2018 IEEE 30th International Conference on Tools with Artificial Intelligence

Constraint-Based Learning for Sensor Failure Detection and Adaptation

Yuan Shi
Information Sciences Institute
University of Southern California
yuanshi@usc.edu

T. K. Satish Kumar
Information Sciences Institute
University of Southern California
tkskwork@gmail.com

Craig A. Knoblock
Information Sciences Institute
University of Southern California
knoblock@isi.edu

***Abstract*—In this paper, we address the problem of automatically detecting and adapting to sensor failures, which is an important step towards building long-lasting survivable software. We present a novel constraint-based learning framework that performs joint sensor failure detection and adaptation. Our framework learns sensor relationships from historical data and expresses them as a set of constraints. These constraints then provide a joint view for detection and adaptation: detection checks which constraints are violated, and adaptation reconstructs failed sensor values. Additionally, we show that our framework can not only identify the mode of sensor failure but can also estimate the quality of the proposed adaptation. Our empirical studies on sensor data from the weather and appliance energy domains demonstrate the advantages of our approach over other methods.**

***Index Terms*—sensor failure detection and adaptation, sensor relationships, constraint-based learning**

I. Introduction

Many software systems run on long-lifespan platforms that operate in dynamic environments. Maintaining the quality, durability and performance of these software systems is very challenging and labor-intensive. Failure to effectively or promptly adapt to hardware and resource changes can result in technically inferior and potentially vulnerable systems [1]. For example, software systems based on sensor data can suffer from sensor failures caused by environmental changes and technical errors. If software systems could automatically detect and adapt to sensor failures, they would significantly reduce the time and effort required for their maintenance. As an important step towards building long-lasting survivable software systems, we study the problem of automatically detecting and adapting to sensor failures. Efficient solutions to this problem can have a broader impact for the increasing number of sensors deployed in real-world systems [2], [3]. Although sensor failure detection has been studied extensively in existing work [4]–[7], automatic adaptation to sensor failures has not been a focus thus far. Typically, human experts are required to examine detected sensor failures and make subsequent decisions about how to adapt to them.

To address the overall problem, we present a novel machine learning framework called **JDA** (Joint Detection and Adaptation) that performs sensor failure detection and adaptation *jointly*. The key idea is to determine the *reconstruction relationships* among sensors, i.e., how one sensor value can be reconstructed from other sensor values. This is based on the observation that, in real-world systems, sensor values are often correlated [8]. Taking weather sensors as an example, temperature, dew point and humidity are highly correlated [9]; and each sensor value can be efficiently reconstructed from the other two. While reconstruction relationships can be generally complex, our framework decomposes this complexity into a set of simpler constraints. In particular, it uses a substrate of inequality constraints that resemble

$$(\text{temperature} - f(\text{dew point}, \text{humidity})) \leq \epsilon^2, \qquad (1)$$

where $f()$ is a function that captures known sensor relationships, and ϵ^2 is the corresponding error bound. These constraints provide a joint view for sensor failure detection and adaptation when new sensor value readings come in.

- Detection: Our framework checks each constraint, and a sensor failure is reported if one or more constraints are violated. We then infer the likely failed sensor(s) from the violated constraints.
- Adaptation: Once the failed sensors are identified, our framework reconstructs the failed sensor values from the remaining working sensor values by solving the set of constraints. Tighter constraints correspond to more accurate reconstruction relationships.

By using the same set of constraints for both detection and adaptation, our approach provides an extensible way to address the interrelated problems in one unified framework.

One important challenge in our framework is that the functions $f()$ are not necessarily given to us beforehand. Thus, a second operating idea in our framework is to extract them from historical sensor data. The extraction procedure considers different combinations of sensors and derives the functions $f()$ using nonlinear regression methods [10]. Compared to existing detection methods that extract only linear relationships [2], our extraction procedure not only enables learning more complex functions $f()$ but also results in lower reconstruction errors produced by the entire framework.

To enhance the usefulness of the proposed framework for practical applications, we provide two additional features. First, when a sensor failure occurs, we not only detect it but also identify its mode of failure. This enables our detection procedure to provide additional information to higher layers

2375-0197/18/$31.00 ©2018 IEEE
DOI 10.1109/ICTAI.2018.00059

of the software system, which in turn facilitates faster recovery operations. We extract features from both observed and reconstructed sensor values within a time window and classify them into five common modes of failure (Outlier, Spike, Stuck-At, High-Noise and Miscalibration) [11].

The second feature is the ability to estimate the quality of a reconstructed sensor value by a relevant error interval. This enables the higher layers of the software system to make decisions about the quality of the reconstructed sensor values and possibly determine the necessity of further refinement. In order to produce these error intervals, we develop a data-driven procedure that first identifies similar sensor values in historical data and then computes the relevant statistics on them.

An empirical study of sensor data from Weather Underground[1] and the Appliances Energy Dataset[2] shows that our framework detects sensor failures more accurately than other competing methods. The results also demonstrate the overall efficacy of our constraint-based framework in: (a) successfully identifying different modes of sensor failures, (b) adapting to failures by efficiently reconstructing the required sensor values, and (c) estimating the qualities of the reconstructed sensor values for higher-level decisions.

