# Robust Anomaly Detection for Large-Scale Sensor Data

Aniket Chakrabarti[*]
The Ohio State University
chakrabarti.14@osu.edu

Manish Marwah
Hewlett Packard Labs
manish.marwah@hpe.com

Martin Arlitt
Hewlett Packard Labs
martin.arlitt@hpe.com

## ABSTRACT

Large scale sensor networks are ubiquitous nowadays. An important objective of deploying sensors is to detect anomalies in the monitored system or infrastructure, which allows remedial measures to be taken to prevent failures, inefficiencies, and security breaches. Most existing sensor anomaly detection methods are local, i.e., they do not capture the global dependency structure of the sensors, nor do they perform well in the presence of missing or erroneous data. In this paper, we propose an anomaly detection technique for large scale sensor data that leverages relationships between sensors to improve robustness even when data is missing or erroneous. We develop a probabilistic graphical model-based global outlier detection technique that represents a sensor network as a pairwise Markov Random Field and uses graphical model inference to detect anomalies. We show our model is more robust than local models, and detects anomalies with 90% accuracy even when 50% of sensors are erroneous. We also build a synthetic graphical model generator that preserves statistical properties of a real data set to test our outlier detection technique at scale.

## CCS Concepts

**•Computing methodologies → Anomaly detection; •Information systems →** *Data mining;*

## Keywords

Anomaly detection; sensors; graphical models; belief propagation

[*]Work done during internship at Hewlett Packard Labs.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

*BuildSys '16, November 16-17, 2016, Palo Alto, CA, USA*
© 2016 ACM. ISBN 978-1-4503-4264-3/16/11...$15.00
DOI: http://dx.doi.org/10.1145/2993422.2993583

## 1. INTRODUCTION

Sensors have become commoditized, and the resulting decreases in cost have led to their deployment in large numbers. The data collected from these sensors is processed and analyzed for monitoring and managing infrastructure and resources. Anomaly detection techniques applied to the collected data allow failures and degradations to be quickly identified, enabling energy savings and more efficient consumption of resources. Some examples where sensors in large numbers are being increasingly used include: 1) Smart campuses and buildings, where sensors measure air temperature, humidity, building occupancy, lighting, etc., to manage heating, ventilation, air conditioning (HVAC) and lighting systems, and to optimize use of resources such as electricity, gas and water; 2) the transportation industry, where, for instance, vehicle fleet management is aided by continuous input from sensors attached to the vehicles. This helps in vehicle maintenance, route optimization, driver safety, etc.; 3) utility infrastructure, where sensors monitor power and water infrastructure, for efficient resource management; 4) weather modeling and forecasting, where data from a large number of distributed sensors measuring quantities like temperature, humidity, precipitation, wind speed, etc., is used for short and long term weather prediction, and aids with critical tasks such as disaster management planning, route optimization for air, ground and sea transportation.

In all the above applications, outlier or anomaly detection in the sensor data is an important task as it helps to identify abnormal conditions that may result from a failure (or impending failure), misconfiguration or malicious activity. The most commonly used outlier detection methods are threshold-based. However, a large category of anomalous behavior does not manifest as a violation of a threshold. These are addressed through model-based outlier detection techniques that build statistical models of sensors [30], typically using historical data. Most anomaly detection models have the following shortcomings — 1) the inputs to the model are local, that is, even when a large number of sensors are deployed, the global structure and dependencies between these sensors are not leveraged; 2) the models are not very robust, that is, if the input to the models is erroneous or corrupted, they typically fail to function correctly; and finally, 3) they do not perform well when there is a large amount of missing data.

In this paper we develop a technique, based on probabilistic graphical models, which utilizes the global dependency structure among the sensors to detect outliers. The resulting model is much more robust in the presence of erroneous data and missing sensor values than local models. Our method consists of creating a pairwise Markov Random Field (MRF) model from the sensor network, where the graph structure is learned from historical data. While most applications of MRF [4, 26, 11] use heuristics and domain knowledge to determine the edge potentials, we learn edge potentials from historical data. We design the MRF topology such that it is very amenable to running the belief propagation inference algorithm on it. We develop a bipartite graphical model that can reduce the number of active trails in the MRF, making belief propagation converge faster and to more accurate results.

While our method applies to any of the application areas where sensors exhibit spatial dependency, including sensor deployments in buildings [2], we demonstrate its effectiveness on geographically distributed temperature sensors, mainly due to data availability. We apply our method on data collected from 5,288 weather stations in California, and compare against several baseline methods, based on local data, and others based on the immediate neighborhood of a sensor. Results indicate that our method outperforms the baseline methods. Both in case of anomaly detection and missing sensor value prediction, our method improves the accuracy as much as 16% compared to the best baseline. Furthermore, we synthetically scaled the graphical model up to 500,000 nodes and nearly 12 million edges and observed that our technique scales almost linearly with the number of cores. Lastly, we generated an even larger graphical model with 5 million nodes and 120 million edges and ran our algorithm on a single configuration to show our method can handle a very large scale sensor deployment.

Specifically, the contributions of this paper are as follows:

- We present a robust method for anomaly detection, using graphical models and belief propagation-based sensor state inference.
- We show how sensor data can be modeled as an MRF, where factors are naturally learned from data (rather than based on experience of domain experts), and the graphical model topology facilitates inference.
- To evaluate performance at scale, we build a synthetic MRF generator that can generate arbitrarily large graphical models while preserving statistical properties of a MRF based on real sensor data.
- Results show that our method can predict missing values with 90% accuracy even when 75% of the data is missing; similarly, it can correctly identify 90% of outliers even when 50% of the sensors are anomalous.

