2014 IEEE International Conference on Data Mining Workshop

Online Fusion of Incremental Learning for Wireless Sensor Networks

H.H.W.J. Bosman[*][†], G. Iacca[†],
[*]*Department of Electrical Engineering*
Eindhoven University of Technology
P.O. Box 513, 5600MB, Eindhoven, The Netherlands
h.h.w.j.bosman, a.liotta @ tue.nl

H.J. Wörtche[†], A. Liotta[*]
[†]*INCAS*^{3}
P.O. Box 797, 9400AT, Assen, The Netherlands
heddebosman, giovanniiacca, heinrichwoertche @ incas3.eu

***Abstract*—Ever-more ubiquitous embedded systems provide us with large amounts of data. Performing analysis close to the data source allows for data reduction while giving information when unexpected behavior (i.e. *anomalies* in the system under observation) occurs. This work presents a novel approach to online anomaly detection, based on an ensemble of classifiers that can be executed on distributed embedded systems.**

We consider both single and multi-dimensional input classifiers that are based on prediction errors. Predictions of single-dimensional time series input come from either a linear function model or general statistics over a data window. Multi-dimensional input stems from current and historical sensor values as well as predictions. We combine the classifier outputs in the ensemble using a heuristic method and Fisher's combined probability test.

The proposed framework is tested thoroughly using synthetic and real-world data. The results are compared to known methods for anomaly detection on limited-resource systems. While individual classifiers perform comparably to known methods, our results show that using an ensemble of classifiers increases the overall detection of anomalies considerably.

***Keywords*-Anomaly detection; Embedded Systems; Online Learning**

I. Introduction

In recent years, driven by the vision of the Internet of Things, an increasing number of devices have been made smart by the use of embedded electronic systems. Such systems are becoming ubiquitous and pervasive, that is, they can be found as everyday objects, such as smart phones and fridges. Advances in microelectronics, sensor systems and software platforms are continuously improving these devices, making them capable of communicating high-level information and performing domain-specific local processing [1]. However, while their sensory and communication apparatus usually offer a rich set of functionalities, their hardware resources, such as CPU, memory, and batteries are still quite limited.

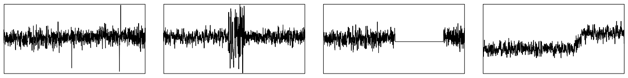
An example of such embedded systems is represented by Wireless Sensor Networks (WSN). A WSN is a network of embedded devices ("nodes") that are tightly coupled to the environment, with limited resources. Numerous research in this field concentrates on reducing the usage of these resources through, for example, efficient communication [2]. However, the WSN community is nowadays shifting its attention from the systems themselves to the applications thereof, for instance, in monitoring for agriculture [3] or in climate research [4], and also in industrial settings [5]. Typically the deployed WSN in such applications are long-lived and provide large datasets. However, the amount of data generated overwhelms manual analysis [6].

The field of anomaly detection provides automated methods of analysis that detect and flag data that contain faults and unexpected events, behaviors or patterns. Such analysis is commonly performed on central storage and processing facilities, by creating models of the environment through either mathematical or statistical means. On the other hand, there is a growing interest in distributing the data processing and the anomaly detection to the nodes: doing so, there can be obvious advantages in terms of data reduction (because only relevant information, corresponding to anomalous behavior, are sent to the users) and resource efficiency (because of a reduced usage of communication channel, and thus a saving of energy).

In this paper we propose a combination of adaptive online estimation and an ensemble of classification methods for anomaly detection on embedded systems. Firstly, we use a sliding-window function-approximation technique called SwiftSeg [7] to predict future measurements of a single sensor. Then, we predict measurements, based on the fusion of measurements and predictions of different sensors, using Recursive Least Squares (RLS) [8]. We show that this combination of methods can be executed within a single embedded device, making use of sensory information obtained either locally or, when communication is available (such as within a WSN), distributedly from neighboring nodes. The prediction results are then analyzed to detect anomalies.

We focus on anomalies in sensor data, that result from the environment or from the sensor system itself. Since our goal is to devise a reusable method applicable to different domains, we target generic anomaly types which can be found in literature, e.g., [9], [10], [11]. These types are: *spike*, *constant*, *noise* and *drift*, and can be caused by environmental factors as well as system faults. *Spikes*, for instance, can occur as a consequence of a bit error, whereas *constant* errors can be caused by a loose wire. *Noise* can occur when batteries drain or by an external disturbance.

978-1-4799-4274-9/14 $31.00 © 2014 IEEE
DOI 10.1109/ICDMW.2014.79



(a) Spike (b) Noise (c) Constant (d) Drift

Figure 1. Injected anomalies in noisy linear data

Finally, *drift* can be caused by a degrading sensor. Examples of these phenomena can be seen in Fig. 1. We evaluate the performance of the proposed detection method on synthetic and real-world WSN datasets that contain instances of these four anomaly types.

