A Distributed Bayesian Approach to Fault Detection in Sensor Networks

G. Lo Re, F. Milazzo, M. Ortolani

In Proceedings of the IEEE Global Telecommunications Conference (GlobeCom), 2012, pp. 634-639

A Distributed Bayesian Approach to Fault Detection in Sensor Networks

Giuseppe Lo Re, Fabrizio Milazzo, and Marco Ortolani
Università degli Studi di Palermo
Viale delle Scienze ed. 6, Palermo, Italy
first.lastname@unipa.it

***Abstract*—Sensor networks are widely used in industrial and academic applications as the pervasive sensing module of an intelligent system. Sensor nodes may occasionally produce incorrect measurements due to battery depletion, dust on the sensor, manumissions and other causes. The aim of this paper is to develop a distributed Bayesian fault detection algorithm that classifies measurements coming from the network as corrupted or not. The computational complexity is polynomial so the algorithm scales well with the size of the network. We tested the approach on a synthetic dataset and obtained significant results in terms of correctly labeled measurements.**

I. Introduction

Sensor networks are an emerging technology in ICT, thanks to their high versatility as an underlying infrastructure for environmental control systems [1]. They are widely used as a sensing module for applications like environmental monitoring, ambient intelligence and military, which require the perception of some physical quantities, such as temperature, humidity, ambient lighting, and so on.

A sensor network consists of a large number of sensor nodes which, in their turn, are made up of four main components: (i) transceiver, (ii) power unit, (iii) sensors array, and (iv) processing unit. Malfunctioning sensor nodes provide erroneous information which is likely to negatively affect the QoS of the overall system; a fundamental issue to be addressed is thus *fault detection*. Literature classifies faults in sensor networks as depending on: crashed nodes, link errors, and corrupted measurements [2]; in our work, we will specifically focus on the problem of fault detection of corrupted measurements.

There exist many causes that could negatively affect measurements; in particular, according to the *system-centric* view of a fault [3], the fundamental causes of corruption are:

- *calibration faults*: the ADC converter has a specific input-output curve; however it is known that such curve changes over time, which results into a calibration fault;
- *connection/hardware faults*: example of such fault is a short circuit caused by a spill over the mainboard;
- *low battery*: this typically causes noisy and stuck-at (a constant value) readings.

In the context of sensor networks, a measurement is deemed as corrupted if it significantly deviates from the *actual* behavior of the monitored quantity; indeed we cannot assume the ground truth to be known in advance, so we are forced to rearrange the definition as "a measurement that significantly deviates from the *expected* behavior of the monitored quantity".

Any fault detection approach should be analyzed both under the architectural and methodological viewpoint. From the architectural point of view, we can easily identify two opposite approaches:

- *centralized*: a fusion center gathers all sensor measurements, and then chooses a subset of all those that can be considered reliable, which are eventually sent to a base station. The remaining measurements are discarded;
- *distributed*: sensor nodes exchange their measurements with nearby nodes, and, by following a distributed agreement protocol, they infer which nodes sent corrupted measurements.

From the methodological point of view, the most common approaches in literature are:

- *majority voting scheme* [4]: each node expresses an opinion about a certain measurement from another node. Its implementation is straightforward, but it works well only if every sensor node has many neighbors;
- *threshold methods* [5], [6]: the whole set of measurements is partitioned into those similar to each other, and outliers. Such methods potentially outperform any other method, but the performances are very sensitive to the chosen threshold value;
- *Bayesian methods* [7]: they take into account background information such as prior probability of a sensor fault; they usually classify a measurement as corrupted or not by maximizing a given likelihood function. Such methods are not sensitive to the parameters choice in that a learning phase automatically provides the optimal ones; their major drawback consists in the exponential growth in terms of computational resources, which is typically addressed by resorting to approximated inference approaches.