The remainder of the paper is organized as follows. Section 2 introduces background information and related work. Section 3 describes our methodology, including the inference algorithm, and the construction of the graphical model from sensor data. Section 4 explains our sensor data collection and synthetic data generation process. Section 5 provides our experimental results. Lastly, Section 6 summarizes our work.

## 2. BACKGROUND AND RELATED WORK

### 2.1 Outlier detection

According to Hawkins, "An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism." [9], Typically, it is assumed that data is generated by some underlying statistical distribution and the data points that deviate from that distribution are called outliers. Outlier or anomaly detection techniques are widely used in applications such as fraud detection in financial transactions, email spam detection and medical symptoms outlier detection. A survey of anomaly detection techniques is presented in [3].

In this work we focus on identifying faults in sensor observations in a large-scale sensor network. [30] provides a thorough overview of outlier detection methods in sensor network data. According to this survey, the types of outlier detection methods used on sensor data include *statistical-based, nearest neighbor-based, cluster-based, classification-based and spectral decomposition-based techniques.* The outlier detection model we propose falls under the broad category of statistical-based techniques, where the sensor data generation is assumed to follow some statistical distribution and any observation that does not fit in that distribution is an outlier. The challenge is to learn the distribution of the sensor data. Local techniques require the learning of data distribution of sensors in isolation. They are only able to capture the temporal dependency in observations, while global techniques try to learn the joint distribution of the sensor data and hence are able to capture the spatial dependency as well. More precisely, our outlier detection model is based on the underlying principles of *statistical relation learning* [7], as we describe in the next section. Outlier detection models may be both supervised or unsupervised. For instance *knn* based outlier detection techniques [21] are the most popular unsupervised methods, while classification based outlier detection methods include SVMs [10], Random Forests [29] and others. Outlier detection models can be further classified into parametric and non-parametric methods. Popular parametric models include learning Gaussian-based models and doing hypothesis tests on new data [14], while non-parametric techniques include histogram learning and kernel density estimators [23]. The drawback of parametric models is that the model assumptions (such as Gaussian distributions) may not be realistic. Also, non-parametric models are easier to generalize. *Our model falls into the category of unsupervised and non-parametric method.* We do have a training phase, however, we do not require class labels, i.e., we do not need to know whether the sensor observations are outliers or not. Our method uses a histogram based approach.

## 2.2 Statistical Relational Learning

Statistical relational learning [7] (SRL) involves prediction and inferences in the case where the data samples obtained are not independently and identically distributed. In most real world applications, there is a complex structural relationship among the individual variables of the system and there is uncertainty in the relationship as well. SRL formalizes the structure of the relationship and accounts for the uncertainty through probabilistic graphical models. The end application such as outlier detection, link prediction, topic modeling etc. usually requires inference on the graphical model. Our problem of detecting anomalies in sensor observations fits nicely in the SRL model. Since properties such as temperatures of close by regions are related, there are strong spatial dependencies among the sensor observations. There will be temporal dependencies for observations on a single sensor as well, however this can be studied in isolation using a local model. In this paper we model the global spatial dependency in the sensor network using a probabilistic graphical model and the outlier detection involves doing inference on the model.

One of the main challenges in SRL is designing the graphical model. The graph needs to capture the dependencies accurately, while not being excessively complex to make inference hard. This trade-off between accurately and cost of inference needs to be carefully made based on the requirements of the application. The two most popular graphical models are Bayesian networks and Markov networks (Markov random field). We choose a Markov network as it is undirected and there is no directionality of dependence in the sensor observations. Bayesian networks are appropriate in relations resulting from causal dependency. The most common methods of doing inference on Markov networks are (i) MCMC based methods (e.g., Gibbs sampling) and (ii) variational methods (e.g., loopy belief propagation) [27, 24]. MCMC based techniques such as Gibbs sampling are known to be extremely slow with high mixing time, therefore requiring an extremely high number of samples. We choose the variational method called loopy belief propagation because of its simplicity and it is in many cases much faster than MCMC based methods (especially for slow mixing chains) [16]. Additionally, with careful design of the graph, the performance of the loopy belief propagation algorithm can be improved significantly. Furthermore, compared to MCMC, loopy belief propagation performs well on large scale, sparse graphical models, such as those constructed from sensors.

Another related work [28] for spatio-temporal event detection uses conditional random fields (DCRF) and belief propagation for the inference. However, DCRFs assume specific functional form of the factors (potentials) that can be factorized over a fixed number of features [25]. We take a non parametric approach in this work. Additionally, we learn the spatial dependency graph structure whereas the aforementioned work fixes the graph and can only learn the parameters. A graphical model based approach has also been used to forecast wind speeds [12], but again a strong assumption of Gaussian processes has been made. In our target application of sensor data analytics we will discuss methods of designing the MRF such that belief propagation converges faster empirically. A survey of anomaly detection techniques based on graphical representation of data, including graphical models is presented in [1].