The paper is structured as follows: the next section reviews related literature, after which the combination of methods is explained in section III. Section IV continues with the experimental evaluation, followed by the conclusions in section V.

II. Related work

Today, anomaly detection and related techniques, such as fault detection, are commonly used, for instance to detect anomalous behavior or behavioral patterns in security card access [12]. Anomaly detection is also common in data-mining systems, where abundant computing power is used to extract data of interest from large databases of historical data [13], [14]. Another example is given by data center management, where anomaly detection can be adopted to detect problems in system performance and, by diagnosing, to provide potential remedies [15]. In all these cases, the data processing typically takes place in a centralized way, over dedicated machines located in a central facility.

In a context of distributed sensor systems, such as WSN, where the sources of data are physically distributed, this data can be processed either locally (distributedly) or on a centralized system. Interestingly however, recent surveys [6], [16], [17] report that most of the anomaly detection techniques for WSN are centralized. If the anomaly detection is local, the models used are often created or learned a priori, for example by training Echo State Networks offline and then using them online to detect time-series anomalies [18].

A common way of performing anomaly detection is to predict future measurements based on a model, for instance obtained by regression. The prediction error is then analyzed to detect anomalous measurements, assuming the predictions are based on normal behavior. Serdio et al. [19] transform sensor data using Principal Component Analysis or Partial Least Squares. The transformed data of current and historical measurements are then the independent variables for which regression finds a linear model to correlate with the target sensor data. The resulting linear model can then predict sensor measurements, and these predictions are analyzed with an adaptive threshold for faults. Sharma et al. [10] compare linear regression techniques with heuristics and Hidden Markov Models. They conclude that these have good performance, except for detecting low-intensity short faults. Furthermore, when the whole dataset is not available, or the input is a stream of data, one can use a technique such as Recursive Least Squares (RLS) to learn a linear model. While this technique is often used in, for instance, adaptive filtering and control or system identification [20], it can also be used to regress current and historical sensor measurements to another sensor [8]. The simplicity of RLS allows for online learning, measurement prediction and detection of anomalies. A similar technique can also be used to determine spatial correlations between distributed sensors of the same modality [21].

In a single-dimensional time-series a technique called Piecewise Linear Regression can be used [22], which fits lines to segment of the time-series data. Predictions can then be made by linear extrapolation. Furthermore, the slope of the current segment can be compared to historical segments in order to detect anomalies [11]. In relation to such a linear model, a polynomial model can be fit to (a segment of) time-series data. A fast online approach of polynomial fitting is demonstrated by Fuchs et al. [7], who present SwiftSeg, a method for segmenting time-series data, by using either a sliding window, or a growing window polynomial function fitting method. These piecewise function regression methods allow, for instance, data segmentation or prediction.

It is well established that by using an ensemble of learners the classification and regression performance can be improved [23]. Employing a diverse set of learning methods, or experts, can result in a more reliable decision. Curiac et al. [24], for example, use an ensemble of five different classifiers, based on predictions from averages of similar sensors. These predictors are based on averages, RLS, neural networks trained by back propagation, neural network trained using an auto regressive method and on an adaptive neuro-fuzzy inference system. However, this ensemble is trained and ran at a centralized point, and is used to detect only spike anomalies.

We propose a combination of methods where each classifier is able to classify measurements from one or more sensors into either the normal or the anomalous class. Such a framework is similar to the ensemble of classifiers, as described in [24] and seen in Fig. 2, but with the fundamental difference that our approach uses algorithms that are able to run on a limited-resource system, thus allowing for an online, embedded implementation, and a reduced usage of data communication.

III. Experimental setup

Our proposed framework includes the following components, structured as seen in Figures 3 and 4.

- A prediction model, which uses the inputs to predict the estimate $\hat{x}_0$. This model is learned online, within the embedded systems, using light-weight unsupervised machine learning techniques (described below), such that the normal behavior of the input values is learned.

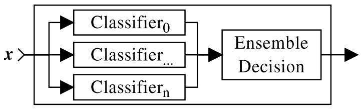


Figure 2. Structure of an ensemble of classifiers. The final decision on $class$ is based upon the outputs of the different classifiers.

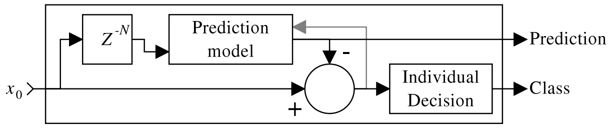


Figure 3. Structure of a single-dimensional time-series classifier. The difference between a prediction, based on the signal delayed by N samples, and the current measurement in time-series x_0 is classified.

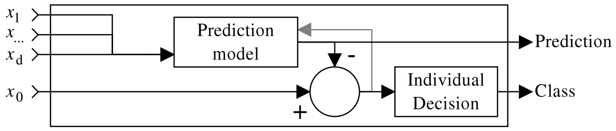


Figure 4. Structure of a multi-dimensional time-series classifier. The difference between a prediction, based on inputs $x_1, \ldots, x_d$, and the current measurement in time-series x_0 is classified.