The main contribution of the present work is the design of a novel distributed Bayesian method for detecting faults in sensor networks, in order to label the collected measurements as corrupted or not. In our approach, each sensor node manages a portion of the Bayesian network and is allowed to decide whether to cooperate with its neighboring nodes or not. If the node opts for cooperation, it receives information about the measurements in its proximity, thus widening its scope with respect to the surrounding environment; intuitively, we expect that such behavior increases the chance of correct classification of the sensed measurements. If the node does not cooperate with its neighborhood, it will rely only on its local information to perform classification; for instance, such behavior can occur whenever the node needs to limit its energy consumption, or when it notices that its transceiver component is not properly working.

The approach is not approximated in the sense that it is able to find the exact solution by maximizing a likelihood function, while maintaining polynomial complexity; in particular, we will show that our approach is characterized by: *linear* message complexity with respect to the number of network nodes, and *polynomial* time complexity with respect to the number of different fault types.

The remainder of the paper is structured as follows: Section II reviews related approaches and provides a brief comparison with our work. Section III mathematically describes the main aspects of the proposed approach. Sections IV and V contain the theoretical and experimental assessment by analyzing both complexity and performances of the algorithm we devised. Finally Section VI presents some concluding remarks.

II. Related Works

Fault detection is a deeply studied topic in the context of sensor networks, and related literature can be broadly subdivided into works dealing with: (i) the definition of the failure model, and (ii) definition of the detection method.

The authors of [3] propose a failure model classifying faulty sensor readings as *outlier* (a measurement that significantly deviates from other sensor readings), *spike* (a change rate much greater than expected over a short period), *stuck-at* (a series of values with zero variations), *noise* (a series of data with a variance greater than expected). The authors of [8] propose a similar taxonomy, with just three different type of faults: *short*, *noise*, and *constant*, where, unlike the previous work, the *short* fault is regarded as a generalization of *outlier* and *spike*. Without loss of generality, in the remainder of the paper we will rely on the latter taxonomy, since the computational complexity of our method is dependent on the number of different faults it can recognize.

Various detection methods have also been proposed in recent literature. The work [9] proposes a centralized detection method where the fusion center solves n different non-linear systems of equations to discover nodes that negatively affect the solution. The two main drawbacks of this approach consist in the fact that the accuracy of the solutions depends on the method used for solving the non-linear system, and the method cannot be implemented in a distributed way, unless every sensor node behaves as a fusion center. The authors of [7] propose a centralized MAP Bayesian method that assigns the most probable class (i.e. faulty or not) to each sensor reading by maximizing a likelihood function. The approach achieves low misclassification rate but it is computationally expensive in that it is exponential in the number of sensor readings that are to be classified. An interesting distributed method was presented in [4]. The method is computationally simple but achieves good classification rates only with a high number of neighbors per node (at least 15); moreover, the classification task depends on the value of two predefined thresholds that can be only experimentally derived. Finally the work [5] is a distributed detection method using spatial correlation among readings to detect corrupted measurements; if the last reading of a sensor node alters the value of the correlation more than a specified threshold, then the reading is labeled as corrupted. The method achieves good performance with an average number of twenty neighbors per node; its applicability is also hindered by the reliance on the threshold that can be only experimentally derived.

The proposal we describe in the present paper consists in a distributed and Bayesian method. Unlike centralized approaches, it does not require the presence of a fusion center and any node is free to choose a cooperative or non cooperative behavior. Thanks to the use of a Bayesian network, the choice of parameters depends on the computation of conditional probability tables, which has analytical solution (unlike threshold methods). We will show that our method achieves good classification performance if network nodes behave cooperatively (i.e. when they do exchange information on their measurements). It also achieves acceptable classification performance when sensor nodes behave non-cooperatively, e.g. when they have no neighbor node.

