STATISTICAL MODEL-BASED SENSOR DIAGNOSTICS FOR AUTOMATION SYSTEMS

Brian Sallans [*] **Dietmar Bruckner** [**] **Gerhard Russ** [*]

[*] *{brian.sallans,gerhard.russ}@arcs.ac.at*
Information Technologies,
ARC Seibersdorf research GmbH, Vienna, Austria
[**] *bruckner@ict.tuwien.ac.at*
Institute of Computer Technology,
Vienna University of Technology, Vienna, Austria

Abstract: A method for the automatic detection of abnormal sensor values in automation systems is described. The method is based on statistical generative models of sensor behavior. A model of normal sensor behavior is automatically constructed. Model parameters are optimized using an on-line maximum-likelihood algorithm. Incoming sensor values are then compared to the model, and an alarm is generated when the sensor value has a low probability under the model. Model parameters are continuously adapted on-line. The system automatically adapts to changing conditions and sensor drift, as well as detecting isolated abnormal sensor values.

Keywords: automation, statistical inference, sensor failures, diagnostic tests

1. INTRODUCTION

Automation systems have seen widespread deployment in modern buildings, and include systems for environmental control, energy management, safety, security, access control, and remote monitoring. As the cost of automation systems falls, and the technology converges towards standardized protocols, we can expect automation to move from the office into the home. It will also encompass not just building management technology, but also entertainment, kitchen appliances and communications devices.

Todays sensor and control systems are primarily based upon the processing of sensor information using predefined rules. The user or operator defines, for example, the range of valid temperatures for a room by a rule – when the temperature value in that room is out of range (e.g. caused by a defect), the system reacts (for example, with an error message). More complicated diagnostics require an experienced operator who can observe and interpret real-time sensor values. However, as systems become larger, are deployed in a wider variety of environments, and are targeted at technically less-sophisticated users, both possibilities (rule-based systems and expert users) become problematic. The control system would require comprehensive prior knowledge of possible operating conditions, ranges of values and error conditions. This knowledge may not be readily available, and will be difficult for an unsophisticated user to input. It is impractical for experienced operators to directly observe large systems, and naive users can not interpret sensor values.

The goal of this work is to automatically recognize error conditions specific to a given sensor, actuator or system without the need of pre-programmed error conditions, user-entered parameters, or experienced operators. The system observes sensor and actuator data over time, constructs a model of “normality”, and issues error alerts when sensor or actuator values vary from normal. The result is a system that can recognize sensor errors or abnormal sensor or actuator readings, with minimal manual configuration of the system. Further, if sensor readings vary or drift over time, the system can automatically adapt itself to the new "normal" conditions, adjusting its error criteria accordingly.

2. BACKGROUND

The diagnostic system is based on statistical "generative" models (SGMs). Statistical "generative" models attempt to reproduce the statistical distribution of observed data. As an example, one of the simplest and most widely used statistical generative models is the Gaussian distribution. The mean and standard deviation are parameters which can be fit to data. The model can then be used to determine how likely a new data value is (its probability under the model), or to "generate" new data (i.e. draw samples from a Gaussian distribution with the same mean and standard deviation). [1]

The Gaussian distribution is not appropriate for many modeling tasks, because much data does not actually come from a Gaussian-distributed source. Because of this, recent work in generative models has focused on non-Gaussian models (see for example (Hinton *et al.*, 1995; Hinton *et al.*, 1998; Lee *et al.*, 1999)). In addition to the Gaussian distribution, the diagnostic system uses a number of different SGMs to capture the distribution of sensor data, including histograms, mixture models (Bishop, 1995), hidden Markov models (Rabiner and Juang, 1986) and Bayesian networks (Pearl, 1988).

Sensor and control data poses several challenges. The data can be anything from very low level temperature readings to higher level "fault" detectors or occupancy data from building entry systems. This very different data must be fused into a single system. The volume of data requires fast algorithms (Jaakkola, 1997; Frey, 1997; Jordan *et al.*, 1999), and algorithms that can work with on-line data as it arrives (Neal and Hinton, 1998). The data values are time-dependent, so a model must (explicitly or implicitly) take time into account (Kalman, 1960; Rabiner and Juang, 1986; Ghahramani and Jordan, 1997; Ghahramani and Hinton, 1998; Sallans, 2000).

