ARTIFICIAL INTELLIGENCE BASED EVENT DETECTION IN WIRELESS SENSOR NETWORKS

Majid Bahrepour

ARTIFICIAL INTELLIGENCE BASED EVENT DETECTION IN WIRELESS SENSOR NETWORKS

Majid Bahrepour

Thesis committee members:

Prof. Dr. Ir. Ton Mouthaan	(UT,SECR)
Prof. Dr. Ing. Paul J. M. Havinga	(UT, PS)
Dr. Ir. Nirvana Meratnia	(UT, PS)
Prof. Dr. Hermie Hermens	(UT, BSS)
Prof. Dr. Johann Hurink	(UT, DMMP)
Prof. Dr. Ben Krose	(University of Amsterdam)
Prof. Marimuthu Palaniswami (Palani)	(University of Melbourne, Australia)

The work described in this thesis was performed at the Pervasive Systems (PS) group, Center for Telematics and Information Technology (CTIT), Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS) University of Twente, PO Box 217, 7500 AE Enschede, The Netherlands.

This research is conducted within the EU projects SENSEI and IS-ACTIVE.

Cover design: Majid Bahrepour and Gerrie van Rooijen
ISBN: 978-90-365-3502-1
CTIT PhD thesis series number: 12-241
ISSN: 1381-3617
DOI: 10.3990/1.9789036535021

ARTIFICIAL INTELLIGENCE BASED EVENT DETECTION IN WIRELESS SENSOR NETWORKS

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
Prof.dr. H. Brinksma,
on account of the decision of the graduation committee,
to be publicly defended
on 23rd of January 2013 at 16:45

by

Majid Bahrepour

Born on the 18th of August 1983
in Mashhad, Iran

This dissertation has been approved by:

Promotor: Prof. Dr. Ing. Paul J. M. Havinga

Assistant promotor: Dr. Ir. Nirvana Meratnia

To my parents: Mohammad Mahdi Bahrepour and Zahra Siami

تقدیم به پدر و مادر عزیزم: محمد مهدی بهره پور و زهرا صیامی

Abstract

Wireless sensor networks (WSNs) are composed of large number of small, inexpensive devices, called sensor nodes, which are equipped with sensing, processing, and communication capabilities. While traditional applications of wireless sensor networks focused on periodic monitoring, the focus of more recent applications is on fast and reliable identification of out-of-ordinary situations and events. This new functionality of wireless sensor networks is known as event detection. Due to the fact that collecting all sensor data centrally to perform event detection is inefficient in many occasions, the new trend in event detection in wireless sensor networks is to perform detection in the network.

Design of in-network event detection methods for wireless sensor networks is by no means straightforward, as it needs to efficiently cope with various challenges and concerns including unreliability, heterogeneity, adaptability, and resource constraints. In this thesis, we tackle this problem by proposing fast, accurate, in-network, and intelligent event detection methods using artificial intelligence (AI) and machine learning (ML) approaches. To this end, the main objective of this thesis is to *analyze*, *investigate applicability*, and *optimize* artificial intelligence (AI) and machine learning (ML) methods for efficient, distributed, local and in-network event detection in wireless sensor networks (WSNs). The main contributions of the thesis can be summarized as:

- ***Feature extraction and reduction in three datasets for event detection in WSNs.*** We choose three datasets containing sensory data from real experiments and find their most contributing features by employing expert knowledge and quantitative analysis. These datasets have three levels of difficulties i.e., easy: fire #1 (small number of event sources), medium: fire #2 (medium number of event sources), and complex: activity (highly overlapping dataset for activity recognition). The datasets are used for validation of our proposed event detection techniques throughout the thesis.
- ***Analysis and optimization of three supervised machine learning approaches for event detection within two classification models.*** Machine learning and artificial intelligence approaches are traditionally developed for resource-rich computing devices and not for resource constrained sensor nodes. Using them on sensor nodes, therefore, requires an optimization. To

this end, we optimize three machine learning approaches, i.e., feed forward neural networks, decision tree, and naïve Bayes classifiers to enable them to be executed on sensor nodes. Consequently, we analyze their applicability for fast and accurate event detection using two classification models, i.e, local and fusion-based.

- ***Proposing a new supervised machine learning technique for event detection within two classification models.*** By combining self-organizing maps (SOM) and K-means clustering algorithms, we propose a new machine learning approach, called SOM K-means. Although SOM and K-means algorithms are both unsupervised clustering techniques, the new technique is a supervised learning technique. The SOM K-means is computationally light, resource friendly and capable of being executed in both local and fusion-based classification models.
- ***Analysis and optimization of an unsupervised machine learning approach for event detection.*** To enable in-network unsupervised event detection, we optimize the well-known K-means algorithm to be faster and less resource exhaustive and to be able to detect events without being trained by label data in a training phase.
- ***Analysis of practical implication of the proposed approaches and providing footprints.*** To analyze practical implications of our event detection techniques, we use two concepts, i.e., theoretical analysis and implementation footprints. The theoretical analysis includes provision of time and space complexity calculation in terms of big-O notation, while analysis of implementation footprints includes analysis of code size, execution time, and energy consumption footprints by direct implementation on TelosB sensor nodes.

Samenvatting

Draadloze sensornetwerken bestaan uit grote aantallen sensornodes. Dit zijn kleine, goedkope apparaten die uitgerust zijn met sensoren, rekencapaciteit en een radio voor het zenden en ontvangen van gegevens. Terwijl traditionele toepassingen van draadloze sensornetwerken toegespitst zijn op het controleren van een omgeving door periodieke metingen, ligt de focus van recentere toepassingen op snelle en betrouwbare identificatie van afwijkende situaties en gebeurtenissen. Deze nieuwe functionaliteit van draadloze sensornetwerken wordt beschreven met de term *event detection*. Omdat het verzamelen van alle sensordata voor verdere verwerking op een centraal punt in het netwerk in veel gevallen inefficiënt is, is de nieuwe trend in *event detection* om deze analyse in het netwerk zelf te doen.

Ontwerp van *in-netwerk event detection*-methodes voor draadloze sensornetwerken is niet eenvoudig, aangezien het vraagt om een oplossing die efficiënt omgaat met de verschillende uitdagende kenmerken van draadloze sensornetwerken, zoals onbetrouwbare hardware, heterogeniteit, flexibiliteit en beperkte middelen. In dit proefschrift worden verschillende snelle, nauwkeurige en intelligente *in-netwerk event detection*-methodes voorgesteld als aanpak voor dit probleem. Deze *event detection*-methodes maken gebruik van technieken uit de kunstmatige intelligentie. Het hoofddoel van dit proefschrift is het analyseren, het optimaliseren en het onderzoeken van de toepasbaarheid van deze methodes voor efficiënt en gedistribueerd gebruik in *in-netwerk event detection*-methodes in draadloze sensornetwerken. De belangrijkste bijdragen uit dit proefschrift worden als volgt samengevat:

- **Extractie van kenmerken uit en het reduceren van het aantal kenmerken van datasets voor *event detection* in draadloze sensornetwerken.**
  We kiezen drie datasets die sensordata van experimenten bevatten en vinden de kenmerkende eigenschappen door gebruik te maken van domeinkennis en kwantitatieve analyse. Deze datasets hebben drie moeilijkheidsgraden, namelijk gemakkelijk **(**klein aantal sensoren), gemiddeld (groter aantal sensoren en ingangskenmerken) en moeilijk (dataset met moeilijk scheidbare klassen). De datasets worden gebruikt

voor validatie van *event detection*-technieken die in dit proefschrift geïntroduceerd worden.