III. Problem Statement

Let us assume that a number of nodes deployed within the sensor field obtain readings about environmental quantities, such as temperature, humidity, or light exposure, and need to check if such readings are corrupted or not. Sensor nodes may try to classify their readings without communicating with nearby nodes by only using local knowledge (such as past readings, predictive models and so on); however they could also exchange sensed measurements in order to obtain additional valuable information about the physical quantity, thus increasing the probability that their own measurements are correctly classified.

We assume here that the sensor field covers a relatively narrow area so that the measurements sensed by different sensor nodes are likely to be correlated; in other terms we are assuming that the variations due to the nature of the sensed physical quantity are relatively small with respect to nearby nodes.

In order to implement such behavior, we chose to devise a distributed Bayesian network with the following key properties: (i) it must allow sensor nodes to freely select their cooperative vs. non cooperative behavior and (ii) it must be able to classify sensed measurements as corrupted or not, in a distributed fashion. In the following, we analyze our Bayesian network model by specifying its graphical representation, the training phase for learning the network parameters, and the distributed inference algorithm run by sensor nodes in order to classify their measurements. The mathematical notation we will use in the rest of the paper is summarized in Table I.

TABLE I
MATHEMATICAL NOTATION FOR THE BAYESIAN FRAMEWORK.

Symbol	Description
$N(i)$	Neighborhood set of node i
m_i	measurements sensed by node i
L_i	local features set of node i
L_i^k	k-th component of L_i
$S_{i,j}$	shared features set of node i and j
$S_{i,j}^k$	k-th component of $S_{i,j}$
X	The whole evidence of the Bayesian network
c_i	The class assigned to the measurement of sensor i
C	The set of possible assignment for c_i

A. Graphical Representation

The distributed Bayesian network is split up into smaller units, contained within sensor nodes. Such smaller networks consists of one hidden (discrete) random variable taking the most probable state of the sensed measurement, and some observed (either discrete or continuous) random variables, also called *features*, that are used to infer the hidden variable value. Such Bayesian structures are typically called *Naive Bayes* [10].

Sensor node i tries to estimate hidden variable c_i by indicating whether measurement m_i is faulty, and in such a case which specific fault has occurred. Variable c_i takes values in the set $C = \{\texttt{short}, \texttt{noise}, \texttt{constant}, \texttt{non-faulty}\}$. The observed variables are intended to capture statistical properties of the measurements, and are used to check whether the measurement is corrupted or not. The authors of [3] propose a standard set of features to be used when performing fault measurements detection:

- *mean*, *variance*, and *correlation*, useful to obtain a regression model and to make predictions;
- *gradient*, which measures the change rate of different time scales;
- *distance from other readings*, useful to consider information from the neighborhood;
- *node residual energy*, useful to check if the sensor has a sufficient amount of energy left before malfunctioning.

Clearly, the above set is not comprehensive and other sophisticated features can be used; however we found that they are widely sufficient for the purpose of this work.

Such features are the evidence of a normal/abnormal value of the sensed measurements. Consider for example a `short` measurement (i.e. an out-of-range value), corresponding to an unexpected value for the gradient feature. From a causal point of view, we deduce that the (hidden) class of a measurement influences the state of the (observed) features; such causal relation is indicated in a Bayesian context by an arrow starting from the influencing variable and pointing at the influenced ones.

In principle, some features (e.g. mean and variance) can be computed locally, while others (distance from other readings) necessarily require communication among sensor nodes; in our framework, this means that, in general, features will be influenced by a variable number of hidden variables. Features that do not force a node to require communication with other sensor nodes will be called *local* features, while in the opposite