Previous work in applying statistical methods for fault detection includes the use of methods from statistical quality control (SQC) (see Fasolo and Seborg (1994) and Schein and House (2003), for example). These methods compare sensor values to prepared statistical "charts". If the sensor values vary from their expected values over a prolonged period of time, the charts can detect this variation.

[1] For probability density models, such as the Gaussian, which are functions of real-valued variables, the probability of the input value is not directly computed from the model. Rather the probability of either exceeding the given value, or of generating a value within a small neighborhood of the given value is computed. See section 3.3 for more details.

These methods differ from the SGM method in a number of important respects. First, and most importantly, because the charts are designed to detect small variations over a long time period, SQC methods require collection of data and detection of faults over a relatively long time window (for example, several faults detected over the course of 1000 sensor readings) in order to detect faults reliably. The SGM approach is designed to detect individual sensor errors, by comparing each new sensor value to a statistical model. Because of this, abnormal sensor values are detected immediately by the SGM-based system.

Second, because SQC methods are designed for long-term use, they can detect but not adapt to system changes or sensor drift. The SGM method was specifically designed to automatically accomodate system behavior changes and long-term sensor drift.

There are several other approaches to fault detection. In classical model-based detection, detailed domain knowledge is used to build a model of the system. Deviations between model predictions and system behavior are flagged as faults (see Isermann (1984) for a survey). In pattern matching detection, faults are induced in a system, and the resulting sensor values are recorded. A classifier, such as a neural network, is trained using this data set of normal and abonormal behavior to detect failures (see House *et al.* (1999) for example). These methods require either a working system for experimentation, or an in-depth knowledge of the system in question, both of which are lacking for large building automation systems.

Despite their success in other domains, SGMs have not been applied to error detection for building automation sensor and control data. There are two reasons for this. First, it has only recently become possible (and economical) to collect a wide range of cross-vendor sensor data at a central location. Second, most algorithms to optimize the parameters of SGMs are quite compute-intensive. Many algorithms have been of only theoretical interest, or are restricted to small toy problems. Only recently have powerful approximation algorithms and powerful computers become available that can handle large quantities of data in real-time. The combination of fast, powerful optimization algorithms, fast computers, and available multi-sensor data makes the time ripe for probabilistic modeling of sensor and control data.

3. THE DIAGNOSTIC SYSTEM

The goal of the diagnostic system is to automatically detect sensor errors in a running automation system. It does this by learning about the behavior of the automation system by observing data flowing through the system. The diagnostic system builds a model of the sensor data in the underlying automation system, based on the data flow. From the optimized model, the diagnostic system can identify abnormal sensor and actuator values. The diagnostic system can either analyze historical data, or directly access live data (see Figure 1).

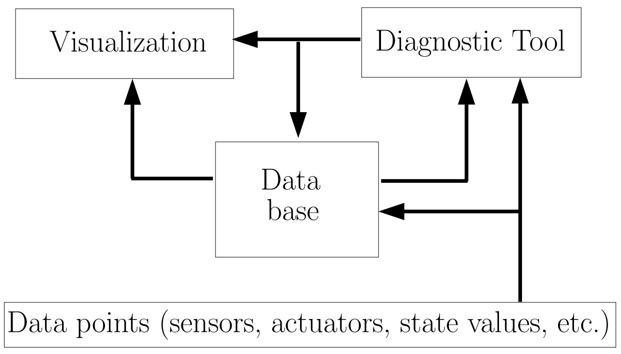

Fig. 1. The Automation System. Sensor and actuator values are stored in a database, and simultaneously analyzed on-line by the error detection system. Connections between the diagnostics system and other system components are either direct, or via the Database.

We use a set of statistical generative models to represent knowledge about the automation system. A statistical generative model takes as input a sensor value, status indicator, time of day, etc., and returns a probability between zero and one. For example, the most widely used example of a SGM is the Gaussian distribution. For each input value, a Gaussian computes a weighted distance from the Gaussian mean, and returns a decreasing probability as the value moves away from the mean.