Contributions: We present a novel computational framework for joint sensor failure detection and adaptation based on learning constraints from historical sensor data. These constraints jointly express sensor relationships. While the relationships themselves could be fairly complex, each individual constraint is simple enough to be amenable to state-of-the-art Machine Learning techniques. Note that a direct application of Machine Learning techniques to predict the value of one sensor based on learning its relationship to other sensor values is unviable for our problem since multiple sensors can fail at the same time. It is in these situations that our constraint-based learning framework becomes indispensable. Importantly, the constraints allow us to first identify the failed sensors and subsequently reconstruct the corresponding sensor values. In addition to automatic sensor failure detection and adaptation, two important features of our approach are that it can automatically identify different modes of sensor failures and provide estimates of the adaptation quality. These features facilitate interpretability and robustness at a practical level.

II. Joint Detection And Adaptation

Our framework exploits the observation that real-world systems are often equipped with sensors that are correlated with each other. Such correlations could exist either between different sensor types (e.g., temperature, dew point and humidity from the same weather station) or within the same sensor type (e.g., wind speed in nearby weather stations). In this paper, we explore a specific type of relationship between sensor values that can be characterized by a *reconstruction function* $f()$. A simple example illustrates this concept. Consider humidity HU in %. It is well known that it can be accurately determined by temperature TP in °C and dew point DP in °C [9]:

$$HU \approx f(TP, DP) = 100 \exp\left(\frac{aDP}{b + DP} - \frac{aTP}{b + TP}\right). \quad (2)$$

Here, f serves as a reconstruction function that takes input sensor values TP and DP and outputs sensor value HU. a and b are constants. In practice, the following constraint holds between the different sensor values.

$$(HU - f(TP, DP))^2 \leq \epsilon^2, \quad (3)$$

where ϵ^2 is an error bound that intrinsically measures the reconstruction quality of HU via $f(TP, DP)$. In addition, ϵ^2 can be derived from historical sensor data. For instance, we can set ϵ^2 to be the minimum value such that 95% of historical sensor values satisfy Eq. (3).

Assuming that the sensors for TP and DP work correctly, a failure in the sensor for HU is characterized by the violation of the constraint in Eq. (3). In fact, in such a case, we can even adapt to the failure by reconstructing HU via $f(TP, DP)$; and doing so automatically satisfies Eq. (3). Additionally, ϵ^2 in Eq. (3) provides an estimate of the adaptation quality (discussed later in Section IV). However, the general challenge is that the sensors for TP and DP may not always work correctly either. If one or more of them fail in addition to the failure of the sensor for HU, Eq. (3) can neither be used to detect this failure nor can it be used to reconstruct the value of HU. This problem persists whether or not $f()$ is explicitly known and whether or not it is learned using state-of-the-art Machine Learning methods. Our framework therefore uses an additional layer of reasoning beyond just a direct application of Machine Learning methods to learn relationships between sensor values. In particular, it builds a substrate of constraints that retain enough simplicity individually and yet capture enough complexity and redundancy collectively. Our constraint-based framework can therefore be effectively used to first address the problem of sensor failure detection and then address the problem of failed sensor value reconstruction.

We begin by introducing some notation. Let $S_1, S_2, \cdots, S_K$ denote the K sensors in the system. We assume that any sensor value can be accessed at any time. Let x_k^t be the sensor value generated by S_k at time t. In the first step that addresses sensor failure detection, we are interested in detecting the possible failure of each S_k at a desired time t, i.e., determining whether or not x_k^t should be deemed as being reliable. Of course, doing so allows us to detect sensor failures instantly, without having to wait for a time window of sensor values. We also assume that we are given N inequality constraints with reconstruction functions. (We discuss how to actually derive such reconstruction functions in Section II-C.) Each such inequality constraint describes the relationship between a set of input sensor values and an output sensor value. Specifically, the nth constraint is as follows.

$$(y_n - f_n(\mathbf{z}_n))^2 \leq \epsilon_n^2, \quad (4)$$

- y_n: output sensor value at some time t, e.g., $y_n = x_1^t$;

[1]https://www.wunderground.com/

[2]https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction

- $\mathbf{z}_n$: input sensor values at time $\leq t$, e.g., $\mathbf{z}_n = [x_2^t, x_3^t]$. Note that $\mathbf{z}_n$ can also involve input sensor values at time $\leq t$, e.g., $\mathbf{z}_n = [x_2^t, x_3^t, x_1^{t-1}, x_2^{t-1}]$, where x_1^{t-1} and x_2^{t-1} can be treated as additional input sensor values;
- $f_n()$: reconstruction function that attempts to reconstruct y_n from $\mathbf{z}_n$ derived from historical sensor data;
- ϵ_n^2: a reconstruction error bound derived from historical sensor data.