- **Analyse en optimalisatie van drie *unsupervised machine learning*-technieken voor event detection binnen twee classificatiemodellen.** Technieken uit de *machine learning* en kunstmatige intelligentie zijn oorspronkelijk ontwikkeld voor computers die veel rekencapaciteit bezitten, en niet voor gelimiteerde apparaten zoals sensornodes. Om deze technieken op sensornodes te gebruiken, is het noodzakelijk om het gebruik van middelen van deze technieken te optimaliseren. We optimaliseren hiervoor drie *machine learning*-technieken, namelijk *feedforward* neurale netwerken, beslisbomen en *naïve Bayes classifiers*. We analyseren de toepasbaarheid van de algoritmes voor snelle en nauwkeurige *event detection*, waarbij gebruik gemaakt wordt van twee classificatiemodellen, namelijk een enkel- en een meervoudig classificatiemodel.
- **Voorstel voor een nieuwe *supervised machine learning*-techniek voor *event detection* binnen twee classificatiemodellen.** We stellen een nieuw machine learning-algoritme genaamd *SOM K-means* voor dat het resultaat is van het combineren van *self-organizing maps* (SOM) en *K-means clustering*-algoritmes. Ondanks dat SOM en K-means beiden *unsupervised* clustertechnieken zijn, is de nieuwe techniek een *supervised* techiek. SOM K-means is licht in termen van rekencapaciteit, is zuinig in het gebruik en kan zowel in enkel- als in meervoudige classificatiemodellen uitgevoerd worden.
- **Analyse en optimalisatie van een machine learning-techniek zonder supervisie voor *event detection.*** Om *unsupervised in-netwerk event detection* te kunnen doen, optimaliseren we het K-means-algoritme, zodat het sneller, en met minder intensief gebruik van middelen, in staat is gebeurtenissen te detecteren zonder getraind te hoeven worden met gelabelde data.
- **Analyse van praktische implicaties van de voorgestelde benadering en het bepalen van de implementatievoetafdruk van de applicatie.** De praktische implicaties van de voorgestelde *event detection*-technieken analyseren we zowel theoretisch als aan de hand van de implementaties. De theoretische analyse bestaat uit een beschrijving van tijd- en geheugencomplexiteit met behulp van de Grote-O-notatie, terwijl de bepaling van de implementatievoetafdruk bestaat uit een meting van de grootte van de programmacode, executietijd en het energieverbruik van implementaties van de algoritmes op TelosB nodes.

Acknowledgment

Give light, and the darkness will disappear of itself.

Desiderius Erasmus (1466-1536) known as Erasmus of Rotterdam, Dutch Renaissance humanist, Catholic priest, social critic, teacher, and theologian.

I would like to thank God who made all the ways possible for me to traverse my journey in life and science. Since I was at fifth grade of elementary school, I had the plan to study computer science. This was because I had a teacher back to elementary school who was a bachelor student himself in applied mathematics, telling us many stories about how computers are capable of solving complex arithmetic and engineering problems in a magical way. My cousins had also a Commodore 64, programming it with Basic. They always let me play some games with their Commodore 64 and meanwhile explained to me how one can write computer programs. Therefore, at the second grade of high school I was determined that I would go to a vocational high school (not the regular high schools in Iran) to study computer science from an earlier age. I remember, since the first year of the vocational high school, I could solve all computer programming problems in our books. Once our teacher, who was a university student himself, gave us one of his own assignments in computer simulation. The problem was simulation of an imaginary factory which could process items in different machines in parallel manner, within certain time ranges and probability of failure (due to some disturbances or maintenance). Then, the problem was studying this factory for specific time period to investigate productivity of this factory. As a second year vocational high school student, I was the only person who could program this problem. Maybe he was the first person who encouraged me not to stop at undergraduate level and continue my education to post-graduation. In last year of my high school, I designed a hardware component and several software components (for different programing languages) to control any external electrical devices through the computer programs I made, while this innovation won one of the best high school innovations at Kharazmi Festival (the most prestigious science festival in Iran). These partial achievements, and some more during my next studies, made me determined to continue my education towards a PhD.

Achieving any success is not done only by myself. Therefore, I would like to firstly thank my parents and my family who were always there for me and supportive all the time. I would thank my father, Mohammad Mahdi, my mother, Zahra, my brother Davoud, and my sister Nafiseh.

I would like to thank all of my teachers (since elementary school) for giving me hope to continue and teaching me how to do my best. Naming all my teachers would require several pages, however, I would like to name a few of them: Mr. Mohammadi, Dr. Yaghoobi, Dr. Toosizadeh, Dr. Naghibi, Prof. Akbarzadeh, Dr. Hamid R. Ekbia, and Dr. Saeid Abrishami.

I would like to appreciate my promoter, Prof. Paul Havinga for employing me and giving me directions during the PhD career. I would like to thank my daily supervisor, Dr. Nirvana Meratnia for her supervisions and her kind attitude.

I would like to acknowledge my committee members Prof. Dr. Ir. Ton Mouthaan, Prof. Dr. Hermie Hermens, Prof. Dr. Johann Hurink, Prof. Dr. Ben Krose, and Prof. Marimuthu Palaniswami (Palani).

I would like to thank two of my best colleagues, Marlies van de Voort and Berend Jan van der Zwaag who helped me during the PhD career and also translated the abstract of my thesis into Dutch.

I would like to appreciate my paranymphs, Jasmien de Vette and Kyle Zhang for helping me during the thesis defense.

I would like to thank many friends of mine around this globe who are in contact with me and are helpful all the time. If I want to name them all, it would need many pages. However, I would name few of them: Mahmoud Ravanan, my best friend here in Netherlands. He is always helpful and effective by giving me advice on my daily matters. Yang Zhang, also a best friend of mine and a colleague at work. Our friendship is beyond the working hours. We lived in the same building, went to badminton together and discussed a lot about our cultures. He was also very effective in my research by helping me to figure out some problems in WSNs. Kyle Zhang, a colleague of mine and a best friend. We went to gym together, talked about various daily matters while he was also very helpful with implementation of my techniques. Amir Latifi, a best friend since high school. Though he lives very far away in Australia but he keeps contact with me and is supportive all the time. I am afraid that this sections is too short to name all my friends, but I would list a couple of more friends: Alireza Masoum, Zahra Taghikhaki, Juan Jimenez Garcia, Ramon

Schwartz, Okan Turkes, Ghazaleh Mobasseri, Arta Dilo, Alireza Rohani, Saeed Sedghi, Behnam Behroozpour, and Amirmehdi Saedi.

I would like to deeply thank my uncle and his wife in Rotterdam, Hamid Siami and Gerarda Schoenmakers, who are always supportive like parents.

I would like to thank my girlfriend, Gerrie van Rooijen, who gave me a new spirit of life and supportive, kind and helpful all the time. Being with her, always brings me peace and joy. She also helped a lot with designing the cover of this thesis. I would also like to thank her family for their kind attitudes.

I would like to thank our secretaries, Nicole Baveld, Thelma Prenger, Marlous Weghorst, being always helpful with problems.

Majid Bahrepour
January 2013
Enschede, the Netherlands

Table of contents

Chapter 1

Introduction

A wise man's question contains half the answer.

Solomon Ibn Gabirol (1021-1058) Andalusian Hebrew poet and Jewish philosopher.

Abstract

Event detection is an emerging functionality of wireless sensor networks . Compared with periodic monitoring, in which data is transmitted even when no change is observed, event detection offers the possibility to be informed when and where events of interest occur. Design of efficient in-network event detection methods for wireless sensor networks is by no means straightforward, as it needs to efficiently cope with various challenges and concerns including unreliability, heterogeneity, adaptability, and resource constraints. This thesis tackles the problem of time critical event detection in wireless sensor networks by proposing fast, accurate, in-network, and intelligent methods using artificial intelligence (AI) and machine learning approaches. This chapter provides details on motivation, challenges and concerns, research objectives and our contributions to the field of event detection in wireless sensor networks.