Using SGMs has several advantages. First, because the model encodes the probability of a sensor value occurring, it provides a quantitative measure of "normality", which can be monitored to detect abnormal events. Second, the model can be queried as to what the "normal" state of the system would be, given an arbitrary subset of sensor readings. In other words, the model can "fill in" or predict sensor values, which can help to identify the source of abnormal system behavior. Third, the model can be continuously updated to adapt to sensor drift.

3.1 *The Statistical Models*

The system implements a number of SGMs (see Table 1).

Table 1. Statistical Generative Models

Model	Variable Type	Parameters
Gaussian	Real	μ,σ^2
Histogram	Discrete, Real	Bin counts
Mixture of Gaussians	Real	μ_i,σ_i^2,π_i
Hidden Markov model	Real	T_{ij},μ_i,σ_i^2
Hidden Markov model	Discrete	T_{ij},Bin counts

The more complex models add additional capabilities, or relax assumptions in comparison to the Gaussian model:

Histogram: This is a very general model that is appropriate for discrete sensor values, as well as real-valued sensors with an arbitrary number of modes. One drawback is that a histogram requires a rather large quantity of data before it becomes usable or accurate.

Mixture of Gaussians: This model relaxes the Gaussian assumption that the distribution has only one mode. A mixture of Gaussians is composed of a number of Gaussian models, and each data value is attributed to the Gaussian modes with a weighting given by a "posterior probability". See for example (Bishop, 1995).

Hidden Markov Model: This model is the equivalent of the Mixture of Gaussians model or the histogram model, but with the addition that the current sensor value can be dependent on previous values.

3.2 *Error Detection*

Given an SGM, implementation of this functionality is straight-forward. The system assigns to each newly-observed data value a probability. When this probability is high, the system returns that the new data value is a "normal" value. When the probability falls below a specific threshold, the system rates the value as "abnormal". The SGM system generates alarm events when it observes abnormal sensor values. This leaves open the question of how to assign the threshold for normality. In practice, the user sets the threshold using a graphical interface. Initially, before the system has learned normal system behavior, many alarms are generated, and the user may decide to set the threshold to a value near zero. As the system acquires a better model of the sensor system, the threshold can be raised. In any case, the threshold parameter tells us how improbable an event should be to raise an alarm. The system can also use a log-probability scale, so that the threshold can easily be set to only register extremely unlikely events.

3.3 *Statistical Generative Models*

The diagnostic system uses SGMs of automation data points. For any given data value x, model M assigns a probability to x: $P_M(x) \rightarrow [0, 1]$.

Note that, for discrete distributions such as a histogram, the value assigned to x by the model $P_M(x)$ is a well-defined probability, since the set of possible assignments to x is finite. For a probability density, such as a Gaussian or mixture of Gaussians, the probability value assigned to x by the model is the probability *density* at that value. In order to convert this density to a probability, the probability of generating a value within a neighborhood $\pm\delta$ around x is computed as $\int_{-\delta}^{\delta} P_M(x+\phi)d\phi$, and approximated as $2\delta P_M(x)$ for small δ. Alternatively the probability under the model of equaling or exceeding the observed value can be computed: $P_M(x' \geq x) = \int_{\phi}^{\phi+\infty} P_M(x+\phi)d\phi$.

The data x can be a sensor reading such as a motion sensor, door contact sensor, temperature sensor, and so on. Given a new data value, the system can assign a probability to this value. When the probability is above a given threshold, the system concludes that this data value is "normal".

Given a sequence of sensor readings $\boldsymbol{x} = \{x_1, ..., x_T\}$ from times 1 to T, the system must create a model of "normal" sensor readings. The system uses maximum-likelihood parameter estimation. We describe this method below.

Assume that model M has parameters θ. For example, the standard Gaussian distribution has two parameters, a mean μ and a variance σ^2. In this case, $\theta = \{\mu, \sigma^2\}$. The log-likelihood of the model parameters given the data $\boldsymbol{x}$ is given by:

$$\mathcal{L}(\theta) = \log P_M(\boldsymbol{x}|\theta) \qquad (1)$$

where the notation $P(\boldsymbol{x}|\theta)$ denotes a conditional probability: The probability assigned to $\boldsymbol{x}$ depends on the current values of the parameters θ. The log-likelihood tells us how "good" the parameters of the model are. When it is high, the data values that we usually see have a high probability under the model. That is, the model captures what are "normal" data values.