A. Detecting Sensor Failures

As the system receives sensor readings, it can check each constraint and identify the violated ones at any given time t. If the nth constraint is violated, then at least one sensor involved in that constraint has likely failed. Furthermore, the system can infer the set of failed sensors from the set of violated constraints. To do this, we first introduce K Boolean variables $\{v_k\}$, for $k = 1, 2, \cdots, K$, where v_k is 1 if sensor S_k has failed, and is 0 otherwise. The existence of at least one failed sensor corresponding to each violated constraint translates to a set of linear constraints on $\{v_k\}$. For instance, if a violated constraint involves sensors S_1 and S_3, then the corresponding linear constraint is $v_1 + v_3 \geq 1$, since at least one of v_1, v_3 should have value 1. More generally, if the nth constraint is violated, then the sum of all v_k involved in $[\mathbf{z}_n, y_n]$ should be greater than or equal to 1.

$$\sum_{k \in [\mathbf{z}_n, y_n]} v_k \geq 1 \tag{5}$$

Our goal is to find an assignment of Boolean values to the variables $\{v_k\}$ so that it represents the best possible explanation for the observed sensor values. Clearly, such an assignment should satisfy all linear constraints of the form Eq. (5). But, of course, this requirement alone is incomplete since it admits a vacuous solution, e.g., $v_k = 1$ for all k. Therefore, we further qualify our solution with the requirement that it has to minimize the total number of failed sensors. This formalization is based on the intuition that sensors behave nominally most of the time and their failure probabilities are typically much smaller than 0.5. Our formalization also matches the ones popularly used in model-based diagnosis [12]. Of course, richer formalizations can be developed with more information on the prior failure probabilities of individual sensors and physical models of how they interact with each other. Importantly, any preferred formalization can be seamlessly incorporated in our framework.

Overall, we now have the following combinatorial optimization problem for sensor failure detection.

$$\min \sum_{k \in [1, K]} v_k \tag{6}$$

$$\text{s.t.} \sum_{k \in [\mathbf{z}_n, y_n]} v_k \geq 1, \forall n \in \mathcal{V} \tag{7}$$

$$v_k \in \{0, 1\}, \forall k \in \{1, 2, \cdots, K\} \tag{8}$$

where $\mathcal{V}$ denotes the set of indices of violated constraints. This problem is a specific kind of a 0-1 Integer Linear Program (ILP), called the *Hitting Set Problem*, and is NP-hard to solve in general. However, there are a number of heuristic and approximation algorithms to solve it efficiently. In our implementation, we use the cutting plane method [13] to convert it into a series of LPs.

B. Adapting to Sensor Failures

When sensor failures are detected, we would like the system to automatically adapt to such failures. Our adaptation strategy is to reconstruct the sensor values of the failed sensors from the sensor values of other working sensors. This essentially replaces failed physical sensors with working virtual sensors that enable the system to continue its operation. For reconstructing a failed sensor's values, our approach identifies a constraint in which the output sensor is the failed sensor and all input sensors are working sensors. Then, the corresponding reconstruction function is used. When multiple constraints qualify to be chosen for reconstruction, our procedure selects the constraint with the lowest reconstruction error bound for more accurate results. Specifically, to reconstruct the values of a failed sensor S_k, we do the following:

1) Find all constraints with working input sensors and output sensor S_k.
2) Select the constraint with the lowest reconstruction error bound from this set of constraints.
3) Apply the corresponding reconstruction function on the working input sensor values.

C. Learning Reconstruction Functions from Historical Data

The detection and adaptation procedures discussed above assume that the reconstruction functions are already given. In practice, however, such functions may not be directly available. Instead, we automatically extract them from historical sensor data. Ideally, the learned relationships are expected to have the following properties.

- Accuracy: Each relationship should give us the capability to reconstruct the output sensor value with reasonably low reconstruction error.
- Comprehensiveness: The relationships should be rich enough to help us detect and adapt to various kinds of sensor failures. That is, we would like to extract various types of useful relationships. For example, temperature can be reconstructed using dew point and humidity from the same weather station, and it may also be reconstructed using temperature from nearby weather stations.
- Compactness: The relationships should be easy to state and understand. There are two levels of compactness. First, each relationship should involve only a small number of sensors. The lower the number of sensors, the smaller the chance the constraint is violated. Using a small number of sensors in each relationship improves the overall robustness of our framework and makes the learned relationships more interpretable by humans. Second, the number of learned relationships should also be small since this affects the complexity of our algorithms.

To learn sensor relationships with the above properties, we developed a method that groups input sensors into a number of subsets and then learns reconstruction functions within each subset. The subsets have the following properties.

- Sparsity: Each subset is of small cardinality.
- Disjointness: Subsets tend to be disjoint from each other.

The above properties significantly reduce the number of constraints without compromising the span of what relationships can be represented. In effect, such a subset selection method ensures the compactness and comprehensiveness properties.

Our grouping procedure works as follows. Suppose we want to learn relationships from the input sensor values $x_1, x_2, \cdots, x_D$ to the output sensor value y. To discover the group of input sensors to include in the first constraint, we learn a sparse vector $\mathbf{w}_1 \in \mathcal{R}^D$ that selects a small subset of the input sensors. Mathematically, we solve the following LASSO problem [14].

$$\min_{\mathbf{w}_1} \sum_t (\mathbf{w}_1^{\mathrm{T}} \mathbf{x}^t - y^t)^2 + \lambda \sum_d |\mathbf{w}_{1d}| \tag{9}$$

where $\sum_d |\mathbf{w}_{1d}|$ enforces the sparsity of $\mathbf{w}_1$ and $\sum_t (\mathbf{w}_1^{\mathrm{T}} \mathbf{x}^t - y^t)^2$ minimizes the reconstruction error between the linear combination of selected $\mathbf{x}^t$ and y^t over historical data. $\lambda > 0$ is a tradeoff parameter that can be tuned using cross validation.