1.1 Introduction

Wireless sensor networks (WSNs) are composed of large number of small, inexpensive devices, called sensor nodes, which are equipped with sensing, processing, and communication capabilities. WSNs are enabled by the convergence of several technologies such as low-power microprocessors, sensor technology, and low-power RF design [1]. Wireless sensor networks are the next evolutionary development in various applications including industrial automation, asset tracking, health care, and logistics [2]. Applications of WSNs are analogous to any sentient organism in the way that they rely on sensor data being collected from the physical world [2]. Generally speaking, there are three types of data collections in WSNs [3]:

1. *Continuous data collection:* In this type of data collection, sensor data is collected by sensor nodes and transmitted to a central point (also known as a base station) periodically. This method of data collection is simple but does not always work well when data collection rate is high. This is because transmitting all collected data to a central point (i) is usually much slower than data collection, causing low

data resolution, (ii) can cause network traffic, data latency, and packet loss, and (iii) is a highly energy consuming task.

2. *Query-driven:* In this type of data collection, users (or sensor nodes) can make a query to the network, upon which sensor nodes sense and send the required data back to the user (or sensor nodes). Compared with continuous data collection, query-driven data collection is more efficient in terms of energy consumption and network traffic. However, as sensors only report upon receipt of an explicit request from the users or other sensor nodes [4], this type of data collection is not very adequate for applications interested in being notified when events with implicit, complex, or unknown patterns occur.
3. *Event-driven:* In this type of data collection, sensor nodes only report their data and/or trigger an alarm when an out-of-ordinary situation occur. By doing so, compared with continuous and query-driven data collections, this type of data collection not only saves considerable amount of data transmission and consequently energy consumption but is also adequate for applications, whose data exhibit implicit, complex, and unknown patterns.

Data collection is usually followed by data processing to turn the raw data into useful information, upon which actuation and decision making processes can start. Generally speaking, data processing occurs in one of the following locations in WSNs:

- *Base station:* at which, the raw data gathered and transmitted is collected. The base station is usually an interface between wireless sensor network and the outside world and is commonly equipped with stronger computational power and more resources compared with regular sensor nodes. Data processing at the base station benefits from having more memory, processing, and energy sources but may suffer from problems related to data transmissions such as network traffic, latency and packet loss.
- *In-network:* in which, raw data is processed on sensor nodes within the network rather than in a base station. By doing so, network traffic, latency and possibility of packet lost are reduced thanks to prevention of transmission of massive amount of sensor data to the base station. Generally speaking, in-network data processing can be conducted in two manners:
  - *Local:* in which, only one sensor node is involved in the data processing task.
  - *Distributed:* in which, a group of sensor nodes are involved in the data processing task. Consequently, sensor nodes should exchange some data.

The base station data processing has similar properties as ‘continuous’ data collection, because in both methods raw data needs first to be transferred to a central point (base station). Although query-driven and event-driven applications were traditionally based on data processing at the base station, the new developments in these applications are moving

towards in-network data processing. Therefore, in this thesis we are, particularly, interested in event-driven data collection which process data in-network both locally and distributedly. *Event detection* is another term for event-driven data collection. Event detection is an emerging functionality of wireless sensor networks, which compared with periodic monitoring offers the possibility to be informed when and where events of interest occur. By doing so, it had become an essential functionality in many WSN applications such as volcano monitoring [5], healthcare [6], fire detection [7], and disaster management [8]. We will delve more into details on event detection later in Chapter 2.

The rest of this chapter is organized as follows. Section 1.2 shortly introduces wireless sensor networks and their characteristics. Data processing and event detection are discussed in details in Section 1.3. Artificial intelligence and opportunities it offers for data processing in wireless sensor networks are described in Section 1.4. Section 1.5 discusses the research question we address in this thesis, while contributions of this thesis are listed in Section 1.6. Lastly, Section 1.7 discusses organization of the thesis.

1.2 Wireless sensor networks

A wireless sensor network (WSN) is composed of a number of sensor nodes communicating wirelessly and forming a network [9]. Each node has commonly the following components [10]: (i) a processor as a microcontroller or CPU, (ii) multiple types of sensors, (iii) RF transceiver with mostly one single omnidirectional antenna, (iv) a power source (usually batteries), (v) memory, and (vi) storage. It can also accommodates different types actuators. Figure 1.1 shows how a sensor node typically looks like.

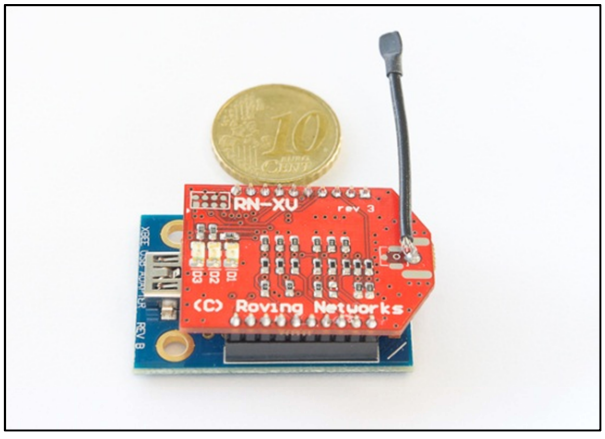


Figure 1.1: A typical sensor node.

WSNs become known as the new frontier in gathering and processing data from remote locations [11]. It is foreseen that in 10-15 years from now, the world will be completely covered with WSNs while they are accessible via the Internet [10]. Then, WSNs can be

considered as the Internet's physical body. WSNs enable numerous applications such as environmental monitoring, medical and healthcare, and disaster management.

1.3 Data processing and event detection in WSNs

As mentioned in Section 1.1, event detection is a significant functionality of WSN, as it can efficiently manage sensor data and alarm when events of interest occur. Generally, event detection should be conducted on time, within a certain (pre-specified) timeframe. However, the time restriction of event detection is application dependent. Generally speaking, time constraints of event detection applications may be of time critical type or best effort type.

- *Time critical:* in which, a certain timeframe is defined, within which events should be detected. In time critical event detection applications, detecting events after the predefined timeframe may have calamitous consequences. Examples of this type of time critical event detection applications are fire, flood, or any other disastrous events detection.
- *Best effort:* in which, a certain timeframe is defined within which the event detection technique makes it best effort to detect events. In the 'best effort' event detection, delay in event detection does not usually have catastrophic consequences. An example of best effort event detection applications is detecting whether there is a 'meeting' in a conference room or detection of leakage in a water pipe at home.

We have already defined event detection in WSNs and discussed some features of event detection (e.g., time criticality), however, designing efficient in-network event detection approaches for wireless sensor networks is by no means straightforward and poses variety of challenges [12] including dealing with:

- *Unreliability of sensor data:* sensor data is well-known to be erroneous and unreliable. Sources of the unreliability include (i) harsh environmental conditions (e.g., high/low temperature, humidity), (ii) limited power source, (iii) wireless communication, and (iv) cheap hardware components. Environmental conditions (such as temperature and relative humidity) can affect operation of sensor nodes and consequently lead to generation of faulty measurements. This type of unreliability is usually inevitable as wireless sensor networks are designed to be usually deployed in harsh and unattended environments. Batteries are the usual power source of sensor nodes. Energy drain of sensor nodes can cause unreliability in sensor node operation and generation of erroneous data. The known fact of unreliability of wireless communication on the one hand, and the cheap sensors used on wireless sensor nodes to keep the price low on the other hand, are two other reasons for generation of erroneous sensor data. A possible solution to deal

with unreliability of sensor data is to include more sensor nodes in the process of event detection. In this case, if one sensor node fails or generates unreliable and faulty data, other sensor nodes can contribute to the event detection process and compensate the failure.

