Fixing Faults in Wireless Sensing Systems with Confidence

Nithya Ramanathan, Tom Schoellhammer, Eddie Kohler, Deborah Estrin

Abstract

This paper presents Confidence, a tool for identifying and addressing faults in wireless sensing systems. Confidence pinpoints potential sensor and network faults in real time, allowing users to validate unexpected data and address any failures in the field. By introducing a well defined, low-dimension feature space, and functions to map sensor data into this space, we are able to achieve fault detection and diagnosis with relatively simple mechanisms such as outlier detection. Users can directly modify system outcomes by altering a classification label in instances when Confidence's automated algorithm draws the wrong inference. This label is applied to all similar points in the feature space, enabling Confidence to learn from user interaction in the field. This abstraction for incorporating user knowledge provides a lightweight and easy-to-understand interface for the user, while limiting user burden and reducing the required a priori environmental knowledge. Confidence has performed well on real-world deployments, including one deployment of 130 sensors, replayed datasets, and network simulations. Confidence accurately detects and diagnoses at least 90% of all data, and user interaction improves it's performance.

1 Introduction

Even well-planned deployments of wireless sensing systems (WSS) encounter large numbers of faults that reduce the quantity and quality of collected data [29, 26, 24, 28, 18, 20]. Nodes suffer environmental depredations [26]; peripherals such as sensor boards have bugs; collection networks are unreliable due to low-power radios and environmental factors [31]. Worse, the sensors themselves are often faulty or produce confusing data [16, 26]. These problems lead to data gaps in space and time, which make it difficult to interpret results reliably.

Distinguishing between data that is faulty or simply unexpected is a challenge even in contexts where the environment is carefully characterized. It becomes more difficult for *exploratory* sensing systems, which are deployed precisely because the expected behavior is unknown. A sensor reporting out-of-range values may be miscalibrated or it may be reporting an unexpected environmental phenomenon, such as a very low concentration —we have observed both. Should the user throw out the sensor and its data, or mark them as the most interesting sensor and data in the deployment?

Scientists handle the uncertainty in these environments by taking concrete actions in the field. An on-site user can extract a physical sample for lab analysis or deploy a high fidelity sensor to verify unexpected data. Simply by being at the site when data is collected, human users collect and incorporate contextual information when detecting and diagnosing faults that the system cannot sense. However, in-field validation and analysis is labor- and time-intensive, and only increases with the number of sensors per deployment. While unassisted manual interaction with each element in the system does not scale, a fully automated approach is also likely to fail in exploratory environments where system inputs are inherently difficult to characterize in advance.

We propose a user-centric solution to address faults in exploratory environments.

> *Confidence pinpoints potential faults in wireless sensing systems accurately, efficiently, and in real time, allowing users to validate unexpected inputs and address failures in the field.*

Simple fault detection and diagnosis algorithms suggest actions a user can take to validate or fix potential sensor and network faults. Algorithms rely on an informal model, built using features that provide a user with intuition about the data, long before information is available to build a domain-specific model. In order to refine this initial model, Confidence incorporates data that is proactively collected by the user into system algorithms. By incorporating user feedback, Confidence is able to better adapt to unknown or unexpected environmental phenomena in real time. System design is intentionally as simple as possible to facilitate this user interaction.

We wrote Confidence out of dissatisfaction with the current state of the art in fault detection and diagnosis systems. The most popular of these techniques is to apply static thresholds to features of the data or to features of the network to identify faults [25, 28, 3, 19, 22, 15, 18]. But in order to use this approach in our deployments, we had to manually tweak system parameters for each new environment. This process was not easy for unexplored environments where little or no knowledge is available a priori; it became especially frustrating when a small change in a static threshold resulted in an abrupt and inexplicable increase in the number of false positives or false negatives that the system reported.

In contrast, Confidence is simple and easy to use in the field. It's contributions are in the design choices that make simplicity work, not the mechanisms themselves. We compare Confidence to two static approaches, including Sympathy [19], and find that our user-centric approach detects and diagnoses faults more quickly and accurately than these static systems.

We outline three design goals necessary to implement such a user-centric system.

System transparency for decision support We define a *transparent* system as one that 1) uses mechanisms that are simple to understand and familiar to the user, and 2) makes system reasoning visible to the user. Confidence is a transparent system. We selected a subset of features from those familiar to scientists, and designed a simple process to map data to the space defined by these features. Users are given access to this mapping.

Because users understand the features, a point's location in the feature space can help identify the source of a problem. Therefore, users with little system specific knowledge can still take appropriate action in the field even when Confidence's automated algorithm draws the wrong inference. For example the GRADIENT feature is defined as the ratio of the difference in values to the difference in timestamps for two readings from a sensor. A point that is uniquely located at one extreme of the GRADIENT axis in the feature space indicates that the data reported by that sensor has changed more rapidly than data from similar sensors. Using this knowledge along with context the system may not have, such as a recent irrigation event in a field, a user can decide on the correct course of action.

Incorporate user feedback In order to learn from the actions that a user takes, Confidence refines simplistic models of the environment, using features selected in advance, with feedback from the user. As they learn more about the environment, users can directly update the detection or diagnosis for any data point. *Outcome-based feedback* and system transparency lead to a simple and deterministic relationship between human input and system output. Outcome-based feedback tells the user exactly how their updates will influence system behavior, making it easy to incorporate deployment knowledge back into the system. This approach makes Confidence easier to modify than static systems.

Manage the tradeoff between user burden and system accuracy Solely relying on user feedback to classify data would put too great a burden on the user. Confidence includes automated fault detection and diagnosis algorithms so that users do not have to label every point in the feature space.

The first key contribution of this paper is a multi-dimensional feature space, defined by a small set of features. This space is designed such that sensors that have the same problem, or are operating correctly, produce points that are located close together in the feature space. This trait reduces fault detection and diagnosis to an automated process that accurately classifies at least 90% of data using dynamic thresholds and other simple mechanisms.

The second key contribution is a transparent process that makes it easy for users to provide feedback to the system. Using a channel to directly modify system *outcomes*, users can specify a point in the feature space as being faulty, as being not faulty, or as requiring an action to validate questionable data or address a fault. Nearby points are given this updated label in order to minimize the outcome-based feedback required of the user.

The third key contribution is Confidence itself, which has been deployed with several real-world wireless sensing systems. Confidence's approach applies to faults in environmental sensors and network nodes. We have tested Confidence in the field with two real-world deployments, on replayed datasets, and in simulation. Confidence quickly and accurately detects and diagnoses both injected and real faults in system health and environmental sensors. We show that system accuracy improves over time with limited user feedback.

2 FAULTS IN WIRELESS SENSING SYSTEMS

Confidence's design was inspired by our experiences with exploratory deployments. In this section we provide concrete examples taken from one of these deployments, undertaken in Bangladesh in January, 2006. Examples include data that initially appeared faulty but was actually not faulty, data that initially appeared not faulty but was faulty, and faults with hard to track causes. In all instances, external validation performed while the data was collected was necessary to disambiguate the state of the data and, in some instances, track down the cause of the fault.

We deployed an exploratory WSS in a rice paddy in Bangladesh to help scientists evaluate the relationship between irrigation and arsenic contamination in the ground water [1]. Tens of millions of people in the Ganges Delta drink well water impacted by arsenic. This massive environmental poisoning is projected to cause approximately 3,000 deaths per year [30]. The experiment was designed and deployed with scientists and civil engineers from the Bangladesh University of Engineering and Technology and MIT. We deployed 42 ion-selective electrodes (ISEs) to monitor ammonium, calcium, carbonate, chloride, pH, oxidation-reduction potential, and nitrate, along with 8 soil temperature, moisture and pressure sensors distributed over 3 different depths and locations. The network collected 26,000 measurements over a period of 12 days.

This deployment was short-lived primarily because it relied on ISEs, which were newly applied to in-situ sensing. While extremely useful at uncovering otherwise difficult to observe phenomena, ISEs become unreliable after extended field exposure, making them a good driver for Confidence. Much of our evaluation is based on successfully detecting and diagnosing ISE faults both on historic data traces and in the field.

In order to distinguish between faulty and non-faulty data in the field, scientists identify the expected operational ranges for a sensor during sensor calibration. (These ranges can also be obtained from the datasheet.) The *linear detection range* (LDR) is the high-precision range where the sensor is most sensitive to changes in concentration. The *non-linear detection range* (NLDR) is the low-precision range above and below the LDR where the sensor is less sensitive to changes in concentration. Data outside of the LDR is traditionally thrown out; however, as we shall see below, in some instances this data can still provide useful information.