Once we learn $\mathbf{w}_1$ and identify the relevant subset of the input sensors, a reconstruction function can be learned in many possible ways. To ensure a high quality, we apply state-of-the-art nonlinear regression methods (e.g., Neural Networks) to learn the reconstruction functions from historical data.

$$\min_f \sum_t (y_t - f(\text{subset of}\, \mathbf{x}_t))^2 \tag{10}$$

The subset of input sensors needed for the second constraint can be learned using a new sparse vector $\mathbf{w}_2 \in \mathcal{R}^D$ that is conditioned on $\mathbf{w}_1$. Specifically, we have

$$\min_{\mathbf{w}_2} \sum_t (\mathbf{w}_2^{\mathrm{T}} \mathbf{x}^t - y^t)^2 + \lambda \sum_d |\mathbf{w}_{1d}||\mathbf{w}_{2d}| \tag{11}$$

where $\sum_d |\mathbf{w}_{1d}||\mathbf{w}_{2d}|$ encourages $\mathbf{w}_2$ and $\mathbf{w}_1$ to retain a disjoint set of input sensors. As before, we can derive the second reconstruction function from $\mathbf{w}_2$ following Eq. (10).

More generally, we can learn the pth vector $\mathbf{w}_p$ that identifies the subset of input sensors in the pth constraint by solving

$$\min_{\mathbf{w}_p} \sum_t (\mathbf{w}_p^{\mathrm{T}} \mathbf{x}^t - y^t)^2 + \lambda \sum_d u_d |\mathbf{w}_{pd}| \tag{12}$$

where $u_d = (\sum_{i=1}^{p-1} |\mathbf{w}_{id}|)/(p-1)$, and $\sum_d u_d |\mathbf{w}_{pd}|$ encourages $\mathbf{w}_p$ to pick sensors different from the ones chosen in the previous $p-1$ subsets. The procedure stops when the reconstruction error in Eq. (10) exceeds a pre-defined threshold or p exceeds an upper bound.

a) Acceleration of Sensor Grouping: When the number of sensors K is large, solving a series of Eq. (12) instances can be computationally very expensive. As an acceleration strategy, we can first cluster the input sensors into several high-level clusters based on their correlation matrix [15]. After that, we can apply the above grouping procedure within each high-level cluster. Although this strategy ignores possible relationships across high-level clusters, it is computationally attractive and performs well empirically.

III. Identifying Modes of Sensor Failures

- Outlier: One or more sensor values are far away from the normal values.
- Spike: A band of consecutive sensor values exhibits a greater-than-expected rate of change.
- Stuck-At: There is zero variation in the sensor values for an unexpected length of time.
- High-Noise: There is an unexpectedly high variation in the sensor values in a period of time.
- Miscalibration: There is a constant offset from the ground truth for the sensor values in a period of time.

Identifying the modes of sensor failures is essentially a multi-class classification problem where the input is a time window of sensor values and the output is the identified mode of failure. For such a classification problem, it is important to consider a time window of sensor values because most modes of failure are defined and identifiable only through characteristics of sensor values over a period of time. While existing studies have already explored Machine Learning techniques like Neural Networks to classify different modes of failure [16], [17], these methods only capture information from the failed sensor itself. On the other hand, in our approach, we are able to naturally leverage information from multiple related sensors and improve the accuracy and robustness of classification. Specifically, our approach extracts features from the observed sensor values as well as the reconstructed sensor values for a failed sensor. Therefore, the extracted features capture essential information from the failed sensor and the other related working sensors.

Let W be the user-specified size of the time window; and let the observed sensor values of a failed sensor within such a time window be $[x^{t-W+1}, x^{t-W+2}, \cdots, x^t]$. Additionally, let the reconstructed sensor values computed for this failed sensor be $[\hat{x}^{t-W+1}, \hat{x}^{t-W+2}, \cdots, \hat{x}^t]$. We first compute informative statistics like the mean, minimum, maximum and standard deviation on both the observed sensor values and the reconstructed sensor values. We then concatenate the raw sensor values and these informative statistics to constitute an input feature vector that can be used to train a classifier. We note that the length of our feature vector depends only on the window size W but not on the number of sensors.

IV. Estimating Adaptation Quality

To build survivable software, estimating the quality of adaptation is also important since it enables higher-level software components to determine whether or not to accept a proposed reconstruction. Towards this end, we would like our framework to estimate an error interval for the gap between the reconstructed sensor value and the ground truth. Suppose the reconstruction relationship $(\mathbf{z}_n, y_n, f_n, \epsilon_n)$ is used for adaptation at time t. One idea is to directly use ϵ_n as an

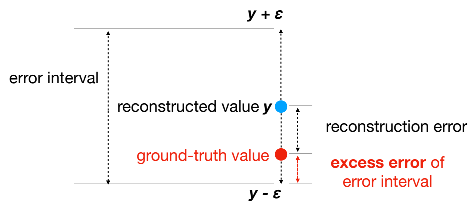


Fig. 1: Notion of excess error of the error interval

error bound. However, this bound is a static quantity and does not provide much information about the *adaptation* itself. We would therefore like to have an adaptive error interval that is dynamically estimated based on the observed sensor values. Our idea is to estimate such an interval using samples in historical data that are similar to the observed sample. This leads to the following approach.