- *Volume of data:* if event detection is conducted in a base station, it needs to deal with large volume of data as all sensor nodes send their sensor data for event detection to a central point. If event detection is conducted in the network and on sensor nodes, it needs to deal with smaller amount of data, however this data contains less information about out-of-ordinary situations, which makes the processes of event detection more difficult.
- *Complex event patterns:* events usually exhibit a pattern, which is different from normal data pattern. Event patterns may be simply definable (e.g., if temperature is greater than 300ºC then there is a fire), complex, or even unknown. In the latter cases, data modeling, pattern recognition, and classification techniques are needed to extract event patterns hidden in sensor data.
- *Heterogeneity and dynamicity of network:* sensor networks are considered heterogeneous as they may have different (i) type of sensor nodes possessing different capabilities and resource constraints and (ii) type of sensors. Environmental disturbances, mobility, and energy exhaustion are a few reasons causing network dynamicity. An efficient event detection approach should be able to handle both heterogeneity and dynamicity well.
- *Adaptability:* an event detection approach should be adaptable to (i) various type of events and (ii) context. Adaptability to different event types means that an event detection approach should ideally be able to detect more than one event type. For example, an event detection detecting fires should also be able to detect floods to some extent. Adaptability to different context means that event detection approach needs to cope with the change of deployment setup and location. A possible solution to adaptability is to use proper type of algorithms that support adaptability and only requires minor parameter changes (e.g., changing weights).
- *Locality of reference:* events are local in terms of spatial location. This indicates that an event only needs to be detected by sensor nodes experiencing the event and being the neighborhood of event and not the entire network. Having locality of reference makes the event detection approaches independent of network scale because as the networks grows, the event detection still remains local among a group of sensor nodes. However, the challenge here is to deal with common data between sensor nodes which usually has a high degree of redundancy. The possible solution to locality of reference is a fusion-based distributed approach, in a way that each sensor node detects events alone (or locally) and send its results of event detection to another sensor node for reaching a consensus.

Having mentioned the challenges of in-network event detection in wireless sensor networks, there are also some concerns that should also be taken into account, when such event detection approach is designed. These concerns include:

- *Limitation in processing power of sensor nodes*: processors in sensor nodes are commonly limited to usually a microcontroller. Such a tiny processor is unable to execute complex algorithms. Consequently, event detection algorithms should be computationally inexpensive.
- *Limitation in communication:* data transmission in wireless sensor networks is much more energy consuming than local data processing. Therefore, event detection approaches should rely more on local data processing and less on data exchange. If necessary to communicate, data communication needs to be kept at minimal level.
- *Real-time decision making:* as mentioned earlier, in this thesis we focus on time critical event detection applications. Therefore, despite of all limitations, event detection technique should be responsive to events in a timely manner.
- *Accuracy:* event detection needs to be accurate to reduce false alarms. A possible solution for accurate event detection is distributed data processing which breaks the data processing task into several parts such that every part is processed by an individual sensor node. Another possible solution is to use artificial intelligence (AI) approaches. AI algorithms usually lead to higher accuracy because they mimic natural process of thinking and decision making [13-16] (we will delve more into this topic in Section 1.4).

This thesis tackles the problem of time critical event detection in WSNs by proposing fast, in-network, intelligent methods taken from artificial intelligence (AI). The motivation behind choosing AI techniques is that they are accurate and adaptable to multiple situations. They also have the ability to learn from data, which makes them promising candidates for data management and event detection in WSNs. We will delve into more details on promises of AI in WSNs in the next section.

1.4 Artificial intelligence and wireless sensor networks

Artificial intelligence (AI) is "the science and engineering of making intelligent machines, especially intelligent computer programs" [9]. Artificial intelligence is the fundamental technology of today's applications ranging from banking (in application such as fraud detection in credit card systems, or portfolio selection for investing on financial projects) to telephone systems (application such as speech detection to give appropriate piece of advice) [17]. Although there exist a number of dedicated AI applications such as industrial robots, or the INTELLIPATH [18] (pathology diagnosis system approved by the American Medical Association and deployed in hundreds of hospitals worldwide), AI algorithms are also widely used in more general purpose applications, databases, and environments to

make them friendlier, smarter, and more sensitive to user behavior and environmental changes.

Artificial intelligence is an important topic because it not only has practical applications, but also it touches the fundamental question of computer science: "Can a machine ever be made to actually think?" [19, 20]. As an ultimate goal, artificial intelligence seeks the ability towards a fully automate human thinking and reasoning [19, 20].

AI has already shown great promises in various WSN applications such as network routing [12, 21-24], anomaly detection [25-28], and event detection [7, 8, 29-34]. Therefore, integration of AI and WSNs is receiving more attention nowadays to make WSNs more intelligent, robust, adaptable, and accurate.

Machine Learning (ML), is a branch of artificial intelligence that takes empirical data (such as data from sensors) as input and generates a pattern [35]. A major focus of machine learning research is the design of algorithms that recognize complex patterns and make intelligent decisions based on input data [35, 36].

For the purpose of event detection in WSNs in presence of unreliable and heterogeneous sensor data, we utilize ML and AI techniques to process data and detect event patterns. Since ML is a branch of AI, sometime ML and AI are used interchangeably in this thesis. Chapter 2 of this thesis covers more details on the motivation of using AI and ML techniques in our event detection process.

1.5 Research objective

Challenges and concerns faced while designing an efficient event detection approach for WSNs mostly stem from (i) resources limitation (i.e, in terms of processing power, memory, and power), and (ii) application requirements that require event detection to be fast, accurate, and adaptable. Consequently, a possible solution to address these challenges and fulfill application requirements can be artificial intelligence based event detection relying on in-network data processing. To this end, our research objective is:

> *To analysis, investigate applicability, and optimize artificial intelligence (AI) methods for efficient, in-network local and distributed event detection in wireless sensor networks (WSNs).*

In the research objective, the term 'efficiency' relates to addressing limitations of WSNs and fulfilling requirements of time critical event detection applications in terms of detection time (in seconds) and detection accuracy. We approach the problem with proposing AI algorithms that are executed in local and fusion-based distributed fashions. In the local approach, events are detected on sensor nodes, while in fusion-based distributed approach several sensor nodes executing local approaches in a spatially correlated area detect events

distributedly inside the network. The local approach is a foundation for the fusion-based distributed approach because the distributed fusion-based approach consists of local approaches. The fusion-based distributed event detection benefits from distributed systems properties such as fault tolerability and higher detection accuracy. In the rest of this thesis we use the term 'fusion-based' as a shorter term for 'fusion-based distributed'. To gauge performance of the proposed approaches, we use Matlab® simulation and implementation on real sensor nodes. Several metrics such as time and space complexity (in terms of big-O notation) as well as execution time and energy consumption on real sensor nodes will be reported. These metrics give more insight about efficiency of the proposed approaches.

1.6 Thesis contributions

To achieve the research goals of this thesis, we provide the following contributions:

1- ***Feature extraction and reduction in three datasets for event detection in WSNs.*** We choose three datasets containing sensory data from real experiments and find their most contributing features by employing expert knowledge and quantitative analysis. These datasets have three levels of difficulties i.e., easy: fire #1 (small number of event sources), medium: fire #2 (medium number of event sources), and complex: activity (highly overlapping dataset for activity recognition). The datasets are used for validation of our proposed event detection techniques throughout the thesis.

2- ***Analysis and optimization of three supervised machine learning approaches for event detection within two classification models.*** Machine learning and artificial intelligence approaches are traditionally developed for resource-rich computing devices and not for resource constrained sensor nodes. Using them on sensor nodes, therefore, requires an optimization. To this end, we optimize three machine learning approaches, i.e., feed forward neural networks, decision tree, and naïve Bayes classifiers to enable them to be executed on sensor nodes. Consequently, we analyze their applicability for fast and accurate event detection using two classification models, i.e, local and fusion-based.