We describe several examples of faults and potential faults observed in the ISE data. In all graphs, solid horizontal lines delineate the LDR, and dashed horizontal lines delineate the NLDR.

Seemingly Faulty Data is Vindicated The top graph in Figure 1 is a graph of nitrate data collected from 3 sensors in Bangladesh. Although almost all of the data is outside of the linear detection range, an indication that the data is likely faulty, after analyzing soil samples in the lab we determined the data to be legitimate. The nitrate concentration was simply lower than the sensitivity of the sensor, so the readings appeared within the NLDR of the sensor.

Seemingly Non-Faulty Data is Faulty However in some instances data in the NLDR is in fact faulty. The bottom graph in Figure 1 shows chloride data collected from 1 sensor in Bangladesh. Measurements show diurnal variations that resemble other non-faulty sensors. However after analyzing soil samples in the lab we determined that the data was faulty.

Transient Sensor Failures The top and bottom graphs in Figure 2 are taken from the same ammonium sensor at two different times. The top graph is of data collected in Bangladesh where a fault is apparent towards the end of the deployment. In order to reproduce the fault we deployed the sensors after returning (bottom graph) and monitored the data using an early version of Confidence. We discovered the cause of the problem to be a short in the wiring. Data readings revert to within range temporarily after we adjusted the wire at those times indicated by ◯ on the graph.

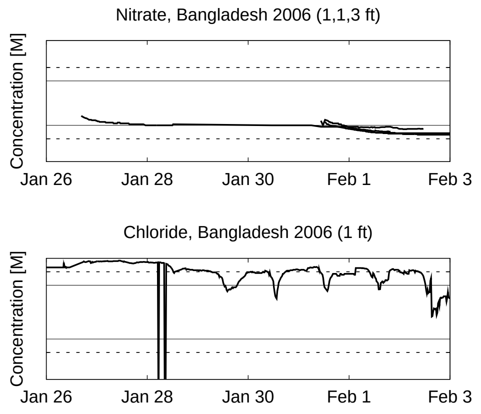

Figure 1: *Data that appears faulty and is good or vice versa* **Top:** Nitrate data taken from three different locations; potentially faulty but lab analysis reveals sensor is good. **Bottom:** Chloride data taken from a single location; Regular diurnal variations lead scientists to believe data is potentially good, but lab analysis reveals data is likely faulty. In all graphs, solid lines delineate the high-precision range and dotted lines delineate the low-precision range.

In summary, many sources of uncertainty present in WSS deployments make it difficult to distinguish between expected behavior and sensor faults. Moreover, minimal resources and short-lived deployments leave users with less time, data, and visibility with which to find and fix faults. Given these characteristics, faults are persistent, and prevent WSS from being deployed at broad scale as a reliable scientific instrument.

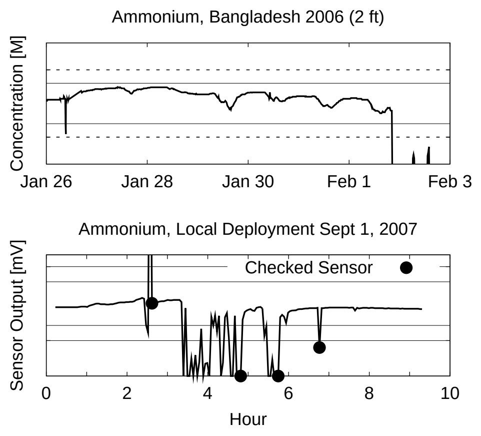

Figure 2: *Actions in the field reveal source of fault.* Top and bottom graphs contain data from the same ammonium sensor. Data in top graph was collected in Bangladesh, and data in bottom graph was collected after returning. Circles in the bottom graph indicate points in time when we checked the sensor and fixed the wiring. Top graph taken from a previous paper [1]

3 RELATED WORK

In this section we discuss systems like Confidence that provide fault detection, fault diagnosis, and/or decision support in order to aid users in dealing with faults in WSS deployments. Fault detection systems define expected behavior; observations that violate the expectation are faulty. Fault diagnosis systems create fault signatures to characterize faults; behavior that matches the signature receives the associated diagnosis. Decision support systems provide information to help a user take the appropriate action.

Hundreds of papers and books have been dedicated to these subjects. We focus on a sampling of systems that have been tested on real-world data. Systems included in this discussion use two common mechanisms to classify faults: rules, decision trees, and thresholds; and machine learning. We discuss each category separately.

3.1 Rules, Decision Trees, and Thresholds

We begin with a discussion of rules, decision trees, and thresholds because they are the most common and simplest approach available to detect and diagnose faults. Szewczyk et al. and Tolle et al. apply static thresholds to sensor data to detect and remove data faults after their deployments at the Great Duck Island and in the San Francisco Redwoods, respectively [25, 28]. Thresholds are defined using domain knowledge of the environment and the sensors, and were only used for a single deployment.

Analyzing *features* of the data, instead of the data, can increase portability of the fault diagnosis system across deployments. Krajewski et al. apply static thresholds to features of sensor data collected from their deployment at an urban sewer to detect and diagnose sensor faults [3]. Data with an anomalous value for a certain feature receive the diagnosis associated by the user. Sympathy [19], MOJO [22], and Tulip [15] use decision trees to analyze network features for WSS, 802.11 networks, and IP networks, respectively.

All of these systems use static thresholds which are difficult to set or modify before or during the deployment [19, 3]. Krajewski et al. found their approach to be most successful only *after* the deployment was completed, when they had access to all of the data and meta-information about the environment. Confidence does not use static thresholds for this reason. However, we select a subset of the features described by Krajewski et al. to characterize sensor faults, and we select a subset of the features used by Sympathy to characterize network faults.

Memento dynamically assigns thresholds to network features extracted from WSS. Dynamic thresholds enable Memento to adapt to different deployments and environments with limited burden on the user [21]. The authors use Chebyshev's inequality to identify outliers because it applies to many different distributions. Confidence uses a normal distribution to dynamically identify outliers in the feature space. While it is more constrained, using a normal distribution fulfills our primary design goals of system simplicity and transparency.

3.2 Machine Learning

Machine learning techniques build dynamic models of expected and faulty behavior in order to adapt to different environments. Systems differ from each other in where they obtain the training data used to build the model.

Kıcıman et al. and Fox et al. design fault detection systems that train on an initial pre-defined period of "normal" operation on a large-scale Internet system [12, 8]. Kıcıman et al. report detecting 89–96% of anomalous behavior correctly. Other approaches use historical traces captured from the system when it is known to be operating correctly to build models for different types of behavior: Magpie [2] characterizes events and resource usage from software components in an IP network [2]; Eskin et al. characterize the expected ordering of calls in an operating system [6]; Hines et al. characterize expected data collected from wired sensor networks [11]. All of these systems assume that training data contains few or no faults, and that training data accurately represents expected behavior. Confidence does not rely on training data.

In order to deal with situations where faults are common, Demiriz et al. [5] and Dara et al. [4] propose systems that train on a dataset labelled by a user. Because labelling a dataset is labor-intensive, like Confidence, these systems reduce the burden on the user by defining a space where similar data groups together. The system only requires a small labelled dataset to automatically associate a label with each cluster before the deployment begins. However, the problem with these approaches is that training datasets are difficult to obtain and label for environments where WSS are deployed. Instead, Confidence allows users to label regions both before and during the deployment.

Larkey et al. [13] and Nath et al. [17] sidestep the problem of obtaining a training dataset by building a model on-line as data is collected. They use Naive Bayes classifiers to build spatial distributions of expected data collected from WSS. However, these systems assume that faults are not common and that statistical spatial relationships between communication neighbors exist.

Most of these machine learning systems are built upon assumptions that cannot be changed once the deployment begins: training datasets should have few or no faults; labelled datasets should accurately represent the expected operating space of the system; or spatial relationships should exist between communication neighbors. These constraints do not hold for the WSS deployments we have undertaken. Confidence's outcome-based feedback makes it easy for users to modify the system's assumptions and algorithms even after the deployment has begun, enabling the system to better adapt to previously uncharacterized environments.

4 System Design

We discuss our system design, beginning with a brief overview of the system.