1) For a tuple of sensor values at time t, search for its κ nearest neighbors in the historical sensor data. When identifying the neighbors, we don't use failed sensor values since they can be far away from the normal data. Instead, we only compute Euclidean distances in the space of the working sensors $\mathbf{z}_n$.
2) For each identified neighbor, compute its exact reconstruction error using the reconstruction function f_n.
3) Compute the average reconstruction error over κ neighbors and denote it as $\hat{\epsilon}^t$.
4) Set the estimated error interval to be $[-\alpha\hat{\epsilon}^t, \alpha\hat{\epsilon}^t]$ at time t, where $\alpha > 0$ is a scaling factor. An ideal α makes the error interval as *tight* as possible. In other words, we would like the length of the interval to be larger than the reconstruction error but not exceed it too much. α can be tuned using historical data as discussed below.

a) Excess Error of the Error Interval: To quantify the tightness of the estimated error interval, we introduce the notion of *excess error* (Fig. 1). The excess error is defined as the gap between the ground-truth value and the closest endpoint of the error interval, when the interval subsumes the reconstruction error. If the interval does not subsume the reconstruction error, we consider the interval as having failed. In practice, we can tolerate a small failure rate of the estimated error interval by setting a recall parameter (e.g., 90%). We can then find the smallest α to achieve the given recall and compute the corresponding excess error. Clearly, we favor a smaller excess error as it results in a tighter error interval.

V. Experimental Results

We evaluate our **JDA** framework on two real-world sensor datasets. One dataset consists of sensor data from personal weather stations, and the other dataset consists of appliance energy data from a wireless sensor network within a house. For both datasets, we use the first half of the time series as historical data required for learning the sensor relationships and the second half of the time series as test data for evaluation. For the weather dataset, the number of sensor failures is fairly small and the modes of sensor failures are also not uniformly distributed. The appliance energy dataset does not contain any sensor failures at all. In order to better evaluate the performances of various algorithms, therefore, we *simulate* sensor failures in both domains based on a prior history of failures for each sensor. To simulate sensor failures, we run the following procedure multiple times for each sensor.

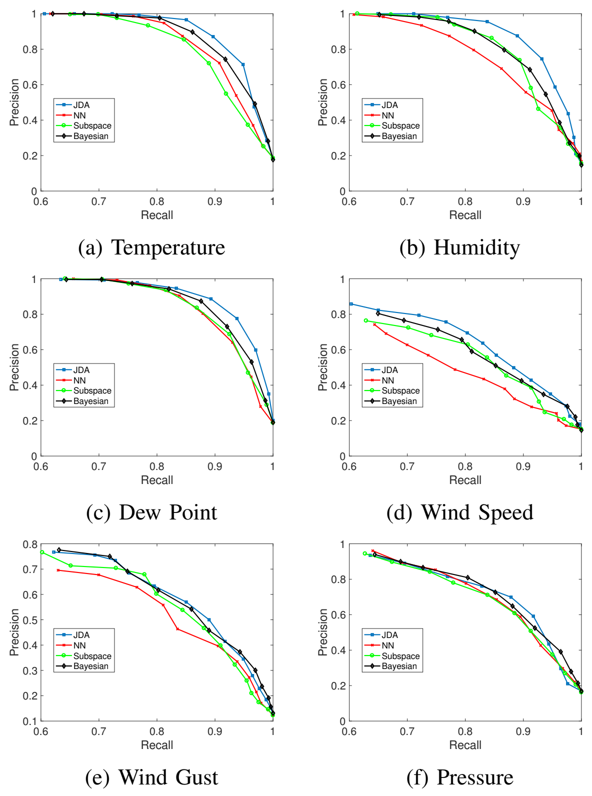
1) Select any point in the time series with probability 0.01.
2) Starting from each selected point, generate a time window with length chosen uniformly at random from the interval $[1, 30]$. The time window should not overlap with already generated time windows.
3) Select one of the 5 modes of failure uniformly at random.
4) Simulate sensor failures based on the selected mode. Specifically, we generate an instance of each mode of failure in the following ways.
   - Outlier: Set the middle point in the time window to an arbitrary value that significantly deviates from the mean (by more than 3 standard deviations).
   - Spike: Set the middle point in the time window as an outlier; and set the remaining points in the time window using linear interpolation between the middle point and the boundary points.
   - Stuck-At: Set all points in the time window to a fixed arbitrary value.
   - High-Noise: Add significant Gaussian noise to all points in the time window.
   - Miscalibration: Offset all points in the time window with a fixed arbitrary bias.

We note that this procedure allows for simultaneous multiple sensor failures since time windows generated for different sensors can overlap.

a) Detection of Failures: For sensor failure detection, we compare **JDA** to several baseline algorithms.

- **NN**: Nearest Neighbor method, which identifies a sensor failure if the sensor values are far away from normal [18]. This method can detect sensor failure at a vector level (i.e., a group of sensors), but cannot identify which individual sensor(s) actually fail. Therefore we use it for each individual sensor and treat a time window of consecutive readings as a vector. The length of the time window is tuned based on historical data.
- **Subspace**: Subspace method, which learns a set of bases from historical data and then identifies sensor failures if the sensor values are difficult to reconstruct via these bases [19]. Since **Subspace** only identifies sensor failures at a vector level, we adopt the same strategy used in **NN**.
- **Bayesian**: Probabilistic method, which captures linear relationships between sensors and is capable of modeling the working status of each sensor [20].