- A prediction error, or residual, computing component, that compares the predicted value $\hat{x}_0$ with the actual value of x_0. That is, it compares the learned behavior with the actual behavior.
- A decision making component that detects anomalies by examining the posterior distribution of the prediction errors. This distribution is dynamically estimated using a running average method, outlined below.

Each classifier accepts a vector of input values, $\mathbf{x} \in \mathbb{R}^d$. These inputs can be the sensor measurements, historical data or the output of another classifier or predictor.

Our approach uses two prediction model learning techniques, namely the aforementioned SwiftSeg [7], which is used to model a part of time-series data as a polynomial function, and Recursive Least Squares (RLS) [8], which is instead used to estimate the parameters of a linear model. The latter models the relation between the output and a linear combination of inputs, such as sensor readings and historical data. The simplicity of learning (see section IV-A for memory and time complexity) in each of these techniques allows them to function on limited-resource embedded devices.

A. Prediction methods

1) Single-dimensional input predictors: The simplest predictors accept only single-dimensional input values from a time-series. As seen in Fig. 3, the delayed signal is used as the input to the predictor. Our approach uses two such predictors.

The first and simplest method of prediction is using statistics. It keeps a short window, of length L_h, of previous measurements in memory. From this window, it can determine a mean and standard deviation, where the mean is a predictor for the next value in the time-series, and the standard deviation can be used to make a decision.

The second method is SwiftSeg, that models a part of the single-dimensional time-series as a function. It keeps a trace of historical input values in a window and fits a polynomial function F to this window, by means of bases of orthogonal polynomials [7]. The maximum degree of this polynomial can be set depending on, for instance, type of data. For simplicity, and to be generic, however, we choose a polynomial degree of 1, i.e. a line. The resulting polynomial function is then used to predict the next value, i.e. $F_{win}(t) = \hat{x}_0[t]$, where $win = x_0[t-1], x_0[t-2], \ldots, x_0[t-L_s]$, a vector of historical input data of length L_s. Moreover, we can predict k-steps ahead using this function, and in our approach we evaluate the use of $k \in \{1, 2, 3\}$.

2) Multi-dimensional input predictor: Our multi-dimensional input predictor combines several types of input to predict $\hat{x}_0$. Its inputs are not only the raw sensor measurements, but also the above outlined single-dimensional predictions, using SwiftSeg and the windowed statistics. Using Recursive Least Squares, a linear weighted combination of these inputs is used to estimate $\hat{x}_0$. Because of the variety of inputs, the multi-dimensional predictor can accurately predict normal sensor values under conditions that are regarded normal.

For example, given a sensor system with three sensors, our approach is to first apply the single-dimensional predictors on each of the three sensors. That results in predictions for the measurements of these three sensors based on the single-dimensional sensor models. Then, for one sensor, the multi-dimensional predictor gets as inputs the measurements of the other two sensors and the intermediate predictions for all the sensors, both from SwiftSeg and the windowed statistics.

3) Baseline predictors: For a baseline, we compare our approach to two earlier proposed predictors, namely Recursive Least Squares (RLS) [8] and Extreme Learning Machine (ELM) [25]. The RLS method learns a linear model, but ELM learns a non-linear model in the form of a Single Layer Feed-forward Neural Network (SLFN). Both methods model a correlation between sensor values to predict one of the sensor values, and thus get only those sensor values as multi-dimensional input.

B. Decision making

1) Individual predictor decision: To detect anomalies from the predictions, the individual decision making component has to decide if the prediction error $(x_0 - \hat{x}_0)$ is normal or anomalous. We assume the residual of the prediction has a Gaussian distribution. If the predictions were almost perfect, the mean would be zero, and the standard deviation would be very small. In the real world, however, the predictions are less perfect, and might be biased, or not even Gaussian.

In the latter case we assume that, under normal conditions, the predictions are still close enough to the target value. Therefore, our decision making component keeps track of a running average and a running absolute deviation. We determine these estimates by using an Exponentially Weighted Moving Average (EWMA) to obtain the running average and, in a similar manner, the running absolute deviation. We use a method outlined in [8] that uses a fast and slow moving absolute deviation, to account for slower changes in the signal, such as drift.

Based on the running mean and average, the decision making component determines the p-value, the probability of obtaining a test statistic as extreme as this one, of the current sample. The end-user can then set a confidence level, such as 95%, so that the sample is regarded anomalous when it falls outside of the set confidence interval.

A special rule-based decision is made over the windowed statistics predictor. Namely, a series of consecutive measurements can never have a standard deviation smaller than ϵ (where $\epsilon \to 0$), because we expect the system under measurement to slowly change, and the sensors to have some noise. Thus, a standard deviation smaller than ϵ would indicate a *constant* anomaly.