We designed a transparent feature space, using a small set of features, so that faults can be detected and diagnosed using simple mechanisms. We select features such that, in general, data from sensors that are performing similarly will have similar feature values. More specifically, features are selected and specified such that: 1) non-faulty data points lie close to the *origin* of the space (the point where the value for all features is 0), 2) faulty data points lie far from the origin of this space, and 3) faulty points that lie close together are likely remedied by the same action. Given these properties, Confidence can detect faults using a simple outlier detection algorithm that identifies points that lie anomalously far from the origin. For simplicity we use Euclidean distance as the distance metric in this space. Confidence can diagnose faults using a simple regioning algorithm that groups neighboring points in the feature space. The user assigns a diagnosis, which is an action to be taken by the user, to each region in the feature space. A point is given the diagnosis associated with the region in space where it is located.

The feature space also simplifies the user's interaction with the system. We select a small set of features that are simple to calculate and familiar to the user. For example, the GRADIENT feature mentioned in Section 1 is calculated using the equation $f_{gradient} = |(x_n - x_{n-1})/(t_n - t_{n-1})|$. Here, x_n is the value and t_n is the timestamp for the nth data point from a sensor. Because the features are easy to understand, a point's location in the feature space provides intuition on the source of the problem. Additionally, we use a small set of features in order to create a low-dimensional space. As a result, the space is easy to visualize and easy to manage for the user.

Figure 3 is a diagram of the overall system behavior. Sensors periodically transmit data to a base station. Upon arrival of sensor data, Confidence extracts the features and the resulting *feature vector* is mapped to the feature space. The figure contains a sample two-dimensional space. We use a multi-dimensional feature space in order to identify faults that are uniquely represented by linear combinations of features. We discuss feature selection and design in Sections 4.1 and 4.2. Confidence uses the outlier detection algorithm to dynamically update a distance threshold in the feature space (repre-

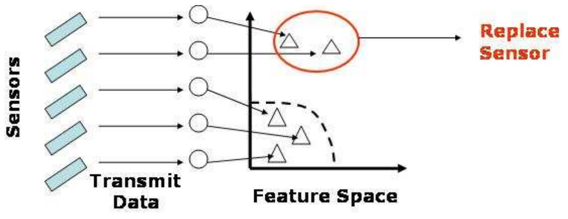

Figure 3: Confidence overview.

sented by the black dotted line in the figure); feature vectors that lie beyond this threshold are faulty. We discuss fault detection in Section 4.3. Confidence groups data using statically designated regions (represented by the solid red line in the figure); and notifies the user of the diagnosis associated with that region of space (represented by the red text in the figure). We discuss fault diagnosis in Section 4.4. Users can update the label for any individual point or region in the feature space at any time. Outcome-based feedback is discussed further in Section 4.5.

4.1 Feature Selection

Features are selected and designed using four guiding principles. 1) The feature should be familiar to the user, easy to calculate, and independently verifiable, in order to give the user intuition about the data. 2) As the feature value increases, the probability that the sensor is faulty should also increase, so that faulty data group far from the origin and non-faulty data group close to the origin. 3) Feature values should be numerically quantifiable so that they can be mapped to the feature space. 4) Only features that meaningfully distinguish diagnoses are included in order to limit the number of features. If two different features are indicators for the same diagnoses in all instances, then only one of the features will be used.

Features that do not fulfill these requirements lead to a system that is less transparent. For example, contrary to intuition and current practice, we did not include spatial redundancy as a feature. Scientists often place sensors close together in order to create redundancy across sensors. This redundancy is used both formally and in-formally to validate a questionable sensor's readings. However we were not able to define a feature to represent spatial redundancy that met our guidelines. It is difficult to identify the minimum distance required between two sensors in order to consider them redundant, or to determine how close together values from two neighboring sensors should be in order to validate each other. Moreover, neighboring sensors can be in vastly different contexts: consider the case when one temperature sensor is in the shade, and a neighboring sensor is in the sun. These two temperature sensors may be spatially close together, but will return different readings. The same issues exist for sensors deployed in soil, which contains complex and heterogeneous structures at very small spatial scales. After considering these issues, we decided not to include a location feature in our feature set.

Confidence uses a separate feature space to classify faults in environmental sensors and a separate feature space to classify faults in network nodes. (For simplicity, we refer to both types of devices as *sensors*, and refer to information received from a sensor as data.) Using the guidelines above, we select three features to define an environmental sensor feature space and three features to define a system health feature space (summarized in Figure 4 and described below).

System Health Features System health features are a subset of those used in Sympathy [19]. These features characterize faults that reduce the quantity of data received at the base station. Like Sympathy, each node in the network is instrumented with code to periodically transmit system metrics to the base station. Confidence extracts a pre-specified set of features from these metrics. These features are summarized in Figure 4, and described in more depth elsewhere [19].

Environmental Features With the aid of domain experts and our guidelines, we selected 3 features from a set of 7 features commonly used by scientists to characterize data quality faults [3]. These features apply to sensors that monitor diffusion phenomena, such as chemical transport in water or molecular movement in air. This large class of environments is characterized by a relatively slow rate of change and an absence of sharp edges. In future work we plan to expand Confidence to include non-diffusion phenomena such as audio or image capture.

GRADIENT*:* Change in value over a period of time. $f_{gradient} = |(x_2 - x_1)/(t_2 - t_1)|$ where (x_i, t_i) is the value and time-stamp for the i'th data point from a sensor. Diffusion phenomena obey physical limits as to how quickly they can change and by how much. Small values for GRADIENT between consecutive readings are typical, due to noise and legitimate environmental variation. Large values for GRADIENT can be a sign of a faulty sensor, a faulty connection, or a battery that has a low voltage.

DISTANCE LDR*:* Distance from a sensor's linear detection range (defined in Section 2). $f_{DistLDR} = max(x - LDR_{upper}, LDR_{lower} - x, 0)$, where LDR_{upper} and LDR_{lower} can be obtained either by calibrating the sensor or from the sensor's datasheet. For example, a humidity sensor has physical limits and should never report humidity above 100% or below 0%. Large values for DISTANCE LDR can be a sign of a faulty or miscalibrated sensor, or of a concentration that is simply outside of the normal detection range of the sensor and requires external validation.

DISTANCE NLDR*:* Distance from a sensor's non-linear detection range (defined in Section 2). $f_{DistNLDR} = max(x - NLDR_{upper}, NLDR_{lower} - x, 0)$. Many sensors have a range where users are less confident of the data but consider it still useful. Large values for DISTANCE NLDR can be a sign of a faulty or miscalibrated sensor.

4.2 Feature Vector Scaling

After a feature value is calculated, it is scaled such that a value of 0 represents non-faulty data, and a value of 10 or more represents almost certainly faulty data. This scaling ensures that no one feature dimension unfairly weights the Euclidean distance calculation used to detect faults. The result is that non-faulty data group near the origin of the feature space and faulty data group far away from the origin.

Feature	Description	Scaling
Environment		
GRADIENT	Change in value divided by the change in time	$\max(0, \log_2(X/16))$
$max(0, \log_2(X))$		
DISTANCE LDR	Distance from the linear detection range	$max(0, \log_2(X))$
DISTANCE NLDR	Distance from the non-linear detection range	$\log_2(X)$
System Health		
NODE DEADNESS	Time elapsed since node *n* heard from by any other node	$\max(0, \log_2(X/8))$
APPLICATION DEADNESS	Time elapsed since base-station received application data from node *n*	$\max(0, \log_2(X/16))$
CONGESTION	Ratio of CRC errors to non-corrupted packets received from node *n*	$10X$

Figure 4: Summary of features used for both environmental data and system health data, with associated scaling functions.

Scaling functions need not be exact. The system can adapt to a large range of scaled values (discussed in the evaluation). This flexibility is available because each feature value is not mapped to an explicit state of faulty or not faulty. Instead, Confidence's fault detection algorithm detects the boundary between the non-faulty data, clustered near the origin, and the faulty data, clustered farther from the origin.

We designed Confidence's scaling functions to be as simple as possible. The system first takes a log base-2 transformation of the value. This works because for all of our features, as the value increases smaller changes in value are less important. A log transformation scales data to match this intuition. In instances when feature values still do not match the 0 to 10 mapping, we divide the value by a constant, *S*, in the scaling function $max(log_2 X/S, 0)$. *S* is chosen for each feature such that $log_2 X < S$ for most non-faulty values of the feature X. The scaling function for each feature is listed in the right column of Figure 4.