We consider different values of recall (60% to 100%) and measure the corresponding precision. For identifying the modes of failures, we compare **JDA** to the following methods.



(a) Temperature (b) Humidity (c) Dew Point (d) Wind Speed (e) Wind Gust (f) Pressure

Fig. 2: Precision-recall curves on sensor data from the Austin weather stations.

- **Neural**: Neural Networks trained on sensor values from a single sensor.
- **Ground**: The same as **JDA**, except that the reconstructed sensor values are replaced by ground truth readings. Although this method is not realistic, it provides an upper bound on the classification accuracy.

In all methods, to classify the mode of failure at time t, we use sensor values in the time window $[t-10, t+10]$. We compute the classification accuracy by comparing the identified mode of failure to the actual mode of failure.

b) Adaptation to Failures: For sensor failure adaptation, we measure both the average reconstruction error and the average excess error of the adaptation error interval. Reconstruction error is measured as the root mean square error (RMSE) between the reconstructed and the ground truth sensor values. For comparison, we introduce a baseline algorithm called **Reference** that uses a simple strategy to reconstruct failed sensor values without using any Machine Learning techniques. We discuss how **Reference** is implemented for each dataset later. The excess error of the error interval is also measured using the RMSE. We compare the excess error of **JDA** to that of a baseline algorithm called **Const** which uses the constant error bound ϵ_n^2 in Eq. (4).

TABLE I: Accuracy of identifying different modes of sensor failures in the Austin weather stations.

Sensor	**Neural**	**JDA**	**Ground**
Temperature	92.4	91.8	96.4
Humidity	89.3	90.7	96.3
Dew Point	91.6	91.3	97.1
Wind Speed	77.4	82.6	94.4
Wind Gust	81.1	83.4	95.8
Pressure	85.7	87.2	96.0

TABLE II: Adaptation performance on sensor data from the Austin weather stations.

Sensor	Reconstruction Error		Excess Error	
	Reference	**JDA**	**Const**	**JDA**
Temperature	1.32	0.23	0.44	0.19
Humidity	5.28	0.41	0.93	0.30
Dew Point	1.15	0.33	1.21	0.17
Wind Speed	5.36	3.81	4.60	3.12
Wind Gust	5.12	3.68	5.09	2.86
Pressure	3.71	2.25	3.35	1.84

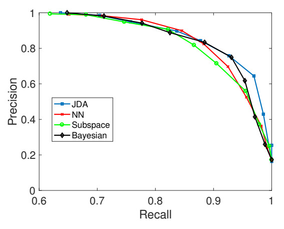
c) Results on Weather Dataset: The weather dataset is collected from Weather Underground[3] which contains a large number of personal weather stations. In our experiments, we study 3 nearby stations in San Francisco and 3 nearby stations in Austin. For each station, we examine 6 sensors including temperature (in °F), humidity (in %), dew point (in °F), wind speed (in mph), wind gust (in mph) and pressure (in Pa). Sensor values are collected every 5-10 minutes, and data collected over 2 years are used in our experiments.

We only show experimental results on the 3 stations in Austin. An explicit discussion of the 3 stations in San Francisco is skipped since these results show very similar trends. We select one station as the target station to evaluate our reconstruction results on. Since there are 3 stations, each sensor type has 3 instances. Due to the spatial proximity of the 3 stations, sensors of the same type are likely to be correlated.

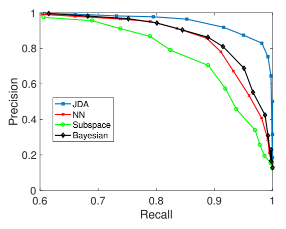
Fig. 2 shows the precision of failure detection on each sensor with recall ranging from 60% to 100%. **JDA** performs the best on all sensors, with a significant margin of improvement on temperature, humidity and dew point. The improvement is less significant on wind speed and wind gust since these signals have relatively large variances and are difficult to reconstruct from other sensor values. When recall is 90%,[4] **JDA** achieves an 8.3% average improvement in precision over all sensors over the second best performer. **Bayesian** performs better than **NN** and **Subspace** on most sensors, showing the benefit of reasoning with multiple sensors. However, **JDA** outperforms **Bayesian** as it captures more nonlinear relationships.

Table I reports on the accuracy of identifying different modes of sensor failures. Here, it is easy to see that **JDA** performs better than **Neural** on most sensors, because **JDA** exploits information from multiple correlated sensors while **Neural** only uses information from a single sensor. **JDA** performs fairly close to **Ground** (except in the case

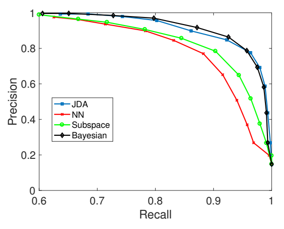
[3]https://www.wunderground.com/
[4]85-95% is a range often used in practice.