The maximum-likelihood parameters are defined as the parameter values which maximize the log-likelihood over the observed data:

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}} \{\log P_M(\boldsymbol{x}|\theta)\}$$

Again, take the Gaussian distribution as an example. The log-likelihood of the 1D Gaussian is given by:

$$\mathcal{L}_{\text{Gaussian}}(\mu, \sigma^2) = -\sum_i \left[\frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}(x_i - \mu)^2\right]$$

where $x_i \in \boldsymbol{x}$ are independent, identically-distributed data values. Taking the derivative of the log-likelihood with respect to the parameters yields:

$$\frac{\partial \mathcal{L}_{\text{Gaussian}}(\mu, \sigma^2)}{\partial \mu} = \sum_i \frac{(x_i - \mu)}{\sigma^2} \qquad (2)$$

$$\frac{\partial \mathcal{L}_{\text{Gaussian}}(\mu, \sigma^2)}{\partial \sigma^2} = \sum_i \left(-\frac{1}{\sigma} + \frac{(x_i - \mu)^2}{\sigma^3}\right) \qquad (3)$$

We can now solve for the maximum-likelihood parameters. Setting the derivatives in Eq.(2) and Eq.(3) to zero and solving the system of two equations yields:

$$\mu_{ML} = \frac{1}{T}\sum_i x_i \qquad (4)$$

$$\sigma^2_{ML} = \frac{1}{T}\sum_i (x_i - \mu_{ML})^2$$

which are the familiar textbook definitions of mean and variance of a Gaussian.

3.4 *On-line Parameter Updates*

In order for the system to continually adapt the model parameters, we need a parameter update algorithm that can incrementally change the parameters based on newly observed sensor values. Such methods are called "on-line" parameter update rules. On-line updates have the advantage that there is no time during which the system is in an "optimization" phase, and unavailable for diagnostics.

One simple on-line method for computing locally-optimal maximum-likelihood parameters is "stochastic gradient ascent". As each new data value x_i is observed, the parameters are adjusted so as to increase the log-likelihood:

$$\theta_{ML}^{\text{new}} = \theta_{ML}^{\text{old}} + \epsilon \frac{\partial \log P_M(x_i|\theta)}{\partial \theta}$$

where ϵ is a "learning rate". Given appropriate conditions on ϵ and the log-likelihood function, this algorithm will find a local maximum of the log-likelihood. In practice, the success and speed of the algorithm depends on the form of the log-likelihood and the order and frequency with which parameters are updated. Good results can be achieved for many generative models, although convergence can be slow (see for example (Dempster *et al.*, 1977; Russell *et al.*, 1995; Olshausen and Field, 1996). Our diagnostic system makes extensive use of this parameter update rule.

For the example of a Gaussian model, the on-line parameter update rules are given by:

$$\mu_{ML}^{\text{new}} = \mu_{ML}^{\text{old}} + \epsilon_\mu \frac{\partial \mathcal{L}_{\text{Gaussian}}(\mu, \sigma^2)}{\partial \mu}$$

$$\sigma^{2}{}_{ML}^{\text{new}} = \sigma^{2}{}_{ML}^{\text{old}} + \epsilon_{\sigma^2} \frac{\partial \mathcal{L}_{\text{Gaussian}}(\mu, \sigma^2)}{\partial \sigma^2}$$

In some cases, as with the Gaussian, where parameters can be explicitly solved for, another on-line parameter update method can be used. The alternative to a gradient update is the on-line moving average, or "decaying average". Consider for example the expression for the mean of a Gaussian (equation 4):

$$\mu_{ML} = \frac{1}{T}\sum_i x_i$$

If the system receives new data values on-line, we can not use the expression as given, because we do not want to store the entire history of values $\{x_i\}, i = 1, ..., T$. However, we can use a sliding window such that:

$$\mu_{SW}^{\text{new}} = \frac{1}{N} \sum_{i=t_1}^{t_2} x_i$$

where we only store the last $N = (t_2 - t_1)$ values. As a final alternative, we can recompute the mean at each step without storing previous values:

$$\mu_{MA}^{\text{new}} = \frac{1}{N} x_i + \frac{(N-1)}{N} \mu_{MA}^{\text{old}}$$

which is a "decaying average" update, and approximates a sliding window of length N. We call this update rule an "on-line moving average" update rule. This rule can be used for some models but is not as generally applicable as the gradient update rule.