## 3. METHODOLOGY

## 3.1 Markov Random Field

A probabilistic graphical model (PGM) [13] combines probabilistic dependency and reasoning with graph theory. Often to simplify a modeling problem, random variables are assumed to be independent, resulting in loss of information. PGMs provide a means to express and reason about these dependencies without making the problem overly complex (e.g., assuming every variable depends on all others). The graph structure provides an elegant representation of the conditional independence relationships between variables. We model the data generative process of a large group of sensors as a Markov random field (MRF), an undirected PGM where each node in the graph represents a random variable and edges represent the dependencies between the random variables. A factor or potential function specifies the relationship between variables in each clique. The joint distribution of all of random variables in a model can be factorized into the product of the clique factors through the Hammersley-Clifford theorem [22].

$$P(x_1, x_2, ..., x_n) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \phi_c(X_c)$$

where $x_i$ are the random variables; $\mathcal{C}$ is the set of all maximal cliques in the graph; $\phi_c$ is the factor for clique $X_c$; and $Z$ is the normalizing constant, also called the partition function. Markov Random fields have three important characteristics; (i) Pairwise Markov Property: Two random variables (represented by nodes in the graph) are conditionally independent given all other random variables $x_i \perp\!\!\!\perp x_j | G \setminus \{x_i, x_j\}$; (ii) Local Markov Property: A random variable (represented by a node in the graph) is conditionally independent of all other random variables given its direct neighbors in the graph $x_i \perp\!\!\!\perp G \setminus \{x_i \cup nbr(x_i)\} | nbr(x_i)$, where the set nbr($x_i$) contains the neighbors of $x_i$. (iii) Global Markov Property: Two groups random variables (represented by nodes in the graph) are conditionally independent given a set of nodes that disconnects the groups of nodes $X_i \perp\!\!\!\perp X_j | X_k$, where $X_i, X_j$ and $X_k$ are mutually exclusive groups of nodes and removal of $X_k$ disconnects $X_i$ and $X_j$.

The Hammersley-Clifford theorem allows the joint probability distribution of the graph to be factorized into a product of the joint distributions of its cliques. This is why we chose a pairwise Markov Random Field (clique size of 2) as it helps bring the down the model complexity significantly making inference scalable. Secondly, from the three MRF properties we observe that between two nodes, if all the paths (called active trails) can be broken, then by virtue of conditional independence, the model will be further simplified making it scalable to huge graphs. We achieve this by designing the MRF graphical model such that many of the active trails are broken making the inference procedure fast.

## 3.2 Loopy Belief Propagation

Belief propagation [19] is a commonly used, message-passing based algorithm for computing marginals of random variables in a graphical model, such as a Markov Random Field. It provides exact inference for trees, and approximate inference for graphs with cycles (in which case it is referred to as loopy belief propagation). Even though loopy belief propagation is an approximate algorithm with no convergence guarantees, it works well in practice for many applications [18] such as coding theory [5], image denoising [6], malware detection [4, 26]. The loopy belief propagation algorithm to compute marginals is shown in Algorithm 1.

```
1 Initialize all messages to 1;
2 while all messages not converged do
3     for $i \in \{active\ vertices\}$ do
4         read messages $m_{ji}(v_i),\ \forall j \in NB(i)$;
5         update own belief,
              $b(v_i) = g(v_i) \prod_{j \in NB(i)} m_{ji}(v_i)$;
6         send updated messages $m_{ij}(v_j) =$
              $\sum_{v_i} \frac{b(v_i)}{m_{ji}(v_i)} \phi_{ij}(v_i, v_j),\ \forall j \in NB(i)$;
      end
  end
```

**Algorithm 1:** Loopy Belief Propagation

The algorithm works in two steps - (i) based on the messages from the neighbors, a node updates its own belief (line 5) and (ii) based on its updated belief, a node sends out messages to its neighbors (line 6). In the update step (i), a node $i$'s belief $b(v_i)$ is a product of its prior belief $g(v_i)$ and messages received from its direct neighbors. $\{v_i\}$ is a set that represents the discrete domain of the probability distribution of random variable $i$. In the send step (ii), a node $i$ generates a new message for node $j$ for value $v_j$ by marginalizing over the other values $v_i (\neq v_j)$. Note that in this step when a node $i$ is creating a message for node $j$, it ignores the message it received from node $j$ itself. This is achieved by dividing the factor by $m_{ji}(v_i)$. These aforementioned steps are repeated until convergence (line 2 and 3). Convergence is achieved when there are no active vertices left. An active vertex is one whose belief has not converged yet.

The computational cost of a single iteration of Loopy Belief propagation is $O(n + e)$, where $n$ is the number of nodes and $e$ is the number of edges. Note that in reality the computational cost does not depend on all nodes, rather for a specific iteration it is only dependent on the active vertices and edges between them. Usually the number of active vertices become much smaller as the iterations progress. Since LBP does not have a convergence guarantee, it is not possible to establish overall computational complexity. However, we design the dependency graph restricting active trails and empirically show that this results in much faster convergence, allowing scale up to large graphs containing millions of nodes.

## 3.3 Graphical Model Design

While our method is generic and applies to any sensor data, in this paper we use real data from temperature sensors to detect outliers. We approach the problem by building a generative model of the temperature sensor data. Given the observations or evidence, this model can be used to predict the temperature state of a particular sensor. We inject anomalies in sensors, and then make predictions using our model. If there is a large discrepancy between the observed value and the predicted value of a sensor, we conclude the sensor observation is an outlier. We can learn a similar model, e.g., a regression model, using features available locally at a sensor, or even features obtained from sensor neighborhood, however, that model will not be able to exploit the global structure of the graph, particularly when large number of observations are missing.

