Anomaly detection in networked embedded sensor systems

Anomaly detection in networked embedded sensor systems

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus prof.dr.ir. F.P.T. Baaijens, voor een
commissie aangewezen door het College voor Promoties,
in het openbaar te verdedigen op
maandag 12 september 2016 om 16.00 uur

door

Hedde Hendrik Wobbele Jan Bosman

geboren te Groningen

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof.dr.ir. A.B. Smolders
1^e promotor:	prof.dr. A. Liotta
copromotor(en):	dr. G. Exarchakos
	dr. G. Iacca (University of Lausanne, INCAS^3, École polytechnique fédérale de Lausanne)
leden:	prof.dr. M. Aiello (Rijksuniversiteit Groningen)
	prof.dr. J.J. Lukkien
	prof.dr. F. Neri (De Montfort University)
	prof.dr. H.J. Wörtche

Het onderzoek of ontwerp dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

A catalogue record is available from the Eindhoven University of Technology Library.
ISBN: 978-90-386-4125-6
NUR 984
Title: Anomaly detection in networked embedded sensor systems
Author: H.H.W.J. Bosman
Eindhoven University of Technology, 2016.

Keywords: Anomaly detection / Outlier detection / Machine Learning / Embedded Systems / Sensor Networks

The cover of this work is made of sensor data from temperature (front) and light (back) sensors on one of the TelosB sensor nodes from the Indoor WSN that was deployed during this thesis work. This data was mapped to an outward clockwise spiral, where a full circle represents 3 days of data. Noteworthy anomalies in the temperature data (front) are the large spikes caused by direct sunlight hitting the sensor, and data remaining constant due to connection issues and maintenance. These anomalies are observed in the light data too. Interestingly, one can see the punctuality of the cleaning lady as a short period of increased light intensity in the early mornings.

Typeset using LaTeX, printed in The Netherlands.

Acknowledgements

I would like to take this opportunity to acknowledge everyone who played a role in enriching my time during my PhD years, and making it possible. First, I would like to thank all the people at INCAS^3 in the present and past. They have inspired, supported and provided great opportunities during this journey. Especially I'd like to thank Heinrich Wörtche and John van Pol for their support and opportunity to pursue my studies.

Second, I would like to thank my promoter, Antonio Liotta, for his guidance and support. Next, I'd like to thank my co-promoters, Giovanni Iacca and Georgios Exarchakos, who have motivated and inspired me in carrying out the research. Further, I am also very grateful to the members of my PhD committee for reviewing, commenting and approving this work. I would also like to thank the PHOENIX project and its members for supporting my final months.

Furthermore, I'd like to thank the current and former people in the Smart Networks group of Prof. Liotta: Vlado Menkovski, Roshan Kotian, Decebal Mocanu, Maria Torres Vega, Stefano Galzarano, and Michele Chincoli. Also, I'd like to express my appreciation to all the current and former members of the ECO group, lead by Prof. Ton Koonen, who helped shine different lights on all sorts of subjects. In no particular order: Huug de Waardt, Henrie van den Boom, Frans Huijskens, Johan van Zantvoort, Patty Stabile, Kevin Williams, Nicola Calabretta, Chigo Okonkwo, Oded Raz, Eduward Tangdiongga, Shihuan Zou, Zizheng Cao, Gonzalo Guelbenzu de Villota, Ketemaw Mekonnen, Netsanet Tessema, Robbert van der Linden, Roy van Uden, John van Weerdenburg, Frederico Forni, Wang Miao, Chenhui Li, Simone Cardarelli, Nikolaos Sotiropoulos, Prometheus DasMahapatra, José Hakkens, Jolanda Levering, and Brigitta van Uitregt-Dekkers, and all the support personnel and other people whom I may have forgotten. Additionally, I'd like to express my gratitude to the TU/e, especially the electrical engineering department, for providing the facilities.

Within INCAS^3 I thoroughly enjoyed the discussions with, and learning experience from, among others, Tjeerd Andringa, Dirkjan Krijnders, Danielle Dubois, Caroline Cance, Maria Niessen, Gineke ten Holt, Elena Talnishnikh, Matt Coler, Doina Bucur, Manuel Mazo, and Arturo Tejada Ruiz. Furthermore, the creativity, ingenuity and fun with the engineers, among others Georges

Meinders, Erik Kallen, Arjen Bogerman, Cor Peters, Rajender Badam, Victor Stoica, and Jan Stegenga. Of course I'd also like to thank the other PhDs during my INCAS^3 time that shared their wisdom, fun and experience, being Peter Dijkstra, Froukje Veldman-de Roo, Yiying Hao, Yafei Wang, Edda Bild, Erik Duisterwinkel, Mike van Diest, Shaojie Zhuang, Charissa Roossien, Dian Borgerink, Anil Yaman, and others who may follow. I'd like to thank also the numerous support personnel, administration and secretaries, and of course, all the other bright people who are, and once were, a part of this exciting journey at INCAS^3.

Finally, I would like to thank my family for their support for all these years and, last but not least, special thanks to Joanne Oh for her love and support and for being there for me in the ups and downs.

Summary

Anomaly detection in networked embedded sensor systems

In the past decade, rapid technological advances in the fields of electronics and telecommunications have given rise to versatile, ubiquitous decentralized embedded sensor systems with ad hoc wireless networking capabilities, giving body to the decade long vision of the internet of things (IoT). Envisioning seamless networking between classic networked systems and ubiquitous smart things, often consisting of a set of resource-limited embedded platforms (nodes) endowed with sensors. One of the first classes of IoT devices are wireless sensor networks (WSNs). Typically, these systems are used to gather large amounts of data, while the detection of anomalies (such as system failures, intrusion, or unanticipated behavior of the environment) in the data (or other types of processing) is performed in centralized computer systems.

The research in this thesis aims to provide an online decentralized anomaly detection framework, applicable to networked embedded systems. In spite of the great interest that anomaly detection attracts, the systematic analysis and porting of centralized anomaly detection algorithms to a decentralized paradigm (compatible with the aforementioned sensor systems) has not been thoroughly addressed in the literature. We approach this task from a new angle by assessing the viability of localized (in-node) anomaly detection framework based on machine learning. Therefore, the goal of this framework is to provide unsupervised anomaly detection methods for unknown environments (providing spatio-temporally correlated data) while operating with limited resources, and to evaluate the framework not only with simulated data, but also in real-world scenarios.

To this end, we first define several environmental monitoring scenarios based on both synthetic and real-world datasets. Then, we implement and analyze single and multi-dimensional input classifiers that are trained incrementally online and whose computational requirements are compatible with the limitations of networked embedded platforms. Our evaluation demonstrates that learning of linear, non-linear and polynomial models is viable in low-resource embedded systems. Next to that, we show how these methods are adapted to deal with the limited memory and processing resources in embedded platforms, which may be applied to other learning methods. After that, mul-

tiple classifiers are combined through fusion (re-using the multi-dimensional input classifiers) and ensemble methods (such as majority voting and Fisher's method). We show that fusion methods improve detection performance and that different ensemble methods have specific effects on performance. For instance, majority voting increases precision at the cost of recall, while Fisher's method improves results in general. Finally, we present a methodology to include local neighborhood information in the detection framework. Through the use of aggregation operators, we demonstrate that information from a dynamic but correlated neighborhood is beneficial to anomaly detection performance, mainly recall. Through an extensive evaluation campaign, we show the effects of algorithm parameters, type of ensemble, neighborhood size and neighborhood correlation on performance. The results demonstrate the feasibility of the proposed online decentralized anomaly detection framework on networked embedded systems.

Overall, this work contributes in different ways to the fields of distributed embedded systems and machine learning: a) by demonstrating the suitability of incremental learners in embedded systems, b) by providing a methodology to adapt those methods to embedded systems, c) by showing the benefits of fusion and ensemble methods, d) by the inclusion of neighborhood information and e) by showing ways to evaluate the proposed unsupervised methods. Furthermore, this work has practical implications in machine learning and in different applications areas of WSN, IoT and embedded systems in general, such as large-scale monitoring of environments or distribution networks.

Contents

List of Figures

List of Tables

Acronyms

ANN artificial neural network

ANOVA analysis of variance

ARIMA auto-regressive integrated moving average

ARMA auto-regressive moving average

CPU central processing unit

CTP collection tree protocolA low-power hierarchical protocol

DSP digital signal processor

ELM extreme learning machine

ESN echo state network

EWMA exponentially weighted moving average

FA function approximation

FN false negative

FP false positive

FFT fast Fourier transform

GCC GNU C compiler

GHz gigahertz

GIS geographic information system

GSB Grand St. Bernard, a deployment of the SensorScope project

HMM hidden Markov model

ICA independent component analysis

IoT internet of things

IP internet protocol

IPv4 internet protocol version 4

IPv6 internet protocol version 6

IPv6LoWPAN internet protocol version 6 (IPv6) over low-power wireless personal area networks

KB kilobyte

KNN k-nearest neighbors

LLSE linear least squares estimation

LOF local outlier factor

MANET mobile ad-hoc network

MCU micro controller unit, integrating CPU, memory and peripherals

MHz megahertz

mJ millijoule

ms millisecond

mW milliwatt

OS-ELM online sequential extreme learning machine

PCA principal component analysis

PLE path-loss exponent

PLS partial least squares

RAM random-access memory

RDA receptor density algorithm

RL reinforcement learning

RLS recursive least squares

RMS root mean square

RMSE root mean square error

RNN recurrent neural network

ROM read-only memory

SLFN single layer feed-forward neural network

SLOM spatial local outlier measure

SOM self-organizing map

SVM support vector machine

TN true negative

TP true positive

WSN wireless sensor network

WSAN wireless sensor and actuator network

Chapter 1

Introduction

The past decades have seen enormous technological advances. Together with ever decaying prices of electronic components, these have made networked embedded systems ubiquitous in our lives. Such embedded systems can be found in, for instance, home automation, automated transportation or large scale environmental data collection [1]. One can also think of smart phones, smart watches, or internet enabled fridges and washing machines. Researchers and developers of these systems are driven by a vision of an internet of things (IoT), a world-wide network of electronic devices. These devices are, in most cases, endowed with sensing, actuating and networking capabilities, and often they are connected to the Internet to enable data processing at remote (central) locations (sometimes referred to as 'the cloud'). The increasing number of devices results in a large amount of data to be sent, stored and processed to enable a desired collective behavior.

The applications of IoT devices harbor some tough challenges related to the envisioned scale and life-time of the network deployments. While many challenges originate from the limited energy available (limiting other resources, such as communication and processing capabilities), we focus on the challenge of analyzing large amounts of data (requiring large communication and processing capacities). All these data, stemming from monitored large outdoor areas or from the many networked appliances in a smart home, are often analyzed in order to find specific information that is meaningful for the application to act upon, at a point in time. For example, one can think of earthquake detection [2] or precursors to a failing machine [3]. Often, the information of interest is out of the ordinary, or, anomalous [4]. Detecting such anomalies on the networked embedded systems, for example, enables pre-filtering of the data collected over the network, provides measures of reliability of the data, enables timely response to local anomalies and events, or creates more autonomous systems. Anomaly detection, therefore, may play an important role in managing the problems arising in such systems.

Anomaly detection is concerned with the detection of events, behaviors or

patterns that are unexpected relative to a concept of what is normal [4]. A typical example is the detection of fraud in, for instance, credit card transactions or identity falsification [5]. Another example is the detection of climate events, such as heat waves and drought, which depend on multiple variables, such as location (drought in the Sahara desert, for instance, is not anomalous) [6]. Moreover, anomaly detection approaches are also used to detect intrusions in information systems, ever more relevant in present day cloud computing [7]. In general, anomaly detection methods require abundant resources, resulting in IoT and wireless sensor network (WSN) systems that send all measured data to a central system or to the cloud for processing. This thesis proposes an unsupervised decentralized online anomaly detection framework for resource-limited embedded systems, to form a complement to the more classical centralized systems.

1.1 Networked embedded sensor systems

While at present white goods, smart cities and buildings are being equipped with IoT technology, one of the earliest IoT related systems were (and still are) WSNs. These consist of a set of nodes, which are (generally) resource-limited embedded platforms endowed with sensors (converting a form of energy into electrical and eventually digital signals) and sometimes actuators (converting electrical signals into physical actions, such as motors), in the latter case also called wireless sensor and actuator network (WSAN) or, when node mobility is involved, a mobile ad-hoc network (MANET). The embedded platform normally has, next to the sensors and actuators, a micro controller unit (MCU) which controls the functioning of the device, and a power supply in the form of, for example, batteries, solar cells or mains electricity. The MCU usually has small amounts of random-access memory (RAM) and read-only memory (ROM), typically in the order of kilobytes (KBs), and executes instructions in the order of 10 megahertz (MHz). Of course these will increase as technology advances. Furthermore, these platforms can communicate wirelessly, using low-power radio modules, to construct an ad hoc network with each other and with one or more sink nodes (i.e., nodes connected to a central facility for data storage and analysis). Commonly, the protocols used for the communication are based on established standards, such as internet protocol version 6 (IPv6) over low-power wireless personal area networks (IPv6LoWPAN) [8], or Zigbee [9] and include energy-saving modes such as duty cycling. The resources of these nodes are limited, not only for cost-saving in the envisioned large scale scenario, but also for energy-saving. This will be even more significant when more powerful processors or more accurate sensors, that require higher energy consumption, are deployed.

For over a decade, the WSN community has focused mainly on the optimization of resource usage through, for example, smart network protocol de-

sign and smart sensing strategies. Recently, however, the community's focus is shifting to the applications of WSN [10]. Typical applications can be found in agriculture, where WSN are used to provide detailed insight on soil conditions [11], or in environmental monitoring, where they are used, for instance, to measure the effect of global warming on glaciers [12]. Other application domains include civil engineering (with various successful case studies in infrastructural monitoring [13], optimal tunnel lighting conditions control [14], water distribution network monitoring [15]), and health care (with many applications such as fall monitoring, medicine intake or medical condition monitoring [16]). Lately, WSNs are slowly being adopted also in industrial settings [17], although these applications are tightly controlled due to stringent reliability, safety and security requirements.

Such applications, as well as those in other IoT scenarios, usually require numerous nodes to be deployed in remote locations. To make such systems affordable, tight costs-saving often compromises on the quality of the sensors and the hardware resources available on each node (such as battery and computing elements), while the overall measurement quality of the networked system is commonly ensured by a high level of redundancy in measurements, resulting in large amounts of data. The most compelling challenge of these applications is to analyze, possibly in real-time, these large amounts of data. For domain specialists, such an analysis could provide new inferred knowledge about the sensed environment or processes that, in turn, can be used to improve their modeling. However, to make this analysis possible, automated analysis strategies [18] and big data analysis techniques [19] are needed, since such large datasets cannot be processed manually.

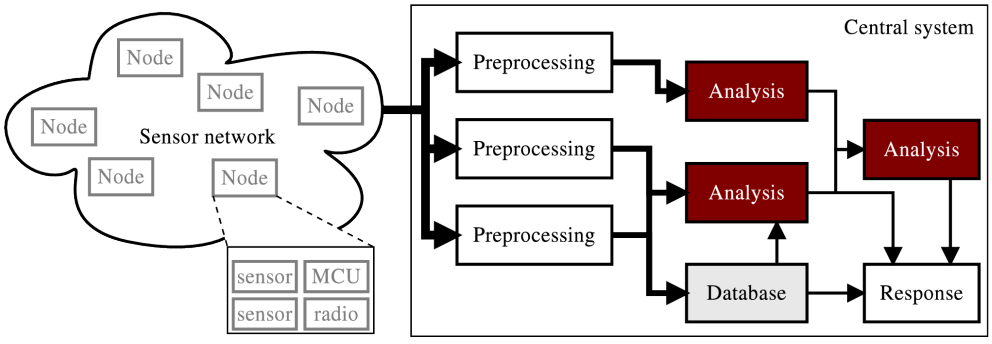


Figure 1.1: Generic architecture for common anomaly detection systems

The automated analysis to extract meaningful information, often called data mining, is commonly performed centrally due to an abundance of resources such as memory and processing, allowing complex models of the environment to be constructed. A generic architecture of such systems can be seen in Figure 1.1. The Argo project, for example, has more than 3500 sensor nodes floating around the worlds oceans. They collect temperature and salinity mea-

surements at different depths, and send the collected data once the so-called floats reach the surface. The collected measurements are then analyzed automatically, and manually checked by domain experts [20, 21]. The extracted information can then be used to get a better understanding of the measured environment, to generate models for prediction, or to detect anomalous events, behaviors or patterns. A good overview of anomaly detection techniques for WSN data is provided in several literature surveys [18], [22], and [23]. However, these surveys pinpoint a major open issue: Remote locations or environmental circumstances, together with a limited energy supply (communication costs are relatively high), make the collection of data resource heavy. Moreover, due to the nature of the measured processes, data from different sensors and locations tends to be highly redundant. Therefore, networked embedded sensor systems can benefit from an initial stage of decentralized online (in-node) anomaly detection (or other processing) to reduce resource usage (of communication or storage), to reduce redundancy, or to indicate (and respond to) information of interest at an early stage.

1.2 Problem definition - Anomaly detection

The main problem addressed in the thesis is the lack of a decentralized online anomaly detection framework for resource-limited embedded systems such as IoT and WSN. To us, humans, selecting the anomalous information, which is often of interest, seems to be a simple task. We easily hear our own name in noisy environments (i.e. a large amount of auditory data) [24]). However, auditory or visual attention can be directed by us through a given task or goal. Thus, while we are competent in attending to the occurrence of hand claps, for example, we do not notice that on the background a set of male ball game players is replaced by female players [25]. Computer systems, on the other hand, are designed for specific goals, and while they can select interesting or anomalous information that it is designed for, unexpected behavior (i.e. behavior not known in advance or not programmed) may go unnoticed.

With the advent of big data anomaly detection approaches are becoming popular. Today, anomaly detection methods are commonly used; for instance to detect anomalous behavior or behavioral patterns in security card access [26]. Furthermore, anomaly detection is also common in large (known environment) data stream mining systems, with abundant processing power, to detect data of interest, or significance. Data center management, for example, can benefit from anomaly detection, which detects problems in system performance and, by diagnosing these problems, can also provide potential remedies [27]. On the other hand, the increased use of networked embedded systems and the increase of their resources have sparked a new interest in online, and possibly decentralized, detection techniques. However, the resources are still too limited to implement known methods on lightweight embedded WSN and IoT systems.

The challenges and limitations of these systems, as described in Section 1.1, require a dedicated framework to provide such data analysis. The lack of memory and the high communication cost, for example, limit the scenario where all WSN nodes send all information to a central facility for storage and processing (as in cloud-assisted WSNs) [28]. Other short-comings from related anomaly detection techniques, as surveyed in [22], are not taking into account multivariate data, static node placement, requiring user input and poor usage of spatio-temporal correlations in sensor nodes and sensor node attributes. Next to known anomaly detection and diagnosis strategies, for specific anomaly types occurring in WSN applications, another point of attention is the usability of the different methods, that is, how the user interacts with the anomaly detection system [18]. Moreover, security threats such as anomalous network activities gain increasing attention in WSN and IoT applications. While many related works focus on offline data mining and computational intelligence approaches, they do recognize challenges such as limited energy resources and the lack of a uniform performance evaluation standard [23].

From these surveys, their research outlooks, and the related work introduced in Chapter 2, we can extrapolate some clear challenges for anomaly detection in sensor systems. For one, while we see an increasing number of online and distributed anomaly detection methods, the adaptation to embedded sensor networks takes time. However, this may have to do with aforementioned challenges such as limited resources. Also, the evaluation of anomaly detection methods does not yet have a standard. This means that we cannot easily compare one method to another, but always have to look at the results in context (of, for example, application and problem). Furthermore, the selection of feature preprocessing and of metrics to compare these features is an open area. With the right metric and transformation for the application, anomalies may become more apparent, and thus the detection accuracy increases. However, the proper choice of models and assumptions for a given system, such as the usage of correlations existing in that system, is often forgotten. We often make assumptions based on Gaussian noise, while a given system may have a Chi-square distribution of noise. Moreover, by making use of the correlations in the system, and changes therein, detection accuracy may be improved. The last challenge for anomaly detection in sensor systems is the usability of the anomaly detection methods. This is an often overlooked challenge by the system designer or researcher, but should be accounted for to increase user acceptance. Giving insights into why a system detects an anomaly, for example, may give the user more confidence in the system.

This thesis focuses on a subset of these challenges, namely the adaptation of online anomaly detection methods to embedded systems, the use and adaptation of evaluation methods for unsupervised anomaly detection methods, and the assumptions used for the typical application of WSN systems. Next, we describe a common categorization of anomalies and the scenarios in which we evaluate the anomaly detection methods.

1.2.1 Anomalies

Anomalies can stem from different sources and manifest in various behaviors. When a system is known and formally modeled, the different sources and manifestations of anomalies can be predicted. This allows system and controller designers to implement fault tolerance techniques to increase, for instance, the safety at a chemical plant. However, in the typical application of WSN and IoT systems, neither the exact deployment environment nor the causes of anomalies are known *a priori*. Therefore, it is a challenge to detect all anomalies. Further analysis of these anomalies (by a human expert or expert system) may then reveal the source or meaning of the anomaly, the decision if the anomaly is critical and possible future actions.

Nevertheless, we can categorize some generic sources of anomalies [18, 22]:

Environment: The environment may give rise to certain events that do not often occur, but might be interesting to the application users. For example, a fire or a deadly increase in Carbon Monoxide concentration. Another source of anomalies is the change in spatial or temporal correlation, i.e., the correlation between sensors, and single sensor history, which may indicate a not-yet-modeled source.

System: The sensor system, or a part of it, may fail. In the case of WSNs, this may be a sensor or a whole node. Common failures originate from the battery, software bugs, sensor hardware malfunctions, or short circuits from, for instance, water damage.

Communication: Similar to communication between humans, the network communication may also be unreliable. This shows well in wireless communications. For instance, packets may be dropped or delayed due to packet routing or interference. While such anomalies are often handled by a communication protocol (such as TCP/IP), applications designers may opt for protocols without guarantees (e.g. on delivery) due to constraints on energy or communication efficiency.

Attack: An attacker can create anomalies that seem to stem from any of the above sources. Most common are communication attacks, as it is relatively easy to inject malicious packets into a wireless network, for example, without physically tampering with a device [29, 30]. Other attacks may be physical in nature, such as destroying a node, or interference with the sensed environment.

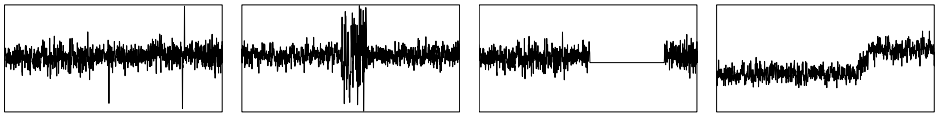
A common categorization of anomalies stemming from the above mentioned sources is identified in the literature as spike, constant, noise and drift [31, 32, 33], also depicted in Fig. 1.2.

Spikes: Short-duration peaks in the data, caused by, for instance, bit-flips, buffer overruns, malfunctioning hardware connections, or disturbances in the environment.

Noise: An unexpected increase in variance, detected through historical data which may, for example, be caused by a depleted battery or electromagnetic interference.

Constant: When changes and noise in the monitored process do not lead to any changes in the measured value, it stays (and is named) constant. Such an anomaly may, for instance, be caused by a sensor getting stuck (physically or in software) or by a loose electrical connection.

Drift: An off-set measurement value relative to the monitored process which is constant or changing. This is due to changes in, for example, sensor performance, calibration, or environment correlations due to an unmodeled process.



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

Figure 1.2: Injected anomalies in noisy linear data

The exact manifestations of these categories vary per application, but this categorization gives us a way to identify possible anomalies in existing data, and to inject synthetic anomalies in data.

1.2.2 Scenarios

To assess our proposed framework's ability to detect the four aforementioned anomalies, throughout this thesis we employ six scenarios, including both synthetic and real-world data. The synthetic data contain anomalies of the above categories injected on purpose at specific moments in time. Thus, each sample is known to be either normal or anomalous (i.e., it is labeled) beforehand, which allows us to compute precise anomaly detection statistics. In the real-world scenarios there is no *a priori* knowledge about the samples, so the detection performance can be estimated only after a preliminary offline manual analysis and labeling of the data, as described at the end of this section.

We consider first the synthetic datasets. These consist of an addition of various signals shown in Figure 1.3. The first synthetic dataset, $Synthetic_L$, consists of 3 signals per node that follow a noisy line. That is, each signal can be described as $s = at + b + \mathcal{N}(0, 0.1)$, where t is time, a is the slope, between -1 and 1, b is a constant intercept between 0 and 1, and $\mathcal{N}(0, 0.1)$ ia a zero-mean Gaussian noise with variance 0.1.

The second synthetic dataset, $Synthetic_{LS}$, is similar to the first, but also includes a periodic signal that simulates periodic environmental changes, such

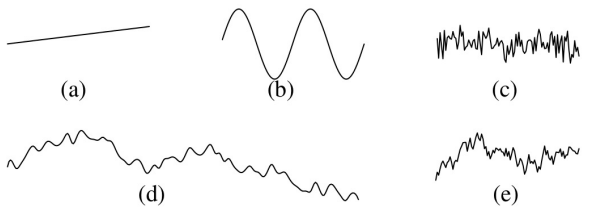


Figure 1.3: Components of synthetic dataset: (a) Line $at + b$, (b) sine ($\sin(2\pi t/d)$), (c) Gaussian noise ($\mathcal{N}(0, 0.1)$), (d) interpolated random walk (trend $R_d(t)$) and (e) a random walk ($R_s(t)$)

as those under day and night conditions, or seasonal changes. This signal can be expressed as $s = at + b + \sin(2\pi t/d) + \mathcal{N}(0, 0.1)$, where a, b, t and $\mathcal{N}(0, 0.1)$ are as before, and where $\sin(2\pi t/d)$ is the cyclic signal with a period of d samples.

The last synthetic dataset, $Synthetic_{RW}$, is inspired by real-world temperature data that contains daily patterns and slowly varying trends. A shared daily trend $R_d(t)$ for each signal is generated from a random walk. This daily value is then interpolated using a cubic spline to generate d values per day, resulting in a smooth temperature trend. Then, for each signal, we add another random walk $R_s(t)$ to generate individual signals. Finally, we add a cyclic component $\sin(2\pi t/d)$, that is amplitude modulated by yet another random walk $R_c(t)$ in order to reflect the changes in temperature across and during the days. Furthermore, all random walks over the given length are normalized to be in the range of -0.5 to 0.5. This results in signals expressed as $s = 4R_d(t) + R_s(t) + R_c(t)\sin(2\pi t/d) + \mathcal{N}(0, 0.1)$, where all stochastic processes are re-evaluated per signal, apart from the daily trend R_d that simulates a common environment.

Each synthetic dataset is composed of 50 groups of three simulated signals, each containing 100 periods (also the non-periodic signals) made of 288 samples, for a total of 28800 samples. Each group corresponds to a wireless node containing three separate sensors. To each copy, a single type of anomaly (see Section 4.1) was added, to allow us to evaluate the performance of our methods per category. For the *spike* and *noise* categories, the amplitude is randomly chosen between two and five times the standard deviation of the incremental (1-step back) signal differences over a window surrounding the insertion of the anomaly. The offset for a drift anomaly is defined as half the amplitude of a total signal. A *spike* anomaly is a single sample long, while the length of a *noise* or a *constant* anomaly is randomly chosen between 10 samples and 1 period, i.e. 288 samples. The drift anomaly lasts 7 periods, where the first and the last two periods are sinusoidal ramps to and from the offset value, to provide a gradual transition to the anomalous offset.

Next to these synthetic datasets, we use three real-world datasets. The first contains the measurement traces collected from an indoor WSN setup

consisting of 19 TelosB nodes, each equipped with three sensors: temperature, humidity and light. This setup collected data every 5 minutes for 5 months, resulting in 42112 samples per sensor. As there is no ground truth for these measurements readily available, we labeled them using the semi-automated method described at the end of this section. This dataset mainly contains *constant* anomalies, and a few instances of *spikes* and *drift*.

In addition, we consider two real-world datasets from two known deployments of WSN, namely the Sensorscope project [34] and the Intel Berkeley Lab [35]. As for Sensorscope, we use the dataset from the Grand St. Bernard (GSB) WSN deployed in 2007. This deployment consisted of 23 nodes distributed on the Grand St. Bernard pass between Switzerland and Italy, depicted in Figure 1.4. Each node had 9 sensors, and measured environmental parameters such as temperature, humidity, solar radiation, soil moisture, rain and wind. Due to varying energy consumption, the number of samples per node ranges from 19800 to 30000 samples. We should note that, due to expected memory limitations (see Section 1.1), we split this dataset into two datasets of three sensors: *temperature* (including ambient and surface temperature, and humidity sensors) and *humidity* (including humidity, soil moisture and watermark sensors).

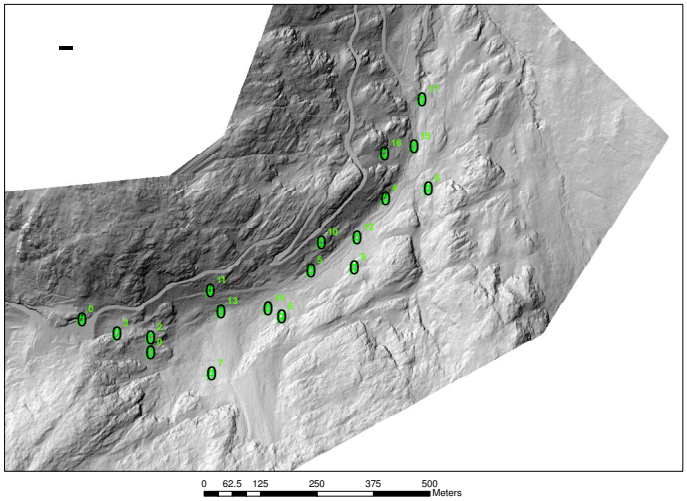


Figure 1.4: The distribution of SensorScope nodes on the Grand St. Bernard pass [34].

The Intel Berkeley Lab deployment consisted of 54 sensor nodes deployed for over a month in 2004 in the Intel Berkeley labs [35], depicted in Figure 1.5. During this time, over 2.3 million readings were collected. However, due to connectivity issues the data for individual nodes are not consistently available. Therefore, we processed the sensor data into measurements with 60 second intervals, where missing samples were imputed by the previously known

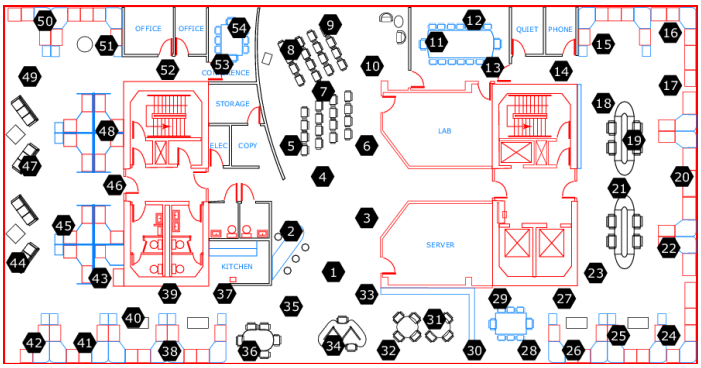


Figure 1.5: The distribution of nodes in the Intel Berkeley Labs [35].

Table 1.1: List of datasets and their properties. The three synthetic datasets are considered as single dataset, with 4 copies for each anomaly category.

Dataset	#samples	#nodes	% anomalous
Synthetic	17.2 M	3*50*4	2.2%
Indoor WSN	0.8 M	19	2.7%
GSB	0.58 M	23	5.1%
Intel Lab	2.30 M	54	19.9%

measurement, and labeled as anomalous. Again due to varying energy consumption, the number of samples per node ranges from 6900 to 50800 samples. Similar to the indoor WSN dataset, both this dataset and the aforementioned GSB dataset contain predominantly *constant* anomalies, and relatively few instances of *spike* and *drift* anomalies. Both these datasets were also labeled using the semi-automated method described below. The main properties (no. of samples, no. of nodes and percentage of anomalous samples) of the six datasets are summarized in Table 1.1.

Such labeled datasets are of key importance to evaluate the results of our unsupervised anomaly detection methods. In contrast with the synthetic datasets, in real-world datasets the labeling is not always exact and largely dependent on expert knowledge, which is often unavailable. Subjective or inconsistent labeling can be improved by using multiple experts, in a procedure similar to ensemble techniques. We can resort to semi-automated techniques when experts are not available for a given dataset. Then, to generate labels we can run several automated anomaly detection techniques that are checked and corrected by hand given the limited expertise of the person correcting the labels.

By manual analysis of the real-world data, we can observe certain behavior that could be labeled by rule-based methods. Such rules can, for instance, label a value that is not changing for a number of samples as a constant anomaly. Other behaviors we observed to be anomalous were values that signify a fault in the measurement, significant shifts in the variation of the signal, large unexpected increases in value, or abnormally large readings. For our experiments, we capture these behaviors in rules that are applied automatically to the real-world data. In a next step, we manually check, correct or add upon these labels. This semi-automated labeling gives us a relatively efficient method to label available data, but still depends on our human judgment.

1.2.3 Evaluation measures

A common way to evaluate detection performance on labeled datasets is through a confusion matrix [36]. This matrix indicates the following four values: the true positive (TP), i.e., the anomalies that are detected; the false positive (FP), i.e. the normal samples that are falsely detected as anomalous; the false negative (FN), i.e., the anomalous samples that are not detected; and, finally, the true negative (TN), i.e. the samples that are normal and also classified as being normal.

In addition to the raw measures, we also use more insightful statistical measures of performance (that are based upon these raw numbers), namely precision, recall, and the F-measure [37, 36]. Precision is the ratio of correct detections over all detections, that is $TP/(TP + FP)$, and indicates the percentage of relevant detections. Recall is the percentage of anomalies actually detected, or $TP/(TP + FN)$. The F-measure is given by $(2 \times \text{precision} \times$

recall)/(precision + recall), indicating a single, weighted average, measure of performance.

Finally, since we use predictors as the basis of our classifiers, we also evaluate the prediction accuracy under normal conditions. We do this by analyzing the root mean square (RMS) prediction error over a sequence of synthetic data without anomalies. We assume that RMS error close to the standard deviation of the noise of the data can indicate better detection performance for a classifier than when it does not resemble the standard deviation.

1.3 Main contributions

The aim of this thesis is to explore decentralized online anomaly detection in resource-limited embedded systems, such as IoT and WSN systems. To provide this, the key challenges of limited resources and of processing large amounts of data have to be addressed. Next to that, the target applications of environmental monitoring often do not have a ground truth of what is "normal". Therefore, only unsupervised methods can be employed, but the evaluation thereof still requires labeled datasets.

In Chapter 2 we present an overview of related work, and how the said work overcomes the key challenges of limited resources and of processing large amounts of data. Anomaly detection, and thus the related work, can be approached from different viewpoints. We distinguish not only the categories of supervised, semi-supervised and unsupervised approaches, but also the separation in offline and online approaches, centralized and decentralized approaches and, of the latter, also distinguish between global and local consensus. While the specific approach depends largely on the application, in the context of networked embedded systems any application can benefit from unsupervised decentralized methods employed as preprocessing or pre-filtering to reduce network resource usage.

The basic building blocks of our proposed framework are presented in Chapter 3. The general approach of the framework is to learn a model of the measured data online, and use this model to predict future measurements. Then, prediction errors are used to improve the models. But, if the deviation is too large, the measurement is flagged anomalous. We show the methodology to implement stable incremental learning methods on resource-limited systems and demonstrate these by implementing several online learning methods that learn linear models, single layer feed-forward neural networks (SLFNs) and segments of time-series data. The evaluation of these methods shows promising results for our proposed framework.

Then, in Chapter 4 these building blocks are combined to fuse predictions or to form ensembles of detectors. We show that such combinations have detection performance benefits, as diverse set of experts can result in a more reliable decision. Moreover, in this chapter we experimentally show that anomalies are

sometimes detected with a delay, and thus discuss possible relaxation of the evaluation metrics to include those delays.

Chapter 5 addresses the distributed/decentralized nature of WSN and IoT applications by including neighborhood information. In this chapter, we first perform a characterization of the scenarios presented in Section 1.2.2. We experimentally chose an aggregation function to provide a summary of neighborhood data, and then show that this neighborhood information does contribute significantly to the detection of anomalies, but only if this neighborhood is well-correlated. This is often the case in the typical application of WSN, that of monitoring environments with diffusive processes.

Finally, in Chapter 6, we summarize our findings, discuss these results and provide an outlook on the future of networked embedded sensor systems.

1.4 List of publications

The work of this thesis has led to the following publications:

H. H. W. J. Bosman, G. Iacca, A. Tejada, H. J. Wörtche, and A. Liotta, "Ensembles of incremental learners to detect anomalies in ad hoc sensor networks," *Ad Hoc Networks*, vol. 35, pp. 14 – 36, 2015, special Issue on Big Data Inspired Data Sensing, Processing and Networking Technologies. Available: http://dx.doi.org/10.1016/j.adhoc.2015.07.013

H. H. W. J. Bosman, G. Iacca, A. Tejada, H. J. Wrtche, and A. Liotta, "Spatial anomaly detection in sensor networks using neighborhood information," *Information Fusion*, vol. 33, pp. 41 – 56, 2017. Available: http://dx.doi.org/10.1016/j.inffus.2016.04.007

H. H. W. J. Bosman, G. Iacca, H. J. Wörtche, and A. Liotta, "Online fusion of incremental learning for wireless sensor networks," in *Data Mining Workshop (ICDMW), 2014 IEEE International Conference on*, 2014, pp. 525–532.

H. H. W. J. Bosman, A. Liotta, G. Iacca, and H. J. Wörtche, "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.

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

A. Tejada, K. Horváth, H. S. Shiromoto, and **H. H. W. J. Bosman**, "Towards waterlab: A test facility for new cyber-physical technologies in water distribution networks," in *Proceedings of the 1st ACM International Workshop on Cyber-Physical Systems for Smart Water Networks*, ser. CySWater'15. New York, NY, USA: ACM, 2015, pp. 9:1–9:4. Available: http://doi.acm.org/10.1145/2738935.2738945

Chapter 2

Related work

The previous chapter introduced resource-limited IoT and WSN systems and the problems arising from the large amounts of data they collect, suggesting the need for a decentralized anomaly detection framework. In this chapter we give an overview of the related work. The first part details the classical approach of systems that send their data to a central processing and storage facility for offline analysis. While this approach is taken by many applications, the ubiquitous and distributed nature of the target systems allows for new approaches that, nevertheless, have to cope with limited resources. The second part of this chapter, starting from Section 2.3, details anomaly detection methods applied to IoT, WSN and related applications. As we will show, while anomaly detection receives increasing interest, there is still a lack of decentralized and online learning approaches.

2.1 Anomaly detection in general

As we have anticipated in the previous chapter, anomaly detection is ubiquitous in a broad range of applications, some of which closely related to the fields of fault and outlier detection. The applications range from fraud detection for credit cards [5], anomalous behavior or behavioral patterns in security card access [26], fault detection [38], such as (sensor) fault detection in safety critical systems [39], intrusion detection in security [40], such as the detection of network intrusions [7], to military applications [41]. Other examples include data center management, where anomaly detection can be adopted to detect problems in system performance and, by diagnosing, to provide potential remedies [27], safeguarding airlines by detecting anomalies in sequences of discrete sets of symbols (in this case switch sensors in aircraft cockpit) [39], and given abundant computing power, complex climate data can be analyzed for anomalous behavior [6].

The detection methods typically make use of a model, the creation of which can be roughly divided into *formal (first principals)* and *data-driven (learning)*

modeling, similar to the fault detection and diagnosis domain [38]. The formal modeling methods use *a priori* knowledge of the physical dynamics of the system under measurement. Commonly, these methods are applied in (controlled) environments of which every detail is known, including the anomalies that can occur. We will briefly discuss such approaches in the remainder of this section. On the other hand, data-driven methods infer a model of the environment from (historical) data. In the typical scenario, WSNs are deployed in *a priori* unknown environments and, thus, require such data-driven methods. We will discuss those more elaborately in Section 2.2.