Figure 2 shows an example of parameter optimization for the Gaussian model. The initial mean and variance were set to $\{-5, 9\}$ (the "X"). The contour lines show lines of equal log-likelihood for the Gaussian model. The "O" shows the true parameters for the model $\{5, 4\}$. The curved (upper) line shows the path in parameter space followed by the gradient-ascent rule, from the original guess $\{-5, 9\}$ to the true parameters $\{5, 4\}$. The parameters are updated as each new data value (sampled from a Gaussian with the true parameters $\{5, 4\}$) arrives. Because the gradient is computed using a single value, the "up-hill" direction is noisy, resulting in the slightly "wandering" line we see. Also, when the guess reaches the true parameter values, it wanders around in this area, because for each new data value, the "optimal" parameters are different.

Similarly, the straight (lower) line shows the path in parameter space followed by the on-line moving average rule. As we can see, the two rules follow completely different paths in parameter space, but arrive at the same answer. Which method works best is an empirical question, and depends on the model used, the learning rate, how much data is available, and so on. We tend to use moving-average rules where possible, because of their fast initial convergence, and gradient-based rules otherwise.

4. RESULTS

The diagnostics system has been implemented in an office environment with 20 rooms, an entrance area, and two long corridors. In this environment, motion, door state, luminosity, temperature and humidity sensors have been installed. All sensors are wireless, and report sensor values to a central computer with an attached receiver station. The motion and door state sensors are standard consumer "X10"-brand sensors. The temperature, luminosity and humidity sensors are custom-built wireless sensors (Mahlknecht, 2004).

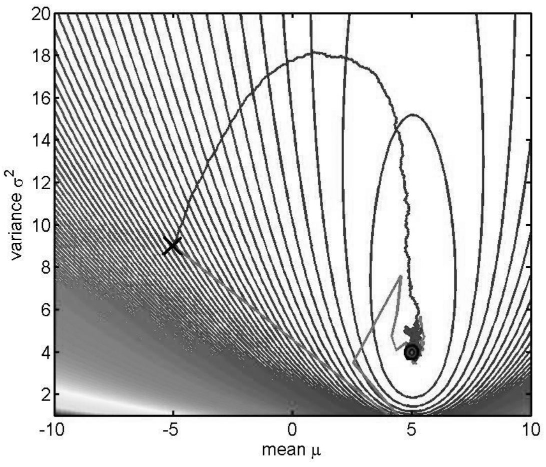

Fig. 2. Example of gradient-based optimization and on-line averaging for the Gaussian model.

The sensors are battery-powered, and the custom sensors have built-in voltmeters which report the current battery state. All data is stored in an SQL database. At the time of writing, data has been collected over the course of more than three weeks. The diagnosis system can initialize its models by analyzing stored historical data, or simply optimize on-line as it reports error conditions.

4.1 *Parameter Optimization*

If the system begins reporting errors without first analyzing historical data, it will initially report many errors. This will continue until the system has acquired enough knowledge of the automation network and sensors to at least have rudimentary models of normal sensor values. As analysis continues, the number of alarms generated falls (see Figure 3).

The model log-likelihood, given by Eq.(1), is a measure of model quality. As the parameter values are optimized on-line, the log-likelihood of the sensor models increases. Figure 4 shows the average log-likelihood during parameter optimization, on average, for the sensor models in the test system. The log-likelihood increases with time, indicating that the models improve over time. The log-likelihood does not consistently increase, however, due to the on-line fitting of parameters simultaneously with reporting of abnormal sensor values. If sensors receive abnormal values, the log-likelihood decreases, until the values return to their normal range or the sensor model adapts to the new range of values.