### *3.3.1 Learning the dependency graph topology*

We learn the spatial dependency graph structure from historical data. This is an offline process. For each pair of sensors we compute the frequency of the co-occurring observations using the historical data and normalize it to a 0-1 scale. We call this the dependency score. For each pair of vertices we get a vector of such scores. If the maximum value in the dependency score vector exceeds a certain threshold, we keep the edge between the pair of vertices. This threshold parameter is experimentally determined. For our specific temperature sensor network application, we use fixed width binning to discretize the temperature domain into sixteen intervals of five degrees F each. Since each sensor can report one of the sixteen states, a pair of sensors can report $16X16 = 256$ jointly occurring states. In other words, the dependency score vector is of length 256.

### 3.3.2 *Creating Markov Random Field*

Given a graph topology, we now need to convert it to a pairwise Markov Random Field. Since the factors in a pairwise MRF essentially capture the dependency among the nodes and need not be proper probability distributions, we can directly use the dependency score vectors as the factors in the MRF. This gives us a way for factor construction for each edge in the MRF. In terms of the vertices, every sensor node in the dependency graph has two corresponding vertices in the MRF: 1) observed temperature of that sensor (observed state); and 2) true temperature at that location (hidden state). Next, we show two different MRF graph construction designs and discuss the benefits of the design choices. Figures 1 and 2 show the two design choices for the graph construction. The circular even numbered nodes represent the random variable for hidden true state of the sensor and the square odd-numbered nodes represents the random variable for the observed state of the sensor. Every sensor is hence represented by two random variables.

**MRF Topology 1:** In the first design choice (Figure 1), if there is an edge between two sensor nodes in the dependency graph, we add an edge between the corresponding true state nodes in the MRF. The factor on that edge will be the dependency score vector between the pair of sensor nodes in the original graph. This implies the true temperatures of the sensor locations are related according to the learned dependency graph, and the observed state of every sensor depends on the true state of that location. For every sensor node in the original dependency graph we add an edge between the true state and the observed state of the same sensor in the MRF. Thus, if there were $N$ nodes and $E$ edges in the original graph, the pairwise MRF will contain $2N$ nodes and $E + N$ edges.

**MRF Topology 2:** In the second approach (Figure 2), instead of the true states of each sensor location being dependent on each other, the true state of a sensor location depends on the observed states of its' neighboring sensors. In other words, we form a bipartite graph of the true states and the observed states. For every sensor node in the original dependency graph we introduce an observed state and true state for that sensor in the MRF. We connect the observed state and true state as before. For every edge $i, j$ in the dependency graph, we add an edge between the observed state of $i$ and the true state of $j$, and another edge between the observed state of $j$ and the true state of $i$ to form the bipartite graph.

**Figure 1: MRF Topology 1**

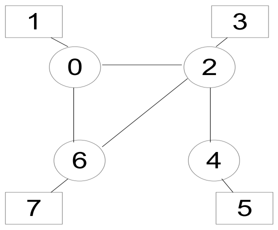
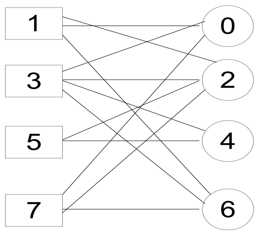


**Figure 2: MRF Topology 2**



### 3.3.3 *Discussion*

At first glance, design choice 1 seems to be the better choice as it captures the intuitive dependencies. Specifically, the true temperatures of the nearby sensor locations should be heavily dependent and the observed state of any sensor should be highly dependent on the true state of the location. This is exactly the dependency captured by the first design choice. However, this design choice has some practical problems. Since all of the random variables corresponding to the true states are hidden (not observed) and these are the ones connected in the MRF, the MRF will have a lot of loops with hidden states. In other words there will be more and longer active trails in this MRF design. As a result, the loopy belief propagation algorithm is likely to take a long time to converge. In fact, in presence of loops, the belief propagation algorithm does not have convergence guarantees. Even if it converges, it could be to a local optima. This is why design choice 1, although intuitively very appealing is not a very practical one. Design choice 2 on the other hand largely overcomes the shortcomings of choice 1. Since we formed a bipartite graph between the hidden random variables and the observed ones, the presence of an observation on the observed random variables breaks the active trail in the loop. Thus, as the number of observations increase, the number of active trail loops in the MRF decreases. The best case scenario is when we have observations for all the designated observed state variables (when all sensors are functioning). In this situation there are no active trail loops, and as a result belief propagation converges in two steps. As long as the observed values are

| | Topology 1 | Topology 2 |
|---|---|---|
| Accuracy (%) | 88.9 | 97.5 |
| Time (secs) | 663.5 | 27.6 |

**Table 1: 5% missing value prediction**

| | Topology 1 | Topology 2 |
|---|---|---|
| Accuracy (%) | 85.1 | 97.6 |
| Time (secs) | 60.1 | 45.5 |

**Table 2: 50% missing value prediction**

correct, the converging state is the global optima. Note that, in practice, a significant number of observations may be missing. Even as missing data increases, design 2 still will have less active trails as compared to design 1 and it will converge faster to more accurate results. We empirically verify this by running belief propagation on a single instance of a test graph generated from real sensor data, described in the next section. We evaluate both design choices under two configurations – 5% missing value and 50% missing value. The results are listed in Table 1 and Table 2, respectively.