An example of formal techniques is the Kalman-filter method, which uses previous measurements to fill in properties of a (physical) model, designed using knowledge of the system. These previous measurements can also be used to predict the current measurement and from the deviations of these predictions, diagnosis can be made, navigation errors in aerospace can be detected and corrected, or submarines can be detected with an airborne magnetic field sensor [42].

Another example of how *a priori* knowledge can be used for anomaly detection is a fault tree, which is an organizing structure for modeling "things that can go wrong". These might indicate anomalies such as network intrusions. In [43], such an intrusion (anomaly) detection system formally models potential action sequences that lead to the intrusion, e.g. a deviation of a defined internet protocol. The authors also indicate that a human expert who constructs the fault tree should consider all events that could reasonably lead to an anomaly. The complexity of this task increases with the complexity of the system, as does the fault tree itself. This can be partially alleviated by splitting the tree into different stages of intrusions, but still requires exhaustive definitions thereof [43].

For instance, a formal modeling approach can deliver a diagnosis of faults occurring in cyber-physical systems through formulating a model of the system and the faults that can occur [44]. Then, when the physical system behaves unexpected, a fault is detected, isolated and identified using this model. In general, for such formal approaches it is assumed that all system faults are known *a priori*. Sometimes even stricter assumptions are made, for example that only a single fault occurs at a given time [44].

Commonly, formal modeling methods are applied to a particular application, where it is assumed that the target environment or process can be formally described with the available *a priori* knowledge. This assumption enables the detection methods to be simple enough for on-line usage (e.g. a threshold [45]), and allows the accuracy (for example, of the prediction or the state classification) to be calculated precisely. However, gathering sufficient knowledge at design time to describe the target environment is already complex for controlled settings, such as industrial factories. Gathering sufficient knowledge for IoT or WSN applications may be impossible, as often the deployment environment is not known in advance.

2.2 Data-driven anomaly detection methods

When a system becomes too complex, or there is not enough *a priori* information, a formal model of the system can not be made. This is often the case with IoT and WSN applications. For such cases one can use data-driven modeling techniques. These techniques, closely related to the field of data mining, first infer (learn) a model from a historical sample of data stemming from the system under investigation. After this step, new data can be tested to fit the inferred model and, for instance, classified as normal or anomalous. Data-driven modeling approaches do not require a formal description of the data, by making (e.g. statistical) assumptions about the system under investigation. However, these assumptions must be valid and justifiable for the particular system. Furthermore, when training supervised learning techniques, or when evaluating, labels of the ground truth must be available to ensure correctness of the inferred model.

Most of the data-driven methods find their origins in disciplines such as statistics, machine learning, pattern recognition and computational intelligence. These disciplines can be categorized from different viewpoints. Typically, these categories are *supervised*, *semi-supervised* and *unsupervised*. The supervision refers to an all-knowing oracle, which provides desired outputs (labels) for given training inputs. Supervised learning, thus, requires all training data to be labeled to infer a model. But, these methods also assume that the labels are (mostly) correct, and the training data size is large enough to accurately represent the real world [46]. However, labeled data are hard to acquire (because they need some separate system or human expertise as oracle). Moreover, infrequent occurrence of anomalies results in a class imbalance, whereby the number of normal samples vastly outweighs the number of anomalous samples (for example, in our scenarios less that 10% of the data are anomalous). As result, the evaluation of the learned model may be biased towards the normal class (often favoring precision over recall).

To reduce the need for labels, semi-supervised learning makes use of unlabeled data to either modify or re-prioritize hypotheses learned from labeled data [47]. For instance, with labeled training data a preliminary clustering can be made, of which the parameters are then updated with the unlabeled training data. Unsupervised learners, in contrast, aim to detect structures (or categories) within the data, based on a given similarity metric of the data features, an approach also used in the field of data mining. Commonly used are clustering techniques such as K-Means or K-Nearest Neighbors, and neural network techniques such as a self-organizing map (SOM) [48].

For example, clustering can be applied to group data based on similarity metrics. While common choices are distance metrics such as the Euclidean distance or Manhattan distance, other metrics may be needed to compare, for instance, semantic contexts to detect novelties and drift of concepts [49]. Furthermore, clustering is an often used unsupervised technique to detect outliers

based on the density distribution of the data [50]. However, without labels the evaluation of unsupervised methods resorts to standard validity measures, such as the mean square error of the clusters.

Another viewpoint is related to the availability and quantity of data. When a stored (historical) dataset is available, *offline* (or batch) learning can be employed. However, if such data are not available, or the dataset is too big to be processed entirely, *online* learning can be employed to infer a model from streaming (sequential, often real-time) data, by updating the model as data becomes available. For example, with offline available labeled historical data a hidden Markov model (HMM) can be trained to recognize anomalous patterns in network privilege transitions that signify an intrusion [51]. Another example demonstrates that, when the offline available data are known to be "normal", a marginal kernel density estimate support vector machine (SVM) can be trained to represent these data. Data that do not fit this model, then, are classified as unhealthy [52]. In contrast, the aforementioned SOM can be used online. For example, such an unsupervised learning approach can model the large number of behavior patterns in crowds to detect anomalous behavior of people [48].

In spite of the apparent abundance of processing and storage capacity in centralized systems, the ever increasing amount of data that needs to be processed pose several major challenges. To overcome the large amount of required processing several possible solutions have been advanced by the data mining research community, including parallelization, distribution, and online processing [53]. Parallel data mining partitions and distributes the global data to different computing nodes, such that, when the nodes' results are aggregated, a global result of the mining process is formed. A well-known example is the Map-Reduce algorithm [54], that maps (partitions) the problem onto computing nodes, and reduces (aggregates) the individual results to the global result. Other challenges lie within the communication of the data to a central facility. In the context of WSN, the wireless communication schemes have inherent drawbacks, such as packet loss, while many detection techniques often assume reliable periodic data and, thus, have to deal with delayed packets due to retransmissions [55, 28].

The typical target applications of WSN and IoT systems are often monitoring of environments that are unknown at the design time of a system. Thus, unsupervised methods are the most logical choice for online embedded anomaly detection methods. Furthermore, models learned from previously acquired data may not be suitable at any given time in the future, and thus may require frequent (online) model updates. Depending on the detection method used, these updates may be intrinsic and lightweight, or may require the reprocessing of all the acquired data [56], where the former are more suitable to implementation on WSN nodes due to the resource requirements.

2.3 Anomaly detection in WSNs

The last viewpoint we will address relates to the location of the data processing, being *centralized*, *hybrid* or *decentralized*. In a context of distributed sensor systems, such as WSNs, where the sources of data are physically distributed, these data can be processed locally (distributed, with a central coordinator, or decentralized, without any coordinator) or sent to a centralized system for processing. Interestingly, however, recent surveys [18, 22, 23] report that many anomaly detection techniques for WSNs are centralized. Under this paradigm, data from all sensors are collected at a central storage facility, where they are processed via standard data mining techniques, such as the ones described above. On the other hand, due to the increasing capabilities of IoT and WSN devices, there is an increasing interest in decentralized anomaly detection techniques.

2.3.1 Centralized methods

The abundance of resources often found in central facilities allow for elaborate (pre-) processing. This is often needed for learning of complex models, such as the previously cited methods HMM [51, 57], SVM [52], clustering [39, 49] or statistics [6]. Moreover, it allows complex transforms of multivariate time-series, such as principal component analysis (PCA) or partial least squares (PLS), to create independent variables for which a model can be learned to correlate the target sensor data [58], or to better recognize targets in military image data [41]. Centralized offline analysis can even benefit from human reinforcement as additional detection method, such as in a large oceanic dataset [59]. Furthermore, a diverse set of learning methods, or experts, can be employed to achieve a more reliable decision. That is, by using an ensemble of learners, the classification and regression performance can be improved [60]. There are several methods to integrate the output from a set of classifiers. Common are the following algorithms:

Winner-takes-all where the classifier with the highest confidence determines the output;

Majority-vote where the output class that is most common is chosen;

Boosting where the output is determined by a weighted average of all classifiers. The weights of the classifiers can be based on the measured precision during training, or by using yet another classifier.

For example, an ensemble of five different classifiers can be trained based on predictions from recursive least squares (RLS) trained linear models, neural networks trained by back propagation, neural network trained using an auto regressive method and on an adaptive neuro-fuzzy inference system [61]. Such techniques can be useful when the environment in which to classify anomalies

is too complicated to be modeled by one learner alone, for instance in the case of network intrusion detection [62]

2.3.2 Hybrid methods

Recently, there have been investigations into methods that still require central coordination, but employ in-node processing too. We can distinguish hybrid anomaly detection methods based on an offline-trained model (and online detection) from methods based on hierarchical processing. Using an offline training phase before deployment allows the usage of relatively complex data-driven modeling. A good example is given by Chang et al. [63], who train echo state networks (ESNs) to detect anomalies in time-series. Another example is an offline-trained auto-regressive moving average (ARMA) model which is used to detect anomalies, such that the system compares new measurements with the predictions of the model and classifies them as normal based on a neighborhood voting system [64]. Although the model parameters could be estimated online, offline parameter estimation can ensure that the model represents normal data, and can leave valuable computing cycles to run additional detection and classification techniques on the nodes.

On the other hand, hierarchical processing relies on small amounts of (measurement) data being processed (often merged) on route to a central facility, which makes a final analysis. Often such an approach is taken to reduce communication in the typical WSN application of collecting sensor data [65, 66, 67]. In the context of anomaly detection for example, the authors of [68] propose the use of a distributed, cluster-based anomaly detection algorithm, which locally clusters data at leaf nodes using fixed-width (fixed radius) clusters. However, this method requires the data to be normalized, and will detect anomalies only globally, at the sink. One can also think of another type of hierarchical approach, where resource-limited nodes only provide basic anomaly detection methods to provide early warnings, while more complex detection methods are executed at a base station. For example, this approach is applied in electronic health care, where WSN nodes provide early warnings based on sliding window features (such as thresholds on the mean or standard deviation), while a base station performs complex processing of multiple sensor nodes, such as pattern recognition [69].

2.3.3 Decentralized methods

The need and interest for decentralized data processing (including anomaly detection) are steadily increasing ([70, 71, 72]). Decentralization can take place at the networking level (for instance, see cognitive radio research [73, 74]) or at the application level (for example, probabilistic inference and regression [75, 76, 77], decision making via fuzzy logics [78] or voting techniques [79]). To eliminate the need for central processing and controlling of anomaly de-

tection approaches entirely, a model of what is "normal" should be inferred, and possibly distributed, within the WSN itself. How "normal" data should look like varies depending on the context of the data under analysis and on the experience and subjectivity of the person analyzing it. Generally, normal data are a domain specific concept that requires expert consensus to be defined. It can be defined at least at two levels: normal data in the global context and normal data in the local (neighborhood) context. The following subsections review both levels of anomaly detection techniques tailored to WSN, where the first section reviews global consensus approaches, and the second subsection reviews methods for a local consensus.

Global consensus

Consensus problems are "situations in which all members of some network are required to achieve some common output value using only local interactions and without access to a global coordinator" [80]. In terms of detecting anomalies, this entails a global consensus of what is "normal", such that all measurements outside of this definition are regarded as "anomalous" ones. A simple example is the task to determine the global mean and standard deviation across the WSN, with which nodes can then locally determine the anomalousness of a measurement with respect to this global consensus. However, to converge to a single global consensus, one has to account for the unreliability and limited bandwidth of wireless communications [81].

Consensus techniques can be used in combination with Kalman filters to improve the estimates of global measures [80]. Such methods assume formal models of the sensors are known globally. Furthermore, multiple communication iterations are needed to achieve a consensus usable in the Kalman filter. Nevertheless, even if not all network members provide true readings (either due to sensor faults or intentional anomalous behavior) a consensus can still be reached, given that less than 50% of the nodes are malicious [82]. The authors prove that, if in any iteration of the consensus update neighboring node values are weighted and the extreme values are excluded from consideration, the network can still reach consensus. However, such techniques are not readily applicable to WSNs, due to their excessive communication and computational requirements, in addition to constrains on the possible network topologies.

Extra communication can be used to iteratively build a shared history of measurement data taken by all the nodes, from which a global density function (in the data space) can be estimated. With this, density-based anomaly detection can be performed [83]. By using only the messages from a local neighborhood, this approach can be adapted to perform semi-global anomaly detection. Instead of a shared history, WSN nodes can also share support vectors to train a global consensus for a support vector machine (SVM) which can then be used to categorize data in normal and anomalous classes [84].

In general, energy requirements to reach a network-wide consensus are large

due to their iterative approach. However, the energy usage can be somewhat optimized by, for instance, choosing the appropriate transmission power for a specific network topology [85]. Another drawback of global consensus methods emerges when monitoring large areas, which may show spatial variations in local processes. Therefore, a global model may not be always optimal in a local sense.

Local consensus

Methods for anomaly detection in a local context are the conceptual opposite to the afore-described methods that rely on globally shared models. In data mining, the notion of locality is often given as distance between data values (given a specific distance metric such as Euclidean distance). A data point is compared to the value of its nearest neighbors in terms of data distance [50]. However, the notion of locality can also be given in a geographical distance between the sources of the data. Many similar values (i.e. data with small distance among each other) result in a higher density, called clusters, while values that are less similar result in a lower density. Anomalies can fall outside of any cluster but, when frequently occurring, can form a cluster too. Determining if a datum is normal or anomalous compared to local neighborhood data is a challenge.

A prime example of such techniques is that of the local outlier factor (LOF) [86]. This approach compares the density around a local data point with the density around its k nearest neighbors. For each data point, a minimal radius around its values is determined such that at least k nearest neighbors are included. The ratio between the local radius and the average neighborhood radii then determines the outlier factor of a data point.

The notion of locality can, of course, also be that of geographical space. The spatial local outlier measure (SLOM) [87], is conceptually similar to LOF, but in this case the nearest neighbors are determined in geographical space. The local data are then contrasted to the trimmed mean of the neighboring data, and corrected for the 'stability' of the neighborhood, a parameter similar to variance. These and other statistical properties of local neighborhoods are described in [88], where a generalized statistical approach to obtain local statistics for further analysis (such as outlier detection) is presented.

Schubert et al. survey the above and other related and derived methods [50]. The authors unified the different approaches in a generalized framework, where the notion of locality can be interchanged between data space and geographical space. However, they note that "making spatial outlier detection truly local remains as a possible improvement for a broad range of existing methods." Moreover, most methods target geographic information system (GIS) databases with stationary data, not time-series with evolving WSN data.

Applying these techniques to WSN is not trivial, due to the relatively high

computation and the high communication cost, as surveyed in [89]. Indeed, only few of these spatial anomaly detection techniques have been applied to WSN. For instance, there are LOF-based approaches that, together with a hierarchical network structure, have been used to detect anomalous data [90, 91]. Another, simplified variation of LOF is presented in [92], where the authors use the median of the data, rather than an average value, arguing that, if one wants to determine the center of a sample, the median operator is more robust to extreme (outlying) values than the mean operator. In this approach the detected outliers are used to localize an event boundary. However, one common drawback of all these LOF-based methods is that in order to acquire the k-nearest neighbors one needs multiple processing iterations over the data of the network, or a more efficient hash-based aggregation technique. In both cases, these algorithms might risk exhausting the limited resources of the network very quickly.

2.3.4 Online learning methods

Batch (offline) learning to obtain a model of what is "normal" requires a relatively large amount of memory, a resource that is often scarce in low-power WSN platforms. However, if such methods are employed then they often use only a small window of historic data. For example, one can infer spatial correlation models of neighboring nodes, in terms of mean and standard deviation statistics, by an initial learning stage of 500 samples per neighbor [93]. While this learning stage is expensive, the resulting models are small and easily applied in statistical tests. Another example is piecewise linear regression [94], where the sensor data are represented, and compressed, as a sequence of line segments that are fitted with linear least squares estimation (LLSE). 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 [33]. However, this method does not take into account the correlations between different sensors.

On the other hand, online learning approaches, such as reinforcement learning (RL) and clustering [95], are more often used for WSN applications. These incrementally (iteratively) build a model when new data becomes available and require less memory resources but more processing power. Most online learning is applied in the organization of the network, in particular in routing protocols. These include techniques such as RL [96], Q-learning and swarm-based methods [97, 98]. However, an application of these methods to online anomaly detection is not always immediate. More specifically, RL, that is also successfully used in contexts such as optimal on-off node scheduling [99], requires that the nodes have some active behavior and that they get a reward from the interaction with the environment. However, in monitoring applications, where nodes are typically passive sensing entities, this is not always possible.

As for clustering, well-known examples include k-nearest neighbors (KNN)

and K-Means. While KNN needs to store all the data in memory (going beyond the hardware limitations of WSNs), K-Means only needs to store the cluster centroids (means). Combined with the possibility to incrementally update the model, K-Means is in fact a suitable candidate for online data mining in WSNs [100], with successful applications for instance in network topology adaptation [101]. On the other hand, in order to apply K-Means to anomaly detection, one needs to distinguish between different clusters (normal vs anomalous) and provide suitable similarity measures, which may require an offline preprocessing phase.

To the best of our knowledge, only few (complex) unsupervised online learning methods exist that target the classification of sensed data. An example of that is an ellipsoidal SVM approach, that fits an ellipsoid to data normalized using the median of a sliding window [102]. In the following sections, we overview in detail the work related to the methods used in this thesis, being RLS, extreme learning machine (ELM), and function approximation (FA).

2.4 Recursive least squares

The method of least squares was described by Gauss and Legendre in the early 19th century to estimate the orbit of comets [103]. Since then, it has been used in fields such as finance [104] or machine learning [105] to estimate parameters of models. While non-linear models approximate the parameters in iterations, LLSE models have closed-form solutions. However, for these solutions all data have to be known.

RLS is a variant of LLSE which recursively finds parameters that minimize the least square error. Without the need to store all previous data, but with the cost of higher computational complexity, this method is suitable for online and real-time applications, such as tracking objects with Radar [106], adaptive filtering and control [107], system identification [108], localization and mapping [109] or even in reinforcement learning [110]. However, none of the above methods focus on anomaly detection. There are signal tracking methods, based on decentralized versions of RLS, that do not require central processing to form global parameter estimates. For instance, these can be found in consensus-based RLS [111] and diffusion RLS [112] methods. Such methods achieve global consensus or estimates by first letting each node send local measurements and regressors to its neighbors. In a second step measurements and regressors of neighbors are aggregated to form the new local regressors estimates, that are eventually sent to the neighbors again. Similar to other global consensus methods, this requires heavy communication usage, which makes it inviable in WSNs.

There are some anomaly detection methods that use RLS as learning method. For instance, RLS may be used to determine spatial correlations between distributed sensors of the same modality [113]. But, this method is limited to

detecting spike anomalies. Another example is using RLS based anomaly detection in network systems to detect security threats, by constructing and adapting a dictionary that approximates normal features, with which a kernel feature space is formed [114]. However, this method is implemented in networking hardware which, in general, has considerably more resources at its disposal compared to WSN nodes, making it inviable for resource-limited platforms.

In this thesis we apply RLS to find a possible linear model between local sensor measurement data, predictions from other models, and neighborhood information. With this model, future measurements are predicted and the resulting prediction error is analyzed to detect anomalies. We will address the stability issues that result from fixed-point RLS computation, and evaluate the anomaly detection performance based on the learned model.

2.5 Extreme learning machine

The artificial neural network (ANN), especially Feed-Forward ANN, has been one of the most influential artificial intelligence methods in the past decades, due to the universal function approximation property [115]. Traditionally, ANNs are trained using the back-propagation method, a gradient descent method that learns the weights and biases of the neurons, typically requiring multiple iterations over a training dataset [116]. More complex learning techniques are also capable of adjusting the structure of the neural network, such as the number of neuron layers and the number of hidden neurons in each layer. Unfortunately, these learning procedures are extremely expensive in computational terms, which make them inviable in decentralized WSN applications.

In recent years an efficient learning approach for a specific type of ANN, the SLFN, has been receiving more attention. That is, by randomly setting input weights one can determine the output weights of an SLFN analytically. This approach was named extreme learning machine (ELM), because “it makes learning extremely fast” [117]. Research showed that ELM is efficient for continuous, non-differentiable and piecewise continuous activation functions [118]. Furthermore, it produces good generalization performance in artificial and real-world problems, without user tuning [119]. Over the years, such an approach has shown to be a powerful non-linear mapping method used in classification and regression [120].

Liang et al. have proposed a version of ELM learning that processes the data online and sequentially, called online sequential extreme learning machine (OS-ELM) [121]. Where the original ELM method uses LLSE techniques to learn the output weights, the OS-ELM method uses RLS. OS-ELM permits online updating of the SLFN weights when not all data are available offline. However, both ELM learning methods assume high precision math operations for their algorithms to work, but floating point precision is not commonly

available on lightweight embedded systems.

In the remainder of this thesis we use OS-ELM as a SLFN learning approach to learn a non-linear model between local sensor measurement data, predictions from other models, and neighborhood information. Similar to RLS, the SLFN model prediction error is analyzed to detect anomalies. We will show that the methods to overcome stability issues for RLS also apply to OS-ELM. Furthermore we will address the memory constraints and show the resulting small SLFN can be used for anomaly detection.

2.6 Polynomial function approximation

A polynomial FA method aims to fit a linear or polynomial function to know historical single-dimensional time-series data. Commonly, FA methods are used to model, filter, interpolate or segment time-series data [122, 123, 124, 125]. For example, the resulting segments can be used for data compression [126] or pattern recognition [127]. Earlier we discussed a piecewise linear regression [94] approach that compares fitted line segments to historical segments to detect anomalies [33].

Often, FA methods use a top-down or bottom-up approach. But, an online incremental approach to fit polynomial functions to a sliding window is demonstrated by Fuchs et al. [128], called SwiftSeg. This method uses bases of orthogonal polynomials to incrementally fit a polynomial function to a sliding or growing data window. The authors aim to segment time-series data for clustering and classification of segments, and show the performance is similar to an optimal segmentation technique.

In this thesis we use the SwiftSeg to fit a polynomial function to a sliding window of data. Contrary to the segment-based approach, we use the fitted function to predict (through interpolation) future measurements and detect anomalies from the prediction error. We address the challenges that result from limited-precision math operations, and evaluate the anomaly detection performance.

2.7 Present day approaches

In recent years, anomaly detection has gained more interest in the WSN community. Typically, WSN are used to collect data from the deployment environment and, thus, most anomaly detection happens after the data have been collected at a central facility. The abundant resources allow for offline (batch) learning of complex classifiers, such as HMM, [51, 57], SVM [52, 129] or ANN [61, 130] either individually or in ensembles [61, 62, 130, 131], but also online learning classifiers, such as statistical techniques [27], mixture modeling [132, 133], SOM [48] kernel RLS [114], or ensembles classifiers [58]. Most classifiers learn one or more classes of normal data, such that data outside

of this class is regarded anomalous. Some learn a prediction model, and detect anomalies based on a large prediction error [61, 58]. Moreover, the large storage capacity facilitates data clustering, again either offline [49, 50] (with methods such as LOF [86] or SLOM [87]), or online using methods such as KNN [134]. In these methods, anomalies are identified by analyzing the data density, where dense areas form clusters and anomalies appear outside of these clusters.

There are some hybrid anomaly detection approaches where the model learning is implemented at a central processing facility, which is then distributed to WSN nodes. The detection then takes place online in the nodes. For example, centrally trained ESN [63] or ARMA models [64] can be applied in-node. Moreover, a two-stage approach, with centrally coordinated rules acting on sliding-window statistics for in-node processing can provide a first alert for a centralized system to analyze further [69].

While there are many machine learning approaches applied to WSN *data*, approaches where the anomaly detection takes place entirely in WSN nodes are few. The authors of [56] surveyed machine learning approaches for WSN and show that most of the in-node detection performed can be categorized as rule-based, (Bayesian) statistics based, or clustering (often hierarchical during data collection). For example, the statistical distribution of the measurement differences between a local node and its neighbors can be learned [93, 135], data density can be estimated online in cluster heads [136], or clusters are formed when data are collected [65, 66, 67, 68]. To the best of our knowledge, the only complex classifier aimed at anomaly detection that can be trained online and decentralized is the one-class SVM [137, 138].

2.8 Conclusions

From the surveyed works, and their research outlooks, we can extrapolate some clear challenges for anomaly detection in sensor systems. For one, the evaluation of anomaly detection methods does not yet have a standard. This means that we cannot easily compare one method to another, but always have to look at the results in context (for example of application and problem). Furthermore, the open area of selecting the right preprocessing of features and metrics to compare these features. With the right metric and transformation for the application, anomalies may become more apparent, and thus the detection accuracy increases. Often forgotten is the proper choice of model for a given system, and the usage of correlations existing in that system. For example, we often make assumption of Gaussian noise, while a given system may have a Chi-square distribution of noise. Moreover, making use of the correlations in the system, and changes therein may, again, improve detection accuracy. While we see more and more online and decentralized anomaly detection methods, the adaptation to embedded sensor networks is slow. However, this may have

to do with the other challenges mentioned. Last, but not least, is the usability of the anomaly detection methods. This is an often overlooked challenge by the system designer or researcher, but should be accounted for to increase user-acceptance. For example, giving insight into why a system detects an anomaly may give the user more confidence in the system.

In this thesis we focus on the adaptation and evaluation of online learning methods to resource-limited systems for decentralized anomaly detection. We address the challenge of evaluation by manually annotated datasets, and by demonstrating that a detected sample might need some context, because a prediction-based detector might have a delayed response to anomalous data. The existing anomaly detection methods for WSNs require either a central processing facility, an iterative approach to build a shared model, a hierarchical network topology, or a large buffer of memory for clustering [56]. While prediction-based anomaly detection approaches have been demonstrated for central systems [58, 61], we will demonstrate the feasibility and performance of such an approach in decentralized resource-limited systems through the adaptation of RLS, OS-ELM and polynomial FA learning methods. Moreover, we will show the methodology of adapting incremental learning techniques for resource-limited systems.

A specific single choice for a type of model is avoidable when a diverse set of classifiers, or experts, can be combined in an ensemble setup, to achieve a more reliable decision [60, 61]. We will demonstrate that such combinations also benefit anomaly detection performance in resource-limited systems, by combining our learning methods. Furthermore, it is known that combining neighborhood information can significantly improve the anomaly detection performance [64, 93, 84]. We will show that, at least for our methods, this only holds when spatial neighborhoods are well correlated.

Chapter 3

Online Learning, Prediction and Detection

The previous chapter surveyed anomaly detection methods, ranging from the traditional centralized offline learning, to decentralized methods and to online learning. However, it showed very few decentralized and online learning anomaly detection methods exist for IoT and WSN applications and, to the best of our knowledge, even fewer for online learning anomaly detection. Therefore, in this thesis we focus on the typical use case for WSNs: the monitoring of environments or processes from which there is (presumably) little *a priori* information and for which it is impossible to ascertain beforehand what "normal" or "anomalous" is. Furthermore, we concentrate on those detection methods that can be applied online (i.e., without the need for an offline learning phase), decentralized (without the need for central organization), and that are characterized by a limited computational footprint (to accommodate the stringent hardware limitations found in WSN nodes). Thus, we consider only data-driven anomaly detection methods based on machine learning. More specifically, we consider incremental learning-based modeling, as such methods require fewer resources and do not require central coordination to learn models.

The trade-off is that the methods presented here are better suited for situations where the environment's "normal" state occurs (and is measured) much more often than its anomalous states. In the sequel, we will assume that such conditions hold so that, provided that sufficient data are acquired, a robust learning mechanism can learn (i.e., model) an approximation of the environment's "normal" state. Note that, although this assumption is expected to be met in practice, it makes the evaluation of the anomaly detection performance particularly difficult, since doing so requires either a "skewed" test dataset (i.e., datasets with few anomalies in them) or a test dataset that contains enough samples and anomalous samples to estimate the probabilities of detection (see Section 1.2.3 for details on evaluation metrics). The scenarios introduced in

The work presented in this chapter has been published as [139, 140, 141]

Section 1.2.2 encompass enough data and anomalous data to allow the estimation of detection performance.

In this chapter, we find that online incremental learning using RLS, OS-ELM and polynomial FA is feasible, and present the methodology to implement and stabilize machine learning methods on embedded systems with fixed-point math operations. We evaluate the resulting models by applying them for in-node anomaly detection and show the results are promising, but some improvements can be made.

In the following, we first present the overall approach of the framework of anomaly detectors presented in this thesis (Section 3.1). We detail four specific data-driven incremental learning methods that generate prediction models (Sections 3.2-3.5), and then continue to explain how predictions can be analyzed such that a classification can be made (Section 3.6). Next, we define a number of offline baseline methods in order to compare the results of our online anomaly detection methods (Section 3.7). The behavior of the proposed methods is then analyzed and evaluated in comparison with these baselines (Section 3.8). Finally, we conclude our findings in Section 3.9. The work presented in this chapter has been published as [139, 140, 141].

3.1 Overall approach

The general anomaly detection procedure goes as follows: after data are acquired and an approximate model of the environment has been learned (and adjusted online), the model is used to predict the future state of the environment. A large "deviation" between this prediction and the next measured state signals a potential anomaly (or unmodeled dynamics of the system). The deviation can be stated in several ways. For instance, if the prediction error were to be modeled by a Gaussian random variable ϵ with (unknown) mean μ and standard deviation σ, then one could use statistical tests to ascertain whether or not the measured state of the environment falls outside the 95% confidence interval for its expected value, indicating the presence of an anomaly. Although this general approach to anomaly detection is simple and follows closely the definition of "anomalous" given in the previous section, it has a drawback: its detection accuracy is limited by the quality of the learned model (a "perfect" model of the environment is not available). That is, the detection accuracy depends on the ability of the learning mechanism to build a "good" model.

The approach described above constitutes the building block of the proposed anomaly detection framework and can be depicted as in Figure 3.1. In this structure, we have a number of inputs $\mathbf{x} \triangleq (x_0, x_1, \cdots, x_d)$, which are usually samples from a time series of sensor measurements, where x_0 is the *target* value, that is, the value that we want to classify as normal or anomalous. The remaining inputs $x_1, x_2, \cdots, x_d$ (that we indicate with $\mathbf{x} \setminus x_0$) form the basis on which the *prediction model* creates an estimate of the target $\hat{x}_0$

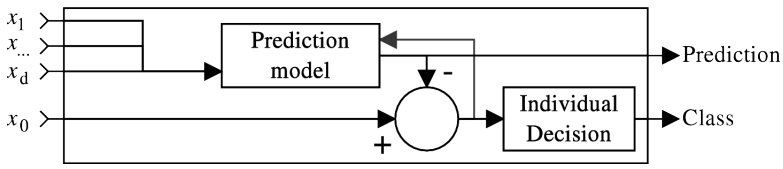


Figure 3.1: The structure of a multi-dimensional classifier. The difference between a prediction, based on inputs $\mathbf{x} \setminus x_0$, and the measurement in x_0 is classified by the decision component.

and can be obtained from, for example, other sensors or historic measurements. The prediction error $\epsilon = x_0 - \hat{x}_0$, is then fed to a *decision making component* that can classify x_0 as normal or anomalous based on learned properties of the prediction error.

In order to construct prediction models of the environment with little or no *a priori* information, we resort to data-based learning methods. Of these, only methods with memory and computational footprints that fit within limited hardware resources are suitable for implementation in embedded systems. Moreover, a WSN node's processing power is limited not only by the speed of its MCU, but also by its capabilities. A common WSN platform such as the TelosB[1] does not have floating-point units. The learning methods that can be adapted to fit these requirements are mostly incremental learning methods, where at each iteration the model is updated to converge to the least squares optimum. While incremental learning methods may incur more computational costs compared to learning from the whole time-series at once (the latter also requiring its storage), the computational and memory cost per new measurement are, in general, low. That, combined with the real-time processing of data, make incremental learning methods ideal for this application.

When a suitable incremental learning method has been found, possible issues related to these limited resources have to be identified. Common limitations are memory consumption and integer or fixed-point computations. For instance, the former limits the number of input dimensions d that can be used, or the number of historical measurement values that can be stored. The fixed-point computations suffer from rounding errors, underflows and overflows, which can make the learning method unstable. Therefore, these problems have to be identified and countered by, for instance, a reset when an overflow is detected, or a variable correction to prevent divisions by zero. For the methods presented below, we have used two open-source fixed-point calculation libraries, `libfixmath` [142] and `libfixmatrix` [143], which allow for 16-bit precision both for the integer and fractional part (denoted as Q16.16). In the next sections, we present the implementation details of several incremental

[1]TelosB is an 16-bit MSP430-based platform with 10 KB ROM, 48 KB RAM and peripherals such as wireless 2.4 GHz radio (TI CC2420), temperature, humidity (SHT11) and light sensors (Hamamatsu S1087 series).

learning algorithms, addressing in particular the effects of the limited precision on the learning process and how those effects can be overcome.

3.2 Sliding window mean

The simplest prediction method is based on plain statistics. In order to generate these statistics, we use a sliding window approach that keeps in memory a short window of L_h data samples and use standard methods to determine the mean μ and standard deviation σ of such data. The estimate of μ can then be used as the prediction for x_0. However, note that due to the risk of overflow in the limited-precision environment, all summands in the sums required to estimate μ and σ are divided by L_h *while* the sums are performed instead of dividing the results *after* the sums are completed. That is, $\hat{x_0} = \sum_{i=0}^{L_h} x_{0,-i}/L_h$, where $x_{0,-i}$ is the i-th historic measurement of x_0.

Additionally, we propose a rule-based decision over the standard deviation σ. That is, we expect the system under measurement to display slow changes, and to display noise in the sensor measurements. This means that a series of consecutive measurements should not have $\sigma \leq \delta$ (where $\delta \to 0$, e.g. $\delta = 1/2^{16}$, the smallest number representable in Q16.16); otherwise, this rule detects a specific type of anomaly, namely a *constant* anomaly (see Sec. 1.2.1 for further details).

3.3 Recursive Least Squares

As we showed in the related work, the method of Least Squares is often used to estimate parameters of models. These models are in the form of

$$y = \boldsymbol{\beta}\mathbf{x}, \tag{3.1}$$

also depicted in Figure 3.2, where the inputs $\mathbf{x}$ can, for example, represent measurements across different sensors taken at one or more instances in time. These are then linearly mapped to the output y, by the weighting of $\boldsymbol{\beta}$. Such models have a closed-form solution, which can be found using the Moore-Penrose generalized inverse of the matrix $\mathbf{x}$, i.e. $\mathbf{x}^\dagger = \left(\mathbf{x}^T\mathbf{x}\right)^{-1}\mathbf{x}^T$ [144]:

$$\hat{\boldsymbol{\beta}} = \mathbf{x}^\dagger\mathbf{y}$$

However, for these solutions all data have to be known and have to fit in memory.

The RLS method, explained in detail in [145, 146], is a recursive method that in each time step t estimates parameters $\boldsymbol{\beta}$ for each data input $\mathbf{x}$. In order to estimate these parameters, only the model and several state variables have to be stored, opposed to storing all previous data. As illustrated in Algorithm 1, the method starts by initializing some parameters (the forgetting factor α

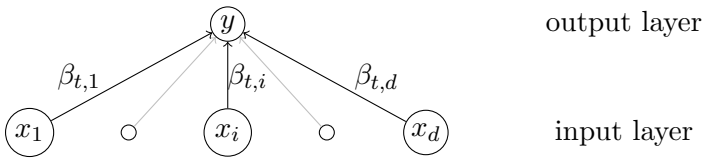


Figure 3.2: Structure of Recursive Least Squares model. The inputs $\mathbf{x}$ are weighted by $\boldsymbol{\beta}$ to form the output y

and the value δ needed to initialize the inverse auto-correlation matrix $\mathbf{P}$) and state variables (the weights $\boldsymbol{\beta}$ and matrix $\mathbf{P}$). The inputs are the vector $\mathbf{x}$ and the estimation target y. Then, for each new iteration $t+1$, the size of the prediction error, ϵ, is determined first, based on the prediction of the previous model. Following this, the variables θ and μ express the updated gain. Next, the direction in which to update $\boldsymbol{\beta}$ and $\mathbf{P}$ is determined using variables K and κ. With these, finally, the weights $\boldsymbol{\beta}$ and inverse auto-correlation matrix $\mathbf{P}$ are updated.

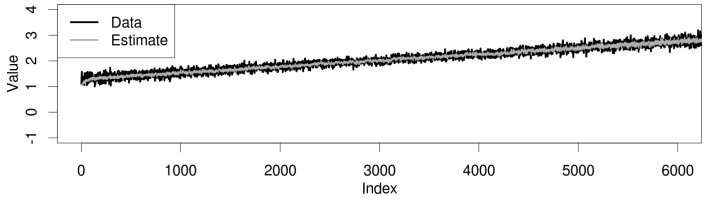
Algorithm 1 Recursive Least Squares

Init $\delta > 1, \alpha > 1, \beta_{0,i} = \{0\}, \mathbf{P}_0 = \delta I$
for each sample step k **do**
  $\epsilon_{t+1} \leftarrow y_{t+1} - \mathbf{x}_{t+1}^{\top} \boldsymbol{\beta}_t$
  $K_{t+1} \leftarrow \mathbf{P}_t \mathbf{x}_{t+1}$
  $\mu_{t+1} \leftarrow \mathbf{x}_{t+1}^{\top} K_{t+1}$
  $\theta_{t+1} \leftarrow \frac{1}{\alpha + \mu_{t+1}}$
  $\kappa_{t+1} \leftarrow \theta_{t+1} K_{t+1}$
  $\boldsymbol{\beta}_{t+1} \leftarrow \boldsymbol{\beta}_t + \kappa_{t+1} \epsilon_{t+1}$
  $\mathbf{P}_{t+1} \leftarrow \frac{1}{\alpha} \left[\mathbf{P}_t - \theta_{t+1} K_{t+1} K_{t+1}^{\top} \right]$

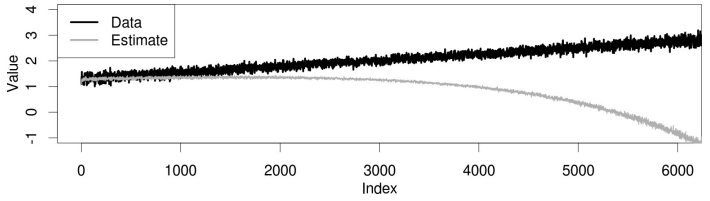
3.3.1 Stability

For stable and accurate estimation, the RLS algorithm requires infinite-precision values. However, embedded systems can only deal with limited precision, fixed-point variables. Therefore, Bottomley [146] has analyzed the implication of quantizing the above computations. This analysis resulted in recursive estimates for the total quantization error, for which Bottomley suggests correction methods.

In our experiments with the fixed point implementation we indeed encountered stability issues, mostly when α is close to 1, as shown in Figure 3.3. The main problem is the underflow in the computations regarding $\mathbf{P}_t$, which can result in coefficients for a certain input x_i to become zero. This has as effect that x_i is not used for correlations in future iterations. Moreover, overflows can occur when the input values are too big, and therefore the inputs have to



(a) Stabilized



(b) Unstabilized

Figure 3.3: Effects of stabilization with $\alpha = 1.1$

be scaled. Based on the work of Bottomley [146], we obtained stable filters by implementing a subset of their suggested correction measures:

- Truncating (instead of rounding) the computational results, to bias the error propagation towards stability.
- Use saturating math operators, such that overflows do not wrap around (e.g. positive to negative).
- Biasing the diagonal values of the auto-correlation matrix $\mathbf{P}$ to counter underflows which, otherwise, might prevent a zero weight to be updated, expressed as

$$\mathrm{C}(\mathbf{P}_t) = \alpha\mathbf{P}_t + \delta\mathbf{I} \tag{3.2}$$

  where $\delta \to 0$. This is contrary to what is suggested in [146], where only non-zero elements are updated. Underflows in $\mathbf{P}_t$ could stop the parameter updates, as can be derived from lines 4 and 9 in Algorithm 1. That is, $\mathbf{P}_t = \mathbf{P}_t + (1/2^{16})I$ (where I is identity matrix) after step 9 in Algorithm 1.
- Ensuring that the input is at most of 16 bit precision which, together with the forgetting factor α and scaling after using the Q16.16 representation of `libfixmath`, prevents overflows.