4.2 *Optimized Sensor Models*

After analyzing a quantity of data the system has models of typical sensor values. In the current implementation, the system uses one model per sensor, for every

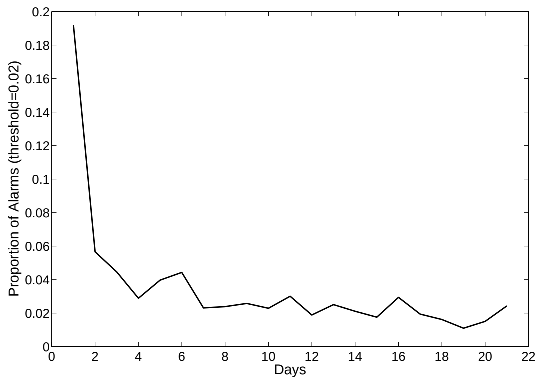

Fig. 3. Proportion of alarms generated (at a threshold of $\log(P_M(x)) = -4$) as a function of training time. The proportion of alarms is the ratio of the number of alarms generated to the number of sensor values received by the diagnostic system. With time, the alarm rate drops to approximately match the chosen threshold, as the statistical models used by the system match themselves to the actual behavior of the system.

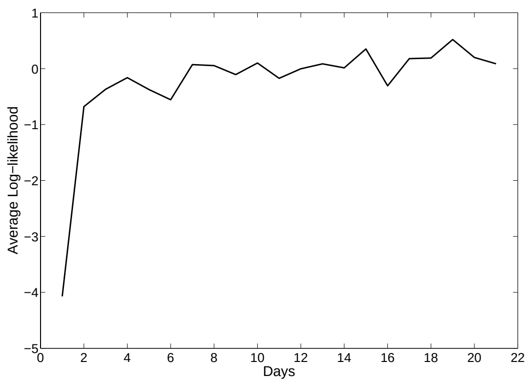

Fig. 4. Average learning curve for sensor models. On average the log-likelihood for the model improves over time. There is a rapid improvement in the first days, followed by a slower improvement over weeks.

half-hour interval. That is, there is a distinct model for each half-hour interval.

The system also has models for "data period" (the typical number of seconds between data packets from a sensor). For sensors such as motion and door state, it is informative to know how often the sensor state changes, in order to detect sensor failures or abnormal conditions. Models are fit to the typical time interval between arrival of sensor data packets. Figure 5 shows models for three sensors: Motion frequency, temperature and luminosity. The first two are examples of "mixture of Gaussians" models, while the third is a simple Gaussian. Note that the temperatures, for example, would not be well-characterized by a simple Gaussian model.

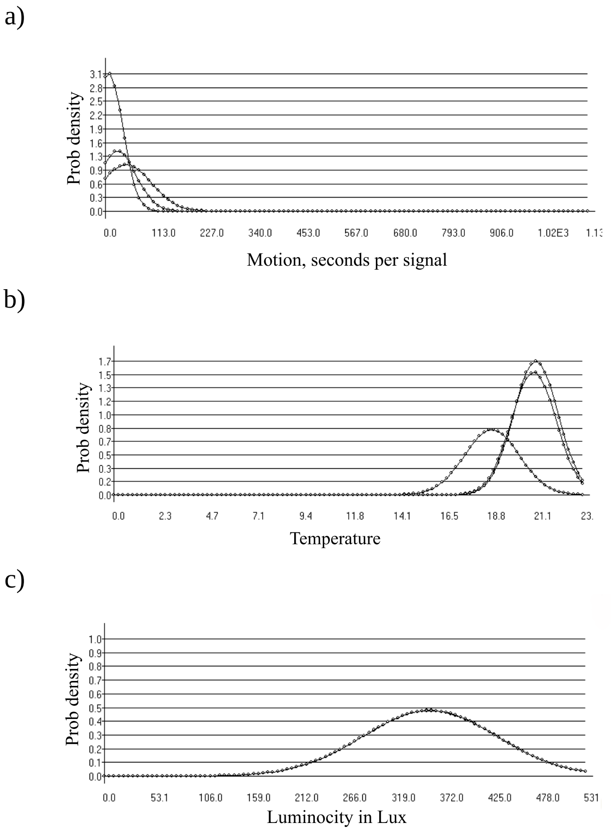

Fig. 5. Examples of models optimized by the system. In some cases, the Gaussian model is sufficient, and in other cases the model has (a) a non-Gaussian shape or (b) multiple modes.
a) A motion sensor from 5:00pm-5:30pm;
b) A temperature sensor from 8:00am-8:30am;
c) A luminosity sensor from 6:00pm-6:30pm.