Note that when there is no missing sensor data, exact inference is possible in topology 2 by considering the joint probability distribution of a node and its neighbors. The joint distribution can be factored into pairwise factors and then marginalized to get the probability distribution of the node. When missing data is low, this approximation can be used at the cost of some accuracy.

## 4. SENSOR DATA

### 4.1 Data Collection

We chose a temperature sensor network as an application in this work as it was possible to collect a large, real data set. We identified about 5,288 California weather stations from weatherunderground.com and collected raw data from the stations for a month. We used a python script to collect raw data on a daily basis for each of the stations for the entire month. Once the raw data was downloaded, we used a second script to extract the time of day, temperature, pressure, humidity and wind speed on an hourly basis for each of these stations. We use 20 days data for training our model. We used a third script to learn the pairwise dependencies between each pair of sensors and create the dependency graph. Since we check all pairs of sensors for dependency, we parallelized the script to reduce the time required by the offline training procedure. Lastly, we used a fourth script to convert the dependency graph to the pairwise MRF (both design choices).

### 4.2 Synthetic graphical model generator

The MRF graphical model of the real California sensor network is approximately 10,576 vertices and 234,968 edges. This is still quite small to understand whether our model will scale to really large scale sensor networks that we expect will be commonplace. Therefore, we develop a synthetic graphical model generator which can generate arbitrarily large graphical models while preserving the properties of the original (smaller) graphical model. The main requirement of our synthetic graphical model generator is that it should generate a Markov Random Field with similar joint probability distributions as the original smaller seed model. The second requirement is to create a set of observations for the MRF which can serve as the ground truth.

There is a lot of prior work on graph generation models that can preserve the topology (degree distribution), as well as co-related vertex attributes. For example, this could be done using the accept-reject sampling technique [20]. Unfortunately, these generic techniques are not suitable for generating MRFs. Specifically, MRFs are non-attributed graphs, where each vertex represents a random variable and each edge represents a factor potential table. A sensor node in a sensor network graph may observe multiple attributes, but when represented as a MRF, each attribute will become a vertex in the MRF and all dependencies will be captured by adding edge factor potential tables. We can still use standard techniques [17] to create a larger graph while preserving the degree distribution of the original MRF. To generate the edge factors and also to generate multiple instances of the graph with different observations that will follow the underlying joint probability distribution, we developed the following technique:

1. From the original dependence graph we extract the degree distribution.
2. From the original dependence graph we extract the dependence score vector distribution.
3. We create a larger graph where the nodes follow the same degree distribution and the edges follow the same dependence score vector distribution. The underlying assumption here is that the number of neighbors of a node is independent of the strength of the dependency with each neighbor, which is a reasonable assumption to make.
4. We then randomly select a small percentage (less than 10) of the nodes and randomly assign temperature values to them. Again the idea is, since the percentage of selected nodes is very small, the temperatures at those locations should be fairly independent of each other. Hence randomly assigning temperatures will not conflict with the dependency structure of the graph.
5. We create an MRF (choice 2) from the synthetically generated larger graph and run belief propagation to predict the state of the entire graph. This serves as the ground truth observations. We empirically show on the real graph that with 90%

missing data, the rest of the graph can be reconstructed with more than 85% accuracy. We believe the same should hold for the synthetically generated larger graph as it has the same properties of the real graph.

## 5. EMPIRICAL EVALUATION

We used an HP ProLiant DL980 server with 80 CPU cores to conduct our experiments. Each CPU core has a speed of 1.9 GHz. The server has 2 TB of memory. Our belief propagation algorithm is developed as a C++ application on top of an open source graph analytics framework called GraphLab/Powergraph [8, 15]. Note that although we use a large memory machine, we could have equally well run GraphLab on a cluster of commodity machines. For baselines and other functionalities such as data collection and graph generation, we use python scripts.

### 5.1 Experimental methodology

#### *5.1.1 Quality evaluation*

We run our belief propagation-based outlier detection algorithm on the real temperature sensor dataset. The MRF we generated had 10,576 vertices and 234,968 edges. We collected hourly observations for a month. We used the first 480 hourly observations for training the MRF (learning the graph structure and factors). We used a threshold of 0.5 for keeping an edge in the graph. In other words, after we learn the factor table for a pair of sensors, if the maximum value in the table is greater than 0.5, we introduce an edge between the pair of sensors. We also tried a lower threshold (0.4), which produces similar results but significantly increases the number of edges (over 1 million). After learning the MRF, we use the next 50 hours' observations to instantiate 50 such graphs and test our algorithm on each of the 50 instantiations.

Based on inference, our method predicts a sensor value. We evaluate our method's accuracy in two scenarios – i) when data is missing; ii) when sensor values are erroneous due to anomalies, data corruption, or malicious activity. Since a sensor value is considered an anomaly if its predicted value significantly deviates from its observed value, we use the prediction accuracy as a proxy for accuracy of anomaly detection. Even when the number of sensors with missing data, or anomalous data is large, we are still able to accurately recover their correct values, as we show in Section 5.

**Predicting missing sensor values:** To evaluate accuracy in this case, we randomly delete a percentage of sensor observations from the MRF, and then try to predict them. We do this for all the 50 test graphs we instantiated. We measure quality by the accuracy of predicting the missing values averaged over all missing values over all 50 graphs. As mentioned earlier, we divide the temperature range into 16 discrete bins. We allow the prediction to be off by at most a single bin to be considered correct. We vary the percentage of missing observations from 5% to 90% and report quality for each setting. Note that there is already missing data in the original data set.