3.3.2 Complexity

We analyze the complexity of this algorithm per model update step, since we do not have a fixed length of the time-series the algorithms will encounter. That is, every time a measurement is taken, we analyze the complexity of updating the model with this new measurement. In contrast, given a fixed dataset of N samples, the overall complexity, in terms of central processing unit (CPU) instructions, can be given by multiplying by N. The resulting

Method	RLS	LLSE	LLSE/Win
CPU	$O(d^2)$	$O(nd^2)$	$O(L_w d^2)$
Memory	$O(d^2)$	$O(n^2)$	$O(L_w^2)$

Table 3.1: Comparison of order of complexity to RLS per update, derived from algorithms

overall computational complexity is, thus, higher compared to an offline non-sequential approach, where all the data are readily available. Nevertheless, the online sequential learning requires only a fixed amount of memory (that of the model), thus allowing for an online embedded implementation.

The complexity is derived from analysis of the pseudo code and compared to offline LLSE and to applying LLSE on a small window of data of length L_w, the results of which can be seen in Table 3.1. The computational complexity and memory usage of RLS are constant in the order of d^2, i.e. all operations described in lines 3 to 9 of Algorithm 1 are in the order of d^2 times a constant number of operations, while LLSE has to recompute the parameter estimates based on all n previous samples. The latter also means that, over time, memory usage and computational cost increase for LLSE. By using a sliding window for LLSE this cost is kept in the order of a window length, L_w. We have implemented the RLS algorithm for the TelosB platform, a 16-bit MSP430 based platform (compiled with GCC, optimized for size) and ran it in MSPsim [147] to count the real CPU cycles from lines 3 to 9 in Algorithm 1.

From Table 3.2 we see that the algorithm complexity agrees with the theoretical models of Table 3.1, i.e the computational complexity per iteration increases in the order of d^2. The large numbers are mainly due to the Q16.16 fixed-point computations, which are implemented in software. The memory usage increases slightly with the number of input parameters, while program size tends to vary due to compiler optimization.

3.3.3 Energy consumption

The CPU time can be translated to energy usage, expressed in millijoule (mJ), by determining the power consumption of an MCU used in milliwatt (mW) from its datasheet. In the case of the MSP430, found on the TelosB sensor node, this is 14.4 mW per second (s) when the CPU runs at 8 MHz [148]. The formula is then mJ=mW$\times s$. Thus, for 5 ms computation it uses 0.072 mJ. In comparison, the TelosB radio, a TI CC2420, uses 52.2 mW in transmit mode. Thus, for a transmission time of 5 ms it would take 0.261 mJ. As we will see in chapter 5, the targeted WSN deployments consist of many nodes that can not only communicate to a sink node, but also with other nodes in the neighborhood. But, when neighboring data are also used as input for the algorithm, a balance has to be found between the communication rate and

d	2	3	4
CPU Cycles/iteration	21k	42k	74k
CPU Time/iteration (ms)	5	10	18
Max Stack/iteration (bytes)	106	118	118
Total program ROM	19788	19914	19878
Total program RAM	2090	2178	2298

Table 3.2: Number of input parameters vs resource usage of RLS, implemented in TinyOS. Compiled for TelosB platform with msp430-GCC and determined in MSPsim.

the estimation error. Nevertheless, this shows that it is worthwhile to detect anomalies locally, and only send anomalous data to a central processing point for further analysis.

3.4 Extreme Learning Machine

The execution of small trained ANN models, such as ESN, has been shown to be implementable on WSN nodes [63]. The simplest of such ANN models is an SLFN, which has the following structure (see Figure 3.4):

$$y = f_{\tilde{N}}(\mathbf{x}) = \sum_{i=1}^{\tilde{N}} \beta_i \mathrm{G}(\mathbf{a}_i, b_i, \mathbf{x}), \quad \mathbf{x}, \mathbf{a}_i \in \mathbb{R}^d, \quad b_i \in \mathbb{R}, \tag{3.3}$$

where $\mathbf{a}_i$ are the input neuron weights, b_i the bias of the ith hidden neuron, $\mathrm{G}(\mathbf{a}_i, b_i, \mathbf{x})$ is the output of that hidden neuron according to activation function $G(\cdot)$, and $\tilde{N}$ is the number of hidden neurons. Also in this case, $\mathbf{x} \in \mathbb{R}^d$ is a vector of input values of dimension d, while β_i is the weight connecting the ith hidden neuron to the output node. One can use several activation functions $\mathrm{g}(x) : \mathbb{R} \to \mathbb{R}$ to determine the output of a hidden node. For example, for sigmoid functions (such as $g(x) = \tanh(x)$) the hidden-node output is determined by:

$$\mathrm{G}(\mathbf{a}_i, b_i, \mathbf{x}) = \mathrm{g}(\mathbf{a}_i \cdot \mathbf{x} + b_i) \tag{3.4}$$

On the other hand, for radial basis functions (e.g. the Gaussian function) the hidden-node output is determined by:

$$\mathrm{G}(\mathbf{a}_i, b_i, \mathbf{x}) = \mathrm{g}(b_i||\mathbf{x} - \mathbf{a}_i||), \quad b_i \in \mathbb{R}^+ \tag{3.5}$$

These complex models with multiple layers of weights are traditionally trained using the back-propagation method [116], which requires all data to be known and updates weights and biases in an iterative process over the training dataset.

Contrary to the back-propagation learning, the main intuition behind the approach called ELM is that, if one sets input weights and biases randomly

and N training samples are considered, the hidden neuron outputs can be aggregated into a matrix $\mathbf{H}$ of $N \times \tilde{N}$, where $H_{j,i} = \mathrm{G}(\mathbf{a}_i, b_i, \mathbf{x}_j)$. We can then rewrite formula 3.3 as:

$$\mathbf{y} = \boldsymbol{\beta}\mathbf{H} \tag{3.6}$$

Huang et. al showed that this is indeed a linear system [117] and can efficiently be solved using the least squares approach.

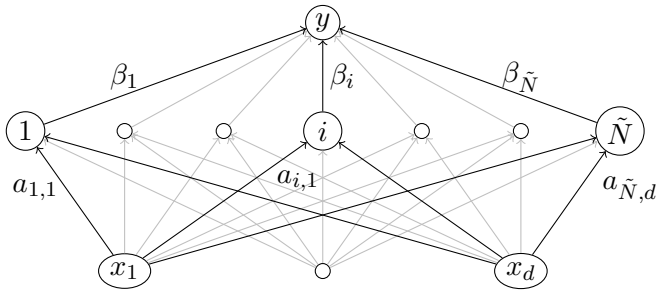


Figure 3.4: Structure of a Single Layer Feed-Forward Neural Network model.

This ELM approach does require that all N training samples are stored in memory. However, as we saw in the previous section, the least squares solution can also be obtained sequentially using RLS. Liang et al. [121] showed that the hidden node weights $\boldsymbol{\beta}_{t+1}$ at iteration $t+1$ can be recursively calculated according to the following formula:

$$\begin{aligned} \mathbf{P}_{t+1} &= \mathbf{P}_t - \mathbf{P}_t\mathbf{H}_{t+1}^T \left(\mathbf{I} + \mathbf{H}_{t+1}\mathbf{P}_t\mathbf{H}_{t+1}^T\right)^{-1} \mathbf{H}_{t+1}\mathbf{P}_t \\ \boldsymbol{\beta}_{t+1} &= \boldsymbol{\beta}_t + \mathbf{P}_{t+1}\mathbf{H}_{t+1}^T \left(\mathbf{Y}_{t+1} - \mathbf{H}_{t+1}\boldsymbol{\beta}_t\right) \end{aligned} \tag{3.7}$$

where $\mathbf{P}_t$ is the inverse auto-correlation matrix between the hidden nodes and the output node.

The above outlined method for learning output weights $\boldsymbol{\beta}$ is called OS-ELM and is described in [121]. While the ELM approach can also be found in random vector functional-link neural networks [149], we have chosen to follow the implementation[2] of ELM and OS-ELM algorithms of [117] and [121]. The OS-ELM pseudo-code, consisting of an initialization and a sequential learning phase, is shown in Algorithm 2. For an extensive description, see the work of Liang et al. [121].

3.4.1 Stability

Analyzing formulas 3.3 to 3.7, and already hinted at in Algorithm 2, we can identify the following issues due to resource constraints. Firstly, as this method

[2] http://www.ntu.edu.sg/home/egbhuang/elm_codes.html

Algorithm 2 Extreme Learning Machine

function InitializeELM($\mathbf{x}_{0,\cdots,\tilde{N}}, y_{0,\cdots,\tilde{N}}$)
  for node $i \in 0, \cdots, \tilde{N}$ **do**
    inputs weights $\mathbf{a}_i \leftarrow$ random number between (-1,1)
    inputs biases $b_i \leftarrow$ random number between (-1,1)
  $\boldsymbol{\beta} \leftarrow$ estimate coefficients from initial data $\mathbf{x}_{0,\cdots,\tilde{N}}, y_{0,\cdots,\tilde{N}}$
    using closed form least squares solution
function LearnSequentialELM($\mathbf{x}_t, y_t$)
  Calculate partial hidden layer output matrix $\mathbf{H}_t$
    with hidden neuron outputs $\mathrm{G}(\mathbf{a}_i, b_i, \mathbf{x}_t)$,
  Determine the desired output y_t,
  Compute output weights $\boldsymbol{\beta}_t$ using RLS, given $\mathbf{H}_t$ and y_t
  Apply stability correction for RLS

uses RLS, the same measures counteract the fixed-point precision issues indicated in Section 3.3 apply to OS-ELM. That is, we apply a correction to the inverse auto-correlation matrix P_t

$$\mathrm{C}(\mathbf{P}_t) = c\mathbf{P}_t + \delta\mathbf{I} \tag{3.8}$$

to prevent underflows on the diagonal, and overflows in the whole matrix. Next, the activation function $G(\cdot)$ may pose limits on the input values due to the fixed-point precision. For example, in the simplest case a linear combination of all inputs may overflow the summation variable. To account for this, the inputs have to be scaled accordingly. Lastly, the size of the hidden matrix $\mathbf{H}$ and the variables needed to update this matrix are limited by the available memory, resulting in a limited hidden layer size $\tilde{N}$ and a limited number of inputs d.

3.4.2 Complexity

For OS-ELM, too, we analyze the method complexity per update step. The analysis of the pseudo-code results in the complexity orders shown in Table 3.3. In this table the RLS complexity returns in terms of the number of hidden nodes $\hat{N}$ from the update, as per equation 3.7. Furthermore, the update of the $\mathbf{H}$ matrix is in the order of the number of inputs d times the number of hidden nodes $\hat{N}$. The quadratic complexity together with the resource limitations of the embedded system do not allow for an SLFN with a large hidden layer.

For this algorithm too, we have used MSPSim [147] to determine values of a real implementation, as seen in Figures 3.5 and 3.6. Our implementation is based on TinyOS, and compiled for the TelosB platform, with compiler flags set to optimize for code size. The plot of Figure 3.5 shows that the number of hidden nodes $\hat{N}$ has the most impact on the CPU time used, as it

Method	OS-ELM	RLS	LLSE
CPU	$O(\tilde{N}d + \tilde{N}^2)$	$O(d^2)$	$O(L_w d^2)$
Memory	$O(\tilde{N}d + \tilde{N}^2)$	$O(d^2)$	$O(L_w^2)$

Table 3.3: Comparison of order of complexity to OS-ELM, derived from algorithms. $\hat{N}$ is the number of hidden nodes, d the number of inputs, and L_w the length of the measurement-history window.

scales exponentially, in agreement with the complexity derived in Table 3.1. Moreover, the fixed-point calculations are implemented in software and thus, contribute significantly to the overall computational time. The effect of the number of input nodes d on the CPU time is, however, not obvious. This may be due to the random weight initialization, or the iterative algorithm used to solve the inverse of a matrix needed in the update of P_{t+1} in equation 3.7.

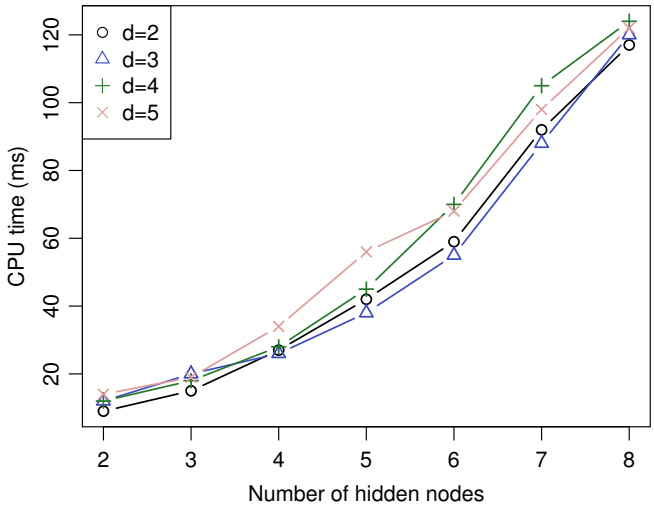


Figure 3.5: The CPU time scales exponentially with the number of nodes $\hat{N}$, while the number of inputs d has little effect. The figure shows CPU time of OS-ELM in a TinyOS program running on an MSP430 (determined with MSPSim).

The memory usage mostly depends on the size of the matrices in memory. Due to implementation details, mainly matrix related [143, 142], this size does not grow proportionally to the number of inputs or to the number of hidden nodes, as seen in Figure 3.6. Instead, in some cases a larger matrix size had to be chosen for the matrix algorithms to function. However, we do see that the number of hidden nodes influences mainly the memory usage, which is largely consistent with Table 3.1. But, we only see a difference in memory usage for different inputs when using 2 or 3 hidden nodes.

3.4.3 Energy consumption

We compute the energy consumption of OS-ELM based on the MSP430 MCU, found on TelosB sensor nodes. Its energy consumption is 14.4 mW per second (s) when the CPU runs at 8 MHz [148]. For example, when we take from Figure 3.5 a CPU-time of 10 millisecond (ms) for 2 hidden nodes, this results in 0.144 mJ. Further, if we take the CPU-time for 8 hidden nodes, being around 120 ms, this would result in around 1.728 mJ. In comparison, the radio chip on the TelosB, a TI CC2420, uses 52.2 mW in transmit mode. Transmitting for 5 ms would take 0.261 mJ, but a message may require multiple nodes to forward the message to reach a central storage/processing facility, depending on the application and factors such as network size.

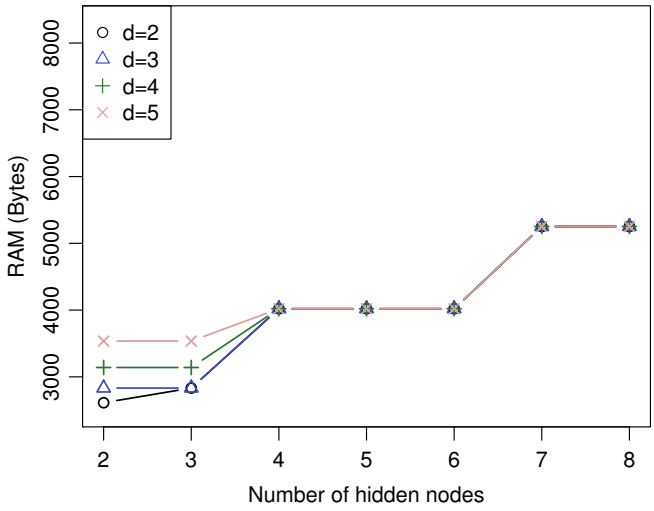


Figure 3.6: The memory usage mostly depends on the size of the matrices in memory, although sometimes larger matrix sizes have to be chosen. The figure shows RAM usage of OS-ELM in a TinyOS program running on an MSP430 (determined with MSPSim).

3.5 Function approximation

The final prediction method considered in our framework is polynomial FA. A polynomial function can be approximately fitted to a sliding window of data and can be used in two cases. In the first case, one can use the fitted polynomial to predict future measurement values. In the second case, one can compare the coefficients of the polynomial fitted to one segment to those from polynomials fitted to other segments, yielding a piece-wise linear anomaly detection method [33]. Here, we have opted for the first case, because it is in

line with the prediction-based anomaly detection approach outlined in Section 3.1 and does not require the storage of historical segments, by implementing a method called SwiftSeg[3] on fixed-point embedded systems, which uses bases of orthogonal polynomials to incrementally fit a polynomial function to a sliding data window [128]. That is, given orthogonal basis $p_k(x)$, the fitted function $p(x) = \sum_{k=0}^{K} a_k / \|p_k\|^2 p_k(x)$. The pseudo-code with fixed-point corrections is shown in Algorithm 3. 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. In the sequel we denote this classifier FA and denote how far ahead it predicts the future as FA(i-step), where i is the number of steps in the future it predicts.

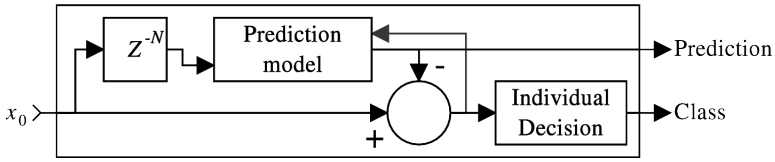


Figure 3.7: Structure of a single-dimensional time-series classifier. The difference between a prediction, based on a model from the signal delayed by N samples, and the current measurement of the signal x_0 is classified.

Algorithm 3 Function Approximation

function INITIALIZEFA(K, N)
    init update coefficients for F_k
        based on max K and window size N
    $\boldsymbol{\alpha} \leftarrow 0$
function UPDATEFACOEFFICIENTS(y_{t+1})
    for degree $k \in 0, \ldots, K$ **do**
        $\alpha_{k,t+1} \leftarrow \alpha_{k,t} + F_k(y_{t+1}, y_t, \alpha_{k-1,t+1}, \ldots, \alpha_{0,t+1})$
        if $\alpha_{k,t+1}$ is overflown **then**
            reset variables and restart with buffered window
function ESTIMATE(t)
    $q \leftarrow 0$
    for degree $k \in K, \ldots, 0$ **do**
        $q \leftarrow combine(q, p_k(t))$
    if q is overflown **then**
        reset variables and restart with buffered window
    return q

[3] http://www.ies-research.de/Software

3.5.1 Stability

The update step recursively calculates the coefficients of each basis polynomial; the estimation step evaluates all the basis polynomials with the current coefficients. In the latter, however, the accumulating variables that are used in this evaluation can run into the limits of the fixed-precision representation and saturate (or overflow). In such case, the model is re-initialized with the previously buffered values. Note that higher degree polynomials require a higher number of accumulating variables and run a higher risk of fixed-precision overflow. Hence, here we will limit our analysis to first degree polynomials.

3.5.2 Complexity

Method	FA	OS-ELM	RLS	LLSE
CPU	$O(deg^2 + L_s)$	$O(\tilde{N}d + \tilde{N}^2)$	$O(d^2)$	$O(L_w d^2)$
Memory	$O((deg + 2)^2 + L_s)$	$O(\tilde{N}d + \tilde{N}^2)$	$O(d^2)$	$O(L_w^2)$

Table 3.4: Comparison of order of complexity to polynomial FA, derived from algorithms. *deg* is the degree of the polynomial used, L_s the length of the sliding window, $\hat{N}$ is the number of hidden nodes, d the number of inputs, and L_w the length of the measurement-history window.

From the pseudo-code in Algorithm 3 we can derive the computational complexity shown in Table 3.4. For instance, the maximum degree of the polynomial of the FA method can be set depending on type of data. But, for simplicity and to be generic, we choose a polynomial degree of 1, i.e. a line. Again, we have used MSPSim [147] to determine the computational complexity of the implementation for a TelosB mote, the results of which are shown in Table 3.5. This table also shows the computational complexity of the sliding window statistics presented in Section 3.2. In both cases, the theoretical complexity scales linear in terms of window length (L_s or L_h) given a constant degree *deg*. The numbers in practice agree with this linear complexity, albeit with a constant overhead.

The memory usage depends largely on the sliding window length L_s, with a constant overhead from the matrices that store the polynomial functions, from the application and from the operating system. Similar to the computational complexity, from Table 3.6 we see that the practical memory usage scales linearly with the window length, in agreement with the theoretical usage.

3.5.3 Energy consumption

For this method, too, the CPU time can be translated into energy consumption. Taking the MSP430, found on the TelosB sensor node, the energy consumption is is 14.4 mW per second when the CPU runs at 8 MHz [148]. For example, if we take a sliding window length $L_s = 20$ and with 3 sensors as seen in Table 3.5,

Table 3.5: Effect of parameter values L_h and L_s on the computation time of an update step of the polynomial FA and sliding window statistics with 3 sensors and first degree polynomial, shown in milliseconds.

L_h \ L_s	**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

Table 3.6: Effect of parameter values L_h and L_s on the memory of an update step of the polynomial FA and sliding window statistics with 3 sensors and first degree polynomial, shown in bytes and including application and operating system overhead.

L_h \ L_s	**12**	**16**	**20**	**24**	**28**	**32**
16	1722	1770	1818	1866	1914	1962
32	1914	1962	2010	2058	2106	2154
48	2106	2154	2202	2250	2298	2346
64	2298	2346	2394	2442	2490	2538

the energy usage ranges from 2.42 to 3.76 mJ. While these numbers are much larger than the 0.261 mJ of energy required for an estimated 5 ms transmission, they include the computational complexity for the sliding window statistics. Moreover, the numbers are only marginally larger than the computational time determined for OS-ELM in Section 3.4.2. 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. Hence, we find this method is suitable for online embedded function approximation.

3.6 Prediction error analysis

As mentioned before, a critical step in anomaly detection is the analysis of the properties of the prediction error ϵ. If a perfect model of the environment was available, its prediction error would be expected to be zero under normal environmental conditions. In practice, however, perfect models are not available, particularly in the case of resource-limited platforms such as found in WSNs: often, the only characteristics that can be estimated are the prediction error's mean μ and standard deviation σ, using statistics over a buffer

of historic prediction errors, or better, using an exponentially weighted moving average (EWMA) [150]. Ideally though, good practical models relying on simple statistics should still be able to yield high-accuracy predictions, thus leading to "almost zero" or "low" prediction errors under normal conditions.

Once the statistics of the prediction error are established (either via analysis or by assuming that it is normally distributed), one can calculate the prediction error's p-value, that is, the probability of obtaining a sample at least as far from the mean as the current sample. Note that the lower the p-value, the more likely the prediction error signals an anomaly. Thus, after acquiring sufficient data to accurately estimate the prediction error's statistical parameters, the decision component (see Figure 3.1) can determine the prediction error's p-value, compare it to a user-defined confidence level (e.g. 95%), and determine whether or not it can be regarded as anomalous. In this way, the end-user sets a fixed confidence threshold, while the p-value is estimated adaptively, depending on the underlying distribution of the prediction error.

In general, for algorithms such as EWMA, it is difficult to set a single learning rate that is suitable for all applications. Thus, in our framework we re-sample and smooth the standard deviation to account for anomalies of longer duration. That is, after each $N_{resample}$ (e.g. 128) new prediction errors, we take the current σ and smooth it using a second EWMA, in order to create a slower moving σ_l, that is more robust to changes. With this slower changing σ_l we determine a second p-value, and we choose the minimum of the p-values resulting from the normal and slow adapting the probability density function. For completeness, we report the pseudo-code of the adaptive threshold detection in Algorithm 4.

This simple, yet effective, adaptive decision-making scheme can be easily applied to the prediction methods described before. Each of those four methods (sliding window mean, RLS, OS-ELM, and function approximation), together with EWMA-based decision making, form a one-class classifier, see respectively the "Prediction model" and "Decision" blocks in Figures 3.1 and 3.7. The added benefit of classifying over prediction errors, is that an anomalous value can be replaced by this prediction.

3.7 Baselines

To compare the above online methods, we contrast them to their offline counterparts, which require all data to be available in memory, and a rule-based heuristic. The latter is relatively simple to implement in a WSN node and, therefore, may show if the computational complexity of the proposed online methods outweighs the simplicity of a rule. However, it does not capture any complex relations between inputs. The rule-based baseline combines a constant classifier, as outlined in section 3.2, with the computation of a p-value over the first-order difference of the sensor signal ($\delta x = x_{0,t} - x_{0,t-1}$). With those, the

Algorithm 4 Adaptive Threshold Detection

initialize μ_t, $\sigma_{t,s}$, $\sigma_{t,l}$, F_s, F_l, λ_s and λ_l.
function AnalyzePredictionError(ϵ_t)
    $\mu_t \leftarrow$ EWMA_update(μ_{t-1}, ϵ_t)
    if initializing or no anomaly **then**
        $\sigma_{t,s} \leftarrow$ EWMA_update($\sigma_{t-1,s}, (\epsilon_t - \mu_{t-1,s})$)
    if t mod $N_{resample} == 0$ **then**
        if initializing or no anomaly **then**
            $\sigma_{t,l} \leftarrow$ EWMA_update($\sigma_{t-1,l}, \sigma_{t,s}$)
    if initializing **then**
        determine multipliers F_s such that
            all past ϵ are within 95% of $\mathcal{N}(\mu_t, F_s\sigma_{t,s})$
        determine multipliers F_l such that
            all past ϵ are within 95% of $\mathcal{N}(\mu_t, F_l\sigma_{t,l})$
        return 1.0
    else
        $p_s \leftarrow$ p-value of ϵ_t given $\mathcal{N}(\mu_{t-1}, F_s\sigma_{t-1,s})$
        $p_l \leftarrow$ p-value of ϵ_t given $\mathcal{N}(\mu_{t-1}, F_l\sigma_{t-1,l})$
        return $\min(p_s, p_l)$

true mean μ and variance σ of δx can be computed, because the offline methods have all data in memory, and with that the p-value. We assume that spike anomalies will significantly exceed the average δx, resulting in a low p-value.

Furthermore, we use the closed form (i.e. non-incremental) counterparts of RLS and OS-ELM, being LLSE and ELM in the same structure as displayed in Figure 3.1, to learn linear and SLFN models of the relation between the target sensor and the remaining sensors in the system. Since these are learned from the dataset as a whole, we assume the resulting models are optimal for the given datasets and, thus, form a sufficiently accurate base for comparison.

The prediction errors are then analyzed, similarly to the methods above, by computing the p-value. However, since the prediction errors are known over the whole time series, again the true mean μ and variance σ of the prediction error can be computed, instead of using EWMA. Both learning techniques, together with the offline prediction error analysis, and the heuristic rules are used as baselines.

3.8 Evaluation

3.8.1 Parameter guidelines

To determine the parameters in the algorithms we analyze their RMS prediction error and their anomaly detection performance. This analysis is done largely on the synthetic dataset, introduced in Section 1.2.1 as its properties

and anomalies are known. Please note that, due to new insights as the research progressed, the datasets were scaled differently to account for stability issues resulting from fixed-point operations and their limited precision, such as overflows. Therefore, results may also have small variations (e.g. values close to zero become zero after downscaling). For example, the hidden node activation function of OS-ELM requires a smaller input range than foreseen when initially investigating RLS. A property shared in all datasets is the number of measurement time-series, i.e. the number of input dimensions, which is $d = 3$. For the RMS prediction analysis we take the sections from the datasets without anomalies and, because the algorithms need some time to converge/learn, we have taken the second half of those sections. While the convergence in the first half is fast, the initial prediction errors may be large, which does not reflect later RMS prediction accuracy.

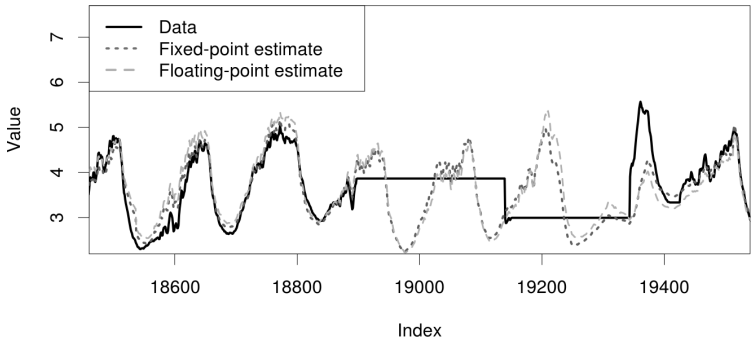


Figure 3.8: Example data with a constant anomaly, and its fixed-point and floating-point RLS estimates, using a high forgetting factor $\alpha = 15$.

The RLS algorithm has one parameter that influences its learning speed, which is the *forgetting factor* α. We consider a reasonable range for α to be from 1 to around 32, where the latter is experimentally determined by the formula in line 6 of Algorithm 1 and the bit size of the fixed-point fraction. The larger this factor, the slower the algorithm converges to a least squares solution, but the more it resists to short-term influences such as anomalies, an example is shown in Figure 3.8. For the OS-ELM learning algorithm there are two parameters of influence: the correction factor c and the number of hidden nodes $\tilde{N}$. The correction factor should be close to, but less than, one. Thus, for stabilization we have chosen to evaluate c over 0.999985, 0.999878, 0.999023, and 0.992188, which are factors of power of 2 near a decimal, e.g. $1 - 1/2^{16} = 0.999985 \approx 0.99999$.

First, we estimate the number of hidden nodes for the SLFN to be learned by OS-ELM. In Figure 3.9 the different choices of c for the OS-ELM method are contrasted against the offline linear least squares fitting LLSE, and an instantiation of RLS. This figure shows the RMS prediction error of OS-ELM can converge to the prediction error of LLSE on the same datasets. However,

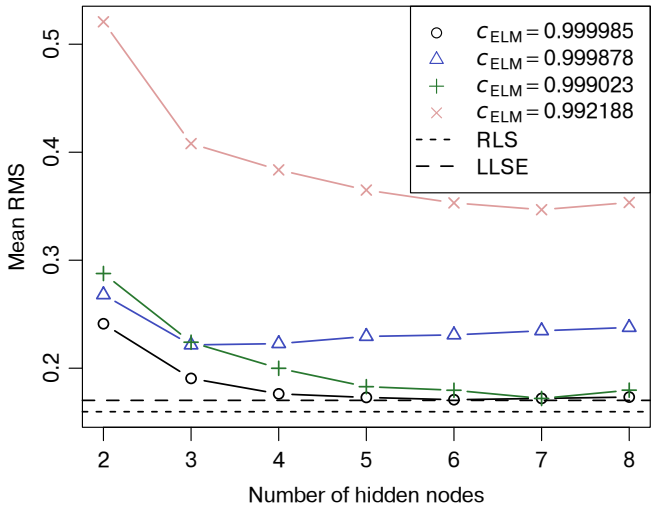


Figure 3.9: The average RMS error of OS-ELM for different numbers of hidden nodes, with number of inputs $d = 3$

this is not guaranteed, due to the random initialization of the hidden node weights and bias. For instance, this shows in $c = 0.999878$, where the RMS prediction error does not seem to converge to the LLSE prediction error, and for $c = 0.992188$ we see a larger prediction error over the whole range of hidden nodes. Given that the number of hidden nodes has a large impact on computational complexity, explained in Section 3.4.2, and the performance increase diminishes with larger $\tilde{N}$, a number of hidden nodes $\tilde{N} = 4$ is established as good choice for our datasets. However, other applications, with for instance more inputs, may find a higher number of hidden nodes more suitable. Furthermore, the figure hints at a correction factor $c = 0.999985$, but we will go in more detail on forgetting and correction factors in Sections 3.8.2 and 3.8.3.

Next, we evaluate a common parameter used in all the classifiers described in Section 3.1, the confidence interval. More specifically, each classifier outputs a p-value, that as we have seen indicates the confidence with which a sample is classified as anomalous or not. Indeed, in order to evaluate the anomaly detection performance in terms of TP, FP, FN, and TN (see Section 1.2.3), a binary classification is needed, which can be obtained by assigning a confidence interval to the data considered to be "normal". This assignment will affect the precision and recall measures. To assess what is a "reasonably good" confidence interval, we conduct a preliminary analysis on the synthetic datasets, where we know the ground truth classification. We generate precision-recall curves for the different anomaly detection methods and confidence intervals of 68.27, 75.00, 80.00, 86.64, 90.00, 95.00, 99.00, and 99.73%.

Figure 3.10 plots the precision against the recall percentage for the range of confidence intervals per anomaly detection method, divided by anomaly

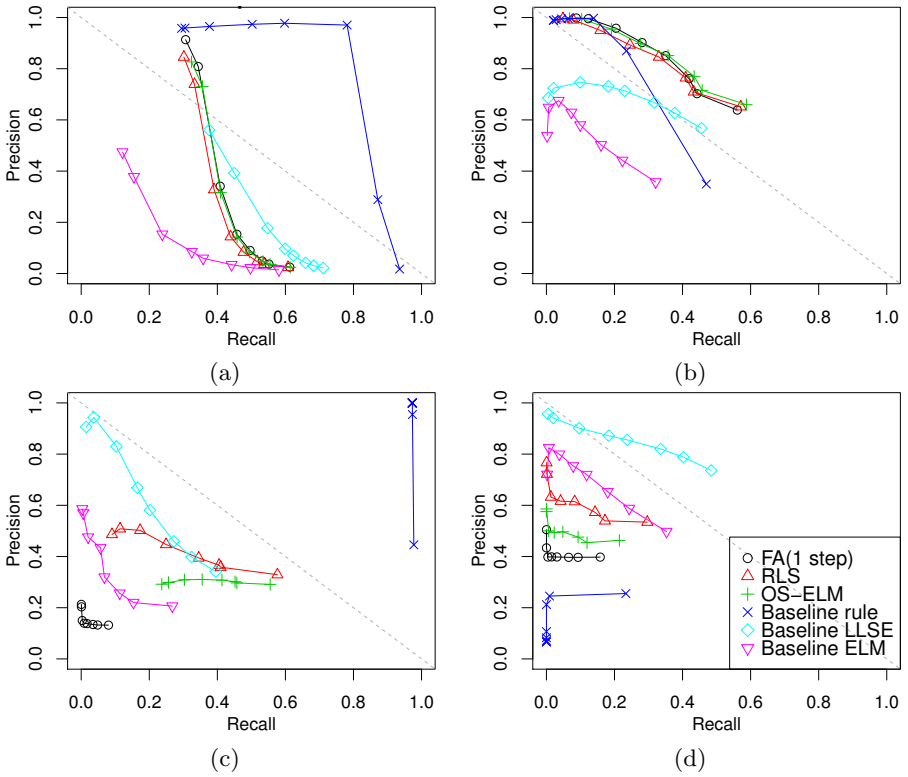


Figure 3.10: Precision-recall curves for several of the individual classification methods. The different points on the curves represent the confidence intervals from 68.27% (more recall) to 99.73% (more precision), for the anomaly types (a) spike, (b) noise, (c) constant and (d) drift. The legend holds for all sub figures.

type. For the 99.73% confidence interval we expect the precision to be the highest, and recall to be lowest, since the confidence of the methods to detect an anomaly should be high. This can be seen as the top left side of the curves. On the other hand, for the 68.27% confidence interval we expect lower precision, but higher recall. This is seen as the bottom right side of the curves. Ideally, the methods have both high precision and high recall, but a reasonable performance should be to the top right of the diagonal.

From Figure 3.10 we can observe that the performance varies greatly with the type of anomaly (described in Section 1.2.1). However, the effect of selecting a specific confidence interval as threshold behaves as expected from the definition of the p-value, being a measure for how likely a sample belongs to the normal class. That is, the higher the confidence interval, the more precise the detection is, but the fewer anomalies are recalled. For example, for the spike anomaly the rule-based method has high precision up to the 80% confidence interval, while the RLS, OS-ELM and FA methods only have high spike detection precision with the 99.73 and 99.00% confidence intervals. For noise anomalies, the precision of RLS, OS-ELM and FA stays relatively high. The constant and drift anomalies are best detected with the baselines.

For most methods the highest confidence intervals result in a very low recall. Therefore, to improve the balance between recall and precision, we establish that a confidence interval of 95% (i.e. p-value < 0.05) gives a reasonable trade-off between precision and recall for our datasets. However, depending on the application goals a different confidence interval may be selected that favors precision (high confidence interval) or recall (low confidence interval). With this interval we can now establish the effect of parameter settings on the anomaly detection performance.

3.8.2 Recursive least squares

Figure 3.11 shows the anomaly detection performance for different types of anomalies and (only for RLS-based algorithms) for different forgetting factors α for the synthetic and Indoor WSN datasets. These results are contrasted against rule-based heuristic baselines, and against offline LLSE prediction-based classifier baselines presented in Section 3.7. The rule-based heuristics are empirically determined and act on the first-order difference of the incoming data. Such an approach is viable in WSN nodes and can show if the complexity of RLS outweighs the simplicity of a rule. However, it does not capture relations between inputs or other information. On the other hand, LLSE is the offline, non-incremental counterpart of RLS, which can capture such relations. However, it requires all data to be stored in memory, which is not viable in WSN nodes. The LLSE/Win model baseline is fitted to a sliding window over the data, which could be possible in WSN nodes but has a very large computational penalty compares to RLS. Furthermore, we contrast the EWMA-based prediction error analysis to a static rule-based error analysis.

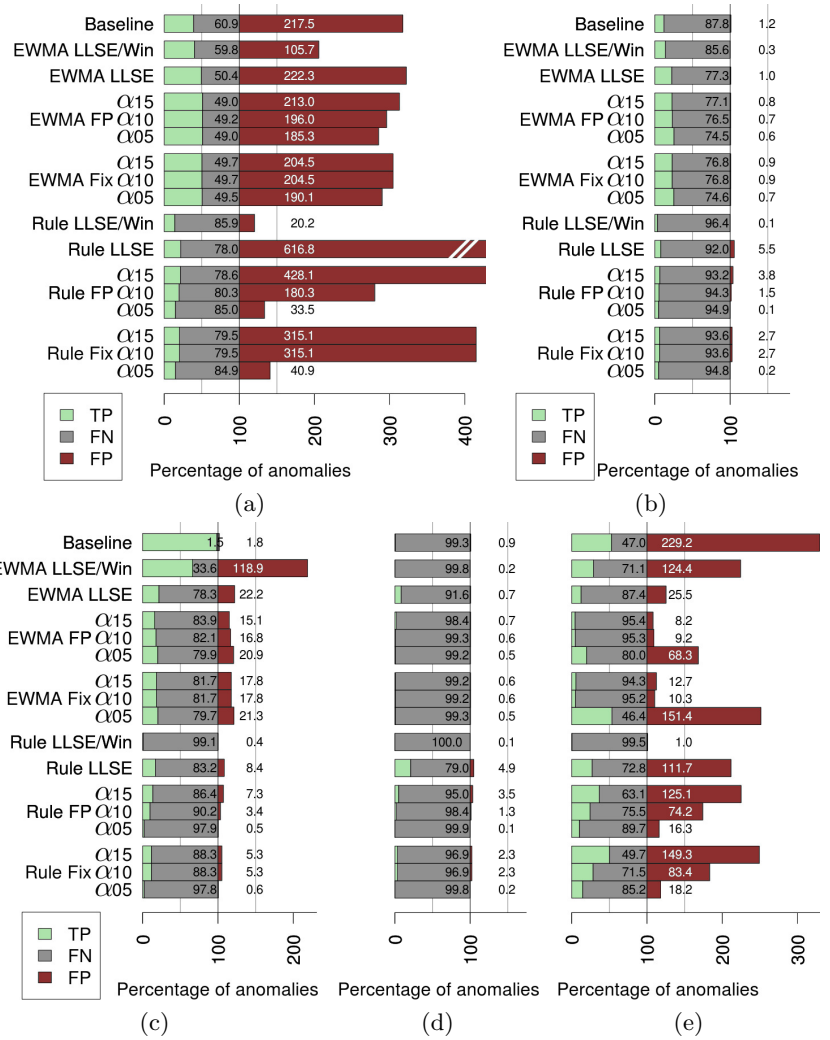


Figure 3.11: The prediction error analysis of *EWMA* is contrasted to a static analysis using a *Rule*. The postfix *Fix* are the result of the embedded fixed point RLS, while Rule/EWMA *FP* values are the result of the floating point implementation. The figure shows TP, FN and FP of RLS as ratio of number of anomalies in synthetic data with (a) spike, (b) noise, (c) constant and (d) drift anomalies, compared to the anomalies in the Indoor WSN dataset (e).