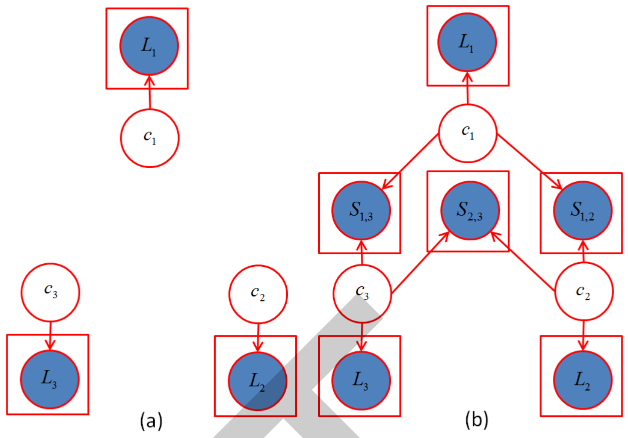

Fig. 1. (a) Three isolated naive Bayes classifiers and (b) the same Naive Bayes connected through shared features (b). We make use of the *plate notation* to denote array of variables (local and shared) enclosed within the boxes.

case they will be called *shared* features. Local features for node i are thus functions of the type $L(m_i)$, while shared features are functions of readings coming from more sensor nodes: $S(m_1, m_2, ..., m_n)$, where the subscript identifies the originating sensor node. As previously said, any node is allowed to decide whether to perform its classification task with or without exchanging information with nearby nodes; the key property of the Bayesian model we described is that such decision corresponds to the operation of adding or removing links toward shared features. In order to ensure convergence of the classification task, we will allow only pairwise interactions among sensor nodes, so shared features will be restricted to functions of the type $S(m_i, m_j)$. Figure 1 shows the resulting distributed Bayesian network in the cases of 3 nodes, in the cooperative and non-cooperative cases, respectively. White circles in the picture represent the hidden variables, while shadowed circles represent the observed ones. If nodes do not cooperate (case a), the whole Bayesian network is just made up of independent naive Bayes classifiers; on the contrary, if nodes cooperate (case b), the whole structure will be much more complex, in that naive Bayes classifiers are connected through shared features.

B. Learning parameters

The learning procedure consists in computing the joint probability for the whole Bayesian network; the joint probability of any Bayesian network is defined as:

$$p(x_1, x_2, ..., x_n) = \prod_i p(x_i | pa(x_i)), \tag{1}$$

where $pa(x_i)$ denotes the set of parent variables for x_i.

Our Bayesian network model comprises two types of variables: hidden (c_i, with no parents), and observed (L_i, with one parent, and $S_{i,j}$, with two parents). By particularizing the above equation to our Bayesian network we easily obtain the joint probability:

$$p(c_1, .., c_n, X) = \prod_{\substack{i,\\ j \in N(i): j > i}} p(L_i | c_i) p(S_{i,j} | c_i, c_j) p(c_i), \tag{2}$$

where X denotes the entire set of feature variables within the Bayesian network.

The above equation shows that the parameters to be learned are the prior probabilities of the class label assignments and the conditional probabilities for local and shared features.

In order to simplify the computation of conditional probabilities, we can exploit conditional independence thus obtaining:

$$\begin{aligned} p(L_i|c_i) &= \prod_k p(L_i^k|c_i) \\ p(S_{i,j}|c_i(t), c_j) &= \prod_k p(S_{i,j}^k|c_i, c_j) \end{aligned} \tag{3}$$

In our work we assume we can use *supervised* learning with a training set made by a fixed amount of observed features with the respective actual label assignments. The computation of conditional probabilities $p(L_i^k|c_i)$, $p(S_{i,j}^k|c_i, c_j)$, and prior probabilities $p(c_i)$ is therefore carried out by using the frequentist approach.