**Detecting anomalies/outliers:** To evaluate accuracy when there is anomalous data, we randomly inject outliers to a percentage of sensor observations and then try to detect them. The outliers we inject are random temperature values. Again we predict the true state of these sensors and see whether they match (the difference can be at most one adjacent bin) with the observed values (the injected outliers). If the values do not match, then we are able to successfully detect the observation as an outlier. We report the average outlier detection accuracy over all 50 test graphs. We vary the percentage of outliers from 5% to 90% and report quality for each setting. Note that the data we introduce outliers to already has a significant amount (about 25%) of missing data as well.

#### *5.1.2 Performance evaluation*

We generate a larger graphical model with 500,000 nodes and 12 million edges using our synthetic graphical model generator, which takes as input the graphical model based on real data. We seed 1% of the nodes with random observations and run belief propagation to evaluate the performance. We measure performance in terms of time taken by the inference engine. We also measure the scale out performance by increasing the number of threads in GraphLab from 1 to 32 and observing the effect on the running time. We test two versions of the belief propagation algorithm, the bulk synchronous processing version and the dynamic asynchronous version.

#### *5.1.3 Baselines for quality comparisons*

We compare our work against four different baselines:

1. **Local regression:** We learn a sensor specific local model, which implies each sensor's model is agnostic of the rest of the sensor network. A sensor can only use its historical data to make predictions. We learn multi-variate linear regression models for each sensor. The features of the regression are *previous two hour's temperature, current pressure, current humidity and current time of day.* The model predicts the current temperature. We inject outliers in the input features (previous two hour's temperature) and see how it affects the quality. We vary the percentage of outliers from 5% to 90% and report the impact in quality. Note that this baseline is only applicable to the outlier detection application.

2. **Neighbors-mean:** Here instead of building a model based on a sensor's data alone, we build a model using a sensor's data and its neighbors' data. Note that to decide which sensors are neighbors, we still

use our MRF graph learning process as described earlier. This model simply averages the observed values of its neighbors as its predicted value. This model is used for both missing value prediction as well as outlier detection. We vary the percentage for both tasks from 5% to 90% and compare the results.

3. **Neighbors-median:** This model is essentially the same as the *Neighbors-mean*. The only difference is instead taking average, this model predicts its own value as the median of its neighbors' observed values. This model is likely to be more robust in presence of outliers.

4. **Neighbors-wmean:** Here we take the mean of the neighbors, but weigh a neighbor's value by the normalized maximum value in their joint factor table.

A more complex model could be trained on a node's local and neighbors' features. However, as missing or anomalous data increases – when our method is most advantageous – the performance of such a model is likely to severely degrade.

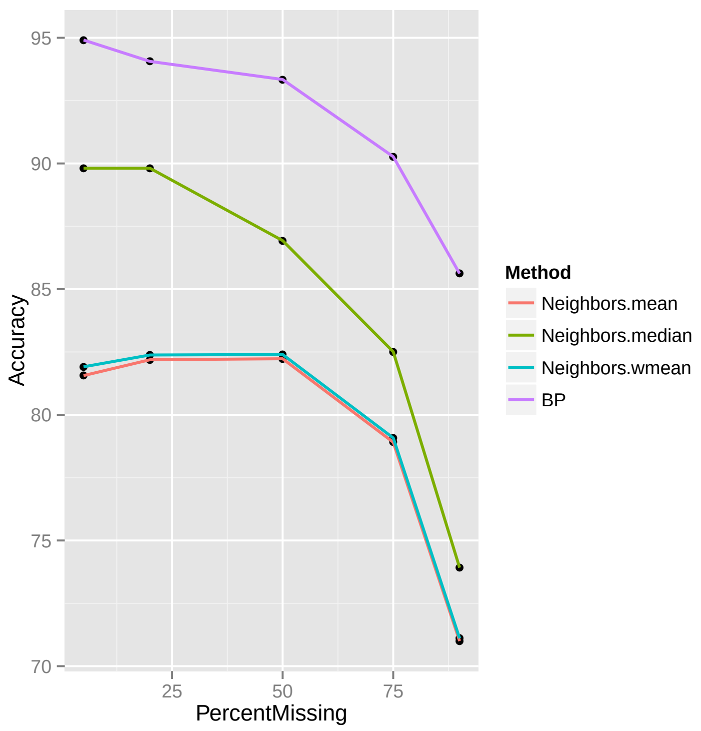
## 5.2 Results

Figures 3 and 4 show the accuracy of correctly predicting sensor values using the baseline methods and our proposed method (marked BP) for increasing percentage of missing values and increasing percentage of anomalous values, respectively.

In the case of predicting missing sensor observations (Figure 3), our belief propagation-based model performs very well. The prediction accuracy decreased only 10% (from 95% to 85%) when the percentage of missing sensor observations was increased from 5% to 90%. This shows the efficacy of learning the factors of the MRF from historical data as compared to the more traditional way where a domain expert determines the graph structure and the factors. Even when only 10% of the sensors had observations available, the whole network could be reconstructed with 85% accuracy. This is due to the ability of the technique to learn the dependencies among sensors correctly through the edge factors in MRF. This is why the dependency among two sensors that are not immediate neighbors can still be modeled through the series of edge potentials connecting them. As expected, the median-based method performs better than the mean-based method. Our method is as much as 16% better than the best baseline (Neighbors-median).