These are denoted as the prefixes *EWMA* and *Rule* to the prediction method names. In the following we first discuss the anomaly detection performance per type of anomaly, after which we establish a guideline for the forgetting factor parameter α.

We show the anomaly detection performance per method for spike anomalies in Figure 3.11a. Each bar represents the ratio of TP, FN and FP detections in relation to the number of anomalies present in the dataset. Thus, the TP and the FN together are 100% of the anomalies present. The FP are shown as red bars to the right of this 100% mark, indicating that those are detections that are non-existing anomalies. The heuristic rule baseline at the top detects 39.1% of the spike anomalies, but also falsely detects 217.5% of the actual anomalies present. The lower bars are split between the EWMA-based prediction analysis (top) and the *Rule*-based prediction analysis (bottom). We see that the offline LLSE/Win baseline is on par with the heuristic rule-baseline in terms of TP detections, given the EWMA prediction analysis. However, it has less FP detections, with only 105.7% of the true anomalies. The LLSE model from the whole dataset detections 49.6% of the anomalies with the EWMA prediction analysis, but has a similar FP ratio to the rule-based baseline. With the rule-based prediction analysis, the TP detection ratio of the LLSE baselines drops to 14.1% for LLSE/Win and 22% for LLSE. Nevertheless, for the sliding-window based LLSE baseline the FP ratio also drops to 20.2%. However, the FP ratio for the global LLSE model with rule-based prediction analysis is far worse, with over 616% false detections.

Furthermore, in this figure we compare the performance of a floating-point RLS implementation (indicated with *FP*) and the embedded fixed-point implementation (indicated with *Fix*), and evaluated for settings of forgetting factor $\alpha = 5$, 10 and 15. The fixed-point implementation performs very similar to the floating-point implementation, which shows in, for example, all TP ratios being around 49% when using the EWMA-based prediction analysis. The influence of the forgetting factor α can mostly be observed from the FP ratios, where a higher α results in more FP for spike anomalies. We observe a small effect of α on the TP ratios, mostly for the rule-based prediction analysis, where a higher α results in more TP.

The performance of RLS for spikes is in general relatively good. Detection is comparable to that of LLSE, with 20 to 50% TP for RLS and 22 to 50% TP for LLSE. The rule-based prediction error analysis performs below the baseline, but is very resource friendly. On the other hand, the EWMA-based prediction error analysis outperforms the baseline method detecting anomalies, without an increase of FP. Noise anomaly detection performance is lower than that of the spikes. Similar to the detection of spikes, the EWMA-based prediction error analysis performs better than the baseline, with a TP rate of around 24% and a lower FP rate of only 0.8%.

When the learning rate is faster than the drift rate, the anomaly is likely to be undetected, because the RLS method adapts to the gradual change. This

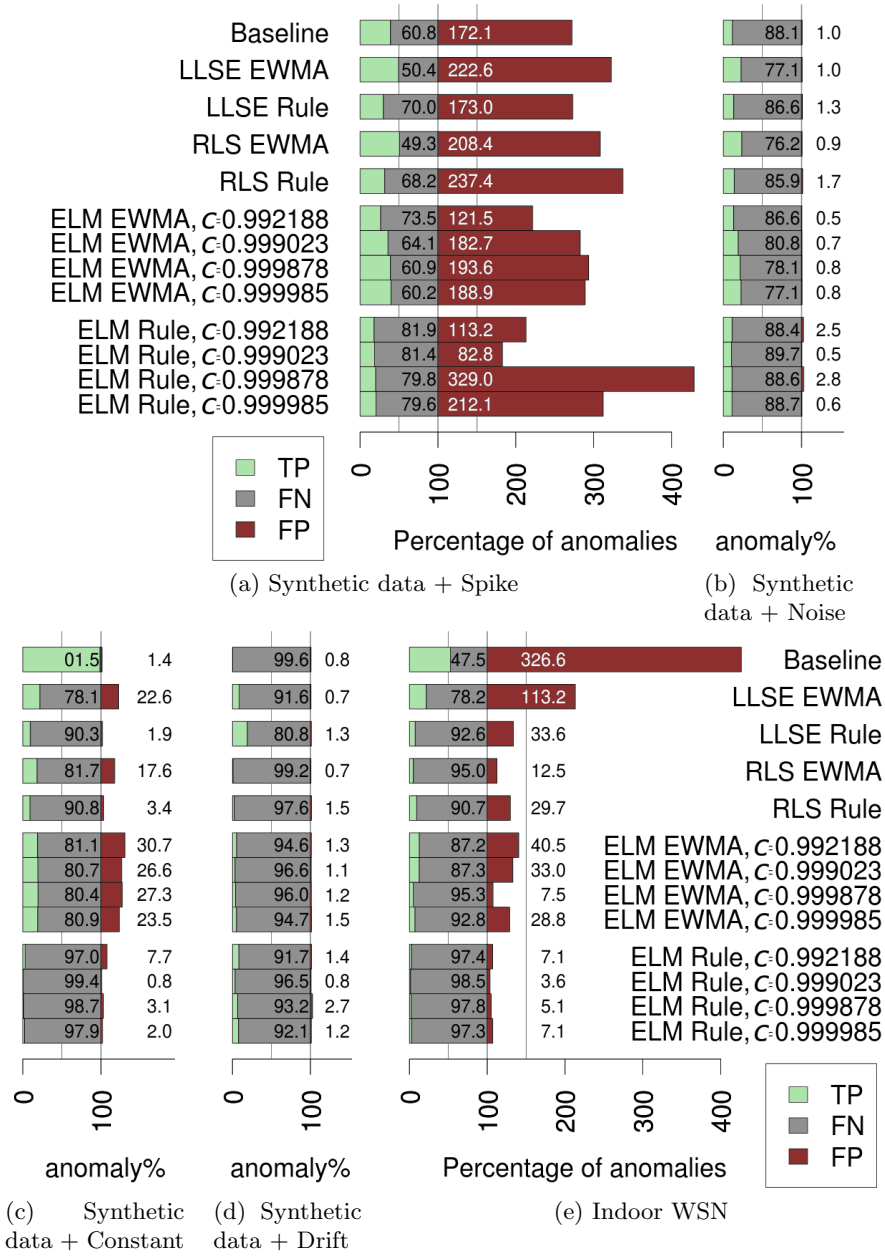
shows in the low TP rates for the RLS based methods in Figure 3.11, where the most detections are made by the rule-based LLSE method, with a TP rate of 21% and only 4.9% FP. Therefore, if any detection is made, it is likely at the start of the drift, with high α (slow learning), while the rest of the drift period is not flagged as anomalous. In general the LLSE method performs better for detecting drift. This is mostly due to the fact that this method is not adaptive but uses the whole dataset, and the fact that the drift anomalies in the generated dataset last for a relatively short period, such that the overall effect on the estimate of LLSE is minor. The FP rate for drift is, however, very low (at most 3.5%), thus the precision is relatively high.

Similarly, constant anomalies are hard to detect with RLS. This shows in the low TP rate, more so for smaller values of α, such as $\alpha = 5$. For such anomalies the low RLS performance is due to the measurement reading, being the last known measurement, which may be relevant for short time periods. When the anomalies persist, and the RLS algorithm adapts slowly (i.e. high α) the constant anomaly may be detected, as is the case in Figure 3.8. The rule-based method, however, performs almost flawless for detecting constant anomalies, resulting in precision and recall of over 98%. However, in real-world applications, such static rules may not apply to unknown and changing environments. Nevertheless, in the following chapters we will combine the static rule to detect constants with the online learning methods presented in this chapter.

In general the TP rate is low, both for the RLS based methods as for the LLSE methods, while the FP rate varies with the type of anomaly. The effect of the forgetting factor α is very noticeable for the rule-based prediction-error analysis, where the constant anomaly detection benefits most. However, in all cases the false positive rate also increases when the forgetting factor increases. Therefore, based on our preliminary experiments, we found that a forgetting factor of $\alpha = 10$ performs well across our datasets. This choice can serve as a guideline for similar environmental monitoring applications, but it is not generic. Specific applications may bias the choice towards lower α values when the mixture of processes in the sensed environment changes quickly, or to higher α when this mixture of processes displays slower change. The EWMA-based prediction error analysis adapts to the changed forgetting factor, which results in a stable performance for the different forgetting factors. It outperforms the rule-based prediction error analysis, and, for spike and noise anomalies, the baseline too.

3.8.3 Online sequential ELM

Earlier we determined that using an SLFN with $\tilde{N} = 4$ hidden nodes should provide a good trade-off between performance and computational complexity for our scenarios with 3 sensors. Now we evaluate the effect of the correction



(a) Synthetic data + Spike

(b) Synthetic data + Noise

(c) Synthetic data + Constant

(d) Synthetic data + Drift

(e) Indoor WSN

Figure 3.12: The anomaly detection performance based on OS-ELM predictions is contrasted to those from RLS predictions and offline baselines. The prediction error analysis of *EWMA* is contrasted to a static analysis using a *Rule*. The figure shows TP, FN and FP of RLS as ratio of number of anomalies in synthetic data with (a) spike, (b) noise, (c) constant and (d) drift anomalies, compared to the anomalies in the Indoor WSN dataset (e).

factor c on the OS-ELM learning anomaly detection performance. Therefore, in Figure 3.12 we compare the OS-ELM results to RLS, LLSE and the same rule-based heuristics baseline as before. For this evaluation, however, the data was scaled differently, due to the requirements for stability posed in Section 3.4.1, which may result in small performance differences. For the RLS stabilization we have used parameters suggested above, in particular the RLS forgetting factor $\alpha = 10$. Again, we first discuss the results per anomaly type, after which we suggest an optimal parameter setting for OS-ELM.

From the results on the synthetic datasets, seen in Figure 3.12a to 3.12d, we observe that the baseline indeed performs well on constant anomalies, with 98.5% TP, and also has reasonable performance for spike anomalies. However, all the regression methods have lower performance on the constant anomalies, around 20% TP if we use the EWMA-based prediction error analysis. This is because the last known measurement may be relevant for a short time period. Moreover, the online learning will slowly adapt to the constant value, and thus the prediction error will go down over time. Nevertheless, we see that LLSE, RLS and OS-ELM are detecting similar amounts of constant anomalies when using the EWMA-based prediction error analysis. On the other hand, this does not hold for the rule-based prediction error analysis, where OS-ELM performs less.

Detecting spike anomalies results in a high FP ratio for all methods, including the baseline method, but we see that LLSE and RLS have higher TP ratios (around 50%) compared to the baseline method (39.2%), when using the EWMA-based prediction error analysis. The OS-ELM with EWMA prediction error analysis, however, does not surpass the baseline method with a little less than 40% TP. Again, the rule-based prediction error analysis performs less well than its EWMA counterpart in all cases.

We see that the other anomalies have much lower FP ratios, commonly in the order of 3%. Noise anomalies, in particular, have FP ratios lower than 2.5%. The TP ratios for noise detection are similar to the TP ratio for the baseline when using the rule-based prediction error analysis with all prediction methods. When using the EWMA-based prediction error analysis, the TP ratio increases to around 22% for all prediction methods. But, we see a decrease in TP ratio when the OS-ELM forgetting factor c decreases, using the EWMA-based prediction error analysis. This decrease is caused by OS-ELM learning more local data with lower c, such that the prediction error increases, and thus the threshold.

Drift anomalies are not detected very well by any method, because we have chosen to model this anomaly with a slow onset and offset, similar to a degrading sensor calibration. When the forgetting rate is faster than the drift rate, this anomaly is likely to go undetected. Otherwise, only the onset or offset of the drift are likely to be detected, before the model adapts through the online learning. This shows in the 20% TP ratio for the LLSE predictions with a rule-based prediction error analysis, both of which do not adapt to the

drift. Moreover, the FP ratio for all methods is very low with less than 3% overall, and in most cases less than 1.5%.

The online learning methods do adapt to drift, but, in this case OS-ELM combined with EWMA-based prediction error analysis has a higher TP rate (around 5%) compared to RLS (less than 1%), regardless of correction factor c. This may indicate that the chosen correction factors for OS-ELM are slower than the drift rate, whereas an earlier study showed a clear effect of the forgetting factor for RLS on the detection of drift [139]. For the anomalies that have a longer stable effect, the drift and constant anomalies, OS-ELM displays more robustness to changes in the parameters of the correction method than RLS, which shows in more stable results.

The real-world dataset contains largely constant anomalies and some spike anomalies. The constant anomalies are mainly due to the unreliability of wireless connections. Furthermore, the spike anomalies in the real-world dataset are caused by direct sunlight hitting the temperature sensors. The large amount of constant anomalies shows in around 50% TP ratio for the baseline, but also results in a high FP ratio. From the other methods, the prediction model based on LLSE has the best performance of the learned models, with a 21.8% TP rate, using the EWMA-based prediction error analysis. RLS performs better with a rule-based prediction error analysis for the real-world dataset, with 9.3% TP, while OS-ELM performs better with the EWMA-based prediction error analysis. With $c = 0.992188$ the TP ratio of OS-ELM is 12.8%, but, with c approaching one the TP ratio goes down to 7.2%. However, the higher TP ration for $c = 0.992188$ also has a somewhat higher FP ratio compared to the rule-based prediction error analysis for RLS.

Overall, the OS-ELM forgetting factor c has a pronounced effect on the detection performance on the synthetic datasets, for instance seen with spike and noise anomalies. However, this effect is not present in the real-world data. Furthermore, this effect is most noticeable when using EWMA-based prediction error analysis. The experiments on our datasets suggest to set c to values close to 1 (e.g. $c = 0.999985$), which is the smallest correction influence on the inverse auto-correlation matrix possible in Q16.16 precision. However, this is specific to our environmental monitoring datasets. Data from different applications may require a reevaluation of this parameter. In particular, larger input ranges may require a larger correction factor due to a larger possibility of instability.

3.8.4 Sliding window statistics and function approximation

Lastly, we show the performance of the sliding window mean and function approximation classifiers, which are influenced by the parameters L_h and L_s. To determine the values of these parameters, we analyze the anomaly detection performance, based on precision and recall. Figure 3.13 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, similar to the other methods presented above. These average rates, however, do establish guidelines on parameter values for all the types of anomalies in the datasets considered in this thesis.

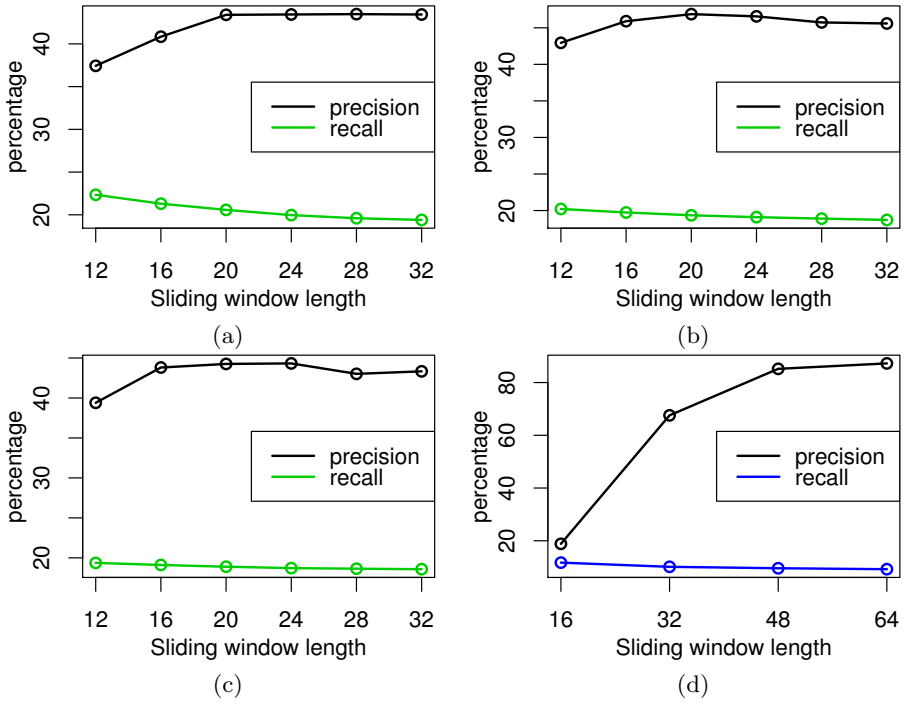
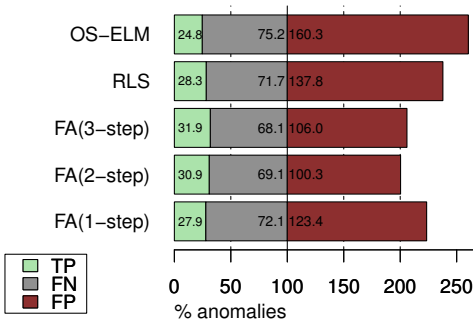
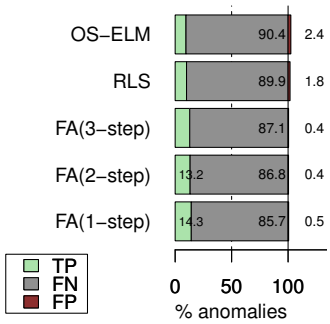


Figure 3.13: Effect of window length on anomaly detection performance of single-dimensional time-series classifiers, evaluated on the synthetic data. The effect of L_s is shown for the (a) 1-step-ahead, (b) 2-step-ahead and (c) 3-step-ahead classifier. The effect of L_h on the windowed statistics classifier is shown in (d).

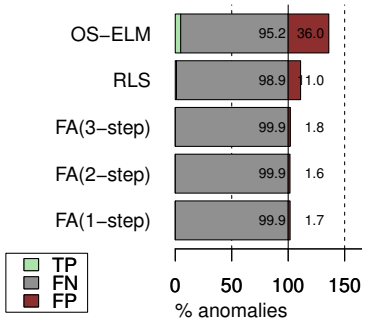
In Figure 3.13 we show the results of precision and recall for the FA and sliding-window statistics anomaly detectors on our datasets. For the FA detector the precision is increasing up to $L_s = 20$, after which it stabilizes. The sliding-window statistics detector shows the precision increases strongly up to $L_h = 32$ and less for larger L_h. For both methods, recall is marginally decreasing for higher window lengths. From the complexity analysis in Section 3.5.2, we see that complexity increases when window sizes increase. Larger window sizes have only small benefit on performance, and large penalty on complexity. Thus, smaller windows should be preferred. Therefore, for the environmental monitoring applications explored in this thesis we keep $L_s = 20$ and $L_h = 32$ as guidelines. However, for application that sense fast changing process dynamics or more noise these window lengths should experimentally be optimized.



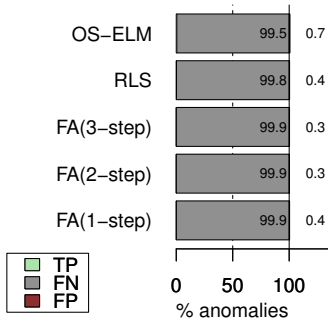
(a) Synthetic data + Spike



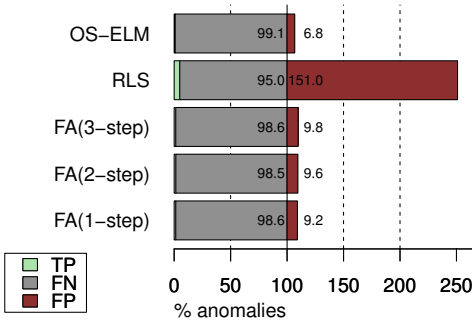
(b) Synthetic data + Noise



(c) Synthetic data + Constant



(d) Synthetic data + Drift



(e) Indoor WSN

Figure 3.14: TP, FN and FP ratios of FA, with 1,2 and 3 steps ahead prediction, in comparison to RLS and OS-ELM.

In Figure 3.14 we contrast the resulting anomaly detection performance of these settings to the earlier presented RLS and OS-ELM methods for the Synthetic and Indoor WSN datasets. This figure shows that the performance of FA is on par with the other methods for the spike and noise anomaly types and for the Indoor WSN dataset, while the detection performance for drift and constant anomalies is lower. The similar performance, but different nature of this anomaly detection classifier (i.e. time-correlation-based vs. input-correlation-based) make this a valuable addition to the proposed framework.

3.9 Conclusions

In this chapter, we have designed and evaluated the building blocks of our approach to embedded online anomaly detection. We have shown that the implementation of incremental learners on resource-limited systems is feasible, and presented the methodology to implement and stabilize machine learning methods on embedded systems with fixed-point math operations. In particular, we have evaluated RLS, OS-ELM and FA online learning methods using the 3 synthetic datasets and the indoor WSN dataset described in Section 1.2.2. These methods train models that predict future measurements, and we have presented a prediction error analysis method that enables the detection of anomalous measurements using the prediction error.

We compared the performance of RLS to LLSE and a heuristic rule-based method as baselines. For the 3 synthetic datasets and the real-world dataset, detecting anomalies with fixed-point RLS is comparable to the LLSE method, while RLS is efficiently implementable in embedded systems. Through evaluation of multiple settings, we determined that a forgetting rate of $\alpha = 10$ for RLS to be suitable for our datasets. While the windowed LLSE baseline seems to perform better in some scenarios, it requires much more resources than RLS, which makes it inviable for WSN nodes. Furthermore, we have shown that our prediction error analysis method shows a benefit over a static rule-based prediction error analysis for RLS by demonstrating a better detection performance.

The performance of OS-ELM trained SLFN was contrasted, again, to LLSE and a heuristic rule-based method as baselines, but also to the RLS method, on the 3 synthetic datasets and the real-world dataset. For these particular scenarios, we have shown that a number of hidden nodes $\tilde{N} = 4$ performs well. However, when the number of inputs is higher, the performance may benefit from a larger number of hidden nodes. Furthermore, for these scenarios we have found that a stability correction factor of $c = 0.999985$ stabilizes the fixed-point OS-ELM implementation, while the influence on the inverse auto-correlation matrix is minimal. Due to the latter, this setting may hold for other scenarios, but may need reevaluating. Across the used datasets RLS slightly outperforms OS-ELM for the detection of spikes and noise anomalies,

but OS-ELM performs comparably on constant anomalies and outperforms RLS for drift anomalies. Overall, OS-ELM performs similar to RLS and it can form a non-linear model with good generalization performance, with minimal user tuning [119]. However, investigating the predictions results for our datasets reveals that OS-ELM seems to have a slight bias, which the rule-based prediction analysis does not account for. This leads us to conclude that our adaptive thresholding approach based on EWMA is a more suitable methods to analyze the prediction error, which shows in its performance and stability thereof to parameter differences.

While the above two methods had multiple input dimensions based on several sensors in parallel, the FA method bases future predictions on the extrapolation of a function fitted to a window over a single time-series. Moreover, the sliding window mean approach bases future predictions on the mean of that window over the time-series. From precision and recall performance on our datasets and the complexity analysis we determined a trade-off between performance and computational complexity as result of window length. For the environmental monitoring datasets used in this thesis we found this trade-off at a window for statistics length of $L_h = 32$ and window for FA length of $L_s = 20$. Nevertheless, for other datasets with different process dynamics, these lengths may need to be optimized. We showed this approach performs on par with RLS, and compliments OS-ELM by performing better on spike and noise anomalies, while OS-ELM is better in detecting drift and constant anomalies.

In summary, we use the following parameters as guidelines for the scenarios used in the rest of this thesis. That is, the methods were tested with environmental monitoring scenarios with 3 sensors ($d = 3$). The confidence level is 95%, the RLS forgetting factor $\alpha = 10$, the OS-ELM correction factor is $c = 0.999985$, and its number of hidden nodes $\tilde{N} = 4$. We suggest the window length of the polynomial FA to be $L_s = 20$, and the length of the window for heuristic detection to be $L_h = 32$. Moreover, our adaptive approach to prediction error analysis using EWMA shows to increase performance significantly. These numbers may serve as guidelines for similar applications, but the exact values have to be reevaluated when using more sensors or datasets with different process dynamics.

To conclude, based on the presented evaluation, the methods 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. The state of the art online learning methods are implemented on centralized systems with abundant resources. In this chapter we have presented how to adapt incremental learning methods to the resource constraints posed in WSN platforms. This was then demonstrated for the learning of linear, polynomial function and SLFN models, resulting in an approach that is decentralized, node-local, and concurrent. However, the analysis of complexity and energy usage does show that these methods take a large toll on the available resources

of a TelosB. This platform is, at the time of writing, more than ten years old and, thus, we expect these methods are better suited for the increasingly powerful networked embedded sensor systems of present day.

While the performance showed comparable results to the baselines for spike and noise anomalies, the drift and constant anomaly detection performed poorly, in terms of recall and precision. The constant anomaly is, nevertheless, easy to detect with a rule. Thus, in the next chapter, among others, we will combine this rule with the learning methods presented here. Moreover, the relatively low recall performance will be addressed in the next chapter. Manual investigation of the prediction output showed delayed detections caused by undetected anomalies being used to predict future outputs.

Another issue is the high false-positive ratio for spikes. The false-positive rate is a ratio of true anomalous samples in the dataset. When there are few true anomalous samples (less than 0.5% for spikes), any false detections will increase this rate significantly. In comparison, the duration of synthetic noise anomalies resulted in more than 2% of the data samples affected, thus a single falsely detected sample has less influence the total false-positive ratio.

Future work may investigate other properties of the above methods that could not be addressed in this thesis. For instance, the use of other activation functions in the SLFNs trained by OS-ELM. Where we have used the sigmoid activation function, one could also use, for example, radial basis, triangular basis, sine, or linear activation functions. These would have different issues with fixed-point operations and may also lead to different results. Next to that, the random initialization of OS-ELM may cause performance differences when rerunning the algorithm. Moreover, we have not investigated inputs from a window of history in the RLS and OS-ELM methods. For example, RLS could be used to train ARMA models. Further investigation into higher-degree polynomial FA may also provide better insight into what degree is best suitable for which process dynamics. For future work we suggest characterizing the target datasets in advance (see Section 5.1.2), which should also result in an appropriate scaling.

In the following chapter, we address the performance issues on the constant anomalies and from the delayed detections. We will also explore the possible combinations and methods of combining anomaly detectors.

Chapter 4

Ensembles, fusion and context

In this chapter we build on the work of the previous chapter, by combining several anomaly detection methods that were presented previously. In particular, we evaluated the usability of these methods, based on online learning through RLS, ELM and FA, for decentralized (in-node) anomaly detection in WSN. These methods have different types of models (linear, non-linear and polynomial functions respectively), which may complement each other. It is well established that an ensemble of diverse learners can improve the reliability and performance of classification [60]. Moreover, multiple sources of information (such as model predictions) can also be fused in a single model to increase its accuracy [152]. In our framework, therefore, we consider two alternative combination schemes, based on *fusion* of predictors and *ensemble* of classifiers, graphically visualized in Figure 4.1 and 4.2, respectively. For *fusion* based combinations, not only sensor measurements, but also the predictions from several other models, are the inputs to another prediction model. The *ensemble* methods combine the output of several classifiers (each employing different prediction models) to increase accuracy.

Furthermore, we find that the combination of individual classifiers, presented in the previous chapter, in an ensemble or fusion approach is feasible in limited-resource systems, and improves anomaly detection performance in terms of recall. Moreover, we address the issue of delayed detection in evaluation by sending a context window around the anomaly.

In the following, first we detail the fusion and ensemble methods. Next, in Section 4.3 we extend our baselines to include such ensemble methods. Section 4.4, then, discusses the effect of anomalies on the framework and its evaluation, and proposes to use a context window around anomalous measurements. In Section 4.5 the methods are evaluated, and the last section concludes the chapter. The work presented in this chapter has been published as [141] and [151].

The work presented in this chapter has been published as [141] and [151]

4.1 Fusion methods

The fusion scheme combines the output of individual predictors (presented in the previous Chapter 3) and the original measurement inputs to form the inputs for a second prediction stage. That is, each individual predictor first generates a prediction $\hat{x}_i$ for each sensor i based on measurements $\mathbf{x}$ without the target x_i. Predictions for all sensors, together with original measurements, then form the inputs to a second stage multi-dimensional input predictor, such as RLS or OS-ELM, which generates new predictions, $\hat{x}_i$, given target sensors i. Graphically, this is depicted in Figure 4.1.

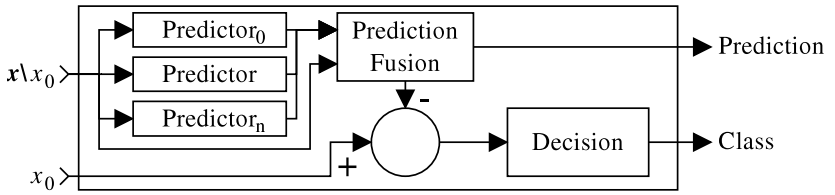


Figure 4.1: Structure of prediction fusion classifier. The predictions of several individual predictors are fused with sensor measurements in another prediction model. A measurement is then classified w.r.t. this prediction using the same decision making component as in Figure 3.1.

In the remainder of this chapter we use the single-dimension time-series predictions from the FA method next to the original measurements as inputs for the fusion methods. These predictors make use of historical measurements, and thus complement the RLS and OS-ELM based predictors, which use only the original measurements, as those shown in the previous chapter. Specifically, the inputs to RLS and OS-ELM consist of the original measurement, the FA(1-step) prediction, and the sliding window mean per sensor. The output of the fusion methods is that of a multi-dimensional classifier, a prediction and a p-value indicating the likelihood of a measurement to be "normal".

4.2 Ensemble methods

Alternatively, the ensemble scheme decides on a classification based on the outputs of multiple individual classifiers. In the simplest case, heuristics can be used. For instance, given the *a priori* knowledge that the rule-based *constant* classifier shows good performance, one can heuristically determine a simple ensemble decision, such that the output of the *constant* classifier is preferred over that from another classifier, say, the RLS-fusion classifier, if a constant anomaly is detected. Note that, although a constant anomaly is detected, the actual p-value may be higher than the selected confidence level, thus the p-value established by the constant classifier is returned. This heuristic ensemble can be expressed with a simple rule, as illustrated in Algorithm 5.

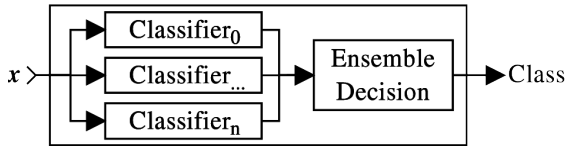


Figure 4.2: Structure of an ensemble of classifiers. The final decision on *class* is based upon the outputs of the different classifiers.

Algorithm 5 Heuristic ensemble

```
if constant anomaly then
    p-value ← Constant classifier p-value
else
    p-value ← RLS-fusion classfier p-value
```


A more generic approach is to assign to each individual predictor's classification result an "outlier" score that reflects the confidence level on the classification result, and then combine the classification results in a meaningful way. This often requires the normalization of the outlier scores and the selection of an appropriate combination function [36]. Since in our approach each classifier outputs a p-value, which indicates the classifier's confidence that a sample belongs to the normal environmental state, there is no need for normalization. Thus, we can combine these p-values using several well-known ensemble combination methods: *mean*, *median*, *minimum*, *maximum*, or *majority voting* p-value selection. While the first four operators apply the respective mathematical operation to the available p-values, the majority voting method classifies an anomaly only when more than half of the ensemble members vote a measurement as anomalous (e.g. 2 out of 3 classifiers should detect an anomaly). We expect that the median and majority voting ensemble produce very similar results according to the median voter theorem [153], which states that a majority vote will produce the preferred outcome of the median voter.

Furthermore, because we are using the p-values as outlier scores, we can directly use Fisher's method [154] to combine p-values by computing the following sum:

$$-2\sum_{i=1}^{k}\ln(p_i) \sim X^2_{2k},$$

where k is the number of classifiers, p_i denotes p-value produced by classifier i, and $\sim X^2_{2k}$ denotes that the sum has a Chi square probability distribution with $2k$ degrees of freedom. Because we are using different learning techniques, underlying the generation of the p-values from the classifiers, we have implicitly assumed that the tests are independent. However, attention should be paid when dependent tests are combined. In turn, Fisher's method yields another p-value, which can be eventually used to classify the input as anomalous or normal.

4.3 Baselines extended

In Section 3.7 we described a heuristic baseline and two baselines that stem from the closed-form (offline) counterparts of RLS and OS-ELM, being LLSE and ELM respectively, as a basis to evaluate the proposed framework. In this chapter, we extend upon those baselines, by including delayed versions of the measurement time-series as the model inputs, denoted with *(d)* in their label. Since the real-world data includes day/night patterns, a logical choice for such a delay is that of one day, or one period, earlier.

Furthermore, to provide a baseline for the ensembles the results of the rule-based, the LLSE and the ELM baselines are combined into a minimum p-value ensemble to give a baseline for the ensemble methods. The minimal p-value can be seen as a logical or, such that if any of the classifiers detects an anomaly, the ensemble flags an anomaly. However, this is at the cost of false detections, which may result in less precision.

4.4 Context for delayed detections

As indicated in Section 1.2.3, in the previous chapter we used the TP, FP, FN, TN, and metrics based thereupon for our evaluation. However, we also saw suboptimal results due to possibly inexact labels (resulting from manual labeling, detailed in Section 1.2.2) and to two particular issues related to the structure of our classifiers (as depicted in Figure 3.1). The first issue is a consequence of using sensor measurements $\mathbf{x} \setminus x_0$ to estimate and classify the target sensor value x_0 (see Section 3). Namely, the prediction error could be classified as anomalous not because there is an anomaly present in the data measured by the sensor x_0, but because there is an anomaly in one or more of the sensors $\mathbf{x} \setminus x_0$ (which in turn leads to a mismatch between x_0 and its predicted value $\hat{x_0}$). The second issue is the presence of delayed anomaly detections. For instance, these can occur when a *spike* anomaly is classified as normal based on a learned model but, after the model is updated with anomalous data, the next prediction is so erroneous that it is classified as anomalous.

To overcome the first issue, we extend the definition of a TP in such a way that a detection is regarded TP when an anomaly occurs in *any* of the input sensors. In other words, we argue that a detection, caused by an anomaly in any of the sensors of a node, but that is not caused by an anomaly in the sensor being classified, is correct and should be viewed as a valid, TP, detection. To overcome the second issue, we assume that if a true anomaly lies within a few samples of the detection, this detection can still be considered valid (since it might prompt the data analyst or an automated second anomaly detection stage to investigate the data around the flagged sample), therefore we extend the definition of TP to include a *context* window, in the form of a window of length L_b around the detection. The effect of this mechanism is thoroughly

evaluated in the following sections.

4.5 Evaluation

For the purpose of evaluating the above explained fusion and ensemble methods, we consider the data flow and methods shown in Figure 4.3. The constant rule, sliding window mean and FA classifiers act on a small window of historic data $x_{win,0}$, while the RLS and OS-ELM classifiers use $\mathbf{x} \setminus x_0$ to predict and classify x_0. The RLS and OS-ELM fusion classifiers take as inputs not only the raw sensor data, but also the predictions generated by sliding window mean and function approximation. The prediction errors are analyzed as described in section 3.6, yielding p-values, that can, in turn, be used to classify x_0 or be used as input for the ensemble combination methods.

The ensemble methods take p-values from the classifiers and combine them into a final decision. The two instantiations of Fisher's method include different combinations of classifiers. That is, "Fisher's method (full)" includes all the individual classifiers in the inputs for the ensembles, and "Fisher's method (part)" includes a partial set of classifiers (including the classifiers based on FA, sliding window statistics and RLS fusion, for more details see [141]). All these combinations are shown in Figure 4.3, where the red arrows show the p-value/classification outputs of each individual classifier and ensemble of classifiers.

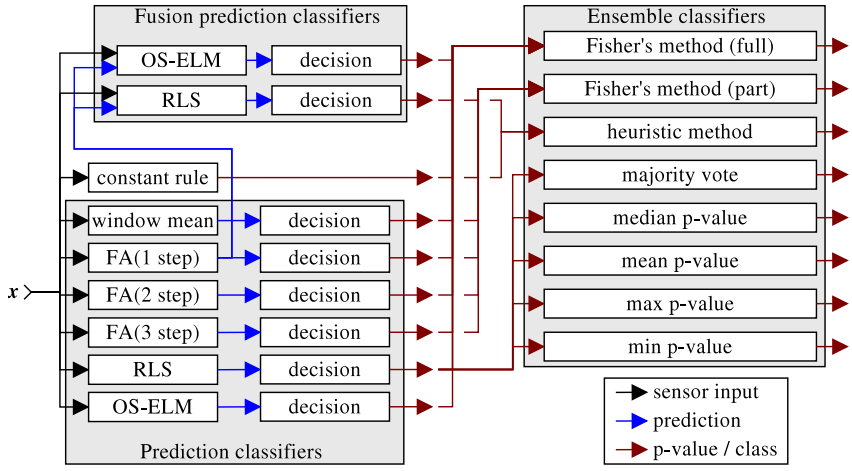


Figure 4.3: The combination of online embedded methods used. Each p-value/class arrow indicates a classification. The ensemble "Fisher's method (part)" refers to a subset of available classifiers that was used in [141].

4.5.1 Confidence interval

To show the effect of the confidence level on the ensemble classifiers (both online and offline baselines), we repeat the analysis shown in Section 3.8 also on the ensembles, including in this case a *context* window around the anomalies of length $L_b = 3$. This means that a detection on one sample prior or one sample after the actual label of the anomaly is regarded TP (recalling the discussion in Section 4.4). Moreover, this is a minimal context of data that is close to the detection in terms of time, resulting in the least extra messages sent. We further discuss the effect of the context window in Section 4.5.3 and the benefit of ensembles in Section 4.5.4. The resulting precision-recall curves can be seen in Figure 4.4. Once again we observe that the detection performance per anomaly type varies, but using ensembles and accounting for context have a positive effect on both recall and precision compared to the individual methods presented in Figure 3.10 of Section 3.8. Most importantly, however, for our scenarios a confidence interval of 95% results in a good balance between precision and recall for further evaluation of the detection performance on these scenarios. That is, over all the different types of anomaly categories, this confidence interval results in a reasonable recall percentage, with reasonable precision. Nevertheless, depending on the application, an end user might accept a different trade-off where, for instance, precision is much preferred over recall.

4.5.2 Prediction errors

The prediction error analysis method, outlined in Section 3.6, assumes errors to be normally distributed, although the parameters of this distribution might change over time. Furthermore, it also assumes that if the distribution has similar properties to the Gaussian distribution (i.e., it is bell-shaped), the method should still perform reasonably. If, however, the prediction error distribution shows a mixture of distribution models or very distinct peaks, which are not due to anomalies, the method might give unreliable results.