2) Ensemble decision: The individual classifiers decide if an input is normal or anomalous. If the input value is anomalous, not only that class is given, but also the p-value of the input. We then combine these decisions to get a more reliable decision. We evaluate two ensemble methods, being a heuristic rule and Fisher's method [26]. The rule states that if a constant anomaly is detected by the single-dimensional constant classifier, it is always an anomaly, and otherwise takes the RLS combination classifier output. This heuristic is given by the fact that constant anomalies are easy to detect.

Fisher's method combines the results from several independent hypothesis tests on the same overall hypothesis. Our hypothesis is that the predictors estimate the normal behavior of the input, and we assume that the predictors are independent, because they use a different modeling technique. We can then combine the classifier outputs using $X_{2k}^2 \sim -2\Sigma_{i=1}^{k} \ln(p_i)$, and use the resulting test statistic.

3) Baseline decision: For completeness, we compare also to a baseline rule-based method, which analyzes the first order differences of the data. In this method, when the difference crosses a set threshold, it is labeled as anomalous. Also, when the difference is zero for a number of consecutive samples (constant) it is anomalous.

C. Test platform and datasets

The above described setup was implemented for WSN platforms, and specifically tested on the TelosB platform[1] running TinyOS [27]. To deal with the lack of a floating point unit on this platform, we ported the original implementation of SwiftSeg [28] to fixed-point operations, while the other methods were implemented from scratch in C.

In order to run repeated experiments, we recorded traces of sensor data collected by a deployed WSN. These traces were collected over a period of 5 months and contained 42113 samples for 19 nodes, each with 3 sensors. The dataset contained predominantly constant and spike anomalies that were manually labeled.

Next to the real-world data, we used synthetic datasets with three different types of sensor value models. These were synthesized using linear components, periodic (with a linear and sinusoidal) components and a combination of (correlated) random walks with a sinusoidal components. Each model also contained a Gaussian noise component with mean 0 and a variance of 0.1. The resulting three synthetic datasets contained 28800 data points for 50 nodes. For evaluation purposes, we injected a number of anomalies into the synthetic data, from one of the types *spike*, *noise*, *constant* or *drift*, and labeled their placement. To mimic a real-world situation, relatively few anomalies were injected, such that the datasets were skewed towards a system showing normal behavior.

Both real-world and synthetic traces were tested in TOSSIM [29], an event-based simulator for TinyOS applications.

IV. Evaluation

To evaluate the above outlined approach, we first evaluate the effect of three algorithm parameters. These affect the computation time and the detection performance. Namely:

- L_h, the length of the window used for the windowed statistics, affects the window mean predictor.
- L_s, the length of the window to which SwiftSeg fits a polynomial function, affects the k-step ahead predictor.
- The choice of confidence interval for the detection of anomalies. Our system returns the measurements, and its p-value. The end-user can then decide on a confidence interval, outside of which the systems flags a measurement as anomalous.

Based on this analysis, we determine the parameter tradeoff for use in the following sections. The following sections evaluate the performance of the predictions and, finally, the anomaly detection accuracy.

A. Algorithmic setting and complexity

Because the predictors are trained continuously online, we determine the complexity per update step. This step is performed every time new measurements are taken from the sensors. If the datasets are known and available a priori, we can determine the total number N of samples with which we can determine the overall complexity of our approach. This is calculated by multiplying the below described complexity by N. The computational complexity of our online approach

[1] TelosB, an 8bit MSP430 based platform with 10KB ROM, 48KB RAM and peripherals such as wireless 2.4GHz radio (TI CC2420), temperature, humidity (SHT11) and light sensors (Hamamatsu S1087 series).

Table I
EFFECT OF PARAMETER VALUES L_h AND L_s ON THE COMPUTATION TIME OF AN UPDATE STEP WITH 3 SENSORS, SHOWN IN MILLISECONDS.

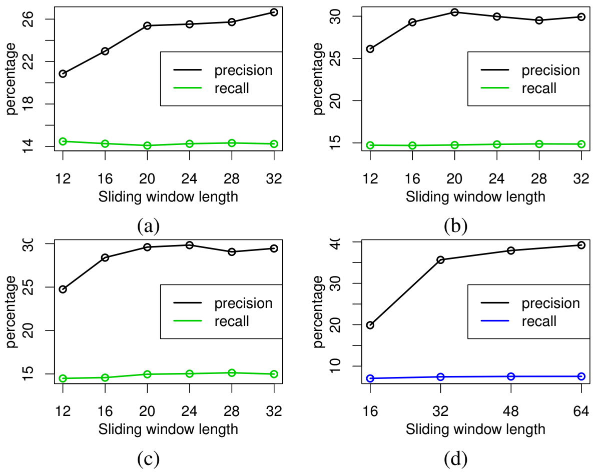
L_s L_h	**12**	**16**	**20**	**24**	**28**	**32**
16	158	163	168	170	173	176
32	192	197	201	204	207	209
48	222	227	231	234	236	239
64	251	256	261	263	266	269

is likely higher than an offline non-incremental method that requires all the data to be available. However, our online incremental learning approach limits memory usage to that of the models, thus enabling implementation on low-resource devices.