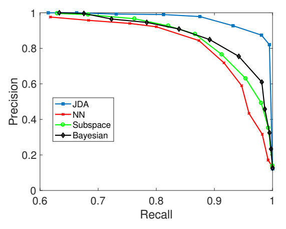
(a) T-kitchen



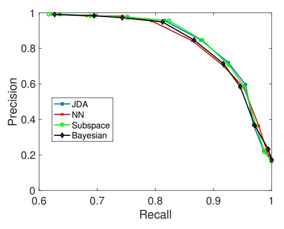
(b) H-kitchen



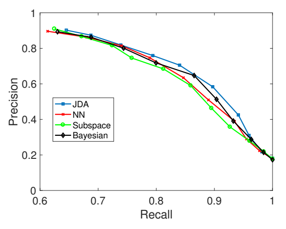
(c) T-living



(d) H-living



(e) T-bath



(f) H-bath

Fig. 3: Precision-recall curves on the appliance energy dataset.

TABLE III: Accuracy of identifying different modes of sensor failures in the appliance energy domain.

Sensor	**Neural**	**JDA**	**Ground**
T-kitchen	91.1	92.5	97.7
H-kitchen	88.7	93.4	96.4
T-living	90.3	91.7	96.8
H-living	87.2	90.6	96.5
T-bath	92.6	93.8	98.0
H-bath	82.4	86.2	94.3

of wind speed and wind gust), highlighting its efficacy in classifying the different modes of sensor failures.

The adaptation performance of **JDA** is given in Table II where **Reference** is computed as the average RMSE between the sensor values of same sensors in nearby stations. Here, the baseline algorithm is to replace a failed sensor with a similar one from a nearby station. **JDA** achieves significantly lower reconstruction errors than **Reference**, especially on sensors with small variances in their readings. The excess error of **JDA** is consistently better than that of **Const**, which validates our claim that dynamic estimation of error intervals is more accurate than static estimation. It is also easy to see that the excess error of **Const** is relatively large compared to the reconstruction error of **JDA**.

d) Results on the Appliance Energy Dataset: The appliance energy dataset consists of 28 sensors measuring energy

TABLE IV: Adaptation performance on sensor data from the appliance energy dataset.

Sensor	Reconstruction Error		Excess Error	
	Reference	**JDA**	**Const**	**JDA**
T-kitchen	1.36	0.72	0.75	0.48
H-kitchen	2.85	1.01	1.32	0.83
T-living	1.69	0.80	0.95	0.66
H-living	3.04	1.12	1.34	0.91
T-bath	0.69	0.73	0.85	0.54
H-bath	10.95	8.19	7.93	6.32

usage, in-house conditions, and outside conditions.[5] Sensor values are sampled every 10 minutes for about 4.5 months. There are multiple temperature and humidity sensors in different rooms. Their physical proximity leads to strong sensor correlations. In our experiments, we used data from all sensors and report reconstruction results on 6 in-house sensors which measure temperature (in °C) and humidity (in %) in the kitchen, living room, and bathroom, respectively.

Figure 3 shows the precision of sensor failure detection by different methods, with recall ranging from 60% to 100%. We observe that **JDA** and **Bayesian** perform better than **NN** and **Subspace** in most cases, demonstrating that sensor relationships are helpful in detecting sensor failures. On the humidity sensors, H-kitchen and H-living, **JDA** achieves significant improvement even over **Bayesian**, demonstrating the benefit of reasoning with a substrate of constraints and nonlinear relationships proposed in our framework. When recall is 90%, **JDA** achieves a 5.2% average improvement in precision for all sensors over the second best performer.

Table III reports the accuracy of identifying different modes of sensor failures by different methods, where **JDA** achieves higher accuracy than **Neural** on all sensors. This is because **JDA** exploits information from multiple sensors while **Neural** only uses information from a single sensor.

Table IV reports the adaptation performance, where the recall is set to 90%. To compute **Reference** for a target sensor, we first find the most similar sensor in terms of sensor values from historical data and then calculate the RMSE between the two sensors in the evaluation data. This can be seen as a simple baseline algorithm for adaptation to sensor failures. **JDA** achieves lower reconstruction errors than **Reference** on 5 sensors. However, on the T-bath sensor, **JDA** performs slightly worse than **Reference** due to overfitting. In terms of excess error, **JDA** consistently outperforms **Const**.

VI. Related Work

Sensor failures can be detected by identifying abrupt changes in the time series of sensor values. This problem is often addressed by change-point or outlier detection methods that have attracted researchers for decades [4]–[7]. A number of detection methods have been developed in the literature, including distribution-based methods [21], [22], reconstruction-based methods [19], [23], [24], probabilistic methods [2], and distance-based methods [18], [25], [26]. Among these

[5]http://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction

methods, the reconstruction-based methods are conceptually the most similar to ours, as they detect failures by checking whether or not sensor values can be reconstructed well. However, most of the existing reconstruction-based methods are unable to identify multiple sensor failures. This prevents them from succeeding at the further tasks of identifying the modes of sensor failures or reconstructing the failed sensor values. Additionally, our framework is capable of reasoning with many different types of sensor relationships, which improves reconstruction accuracy and robustness. While probabilistic methods are capable of identifying multiple sensor failures [27], existing work in this field largely models only linear relationships for tractable reasoning.

Identifying the different modes of sensor failures [11] has also attracted significant research interest. Machine Learning methods have showed better performance than methods originating from the Digital Signal Processing and/or Physics communities [16], [17]. Our approach differs from such existing work in that it exploits information from multiple sensors.