While, in theory, each class of environmental sensor needs its own scaling function for each feature, in our experience with over 15 different types of sensors, we are able to use the same scaling constants for most sensors and features. We explore this further in the evaluation.

4.3 Fault Detection

In order to detect faulty data, Confidence dynamically calculates a threshold to separate faulty and non-faulty data in the feature space. Feature vectors that lie beyond this threshold in the feature space are classified as faulty.

Confidence builds and continuously refines a distribution of the distances of *non-faulty* feature vectors. We assume these distances are normally distributed because this distribution is simple to update in real time and transparent to users. Updating the distribution consists of maintaining running estimates for the moving average (μ) and standard deviation (σ). Points that lie outside the threshold $\mu + 2\sigma$ are labelled faulty, and are not used to update the distribution.

Confidence uses an Exponentially Weighted Moving Average (EWMA) to update parameters. A EWMA slowly decays the weight of older points, enabling Confidence to better adapt to dynamic environments. Static environments are not adversely impacted by the use of a EWMA. We set the EWMA parameter, α, to a standard value of .9, and incorporate the distance of the *N*th feature vector, d_N, into the current moving average, μ_N: $\mu_N = (1-\alpha)d_N + \alpha\mu_{N-1}$. σ_N is similarly updated using σ_{N-1} and a running estimate of the standard deviation, σ, calculated using a standard equation: $\sigma = 1/N \times \sqrt{N \times (\sum_{i=1}^{N} d_i^2) - (\sum_{i=i}^{N} d_i)^2}$

When the system is initialized, μ and σ need to be bootstrapped. During an initial phase, P_I, feature vectors with distances from the origin greater than a static threshold, D_I, are considered faulty. D_I is set to 5, the midway point between the minimum and maximum scaled values for features. Feature vectors with distances less than D_I are considered not faulty, and used by Confidence to initialize the distribution parameters.

The disadvantage of assuming a normal distribution is, of course, that in many instances highly correlated faulty behavior appears not faulty. However, as we discuss in the evaluation, this approach adapts to a wide range of scenarios and is robust to a variety settings for D_I and P_I, even when over one half of the dataset is faulty or when faults occur during the bootstrap phase.

4.4 Fault Diagnosis

Confidence suggests actions a user can take to remediate or further elucidate potentially faulty data. We design the feature space such that feature vectors in a similar region are likely to be addressed by the same action or actions. The simplest approach to group similar feature vectors in the multi-dimensional feature space is to divide the space into symmetric regions. Confidence defines *R* regions in each dimension of the feature space. For an *N*-dimensional space, R^N regions are defined. Because *N* is automatically set by the number of features, *R* is used to control the number of regions in the space.

Many values of *R* result in acceptable system performance, and we evaluate the tradeoffs in setting this parameter in Section 5. Setting *R* to 3 or higher provides acceptable results, so we set *R* to 3 to minimize the number of regions the user has to label.

The user can assign an action or set of actions to each region at any point in time. For our deployments, we initialized the environmental feature space with one of four possible actions for a user to take: 1) validate questionable sensor data, 2) re-calibrate a sensor, 3) replace a sensor, or 4) replace batteries. We assigned actions to regions with the help of domain experts and past deployment experience. These actions are refined based on user feedback (described below). The user is notified of the action(s) associated with the region where the faulty feature vector is located.

4.5 Outcome-Based Feedback From Users

Confidence's performance improves with feedback from a user. The system design enables two modes of user feedback to Confidence's fault detection and diagnosis algorithms.

First, the user can provide information to refine the detection and diagnosis algorithms. Although the feature space makes fault detection simple, it incorrectly assumes that faulty data are always located far from the origin. In our experience, instances arise when data located far from the origin are not faulty, or vice versa. This scenario could arise due to unexpected environmental conditions: For example an unusually heavy rain may lead to unexpectedly low concentrations of certain ions in the soil (further examples provided in Section 5.6).

The use of regions in the feature space makes it easy for users to rectify these problems. The user begins by requesting a snapshot of the feature space. For each region in the space, the snapshot includes 1) the feature vector located at the region's center, and 2) the list of the most recent data points associated with that region, each point's feature vector, and if the point is faulty or not. Users can manually assign labels to specific regions, updating either the action associated with that region or the fault state that should be associated with data in that region. Data can either be from a specific type of sensor or all sensors associated with that region. We evaluate the impact of this kind of interaction on detection accuracy using data from a recent deployment in our evaluation.

Second, the user can provide information to refine the calculation of feature vectors, which feeds into the detection and diagnosis algorithms. Data that generate feature vectors associated with regions in the middle of the feature space are not definitely faulty, and require further validation. Upon notification, the user can validate readings from a sensor and inform Confidence if the reading is valid or not. If the reading is not valid, the user needs to further investigate the problem. If the reading is valid, the user informs Confidence. Confidence uses this information to expand the detection range for this sensor. One of the reasons we selected DISTANCE LDR and DISTANCE NLDR was to facilitate this kind of interaction.

The user provides feedback to Confidence through a command line interface, and obtains information about the system in two log files.

4.6 Discussion

Our current feature space does not detect *stuck-at-faults* that occur inside the detection range of a sensor. Data from a sensor that is "stuck" at a certain value has a GRADIENT of 0, which is considered non-faulty in our current feature space definition. One way to address this situation would be to introduce a component of time into the feature space, because a GRADIENT of 0 for a short period of time likely indicates non-faulty behavior, but a GRADIENT of 0 for a long period of time likely indicates a fault.

5 EVALUATION

Our performance hypothesis is that 1) the system correctly detects and diagnoses at least 90% of all data in a wide range of deployment scenarios; 2) system accuracy improves when a user incorporates outcome-based feedback; and 3) Confidence performs better than common thresholding techniques, with less burden on the user. We show that Confidence detects and diagnoses most faults, in many different deployment scenarios, quickly and with few false positives and negatives, even when over one half of the data are faulty. In addition, we evaluate Confidence's sensitivity to values of various system parameters.

5.1 Methodology

Our primary metric for system performance is the fraction of non-faulty and faulty data that is correctly detected and diagnosed. We evaluate system performance in addressing injected and actual sensor faults in real-world deployments, real faults in replayed data-sets collected by exploratory sensor deployments, and injected network faults in simulation.

To use accuracy as our metric, we need to know what data is truly faulty, what data is truly not faulty, and when any faults occurred. In other words, we need access to ground truth. In a simulation, ground truth is precise because we can inject a known fault at a known time into the simulation. Attaining ground truth in the field, especially for environmental data, is not so straightforward. Our exploratory sensing deployments collected data about environments where little is known about the exact chemistry and daily biological reactions. The scientists did not know what to expect from the data, and in fact data from our deployments revealed diurnal variations in ammonium and carbonate concentrations in Bangladesh, and in nitrate concentration in the San Joaquin River, that were previously unexpected. In these instances, an approximation to ground truth is achieved using a combination of manual analysis by domain experts, results of physical samples that were extracted during the deployment and analyzed in a lab, and post-deployment analysis and calibration of sensors. We treat the results of this analysis as ground truth when evaluating accuracy for environmental sensor faults.

We describe our methodology in evaluating Confidence's performance in detecting and diagnosing sensor faults and network faults separately.

Sensor Faults Environmental sensor faults are difficult to simulate authentically, so we use replayed data sets collected from past deployments. However, the validation process we describe above is a laborious task, so even these data sets are rare.

We use a subset of data from two deployments for which we can systematically detect and diagnose faults using our ground truthing process. The first data set is from a deployment undertaken in Bangladesh in January, 2006 (described in Section 2). We use 15,000 points collected over the course of 2 weeks from 33 sensors (30 ion-selective electrodes, 3 temperature sensors). Of the 8,000 points that are known to be faulty, we are able to assign confirmed diagnoses to 4,000 points. The second dataset is from an ongoing deployment at James Reserve (JR) initially set up in October, 2005. The purpose of this deployment is to explore the spatial and temporal scales at which sub-surface measurements should be taken, and to study the relationship between soil CO_2 fluxes, moisture, and temperature conditions in the soil. We use 35,400 points collected over the course of a day from 130 sensors, 3800 of which are known to be faulty. We could not confirm the diagnoses for any of the faults that occur at JR although the faults themselves were confirmed by domain experts. This information is summarized in Figure 5.