We can roughly estimate the complexity of the individual predictors by looking at the number of operations per update step manually. For the windowed statistics, the complexity depends on the window size L_h, i.e. $O(L_h)$. For SwiftSeg, the update step is $O(deg \cdot L_s)$, where deg is the degree of the polynomial used, and L_s the window size. For RLS, the update step is in the order of $O(d^2)$, where d is the number of input dimensions to the algorithm. In our case, for each sensor we input the raw measurement, the 1-step ahead prediction, and the mean and standard deviation over the window of length L_h, i.e. 4 values per sensor. In our setup we use 3 sensors, thus we feed a 12-dimensional feature vector into RLS. Also, due to the lack of FPU on the TelosB platform, non-native fixed-point operations are needed to implement the predictors. Because of this overhead, complexities are actually scaled by a constant factor.

To determine a ballpark estimate of the actual implementation, we have used MSPSim [30], an instruction-level simulator of the MSP430 MCU present on TelosB. Using the debug commands, this simulator returns the MCU time spent in the predictor update step of the sensing application that, in our case, uses 3 sensors. The results are shown in Table I. Detailed profiling showed that the majority of the computation time can be contributed to the fixed-point operations. For comparison, in a typical communication scenario an average package may take 5ms to transmit. However, that does not take into account the number of hops needed to reach the sink node, nor the number of retries due, for example, to collisions. The numbers in Table I are a factor 31 to 53 larger than this transmission time. However, using local calculations does not burden the whole WSN with the (re-)transmission of a measurement packet to the sink. Furthermore, reducing the number of (re-transmission) can further reduce the resource usage of the whole network by adapting the radio duty-cycle.

Next to the influence on complexity, we evaluated the anomaly detection performance of the classifiers that are influenced by the parameters L_h and L_s, i.e., the classifiers using single-dimensional k-step-ahead predictions and windowed statistics. Fig. 5 shows average prediction and recall rates over all the synthetic datasets and anomalies for a fixed confidence interval. The relatively low performance is caused by the difference in detection performance of anomaly types and the skewed nature of the dataset, as further explained in following sections. These average rates, however, do give a good indication for optimal parameter values for all the types of anomalies considered in this paper.



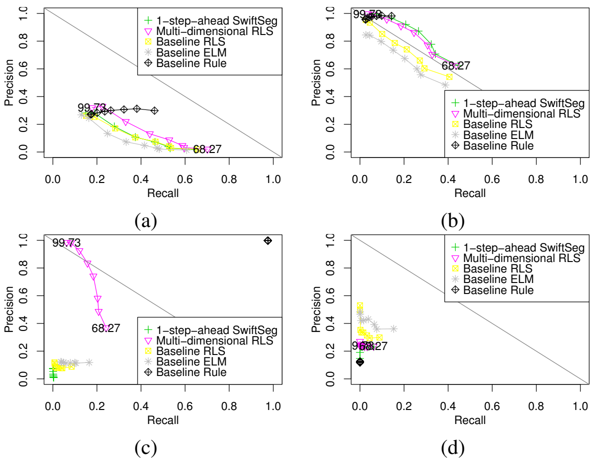
(a) (b) (c) (d)

Figure 5. Effect of window length on anomaly detection performance of single-dimensional time-series classifiers, evaluated on the synthetic data. The effect of L_s is shown for the (a) 1-step-ahead, (b) 2-step-ahead and (c) 3-step-ahead classifier. The effect of L_h on the windowed statistics classifier is shown in (d).

From the complexity analysis, we see that complexity increases when window sizes increase. Therefore, smaller windows should be preferred. The results of precision and recall, seen in Fig. 5, show that precision is increasing significantly up to $L_s = 20$ and $L_h = 32$, whereas recall is not affected much. Larger window sizes have only small benefit on performance, and large penalty on complexity. Thus, aforementioned values are chosen for use in the following evaluations.

An optimal confidence interval may depend on the application, because there is a balance between precision and recall. When one opts for recalling most anomalies, precision suffers, and vice versa. In the case of these methods being used as pre-analysis step, one could also opt to send more data (e.g. the windows) to the sink upon detection, to improve precision/recall rates offline. However, to give an indication of the effect of the confidence interval parameter, we have performed an evaluation using the following intervals: 68.3, 75.0, 80.0, 86.6, 90.0, 95.0, 99.0, and 99.7.

The precision-recall curves that result from evaluating these choices are shown in Fig. 6. We see that confidence intervals of more than 95% provide a good balance, thus in the following evaluations we use 99.0% as the threshold.



(a)

(b)

(c)

(d)

Figure 6. Precision-recall curves for several of the above described methods, for different confidence intervals (from 68.27 to 99.73%). We can distinguish the anomaly types (a) Spike, (b) Noise, (c) Constant and (d) Drift.