C. Distributed inference algorithm

The inference algorithm we implemented allows sensor nodes to compute the classes to be assigned to the sensed measurements. It computes the output class labels $(c_1^{opt}, ..., c_n^{opt})$ by maximizing the joint probability of the whole distributed Bayesian network, given the evidence, as follows:

$$\begin{aligned} (c_1^{opt}, ..., c_n^{opt}) =& \underset{c_1,...,c_n}{argmax}\ p(c_1, ..., c_n|X) = \\ = \underset{c_1,...,c_n}{argmax} & \prod_{\substack{i,\\ j\in N(i):j>i}} p(L_i|c_i)p(S_{i,j}|c_i, c_j)p(c_i) \end{aligned} \tag{4}$$

It is worth pointing out that such formulation of the problem corresponds to a MAP approach and it is quite similar to the work [7]; however we will show that our algorithm computes the class labels without using a mere brute force approach, whose time complexity would be $O(|C|^n)$; rather, by exploiting factorization properties of the distributed Bayesian network, we are able to compute the optimal class labels with time complexity proportional to $|C|^2$.

Without loss of generality, we will assume in the following that every sensor node uses a cooperative approach; the case of non cooperative nodes can be obtained by canceling all terms in the above equation containing shared features terms.

Our distributed inference algorithm is based on "sum-product" [11], which is an inference message passing algorithm that aims to compute optimal values of hidden variables in a Bayesian network. However, its specification is provided in terms of messages exchanged among Bayesian variables; since the purpose of our work is to design a distributed algorithm for sensor networks, instead, we rearranged the original mathematical framework in a way that the procedure is defined in terms of messages exchanged among network nodes. The convergence of "sum-product" algorithm is guaranteed only if the Bayesian network is free of loops, so we restricted the interactions among sensor nodes to be pairwise; such assumption will also constrain the communication topology of the wireless sensor network to a tree. Note that such restriction is not unrealistic in a real deployment; the issue can be easily addressed by building a spanning tree for the WSN [12], [13].

Our algorithm is divided into two phases: during the first phase each node i computes local and shared features by exchanging the sensed measurement with its neighboring nodes. The second phase is the inference message passing algorithm that implements a sort of convergecast-broadcast.

1) Phase one: local features are functions of the type: $L_i^k = f^k(m_i)$ and do not require any exchange of messages among network nodes. Shared features are functions of the type: $S_{i,j}^k = f^k(m_i, m_j)$, $j \in N(i)$ and require the knowledge of measurements in the neighborhood set of node i.

For this reasons the only message that (cooperative) nodes need to exchange, is their own:

$$\mu_{i\to j} = m_i \text{ and } j \in N(i), \tag{5}$$

If any node i chooses not to cooperate with its neighborhood nodes it simply does not forward its measurement; such an operation corresponds to the removal of any observed variable $S_{i,j}$ from the distributed Bayesian network.

2) Phase two: this phase is composed of two rounds: convergecast started at leaf nodes, and broadcast started at root node. The convergecast round makes any sensor node aware of the probability of its class label assignments given the information of their neighbors, while the broadcast round actually computes the most probable class to be assigned to each sensor node.

Before introducing the convergecast message $\mu_{i\to j}(c_j)$, let us define the following pairwise belief:

$$\phi(c_i, c_j) = p(c_i)p(L_i|c_i)p(S_{i,j}|c_i, c_j) \prod_{z\in N(i)/j} \mu_{z\to i}(c_i). \tag{6}$$

Such quantity represents the belief of node i with respect to node j about the pair of class labels to be assigned to both nodes. Note that node i can compute the above quantity only after receiving messages $\mu_{z\to i}(c_i)$ from all its children.

The convergecast round message (children-to-parent) is then the unary belief:

$$\mu_{i\to j}(c_j) = \max_{c_i} \phi(c_i, c_j). \tag{7}$$

Such quantity is a marginalized version of the previous equation 6 and represents the belief of node i with respect to node j about the single class label to be assigned to node j; also note that such quantity is not defined for the root node since the convergecast ends on that node.

Convergecast ends at root node that computes its optimal class label assignment as:

$$c_r^{opt} = \underset{c_r}{argmax}\ p(c_r)p(L_r|c_r) \prod_{z\in N(r)} \mu_{z\to r}(c_r) \tag{8}$$

All the quantities $p(c_i)$, $p(L_i|c_i)$ and $p(S_{i,j}|c_i, c_j)$ are computed by using a simple table look-up since they have been previously stored during the learning of conditional probability

TABLE II
NOTATION USED IN THE ALGORITHM COMPLEXITY SECTION.