Most of the existing work that studies sensor failures focuses on failure detection and relies on human experts to take subsequent actions. While there exist some methods that reconstruct failed sensor values [28], they are based on the assumption that the failed sensors have already been identified. In contrast, our framework performs joint detection and adaptation, which is more practical. Additionally, our framework provides a way to reliably estimate the adaptation error interval towards guaranteeing a quality of reconstruction.

VII. Conclusions

We presented a novel Machine Learning framework for automatically detecting and adapting to sensor failures. This is important for developing long-lived, survivable software. Our framework is capable of extracting nonlinear sensor relationships from historical data, turning learned relationships into constraints, and then leveraging these constraints to perform joint sensor failure detection and adaptation. Supported by our empirical study, the proposed approach outperforms existing methods that are based on processing information from only a single sensor or extracting linear relationships between multiple sensors. Our future work consists of applying the proposed framework to more real-world applications as well as scaling to larger numbers of sensors.

Acknowledgements

This material is based upon work supported by the United States Air Force and the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-16-C-0045. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force and DARPA.

References

[1] J. Hughes, C. Sparks, A. Stoughton, R. Parikh, A. Reuther, and S. Jagannathan, "Building resource adaptive software systems (brass): Objectives and system evaluation," *ACM SIGSOFT Software Engineering Notes*, vol. 41, no. 1, pp. 1–2, 2016.

[2] E. W. Dereszynski and T. G. Dietterich, "Spatiotemporal models for data-anomaly detection in dynamic environmental monitoring campaigns," *ACM Transactions on Sensor Networks (TOSN)*, 2011.

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[4] M. Basseville, I. V. Nikiforov *et al.*, *Detection of abrupt changes: theory and application*. Prentice Hall Englewood Cliffs, 1993, vol. 104.

[5] F. Gustafsson and F. Gustafsson, *Adaptive filtering and change detection*. Citeseer, 2000, vol. 1.

[6] E. Brodsky and B. S. Darkhovsky, *Nonparametric methods in change point problems*. Springer Science & Business Media, 2013, vol. 243.

[7] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[8] E. Elnahrawy and B. Nath, "Context-aware sensors," in *European Workshop on Wireless Sensor Networks*. Springer, 2004, pp. 77–93.

[9] O. A. Alduchov and R. E. Eskridge, "Improved magnus form approximation of saturation vapor pressure," *Journal of Applied Meteorology*, vol. 35, no. 4, pp. 601–609, 1996.

[10] T. McConaghy, "Ffx: Fast, scalable, deterministic symbolic regression technology," in *Genetic Programming Theory and Practice IX*, 2011.

[11] 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 (TOSN)*, vol. 5, no. 3, p. 25, 2009.

[12] J. Chen and R. J. Patton, *Robust model-based fault diagnosis for dynamic systems*. Springer Science & Business Media, 2012, vol. 3.

[13] H. Marchand, A. Martin, R. Weismantel, and L. Wolsey, "Cutting planes in integer and mixed integer programming," *Discrete Applied Mathematics*, vol. 123, no. 1-3, pp. 397–446, 2002.

[14] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B*, 1996.

[15] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.

[16] A. L. Christensen, R. OGrady, M. Birattari, and M. Dorigo, "Fault detection in autonomous robots based on fault injection and learning," *Autonomous Robots*, vol. 24, no. 1, pp. 49–67, 2008.

[17] P. Przystałka, "Model-based fault detection and isolation using locally recurrent neural networks," in *International Conference on Artificial Intelligence and Soft Computing*. Springer, 2008, pp. 123–134.

[18] E. M. Knox and R. T. Ng, "Algorithms for mining distancebased outliers in large datasets," in *Proceedings of the International Conference on Very Large Data Bases*. Citeseer, 1998, pp. 392–403.

[19] T. Idé and K. Tsuda, "Change-point detection using krylov subspace learning," in *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 2007, pp. 515–520.

[20] T. Dietterich, E. Dereszynski, R. Hutchinson, and D. Sheldon, "Machine learning for computational sustainability." in *IGCC*, 2012.

[21] Y. Kawahara and M. Sugiyama, "Sequential change-point detection based on direct density-ratio estimation," *Statistical Analysis and Data Mining*, vol. 5, no. 2, pp. 114–127, 2012.

[22] K. Yamanishi and J.-i. Takeuchi, "A unifying framework for detecting outliers and change points from non-stationary time series data," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 676–681.

[23] P. A. Crook, S. Marsland, G. Hayes, and U. Nehmzow, "A tale of two filters-on-line novelty detection," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, 2002.

[24] T. Idé and K. Inoue, "Knowledge discovery from heterogeneous dynamic systems using change-point correlations," in *Proceedings of the 2005 SIAM International Conference on Data Mining*. SIAM, 2005.

[25] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2002, pp. 15–27.

[26] S. Budalakoti, A. N. Srivastava, R. Akella, and E. Turkov, "Anomaly detection in large sets of high-dimensional symbol sequences," 2006.

[27] T. L. Dean and K. Kanazawa, "Probabilistic temporal reasoning." in *AAAI*, 1988.

[28] W. Lu, J. Teng, C. Li, and Y. Cui, "Reconstruction to sensor measurements based on a correlation model of monitoring data," *Applied Sciences*, vol. 7, no. 3, p. 243, 2017.