We have performed an offline analysis on the prediction error to get a general idea of its distribution. For each combination of method, dataset, node and sensor within that node, we have uniformly sampled the prediction error for that combination and aggregated these prediction-error samples per dataset. We then analyzed the distribution of the resulting aggregation. Due to the large number of combinations, we only show a selection of the most significant results in Figure 4.5. These show that the error distribution does not closely follow a normal distribution, but seems to be very much centered within the fitted normal distribution and, therefore, the p-values are expected to reflect a reasonable outlier score. Furthermore, the distribution parameters vary greatly with dataset and type of sensor, but in all cases the prediction

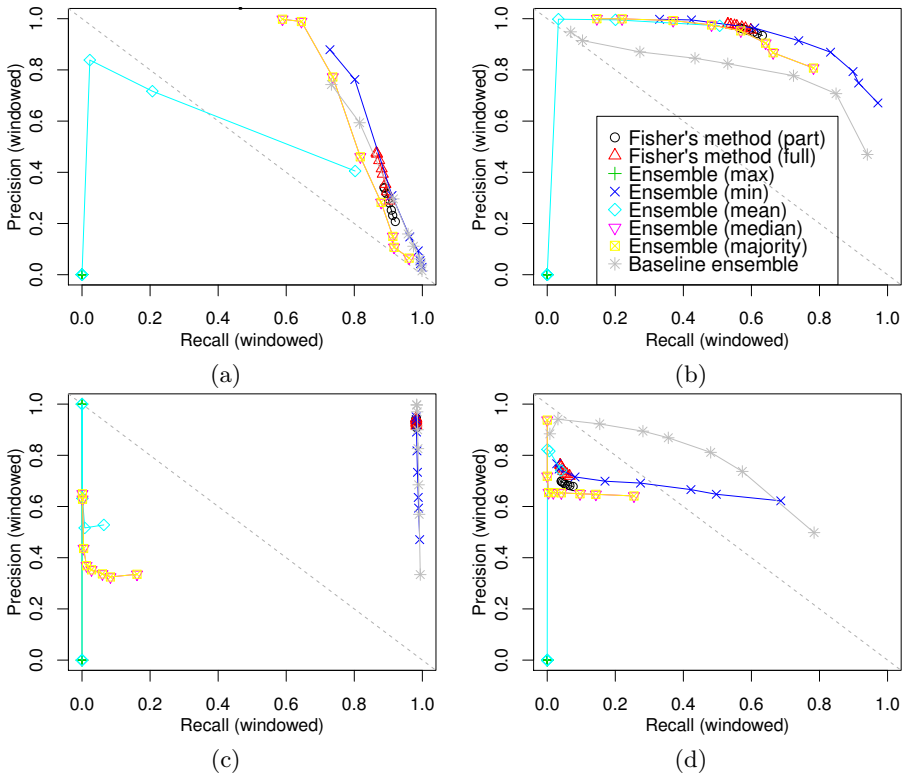


Figure 4.4: Precision-recall curves for the ensemble methods, and sending a window of data around the detection of length $L_b = 3$. The different points on the curves represent the confidence intervals from 68.27% (more recall) to 99.73% (more precision) for the anomaly types (a) spike, (b) noise, (c) constant and (d) drift. The legend holds for all sub figures.

error distribution does display a single peak with a bell-shaped curve.

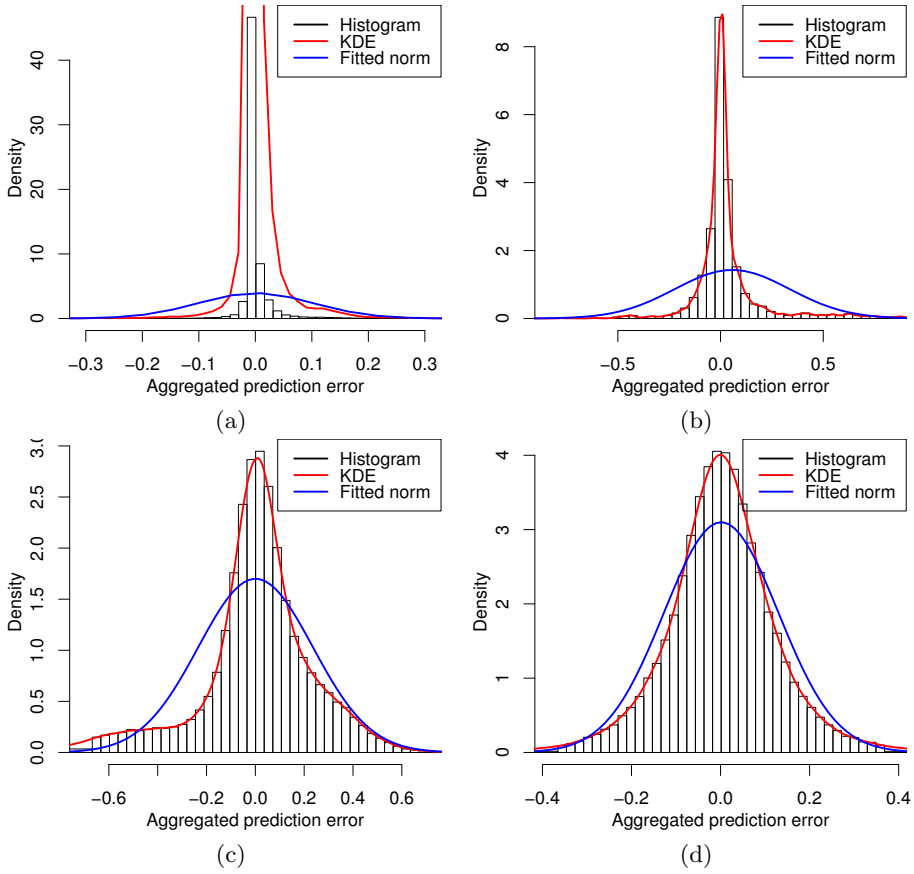


Figure 4.5: Comparison of different prediction error distributions with fitted normal distributions and Kernel Density Estimates (KDE) of the distributions. The prediction error distribution depends on dataset and on the prediction method used. (a) Intel Lab data, sliding window mean, (b) Intel Lab data, RLS-fusion. (c) Synthetic Drift data, sliding window mean, and (d) Synthetic Drift data, RLS-fusion.

As Figure 4.5 shows, the prediction errors generated by applying the detection methods to the synthetic data can exhibit a close-to-normal distribution. This can be expected from the synthesis method of this data, which uses Gaussian noise. Furthermore, because in all synthetic datasets we have added a Gaussian component with zero mean and 0.1 variance, we expect the average mean square error to be no better than 0.1 units. This is shown in Figure 4.6 where, for each method, we have summarized the mean square prediction error per node in a box plot. Moreover, we can see that the performance of the sliding window mean predictor varies the most, which can be expected from the short window over which the mean is estimated. The larger difference in performance of OS-ELM fusion can be attributed to the random initialization

of input node weights, combined with fixed-point precision.

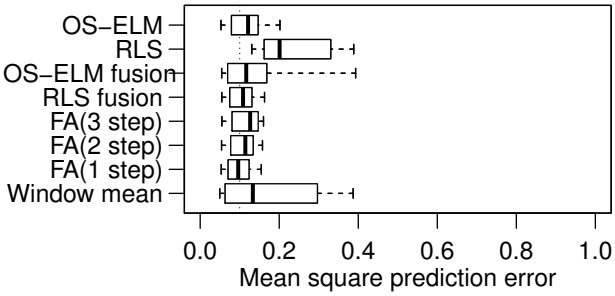


Figure 4.6: Comparison of the mean square prediction error of the different methods on the synthetic data, who has zero mean Gaussian noise with a variance of 0.1.

4.5.3 Effect of a detection context window

We now turn our attention to the effect of context around the detected anomalous measurement (i.e., if a labeled anomaly lies within a *context* window around a classified anomaly, the latter can be regarded a TP). As discussed in Section 4.4, here we assume the minimum context window around a detection, by including the data samples right before and after a sample classified as anomalous ($L_b = 3$).

We have estimated the performance of the anomaly detection methods when applied to the test datasets and compared the results with and without using a context window. These results are shown in Figures 4.7 to 4.10. Figures 4.7 and 4.8 show the results based on the synthetic datasets. When we take a context window into account, we see an almost doubling of the TP ratio for all methods, and for all types of anomalies other than the constant anomaly, suggesting the methods may indeed suffer from a delayed detection. The effect of the context window is less noticeable on the real-world data shown in Figures 4.9 and 4.10. Conceivably, this can mainly be attributed to the fact that most of the anomalies in the real-world data are constant anomalies. The constant classifier therefore has a high ratio of TP, which is reflected in the performance of the ensemble methods. On the other hand, the FP ratio of the constant classifier is much higher for the real-world data than for the synthetic data, which can be attributed to constant natural signals, such as low light intensity signals at night.

The impact of using a context window over the anomaly detection performance is, on the whole, significant. This is consistent with our hypothesis that an anomaly does not only show when it differs from a prediction, but it can also have an effect on the model's prediction after it is updated with undetected anomalous data. Transmitting the context window has the added benefit of allowing one to examine the system's behavior before and after an anomaly is

detected. In addition, one could also apply a post-processing step to identify the actual anomalous sample. Overall, many anomalies are detected within the context window using this online decentralized approach.

4.5.4 Anomaly detection performance on synthetic datasets

Lastly, we evaluate the anomaly detection performance of our methods by analyzing the results according to the metrics described in Section 1.2.3, extended with the insights in Section 4.4. We measure those metrics on both the online and offline (baseline) methods with application of the context window, as described in the previous section. These results are presented in Figures 4.8, 4.10 and Table 4.1 (it is worth noting that the TP ratio shown in the figures is equal to the recall rate presented in the table, whereas the TP divided by the sum of TP and FP ratio can be related to the precision). We first evaluate the performance of the different methods for the synthetic datasets. Then, the methods are evaluated on three real-world datasets in Section 4.5.5.

As remarked in Section 3.8, there is a performance difference in detecting anomalies of different types, as clearly shown in Figure 4.8. For instance, we observe that the constant classifier, as expected from its nature, has very high performance for *constant* anomalies in the synthetic data, with over 98% recall and zero FP, which is equal to 100% precision. This performance is also reflected in some ensemble methods, mainly the heuristic ensemble, those based on Fisher's method and the minimum p-value ensemble. These ensembles, however, do have 1.6% to 12.0% FP due to the inclusion of other individual classifiers. Among the latter, the RLS and OS-ELM classifiers show over 20% TP ratio, but also a high amount of FP. Both effects can be attributed to online learning, which adapts slowly to constant anomalies. The FA classifier shows very few TP. This, we suspect, is due to the window length ($L_s = 20$) over which the function is approximated, which is almost as large as the duration of the average constant anomaly. The RLS and OS-ELM fusion classifiers fuse the FA predictions together with window mean predictions, as shown in the TP and FP ratios. Compared to the baselines, the constant rules perform equally. On the other hand, centralized offline analysis is advantageous for the LLSE and ELM classifiers which, compared to RLS and OS-ELM, have better precision, albeit with lower recall.

The *spike* detection results in Figure 4.8 show that function approximation, RLS and OS-ELM perform similarly, with around 77-83% recall. Increasing the length of the prediction horizon in FA has a small negative effect on the recall ratio, because the further ahead in the future a prediction is, the less relevant it becomes, thus increasing the average prediction error and the detection threshold. All methods have similar precision a little over 50%. Here, again, heuristic, Fisher's and minimum p-value ensemble methods outperform the individual classifiers in terms of true detection, gaining up to 7%. But, in

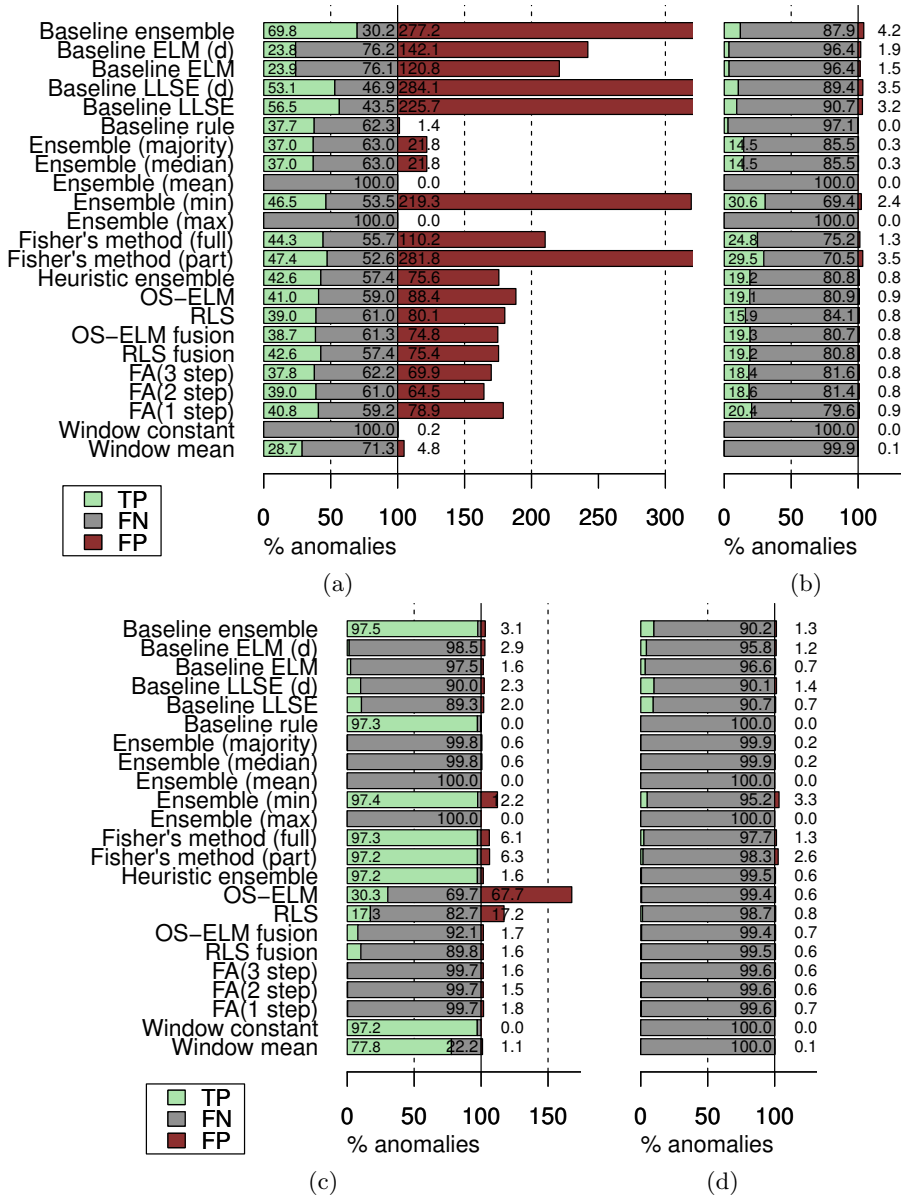


Figure 4.7: True Positives (TP), False Negatives (FN) and False Positives (FP) as the ratio of number of anomalies in synthetic datasets. We can distinguish the anomaly types (a) spike, (b) noise, (c) constant and (d) drift.

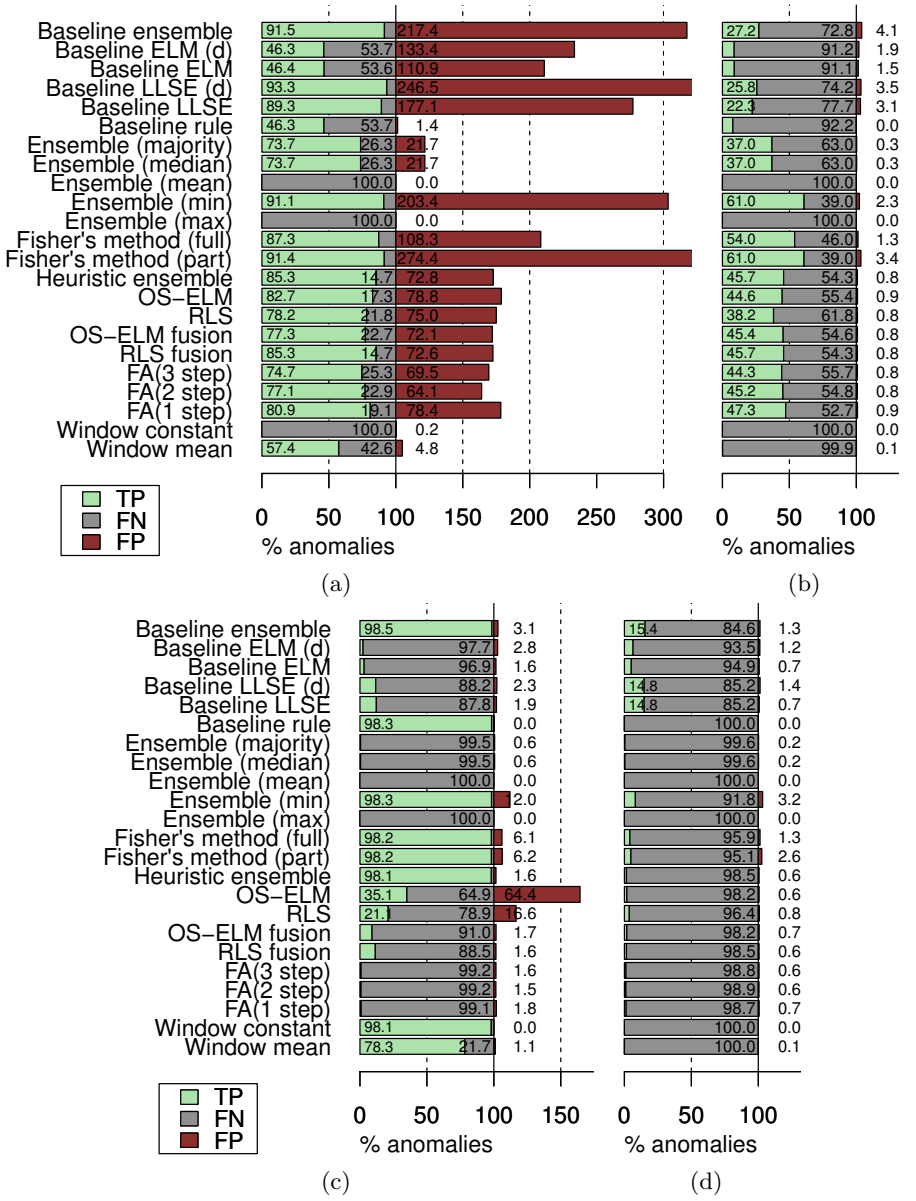


Figure 4.8: TP, FN and FP as the ratio of number of anomalies in synthetic datasets. The TP and FP are improved due to a context window of length $L_b = 3$ around a detection, as discussed in Section 4.4. We can distinguish the anomaly types (a) spike, (b) noise, (c) constant and (d) drift.

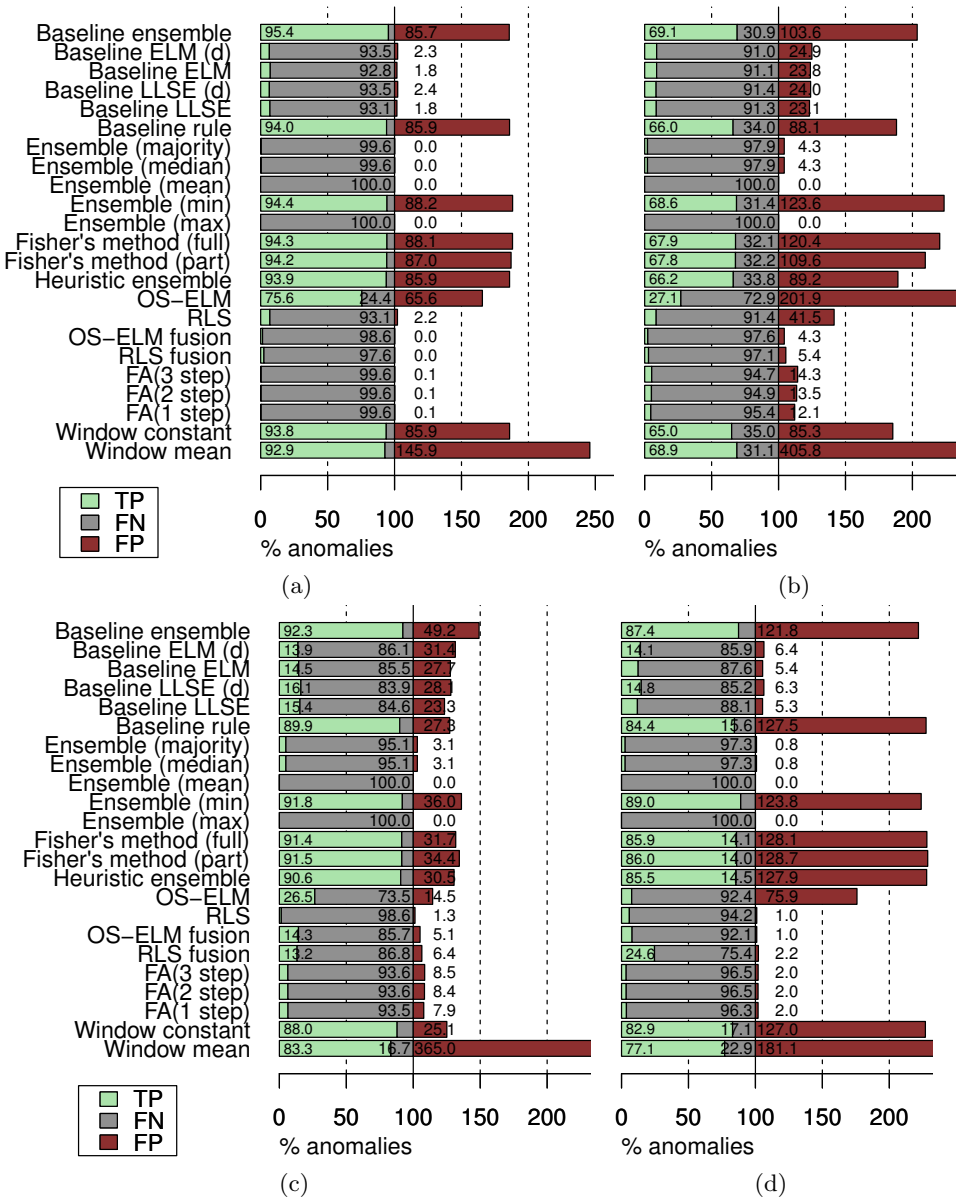


Figure 4.9: TP, FN and FP as the ratio of number of anomalies in real-world datasets. The *constant* anomaly prevails, resulting in the good scores of the constant rule and Baseline rule (offline). We can distinguish the datasets (a) Intel Lab, (b) Indoor WSN, (c) GSB temperature and (d) GSB humidity.

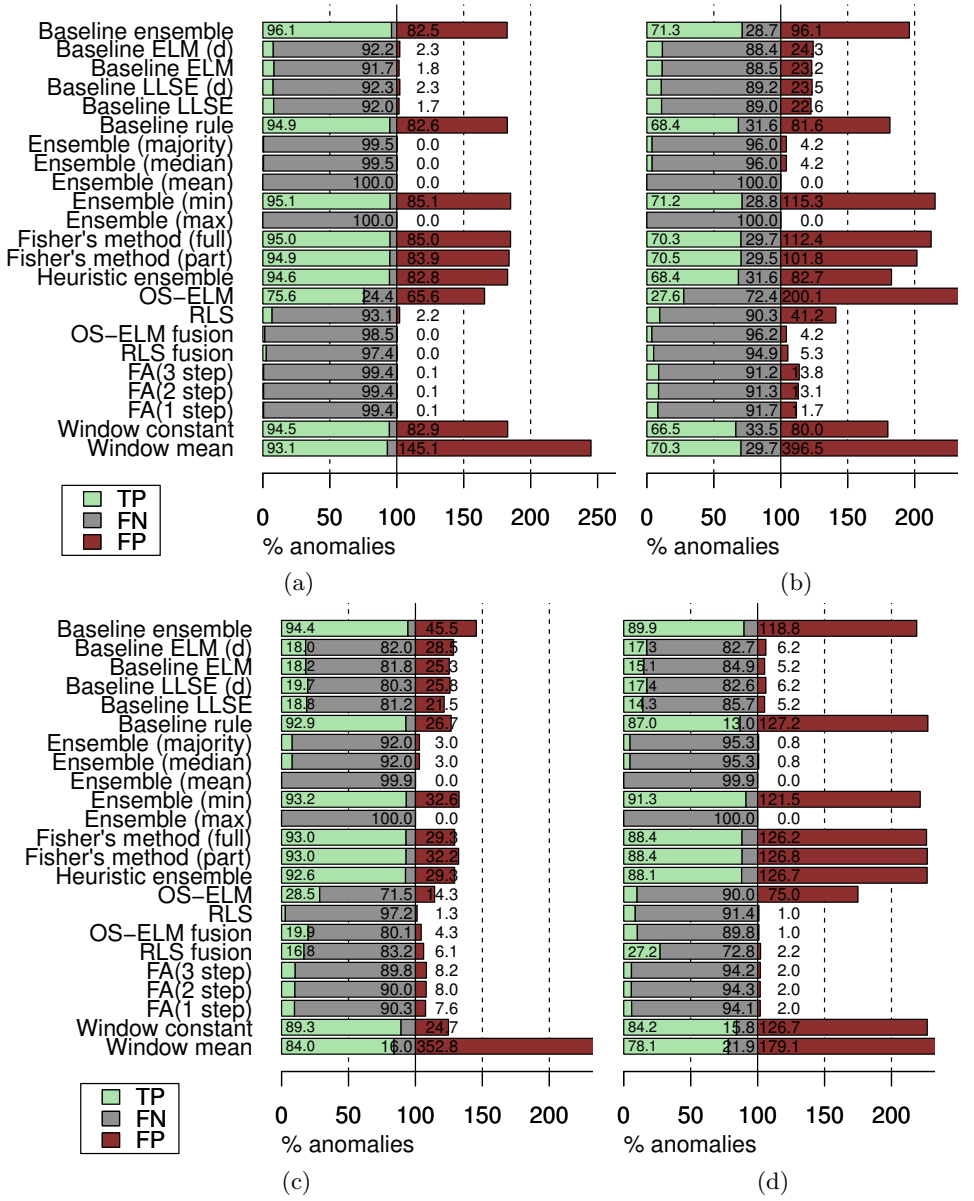


Figure 4.10: TP, FN and FP as the ratio of number of anomalies in real-world datasets. The TP and FP are improved due to a context window of length $L_b = 3$ around a detection, as discussed in Section 4.4. The *constant* anomaly prevails, resulting in the good scores of the constant rule and Baseline rule (offline). We can distinguish the datasets (a) Intel Lab, (b) Indoor WSN, (c) GSB temperature and (d) GSB humidity.

the case of Fisher's method (part), which combines a partial set of classifiers, and the minimum p-value ensemble, the precision is cut by half. A similar performance can be observed for the baselines (other than the baseline rule), that also have low precision but high recall. The majority voting and median ensemble behave equally (as expected from the median voter theorem [153]), but have relatively low TP ratio, similar to the offline rule baseline. The positive effect of voting, however, results in the relatively high precision. Overall, if we calculate the F-measure, the median/majority voting ensemble has the best score for *spike* anomalies, outperforming even the baselines.

The detection performance for *noise* anomalies is similar to that of spike anomalies for most methods. However, because the number of samples affected by the noise anomaly is larger, the resulting TP ratio (around 40%) is less than for *spikes*. The FP ratio, on the other hand, is very low for all methods, implying a high precision (over 97%). This could indicate that the methods do signal the anomalies, although not all anomalous samples are detected. Similar to the performance on *spike* anomalies, the heuristic, Fisher's and minimum p-value ensembles also display a doubling in FP ratio, with respect to the individual classifiers, from 1.4% to 3.4% FP. But, in this case, the increase in TP ratio from 44% to around 54 to 60% for ensemble methods is more than the increase in detection for *spike* anomalies. The maximum F-measure shows that the online minimum p-value ensemble has the best average results, although Fisher's method shows similar performance. For the *noise* anomaly, again, the online methods seem to outperform the offline baselines: while for the baseline the precision is high (99% for the rule baseline), the recall is low compared to the online methods.

The *drift* anomaly is the most difficult to detect, as we can see from the performance results shown in Figure 4.8. Overall the online detection methods seem to adapt to drift, but, if they detect it, they do it only in the beginning of the anomaly. The individual classifiers have a TP ratio of around 1% and a FP ratio that is only a little lower, with a precision around 65%. The ensemble methods do, also in this case, display an increased TP, but also an increased FP ratio, giving them a small benefit in recall but with equal precision. It is here that the offline baseline methods have a clear advantage. Where the online learning methods slowly adapt to the drift, the static model of LLSE and ELM generate larger prediction errors, resulting in a much higher TP ratio, and a higher precision. This is reflected in the maximum F-measure for drift, seen in Table 4.1, which is established by the baseline ensemble with 92% precision and 15% recall. If, however, we only evaluate the online methods, the minimum p-value ensemble gives the best F-measure with 81% precision and 8% recall.

Overall, the ensembles in general do have a positive benefit on the performance. However, the max and mean p-value ensemble do not perform well at all, as they are both affected too strongly by the "least confident" classification. Another observation is that the majority voting and median ensemble schemes perform equally, as expected from the median voter theorem [153]. While their

TP ratio (recall) is less than any other classifier, the number of FP produced by these ensembles is much less than other classifiers, thus resulting in relatively high precision. If the target were, however, to detect (recall) the largest number of anomalies, then the heuristic, Fisher's or minimum p-value methods are a better choice for our scenarios . Furthermore, these ensembles come the closest to matching the performance of the baseline (offline) ensemble.

4.5.5 Anomaly detection performance on real-world datasets

We have performed the same analysis for the three real-world datasets, where we have split up the GSB data in temperature-related and humidity-related sensors due to memory limitations. The results for these datasets are shown in Figure 4.10 and Table 4.1. From observations made after manually checking the labels (according to the procedure outlined in Section 1.2.2), we know that the predominant anomalies are constants, often resulting from missing values. This predominance is the reason for the performance of the constant classifier, which has very high TP ratios of 70% and higher. However, the FP ratio of the constant classifier is much higher for the real-world data than for the synthetic data, resulting in a precision from 40% to 78%. This is caused by stable natural signals, such as the low light intensity readings at night.

The OS-ELM-based classifier performs well across most real-world datasets, except for the GSB humidity-related data. Its performance matches or exceeds that of the baseline ELM method in terms of recall. However, its precision is lower (from 11% to 53%), resulting in a high number of FP. The latter could be caused by the random initialization of the hidden node weights and biases. The RLS classifier, on the other hand, has slightly lower recall than its baseline counterpart, LLSE. For example, on the GSB temperature data the RLS recall is only 2.8%, while LLSE recalls around 19% of the data. However, for this dataset in particular its 68% precision is higher than the 38% of LLSE. On the other datasets, the precision is similar to LLSE. The recall performance of the FA-based classifiers seems to be low compared to the other individual classifiers, particularly for the Intel Lab data recalling less than 1%. However, their precision is on par with the other methods. Moreover, for the GSB temperature data, the FA-based classifiers have better recall than RLS.

Combining the predictions of these classifiers, using the RLS or OS-ELM fusion, shows varying results. For all the real-world datasets, the precision of these fusion methods is the highest of all classifiers (see the bold face numbers of Table 4.1). However, for the Intel Lab and Indoor WSN datasets, the recall (with 1 to 5%) is much lower than the individual RLS or OS-ELM methods (having 7 to 27% recall). On the other hand, the fusion classifiers perform on par with the LLSE and ELM baseline classifiers for the GSB datasets in terms of recall, and outperform them in terms of precision.

There are similarities in the datasets for the Intel Lab and Indoor WSN data, in terms of type of sensors and environment. This also holds, to a lesser

extent, for the GSB temperature and GSB humidity sensors. These similarities are evident from the results of the different methods where, for instance, the baseline methods for the GSB datasets show higher recall than for the other real-world datasets (around 18% vs around 10%). Similarly, the individual classifiers, and mainly the fusion classifiers, show better performance for the GSB data. This might be the result of the indoor (Intel Lab, Indoor WSN) versus outdoor (GSB) nature of the data: changes in the signal for outdoor data may occur more slowly than changes indoor, that are affected by human activity.

Overall, a proper choice of the ensemble method is beneficial to the detection performance. However, similar to the results from the synthetic datasets, the max and mean p-value ensembles do not perform well on the real-world data. The majority voting and median p-value ensembles perform equally, and outperform the baseline ensemble in terms of precision, as clearly shown by the FP ratios. Their drawback, however, is the very low TP ratio, or recall, which is the lowest across all methods. A much better performance is achieved by the Fisher's method, the heuristic and the minimum p-value ensembles. Their results are dominated by the constant classifier, but the other methods do contribute extra TP, resulting in higher recall. The performance of the heuristic ensemble is, according to the F-measure shown by the dashed underlines in Table 4.1, often the best of the online ensembles, closely followed by Fisher's method and the minimum p-value ensembles. On the whole, the performance of the decentralized online learning methods in Fisher's method or minimum p-value ensembles is very close to the baseline ensemble, showing that ensembles of classifiers are a viable approach to anomaly detection in resource-limited systems.

4.6 Conclusions

In this chapter we combined the diverse individual classifiers presented in Chapter 3 in *ensemble* and *fusion* approaches, which we hypothesized (and confirmed through experimentation) would further improve the anomaly detection accuracy, overcoming the limitations of the individual classifiers. In the ensemble, the individual classifiers act in parallel on the same data, while their classifications can be aggregated either by using simple heuristic rules (which order the independent classifications according to the most likely kind of anomaly), by applying algebraic combiners, such as the median, or by applying the Fisher's method.

In order to assess the performance of the proposed decentralized anomaly detection framework, we extensively evaluated the ensemble methods, as well as the individual online learning methods (from the previous chapter), and compared to their centralized offline counterparts (taken as baseline). The evaluation was performed using various large synthetic and real-world datasets

Table 4.1: The precision/recall results. Confidence level > 95%, $L_h = 48$, $L_s = 20$, $L_b = 3$. The baselines followed with (d) include a delayed version of the signal. The bold-face numbers indicate the maximum precision or recall. The underlined numbers are the best combination of precision and recall, the dashed underlined numbers are the best online embedded combinations (both calculated according to the F-Measure).

Classifier \ Dataset	Intel Lab		Indoor WSN		GSB temp-erature		GSB humidity		Constant		Drift		Noise		Spike	
metric	pr.	re.	pr.	re.	pr.	re.	pr.	re.	pr.	re.	pr.	re.	pr.	re.	pr.	re.
Window mean	39.07	93.05	15.07	70.33	19.23	84.01	30.37	78.11	98.66	78.31	21.63	0.02	62.16	0.15	92.26	57.42
Window constant	53.29	94.54	45.39	66.50	78.35	89.26	39.93	84.23	**100.00**	98.09	48.28	0.00	23.08	0.00	0.00	0.00
FA(1 step)	88.37	0.58	41.40	8.27	56.15	9.67	74.72	5.91	33.61	0.92	65.95	1.29	98.15	47.25	50.78	80.91
FA(2 step)	88.00	0.57	39.98	8.74	55.38	9.98	74.35	5.73	34.41	0.81	66.66	1.11	98.34	45.19	54.59	77.09
FA(3 step)	85.36	0.60	38.98	8.83	55.43	10.20	73.97	5.78	34.45	0.84	65.88	1.15	98.20	44.27	51.80	74.69
RLS fusion	**99.39**	2.58	**48.93**	5.11	73.29	16.78	**92.53**	27.17	87.73	11.45	71.02	1.52	98.24	45.69	54.05	85.35
OS-ELM fusion	97.80	1.47	47.08	3.78	**82.15**	19.86	91.02	10.20	84.37	9.03	72.80	1.79	98.20	45.36	51.72	77.26
RLS	76.08	6.91	19.03	9.67	68.59	2.83	89.56	8.61	55.92	21.09	82.77	3.64	97.88	38.18	51.02	78.18
OS-ELM	53.55	75.63	12.12	27.60	66.58	28.51	11.74	9.97	35.27	35.09	74.24	1.84	98.04	44.64	51.21	82.67
Heuristic ensemble	53.32	94.61	45.27	68.43	75.99	92.58	41.01	88.07	98.39	98.14	70.97	1.52	98.23	45.69	53.98	85.35
Fisher's method (part)	53.07	94.92	40.91	70.46	74.27	93.00	41.10	88.44	94.05	98.20	65.07	4.87	94.68	60.96	24.98	91.37
Fisher's method (full)	52.78	95.02	38.49	70.31	76.04	92.97	41.18	88.35	94.17	98.20	75.51	4.11	97.65	53.97	44.64	87.32
Ensemble (max)	100.00	0.00	0.00	0.00	0.00	0.00	100.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Ensemble (min)	52.78	95.06	38.17	71.19	74.09	93.18	42.92	**91.34**	89.09	98.28	71.55	8.17	96.35	**60.98**	30.93	91.08
Ensemble (mean)	100.00	0.00	0.00	0.00	100.00	0.06	100.00	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Ensemble (median)	97.23	0.53	48.52	3.96	72.69	7.96	85.37	4.71	43.59	0.46	65.46	0.39	99.27	36.99	77.24	73.69
Ensemble (majority)	97.23	0.53	48.52	3.96	72.69	7.96	85.37	4.71	43.59	0.46	65.46	0.39	99.27	36.99	77.24	73.69
Baseline rule	53.47	94.89	45.62	68.42	77.68	92.90	40.62	86.98	99.97	98.32	37.20	0.01	**99.69**	7.80	**97.15**	46.34
Baseline LLSE	82.19	8.05	32.73	11.00	46.69	18.80	73.27	14.30	86.27	12.24	**95.31**	14.78	87.79	22.34	33.51	89.26
Baseline LLSE (d)	76.73	7.65	31.59	10.83	43.22	19.65	73.73	17.35	83.96	11.80	91.62	14.81	88.09	25.78	27.46	**93.30**
Baseline ELM	82.48	8.33	33.15	11.49	41.79	18.15	74.26	15.13	65.96	3.11	87.69	5.14	85.66	8.86	29.51	46.44
Baseline ELM (d)	77.25	7.78	32.40	11.63	38.72	18.00	73.64	17.27	45.07	2.33	84.58	6.45	81.99	8.80	25.75	46.26
Baseline ensemble	53.82	**96.11**	42.58	**71.26**	67.48	**94.39**	43.06	89.88	96.98	**98.49**	92.25	**15.44**	87.03	27.24	29.62	91.49

and was based on prediction accuracy and confusion matrix metrics. In the latter we have accounted for false positives caused by anomalies in correlated sensors, and for false positives caused by a delayed detection. Our experiments verified the general viability of the local classifiers and the ensembles for use in decentralized online learning anomaly detection. While the performance of both the offline centralized and online decentralized methods varies with the datasets and the kinds of anomaly to be detected, the ensembles based on Fisher's method, minimum p-value and heuristics perform better than individual classifiers on our scenarios. A general trend is that individual online learning methods are characterized by a reduced recall, while their precision is often similar to that of their offline counterparts. Considering known anomaly types (spike, noise, constant and drift), offline centralized methods seem to be more suitable for detecting slow long-term effects such as drift, while noise anomalies are best detected with online decentralized methods. Moreover, depending on the application goal (such as detecting the most anomalies, detecting with few false positives, etc.), different combinations of online classifiers might be appropriate. The rule-based constant classifier is computationally cheap and performs well. If the most frequent anomalies for an application relate to *spike*, *noise* or, to a lesser extent, *drift*, then an efficient choice might be the RLS-based classifier.

The main benefit of our proposed framework, however, derives from the combination of classifiers as prediction fusion or ensembles. By combining the classifications of the different learning techniques, the average performance of the online combinations is increased for all datasets, approaching that of the offline ensembles. For instance, the bold-face numbers in Table 4.1 show that the RLS and OS-ELM-based prediction fusion classifiers show the best precision across the real-world data. And, notably, among the different tested combination mechanisms, the Fisher's and the minimum p-value methods make the online ensembles match the performance of their offline counterparts in our scenarios. However, in terms of balance between precision and recall, the simpler heuristic or minimum p-value ensemble classifiers give the best F-measure for our scenario, shown by the dashed underlines in Table 4.1. For our scenarios the most accurate detection precision is obtained by a majority voting or median p-value ensemble scheme, which delivers the best precision (at the cost of low recall), whereas the most reliable performance across all our scenarios seems to be given by the ensemble based on Fisher's method. However, this may depend on the classifiers in the ensemble and scenarios used. Overall, combining the online methods can provide a valid alternative to the offline baseline methods, to detect anomalies in an online, decentralized manner.