Symbol	Description
h	The depth of the tree
n	The number of nodes of the tree
D	The maximum number of children of any sensor node
k_l	The number of local features of any sensor node
k_s	The number of shared features of any sensor node
$\|C\|$	The number of possible assignment for c_i
$\|Z\|$	The maximum number of discrete values a feature can take

tables.
Before sending the convergecast message, any node i, only needs to store the quantity $\phi(c_i, c_j)$ that will be used during the broadcast round.

The broadcast round message (parent-to-children) from parent node j to its children i is:

$$\mu_{j \to i} = c_j^{opt} \tag{9}$$

Finally, any node i (other than the root) computes its optimal class assignment as:

$$c_i^{opt} = \underset{c_i}{argmax}\ \phi(c_i, c_j^{opt}), \tag{10}$$

where $\phi(c_i, c_j)$ was stored during convergecast round.

IV. Theoretical Assessment

The aim of this section is to prove that the algorithm scales well with the size of the network. In the following we will refer to the mathematical notation contained in Table II.

Theorem 1: The algorithm has memory complexity:

$$O(k_f|C|^2|Z|) \tag{11}$$

Proof: The learning task stores a conditional probability table for every feature (local and shared) and a prior table for the class label. Let us suppose any node has at most k_l local features and at most k_s shared features; additionally we define $k_f = max(k_l, k_s)$

The required memory for such tables are bounded by: $k_f|C||Z|$ for local features, $k_f|C|^2|Z|$ for shared features and $|C|$ for priors; by summing up the above terms, we can easily see that the overall required memory is $O(k_f|C|^2|Z|)$.

Theorem 2: The algorithm has time complexity:

$$O(h|C|^2(2k_f + D)) \tag{12}$$

Proof: Phase one involves a simple message exchange among neighbor nodes, and does not involve any computation.

Phase two is a convergecast-broadcast procedure. The convergecast message computation involves mono-dimensional maximizations (equation 7). To perform such operation the algorithm computes all the entries of the matrix $\phi(c_i, c_j)$ that are $|C|^2$.

The computation of a single value of $\phi(c_i, c_j)$ is a multiplication among $1 + 2k_f + (D - 1)$ terms (see equation 6), that amounts to $2k_f + D - 1$ total multiplications. Time complexity for such round thus is $|C|^2(2k_f + D - 1)$.

During broadcast round the optimal class label assignment (see equation 10) is computed by a table lookup on $\phi(c_i, c_j)$, that requires $|C|$ operations. The time complexity for such round thus is simply $|C|$.

By summing all terms for phase one and phase two, and considering that tree depth is h we obtain the whole time complexity of the algorithm: $O(h|C|^2(2k_f + D))$, where we discarded all irrelevant terms.

Theorem 3: The algorithm has message complexity $O(n)$.

Proof: Phase one involves sending measurements. In this case two messages will travel on every tree link, that amounts to $2(n-1)$ messages. Phase two is a convergecast-broadcast of messages involving also $2(n-1)$ messages.

It is easy to see that the whole amount of messages exchanged by the algorithm is $O(n)$.

V. Experimental Assessment

The aim of this section is to evaluate the classification performance of our distributed Bayesian network. We computed false rejection ratio (FRR) and false acceptance ratio (FAR) for different corrupted samples percentage while nodes behaved cooperatively or not.