In order to simulate scenarios beyond these two deployments, we vary system parameters and re-run Confidence on our two datasets.

We evaluate the value of outcome-based feedback by quantifying fault detection accuracy with and without user interaction for the deployment at JR.

We evaluate Confidence's ability to manage system accuracy versus user burden by comparing Confidence with two common thresholding techniques used by scientists to identify faults.

We also deploy Confidence in the field with two real deployments: one in the San Joaquin River Valley, and one at the James Reserve (discussed in Section 5.6).

Network Faults To evaluate system performance in addressing network faults, we compare Confidence with a previous similar system, Sympathy [19], in simulation. We simulate different deployment scenarios by varying network protocol parameters and topologies in order to validate the first part of the performance hypothesis. Simulations are run on a 25 node network, where each node transmits a system health packet to the sink once every 3 minutes and environmental data from each sensor once every 3 minutes. Simulations were performed in EmStar [9], and parameters were set to reflect a realistic wireless sensing system deployment. After an initial start up period, fail-stop faults are injected into either a single node or multiple nodes in the network at some random time. For a single fault, we ran 24 simulations with faults injected into different nodes. For multi-fault simulations, a test consisted of injecting a set of faults simultaneously. Sets range in size from 2 to 9 nodes, and each test is repeated 5 times.

5.2 Detection and Diagnosis Accuracy for Sensor Faults

Base Performance We begin by establishing a point of comparison for subsequent experiments by quantifying system accuracy when running Confidence with default parameter settings on datasets from Bangladesh and James Reserve. System parameters are set to their default values: the fault detection EWMA α is set to 0.9; initialization distance and duration D_I and P_I are set to 5 and 300, respectively; the number of regions in each dimension of the parameter space, R, is 3; and feature selection and scaling factors are as defined in Figure 4. The system meets our baseline performance constraints.

	Bangladesh	**JR**
Number Sensors	33	130
Faulty / Total # Points	8000 / 15000	3800 / 35400
Non-Faulty Detected	98%	90%
Faulty Detected	94%	83% / 99.9%
Faulty Diagnosed	92%	NA

Figure 5: Confidence fault detection and diagnosis accuracy using default parameter settings.

Of the 15,000 points from Bangladesh, Confidence correctly detects 98% of non-faulty points and 94% of all faulty points. Of the 4,000 faulty points with known diagnoses, Confidence assigns the correct diagnosis to 92%. Figure 5 summarizes the results and introduces labels for these three categories used in all subsequent plots.

Of the 35,400 points from James Reserve, Confidence correctly detects 90% of non-faulty points and 83% of all faulty points, with an overall detection accuracy of 89%. Most of the faults that Confidence does not detect occur at a node which returns a value of 0 for all sensors for limited time. This occurred because the software on the node returns 0 in a fault condition; unfortunately 0 is a valid reading for many sensors. When we manually modified the data set to reflect a fix in the software to return -10000 instead of 0, detection of faulty data improves from 83% to 99.9%.

In subsequent sections, we only provide results from Bangladesh due to space constraints. Confidence's performance on the dataset from James Reserve is similar to performance on the dataset from Bangladesh.

Feature Selection We evaluate our feature set by comparing detection and diagnosis accuracy when using any combination of 1, 2, or all 3 features. All of the features Confidence uses are necessary to meet our baseline performance constraints, though each feature is not equally important in detecting and diagnosing faults.

Figure 6 contains two bar graphs. Each bar represents one run of Confidence using a subset of features, and tics on the x-axis are labelled with the features that were used for that test. The top plot quantifies the amount of faulty and non-faulty data Confidence correctly detects, the bottom plot quantifies the amount of faulty data Confidence correctly diagnoses. Detection performance of non-faulty data is not impacted as long as at least one feature is used, so all performance differences can be attributed to differences in identifying faulty data.

Not all features contribute equally to the detection and diagnosis of faulty data. DISTANCE LDR is the most important feature in detecting faulty data. When the DISTANCE LDR feature is excluded (3rd bar on the plot), detection accuracy drops to 83%, and diagnosis accuracy drops to 66%. In other words, for 17% of all data and 26% of faults with known diagnoses, since the GRADIENT and DISTANCE NLDR features have normal values, DISTANCE LDR is the only feature that can be used to detect and diagnose these faults. Excluding DISTANCE NLDR (4th bar on plots) does not impact fault detection. This is expected because the DISTANCE LDR feature is always greater than or equal to the DISTANCE NLDR.

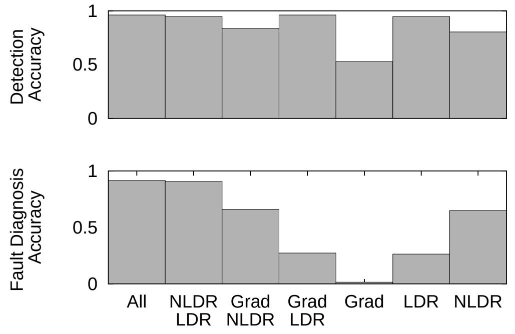

Figure 6: Detection and diagnosis accuracy achieved when Confidence uses every possible subset of features to define the feature space. For the X-axis, **All** means that all features were used in detecting and diagnosing faults, **NLDR** refers to DISTANCE NLDR, **LDR** refers to the DISTANCE LDR, and **Grad** refers to GRADIENT.

However, diagnosis accuracy drops to 26% when DISTANCE NLDR is not used.

GRADIENT is the least powerful of the features for our datasets. When it is the only feature used, 52% of faulty and non-faulty data are correctly identified, and only 2% of faults are correctly diagnosed. Excluding GRADIENT reduces detection accuracy by only 2%, and drops diagnosis accuracy by only 1%. While not impacting performance significantly for these datasets, GRADIENT is necessary to detect and diagnose certain types of faults, as we will see in our deployment experiences.

Feature Scaling Confidence relies on domain expertise to set the scaling constant, S, for each feature/sensor pair. Similar to the case where a feature is excluded from the feature set, scaling constants impact detection and diagnosis accuracy because they control each feature's weight in the distance calculation and the region of space where the feature vector is located. We evaluate the sensitivity of detection and diagnosis algorithms to the scaling constant, and find that system performance is robust to a wide range of constants.

Figure 7 contains representative results from our simulations. We plot detection and diagnosis accuracy on the Y-axis and the setting for the inverse scaling constant ($1/S$) on the X-axis. As $1/S$ increases, so does the scaled feature value. Each line in the plot corresponds to a set of tests where a constant group of features was scaled by the value on the X-axis. The top two lines correspond to detection and diagnosis accuracy when scaling DISTANCE LDR, and the bottom line corresponds to the detection accuracy when scaling DISTANCE LDR and DISTANCE NLDR together. In all instances, as $1/S$ falls below 1 for a feature, the system behaves as if that feature were not included. As 1/S increases, detection and diagnosis accuracy are not impacted. Detection accuracy is not impacted because the threshold is dynamically calculated and can adapt to the changing spatial distribution of faulty data. We initially expected diagnosis accuracy to suffer because the regions are statically assigned. However, diagnosis accuracy is not significantly impacted because, even in the extreme case, all faults receive the action associated with the most outer region. In most instances, this action is the same as the action assigned to the neighboring region in the space.

As users update the feature space with different actions, diagnosis accuracy will be impacted if 1/S is set inconsistently with the region labeling. We ignore this case because the user assigns both the scaling constants and the actions in the regions, so both should be consistent.

We also evaluate performance when the log base used to scale feature vectors is varied; any log base less than 4 results in acceptable performance.

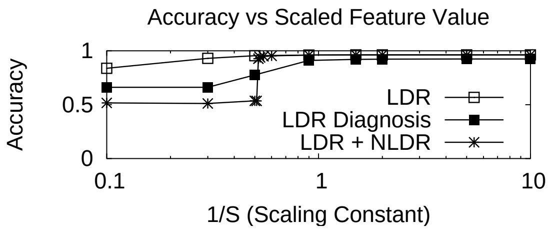

Figure 7: Detection and diagnosis accuracy when varying scaling factors applied to DISTANCE LDR individually, DISTANCE NLDR individually, and both at the same time.

Fault Detection We continue our sensitivity analysis on fault detection parameters (α, D_I and P_I). We vary settings for these parameters and quantify detection accuracy. (We do not quantify diagnosis accuracy because these parameters do not impact the diagnosis algorithm.) α and P_I impact detection accuracy when spatial distributions of faulty and non-faulty data change over time. Data distributions remain relatively stable for our datasets, so detection accuracy is not significantly impacted by these parameters (plots not shown). D_I is varied in integer steps from 1 to 10.