4.3 *Error Detection and Drift Adaptation*

We can see the trade-off between adaption to sensor drift and detection of short-term behavior changes by examining an individual model more closely. The wireless sensors are powered by small batteries, which leaves open the possibility that an unusual power drain can deplete the batteries and cause a sensor failure. The sensors are equipped with voltage sensors, indicating the voltage currently available from the battery. One of the sensors in the test system did in fact have an unusual power drain during the test period. Figure 6 and 7 show the diagnostic system behavior for two different speeds of parameter updates. In Figure 6, the model parameters are optimized during the initial days, and then held fixed. In this case, the drift is detected relatively quickly.

In Figure 7, the model parameters are continuously optimized on-line. In this case, the drift is detected by the model, but does not result in an alarm until the drift rate increases beyond the adaptation rate of the model.

In the case of battery voltage level, the initial behavior may be more desirable. For other sensors (luminosity,

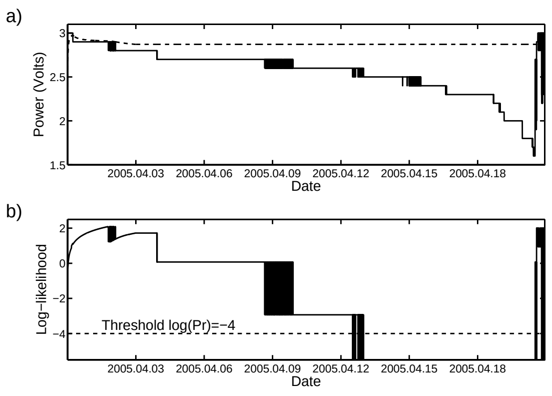

Fig. 6. a) Voltage as a function of time for a wireless sensor. The sensor was faulty, resulting in battery drain and a decrease in system voltage over time. The available voltage increased at the end, when the batteries were changed. The solid line is the actual voltage, and the dashed line is the expected voltage under the voltage model.
b) Log-likelihood curve for the voltage model. Model parameters are quickly optimized for the observed voltage level. Because the model parameters are not adapted after the initial period, the model detects the slow voltage drift relatively quickly.

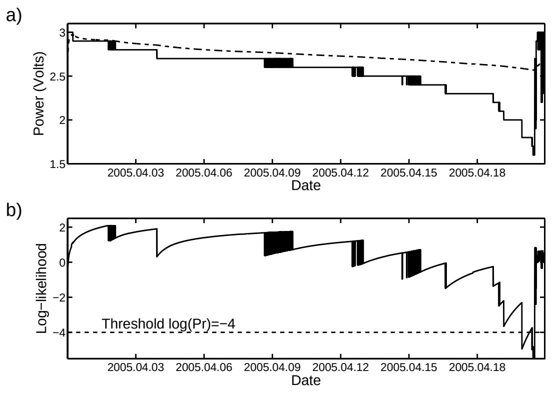

Fig. 7. a) Voltage as a function of time for a wireless sensor. The sensor was faulty, resulting in battery drain and a decrease in system voltage over time. The available voltage increased at the end, when the batteries were changed. The solid line is the actual voltage, and the dashed line is the expected voltage under the voltage model.
b) Log-likelihood curve for the voltage model. Model parameters are quickly optimized for the observed voltage level. As the batteries are drained, there is a slow drift downward. The model continues to adapt, resulting in no alarm. As the rate of drift accelerates, the change is too rapid for the model to adapt, and an alarm is triggered.

temperature) it may be preferable for the system to adapt to drifting sensor values without generating an alarm. Behavior such as shown in Figure 7 may be the most preferable, where some drift can be accommodated, but rapid changes will be detected.

5. CONCLUSION

This paper describes a method for the automatic detection of abnormal sensor values, based on statistical generative models of sensor behavior. The system can automatically build a model of normal sensor behavior. It does this by optimizing model parameters using an on-line maximum-likelihood algorithm. Incoming sensor values are then compared to the model, and an alarm is generated when the sensor value has a low probability under the model. The model parameters are continuously adapted on-line. The result is a system that automatically adapts to changing conditions and sensor drift. As well as detecting isolated abnormal sensor values, the system can either detect or adapt to slower sensor drift, depending on the rate of model parameter adaptation.