We conducted experiments onto ten simulated sensor nodes using BRML Matlab toolbox [14]. The dataset is made of temperature readings sensed at intervals of 30 seconds for six days (2880 readings per day); since sensor nodes are simulated, we used a real Mica2Dot sensor node as measurements generator. Each of the ten simulated sensor nodes sampled measurements from the real sensor node, and then added a small Gaussian noise $\mathcal{N}(0, \sigma_E^2)$ to simulate small environmental fluctuations among nearby sensor node (we set $\sigma_E^2 = 0.1$). Every measurement coming from such (clean) dataset was thus labeled as `non-faulty`. The successive step consisted in randomly injecting faults of the type: `short`, `noise` and `constant`; in the following we will indicate as $m(t)$ the clean measurement, and $\tilde{m}(t)$ the corrupted measurement.

The corruption formulas we used were [8]:

- `Short`: $\tilde{m}(t) = m(t) + g \times m(t)$
- `Noise`: $\tilde{m}(t_0 + k) = m(t_0 + k) + \mathcal{N}(0, \sigma_N^2)$
- `Constant`: $\tilde{m}(t_0 + k) = m(t_0)$,

where g is a gain constant, $k \in \{0, 1, 2, ..., K\}$ and K is the duration of the fault, t_0 is a random starting time instant, and $\mathcal{N}(0, \sigma_N^2)$ is a Gaussian noise with zero mean and σ_N^2 variance (note that $\sigma_N^2 >> \sigma_E^2$).

We conducted experiments by considering different amounts of nodes behaving cooperatively and different amounts of corrupted readings; we varied the percentage of cooperating nodes from 0 to 100 with 25 percent increments, while we varied the percentage of corrupted samples from 0 to 25 with 5 percent increments; finally, we used the first day of readings as training set. Parameters setting for all the experiments were: $g = 1$, $\sigma_N^2 = 3$, $K = 100$.

The features we used were:

- Gradient: $L_i^1(t) = m_i(t) - m_i(t-1)$
- Number of measurements without changes: $L_i^2(t) = \sum_{t=t-k} \mathbb{I}[m(t) = m(t-1)]$

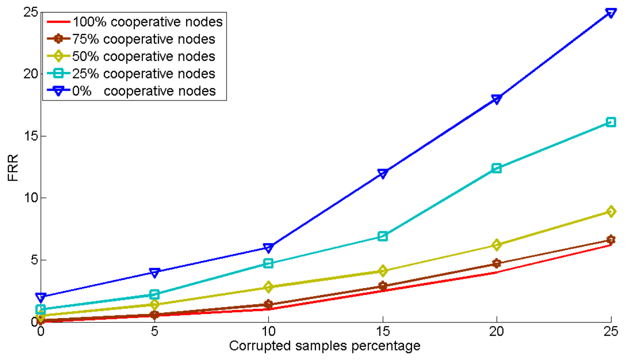

Fig. 2. The picture shows false rejection ratio for different amounts of corrupted samples and different amounts of cooperative nodes.

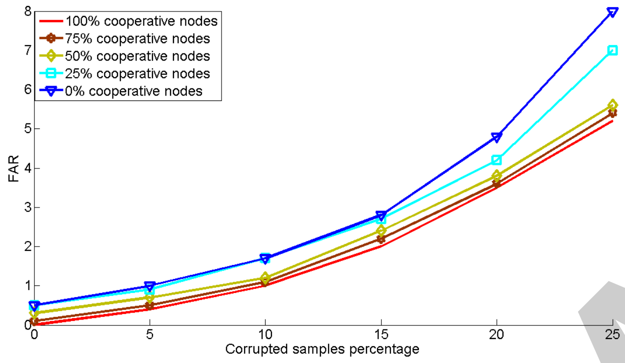

Fig. 3. The picture shows false acceptance ratio for different amounts of corrupted samples and different amounts of cooperative nodes.

- Distance from other readings: $S^1_{i,j}(t) = m_i(t) - m_j(t)$,

where $\mathbb{I}[\cdot]$ is the indicator function.

Classification results consist in FRR and FAR (figures 2 and 3) for the different amounts of cooperating nodes and corrupted measurements.