3- ***Proposing a new supervised machine learning technique for event detection within two classification models.*** By combining self-organizing maps (SOM) and K-means clustering algorithms, we propose a new machine learning approach, called SOM K-means. Although SOM and K-means algorithms are both unsupervised clustering techniques, the new technique is a supervised learning technique. The SOM K-means is computationally light, resource friendly and capable of being executed in both local and fusion-based classification models.

4- ***Analysis and optimization of an unsupervised machine learning approach for event detection.*** To enable in-network unsupervised event detection, we optimize the well-known K-means algorithm to be faster and less resource exhaustive and to be able to detect events without being trained by label data in a training phase.

5- ***Analysis of practical implication of the proposed approaches and providing footprints.*** To analyze practical implications of our event detection techniques, we use two concepts, i.e., theoretical analysis and implementation footprints. The theoretical analysis includes provision of time and space complexity calculation in terms of big-O notation, while analysis of implementation footprints includes analysis of code size, execution time, and energy consumption footprints by direct implementation on TelosB sensor nodes.

1.7 Thesis organization

The organization of the thesis is shown in Figure 1.2, which demonstrates the information flow of the main research topics, the contributions and relationship among the chapters of the thesis. Chapter 2 provides a technique-based taxonomy of current state-of-the-art on event detection techniques used in WSNs. It also proposes a new direction for event detection studies. Chapter 3 introduces and analyzes the datasets used in this thesis for evaluation purposes. Chapter 4 presents our proposed supervised event detection approaches while Chapter 5 presents our proposed unsupervised event detection approaches for WSNs. Chapter 6 reports on practical implications of the proposed approaches, while finally Chapter 7 concludes the thesis and highlights the lesson learned.

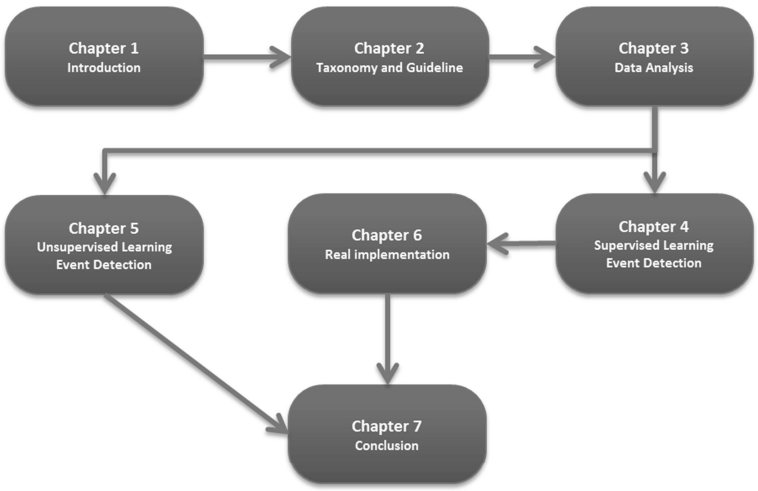


Figure 1.2: Organization of the thesis.

Chapter 2*

Literature Survey and Taxonomy

The knowledge of anything, since all things have causes, is not acquired or complete unless it is known by its causes.

Ibn Sina (Avicenna) (980-1037) Persian polymath.

Abstract:

Recently, WSN community has witnessed an application focus shift. Although, monitoring was the initial application of wireless sensor networks, in-network data processing and (near) real-time actuation capability have made wireless sensor networks suitable candidate for event detection and alarming applications as well. Event detection is a new topic in WSNs, and therefore, there are not many studies specifically focused on event detection as a standalone topic. In this chapter, we first define what event detection is, then report and classify previous studies which fit in our event detection definition based on their underlying techniques, and discuss their important features. At the end of this chapter, we analyze previous works to understand their shortcomings and propose a guideline and future direction for event detection studies.

2.1 Introduction

An event, according to definitions, is a 'considerable' phenomenon that happens, or may have happened [37-39]. Event detection is the process of detection of such a considerable phenomenon. this thesis, we define events as rare out-of-ordinary occasions that exhibit different patterns than normal patterns of data. Among all applications of event detection, our focus is on event detection in wireless sensor networks. Event detection in wireless sensor networks is an emerging domain and as such it is not well explored and not many related work exist. This chapter presents a survey and taxonomy of existing event detection approaches designed for WNSs.

Event detection functionality of WSNs has become an essential element for wide variety of applications ranging from malfunctioning of a monitored machinery to biomedical health monitoring. Instead of acquiring a complete view of sensory data over time and space,

* This chapter is partly published with the title, "Automatic Fire Detection: A Survey from Wireless Sensor Network Perspective". Technical Report TR-CTIT-08-73, 2008 [116].

event detection functionality aims to indicate occurrence, probability of occurrence and/or type of events that have already happened or are about to happen.

Event detection in WSNs is a rather new field of research and as such not many related work can be found. The initial and simple idea of event detection is to use a threshold value, using which an event is detected if value of sensor data exceeds the pre-defined threshold. For example, 'temperature >30' may indicate a fire event. The problem with threshold-based event detection approaches is that they are not adequate for complex situations. Therefore, the new trend in accurate event detection is to detect events in the network using pattern recognition techniques, as generally events exhibit a pattern [8, 40-43].

Designing event detection approaches for wireless sensor networks is by no means straightforward, because sensor nodes are limited in their computational power, memory, communication range, bandwidth and power source. Therefore, an ideal event detection technique for WSNs needs to be energy efficient, fault tolerant and robust, resource friendly, and adaptive to multiple event types and environment while it still can provide fast and accurate event detection.

In basic event detection approaches, sensor nodes collect data and transfer them to a resource-rich central point for further analysis and event detection. The problem with processing the data centrally is the delay in event detection and low reliability caused by network traffic, congestion, and packet loss [44]. In the new trend of in-network event detection, data is processed by sensor nodes in the network. In this way, sensor nodes immediately process data and if it is necessary (i.e. an event has happened or is about to happen) sensor nodes trigger an alarm to the outside world to inform about the event [45, 46].

In this chapter, we first review a number of existing event detection techniques designed for wireless sensor networks and present in Section 2.2 a taxonomy based on the techniques utilized by them. Event detection techniques have a number of distinguishing features, which need to be considered when studying them. These features, presented and discussed in Section 2.3, are used to compare existing event detection techniques later on in Section 2.4 in order to identify a set of guidelines for designing optimal event detection techniques in WSNs.

2.2 Taxonomy of event detection techniques for WSNs

We choose to categorize existing event detection approaches based on their underlying techniques. As it can be seen in Figure 2.1, existing event detection techniques can be classified into threshold-based, model-based, pattern matching-based, and AI-based techniques. Models-based techniques may be based on arithmetic models, statistical models or maps. Pattern matching-based techniques can be further classified into signature

matching and prototype matching. AI-based techniques can be divided into supervised learning and unsupervised learning event detection approaches.

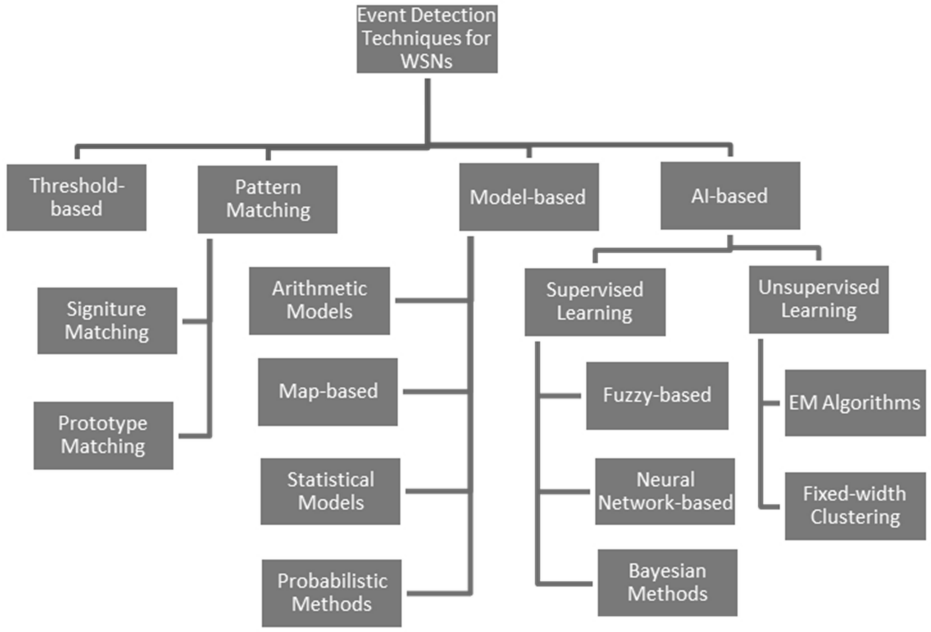


Figure 2.1: Taxonomy of event detection techniques for WSNs.