While, at a first glance, the precision and recall for spikes might seem low, it should be considered that our datasets are skewed towards the normal class (i.e. normal behavior). For instance, the number of spikes in the dataset is about 0.1% of the size of the dataset. The same percentage holds for the other anomaly types, however, their duration is longer. Thus, for the other data types, the number of anomalous samples is in the order of 10% of the dataset. Therefore, in following sections we evaluate the detection performance using different, but related, measures, namely true positives (TP), false negatives (FN) and false positives (FP), expressed as percentages of the number of anomalies.

B. Prediction residuals

Given the above choices of parameters for window lengths, we can also look at the prediction residual for different methods next to the anomaly detection performance. These statistics are taken over the initial part of the synthetic data, where no anomalies are injected. Because we have added Gaussian noise to the synthetic datasets, with variance 0.1, we expect the predictions to be no better than 0.1 units accurate on these datasets.

The prediction error statistics are shown in the box plots in Fig. 7. The large variance and outliers for the baseline ELM are due to the random initialization of input node weights combined with fixed-point precision. Furthermore, the mean of the window of length L_h does not give a very accurate prediction result. Other than that, we can see that the baseline RLS method, as presented in [8], performs similarly to the k-step-ahead SwiftSeg predictions. However, the combination of SwiftSeg and the window by using RLS shows the best prediction on average.

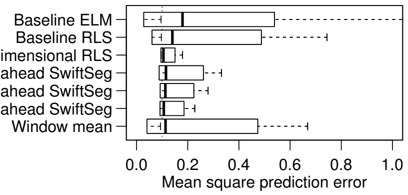


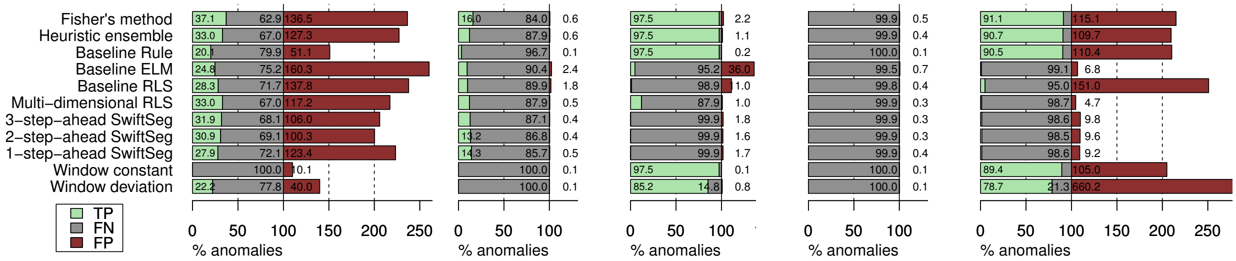
Figure 7. Comparison of the mean square prediction error of the different methods on the synthetic data, which has Gaussian noise of mean 0 and a variance of 0.1.

C. Comparative analysis of anomaly detection performance

Our proposed approach is compared against the baselines outlined in section III-A3 for both the synthetic and real-world datasets. The precision-recall curves already gave an indication of performance earlier, but in this section we further analyze these results by looking at the true positive (TP), false negative (FN) and false positive (FP) detection ratios of anomalies. As mentioned before, our datasets are skewed towards the normal class, and the precision and recall for spikes seem low. However, the number of spikes in the dataset is about 0.1% of the size of the dataset, while the number of anomalous samples of other anomaly types is in the order of 10% of the dataset, merely because their duration is longer.

Fig. 8 shows a comparison of the anomaly detection performance of the methods outlined in previous sections, for a fixed confidence interval threshold of 99.0%. Due to the different normal-state modeling properties of each of the classifiers, we have split the performance results per anomaly type. It can be seen that not all anomaly types are easily detected. This holds mainly for drift that, when the drift rate is slower than the learning rate, can go undetected. If detections occur, these are mostly at the onset of the anomaly. With TP rates of less than 1% in our synthetic dataset injected with drift anomalies, we can conclude that our approach, and also the baseline methods, fail to detect this model of the drift anomaly. This is likely due to fast learning rates that adapt the models to the occurrence of drift. Nevertheless, our investigations show that the few TP which are found indicate the onsets of drift.

Constants can be easily detected by the simple rule that governs the standard deviation over a window of size L_h. This shows in the results seen in Fig. 8c, having 97.5% true detections, and very few FP. The classifier using this window shows very good detection performance. The same holds for the window deviation classifier, that detects large parts of the anomaly onset. The single-dimensional k-step-ahead classifiers, however, show similar performance to detecting drift anomalies. They detect merely a small percentage of the constant anomaly onsets. In this case, we do see that the fusion of predictions and measurements, using RLS, has a benefit in detecting these onsets over both the k-step-ahead classifiers and the baselines. Moreover, both the ensemble



(a) Spike (b) Noise (c) Constant (d) Drift (e) Real-world mixed

Figure 8. True Positives (TP), False Negatives (FN) and False Positives (FP) as the ratio of number of anomalies in dataset.

methods favor the good detection performance of the rule-based classifiers and, as a consequence, perform well.

