hardcover)

[2] Software Stack Architecture for Self-Organizing Sensor Networks, Vasanth Iyer, G.Rama Murthy and M.B. Srinivas- ICST 2007, Palmerston North New Zealand.

[3] Battery drain $(http://www.techlib.com/reference/batteries.html)$

[4] Power Law math $(http://en.wikipedia.org/wiki/Power\_law)$.

[5] Environmental measurement OS for a Tiny CRF-STACK Used in Wireless Network. Vasanth Iyer, G.Rama Murthy and M.B. Srinivas- Sensors & Transducers Journal-April 2008, ISSN 1726-5479 2006 by IFSA.

[6] Distributed Wireless Sensor Network Architecture: Fuzzy Logic based Sensor Fusion, G.Rama Murthy, Vasanth Iyer. ISBN 978-80-7368-387-0 pages 71-78 VOL II, Proceedings of the 5th EUSFLAT Conference, Ostrava, Czech Republic, 2007.

[7] Min Loading Max Reusability Fusion Classifiers for Sensor Data Model, Vasanth Iyer, G.Rama Murthy and M.B. Srinivas, 2008 IEEE SENSORCOMM, August 25, 2008 - Cap Esterel, France.

[8] B.Krishnamachari, S.S. Iyengar, IEEE.,Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in WSN, MARCH 2004.

[9] Vasanth Iyer, Garimella Rammurthy and M.B. Srinivas., Training Data Compression Algorithms and Reliability in Large Wireless Sensor Networks. IEEE International Workshop on Embedded Processors, Sensors, and Actuators (EPSA-2008) to be held in Taichung, Taiwan, June 11-13, 2008.