2.2.1 Threshold-based event detection techniques

Threshold-based event detection techniques in general use some if-then-else rules to check whether the sensor values exceed specific predefined thresholds. Threshold-based event detection approaches have the following properties: (i) are simple to implement, as they are some if-then else rules, (ii) require an expert knowledge for setting the right threshold values, and (iii) may not be accurate in complex situations as they are simple and model events as linear functions.

C. T. Vu et al. [47] propose a threshold-based composite event detection technique, in which an event is decomposed into a number of sub-events. An event is detected if all sub-events occur (either simultaneously or sequentially). For example, an explosion event may be composed of two sub-events, i.e., (i) a loud noise and (ii) heat. Therefore, explosion event is detected if the two sub-events happen simultaneously. This study features distributed processing for large scale networks and is evaluated in a simulation environment. A threshold-based distributed event detection scheme for heterogeneous sensor networks is proposed in [48]. The authors propose to aggregate data in two levels, i.e., at parental nodes and at the sink in order to detect events. The study proposes a

complete framework for collaborative event detection and the approach was implemented by COMiS middleware [49]. A consensus-based threshold-based event detection scheme for volcano monitoring is presented in [5]. The authors propose a complete framework for monitoring volcano activities and then implement the approach on TMote Sky sensor node [50]. The implemented approach was installed and evaluated for 19 days on Reventador, i.e., an active volcano in Ecuador.

2.2.2 Model-based event detection

Model-based event detection techniques are techniques that model event phenomena in a specific form such as mathematic formulas or maps. These models are usually proposed for specific applications. Model based event detection approaches have the following properties (i) can handle complex events, as they mainly represent a non-linear model of events (ii) are typically application dependent and are inflexible towards other application areas, (iii) need an expert to tune the parameter of the models, and (iv) are commonly computationally intensive due to complexity of their models.

2.2.2.1 Arithmetic model-based techniques

Arithmetic model-based techniques utilize discrete or continuous mathematical models to model events and to detect them according to how well data fits their predefined models.

M. Bager et al. [51] propose a voting graph neuron (VGN) algorithm to detect events distributedly in large-scaled sensor networks. VGN algorithm is based on the distributed cooperative problem solving concept that solves a problem by breaking it into smaller parts. In this approach, event patterns are stored in a distributed graph over the network and then events are detected by matching sensory data of each sensor node with a subset of the graph. Their proposed approach was tested on how well pattern matching of shapes can be performed in Matlab simulation environment. C. Zhang et al. [52] propose two novel arithmetic model-based techniques to detect events locally by a sensor node. The techniques rely on if-then-else decision making based on arithmetic operations that actually check the data against event models to determine whether an event has happened. The authors validate their event detection approaches in Castalia simulator [53] to detect “over heat” events and compare their results with pure threshold-based event detection.

2.2.2.2 Map-based techniques

The concept behind map-based studies is that maps are prominent representation of the physical world and its physical phenomena over space and time. Therefore, a map can be a representative of the events as well and map processing can assist detecting incidents being happening or have happened in an environment.

Khelil et al. [54] propose a distributed event detection technique based on MWM (Map-based World Model), in which each sensor node creates a map of its environment and sends this model to a sink for detection of environmental events. Authors show that the proposed approach is able to detect events in a timely manner. M. Li et al. [55] propose a distributed event detection approach based on creation of a 3D map and an aggregation method. The authors argue that sensor nodes are located in a 3D environment and therefore can model the 3D environment in form of a cubic map cell. These cubic map cells can then be aggregated in a cluster head or sink to make a bigger cubic map cell of the entire monitored surroundings. At the final stage of event detection process, the map from the whole environment represents the event and event location. Isolines [56] is a map-based aggregation technique for distributed event detection that takes advantage of the spatial correlation of data. The approach groups sensor nodes that report similar readings into some clusters (so called isocluster) represented by an area of the same value (color) on a map. Maps are further aggregated to reach a consensus whether an event has occurred. The authors evaluate their technique using synthetic data in a simulation environment.

2.2.2.3 Statistical model-based techniques

Statistical model-based methods look into data distribution or other statistical indices to model and then detect events.

NED (Noise-Tolerant Event and Event Boundary Detection) [57] is an event detection scheme that distributedly detect both events and event boundaries in WSNs. The approach uses a moving average technique for noise effect reduction and a statistical method for event and event boundary detection. The approach is evaluated in Matlab simulation environment. An efficient in-network event detection algorithm is proposed in [58], which uses statistical indices to detect events on the sensor nodes. Then, the result of event detection of each individual sensor node is sent to the sink (or the cluster head) to make the final judgment on occurrence of events. The authors show their approach is accurate, error tolerant, and energy efficient in a simulation environment.

2.2.2.4 Probabilistic model-based techniques

Probabilistic model-based techniques utilize probability theory to model events and to detect them if their occurrence is probable .

STED (Spatio-Temporal Event Detection) [59] is an in-network real-time event detection scheme that detects events using a belief propagation technique. The approach is evaluated by being implemented on TMote Sky sensor nodes in a small scale network as well as in a simulation environment for a large scale network.

2.2.3 Pattern matching-based event detection

Pattern matching-based event detection methods define a data pattern for events and then an event is detected if data pattern matches with event pattern. Pattern matching techniques use some functions to investigate data pattern or trend and then decide about occurrence of events. These techniques have the following properties: (i) usually can handle complex events as they look for data pattern using a non-linear function (ii) are flexible and adaptable to various applications, (iii) need an expert knowledge to tune the right parameters of the approaches, and (iv) are mostly computationally intensive due to complexity of the approach.

2.2.3.1 Prototype matching approaches

A prototype is usually a vector of numbers that are derived from features (or sensors). Therefore, event data produce a different prototype than non-event data. Comparing and matching data prototypes leads to detection of out of ordinary instances (events).

In [60] authors propose a distributed approach to detect events by making and comparing the premade prototypes. In this study the event prototypes are made during a learning phase and then by fusing individual sensor nodes decisions, events are detected. The approach is implemented on ScatterWeb MSB-430 sensor nodes for a fence monitoring application.

2.2.3.2 Signature matching techniques

Signature of data can be created by converting data into a feature shrank domain. By having event signatures, event data can then be discriminated from normal non-event data. The difference between prototype matching and signature matching methods is mainly in use of data conversion (in signature matching methods), that transforms data into other domains such as frequency domain or symbolic domain.