The results for the case where an increasing percentage of outliers are introduced into the data set are quite interesting. We see from Figure 4 that our belief propagation based model's accuracy is very high (about 90%) when the percentage of injected outliers is less than 50%. Once the number of outliers increases beyond 50%, there is a substantial degradation in accuracy. We

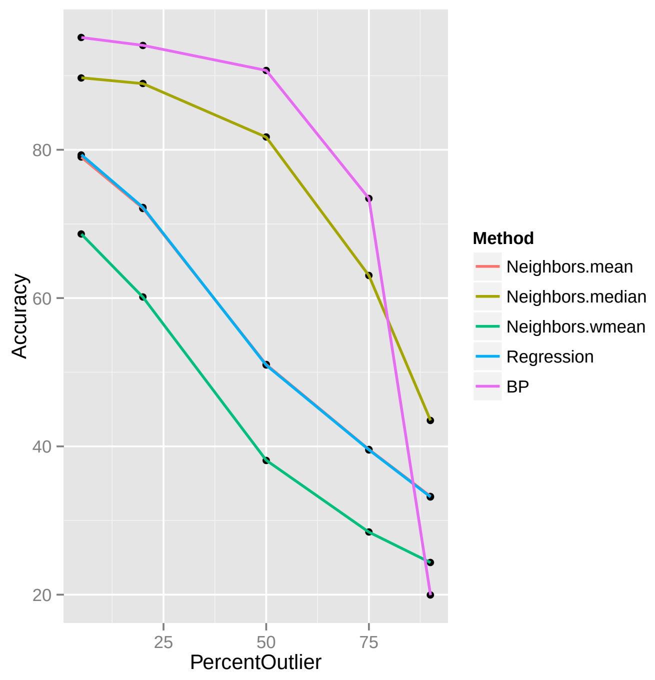
**Figure 3: Model accuracy results – Data with missing values.**



believe this is because belief propagation is a gossip-based, message passing algorithm. As a result, once the number of outliers is a majority, the global belief converges to the outlier values. When the number of correctly observed values are a majority, most beliefs still converge to correct values. However, once the number of outliers become the majority, the model starts degrading rapidly. While our method performs better than the baseline methods up to around 80% outliers, beyond that it does worse. This is expected due to the global impact of the outlier nodes in BP. In the local regression based model, almost any injected outlier results in an incorrect prediction. This is why the degradation in quality is roughly linear with the increase in outlier percentage. Another point to note is that the degradation in accuracy is more severe with outliers than with missing data. This is true for both the baselines and our method. Here again, our method is as much as 16% better than the best baseline (Neighbors-median).

The performance numbers are shown in Figures 5 and 6. We vary the number of threads from 1 to 32 and report the scale-out pattern of the belief propagation inference engine on GraphLab for two MRFs - 1) 10,576 vertices and 234,968 edge real dataset and 2) 500,000 vertices and 12 million edge synthetic dataset. While the asynchronous version performs better than the synchronous version, both version performed quite well with increasing parallelism, as shown in Figures 5 and 6. On the largest graph (5M nodes, 120M edges)

**Figure 4: Model accuracy results – Data with anomalies/outliers.**



| Nodes, Edges | Memory (GB) |
|---|---|
| 11K, 235K | 3.5 |
| 500K, 12M | 160 |
| 5M, 120M | 1,600 |

**Table 3: Memory consumption of belief propagation**

the synchronous version using 32 threads completed in 12 hours. Table 3 provides the memory requirements of each graph. These results indicate that the memory consumption scales linearly with the size of the graphical model.

## 6. CONCLUSIONS

We developed techniques for anomaly detection in sensor data as well as for predicting missing sensor observations. We presented a novel way for learning the dependence structure among sensors and represented it using a bipartite Markov Random Field graphical model which is very amenable to the belief propagation inference algorithm both in terms of accuracy and performance. We tested our algorithm on real temperature sensor data collected from 5,288 California weather stations. Results showed that our method can predict missing values with 90% accuracy even when 75% of the data is missing; similarly, it can correctly identify 90% of outliers even when 50% of the sensors are anomalous. Our algorithm provides most benefit when large amounts of data is missing by exploiting transitive dependencies through use of graphical models and belief propagation. We also developed a synthetic graphical model generator which can generate a large MRF while preserving the properties of a smaller graphical model. We use it to generate a 500,000 nodes and 12 million edges graphical model to show scalability of our method.

**Figure 5: Runtimes for loopy BP on MRF with 11K nodes, 235K edges.**

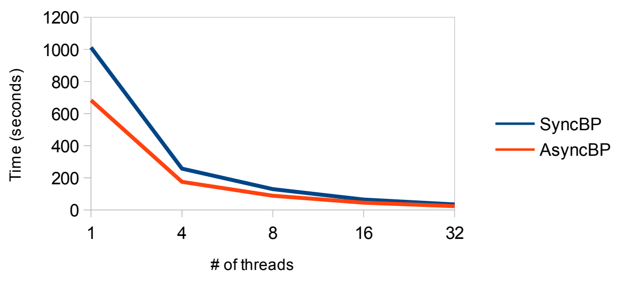
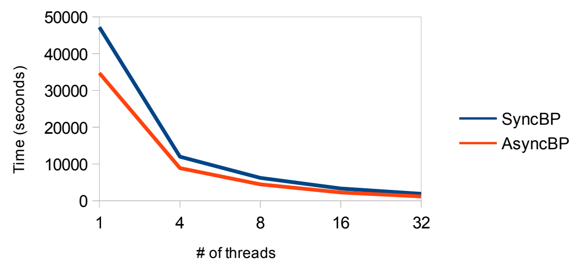


**Figure 6: Runtimes for loopy BP on MRF with 500K nodes, 12M edges.**



## 7. REFERENCES

[1] Akoglu, L., Tong, H., and Koutra, D. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery 29*, 3 (2015), 626–688.