When all nodes behave cooperatively, the algorithm achieves very low false negative/positive rates; in particular FRR and FAR are consistently below 10 percent, even for an amount of 25 percent of corrupted samples; on the contrary when no node behaves cooperatively we can see that false negative rates and false positive rates increase because of the lack of shared features; however, also in such extreme case we obtained acceptable performance. The algorithm achieves better performance when more sensor nodes cooperate because they take into account dependence among hidden variables, unlike the non cooperative case where all hidden variables are considered independent of each other.

VI. Conclusions

We developed a cooperative distributed algorithm using a probabilistic approach to detect corrupted measurements within sensor networks. The algorithm works using a distributed Bayesian network where hidden variables assume the most probable class (faulty or not) explaining observed data. Theoretical results have shown that unlike other works in literature, the algorithm scales well in time, message and memory complexity. Experimental results were carried out for different amounts of corrupted samples and cooperating nodes. We noticed that when nodes behave cooperatively the inference algorithm achieves classification results better than in the opposite case; the empirical explanation was that in the latter case hidden variables are considered independent of each other, whereas in the former case they are kept dependent of each other by considering relationship with shared features.

We carried out supervised training onto a dataset containing the actual class label assignments for a fixed amount of time (one day of measurements); future works include the possibility of working without labeled training set, performing unsupervised training and by using Expectation Maximization algorithm to compute conditional and prior probabilities.

Further investigation is needed to allow our method to cover wider sensor field; in particular, a future work may consist of make sensor nodes able to learn the most probable Bayesian structure they should adopt independently from the size of the sensor field; this could be achieved by implementing Bayesian Networks Structure Learning methods.

References

[1] D. Culler, D. Estrin, and M. Srivastava, “Guest editors’ introduction: overview of sensor networks,” *Computer*, pp. 41–49, 2004.

[2] L. Paradis and Q. Han, “A survey of fault management in wireless sensor networks,” *Journal of Network and Systems Management*, vol. 15, no. 2, pp. 171–190, 2007.

[3] K. Ni, N. Ramanathan, M. N. H. Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen, and M. Srivastava, “Sensor network data fault types,” *ACM Transactions on Sensor Networks*, vol. 5, no. 3, 2009.

[4] J. Chen, S. Kher, and A. Somani, “Distributed fault detection of wireless sensor networks,” in *Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*, 2006, pp. 65–72.

[5] W. Wu, X. Cheng, M. Ding, K. Xing, F. Liu, and P. Deng, “Localized outlying and boundary data detection in sensor networks,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 8, pp. 1145–1157, 2007.

[6] N. Giatrakos, Y. Kotidis, A. Deligiannakis, and Vasilis, “In-network approximate computation of outliers with quality guarantees,” *Information Systems*, 2011.

[7] K. Ni and G. Pottie, “Bayesian selection of non-faulty sensors,” in *IEEE International Symposium on Information Theory, 2007*, 2007, pp. 616–620.

[8] A. B. Sharma, L. Golubchik, and R. Govindan, “Sensor faults: Detection methods and prevalence in real-world datasets,” *ACM Transactions on Sensor Networks*, vol. 6, no. 3, p. 23, 2010.

[9] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli, “On-line fault detection of sensor measurements,” in *Sensors, 2003. Proceedings of IEEE*, vol. 2, 2003, pp. 974–979.

[10] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[11] F. Kschischang, B. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[12] S. Hussain and O. Islam, “An energy efficient spanning tree based multi-hop routing in wireless sensor networks,” in *IEEE Wireless Communications and Networking Conference.*, 2007, pp. 4383 –4388.

[13] S. Upadhyayula and S. Gupta, “Spanning tree based algorithms for low latency and energy efficient data aggregation enhanced convergecast (dac) in wireless sensor networks,” *Ad Hoc Networks*, vol. 5, no. 5, pp. 626 – 648, 2007.

[14] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.