These results do not only show that, indeed, ensembles can improve performance by overcoming limitations of individual classifiers, but, more importantly, show that ensembles are feasible in resource-limited systems. Nevertheless, due to the resource-limited environment that is targeted by this thesis,

there are some limitations in the use of these methods. These limitations mainly concern memory usage: as the memory complexity of the proposed methods grows with the number of inputs (i.e., sensors), the number of inputs that can be processed and the number of classifiers is inherently bounded by the amount of memory that is available on the sensor platform. A partial solution to that would be to share memory buffers between classifiers. However, to the best of our knowledge, this is the first time ensemble methods were shown feasible in resource-limited platforms and, due to technological advances, we assume this limitation will become less strict in the near future.

Delayed detections are an issue for any anomaly detector when anomalies reach the input of the prediction models undetected. While the anomaly may still be detected, as it causes a large prediction error for the next predictions, this makes the evaluation of detection methods through exact time-series label comparison difficult. We have shown that this can be accounted for by sending a context window around a detection. However, future work may investigate better methods for judging detections. For example, the delay (in time) between the anomaly and a detection can serve as weighting for its correctness, or a fuzzy metric can be applied.

While the addition of the rule-based constant anomaly classifier is an obvious one, as we discussed in the related work it is infeasible to add rules for every occurring anomaly since their manifestation in deployment environments is commonly not known at design time. The ensemble methods facilitate the inclusion of generic anomaly detectors in our framework. Therefore, the ensemble of the individual online learning anomaly detectors from the previous chapter demonstrates the feasibility of such an approach in resource-limited systems.

The fusion methods show a less significant increase in recall, but do show a significant increase in precision. This may be due to the redundant inputs from other predictors. Future work may investigate pre-processing methods to generate additional inputs. Furthermore, in the following chapter we explore addition input features stemming from a sensor node's neighborhood, making use of the assumption that an IoT or WSN deployment usually consists of multiple nodes that are spatially distributed over similar environments, and can communicate with each other.

Chapter 5

Neighborhood information

The decentralized nature of WSN results in measurements taken in different points in space, over time. Due to the decreasing cost of the hardware, more nodes can be deployed, which results in higher quality data through redundancy. However, the measurements can contain anomalies that occur with respect to local sensors, to neighborhood information or to global information. For instance, using anomaly detection techniques a node can generate an initial estimate of the reliability of measurements in the context of local spatial neighborhood information.

In this chapter we address the following question: Can the local detection of anomalies, using the methods presented in the previous chapters, be improved (in terms of precision or recall) by combining data from groups of spatially co-located sensor nodes? To answer this question, we present an anomaly detection scheme based on the framework introduced in the previous chapters, with the addition of the incorporation of local neighborhood information. We extensively evaluate this approach over the scenarios presented in Section 1.2.2, which stem from different real-world domains. Then, in order to show the effect of the neighborhood information on the anomaly detection, we compare the performance of the framework with and without the use of neighborhood information.

In this chapter we find that information from spatially co-located nodes can contribute to the anomaly detection performance. However, we find that this is only the case when the data stem from an environment with a similar mixture of dynamic and diffusive processes, resulting in correlated data.

The remainder of this chapter is structured as follows: Section 5.1 presents the new anomaly detection approach and the experimental setup. Then, Section 5.2 shows the characteristics, and Section 5.3 explores the influence that neighborhood size can have. Next, Section 5.4 discusses our experimental results on anomaly detection performance. Finally, Section 5.6 concludes the effects of including neighborhood information in our decentralized framework.

The work presented in this chapter has been published as [155].

The work presented in this chapter has been published as [155].

5.1 Methodology

To evaluate the extent by which neighborhood information fusion can improve the detection performance of our online anomaly detection approach, we first have to provide a context in which this approach can be applied. As mentioned earlier, our work specifically targets anomaly detection on networked embedded devices such as those used in WSNs. In such applications, the network is commonly made of a reasonably large number of nodes (tens to hundreds) with limited resources in terms of computation, memory and energy, but with several transducers to sense their environment. Within this context, in the following we assume that:

- Nodes are deployed within communication range, i.e., each node can wirelessly communicate with at least 1 neighbor.
- Nodes communications can be overheard by neighboring nodes. For instance, this can be achieved by using collection tree protocol (CTP), gossip, or other network protocols.
- Every node measures the same modalities. Although sensors do not have to be of the same make and model, their output should have the same units (such as temperature in Celsius, humidity in RH, or light in Lux).
  - Communication is reliable, that is, if a node is within communication range, it can always communicate.
  - The node positions are static.

Furthermore, we make few assumptions on the process or environment that the WSN will monitor:

- The nodes are monitoring a similar mixture of dynamic processes [156]. This mixture of processes is diffusive, i.e., overall the process behavior is correlated over space/time [157]. For example, thermodynamic processes are diffusive over space/time.
- The process (and its diffusive properties) may change over time.
- Anomalies may occur in the process and/or in the sensor system and show a disturbance of the correlation in time or space.
  - The measurement period is smaller than the shortest time-constant of the dynamic process, such that the measurement is relevant (i.e. correlated) in this period.
  - The occurrence of anomalies is asynchronous (unrelated in time/space).

The above assumptions, the most straightforward ones indicated with the open bullets, may also be relaxed. For instance, the reliable communication may be relaxed if the measured process dynamics are much slower than the measurement period, or when there are enough nodes in the neighborhood

for aggregation. The latter also is required when nodes are mobile, to ensure a stable aggregate value. Furthermore, if the measurement period is larger than the dynamic process speed, the measurements may still contribute if the correlation is high. Also, anomalies could occur synchronously and may be detected, if the number of anomalous nodes is the minority. However, to focus the investigation on the effect of neighborhood information, we do not relax these assumptions.

These assumptions allow us to propose that prediction-based anomaly detection methods can be improved with the use of dynamically aggregated neighboring information. This information stems from periodic updates that are sent out by neighboring nodes. The updates can be stored in neighboring nodes and, through the use of an aggregation operator, can provide extra input features that are robust to the dynamic nature of the network. In the following sections we outline our approach, and how we evaluate this proposition using a WSN simulator with topologies and sensor data traces from real-world applications. We first characterize the neighborhood by analyzing the possible relevance of neighboring information. As a measure of relevance we use correlation coefficients, which are calculated between local sensor measurements, neighborhood sensor measurements and (using different operators) aggregated neighborhood information. Then we examine the influence of the neighborhood size on the anomaly detection performance of fusion classifiers that include neighborhood information. That is, we try to answer the question: Will including more neighbors make the anomaly detection perform better? Finally, we analyze the detection performance for an optimal neighborhood size, and for a less than optimal size.

5.1.1 Neighborhood aggregation

In common monitoring applications, where all measurements are forwarded to a sink node, every node periodically communicates its latest measurement. To aggregate neighborhood information, therefore, a node in the neighborhood can overhear these messages and store them locally. This push-based approach is further motivated by the claim that push messages seem more efficient (that is, have lower overhead) than pull messages [158]. However, in order to reduce data communications, an application designer can choose to broadcast messages only in the neighborhood, or with a longer period as long as the period for a modality is smaller than the shortest period in the dynamic process for a given modality. The aggregated data is then summarized through the use of an aggregation operator such as the average (or mean), the standard deviation, the median, the minimum or the maximum. While the number of neighbors may vary, for example due to network conditions or anomalies in the data, an aggregation operator reduces these effects to a single, more stable measurement.

The measurement and anomaly detection protocol is as follows:

1. Each node measures d sensors/modalities, each with their own period p_d.
2. Each measurement is appended with a score of anomalousness based on previous local and neighborhood information.
3. Each modality can be sent separately (different packets for different sensors/modalities, because measurement periods may differ).
4. Each node buffers *recent* neighbor measurement messages per modality, with the above assumption that recent measurements are still relevant.
5. One or more aggregation operators are applied to the buffer (known anomalous measurements are excluded).
6. The aggregates are included as prediction inputs for anomaly detection.

Since, by assumption, communication is reliable and the nodes are static, then every node has at least one recent measurement from any of its neighbors, and the number of neighbors of a given node does not vary. This allows us to focus on the contribution that neighborhood information may have on the anomaly detection performance. In our experiments, the term *recent* is defined as being not older than 1 measurement period p_d. Due to aforementioned assumptions on the time constant of the monitored dynamic process, measurements within this recent period are assumed relevant (i.e. correlated). If the process dynamics allow it, the definition of 'recent' can be relaxed to include multiple measurement periods, for example to account for less reliable networks. Another option could be to weigh the stored values, where each weight is determined by the age of the value. Measurements older than a certain time-window, then, have zero weight. If all measurements are older than this threshold, an anomaly at network level may have occurred. In this case the latest stored messages might still contain relevant information (and/or this anomaly should be reported, if communication is still available).

5.1.2 Neighborhood characterization

In order to evaluate the influence of the neighborhood information aggregate on the anomaly detection performance, two aspects should be considered. The first is the amount of information that can be extracted from a neighborhood. This can be estimated by the (cross) correlation between the neighborhood, the local sensors, and the aggregated neighborhood information. An alternative to correlation is a measure of spatial entropy, explained in the sequel. Establishing the amount of correlation also allows us to validate the aforementioned assumptions on the process. For example, measurements at different locations from a similar mixture of diffusive processes should be correlated because the physical phenomena of one location diffuse to another. The second aspect is the size of the neighborhood, which correlates to the network density. While one can argue that more neighboring information can result in more reliable

statistics, it is reasonable to assume that neighbors at the edge of that neighborhood may measure a different (part of a) physical process that does not correlate.

Both aspects affect the correlation of the aggregated neighborhood information. That is, how well the aggregated information correlates may depend on the size of the neighborhood, and on the aggregation operator chosen. Furthermore, the latter may prove more or less robust to variations in neighborhood size. Thus, we first investigate the correlation of the neighborhood and of the aggregation operators applied to that neighborhood for varying neighborhood sizes. Then an aggregation operator is chosen that correlates best across those sizes. Finally, using the chosen aggregation operator, the influence of neighborhood size on the anomaly detection performance is investigated.

We use the Pearson correlation coefficient as a measure of the linear correlation between two variables **a** and **b** [159]. Its value ranges from -1 to 1, where 1 is a perfect positive correlation, -1 is a perfect negative correlation, and 0 means no correlation. A low correlation would be $|r| < 0.25$, a high correlation means $|r| > 0.75$. For a given sample of size n for both **a** and **b**, the correlation coefficient r can be expressed as:

$$r = \mathrm{corr}(\mathbf{a}, \mathbf{b}) = \frac{\sum_{i=1}^{n}(a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_{i=1}^{s}(a_i - \bar{a})^2}\sqrt{\sum_{i=1}^{s}(b_i - \bar{b})^2}}$$

Since negative correlation is also correlation that contributes to the model, we take the absolute correlation $|r|$. In the sequel, correlations between local sensors and neighborhood information are averaged over each node in a deployment scenario to account for differences in node neighborhoods, which depends on the topology in the scenario. In order to account for bias in averaging of correlations, Fisher's Z-transform, $z = Z(r) = \mathrm{arctanh}(r)$, should be applied before averaging, and the inverse transform, $r = Z^{-1}(z) = \tanh(z)$, on the result [160].

The correlation coefficients are averaged over all nodes in a scenario and stored in a matrix, which can be graphically depicted as a heat map (i.e., a correlation map). In the following, the definition of this matrix is explained. We refer to the neighborhood of node i as $N(i)$. Sensor modalities are referred to with subscript indexes s and m. Then, measurement time-series data of node i for sensor m are referred to as x_m^i. For the buffered neighborhood data of node $j \in N(i)$ the data are referred to as $x_m^{i,j}$. The set of neighborhood measurement time-series for sensor s is $\{x_s^{i,j} : j \in N(i)\}$, for brevity sometimes referred to as X_s^i, and we can aggregate those per time-instance using an operator OP, such as the mean, resulting in a single time-series. The correlation coefficients r are calculated for each pair **a** and **b** of measurement time-series from local sensors (in a given node i, e.g. $\mathrm{corr}(x_0^i, x_1^i)$), local sensors and neighborhood aggregates (e.g. $\mathrm{corr}(x_0^i, \mathrm{OP}(\{x_m^{i,j} : j \in N(i)\}))$ where i is the node under investigation), local sensors and sensors of neighboring nodes (e.g. $\mathrm{corr}(x_0^i, x_0^{i,j})$

where i is the node under investigation and $j \in N(i)$ a neighbor), pairs of neighborhood nodes (e.g. $\text{corr}(x_0^{i,j}, x_0^{i,k})$ where $j, k \in N(i)$ are two neighbors of i). Similarly, we compare neighborhood aggregate time-series to the neighboring node measurements and neighborhood aggregate time-series to other neighborhood aggregate time-series of different operator types (for example mean and median) in order to explore how well the operator correlates with (summarizes) the neighborhood and if it differs from other operators. These are then averaged for all nodes in a given scenario using Fisher's Z-transform.

This process is summarized in Algorithm 6, where the keys of the map M are strings and, as such, they are indicated with double quotes (“ ”) to distinguish them from numerical values. Moreover, because the correlation coefficient between **a** and **b** is the same as the correlation between **b** and **a**, the matrix is symmetric. The diagonal of this matrix should be one, as the correlation coefficient of a signal with itself is one. However, in our matrix, we also compare the correlation among the neighbors of a node, which results in a less than one correlation because we are not comparing a neighboring signal with itself, but with another neighbor's signal of the same sensor. Using the correlation map, we can see stronger correlations having a darker color, which allows us to visually examine the relevance of neighborhood aggregates to local sensor values, and compare them to the raw correlation per neighbor.

The correlation coefficients between local sensors of a node shows how well they correlate and, thus, how much information can be obtained locally. This can be compared to how well the neighboring sensors correlate, which gives an indication of how well aggregated neighborhood information should correlate. Most important, the average correlation between pairs of neighboring nodes can be compared to the correlation between a local sensor and the neighborhood aggregate, to form an indication of the aggregation operator's capability to represent the neighborhood information. In a later experiment, then, we test the intuition that more correlated information contributes more to the anomaly detection performance.

As an alternative to correlation, we use the spatial entropy [161], a measure of complexity of spatial systems defined as:

$$H = -\sum_i p(e_i)\Delta e_i \log(p(e_i)\Delta e_i).$$

This value gives an entropy figure based on the probability $p(e_i)$ of an event in an area Δe_i. The probability in this case is the chance of an anomaly occurring at a specific node i, and is estimated using labeled events from our data sets. The exact area that a node senses, however, is unknown. But, we do know the node positions. With those, either a Delaunay triangulation or a Dirichlet tessellation (a Voronoi diagram) can be constructed to create an estimated area (around the node positions) that a node can sense [162]. To calculate Δe_i, then, we can then take either 1/3 of the total area of the three Delaunay triangles emanating from a node position e_i, or the cell size

Algorithm 6 Correlation map creation

correlation map $M \leftarrow 0$
for each node i in scenario **do**
  for each local sensor pair m, s of node i **do**
    $M[\text{“}x_m\text{”} , \text{“}x_s\text{”}] \mathrel{+}= Z(|\text{corr}(x_m^i, x_s^i)|)$
  $l \leftarrow$ number of neighbors in $N(i)$
  for each sensor pair m, s **do**
    for each aggregate operator OP_a **do**
      $X_m^i \leftarrow \{x_m^{i,j} : j \in N(i)\}$
      $X_s^i \leftarrow \{x_s^{i,j} : j \in N(i)\}$
      $M[\text{“}x_m\text{”} , \text{“OP}_a(X_s)\text{”}] \mathrel{+}= Z(|\text{corr}(x_m^i, \text{OP}_a(X_m^i))|)$
      for each aggregate operator $\text{OP}_b : \text{OP}_b \neq \text{OP}_a$ **do**
        $M[\text{“OP}_a(X_m)\text{”} , \text{“OP}_b(X_s)\text{”}]$
          $\mathrel{+}= Z(|\text{corr}(\text{OP}_a(X_m^i), \text{OP}_b(X_s^i))|)$
      for each neighbor $j \in N(i)$ **do**
        $M[\text{“OP}_a(X_m)\text{”}, \text{“}x_s\text{”}]$
          $\mathrel{+}= Z(|\text{corr}(\text{OP}_a(X_m^i), x_s^{i,j})|)/l$
    for each neighbor $j \in N(i)$ **do**
      $M[\text{“}x_m\text{”} , \text{“}X_s\text{”}] \mathrel{+}= Z(|\text{corr}(x_m^i, x_s^{i,j})|)/l$
  $p \leftarrow$ number of neighbor pairs in $N(i)$
  for each pair of neighbors j, k in $N(i)$ **do**
    for each sensor m of node j **do**
      for each sensor s of node k **do**
        $M[\text{“}X_m\text{”} , \text{“}X_s\text{”}] \mathrel{+}= Z(|\text{corr}(x_m^{i,j}, x_s^{i,k})|)/p$
$M = Z^{-1}(M/(\text{number of nodes in scenario}))$

from the Dirichlet tessellation, as seen in Figure 5.1c. The resulting values represent the information that can be gained from neighborhood information, where lower spatial entropy values imply more gain. Both the spatial entropy and the cross correlation measures can give us an indication of the validity of the assumption that the process or environment consists of a diffusive mixture of dynamic processes, resulting in correlated behavior over space and time.

The influence of the size of the neighborhood depends on either the radio communication range, or the density of the deployment. In order to simulate a change on these, either the radio range or the positions of nodes can be changed. Since the latter are known for the real-world datasets used in this study, but the exact radio parameters are not, we opt to change the communication range by changing the parameters of the radio model in the simulator. The radio propagation model is a Log-distance path loss model, which predicts the path loss over distance to have a logarithmic decay, with optional Gaussian noise to simulate interference [163]. The static parameters of the model are the unit distance $D_0 = 1.0$, the path loss at this reference distance $PL_{D_0} = 55.4$, and the noise floor is -106.0 dB. Since we assume reliable communications, we do not add a noise component to the radio model. The main parameter is the path-loss exponent (PLE), which dictates the decay of signal strength over distance. That is, higher values of PLE result in higher path loss and thus a smaller radio communication range, whereas low values result in less path loss and a larger communication range, an example can be seen in Figure 5.1. Therefore, we vary the PLE, effectively changing the number of neighbors in the neighborhood, and measure the result on the change in anomaly detection performance, compared to the same classifiers without neighboring information.

5.1.3 Embedded online anomaly detection

In Chapters 3 and 4 we have presented decentralized online anomaly detection methods, suitable for resource-limited systems. By online learning linear, non-linear and polynomial function models future measurements can be predicted. The predictions of these models are then compared to the measured value, and the difference is analyzed to detect anomalies. In the methods presented in the previous chapters, hereafter called the *stand-alone* or SA methods, these features were either from only local sensor data (Chapter 3), or the fusion of local sensor data with local predictions from other models (Chapter 4). The latter (referred to as RLS fusion or OS-ELM fusion) included for each modality the raw sensor data, the previous measurement, the FA prediction, and the window mean. In the previous chapter, this fusion of raw data with other local predictions showed a clear improvement in the precision of anomaly detection.

In this chapter, we replace two of the input features of the stand-alone methods by neighborhood aggregated measurements. Specifically, by replacing the local input features of previous measurement and of window mean by two

neighborhood aggregates in the classifiers, the change in detection performance can be evaluated with the metrics described in Section 1.2.3. We test this addition of neighborhood data using the real-world environmental monitoring scenarios described in Section 1.2.2. Due to the assumption of a similar mixture of diffusive processes, we expect that including these aggregates the anomaly detection performance will increase the RLS- and OS-ELM-fusion classifier performance. Moreover, the ensembles (that combine multiple classifiers) are also expected to be positively affected.

5.2 Neighborhood characteristics

In order to minimize the effect that neighborhood size may have on our choice of aggregation operator, we analyzed correlation maps from the same dataset with different radio model parameters. The correlation maps shown in Figure 5.2 and 5.3 show stronger correlation as a darker color. On both the x and y axes we list the local sensor measurements x_i, the local aggregated neighborhood information $\mathrm{OP}(X_i)$ and the neighborhood information X_i, for sensor modality i. Since $\mathrm{corr}(\mathbf{a}, \mathbf{b}) = \mathrm{corr}(\mathbf{b}, \mathbf{a})$, the correlation map is symmetric. For a given local sensor or aggregate, the correlation with itself is 1 (high), and thus the diagonal is black. However, for the neighborhood information we correlated pairs of nodes, resulting in a lower correlation in the top-right diagonal (for the details see Section 5.1.2).

For example, the correlation coefficients in the bottom left of these maps in Figure 5.2 show that local sensor 2 and 3 are more correlated than sensor 1 and 2 or sensor 1 and 3. This pattern repeats in the top right of the map, showing that the values of sensors 2 and 3 between neighbors are more correlated than the other sensors in the neighborhood, albeit less than the correlation between the local sensors. Moreover, we can see that the neighborhood aggregates correlate better between aggregates of the same sensor than others, resulting in the darker squares. Especially the mean and median are highly correlated, while the standard deviation has low correlation throughout. The mean and median also correlate to the minimum and maximum values in the neighborhood, but to a lesser extent than the correlation between mean and median. Overall, the effect of the neighborhood size can be seen by the overall darker colors of the correlation map of the denser network, showing a higher correlation. The correlation between local sensors and aggregate operators in a denser network also increase significantly, seen in the bottom three rows. Specifically, the mean and median operator gain significantly for the first sensor, x_1. Furthermore, As neighborhood size increases in our scenarios, the mean and median aggregate operator stay highly correlated. Therefore, the mean and median seem a reliable choice. While this does not necessarily hold for other scenarios, we expect that these operators are the most suitable under the assumption of a similar mixture of diffusive processes.

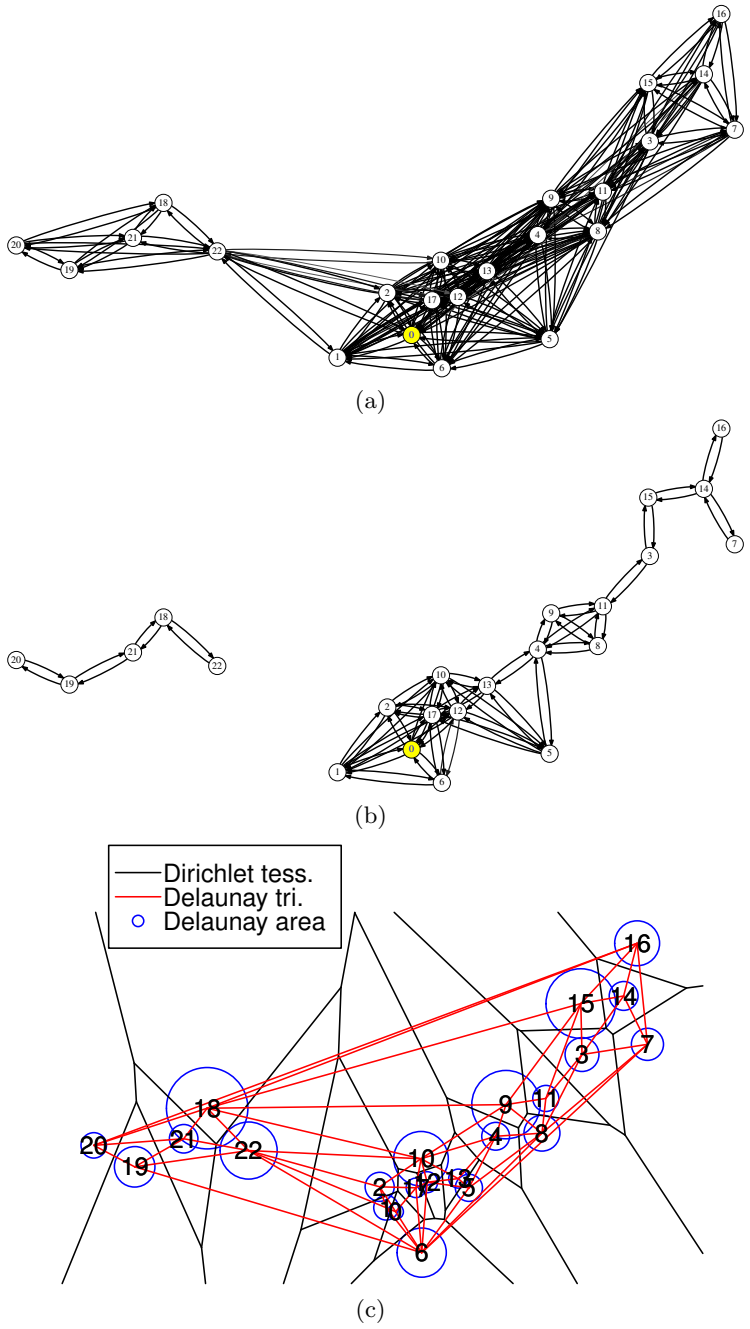
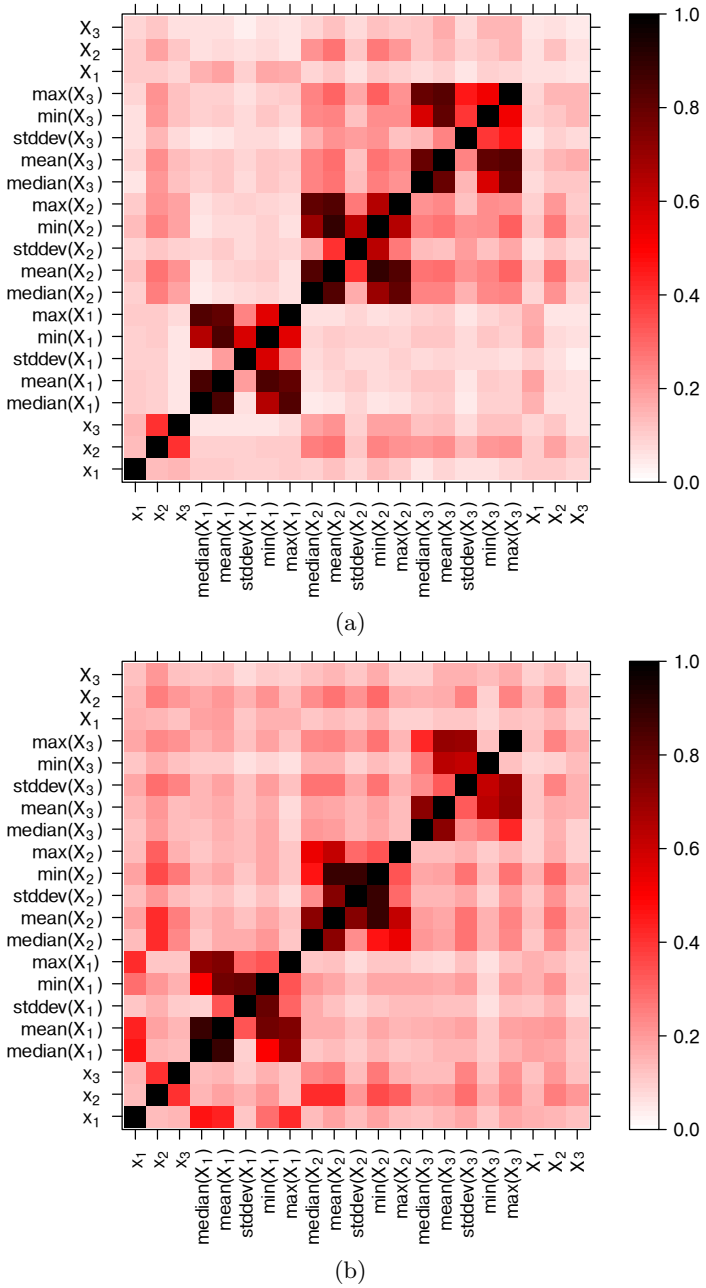


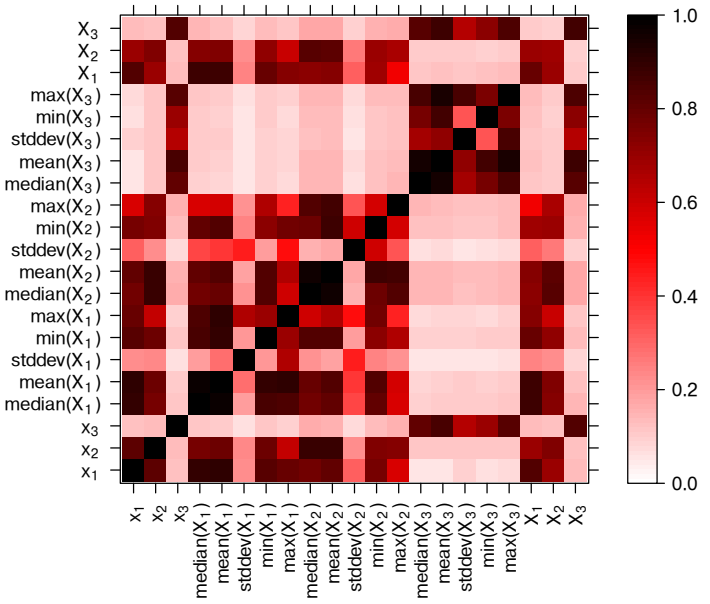
Figure 5.1: SensorScope GSB network topology. (a) A dense network simulation with PLE=4.0, (b) A sparse, disconnected, network simulation with PLE=6.2, (c) The Dirichlet tessellation, Delaunay triangulation, and node area based upon the triangulation. Note that even though the network in (b) is disconnected, each node can communicate with a neighbor.



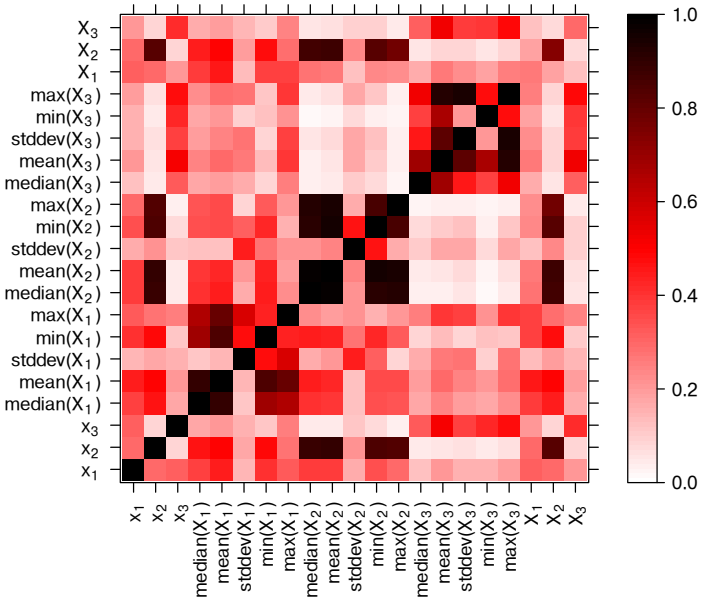
(a)

(b)

Figure 5.2: A denser connected network results in better correlation with neighborhood aggregates. Furthermore, it shows different sensor modalities have different correlations with their neighborhood and with other sensors (sensor 2 and 3 are more correlated than sensor 1 and 2). The correlation maps show the average correlation of the GSB humidity related sensors, with (a) a sparse network (PLE=6.2), and (b) a denser network (PLE=4.0).



(a)



(b)

Figure 5.3: With the same set of sensors but different environments the correlation between sensors and their neighborhood aggregates shows similar patterns. However, due to environmental differences, the average correlations are lower in the Indoor WSN dataset (b). The correlation maps stem from (a) The Intel dataset (PLE=4.0) and (b) the Indoor WSN dataset (PLE=4.6).

In Figure 5.3 two different scenarios are depicted, namely the Intel Lab and the Indoor WSN testbed datasets. Both have similar environments (indoor offices), and have the same set of sensors (temperature, humidity and light), but the Intel Lab dataset shows much higher correlation throughout. Interestingly, the dataset similarities also show in the correlation patterns. That is, sensors 1 and 2 (temperature and humidity) have higher correlation between each other than the light measurements have with any of them. However, there are large differences due to the environment. The main finding is that here, too, mean and median of a neighborhood, for a specific sensor type, are highly correlated to the local sensors. The minimum and maximum show a slightly lower correlation and more variation. This indicates that both mean and median operators should be a good choice to aggregate neighboring information.

Table 5.1: The average cross-correlation and spatial entropy of the datasets. The spatial entropy [161] is based on the chance of anomalous measurements per area, where the area is based on the Delaunay triangulation or Dirichlet tessellation over known node positions.

Dataset	Cross-corr.	Spatial Entropy	
		Delaunay	Dirichlet
GSB Humidity	0.131	0.159	0.211
GSB Temperature	0.191	0.255	0.269
Intel Lab	0.615	0.141	0.143
Indoor WSN	0.261	0.296	0.268

Lastly, to characterize the datasets, we analyze their average cross-correlation and spatial entropy. In Table 5.1, the cross-correlation value is the average cross-correlation over the whole dataset. The table shows that the Intel Lab dataset has the highest cross-correlation and the lowest spatial entropy (with both area measures). Therefore, we expect the information gain of neighborhood information to be high. While the second highest cross-correlation value is from the Indoor WSN, the higher spatial entropy values might indicate that the information gain of spatial neighborhoods might be less. Both GSB datasets have relatively low cross-correlation and higher spatial entropy. Therefore, the neighborhood information gain will most likely be little. Overall, we expect that the Intel Lab dataset will gain the most from neighborhood information.

5.3 Evaluation over neighborhood size

Having established that the mean and median are suitable choices for aggregation operators, and that the Intel Lab dataset fulfills the assumption of a similar mixture of diffusive processes more than the other datasets, the influence of neighborhood size can be analyzed. By changing the PLE the global

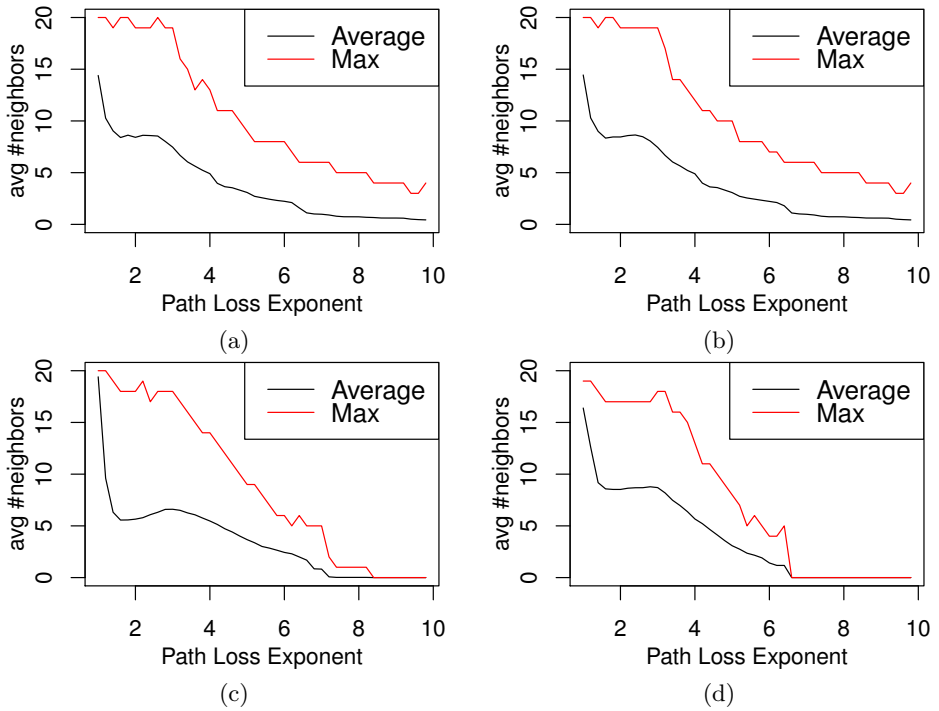


Figure 5.4: The number of neighbors decreases when the radio range decreases. The maximum of 16 is the result of the neighborhood buffer size. The figures show PLE vs average number of neighbors for the GSB topology in the case of (a) humidity and (b) temperature, which should be equal. For the Intel Lab topology (c) and for the Indoor WSN topology (d) the ratios are different.

average number of neighbors changes in the network, as seen for each dataset in Figure 5.4. Due to the limited memory size available in the WSN nodes, the number of neighborhood messages stored is 20, which shows as the maximum number of neighbors in the figure. Thus, the size can be analyzed on the global network, but also per number of neighbors that individual nodes have. For the sake of brevity, in the remainder of this section we show only the results of the RLS-fusion classifier. The results of other affected classifiers result in equal findings.

5.3.1 Network average number of neighbors

We first show the change in F-measure for the RLS-fusion detector, as a result of replacing some input features with neighborhood aggregates, plotted against the PLE setting in Figure 5.5. We have regressed these values with a line to get an understanding of their trend. Here, we see large differences per application scenario. Most notably, the Intel Lab dataset shows the highest relative improvement in F-measure of over 150%, which can be explained from the high correlation that was observed in the previous chapter. Moreover, the other scenarios show a peak in F-measure at different PLE settings, suggesting that there is no one optimal PLE setting for a WSN deployment.

Then, instead of the PLE setting, we plot the change in F-measure for the RLS-fusion detector against the average number of neighbors (derived from Figure 5.4). To get a better insight in the trend, polynomials of first to fourth order are regressed to this data. The highest order polynomial that has significant improvements over lower-order polynomials according to the ANOVA test with p-value=0.05 is displayed. The results in Figure 5.6 again show a difference per topology and per dataset. However, analyzing the F-measure per average number of neighbors shows a more clear effect of the number of neighbors on the F-measure improvement compared to the effect of PLE.

From these figures, we can see that for all the datasets the inclusion of neighborhood information does seem to improve the F-measure performance, albeit to a varying degree. The cases where the PLE is too large (and thus radio range too small) to allow any communications mostly result in zero improvement. Moreover, in most cases there is an optimum average number of neighbors (and thus an optimum radio range). However, the optimal radio range depends not only on the topology but also on the dataset. For example, the GSB temperature and humidity datasets in Figure 5.6a and 5.6b share the same topology, but have different optima. On the other hand, the Indoor WSN dataset shows a clear peak around an average of 4 neighbors, and the optimum of the Intel Lab dataset averages around 5 to 6 neighbors. The Intel Lab dataset also shows the highest relative improvement in F-measure of over 150%, which can be contributed to the dataset characteristic of being highly correlated. In the future, the framework may benefit from adaptive transmission power control, such that an optimal number of neighbors can be chosen

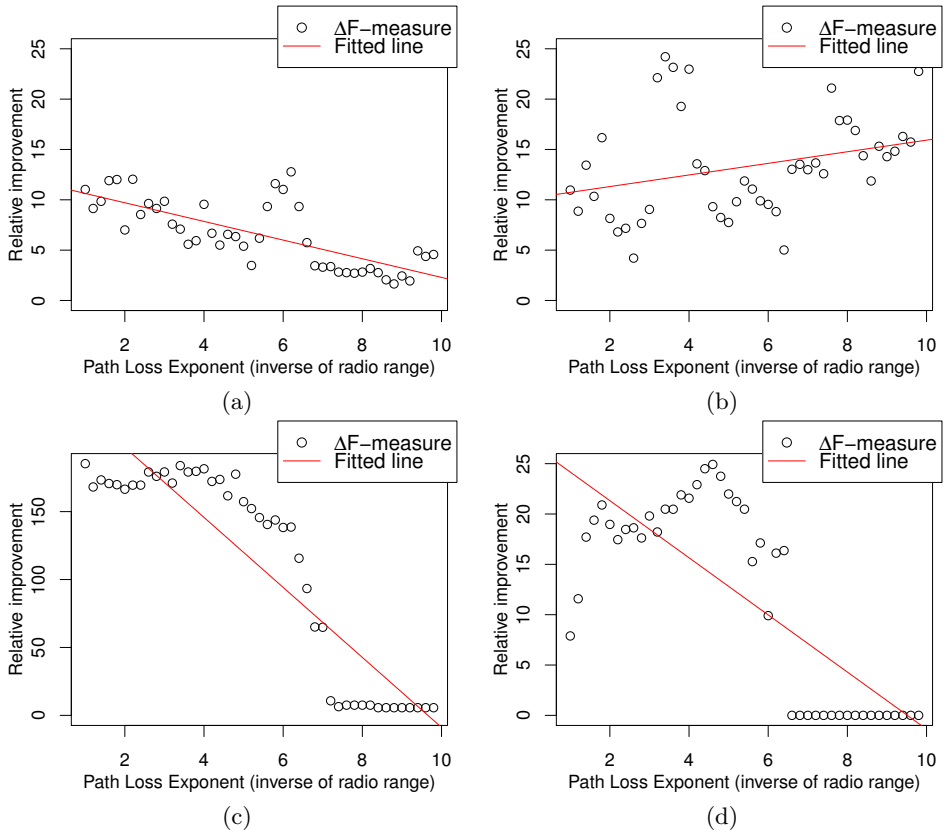


Figure 5.5: The PLE vs the relative change in F-Measure of the RLS-fusion classifier. The datasets are (a) GSB humidity, (b) GSB temperature, (c) Intel Lab (note the different y scale), and (d) Indoor WSN.