[2] Balaji, B., Verma, C., Narayanaswamy, B., and Agarwal, Y. Zodiac: Organizing large deployment of sensors to create reusable applications for buildings. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments* (New York, NY, USA, 2015), BuildSys '15, ACM, pp. 13–22.

[3] Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM Computing Surveys (CSUR) 41*, 3 (2009), 15.

[4] Chau, D., Nachenberg, C., Wilhelm, J., Wright, A., and Faloutsos, C. Polonium: Tera-scale graph mining and inference for malware detection. In *SIAM International Conference on Data Mining* (2011), vol. 2.

[5] Chen, J., and Fossorier, M. P. Near optimum universal belief propagation based decoding of low-density parity check codes. *Communications, IEEE Transactions on 50*, 3 (2002), 406–414.

[6] Freeman, W. T., Pasztor, E. C., and Carmichael, O. T. Learning low-level vision. *International journal of computer vision 40*, 1 (2000), 25–47.

[7] Getoor, L. *Introduction to statistical relational learning*. MIT press, 2007.

[8] Gonzalez, J. E., Low, Y., Gu, H., Bickson, D., and Guestrin, C. Powergraph: Distributed graph-parallel computation on natural graphs. In *OSDI* (2012), vol. 12, p. 2.

[9] Hawkins, D. M. *Identification of outliers*, vol. 11. Springer, 1980.

[10] Jordaan, E. M., and Smits, G. F. Robust outlier detection using svm regression. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on* (2004), vol. 3, IEEE, pp. 2017–2022.

[11] Kang, U., Chau, D. H., and Faloutsos, C. Mining large graphs: Algorithms, inference, and discoveries. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on* (2011), IEEE, pp. 243–254.

[12] Kapoor, A., Horvitz, Z., Laube, S., and Horvitz, E. Airplanes aloft as a sensor network for wind forecasting. In *Proceedings of the 13th international symposium on Information processing in sensor networks* (2014), IEEE Press, pp. 25–34.

[13] Koller, D., and Friedman, N. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[14] Kriegel, H.-P., Kröger, P., and Zimek, A. Outlier detection techniques. In *Tutorial at the 16th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Washington, DC* (2010).

[15] Low, Y., Gonzalez, J., Kyrola, A., Bickson, D., Guestrin, C., and Hellerstein, J. M. Graphlab: A new parallel framework for machine learning. In *Conference on Uncertainty in Artificial Intelligence (UAI)* (Catalina Island, California, July 2010).

[16] Matthews, P. A slowly mixing markov chain with implications for gibbs sampling. *Statistics & probability letters 17*, 3 (1993), 231–236.

[17] Miller, J. C., and Hagberg, A. Efficient generation of networks with given expected degrees. In *Algorithms and Models for the Web Graph*. Springer, 2011, pp. 115–126.

[18] Murphy, K. P., Weiss, Y., and Jordan, M. I. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence* (1999), Morgan Kaufmann Publishers Inc., pp. 467–475.

[19] Pearl, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[20] Pfeiffer III, J. J., Moreno, S., La Fond, T., Neville, J., and Gallagher, B. Attributed graph models: modeling network structure with correlated attributes. In *Proceedings of the 23rd international conference on World wide web* (2014), International World Wide Web Conferences Steering Committee, pp. 831–842.

[21] Ramaswamy, S., Rastogi, R., and Shim, K. Efficient algorithms for mining outliers from large data sets. In *ACM SIGMOD Record* (2000), vol. 29, ACM, pp. 427–438.

[22] Robert, C., and Casella, G. A short history of markov chain monte carlo: subjective recollections from incomplete data. *Statistical Science* (2011), 102–115.

[23] Scott, D. W. Kernel density estimators. *Multivariate Density Estimation: Theory, Practice, and Visualization* (2008), 125–193.

[24] Sen, P., Namata, G. M., Bilgic, M., Getoor, L., Gallagher, B., and Eliassi-Rad, T. Collective classification in network data. *AI Magazine 29*, 3 (2008), 93–106.

[25] Sutton, C., McCallum, A., and Rohanimanesh, K. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research 8*, Mar (2007), 693–723.

[26] Tamersoy, A., Roundy, K., and Chau, D. H. Guilt by association: large scale malware detection by mining file-relation graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014), ACM, pp. 1524–1533.

[27] Yedidia, J. S., Freeman, W. T., and Weiss, Y. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium 8* (2003), 236–239.

[28] Yin, J., Hu, D. H., and Yang, Q. Spatio-temporal event detection using dynamic conditional random fields. In *IJCAI* (2009), vol. 9, Citeseer, pp. 1321–1327.

[29] Zhang, J., and Zulkernine, M. Network intrusion detection using random forests. In *PST* (2005), Citeseer.

[30] Zhang, Y., Meratnia, N., and Havinga, P. Outlier detection techniques for wireless sensor networks: A survey. *Communications Surveys & Tutorials, IEEE 12*, 2 (2010), 159–170.