With the current system, some knowledge of the expected rate of sensor drift is required, in order to correctly set the parameter adaptation rate. An alternative would be to use two error-detection mechanisms. For isolated abnormal values, the current system would be used. For the detection of long-term drift, changes in model parameter values themselves would be monitored over a longer time-scale. The result would be a hierarchical model, with sensor values at the lower level, and model parameters at the higher level.

The current system only detects abnormal values for single sensors. The system is currently being extended to include multivariate models which capture correlations between sensor values. In this way, abnormal groups of sensor values can be automatically detected.

6. REFERENCES

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press Inc.. New York NY.

Dempster, A. P., N. M. Laird and D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society Series B* **39**, 1–38.

Fasolo, Paul and Dale E. Seborg (1994). An SQC approach to monitoring and fault detection in HVAC control systems. In: *Proceedings of the American Control Conference*. Baltimore, Maryland.

Frey, Brendan (1997). Continuous sigmoidal belief networks trained using slice sampling. In: *Advances in Neural Information Processing Systems* (Michael C. Mozer, Michael I. Jordan and Thomas Petsche, Eds.). Vol. 9. The MIT Press, Cambridge. pp. 452–458.

Ghahramani, Z. and G. E. Hinton (1998). Variational learning for switching state-space models. *Neural Computation* **4**(12), 963–996.

Ghahramani, Z. and M. I. Jordan (1997). Factorial hidden Markov models. *Machine Learning* **29**, 245–273.

Hinton, G. E., B. Sallans and Z. Ghahramani (1998). A hierarchical community of experts. In: *Learning in Graphical Models* (M. I. Jordan, Ed.). pp. 479–494. Kluwer Academic Publishers.

Hinton, G. E., P. Dayan, B. J. Frey and R. M. Neal (1995). The wake-sleep algorithm for unsupervised neural networks. *Science* **268**, 1158–1161.

House, J.M., W. Y. Lee and D. R. Shin (1999). Classification techniques for fault detection and diagnosis of an air-handling unit. *ASHRAE Transactions* **105**(1), 1987–1997.

Isermann, R. (1984). Process fault detection based on modeling and estimation methods – a survey. *Automatica* **20**(4), 387–404.

Jaakkola, T. S. (1997). *Variational Methods for Inference and Estimation in Graphical Models*. Department of Brain and Cognitive Sciences, MIT. Cambridge, MA. Ph.D. thesis.

Jordan, M. I., Z. Ghahramani, T. S. Jaakkola and L. K. Saul (1999). An introduction to variational methods for graphical models. *Machine Learning* **37**, 183:233.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Trans. ASME, Series D, Journal of Basis Engineering* **82**, 35–45.

Lee, T-W, M. Girolami and T. Sejnowski (1999). Independent component analysis using an extended infomax algorithm for mixed sub-gaussian and super-gaussian sources.

Mahlknecht, S. (2004). *Energy-Self-Sufficient Wireless Sensor Networks for the Home and Building Environment*. Institute of Computer Technology, Technical University of Vienna. Vienna, Austria. Dissertation thesis.

Neal, R. M. and G. E. Hinton (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In: *Learning in Graphical Models* (M. I. Jordan, Ed.). pp. 355–368. Kluwer Academic Publishers.

Olshausen, B. A. and D. J. Field (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* **381**, 607–609.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann. San Mateo, CA.

Rabiner, Lawrence R. and Biing-Hwang Juang (1986). An introduction to hidden Markov models. *IEEE ASSAP Magazine* **3**, 4–16.

Russell, Stuart, John Binder, Daphne Koller and Keiji Kanazawa (1995). Local learning in probabilistic networks with hidden variables. In: *Proc. Fourteenth International Joint Conference on Artificial Intelligence*. Montreal, Canada.

Sallans, B. (2000). Learning factored representations for partially observable Markov decision processes. In: *Advances in Neural Information Processing Systems* (S. A. Solla, T. K. Leen and K-R Müller, Eds.). Vol. 12. The MIT Press, Cambridge. pp. 1050–1056.

Schein, J. and J.M. House (2003). Application of control charts for detecting faults in variable-air-volume boxes. In: *ASHRAE Transactions*. Vol. 109. pp. 671–682.