Confidence meets our performance constraints as long as at least 50% of faulty feature vectors lie farther than D_I units from the origin of the feature space. In our Bangladesh dataset, 50% of faulty feature vectors are 9 or more units away from the origin. So Confidence accurately classifies most data if D_I is set to 9 or less for this dataset.

Confidence does not impose assumptions on the frequency of faults. However, there is a dependence on where the faulty feature vectors lie in the feature space.

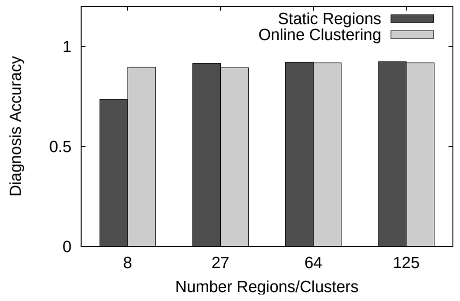

Figure 8: Diagnosis accuracy with varying number of regions or clusters in the feature space.

Fault Diagnosis We continue our sensitivity analysis for the fault diagnosis algorithm. We evaluate the impact of 1) the number of regions in the feature space, and 2) the choice of static regioning versus on-line clustering on diagnosis accuracy. (We do not evaluate detection accuracy because the number of regions does not impact the detection algorithm.)

We begin with an evaluation of the number of static regions. As discussed in the architecture section, the number of regions is set to N^R where N is the number of features in the feature set, and R is the number of divisions in each dimension. We vary R from 2, the minimum number of divisions, to 5, producing from 8 to 125 regions in the space. System transparency drops as the number of regions increases.

The darkly shaded boxes in Figure 8 represent the relationship between diagnosis accuracy and the number of static regions in the feature space. Diagnosis accuracy is not impacted when R is set to at least 3, with 27 corresponding regions in the 3-dimensional feature space. As the number of regions falls below 27 dissimilar feature vectors are grouped together, resulting in an increase in the number of incorrect diagnoses. As the number of regions increases beyond 27 accuracy is not initially impacted, but similar feature vectors are more likely to be separated across multiple regions. In this case, users will have to label and update more regions, making diagnosis inefficient and the system less transparent.

We also evaluate performance when using an online variant of K-means clustering to group similar data in the feature space. K-means is a lightweight, online clustering algorithm commonly used by systems to classify faults [14, 10, 7, 27]. Our implementation is largely based on an on-line version of K-means clustering described in [23], which is one of the few algorithms to meet our criteria. The lightly shaded boxes in Figure 8 represent diagnosis accuracy as we vary the initial number of clusters. As the number of initial regions is set to 27 or more, static regions performs as well as well as online clustering because cluster movement drops and clusters essentially mimic static regions. However, clustering also performs well even with fewer clusters. This is likely because the clusters move around in the space, adapting their location to the data (unlike the static regions).

While clustering performs as well or slightly better than static regioning, we chose the latter because static regions lead to a more transparent system. A user assigns an action to each region of the feature space. Static regions do not change location in the feature space, unlike clusters, so the relationship between user updates and expected outcomes is deterministic and transparent to the user.

5.3 Network Faults

We quantify Confidence system performance in detecting and diagnosing network faults in a range of different deployment scenarios: a simulation with a single injected fault, a simulation with multiple injected faults, and simulations where network and application parameters are varied to simulate different types of deployments. We can evaluate both fault detection and fault diagnosis accuracy because we control the simulation. A *false positive* is either a notification of a fault at a non-faulty node, or a notification of the incorrect diagnosis at the faulty node. We can also evaluate detection latency in this context because we have access to the time the fault is introduced into the network. Detection latency is important for systems that aim to enable real-time user interaction. We run Confidence and Sympathy in simulation and compare their accuracy and latency.

Confidence reports significantly fewer false positives than Sympathy in each simulation. The top graph in Figure 9 is a histogram of the number of false positives that occur in each simulation. Some of the false positives generated by Sympathy occur because the decision tree and thresholds are statically defined. Correlated behavior that initially appears faulty triggers Sympathy's decision tree, but does not appear anomalous in Confidence's feature space. Furthermore, Sympathy's attempt to provide precise diagnoses is more often wrong than Confidence's more broad action-based diagnoses.

Confidence correctly identifies faults in less than half the time it takes Sympathy. The bottom graph in Figure 9 is an empirical cumulative distribution function (CDF) of the detection latency of faults for all 24 runs. Sympathy takes longer to classify faults because 1) Sympathy obtains system health metrics, such as the neighbor list, from the routing layer, which is subject to its own timers; and 2) faults must persist in Sympathy for 540 seconds (a hard threshold set in the fault tree) before they are identified. The system designers of Sympathy note that decreasing this timer may improve detection latency, but increases false positive notifications [19].

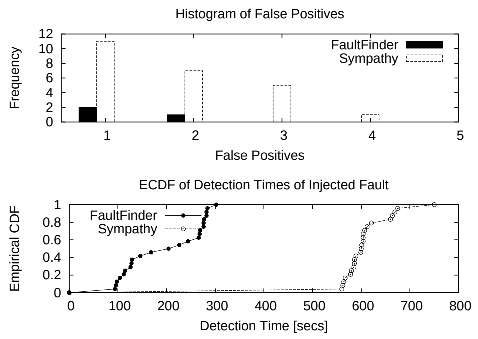

Figure 9: When compared to Sympathy, Confidence produces fewer false positives (top graph) and detects faults more quickly (bottom graph).

We evaluate Confidence's ability to adapt to different deployment scenarios by changing network and application parameters, and injecting faults in up to a third of the network. We vary the periodicity of routing beacons from 10 to 100 seconds and application data periodicity from 2 to 10 minutes. Periods are bounded above by simulation lifetime. Detection latency is not impacted, and the average number of false notifications reported in one simulation doubles (from .45 to .9) when beacon periodicity increases by a factor of 10 and application data increases by a factor of 5.

We evaluate Confidence's ability to correctly classify faults when multiple faults are injected into the system. When up to a third of the network is faulty (i.e. 9 faulty nodes in a network of 25 nodes), Confidence detects most of the injected faults with few false positives. Figure 10 is a graph of the number of faults correctly detected divided by the number of total injected faults, averaged over all runs. As expected, as the number of faults injected increases (plotted on the X-axis), the percent of faults correctly detected decreases (plotted on the Y-axis). In all instances, Confidence never reports more than 3 false positives. Confidence's dynamic calculation of the distance threshold in the feature space enables Confidence to automatically adapt to the different scenarios. Sympathy was not designed to detect more than one fault and performs significantly worse.

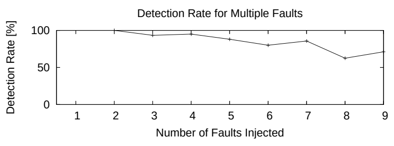

Figure 10: Confidence detects most faults with few false positives when up to a third of the network is faulty.

5.4 Incorporating Outcome-Based Feedback

To validate that system accuracy improves when a user incorporates outcome-based feedback (the second part of the performance hypothesis), we quantify fault detection performance with and without user feedback using replayed data collected at JR. We use the dataset from JR because we had deployed Confidence with this deployment and took actions in the field to validate potential faults. Confidence correctly classifies 90% of non-faulty data without user interaction, and correctly classifies almost 100% of non-faulty with outcome-based feedback from the user.

The top plot in Figure 11 is a snapshot of the feature space containing data from the deployment. We project the data onto two (of the three total) dimensions: GRADIENT and DISTANCE LDR. The plot contains data that Confidence initially classified as faulty, and our outcome-based feedback to the system. We used the three dimensional version of this snapshot in the field to identify actions to take during the deployment. We describe the actions we took in the field and our use of outcome-based feedback to improve system accuracy.

Confidence first notified us of faulty soil moisture data that occurred outside of the LDR for those sensors; data is represented by red Xs in the figure. After taking a soil sample we determined that the environment contained no moisture, so the sensors were operating outside of their expected sampling range. We updated the label for soil moisture data in this region as "not faulty" (updated label shown in red text in the figure). When we re-run Confidence on the replayed dataset with this updated label, Confidence classifies an additional 8% of non-faulty data correctly.