In [61] authors propose a local event detection scheme that uses principal component analysis (PCA) to get the signature of events. They then use a threshold to separate event data from non-event data. Their approach is evaluated on MicaZ sensor nodes. M. Zoumboulakis et al. [62] define complex events as “sets of data points that constitute a pattern” and then detect events using symbolic aggregate approximation (SAX) transformation. The idea is to transform data in symbolic space and then to find the shortest distance between previously stored event patterns and incoming data to determine whether incoming data is an event. The approach is a local approach and the authors evaluate the technique in Matlab simulation. Another signature matching technique is proposed by F. Martincic et al. that partition the whole network into cells and detect events by comparing the cell’s signature with event’s signature [63]. Event signatures are 2D matrices storing special-related values and event detection is the process of comparing the sensor node’s data with event signatures. This approach is distributed and is evaluated in a TinyOS

simulator environment using 2 × 2 and 5 × 5 sensor network topologies. Patrec [64] is a signature matching event detection approach that extracts features from sensor nodes and considers them as signatures. It employs different algorithms for distribution, classification, and fusion of "fingerprints" or "signatures" of raw data sampled on each sensor. Euclidean distance is used to match signatures. The shortest distance between an unclassified data and a stored signature represents the maximum similarity. The results from sensor nodes are sent to a base station for fusion. Authors propose various algorithms for combining decision of different nodes. This approach is quantitatively tested in a small-scaled network on geometric shape data.

2.2.4 AI-based techniques

AI-based event detection techniques use artificial intelligence-based methods (also known as machine learning (ML)) to detect events. AI-based techniques are generally classified into two classes, i.e., supervised and unsupervised. Supervised learning techniques use labeled data for training, while unsupervised learning techniques require no labeled data or a priori knowledge. AI-based methods are also pattern matching approaches as they look for data pattern or trend usually using a nonlinear function. The most important advantage of AI-based techniques is that they do not require expert knowledge to set the parameters of the approaches. Therefore, an AI-based technique attunes its parameter by learning automatically from data. These techniques are, in general, less computational intensive than model-based (specially statistical) approaches [158]. Since events generally exhibit a pattern [8, 40, 41], learning-based techniques appear to be quite promising for accurate event detection in WSNs as they can learn data pattern without human intervention. The new trend in data driven decision making systems (e.g., event detection in WSNs) is, therefore, using of AI-based approaches [65].

2.2.4.1 Supervised learning

Supervised learning techniques strongly rely on presence of labeled data and require a training phase.

2.2.4.1.1 Fuzzy-based techniques

EDA (Event-oriented Data Aggregation) is a distributed fuzzy-based event detection approach that uses fuzzy engines to detect events [66]. This approach was implemented on TelosB sensor nodes in an offshore test bed for ocean surveillance application. An event detection approach using fuzzy credibility for distributed in-network event detection is proposed by T. Kieu Xuan et al. [67]. The approach clusters the entire sensor network and gives a credibility value to each cluster. The credibility value then represents the possibility that an event occurs within each cluster. FED [68] is another event detection approach that groups sensor nodes into clusters and detects events distributedly within each cluster. Each sensor node detect events using a threshold-based approach to detect events locally. Then

the results are sent to the cluster head, in which a fuzzy-based classifier is executed. This approach targets both homogeneous and heterogeneous WSNs. A local event detection scheme based on fuzzy logic decision making is presented in [69]. In this study, authors investigate applicability of fuzzy logic for residential fire detection. The approach is mainly designed to be executed on individual sensor nodes requiring no communication between sensor nodes.

2.2.4.1.2 Neural network-based techniques

A distributed neural network based event detection approach is presented in [70]. This study targets forest fire detection in WSNs by processing event detection to be performed at the sensor node and cluster head both. On the sensor nodes, a threshold-based event detection approach is executed, while in the cluster head a neural network based event detection algorithm makes the final decision about occurrence of forest fires. This approach is simulated for both small and large scale networks.

2.2.4.1.3 Bayesian methods techniques

An event detection technique using distributed Bayesian algorithm is presented in [71]. The authors state that there are situations, in which faulty sensor nodes may exhibit unusual patterns. Occurrence of these patterns is considered as an event. This approach is similar to outlier detection in WSNs. It is mainly designed for large-scale networks and is tested in an simulation environment. M. Moradi et al. [72] present a fusion-based event detection method based on Bayesian approach, in which a Kalman estimator is used to estimate missing data. Events are then detected distributedly by Bayesian detection scheme. The authors evaluate their approach in a simulation environment.

2.2.4.2 Unsupervised learning

Unlike supervised learning techniques, unsupervised learning techniques do not require labeled data, training phase, and a priori knowledge about event patterns.

2.2.4.2.1 Fuzzy adaptive resonance theory (Fuzzy ART)

Adaptive Resonance Theory (ART) is a class of neural networks that performs clustering task [73]. Fuzzy ART is integration of fuzzy logic into ART algorithm in order to augment generalizability of ART algorithm. In [74] a fuzzy ART algorithm to detect unusual events is proposed. The idea is to cluster data and those clusters, which have minimal population are considered as events.

2.2.4.2.1 Fixed-width clustering

In [75] authors propose a fixed-width clustering technique for intrusion detection. The idea of fixed width clustering is to cluster training data into a dynamic number of clusters in the

training phase. Then, based on the population of clusters, a decision is made on which cluster belongs to the intrusions. After the training phase, the data which is closer to intrusion cluster is reported as intrusion attacks. Another fixed-width clustering technique is proposed in [76]. The authors propose that single nodes cluster data and then in a sink (or cluster head) these clusters get merged together to detect anomalous data. The approach is implemented in C++ on Great Duck Island data (humidity, temperature and pressure sensor data) for detecting anomalous data.

2.2.4.3 Other AI techniques potentially suitable for event detection in WSNs

In the previous subsections, we reviewed AI-based event detection techniques used in existing studies. However, as one may notice, there are a number of other AI techniques, which potentially can be used for event detection in WSNs and have not yet been explored. These techniques include Averaged one-dependence estimators (AODE) [77], Artificial neural network [78], Naive Bayes classifier [79], Bayesian network [80], Bayesian knowledge base [81], Case-based reasoning [82], Decision trees [83], Inductive logic programming [84], Gaussian process regression [85, 86], Group method of data handling (GMDH) [87], Learning Automata [88], Learning Vector Quantization [89, 90], Lazy learning [91], Instance-based learning [92], Nearest Neighbor Algorithm [93], Analogical modeling [94], Symbolic machine learning algorithms [95], Support vector machines [96], Random Forests [97], Ordinal classification [98], Regression analysis [99], and Information fuzzy networks (IFN) [100, 101] for supervised learning based event detection and K-means [102], Self-organizing map [103], Radial basis function network [104], Vector Quantization [105], Generative topographic map [106], and Information bottleneck method [107] for unsupervised learning event detection in WSNs. In this thesis, we explore applicability of a number of these techniques.

2.3 Important features of event detection techniques

To make appropriate choices for designing event detection techniques for WSNs, a number of issues need to be considered. The main issues relate to the location(s) where event detection technique is executed, i.e., distributed vs. centralized, real-timeness, scale, and type of sensor data used for the process of event detection, i.e., heterogeneous vs. homogenous (Figure 2.2). Since event detection techniques vary in the way they address these issues, we use these very features later on to compare the existing solutions.

2.3.1 Event detection processing model

Event detection processing model implies how sensor data used by event detection technique is processed and at which location the event detection technique is executed. Early studies of event detection in WSNs assume sensor data is collected in a central location and events are detected offline. Coming about of more real-time applications of WSNs has invalidated this assumption. Our focus, therefore, is not on this type of

centralized event detection techniques. Two other processing models that have recently been developed and used in WSN-based event detection techniques are (i) local and (ii) distributed processing models.

A local processing model is a model employed by each and every individual sensor node. This implies that local processing models do not accommodate communication between sensor nodes. In this model, the sensor node is on its own to detect events without getting information from other nodes.

In contrary, a distributed processing model is a model, in which events are detected through information exchange between sensor nodes. The main idea of distributed processing models is to conduct event detection process in the network by engaging more than one sensor node.

We will show in Chapter 4 and Chapter 5 that although distributed processing models may be more attractive, local processing models may be more suitable for small scale and static (relatively unchanged) networks.