The detection performance of our approach regarding noise anomalies is better. Compared to the baseline classifiers, having 10.1% TP and 1.8% FP, our single-dimensional k-step-ahead classifiers have a higher TP ratio of 12.1%, and lower FP ratio. The window deviation classifier does not detect the noise anomalies, mainly because the short window incorporates the anomalous samples. While the multi-dimensional RLS classifier in this case does not benefit from the performance of the single-dimensional classifiers of our approach, the ensemble method does. With a TP ratio of 16% and only 0.6% FP it performs best for this anomaly type. It is worthy to note that the noise anomaly affects a region of data in time, and thus that central (post) analysis around the time of detection could show the full noise anomaly, and thus improve the TP rate further.

For most methods, spikes are detected well, but with high FP ratio. As described earlier, this might partly be due to the skewed class distribution in the dataset. Again, the methods in our approach perform better than the baseline methods, with higher TP (i.e. 33.0% TP for the multi-dimensional RLS vs 28.3% TP for the baseline RLS) and lower FP ratio. The window deviation classifier does perform better than the baseline rule with 22% TP. On the other hand, the k-step-ahead classifiers detect up to 33% TP at a cost of higher FP ratio. Furthermore, these classifiers seem to benefit slightly from higher k, i.e. predict further in time, where $k = 2$ seems to balance TP (30.9%) and FP (100.3%) ratio best. Again the multi-dimensional RLS classifier has a higher TP ratio than the single classifiers, with 33%, but the FP suffers slightly. Also here, it seems that Fisher's ensemble method benefits from the individual results, with 37% TP ratio and FP ratio comparable to the baseline RLS.

The heuristic ensemble method favors the window constant classifier if it detects an anomaly, and otherwise uses the multi-dimensional RLS classifier. In general, this results in good TP ratios for all anomaly types, however, at a cost of somewhat higher FP ratios. Using Fisher's method to combine the classifiers ensemble output, however, does also improve on the TP ratio.

The last dataset on which we evaluated our approach is the real-world data. These data contain mostly constant anomalies, which shows in the excellent detection performance of the window method and the rule-based baseline. The remaining anomalies in the dataset are of multi-sample spikes, which could also be considered very short drifts, caused by direct sunlight on the sensor. Due to their gradual nature, they are hard to detect, which shows in the performance of the k-step-ahead and RLS combination classifiers. Moreover, the window deviation classifier does detect a reasonable amount of TP, but has a very high FP ratio, of over 600%. The ensemble of classifiers does benefit slightly from the detection of onsets from the RLS combination and k-step-ahead classifiers, which shows in a TP ratio of 91.1% compared to 90.5% for the rule based baseline.

V. Conclusion

In this work, we have designed and evaluated an online embedded approach to anomaly detection using an ensemble of classifiers. Compared to other online embedded methods, this approach improves upon the detection of anomalies. The ensemble allows targeting of different types of anomalies using specific classifiers, but at the cost of added computational complexity. In our case, two types of single-dimensional classifiers complement each other to detect spike, noise or constant anomalies, and perform on par, or outperform the baseline methods. Drift is, however, difficult to detect by any method. Furthermore, we show that using RLS to combine several single-dimensional time series, and predictions thereof, improves both prediction and anomaly detection performance. Overall, the combination of methods improves prediction and detection, allowing to not only detect anomalies, but also to better estimate the true value of the monitored system.

These results are a promising step towards a two-stage anomaly detection approach, where embedded online anomaly detection provides a pre-analysis for a central analysis system. Such an approach would save the WSN network valuable resources, such as energy, through reducing communication towards this central system. Future research may add other classification methods to the ensemble, for instance to target a specific type of anomaly, in order to provide more accurate detections. Moreover, future investigations may explore the use of local neighborhood knowledge in the form of, for instance, collaborative learning among local neighbors, or evaluating the information and performance of neighboring WSN nodes. Such mechanisms can benefit the performance more due to the integration of localized information.

Acknowledgment

INCAS^3 is co-funded by the Province of Drenthe, the Municipality of Assen, the European Fund for Regional Development and the Ministry of Economic Affairs, Peaks in the Delta.

References

[1] A. Liotta, "The cognitive net is coming," *IEEE Spectrum*, vol. 50, no. 8, pp. 26–31, August 2013.

[2] S. Galzarano, G. Fortino, and A. Liotta, "A learning-based mac for energy efficient wireless sensor networks," in *Proc. of the 7th International Conference on Internet and Distributed Computing Systems*. Springer, LNCS, September 2014.

[3] L. Bencini, F. Chiti, G. Collodi, D. Di Palma, R. Fantacci, A. Manes, and G. Manes, "Agricultural monitoring based on wireless sensor network technology: real long life deployments for physiology and pathogens control," in *Sensor Technologies and Applications, 2009. SENSORCOMM'09. Third International Conference on*. IEEE, 2009, pp. 372–377.