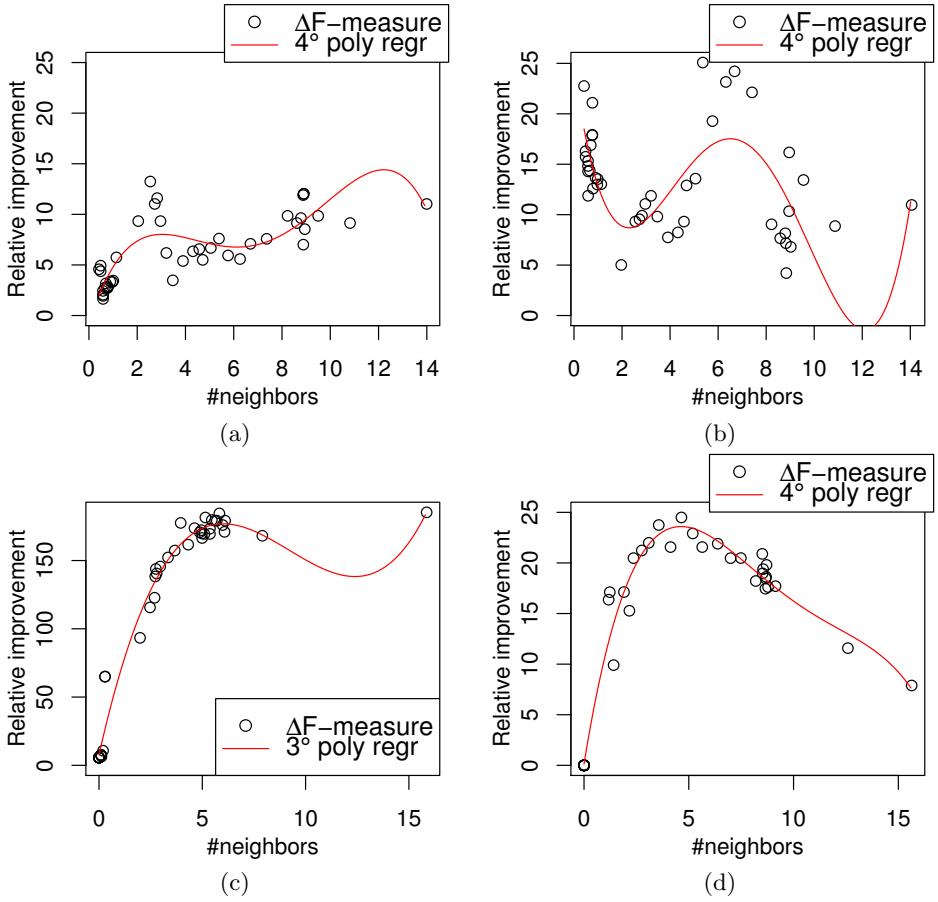


Figure 5.6: The optimal number of average neighbors depends on topology and sensor modality. The plots show average number of neighbors vs the relative change in F-Measure of the RLS-fusion classifier. The datasets are (a) GSB humidity, (b) GSB temperature, (c) Intel Lab (note the different y scale), and (d) Indoor WSN.

for a deployment.

5.3.2 Exact number of neighbors

The above analysis was made using the average number of neighbors a network had at a given PLE setting. To better understand the exact influence of the neighborhood size, we now analyze the results of the RLS-fusion classifier per number of neighbors, over the whole PLE range (from 1.0 to 10.0). In this case, we study the effect of replacing certain input features with the neighborhood aggregates on the precision and recall separately. Both the results of the stand-alone methods and the methods with neighborhood information form two distributions. Using the Kolmogorov-Smirnov test [164], these distributions can be compared and tested if the new recall and precision measurements statistically significantly improve over the stand-alone methods.

Figures 5.7 and 5.8 show the results of the Kolmogorov-Smirnov test for precision and recall respectively. Our null-hypothesis, H0, is that the anomaly detection performance (in terms of precision or recall) when using aggregated neighborhood information is not better than the performance obtained by using only local information. Our alternative hypothesis, H1, is that the performance of the methods that include aggregated neighborhood information is better than when using only local information. Note that in the figures, we test also the case that a node has zero neighbors, because in the new method we did replace two input features, which might affect performance. We test the hypothesis for a significance of $\alpha = 0.1$ (or 10%) and $\alpha = 0.05$ (or 5%).

Figure 5.7 shows that the precision is only significantly better with the Intel Lab dataset, while all other datasets show no improvement (i.e. H0 is accepted). Recall in Figure 5.8, on the other hand, more often shows a significant improvement. Again the highly correlated Intel Lab dataset shows a significant improvement in all cases. For the classifiers other than the RLS-fusion classifier that include neighborhood information, and the ensembles, a similar pattern shows. That is, the methods show a significant increase in recall with the Intel Lab dataset only. The performance in the Intel Lab dataset can be well explained if we go back to Section 5.2, where we characterized the datasets in terms of average cross-correlation and in terms of spatial entropy. The Intel Lab dataset is the only dataset that has a relatively high cross-correlation value, and a low entropy. This leads us to conclude that the assumption (or requirement) of a similar mixture of diffusive processes is valid and, thus, that when neighborhood information correlates, the inclusion of aggregate neighborhood information in the prediction significantly improves the results.

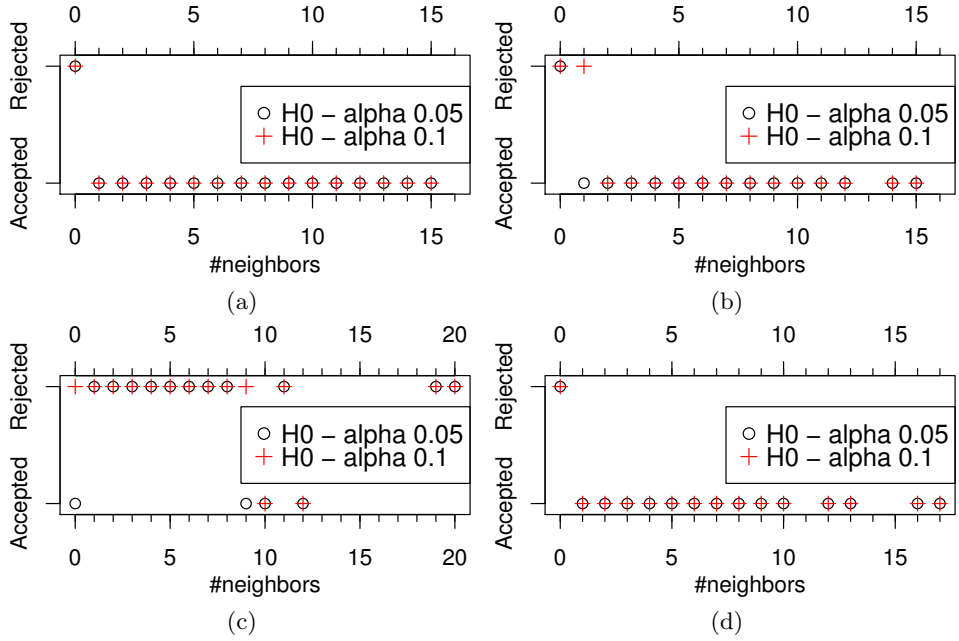


Figure 5.7: Precision does not improve except with the Intel Lab dataset. The plots show if the *precision* statistically significantly improved, and thus H0 is rejected. The datasets are (a) GSB humidity, (b) GSB temperature, (c) Intel Lab, and (d) Indoor WSN.

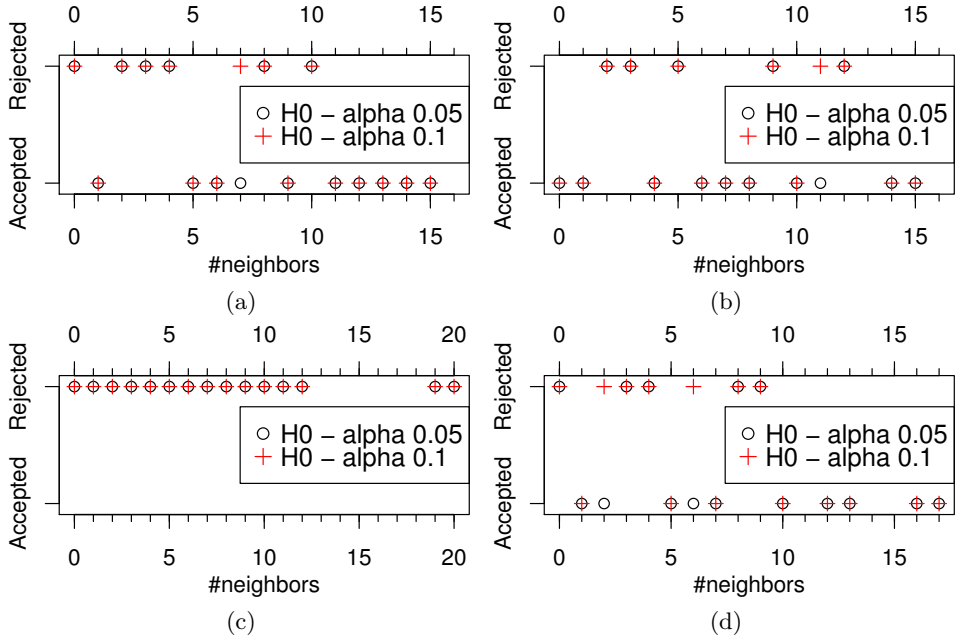


Figure 5.8: Recall does improve depending on the number of neighbors and dataset used. The plots show if the *recall* statistically significantly improved, and thus H0 is rejected. The datasets are (a) GSB humidity, (b) GSB temperature, (c) Intel Lab, and (d) Indoor WSN.

Table 5.2: PLE settings for further evaluation, based on Figures 5.4 and 5.6

Dataset	PLE_{opt}	$N()_{avg}$	PLE_{less}	$N()_{avg}$
GSB Humidity	4.0	5.3	6.2	2.5
GSB Temperature	4.0	5.3	6.2	2.5
Intel	3.4	5.8	6.2	2.7
Indoor WSN	4.6	4.1	6.2	1.2

5.4 Detection performance

With the above information, the detection performance of the other individual classifiers and ensembles thereof can be analyzed in more detail. Again we choose the mean and median neighborhood aggregation to replace the previous measurement and mean as input features in the fusion classifiers. In a real world deployment, often one cannot choose a perfect number of neighbors for each node. Therefore, we opt to evaluate two different PLE settings per scenario, to represent an optimal case in a dense network, and a sub-optimal case in a sparse network. These choices are guided by Figures 5.4, 5.6 and 5.8 and can be seen in Table 5.2.

5.4.1 Absolute precision and recall

The absolute values of precision and recall for all methods are depicted in Table 5.3. In this table, the fusion classifiers that include neighborhood aggregate data are indicated with the postfix '.AG', and those that do not include neighborhood information (the stand-alone methods from our previous chapter) with the postfix '.SA'. We see that in all cases, the offline baseline ensemble has the highest recall. Furthermore, we can see that in the case of the GSB humidity data and especially in the case of the Intel Lab data, the aggregate neighborhood information contributes significantly to the overall performance. For the fusion classifiers we see a clear benefit in recall for most datasets. In the following subsections, we will detail the change in precision, recall and F-measure derived from this table.

5.4.2 Change in precision and recall

In Table 5.4 we show the results of including neighborhood information with these settings, as percentage of improvement over the stand-alone methods (derived from the numbers in the previous section). Here, too, we see that in general recall benefits from neighborhood information, and precision is similar or slightly reduced. The OS-ELM based classifier shows more extreme results due to its random initialization of input weights and biases, but does also benefit from neighborhood information. The results of the Intel Lab dataset show significant improvement for all classifiers. That is, statistical analysis

Table 5.3: Absolute precision/recall numbers. The baseline LLSE and ELM are the non-iterative variants of RLS and OS-ELM respectively. The postfix (d) denotes the inclusion of a day period. The postfix '.SA' indicates the results of the stand-alone methods described in previous chapters, while the postfix '.AG' indicates the current results, including the neighborhoog aggregates.

classifier \ dataset	GSB Humidity PLE=4.0		GSB Humidity PLE=6.2		GSB Temperature PLE=4.0		GSB Temperature PLE=6.2		Intel Lab PLE=3.4		Intel Lab PLE=6.2		Indoor WSN PLE=4.6		Indoor WSN PLE=6.2	
metric	pr	re	pr	re	pr	re	pr	re	pr	re	pr	re	pr	re	pr	re
FA(1 step)	66.69	5.23	66.53	5.23	53.49	9.22	53.49	9.22	89.71	0.59	89.71	0.59	38.57	7.97	38.57	7.97
OS-ELM	11.23	10.50	13.32	13.77	77.83	24.13	47.14	19.24	48.64	58.76	49.41	69.73	15.38	13.35	16.06	12.57
RLS	16.68	16.16	16.68	16.16	80.21	20.72	80.21	20.72	45.32	28.26	45.32	28.26	13.48	10.79	13.48	10.79
Window constant	41.35	83.56	41.35	83.56	**97.81**	88.72	**97.81**	88.72	47.05	95.30	47.05	95.30	**54.14**	63.48	**54.14**	63.48
OS-ELM fusion.AG	**92.59**	15.30	86.39	11.27	85.60	18.28	55.65	5.25	98.64	5.15	98.46	3.58	46.32	5.72	46.82	3.92
OS-ELM fusion.SA	91.72	16.23	86.87	7.77	76.48	10.09	63.88	4.15	99.14	0.31	**99.02**	0.28	43.36	3.06	47.06	2.75
RLS fusion.AG	90.12	25.15	90.63	26.82	61.61	18.05	56.30	15.58	95.99	14.87	95.95	11.26	42.68	8.72	41.88	8.38
RLS fusion.SA	90.26	22.91	90.27	22.91	60.11	13.71	60.11	13.71	98.65	4.74	98.65	4.74	44.87	6.87	44.88	6.87
Fisher's method.AG	41.91	85.78	41.82	85.66	93.35	91.49	92.78	91.45	49.23	95.30	49.18	95.12	52.03	65.60	52.02	65.59
Fisher's method.SA	41.79	85.62	41.80	85.51	93.33	91.34	93.50	91.36	49.13	94.95	49.13	94.95	52.09	65.22	52.16	65.14
Ensemble (heuristic).AG	42.32	86.88	42.14	86.66	88.03	92.37	87.34	92.25	47.25	95.98	47.20	95.75	52.05	67.12	52.07	66.99
Ensemble (heuristic).SA	42.11	86.61	42.11	86.61	89.76	92.15	89.76	92.15	47.12	95.49	47.12	95.49	52.80	66.60	52.80	66.60
Ensemble (min).AG	42.64	87.56	42.34	87.29	84.41	92.80	83.50	92.76	47.30	96.17	47.24	95.90	50.12	68.24	50.42	68.39
Ensemble (min).SA	42.37	87.24	42.43	87.16	86.22	92.61	86.66	92.66	47.15	95.62	47.15	95.62	50.89	67.49	51.00	67.49
Ensemble (median).AG	91.09	8.12	**91.06**	8.08	90.23	8.10	77.98	3.53	**99.75**	4.34	98.40	3.17	47.58	2.91	48.78	2.42
Ensemble (median).SA	90.05	9.12	86.58	5.10	83.10	5.21	77.62	3.41	98.24	0.11	98.71	0.23	47.12	2.40	49.78	2.23
Baseline rule	42.12	87.19	42.12	87.19	90.86	93.31	90.86	93.31	44.62	96.49	44.62	96.49	52.46	66.69	52.46	66.69
Baseline LLSE	66.52	9.64	66.52	9.64	45.28	16.06	45.28	16.06	80.80	8.12	80.80	8.12	32.62	11.41	32.62	11.41
Baseline LLSE (d)	67.72	10.84	67.72	10.84	42.73	16.04	42.73	16.04	75.07	7.78	75.07	7.78	31.19	11.36	31.19	11.36
Baseline ELM	62.86	9.55	63.40	9.39	41.45	17.86	42.75	17.93	79.89	8.12	80.35	7.98	33.84	11.74	32.61	11.31
Baseline ELM (d)	59.80	9.58	63.66	10.02	40.13	17.23	41.50	17.13	73.66	7.37	74.64	7.23	32.77	11.74	32.03	11.60
Baseline ensemble	44.16	**89.36**	44.17	**89.25**	74.85	**94.53**	75.86	**94.55**	44.96	**97.39**	44.95	**97.37**	47.89	**70.52**	47.82	**70.52**

similar to that of Section 5.3.2 for these classifiers shows that, for all of them, there is a significant improvement in recall, but not necessarily in precision. Going back to the dataset characterization, and specifically to Table 5.1, we can observe that indeed the assumption of a similar mixture of diffusive processes is key to achieving good results. That is, the neighborhood correlation should be relatively high.

Choosing an optimum average neighborhood size also results in a better recall performance. The exception here is the GSB humidity dataset, which in Figure 5.6a also showed a different trend (showing a peak around PLE=6.2) and from Table 5.4 we see that this mainly concerns the recall. While this is most likely due to a difference in measured processes (where, for this dataset, the sensors relative humidity, soil moisture and watermark are better correlated with fewer near neighbors), these results are not significant.

Table 5.4: The impact of neighborhood information on recall is beneficially large. precision, however, suffers slightly from neighborhood information. Relative change in precision and recall, in percent.

Classifier \ Dataset	GSB humidity PLE=4.0		GSB humidity PLE=6.2		GSB temperature PLE=4.0		GSB temperature PLE=6.2	
metric	pr	re	pr	re	pr	re	pr	re
RLS fusion	-0.17	9.78	0.40	17.06	2.50	31.66	-6.34	13.64
OS-ELM fusion	0.95	-5.67	-0.55	45.06	11.93	81.31	-12.88	26.51
Ensemble (heuristic)	0.47	0.31	0.07	0.06	-1.94	0.24	-2.71	0.11
Fisher's method	0.29	0.19	0.05	0.18	0.03	0.16	-0.77	0.11
Ensemble (min)	0.64	0.37	-0.21	0.15	-2.10	0.21	-3.65	0.11
Ensemble (median)	1.15	-10.97	5.17	58.40	8.57	55.43	0.46	3.52

Classifier \ Dataset	Intellab PLE=3.4		Intellab PLE=6.2		Rwtb PLE=4.6		Rwtb PLE=6.2	
metric	pr	re	pr	re	pr	re	pr	re
RLS fusion	-2.70	213.58	-2.74	137.45	-4.90	26.93	-6.68	21.82
OS-ELM fusion	-0.49	1557.88	-0.57	1172.76	6.83	86.88	-0.51	42.57
Ensemble (heuristic)	0.28	0.51	0.17	0.27	-1.42	0.78	-1.38	0.59
Fisher's method	0.20	0.37	0.10	0.18	-0.13	0.58	-0.29	0.69
Ensemble (min)	0.32	0.59	0.19	0.29	-1.51	1.11	-1.14	1.33
Ensemble (median)	1.55	3709.42	-0.32	1267.51	0.98	21.22	-1.99	8.52

Noteworthy is also the slight effect that neighborhood information has on the ensemble methods. The ensembles consist of multiple classifiers, of which only two (RLS and OS-ELM fusion) include neighborhood information. The other classifiers are the window constant, detecting if a constant anomaly occurs, and the 1-step-ahead function approximation classifier. From the dataset analysis, we know that 70 to 90% of the anomalies in the datasets is of the 'constant' anomaly type, and from investigations in previous chapters, we es-

tablished that a simple rule-based classifier can detect these anomalies with very high accuracy. Therefore, we hypothesize that only few of the extra recalled anomalies are not of the constant type, and thus the ensembles do not benefit much more from the improved recall.

5.4.3 Change in F-measure

Figure 5.6 shows the relative change in F-measure for the RLS-fusion classifier. With the above settings and the resulting precision and recall numbers presented in Section 5.4.1, we further analyze the effect of neighborhood information on the OS-ELM-fusion based classifier, and the resulting effect on the ensemble classifiers. From this analysis, seen in Table 5.5, we can see that not only for RLS but also for the OS-ELM-based classifier including neighborhood information mostly has a positive benefit. However, the change for OS-ELM is more extreme. This is partly because the F-measure resulting from the anomaly detection without neighborhood information is low, specifically in the case of the Intel Lab dataset, so any change therein is relatively large. Another cause for the higher variability for the OS-ELM based detection is the random initialization of input weights and biases.

Furthermore, from Table 5.5, we can see that the effect of the inclusion of neighborhood information on the ensembles is low. We hypothesize this has two reasons: First, the ensembles consists of a mix of classifiers, which include not only the RLS and OS-ELM fusion classifiers, but also the constant rule classifier and the 1-step-ahead function prediction classifier. The additional neighborhood information only affects the RLS and OS-ELM fusion classifiers and, therefore, the total effect on the ensembles is less. Second, the constant anomaly is the dominant anomaly, covering 70 to 90% of the anomalies in all datasets. Therefore, extra detections of anomalous samples is likely to be a constant anomaly which, thus, will not improve the ensemble score. This is further investigated in the following section, where the classifier agreement is evaluated.

Table 5.5: The fusion classifiers are positively affected by neighborhood information, but the end result on most ensembles in negligible. The table shows relative change in F-Measure, in percent.

classifier \ dataset	GSB Humidity PLE=4.0	GSB Humidity PLE=6.2	GSB Temperature PLE=4.0	GSB Temperature PLE=6.2	Intel Lab PLE=3.4	Intellab PLE=6.2	Indoor WSN PLE=4.6	Rwtb PLE=6.2
RLS fusion	7.61	13.24	25.08	9.32	184.51	122.73	21.57	17.11
OS-ELM fusion	-4.75	39.83	69.08	23.10	1482.04	1131.55	77.98	39.28
Ensemble (heuristic)	0.42	0.07	-0.88	-1.33	0.35	0.21	-0.46	-0.53
Fisher's method	0.25	0.09	0.10	-0.34	0.26	0.14	0.19	0.14
Ensemble (min)	0.54	-0.09	-1.01	-1.88	0.41	0.22	-0.40	-0.09
Ensemble (median)	-9.96	54.06	51.57	3.21	3559.71	1227.27	19.90	8.20

5.5 Classifier agreement

Finally, we evaluate the agreement between classifiers, to get a better understanding why the neighborhood information is of small influence on the ensemble classifiers. The behavior of the median ensemble and the Fisher's method ensemble is that when multiple classifiers agree on a sample being anomalous, the more likely it is to be anomalous. Thus, if more classifiers detect the same sample as anomalous, the better the performance of these ensembles. The minimum p-value and heuristic ensemble operate differently. The former is not influenced by multiple classifiers that judge similarly, but only returns the minimum of the p-values within the ensembled classifiers. Thus, this ensemble would benefit from more confident classifiers. Such an approach should improve recall, but the precision may suffer. The heuristic ensemble combines the constant rule and the RLS-fusion classifier. When a constant is detected by the constant rule, the RLS-fusion classification is ignored. Therefore, when the latter detects a constant anomaly, the performance of the heuristic ensemble is not improved.

The agreement between classifiers is measured as the ratio of equality between two time series of logical values, a and b, divided by the total number of agreed detections possible between a and b, i.e., $sum(and(a,b))/sum(or(a,b))$. The logic values denote the detected or known anomalies in the time series. These ratios can be denoted in a confusion matrix and displayed similarly to the correlation maps in Section 5.2. The resulting matrix is displayed graphically in Figure 5.9. In this figure, the fusion classifiers that include neighborhood aggregate data are indicated with the postfix '.AG', and those that do not include neighborhood information (the stand-alone methods from our previous chapter) with the postfix '.SA'.

Figure 5.9 shows that the constant anomaly agrees well with the original labels and the window mean anomaly detector. The other classifiers do not agree much with the constant classifier, although a slight increase in agreement can be seen between it and the RLS-fusion method that includes neighborhood information. Furthermore, the figure shows that the FA classifier agrees reasonably with the fusion classifiers, which is the result of the FA prediction being included in the input of the fusion classifiers. The fusion methods agree more among each other. Their agreement, however, becomes lower when neighborhood information is included. That would mean that the OS-ELM and RLS fusion detect different (types) of anomalies using neighborhood information.

Overall, the classifiers that include neighborhood information show moderate agreement between themselves and moderate to low agreement to the other classifiers. The effect of neighborhood information on the median and the Fisher's method ensemble should, therefore, be low. The median ensemble, however, does show a reasonable increase in performance in Tables 5.5 and 5.4. But, previous work showed that such an ensemble has high precision but very low recall, and thus improvements in recall are large in relative terms.

The minimum p-value ensemble would benefit only from higher confidence in the detection of a single classifier, which cannot be tested by classifier agreement. The heuristic ensemble should benefit from neighborhood information in the RLS-fusion classifier, if this classifier has little agreement with the constant classifier. However, from Figure 5.9 we see that the RLS-fusion has more agreement with the constant classifier when including neighborhood information, compared to the stand-alone method without neighborhood information. Therefore, their detections may overlap, and the ensemble may not benefit from the neighborhood information. This is in agreement with the earlier evaluation of F-measure, prediction and recall changes in previous sections.

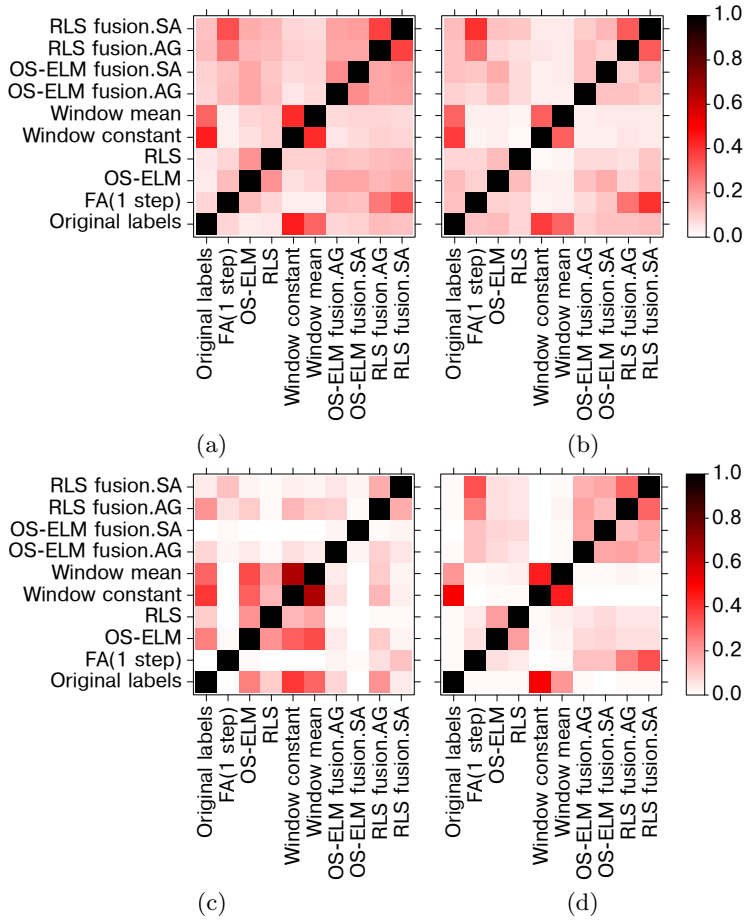


Figure 5.9: Classifier Agreement, sum(and(a, b))/sum(or(a, b)), with PLE=4.0 for the (a) GSB humidity, (b) GSB temperature, (c) Intel Lab, and (d) Indoor WSN datasets

5.6 Conclusions

In this chapter we have empirically demonstrated that incorporating neighborhood information improves the anomaly detection methods presented in previous chapters, yet this is valid only in cases where the dataset is well-correlated and shows relatively low spatial entropy. These assumptions typically occur in the most common application of sensor networks, that of monitoring natural environments. While this brings significant advantages in these cases (because the neighborhood information is correlated), for cases in which these assumptions do not hold there is no immediately benefit from aggregating neighborhood information. That is, our evaluation showed that precision may suffer from including low correlated information. On the other hand, in some scenarios recall may be positively affected. Since communication cost is high, in such cases where the assumptions do not hold it is not always valuable to aggregate neighborhood information locally when using prediction-based anomaly detection.

We explored this hypothesis in several steps. First, we showed that the above assumptions hold only to varying degree through the assessment of correlation in real-world data. Moreover, we showed that the mean and median aggregate operators are valid choices to reduce a dynamic neighborhood to a fixed measure that can be used in fusion methods. Next, we have evaluated the effect of neighborhood density (or communication range) on the quality of the data, by analyzing these effects on the RLS-fusion anomaly detector in simulation, with real-world datasets and topologies. It showed that the amount of improvement mainly depends on the correlation within the dataset. This correlation is the result of the sensed processes, the type of sensors, the type of anomalies, and the topology. Thus, the amount of improvement depends on the application. Nevertheless, the neighborhood information significantly contributed to the anomaly detection recall performance in the well-correlated dataset of the Intel Lab. The other datasets showed sporadic recall improvement even though the neighborhoods were low correlated. On the other hand, precision performance stayed equal or reduced slightly. Again, the exception is the Intel Lab dataset, which also benefited significantly from correlated neighborhoods.

Finally, the analysis of performance at a dense and a sparse network setting showed that adding neighborhood information to the RLS and OS-ELM fusion-based anomaly detectors can be beneficial to the recall, although not significant for the low correlated scenarios. The ensemble methods, however, did not benefit greatly due to additional classifiers that did not make use of neighborhood data, due to the fact that the majority of occurring anomalies is of the constant type. The latter are well detected by a simple rule that does not need neighborhood information.

The overall results show that, when a dataset stems from a similar mixture of diffusive processes (and thus is well-correlated), precision benefits, and a sig-

nificant improvement in terms of recall can be established. However, one has to consider the target application (regarding sensors, anomalies and topology) to evaluate the need for local neighborhood information in online anomaly detection. In cases where a network is too sparse, or in cases where the environment under monitoring has no correlated diffusive processes, a local-only anomaly detection approach may be preferred to spare the limited resources available in an embedded context such as a WSN.

We have made a number of assumptions that may constrain the use of neighborhood information. Future work may investigate if these assumptions can be relaxed, and under what conditions. For example, we assume the network is reliable. This allows timely information sharing with a neighborhood: as the wireless communication is inherently unreliable, missing data may or may not affect the results significantly, depending on the process properties. The slower the dynamics of the process are, the more time can be allowed to share a measurement of that process. Another assumption is that of static node positions. Under the assumption of the environment consisting of a similar mixture of diffusive processes, a mobile node within communication range should still send correlating information. However, future work should investigate to what extent the mobility changes this correlation, given a certain node density.

The assumptions on the process or environment that the WSN will monitor which could be relaxed are that of the measurement period being shorter than the shortest time-constant of the dynamic process, and of the asynchronous occurrence of anomalies. The first allows us to assume that received information is still correlated when it arrives. Depending on the process and anomaly detection method used, future work can investigate the relevance of information when this measurement period is longer than the shortest time-constant of the dynamic process. Moreover, a measure of relevance could be determined with respect to time of reception, depending on the processes under scrutiny. However, it may be that there is a correlation delay due to a slow diffusive process. Future work may investigate methods to automatically determine a delay in correlation.

Other questions that can be addressed are the use of aggregates or models as neighborhood information, instead of raw measurement data and the energy balance between more complex local processing and more decentralized local neighborhood communications. Furthermore, the neighborhood size might be optimized using adaptive transmission power control, or through filtering less correlating neighbors.

Chapter 6

Conclusions

6.1 Findings and contributions

Due to rapid technological advances in the last decade, embedded electronic systems have become ubiquitous in consumer goods and industry. Commonly equipped with sensors, but limited in resources, these systems gather data not only from their state but also from the environment they are in. Moreover, the increasing (wireless) networking possibilities allow this data to be distributed (e.g. to neighboring devices) or be collected (e.g. in 'the cloud'), resulting in the vision of an IoT quickly becoming a reality. However, the envisioned scale and lifetime of the current and future IoT deployments result in challenges in terms of available resources, in terms of management and maintenance, and in terms of data processing. The ever increasing amount of data measured by such systems necessitates new processing and analysis methods, in order to find specific information at specific points in time that are meaningful for the application or domain expert.

Often, data of interest or meaning are out of the ordinary, unexpected, or anomalous. Classical systems use *a priori* knowledge of controlled environments to model and define what information will be "normal" or "anomalous" during deployment. However, IoT systems are commonly placed in unknown environments and generate large streams of data to be analyzed. For example, the mining industry desires to understand the environment (e.g. underground reservoirs) and the effect of mining on the environment (e.g. pollution). This concept is studied by anomaly detection research, which aims to detect events, behaviors or patterns that are unexpected relative to a concept of what is "normal" that is often inferred in a data-driven manner. Anomaly detection, therefore, will play an increasingly important role to (pre-)filter data streams, in order to select actionable or meaningful information.

Related works have shown great interest in anomaly detection through the existence of a plethora of anomaly detection methods in real-world applications, ranging from fraud to earthquake detection. The models are most often

inferred by data-driven methods such as machine learning, pattern recognition or computational intelligence. However, surveying these works in Chapter 2 did show a gap in the literature of anomaly detection methods that are fully decentralized and learn models of normal data online, embedded in an IoT or WSN node. Instead, the common approach is to collect data at a central point with an abundance of resources, where they are analyzed and acted upon. The collection of this data, however, burdens the network and increases response time. Moreover, the envisioned scale and lifetime of IoT systems make long-term central governance difficult. Less common are hybrid systems, which centrally learn models that are then deployed to IoT nodes. They allow more complex models of what is normal, but require extra (network) resources to create and update them. In-node anomaly detection does occur, but is typically hierarchical (requiring a centralized network organization), rule based or static. Online learning anomaly detection would make IoT systems adaptive and independent of central systems. The key challenges to decentralized online-learning in-node anomaly detection are the limited resources (in terms of storage, energy and communication), the large amount of data to be processed, and the lack of *a priori* knowledge of the deployment environment, inherent to the nature of IoT developments. Nevertheless, the advances in technology used for WSN nodes increases the possibility to use complex anomaly detection methods as shown in this thesis.

In this thesis we have addressed these challenges and provided a decentralized online anomaly detection framework for resource-limited embedded systems, such as IoT and WSN. We have shown how incremental learning methods can be analyzed and adapted to limited memory and processing resources of embedded platforms, an approach that can be applied to include other learning methods. This was demonstrated for linear, non-linear and polynomial model learning methods, specifically RLS, OS-ELM and sliding window FA, and combinations thereof in a fusion and ensemble setup. Through an extensive evaluation campaign, consisting of synthetic and real-world scenarios, we have demonstrated that our proposed decentralized approach is feasible and can perform on par with similar offline baseline methods. Moreover, using the scenarios from real-world indoor and outdoor WSN deployments we have shown that anomaly detection in the spatially distributed nature of WSN can benefit from local neighborhood information. However, the latter did require some assumptions of the environment, in particular that it consists of a dynamic mixture of diffusive processes. Although the proposed framework is restricted within the resource limitations that can reasonably be expected from IoT systems, the results have shown a promising anomaly detection performance suitable as a (pre)filter for data streams.

In resource-limited platforms of IoT and WSN systems, not all learning methods are suitable. In Chapter 3 we have outlined our approach of prediction-based anomaly detection and have shown how incremental learning methods are applicable to such platforms. By identifying issues (such as stabil-

ity) caused by limited precision (that can result in buffer under- and overflows during math operations), and proposing solutions for such issues, we have detailed the implementation of RLS, OS-ELM and polynomial FA learning methods. The former two methods capture the linear and non-linear relation between inputs (from measurements or other data) and target measurements in which anomalies are to be detected, while the latter results in a time-series model of a single sensor.

For the evaluation in this thesis we have defined four scenarios based on real-world datasets. Two of which stem from indoor deployments of WSN over several months, being the Intel Lab scenario and the Indoor WSN scenario. The latter was gathered at the beginning of this project. Due to memory limitations for our framework, we have split the SensorScope GSB deployment dataset in two sets of three sensors, that either relate to temperature or humidity. The data of these scenarios have been labeled manually, with the assistance of some rules to automatically flag apparent anomalies such as constants. Next to that, we have used a synthetic dataset with injected anomalies of specific types. Furthermore, the node placement was known, allowing us to fully simulate WSN deployments in these scenarios.

We then extensively evaluated these individual methods. First, for each method parameter settings were suggested that performed well across our datasets, and could serve as guidelines that can be optimized for specific applications. Next, by using synthetic datasets, we have demonstrated the anomaly detection performance per anomaly type in terms of precision and recall. We have shown that online learning methods performed differently per anomaly type. While spike and noise anomalies were detected reasonably well compared to the baseline methods, drift and constant anomalies were poorly detected. For those, the offline baseline methods had the advantage of (a large window of) historic data, while our learning methods adapted to the anomaly. Finally, based on real-world data collected from an indoor WSN, we have tested if this performance held in such a real-world scenario. The majority of occurring anomalies in this scenario is of the constant type (with 70 to 90% of the anomalies), which resulted in relatively poor performance compared to the baselines (especially the rule-based baseline that targets constants). This was, however, anticipated from the results per anomaly type.

To improve the overall performance, we have proposed the use of a combination of classifiers (an obvious addition being the rule to detect constant anomalies). Furthermore, we have analyzed our evaluation methods and have demonstrated that, due to the structure of our anomaly detection approach, an anomaly can influence the prediction for a next sample if undetected. This results in a delayed detection. In a real application, not only the anomalous sample should be sent for further analysis, but also some context, such as a buffer of samples around that detection. Therefore, we have adapted our evaluation metrics to take a small delay into account. In Chapter 4, we have evaluated the combination of classifiers through the fusion of raw mea-

surements and predictions from several models, and through the ensemble of multiple classifiers. While such combinations are commonly used in central systems, the use in resource-limited systems is more challenging. We have demonstrated the fusion approach through the use of earlier investigated RLS and OS-ELM modeling methods. These combined sensor measurements with the predictions from the polynomial FA models and the statistics from a sliding window. The ensemble methods aggregated the p-values of the individual classifiers by using either median (equal to majority voting), mean, minimum or maximum operators. Moreover, Fisher's method to combine p-values was also employed.

The evaluation has shown that, when taking a delayed detection into account, the recall of all methods increased significantly, and in some cases almost doubled. This demonstrated that anomalies are not always directly detected. Nevertheless, the combination of classifiers has also shown a clear improvement. Where the fusion methods mostly improved the results of (correlated) real-world datasets, such as the Intel Lab and Indoor WSN dataset, the minimum p-value, the heuristic and the Fisher's method ensembles greatly improved the anomaly detection performance on all real-world data. However, these methods did have significant false positive ratios, unlike the majority vote (median) ensemble. Since the majority of classifiers must detect an anomaly, it had a consistently high precision, but low recall. Overall, we have demonstrated that ensemble methods are feasible in resource-limited systems and the evaluation shows that classifier ensembles are a valuable addition to the decentralized anomaly detection framework, significantly improving the recall performance.