Confidence also notified us of faulty PAR (i.e. light) data that had high gradients: data is represented by grey triangles in the figure. After physically observing the sensors, we discovered that the sensors were located directly under a tree. Because it was an extremely windy day the tree branches moved back and forth, causing the sensors to perceive extreme changes in light intensity over short periods of time. We updated the label for PAR data in this region as "not faulty" (updated label shown in grey text in the figure). Confidence classifies an additional 1.9% of non-faulty data correctly when we re-ran the system on replayed data from the field with this updated label. We were unable to improve the incorrect classification of the final 600 faulty points because these points clustered near the origin of the feature space, near points from the same sensors that were not faulty.

Confidence's static regions accurately group similar data, so a single label can be applied to a cluster of data. As a result, in many instances system accuracy improves significantly each time the user updates a single label. The bottom plot in Figure 11 contains the same snapshot of the feature space with the addition of Confidence's static regions. In most cases, similar types of faults group together in the same region. Therefore a single updated label is accurately generalized to the group of similar data, leading to improved system accuracy.

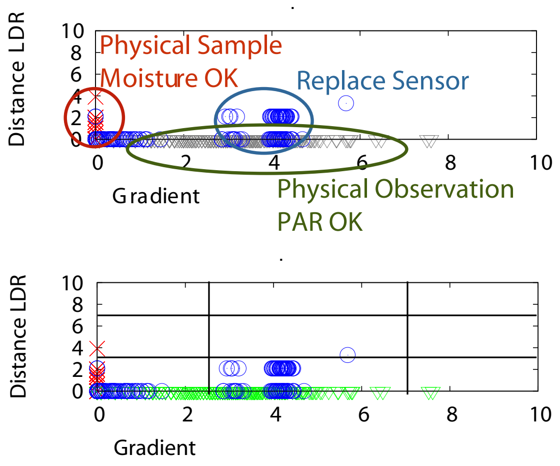

Figure 11: Both plots are snapshots of a Confidence feature space with data from the deployment at JR projected onto two (of the three total) dimensions: GRADIENT and DISTANCE LDR. The top plot contains data that Confidence initially classified as faulty, and our outcome-based feedback to the system.

Thus, preliminary results indicate that outcome-based feedback can improve system accuracy – in this deployment, 10% fewer false positives – with limited work by the user. Future work will include running Confidence in more deployments in order to evaluate performance in the presence of more heterogeneous behavior.

5.5 Comparison to Thresholding

To validate that Confidence performs better than common thresholding techniques with less burden on the user (the third part of our performance hypothesis), we compare Confidence to two common thresholding techniques. Rejecting data that falls outside of a threshold is one of the easiest, and therefore most commonly used, outlier detection methods. With limited feedback from the user, Confidence more accurately classifies data than these thresholding techniques, even when thresholds are manually tuned to maximize detection accuracy. We use the data collected during our James Reserve deployment because we took actions in the field and incorporated feedback into the system. Therefore, we can quantify user feedback to Confidence (in the form of updating region labels) and detection accuracy as described in the previous subsection.

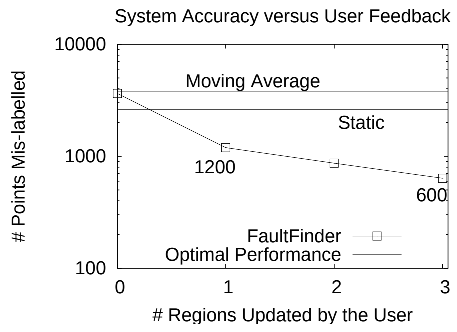

Figure 12: Plot of number of points that have been mis-classified by one of three techniques: Dynamic thresholding, static thresholding, and Confidence. The two horizontal lines in the plot correspond to the performance achieved by the thresholding techniques with optimal parameter settings; the third line corresponds to Confidence's performance as the user interacts with the system.

We implement two common thresholding techniques and manually tune parameters in order to identify the thresholds that achieve the best detection accuracy for each sensor. The first approach applies static ranges to data values. We begin with ranges assigned by domain experts. (These are the ranges used by Confidence to calculate the DISTANCE LDR for each sensor.) We then adjust each range by small increments to find the range which correctly classifies the most data for each sensor. The second approach dynamically calculates a range using running estimates of the mean (μ) and standard-deviation (σ): $Range_{upper} = \mu + N\sigma$, and $Range_{lower} = \mu - N\sigma$. We identify the value of N that achieves the best detection accuracy.

The plot in Figure 12 summarizes our results. The two horizontal lines correspond to the performance achieved by the best possible threshold value for both thresholding techniques; the final line corresponds to Confidence's performance as the user updates region labels.

User feedback lets Confidence achieve better accuracy than both thresholding techniques. We tuned the thresholding techniques once we had access to all of the data and ground truth, but this approach was still more difficult than using the outcome-based feedback channel to update region labels. Instead of attempting to modify thresholds or add new, possibly conflicting, rules to the system, Confidence users simply update the label for a data point. Because the feature space accurately groups similar points together, the system can generalize this label to all neighboring points. This approach makes it easy for users to identify and correct a large number of corner cases in the field. Moreover, users can visually see the results of their interaction immediately in the feature space, leading to a more transparent and intuitive interface.

5.6 Deployment Experience

Scientific deployments differ significantly from indoor or outdoor testbeds because scientific goals supersede all others, introducing an unexpected set of challenges and faults that are otherwise ignored. Therefore, we evaluated Confidence in the field with two real-world WSS deployments, a deployment in a riverbed in the San Joaquin River Valley and a deployment in a forest in the James Reserve. This second deployment is separate from the JR deployment described in the previous subsections. We used Confidence to detect real sensor faults and injected sensor faults in the field. Our experience with using Confidence has been positive.

5.6.1 San Joaquin

In our first deployment, we accompanied a group of scientists to install a WSS at the confluence of the Merced and San Joaquin River. The group's scientific goal was to better understand how the agricultural runoff that pollutes the San Joaquin river impacts the riverbed just past the confluence. We deployed 14 ammonium and nitrate ion-selective electrodes and 7 temperature sensors in the soil. We systematically validated data collected from all 21 sensors to ensure that Confidence did not miss any faults, and did not direct us to take any unnecessary actions in the field. Detection and diagnosis accuracy are important because in-field actions are time and labor intensive. Confidence correctly detected and diagnosed all faults, with no false positives.

We took two steps to validate data collected from sensors. First, we deployed a redundant set of sensors to shadow the scientist's main deployment. However, the heterogeneous nature of soil makes it virtually impossible to provide true measurement redundancy in these environments. Even temperature has been observed to differ by several degrees in underground measurements located within a foot of each other. So we took a second step to further validate sensor readings: We periodically extracted water samples from each sampling site to obtain an independent measurement of the ammonium and nitrate concentrations. Water samples were analyzed using a Hach Kit, which is a mobile spectrometer designed to analyze samples in the field. Because pollutant concentrations can vary during the day, we extracted samples at the beginning and end of each of the five days from all seven sampling sits. Extracting and analyzing even a single sample requires two people working in parallel for half an hour, so our six person team spent most of the deployment either validating sensor data, deploying sensors, or testing sensors.

We describe two of the faults. The first fault was a bad nitrate sensor. Confidence notified us to validate the sensor because the value for GRADIENT was anomalously high compared to all other sensors in the deployment, and data from the sensor occasionally fell outside of the LDR. The scientist's initial intuition was that Confidence was wrong and the sensor was OK because much of the nitrate data fell within the operating range of the sensor. We extracted a water sample to validate the data, and confirmed that the data was actually faulty. The validation point and sensor data are shown in Figure 13. In the graph the small points correspond to sensor readings, and the large □ corresponds to the results of the physical sample. Upon removing and re-calibrating the sensor, we discovered that the sensor was not completely dead, but that the membrane had lost much of its sensitivity and needed to be replaced.