[4] K. Martinez, J. K. Hart, and R. Ong, "Deploying a wireless sensor network in iceland," in *GeoSensor Networks*. Springer, 2009, pp. 131–137.

[5] J. Åkerberg, M. Gidlund, T. Lennvall, J. Neander, and M. Björkman, "Efficient integration of secure and safety critical industrial wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, no. 1, pp. 1–13, 2011.

[6] R. Jurdak, X. Wang, O. Obst, and P. Valencia, "Wireless sensor network anomalies: Diagnosis and detection strategies," *Intelligence-Based Systems Engineering*, pp. 309–325, 2011.

[7] E. Fuchs, T. Gruber, J. Nitschke, and B. Sick, "Online segmentation of time series based on polynomial least-squares approximations," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 12, pp. 2232–2245, 2010.

[8] H. H. W. J. Bosman, A. Liotta, G. Iacca, and H. J. Wortche, "Anomaly detection in sensor systems using lightweight machine learning," in *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7–13.

[9] K. Ni, N. Ramanathan, M. Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen, and M. Srivastava, "Sensor network data fault types," *ACM Transactions on Sensor Networks*, vol. 5, no. 3, p. 25, 2009.

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

[11] Y. Yao, A. Sharma, L. Golubchik, and R. Govindan, "Online anomaly detection for sensor systems: A simple and efficient approach," *Performance Evaluation*, 2010.

[12] R. Biuk-Aghai, Y.-W. Si, S. Fong, and P.-F. Yan, "Individual movement behaviour in secure physical environments: Modeling and detection of suspicious activity," in *Behavior Computing*, L. Cao and P. S. Yu, Eds. Springer London, 2012, pp. 241–253.

[13] C. Phua, V. Lee, K. Smith, and R. Gayler, "A comprehensive survey of data mining-based fraud detection research," *Arxiv preprint arXiv:1009.6119*, 2010.

[14] S. Kao, A. Ganguly, and K. Steinhaeuser, "Motivating complex dependence structures in data mining: A case study with anomaly detection in climate," in *2009 IEEE International Conference on Data Mining Workshops*. IEEE, 2009, pp. 223–230.

[15] C. Wang, K. Viswanathan, L. Choudur, V. Talwar, W. Satterfield, and K. Schwan, "Statistical techniques for online anomaly detection in data centers," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*. IEEE, 2011, pp. 385–392.

[16] Y. Zhang, N. Meratnia, and P. Havinga, "Outlier detection techniques for wireless sensor networks: A survey," *Communications Surveys & Tutorials, IEEE*, vol. 12, no. 2, pp. 159–170, 2010.

[17] M. Xie, S. Han, B. Tian, and S. Parvin, "Anomaly detection in wireless sensor networks: A survey," *Journal of Network and Computer Applications*, 2011.

[18] M. Chang, A. Terzis, and P. Bonnet, "Mote-based online anomaly detection using echo state networks," in *Distributed Computing in Sensor Systems*. Springer, 2009, pp. 72–86.

[19] F. Serdio, E. Lughofer, K. Pichler, T. Buchegger, M. Pichler, and H. Efendic, "Fault detection in multi-sensor networks based on multivariate time-series models and orthogonal transformations," *Information Fusion*, 2014.

[20] L. Ljung and S. Gunnarsson, "Adaptation and tracking in system identification - a survey," *Automatica*, vol. 26, no. 1, pp. 7–21, 1990.

[21] C. Lo, J. Lynch, and M. Liu, "Reference-free detection of spike faults in wireless sensor networks," in *Resilient Control Systems (ISRCS), 2011 4th International Symposium on*. IEEE, 2011, pp. 148–153.

[22] F. L. Bookstein, "On a form of piecewise linear regression," *The American Statistician*, vol. 29, no. 3, pp. 116–117, 1975.

[23] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, pp. 993–1001, 1990.

[24] D.-I. Curiac and C. Volosencu, "Ensemble based sensing anomaly detection in wireless sensor networks," *Expert Systems with Applications*, vol. 39, no. 10, pp. 9087–9096, 2012.

[25] H. H. W. J. Bosman, A. Liotta, G. Iacca, and H. J. Wortche, "Online extreme learning on fixed-point sensor networks," in *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*. IEEE, 2013, pp. 319–326.

[26] R. A. Fisher, *Statistical methods for research workers*. Genesis Publishing Pvt Ltd, 1925.

[27] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer *et al.*, "Tinyos: An operating system for sensor networks," in *Ambient intelligence*. Springer, 2005, pp. 115–148.

[28] E. Fuchs, T. Hanning, O. Schwarz, T. Gruber, J. Nitschke, and B. Sick, "Fast function approximation library." http://www.ies-research.de/Software

[29] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: Accurate and scalable simulation of entire tinyos applications," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003, pp. 126–137.

[30] J. Eriksson, A. Dunkels, N. Finne, F. sterlind, and T. Voigt, "MSPsim – an Extensible Simulator for MSP430-equipped Sensor Boards," in *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*, jan 2007.