The typical IoT system consists of a (large) number spatially co-located sensor nodes, likely to measure the same environment. Therefore, in Chapter 5 we aimed to improve the performance of the methods presented in the previous chapters by including combined data from groups of spatially co-located sensor nodes. Therefore, we proposed to collect neighborhood information, by overhearing communication of nearby nodes, assuming that those nodes periodically send (or broadcast) their measurements and assuming the neighborhoods are correlated. The collected data should then be aggregated using the median (majority vote), the mean, the minimum or the maximum operators to reduce the effect of variations in neighborhood size. The aggregated neighborhood information was then used as additional input to the earlier proposed RLS and OS-ELM fusion-based anomaly detectors.

However, the spatial information from neighboring nodes is only correlating under the assumption that the environment in which the nodes are placed consists of a similar mixture of dynamic and diffusive processes. To verify that this assumption holds, we have performed additional characterization on the real-world scenarios used in this thesis by analyzing the cross-correlation of in-node sensor measurements and neighboring-node sensor measurements, and by analyzing the spatial entropy of the datasets. This investigation showed that the SensorScope GSB scenarios had low neighborhood correlation and

relatively high spatial entropy and, thus, did not meet the assumption of a similar mixture of dynamic and diffusive processes. The Indoor WSN dataset, stemming from an indoor deployment, displayed moderate correlation. Only the Intel Lab dataset was highly correlated, and thus met the assumption of similar mixture of dynamic and diffusive processes. Although not ideal, this division in scenarios can also give an indication of how our methods behave when the assumptions are not met.

With this knowledge, we have studied the effect of neighborhood size and the anomaly detection performance when including neighborhood information. The effect of neighborhood size was studied with the fusion-model-based anomaly detectors. The results from a network-average number of neighbors for the moderate and high correlated scenarios suggested a peak F-measure performance when using a small neighborhood. However, evaluating nodes based on exact numbers of neighbors showed that for all but the highly correlated Intel Lab scenario, there was no significant improvement in precision. Moreover, this scenario had significantly improved recall for any number of neighbors. This has shown that, to gain an overall improvement from neighborhood information, the assumption that an environment consists of a similar mixture of diffuse processes must hold.

We then chose an optimal and sub-optimal setting for the network-wide PLE, such that the average number of neighbors for a deployment was optimal or sub-optimal. With these settings, we evaluated the average anomaly detection performance in terms of precision and recall for the fusion- and ensemble-based anomaly detectors (which include neighborhood information). For the fusion-based detectors, again, the precision did not improve, but recall did. Surprisingly, the neighborhood information had little impact on the Fisher's and the minimum p-value ensemble methods. However, the majority voting ensemble benefited in terms of recall. Overall, neighborhood information can have a significant positive impact (mainly on recall), when the assumptions, notably of the environment being a similar mixture of dynamic and diffusive processes, hold. When the assumptions are not met, however, the results were less predictable. Therefore, it is still important to have some knowledge of the targeted deployment environment in order to judge if neighborhood information can contribute.

6.2 Discussion

The current trends show a movement towards cloud computing, where all big data analysis is performed in a central facility with an abundance of resources (the cloud). This allows the offline creation of complex models (which can be distributed to individual IoT nodes) for anomaly detection. Nevertheless, networked embedded systems are fast approaching a point where complete central control/governing (through analysis) cannot be maintained, due to the

large scale of connected devices in the envisioned IoT. Moreover, this strategy has a single point of failure (or attack). Therefore, decentralization strategies are key.

There are hierarchical methods that merge (often cluster) data that is on-route to this central facility which save valuable resources, mainly in terms of communication, but still require central governing. Furthermore, there are few methods that infer online statistics and heuristics to estimate correlations between neighbors, or require an online iterative approach to establish simple local models. To the best of our knowledge, however, there are very few approaches that learn complex models online decentralized (in-node) in order to detect anomalies. The work presented in this thesis may provide a framework for decentralized online-learning anomaly detection for resource-limited embedded devices. However, our methods are not perfect and thus some central orchestration should remain, for instance to provide a second analysis stage. In the following, we discuss the limitations, issues and the unexplored research directions that follow from this thesis.

6.3 Future work

Before we discuss the issues, limitations and assumptions regarding the algorithms, we would like to mention two issues we encountered during this project related to datasets. The first issue is the very **limited availability** of real-world sensor network **datasets** that have **labeled anomalies**. Although the definition of an anomaly allows subjective interpretation, such labels are the main requirement to provide a quantitative evaluation. Works such as [50], therefore, reside to qualitative evaluation of the top k outliers found, by evaluating the results based on interpretation of additional data from the same geographical locations. Therefore, future work should consider indexing and taxonomizing labeled datasets or suggest methods to obtain labels. An approach to obtaining qualitative labels would be to create an ensemble of well-known anomaly detection methods (or a panel of human experts) that can be applied to real-world sensor network data. This would provide a baseline labeling that can be extended with expert knowledge. To the best of our knowledge, such a system does not exist for real-world sensor network data.

The second issue for our project, in relation to datasets, was the lack of known properties of these datasets. Starting researchers should, next to finding them, thoroughly **characterize the datasets**, not only in terms of anomalies, but also in terms of, for example, existing correlations, periodicity and transducer specifications. In our case, the characterization of correlations would have helped us better evaluate early results. Future work may suggest a characterization framework for sensor and sensor network data.

The remainder of this section discusses the issues, limitations and assumptions regarding the algorithms and methods in this thesis. Although still relat-

ing to datasets, the issue of **scaling** data to fit either limited precision buffers or functions over this data applies to algorithms. Related work often assumed that data are normalized before anomaly detection is performed. In the real world, however, such assumptions may not be feasible. Thus, one has to define an expected input range. Due to the inclusion of new methods over the period of our research, new requirements were posed on this input range in order to prevent buffer overflows. However, this down scaling caused buffer underflows in other operations, which resulted in small differences in the results. Thus, future work could investigate for example automated scaling or scale-invariant methods.

Scaling is one of many **preprocessing** methods that can increase the potential of a classifier. Other examples include fast Fourier transform (FFT), independent component analysis (ICA), PCA and PLS, which transform the input data such that features maybe more easily separable by classifiers. However, these often require either large processing capacity (in the case of FFT) or large storage capacity (in the PCA, ICA or PLS transforms), which are not available on the typical low-resource IoT platform. Nowadays, it could be possible to opt for (or create) a platform that includes a dedicated digital signal processor (DSP), for example with FFT methods. Moreover, future work may investigate if methods such as incremental PCA [165] are feasible in resource-limited platforms.

An issue of online incremental learning is the **gradual adaptation to an anomalous state**. If an anomaly occurs for a very long period, or gradually starts to occur (in the case of drift), it may be learned instead of detected. This behavior can be observed throughout the thesis for the synthetic constant and drift datasets. Although the constant anomaly can be detected with a small buffer and a rule (added in the ensembles), the limitations in memory prevent comparison to large buffers of historic data to detect drift. Future work could attempt to find a compression scheme to retain such a buffer, albeit possibly with reduced precision. Another approach that could be investigated is to implement adaptive forgetting factors. This could enable fast learning upon initialization followed by a greater resilience to change in following periods.

Another major limitation of our work is the **assumption** of **correlated data**. The RLS- and OS-ELM-based anomaly detectors assume their inputs (sensor measurements or neighborhood information) to be correlated. The polynomial-FA-based anomaly detector assumes the data to be correlated in time. While these assumptions are probable when a WSN is deployed in a shared environment, with a similar mixture of dynamic and diffusive processes, the analysis in Section 5.2 showed this is not always the case. The aforementioned issue of unavailable labeled datasets prevented a more thorough analysis on the influence of this assumption through the use of more datasets. Moreover, diffusion may be slow, causing a delay in correlation, which needs to be accounted for (and could be the cause of the low correlation seen in Section 5.2). Therefore, future work may investigate methods to automatically deter-

mine this delay in correlation in resource-limited platforms, or explore methods that do not require correlated data. For example, diffusion might be blocked in an office setting with a closed room. Although network nodes may be spatially co-located, such a block may prevent or change a correlation. Future work may also explore ways to weigh or classify neighboring nodes based on the presence or absence of correlation with a local node.

One might also explore **additional anomaly detection methods** that are either generic or target specific anomalies. This creates a more diverse set of anomaly detectors which, in turn, can improve the ensemble results. A starting point could be to investigate the use of RLS to infer ARMA models by including historic data as input (for example from 24 hour periods). A more complex exploration would entail the use of OS-ELM techniques to learn ESN or recurrent neural network (RNN) models, which are closely related to the ELM approach. Other incremental methods, such as clustering or SVM are also good targets for future work.

Moreover, the anomaly detection approach we have taken relies on the analysis of predictions made by the learned models. Therefore, **additional prediction analysis** methods may also benefit the performance. When using formal modeling approaches, an exact threshold can be established based on the model to signal an anomaly. Related work shows that data-driven approaches that lack such knowledge often use statistical tests or heuristics. One could think of using more complex classifiers such as the receptor density algorithm (RDA) [166].

Finally, there are two major open issues regarding the effects of using anomaly detection as a first stage pre-filtering method. First, we did not go into detail on the **effect** of pre-filtering **on the results of a second anomaly detection stage** occurring centrally offline, or in more powerful network nodes. This is a complex question that would require an entirely new research project. In this work, we have assumed that such a pre-filtering would send a context of the detected anomaly, in the form of a window of data around the detection, to a central facility. But, more data could be needed, such as the current state of the learned model, or periodic updates of the data gradient. Moreover, the pre-filtering may require more complex pre-processing to provide workable input for offline anomaly detectors. Thus, future work should study these effects and the trade-off between energy usage, complexity and detection performance.

Second, we did not study the **effect** of such a pre-filtering **on the network resource usage**. While we do assume that:

- WSN use multi-hop communications and, thus, collecting data at a central location burdens the whole network (and nodes close to the sink disproportionately much).
- The energy required to send data over a certain inter-node distance with a certain frequency stays relatively constant although technology advances (that is, one needs a certain wattage to achieve a certain signal strength).

- The energy required for computation decreases as technology advances.

Thus, we deduce that reducing the network traffic reduces the resource usage across the network. Future work should thoroughly study the energy consumption trade-off between in-node analysis and central data collection and analysis.

6.4 Conclusion

Despite these limitations and open ends for future research, this thesis has shown the feasibility of decentralized anomaly detection with a combination of online learning classifiers in resource-limited embedded systems, such as found in IoT and WSN applications. To that end, we have shown how to overcome the issues related to implementing and adapting incremental learning techniques to resource-limited embedded systems, and demonstrated this approach for linear, polynomial and non-linear models. Nevertheless, their individual performance, although promising, was not on par with the baselines. To increase the anomaly detection performance, we have demonstrated that the combination of the individual anomaly detectors in a fusion or ensemble setup significantly and consistently improves performance. Moreover, we have shown that the addition of neighborhood information (commonly available in a IoT or WSN setting) can further improve the real-world results, under the assumption that this information is spatio-temporally correlated, and if the neighborhood is small. The results demonstrate that the proposed online decentralized anomaly detection framework is not only feasible on networked embedded systems but also provides promising performance as a first stage (first responder) anomaly detection and data analysis system for the ever increasing amount of data produced by IoT systems.

Bibliography

[1] A. Whitmore, A. Agarwal, and L. Da Xu, "The internet of things – a survey of topics and trends," *Information Systems Frontiers*, vol. 17, no. 2, pp. 261–274, 2015.

[2] G. Liu, R. Tan, R. Zhou, G. Xing, W.-Z. Song, and J. M. Lees, "Volcanic earthquake timing using wireless sensor networks," in *Proceedings of the 12th international conference on Information processing in sensor networks*. ACM, 2013, pp. 91–102.

[3] A. Alzghoul, M. Lfstrand, and B. Backe, "Data stream forecasting for system fault prediction," *Computers & Industrial Engineering*, vol. 62, no. 4, pp. 972 – 978, 2012.

[4] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2007/2009.

[5] C. Phua, V. Lee, K. Smith, and R. Gayler, "A comprehensive survey of data mining-based fraud detection research," *Arxiv preprint arXiv:1009.6119*, vol. abs/1009.6119, pp. 1–14, 2010.

[6] 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.

[7] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in cloud," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 42–57, 2013.

[8] G. Mulligan, "The 6lowpan architecture," in *Proceedings of the 4th workshop on Embedded networked sensors*, ser. EmNets '07. New York, NY, USA: ACM, 2007, pp. 78–82. Available: http://doi.acm.org/10.1145/1278972.1278992

[9] P. Kinney *et al.*, "Zigbee technology: Wireless control that simply works," in *Communications design conference*, vol. 2, 2003.

[10] G. Fortino, M. Bal, W. Li, and W. Shen, "Collaborative wireless sensor networks: Architectures, algorithms and applications," *Information Fusion*, vol. 22, pp. 1 – 2, 2015.

[11] 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.

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

[13] M. Reyer, S. Hurlebaus, J. Mander, and O. Ozbulut, "Design of a wireless sensor network for structural health monitoring of bridges," in *Wireless Sensor Networks and Ecological Monitoring*, ser. Smart Sensors, Measurement and Instrumentation, S. C. Mukhopadhyay and J.-A. Jiang, Eds. Springer Berlin Heidelberg, 2013, vol. 3, pp. 195–216.

[14] M. Ceriotti, M. Corrà, L. D'Orazio, R. Doriguzzi, D. Facchin, S. Guna, G. P. Jesi, R. L. Cigno, L. Mottola, A. L. Murphy *et al.*, "Is there light at the ends of the tunnel? wireless sensor networks for adaptive lighting in road tunnels," in *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on.* IEEE, 2011, pp. 187–198.

[15] M. Lin, Y. Wu, and I. Wassell, "Wireless sensor network: Water distribution monitoring system," in *Radio and Wireless Symposium, 2008 IEEE*, Jan 2008, pp. 775–778.

[16] H. Alemdar and C. Ersoy, "Wireless sensor networks for healthcare: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2688–2710, 2010.

[17] 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.

[18] R. Jurdak, X. Wang, O. Obst, and P. Valencia, "Wireless sensor network anomalies: Diagnosis and detection strategies," *Intelligence-Based Systems Engineering*, vol. 10, pp. 309–325, 2011.

[19] C. C. Aggarwal, N. Ashish, and A. Sheth, "The internet of things: A survey from the data-centric perspective," in *Managing and mining sensor data.* Springer, 2013, pp. 383–428.

[20] "Argo - part of the integrated global observation strategy." Available: http://www.argo.ucsd.edu

[21] D. H. Roemmich, R. E. Davis, S. C. Riser, W. B. Owens, R. L. Molinari, S. L. Garzoli, and G. C. Johnson, "The argo project. global ocean observations for understanding and prediction of climate variability," DTIC Document, Tech. Rep., 2003.

[22] 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.

[23] M. Xie, S. Han, B. Tian, and S. Parvin, "Anomaly detection in wireless sensor networks: A survey," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1302–1325, 2011, advanced Topics in Cloud Computing. Available: http://www.sciencedirect.com/science/article/pii/S1084804511000580

[24] E. C. Cherry, "Some experiments on the recognition of speech, with one and with two ears," *The Journal of the Acoustical Society of America*, vol. 25(5), pp. 975–979, 1953.

[25] U. Neisser and R. Becklen, "Selective looking: Attending to visually specified events," *Cognitive psychology*, vol. 7, no. 4, pp. 480–494, 1975.

[26] 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.

[27] 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.

[28] Y. Liu, Y. He, M. Li, J. Wang, K. Liu, L. Mo, W. Dong, Z. Yang, M. Xi, J. Zhao *et al.*, "Does wireless sensor network scale? a measurement study on greenorbs," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 873–881.

[29] D. Bucur, G. Iacca, G. Squillero, and A. Tonda, *Applications of Evolutionary Computation: 18th European Conference, EvoApplications 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings*. Cham: Springer International Publishing, 2015, ch. Black Holes and Revelations: Using Evolutionary Algorithms to Uncover Vulnerabilities in Disruption-Tolerant Networks, pp. 29–41.

[30] D. Bucur, G. Iacca, M. Gaudesi, G. Squillero, and A. Tonda, "Optimizing groups of colluding strong attackers in mobile urban communication networks with evolutionary algorithms," *Applied Soft Computing*, vol. 40, pp. 416 – 426, 2016.

[31] 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.

[32] A. B. Sharma, L. Golubchik, and R. Govindan, "Sensor faults: Detection methods and prevalence in real-world datasets," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 3, pp. 23:1–23:39, Jun 2010. Available: http://doi.acm.org/10.1145/1754414.1754419

[33] Y. Yao, A. Sharma, L. Golubchik, and R. Govindan, "Online anomaly detection for sensor systems: A simple and efficient approach," *Performance Evaluation*, vol. 67, no. 11, pp. 1059–1075, 2010.

[34] "Sensorscope project," 2014. Available: http://lcav.epfl.ch/op/edit/sensorscope-en

[35] P. Bodik, W. Hong, C. Guestrin, S. Madden, M. Paskin, R. Thibaux, J. Polastre, and R. Szewczyk, "The intel lab, berkely dataset," 2004. Available: http://db.csail.mit.edu/labdata/labdata.html

[36] A. Zimek, R. J. Campello, and J. Sander, "Ensembles for unsupervised outlier detection: challenges and research questions a position paper," *ACM SIGKDD Explorations Newsletter*, vol. 15, no. 1, pp. 11–22, 2014.

[37] D. M. Powers, "Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation," *Evaluation*, 2007.

[38] Y. Zhang and J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control systems," *Annual Reviews in Control*, vol. 32, no. 2, pp. 229–252, 2008.

[39] S. Budalakoti, A. Srivastava, and M. Otey, "Anomaly detection and diagnosis algorithms for discrete symbol sequences with applications to airline safety," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 39, no. 1, pp. 101–113, 2009.

[40] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *computers & security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[41] K. Tiwari, M. Arora, and D. Singh, "An assessment of independent component analysis for detection of military targets from hyperspectral images," *International Journal of Applied Earth Observation and Geoinformation*, vol. 13, no. 5, pp. 730–740, 2011.

[42] M. Grewal and A. Andrews, "Applications of kalman filtering in aerospace 1960 to the present [historical perspectives]," *Control Systems Magazine, IEEE*, vol. 30, no. 3, pp. 69–78, 2010.

[43] G. Helmer, J. Wong, M. Slagell, V. Honavar, L. Miller, and R. Lutz, "A software fault tree approach to requirements analysis of an intrusion detection system," *Requirements Engineering*, vol. 7, no. 4, pp. 207–220, 2002.

[44] S. Narasimhan and G. Biswas, "Model-based diagnosis of hybrid systems," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 37, no. 3, pp. 348–361, 2007.

[45] A. Giridhar and N. El-Farra, "A unified framework for detection, isolation and compensation of actuator faults in uncertain particulate processes," *Chemical Engineering Science*, vol. 64, no. 12, pp. 2963–2977, 2009.

[46] T. Mitchell, "Machine learning," *Burr Ridge, IL: McGraw Hill*, 1997.

[47] X. Zhu, "Semi-supervised learning literature survey," Computer Sciences, University of Wisconsin-Madison, Tech. Rep. 1530, 2005.

[48] J. Feng, C. Zhang, and P. Hao, "Online learning with self-organizing maps for anomaly detection in crowd scenes," in *2010 International Conference on Pattern Recognition*. IEEE, 2010, pp. 3599–3602.

[49] N. Fanizzi, C. d'Amato, and F. Esposito, "Conceptual clustering and its application to concept drift and novelty detection," in *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*. Springer-Verlag, 2008, pp. 318–332.

[50] E. Schubert, A. Zimek, and H.-P. Kriegel, "Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection," *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 190–237, 2014.

[51] S. Cho and H. Park, "Efficient anomaly detection by modeling privilege flows using hidden markov model," *Computers & Security*, vol. 22, no. 1, pp. 45–55, 2003.

[52] V. Sotiris, P. Tse, and M. Pecht, "Anomaly detection through a bayesian support vector machine," *Reliability, IEEE Transactions on*, vol. 59, no. 2, pp. 277–286, 2010.

[53] L. Zeng, L. Li, L. Duan, K. Lu, Z. Shi, M. Wang, W. Wu, and P. Luo, "Distributed data mining: a survey," *Information Technology and Management*, vol. 13, no. 4, pp. 403–409, 2012.

[54] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[55] K. Langendoen, A. Baggio, and O. Visser, "Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*. IEEE, 2006, pp. 8–pp.

[56] M. Abu Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 4, pp. 1996–2018, Fourthquarter 2014.

[57] A. Sharma, L. Golubchik, and R. Govindan, "On the prevalence of sensor faults in real-world deployments," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON'07. 4th Annual IEEE Communications Society Conference on*. IEEE, 2007, pp. 213–222.

[58] 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*, vol. 20, pp. 272 – 291, 2014.

[59] F. Gaillard, E. Autret, V. Thierry, P. Galaup, C. Coatanoan, and T. Loubrieu, "Quality control of large argo datasets," *Journal of Atmospheric and Oceanic Technology*, vol. 26, no. 2, pp. 337–351, 2009.

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

[61] 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.

[62] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee, "Mcpad: A multiple classifier system for accurate payload-based anomaly detection," *Computer Networks*, vol. 53, no. 6, pp. 864–881, 2009.

[63] 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.

[64] T. A. Nguyen, D. Bucur, M. Aiello, and K. Tei, "Applying time series analysis and neighbourhood voting in a decentralised approach for fault detection and classification in wsns," in *Proceedings of The 4th International Symposium on Information and Communication Technology*. ACM, 2013, pp. 234–241.

[65] H. Luo, J. Luo, and Y. Liu, "Energy efficient routing with adaptive data fusion in sensor networks," in *Proceedings of the 2005 joint workshop on Foundations of mobile computing*. ACM, 2005, pp. 80–88.

[66] J. Cabrera, C. Gutiérrez, and R. Mehra, "Ensemble methods for anomaly detection and distributed intrusion detection in mobile ad-hoc networks," *Information Fusion*, vol. 9, no. 1, pp. 96–119, 2008.

[67] H. Kumarage, I. Khalil, Z. Tari, and A. Zomaya, "Distributed anomaly detection for industrial wireless sensor networks based on fuzzy data modelling," *Journal of Parallel and Distributed Computing*, vol. 73, no. 6, pp. 790–806, 2013.

[68] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. Bezdek, "Distributed anomaly detection in wireless sensor networks," in *Communication systems, 2006. ICCS 2006. 10th IEEE Singapore International Conference on*. IEEE, 2006, pp. 1–5.

[69] G. Fortino, S. Galzarano, R. Gravina, and W. Li, "A framework for collaborative computing and multi-sensor data fusion in body sensor networks," *Information Fusion*, vol. 22, pp. 50 – 70, 2015.

[70] J. Predd, S. Kulkarni, and H. Poor, "Distributed learning in wireless sensor networks," *Signal Processing Magazine, IEEE*, vol. 23, no. 4, pp. 56–69, July 2006.

[71] D. Talia and P. Trunfio, "How distributed data mining tasks can thrive as knowledge services," *Commun. ACM*, vol. 53, no. 7, pp. 132–137, Jul. 2010. Available: http://doi.acm.org/10.1145/1785414.1785451

[72] G. Iacca, "Distributed optimization in wireless sensor networks: an island-model framework," *Soft Computing*, vol. 17, no. 12, pp. 2257–2277, 2013.

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

[74] G. P. Joshi, S. Y. Nam, and S. W. Kim, "Cognitive radio wireless sensor networks: Applications, challenges and research trends," *Sensors*, vol. 13, no. 9, pp. 11 196–11 228, 2013.

[75] M. Paskin, C. Guestrin, and J. McFadden, "A robust architecture for distributed inference in sensor networks," in *Proceedings of the 4th international symposium on Information processing in sensor networks*. IEEE Press, 2005, p. 8.

[76] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, "Distributed regression: an efficient framework for modeling sensor network data," in *Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on*. IEEE, 2004, pp. 1–10.

[77] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Regression in sensor networks: Training distributively with alternating projections," in *Optics & Photonics 2005*. International Society for Optics and Photonics, 2005, pp. 591 006–591 006.

[78] L. Pirmez, F. Delicato, P. Pires, A. Mostardinha, and N. de Rezende, "Applying fuzzy logic for decision-making on wireless sensor networks," in *Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International*, July 2007, pp. 1–6.

[79] T. A. Nguyen, M. Aiello, and K. Tei, "A decentralized scheme for fault detection and classification in wsns," in *The 1st IEEE International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA 2013, Work in Progress session)*, 2013, pp. 1–4.

[80] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, "Distributed sensor fusion using dynamic consensus," in *IFAC World Congress*. Prague Czech Republic, 2005.

[81] T. C. Aysal, M. J. Coates, and M. G. Rabbat, "Distributed average consensus with dithered quantization," *Signal Processing, IEEE Transactions on*, vol. 56, no. 10, pp. 4905–4918, 2008.

[82] H. J. LeBlanc, H. Zhang, S. Sundaram, and X. Koutsoukos, "Consensus of multi-agent networks in the presence of adversaries using only local information," in *Proceedings of the 1st international conference on High Confidence Networked Systems*. ACM, 2012, pp. 1–10.

[83] J. W. Branch, C. Giannella, B. Szymanski, R. Wolff, and H. Kargupta, "In-network outlier detection in wireless sensor networks," *Knowledge and information systems*, vol. 34, no. 1, pp. 23–54, 2013.

[84] K. Flouri, B. Beferull-Lozano, and P. Tsakalides, "Distributed consensus algorithms for svm training in wireless sensor networks," in *Signal Processing Conference, 2008 16th European*, Aug 2008, pp. 1–5.

[85] S. Barbarossa, G. Scutari, and A. Swami, "Achieving consensus in self-organizing wireless sensor networks: The impact of network topology on energy consumption," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 2. IEEE, 2007, pp. II–841.

[86] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM Sigmod Record*, vol. 29, no. 2. ACM, 2000, pp. 93–104.

[87] S. Chawla and P. Sun, "Slom: a new measure for local spatial outliers," *Knowledge and Information Systems*, vol. 9, no. 4, pp. 412–429, 2006.

[88] F. Chen, C.-T. Lu, and A. P. Boedihardjo, "Gls-sod: a generalized local statistical approach for spatial outlier detection," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 1069–1078.

[89] D. McDonald, S. Sanchez, S. Madria, and F. Ercal, "A survey of methods for finding outliers in wireless sensor networks," *Journal of Network and Systems Management*, pp. 1–20, 2013.

[90] N. Chitradevi, V. Palanisamy, K. Baskaran, and K. Swathithya, "Efficient density based techniques for anomalous data detection in wireless sensor networks," *Journal of Applied Science and Engineering*, vol. 16, no. 2, p. 211223, 2013.

[91] L. Xu, Y.-R. Yeh, Y.-J. Lee, and J. Li, "A hierarchical framework using approximated local outlier factor for efficient anomaly detection," *Procedia Computer Science*, vol. 19, pp. 1174–1181, 2013.

[92] W. Wu, X. Cheng, M. Ding, K. Xing, F. Liu, and P. Deng, "Localized outlying and boundary data detection in sensor networks," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 19, no. 8, pp. 1145–1157, 2007.

[93] L. Fang and S. Dobson, "Data collection with in-network fault detection based on spatial correlation," in *Cloud and Autonomic Computing (ICCAC), 2014 International Conference on*. IEEE, 2014, pp. 56–65.

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

[95] W. Barbakh, Y. Wu, and C. Fyfe, "Online clustering algorithms and reinforcement learning," in *Non-Standard Parameter Adaptation for Exploratory Data Analysis*, ser. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2009, vol. 249, pp. 85–108.

[96] P. Wang and T. Wang, "Adaptive routing for sensor networks using reinforcement learning," in *Computer and Information Technology, 2006. CIT'06. The Sixth IEEE International Conference on*. IEEE, 2006, pp. 219–219.

[97] A. M. Zungeru, L.-M. Ang, and K. P. Seng, "Classical and swarm intelligence based routing protocols for wireless sensor networks: A survey and comparison," *Journal of Network and Computer Applications*, vol. 35, no. 5, pp. 1508 – 1536, 2012, service Delivery Management in Broadband Networks.

[98] M. Saleem, G. A. Di Caro, and M. Farooq, "Swarm intelligence based routing protocol for wireless sensor networks: Survey and future directions," *Information Sciences*, vol. 181, no. 20, pp. 4597–4624, 2011.

[99] M. Mihaylov, K. Tuyls, and A. Nowé, "Decentralized learning in wireless sensor networks," *Lecture Notes in Computer Science (Springer Berlin/Heidelberg)*, vol. 5924, pp. 60–73, 2010.

[100] G. D. Fatta, F. Blasa, S. Cafiero, and G. Fortino, "Fault tolerant decentralised k-means clustering for asynchronous large-scale networks," *Journal of Parallel and Distributed Computing*, vol. 73, no. 3, pp. 317–329, 2013, models and Algorithms for High-Performance Distributed Data Mining.

[101] P. Sasikumar and S. Khara, "K-means clustering in wireless sensor networks," in *Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference on*, Nov 2012, pp. 140–144.

[102] Y. Zhang, N. Meratnia, and P. J. Havinga, "Distributed online outlier detection in wireless sensor networks using ellipsoidal support vector machine," *Ad hoc networks*, vol. 11, no. 3, pp. 1062–1074, 2013.

[103] H. Sorenson, "Least-squares estimation: from gauss to kalman," *Spectrum, IEEE*, vol. 7, no. 7, pp. 63–68, 1970.

[104] M. Moreno and J. Navas, "On the robustness of least-squares Monte Carlo (LSm) for pricing American derivatives," *Review of Derivatives Research*, vol. 6, no. 2, pp. 107–128, 2003.

[105] J. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[106] N. Iqevine, "A new technique for increasing the flexibility of recursive least squares data smoothing," *Bell System Technical Journal*, 1961.

[107] K. Åström, "Theory and applications of adaptive control – a survey," *Automatica*, vol. 19, no. 5, pp. 471–486, 1983.

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

[109] S. Challa, F. Leipold, S. Deshpande, and M. Liu, "Simultaneous localization and mapping in wireless sensor networks," in *Intelligent Sensors, Sensor Networks and Information Processing Conference.* IEEE, 2005, pp. 81–87.

[110] X. Xu, H. He, and D. Hu, "Efficient reinforcement learning using recursive least-squares methods," *Journal of Artificial Intelligence Research*, vol. 16, pp. 259–292, 2002.

[111] I. Schizas, G. Mateos, and G. Giannakis, "Consensus-based distributed recursive least-squares estimation using ad hoc wireless sensor networks," in *Signals, Systems and Computers, 2007.* IEEE, 2007, pp. 386–390.

[112] F. Cattivelli, C. Lopes, and A. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *Signal Processing, IEEE Transactions on*, vol. 56, no. 5, pp. 1865–1877, 2008.

[113] 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.

[114] T. Ahmed, M. Coates, and A. Lakhina, "Multivariate online anomaly detection using kernel recursive least squares," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications.* IEEE, 2007, pp. 625–633.

[115] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[116] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural Networks, 1989. IJCNN., International Joint Conference on.* IEEE, 1989, pp. 593–605.

[117] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2. IEEE, 2004, pp. 985–990.

[118] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *Neural Networks, IEEE Transactions on*, vol. 17, no. 4, pp. 879–892, 2006.

[119] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.

[120] S. Ding, H. Zhao, Y. Zhang, X. Xu, and R. Nie, "Extreme learning machine: algorithm, theory and applications," *Artificial Intelligence Review*, vol. 44, no. 1, pp. 103–115, 2013.

[121] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *Neural Networks, IEEE Transactions on*, vol. 17, no. 6, pp. 1411–1423, 2006.

[122] T. Pavlidis, "Waveform segmentation through functional approximation," *IEEE Transactions on Computers*, vol. C-22, no. 7, pp. 689–697, July 1973.

[123] X. Wu, "Adaptive split-and-merge segmentation based on piecewise least-square approximation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 8, pp. 808–815, Aug 1993.

[124] M. Unser, A. Aldroubi, and M. Eden, "Polynomial spline signal approximations: filter design and asymptotic equivalence with shannon's sampling theorem," *IEEE Transactions on Information Theory*, vol. 38, no. 1, pp. 95–103, Jan 1992.

[125] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "Segmenting time series: A survey and novel approach," *Data mining in time series databases*, vol. 57, pp. 1–22, 2004.

[126] W. Philips and G. D. Jonghe, "Data compression of ecg's by high-degree polynomial approximation," *IEEE Transactions on Biomedical Engineering*, vol. 39, no. 4, pp. 330–337, April 1992.

[127] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmaki, and H. T. T. Toivonen, "Time series segmentation for context recognition in mobile devices," in *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, 2001, pp. 203–210.

[128] 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.

[129] M. Amer, M. Goldstein, and S. Abdennadher, "Enhancing one-class support vector machines for unsupervised anomaly detection," in *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*. ACM, 2013, pp. 8–15.

[130] M. Amini, J. Rezaeenour, and E. Hadavandi, "Effective intrusion detection with a neural network ensemble using fuzzy clustering and stacking combination method," *Journal of Computing and Security*, vol. 1, no. 4, 2015.

[131] S. A. Haque, M. Rahman, and S. M. Aziz, "Sensor anomaly detection in wireless sensor networks for healthcare," *Sensors*, vol. 15, no. 4, p. 8764, 2015. Available: http://www.mdpi.com/1424-8220/15/4/8764

[132] R. Fujimaki, Y. Sogawa, and S. Morinaga, "Online heterogeneous mixture modeling with marginal and copula selection," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11. New York, NY, USA: ACM, 2011, pp. 645–653. Available: http://doi.acm.org/10.1145/2020408.2020509

[133] A. Declercq and J. H. Piater, "Online learning of gaussian mixture models-a two-level approach." in *VISAPP (1)*, 2008, pp. 605–611.

[134] D. T. Anderson, O. Sjahputera, K. Stone, and J. M. Keller, "Causal cueing system for above ground anomaly detection of explosive hazards using support vector machine localized by k-nearest neighbor," in *Computational Intelligence for Security and Defence Applications (CISDA), 2012 IEEE Symposium on*, July 2012, pp. 1–8.

[135] R. Rajagopal, X. Nguyen, S. C. Ergen, and P. Varaiya, "Distributed online simultaneous fault detection for multiple sensors," in *Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on*. IEEE, 2008, pp. 133–144.

[136] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos, "Online outlier detection in sensor data using non-parametric models," in *Proceedings of the 32Nd International Conference on Very Large Data Bases*, ser. VLDB '06. VLDB Endowment, 2006, pp. 187–198. Available: http://dl.acm.org/citation.cfm?id=1182635.1164145

[137] N. Shahid, I. H. Naqvi, and S. B. Qaisar, "One-class support vector machines: analysis of outlier detection for wireless sensor networks in harsh environments," *Artificial Intelligence Review*, vol. 43, no. 4, pp. 515–563, 2013. Available: http://dx.doi.org/10.1007/s10462-013-9395-x

[138] Y. Zhang, N. Meratnia, and P. Havinga, "Adaptive and online one-class support vector machine-based outlier detection techniques for wireless sensor networks," in *Advanced Information Networking and Applications Workshops, 2009. WAINA '09. International Conference on*, May 2009, pp. 990–995.

[139] **H. H. W. J. Bosman**, A. Liotta, G. Iacca, and H. J. Wörtche, "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.

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

[141] **H. H. W. J. Bosman**, G. Iacca, H. J. Wörtche, and A. Liotta, "Online fusion of incremental learning for wireless sensor networks," in *Data Mining Workshop (ICDMW), 2014 IEEE International Conference on*, 2014, pp. 525–532.

[142] B. Brewer, "libfixmath - cross platform fixed point maths library," 2012. Available: http://code.google.com/p/libfixmath/

[143] P. Aimonen, "libfixmatrix - c library for fixed point matrix, quaternion and vector calculations," 2012. Available: https://github.com/PetteriAimonen/libfixmatrix

[144] R. Penrose, "A generalized inverse for matrices," in *Proceedings of the Cambridge Philosophical Society*, vol. 51, no. 3. Cambridge Univ Press, 1955, pp. 406–413.

[145] I. M. Grant, "Recursive least squares," *Teaching Statistics*, vol. 9, no. 1, pp. 15–18, 1987.

[146] G. E. Bottomley, "A novel approach for stabilizing recursive least squares filters," *IEEE Transactions on Signal Processing*, vol. 39, no. 8, pp. 1770–1779, 1991.

[147] 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, p. 27.

[148] T. Instruments, "Msp430f15x, msp430f16x, msp430f161x mixed signal microcontroller (rev. g)," March 2011. Available: http://www.ti.com/product/msp430f1611

[149] Y.-H. Pao, G.-H. Park, and D. J. Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, no. 2, pp. 163–180, 1994.

[150] S. Roberts, "Control chart tests based on geometric moving averages," *Technometrics*, vol. 1, no. 3, pp. 239–250, 1959.

[151] **H. H. W. J. Bosman**, G. Iacca, A. Tejada, H. J. Wörtche, and A. Liotta, "Ensembles of incremental learners to detect anomalies in ad hoc sensor networks," *Ad Hoc Networks*, vol. 35, pp. 14 – 36, 2015, special Issue on Big Data Inspired Data Sensing, Processing and Networking Technologies. Available: http://dx.doi.org/10.1016/j.adhoc.2015.07.013

[152] D. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 6–23, 1997.

[153] A. Downs, "An economic theory of political action in a democracy," *The Journal of Political Economy*, vol. 65, pp. 135–150, 1957.

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

[155] **H. H. W. J. Bosman**, G. Iacca, A. Tejada, H. J. Wrtche, and A. Liotta, "Spatial anomaly detection in sensor networks using

neighborhood information," *Information Fusion*, vol. 33, pp. 41 – 56, 2017. Available: http://dx.doi.org/10.1016/j.inffus.2016.04.007

[156] L. G. Birta and G. Arbez, *Modelling and Simulation*, ser. Simulation Foundations, Methods and Applications. Springer, 2013.

[157] S. Basu and N. Kumar, *Modelling and Simulation of Diffusive Processes*, ser. Simulation Foundations, Methods and Applications. Springer, 2014.

[158] K. Romer and B.-C. Renner, "Aggregating sensor data from overlapping multi-hop network neighborhoods: Push or pull?" in *Networked Sensing Systems, 2008. INSS 2008. 5th International Conference on.* IEEE, 2008, pp. 107–110.

[159] J. Lee Rodgers and W. A. Nicewander, "Thirteen ways to look at the correlation coefficient," *The American Statistician*, vol. 42, no. 1, pp. 59–66, 1988.

[160] N. C. Silver and W. P. Dunlap, "Averaging correlation coefficients: should fisher's z transformation be used?" *Journal of Applied Psychology*, vol. 72, no. 1, p. 146, 1987.

[161] M. Batty, R. Morphet, P. Masucci, and K. Stanilov, "Entropy, complexity, and spatial information," *Journal of Geographical Systems*, vol. 16, no. 4, pp. 363–385, 2014.

[162] M. De Berg, M. Van Kreveld, M. Overmars, and O. C. Schwarzkopf, *Computational geometry.* Springer, 2000.

[163] H. Lee, A. Cerpa, and P. Levis, "Improving wireless simulation through noise modeling," in *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, April 2007, pp. 21–30.

[164] F. J. Massey Jr, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[165] A. Bhushan, M. Sharker, and H. Karimi, "Incremental principal component analysis based outlier detection methods for spatiotemporal data streams," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 1, pp. 67–71, 2015.

[166] N. D. Owens, "From biology to algorithms," 2010, phD Thesis.

Curriculum Vitae

Hedde Bosman was born in Groningen, the Netherlands, on November 11, 1984. He received his B.Sc. degree in computer science and M.Sc. degree in computer science in 2007 and 2010, respectively, from the University of Groningen, Netherlands. The topic of his master thesis was "Relating head-movement to sound events" in the Intelligent Systems group. After graduation he joined INCAS^3, researching and developing sensor systems.

In December 2011, he started a PhD project under the supervision of Prof. Antonio Liotta in the Electro-Optical Communications group (ECO) at Eindhoven University of Technology (TU/e), The Netherlands. Here he conducted research work in the area of anomaly detection in networked embedded systems, of which the results are presented in this dissertation. His PhD work has been sponsored by INCAS^3.