A second fault was a temperature sensor with a disconnected wire. The temperature data was within the operat-

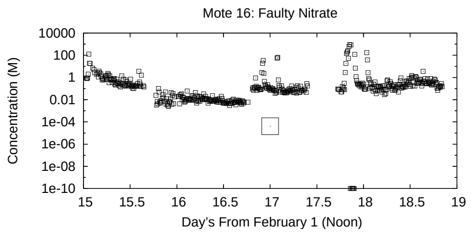

Figure 13: Nitrate readings generated by an ISE. Confidence identified this sensor due to a large value for GRADIENT. Samples used for lab analysis of nitrate (large square) show large discrepancies compared to the ISEe. The ISE was determined to be broken when calibrated after the deployment.

ing range of the sensor, but values for GRADIENT were anomalously high. Confidence notified us to check the sensor, and upon inspection we found that the sensor's ground wire had become disconnected. With the ground disconnected the ADC pin was floating, producing wild variation in readings. After reconnecting the wire, we notified Confidence of the action we had taken. The data returned to normal once we made the fix.

Confidence accurately classified data from 10 ISEs and 6 temperature sensors as not faulty. We verified that the data from the ISEs were not faulty by comparing the readings to the physical samples extracted during the deployment.

5.6.2 James Reserve

We also deployed Confidence in the field with our deployment at James Reserve. We have already described this deployment in the beginning of this section. Over the course of two days, we injected faults into sensors to further validate Confidence's detection and diagnosis accuracy. We chose sensors that were easily accessible, and replaced them with one of our own sensors so as not to damage deployed sensors collecting critical scientific data. In order to inject faults we moved sensors to an extreme micro-climate specific to the sensor. We moved soil temperature and moisture sensors from the soil into a cold bag of water, and moved humidity and air temperature sensors into an enclosed pitcher of hot water. In all instances, Confidence detects the injected fault within 5 minutes.

Confidence also notified us of a fault on a temperature sensor located next to the site where we were working. The sensor was reporting temperature 10 degrees higher than the surrounding sensors. After checking on the sensor, we immediately discovered that it had become unearthed and was exposed to direct sunlight. This fault was likely an unintended consequence of our work at the neighboring site. Without being at the site, it would be virtually impossible to determine the cause of the problem.

5.7 Evaluation Summary

We show that Confidence detects and diagnoses both real and injected sensor and network faults quickly and with few false positives and negatives, even when over one half of the data are faulty. We use replayed data from two deployments, network simulations, and two real-world deployments to evaluate a three-part performance hypothesis. First, we show that Confidence correctly detects and diagnoses at least 90% of data in this wide range of deployment scenarios. We establish ground truth for the datasets using a combination of sensor calibration, redundant sensor deployment, in-field observation, and soil sample analysis. Second, we show that system accuracy improves when a user incorporates outcome-based feedback. We quantify fault detection performance with and without user feedback using replayed data from our first deployment at JR. Third, we show that with limited feedback from the user, Confidence accurately classifies more data than common thresholding techniques, even when thresholds are manually tuned to maximize detection accuracy. Our sensitivity analysis shows that Confidence performs well under a wide range of system parameter settings.

6 CONCLUSION

We present a novel solution to the problem of deploying systems in uncharacterized environments by incorporating user feedback into system algorithms during the deployment. Confidence's outcome-based feedback is transparent and enables users to directly modify outcomes, thereby avoiding such frustrating events as forcing a user to blindly tune parameters in the field. Confidence limits the outcome-based feedback required of the user through the design of a feature space that: 1) reduces fault detection and diagnosis to outlier-detection and other simple automated mechanisms, and 2) enables the system to re-use user feedback on future data. We have run Confidence on data from several wireless sensing systems: We have run it in the field with a deployment of up to 130 sensors, and tested it on replayed data traces collected from past deployments. Confidence has been tested on over 15 different types of sensors. We demonstrate that Confidence detects and diagnoses most faults, in many different deployment scenarios and with little knowledge about the environment required in advance, quickly and with few false positives and negatives, even when over one half of the data are faulty.

REFERENCES

[1] Anonymized.

[2] P. Barham, R, Isaacs, R. Mortier, and D. Narayanan. Magpie: Online modelling and performance-aware systems. In *Procs. of HotOS*, 2003.

[3] J. Bertrand-Krajewski, J. Bardin, M. Mourad, and Y. Beranger. Accounting for sensor calibration, data validation, measurement and sampling uncertainties in monitoring urban drainage systems. *Water Science and Technology*, pp. 95–102, 2003.

[4] R. Dara, S.C. Kremer, and D.A. Stacey. Clustering unlabeled data with soms improves classification of labeled real-world data. In *Procs. of the Intl. Joint Conf. on Neural Networks*, 2002.

[5] A. Demiriz, K. Bennett, and M. Embrechts. Semi-supervised clustering using genetic algorithms. In *Procs. of Artificial Neural Networks in Engineering*, 1999.

[6] E. Eskin. Anomaly detection over noisy data using learned probability distributions. In *Procs. of the 17th Intl. Conf. on Machine Learning*, 2000.

[7] Lin et. al. Fuzzy reasoning model for semiconductor process fault detection using wafer acceptance test data, Patent Number 7,035,770, issued April 25, 2006.

[8] A. Fox, E. Kiciman, D. Patterson, M. Jordan, and R. Katz. Combining statistical monitoring and predictable recovery for self-management. In *Proc. Workshop on Self-Managed Systems*, October 2004.

[9] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin. EmStar: A software environment for developing and deploying wireless sensor networks. In *Proc. USENIX*, Boston, MA, 2004. USENIX. To appear.

[10] Q.P. He and S.J. Qin. A new fault diagnosis method using fault directions in fisher discriminant analysis. *AIChE*, 2005.

[11] J. W. Hines and B. Rasmussen. On-line sensor calibration verification: A survey. In *14th International Congress and Exhibition on Condition Monitoring and Diagnostic Engineering Management*, September 2001.

[12] E. Kıcıman and A. Fox. Detecting application-level failures in component-based Internet services. In *IEEE Transactions on Neural Networks*, Spring 2005.

[13] L. B. Larkey, L.M.A. Bettencourt, and A.A. Hagberg. In-situ data quality assurance for environmental applications of wireless sensor networks. Technical Report LA-UR-06-1117, Los Alamos Laboratory, 2006.

[14] X. Li and L. E. Parker. Design and performance improvements for fault detection in tightly-coupled multi-robot team tasks. In *IEEE SoutheastCon*, 2008.

[15] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. User-level internet path diagnosis. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, 2003.

[16] D. Manov, G. Chang, and T. Dickey. Methods for reducing biofouling of moored optical sensors. *Journal of Atmospheric and Oceanic Technology*, 2003.

[17] E. Nath and B. Nath. Context-aware sensors. In *Proc. of European Workshop on Sensor Networks*, 2004.

[18] N. Ramanathan, L. Balzano, M. Burt, T. Harmon, C. Harvey, J. Jay, E. Kohler, S. Rothenberg, M. Srivastava, and D. Estrin. Rapid deployment with confidence: Calibration and fault detection in environmental sensor networks. In *CENS Tech Report #62*, 2006.

[19] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy for the sensor network debugger. In *Procs. of SenSys*, 2005.

[20] N. Ramanathan, T. Schoellhammer, D. Estrin, M. Hansen, T. Harmon, E. Kohler, and M. Srivastava. The final frontier: Embedding networked sensors in the soil. Technical Report 68, CENS, November 2006.

[21] S. Rost and H. Balakrishnan. Memento: A Health Monitoring System for Wireless Sensor Networks. In *IEEE SECON*, Reston, VA, September 2006.

[22] A. Sheth, C. Doerr, D. Grunwald, R. Han, and D. Sicker. Mojo: a distributed physical layer anomaly detection system for 802.11 wlans. In *MobiSys 2006: Proceedings of the 4th international conference on Mobile systems, applications and services*, 2006.

[23] C. Stauffer and W.E.E. Grimson. Adaptive background mixture models for real-time tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 2, 1999.

[24] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In *Procs. Sensys*, 2004.

[25] R. Szewczyk, J. Polastre, and A. Mainwaring. Lessons from a sensor network expedition. In *Proc. EWSN*, Berlin, Germany, 2004.

[26] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a sensor network expedition. In *Proc. EWSN*, January 2004.

[27] W. Tang and T.M. Khoshgoftaar. Noise identification with the k-means algorithm. In *16th IEEE International Conference on Tools with Artificial Intelligence*, 2004.

[28] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A macroscope in the redwoods. In *Proc. SenSys*, November 2005.

[29] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. In *Procs. of EWSN*, 2005.

[30] W. Yu. Socio-hydrologic approaches for managing groundwater contamination problems: strategies for the arsenic problem in bangladesh. doctoral thesis, division of engineering and applied sciences. 2003.

[31] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proc. SenSys*. ACM, 2003.