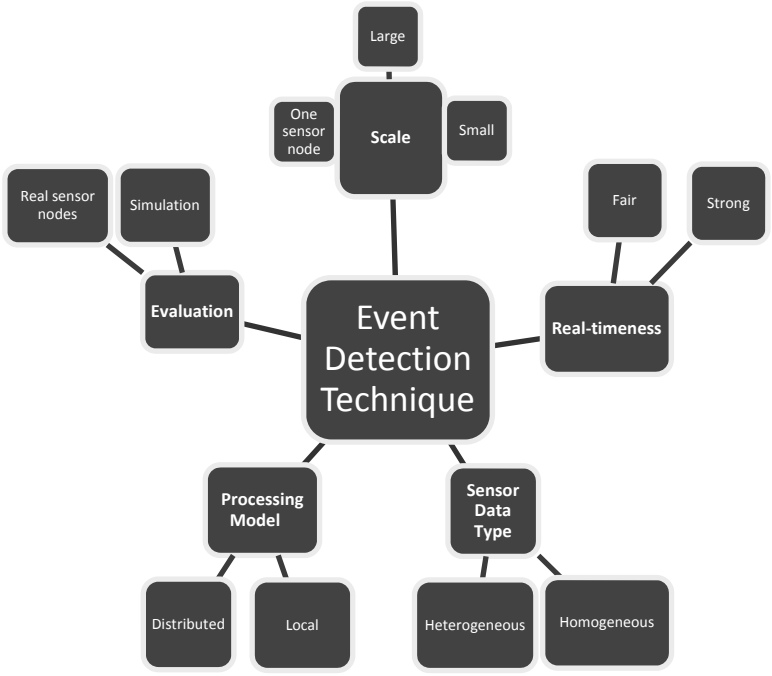


Figure 2.2: Main distinguishing features of event detection techniques.

2.3.2 Scale

Existing event detection techniques address various network scales, i.e., (i) large scale, in which up to thousands sensor nodes are used for event detection, (ii) small scale, in which a

handful of sensor nodes participate in the event detection process, and (iii) a single sensor node, in which the entire event detection process is performed only by one sensor node.

2.3.3 Sensor data

Events may be detected based on homogenous or heterogeneous sensor data. Designing event detection techniques capable of handling heterogeneous sensor data is challenging, as availability of more sensor data type increases data dimensionality and requires more sophisticated event detection approaches.

2.3.4 Real-timeness

Real-timeness is a must in event detection approaches because events should be detected on time. However, some event detection techniques are stronger in their real-timeness guarantees than others by providing a faster event detection.

2.3.5 Evaluations of event detection technique

How event detection techniques are evaluated is quite important because it gives an insight about feasibility of their implementation on wireless sensor nodes and consequently their real applicability. Some studies report on implementation of their approach on sensor nodes to show that their techniques are certainly feasible to be executed on sensor nodes. There are very few studies that calculate computational cost and space complexity of their techniques. Computation and space complexity calculation indicate how much resources are required for execution of the event detection approach. Majority of existing studies, however, rely on simulation results only and do not provide any real measurements and indications on feasibility of implementation and real application of the proposed approach.

2.3.6 Density

Density is the number of sensor nodes in a region [44]. Although the network density has an effect on quality of event detection, this parameter is usually ignored in event detection studies. Authors usually discuss network scale but not network density. As the density is not a factor that is commonly discussed is the previous works, we do not consider this parameter further in this chapter.

2.4 Comparison of event detection techniques for WSNs and guidelines

Having discussed important features of event detection approaches, in this section we summarize previous studies based on their underlying techniques and their important features. Table 2.1 provides a comparative view on how previously mentioned event detection techniques address the aforementioned important issues. We use this table to highlight the shortcomings of the exiting techniques and to identify a set of guidelines for designing optimal event detection techniques for WSNs.

Table 2.1: A comparative view of existing event detection approaches designed for WSNs.

	Technique	Real-timeness		Process model		Evaluation		Scale			Network type	
		Strong	Fair	Distributed	Local	Simulation	Implementation	Large	One sensor	Small	Homogenous	Heterogeneous
[47]	Threshold-base	✓		✓		✓		✓				
[48]	Threshold-base	✓		✓		✓						✓
[5]	Threshold-based	✓		✓			✓					
[51]	Model-based	✓		✓		✓		✓				
[64]	Model-based	✓			✓	✓			✓			
[54]	Model-based		✓	✓		✓						✓
[55]	Model-based		✓	✓		✓		✓		✓		
[56]	Model-based		✓	✓		✓		✓				
[57]	Model-based		✓	✓		✓						
[58]	Model-based		✓	✓		✓		✓		✓		
[59]	Model-based	✓		✓		✓	✓	✓		✓		
[60]	Pattern-matching	✓		✓			✓	✓		✓		
[61]	Pattern-matching	✓			✓		✓		✓			
[62]	Pattern-matching		✓		✓	✓	✓		✓			
[63]	Pattern-matching		✓	✓		✓		✓		✓		
[64]	Pattern-matching	✓		✓			✓			✓		
[66]	AI-based		✓	✓			✓	✓				
[67]	AI-based	✓		✓		✓		✓				
[68]	AI-based	✓		✓		✓		✓				✓
[69]	AI-based	✓			✓	✓			✓			
[70]	AI-based	✓		✓		✓		✓		✓		
[71]	AI-based			✓		✓		✓				
[72]	AI-based		✓	✓		✓		✓		✓		
[74]	AI-based		✓	✓		✓		✓		✓		
[75]	AI-based	✓		✓		✓		✓		✓		
[76]	AI-based		✓	✓		✓					✓	

Table 2.2 presents a compact form of Table 2.1.

Table 2.2: Summary of event detection studies.

Technique	Real-timeness		Process model		Evaluation		Scale			Network type	
	Strong	Fair	Distributed	Local	Simulation	Implementation	Large	One sensor	Small	Homogenous	Heterogeneous
Threshold-base	✓		✓		✓		✓			✓	
Model-based		✓	✓		✓		✓			✓	
Pattern-matching	✓		✓			✓			✓	✓	
AI-based	✓		✓		✓		✓		✓	✓	

Considering Table 2.1 and Table 2.2, we can generally summarize the shortcomings of the existing techniques as the following:

- Event detection in WSNs is a rather new field of research and as such not many related work can be found.
- Very few studies implement their approach on real sensor nodes. Authors mostly simulated their approaches in simulation environments (e.g., MATLAB).
- Those studies which use simulation environments mostly provide no indication (e.g. time and space complexity calculation) about feasibility and applicability of their approach in real scenarios. To prove appropriateness of event detection approaches for sensor nodes, some metrics are required to give insights about resource intensity of the techniques. The metrics can be big-O calculation for simulated studies, and time and energy consumption for real implementations. However, majority of existing studies do not present such metrics.
- AI-based techniques are the most adaptable to network scales. It can be concluded from Table 2.2 that AI-based techniques are able to be executed in both large and small scale networks.
- As discussed earlier in Subsection 2.2.4, AI-based (or machine learning-based) techniques appear to be the most promising techniques for adaptable and accurate event detection. However, there are still not enough studies focusing on machine learning techniques for event detection. There are more AI techniques (listed in Subsection 2.2.4.3) that can potentially be utilized for various event detection situations.
- Although unsupervised learning approaches do not inheritably need training phase, the unsupervised learning approaches, proposed till now, still require a training phase [74-76].
- There are very few event detection methods designed for heterogeneous networks. Previous studies mainly address homogenous networks.
- In most of the previous studies the event detection technique itself is not the main concern of the study. These papers present a complete framework for event detection, which makes the event detection method less significant.
- Existing techniques are proposed for specific network size and scale and they are mostly inadaptable to other scales. This statement can be concluded from Table 2.2 that threshold-based, model-based and pattern-matching approaches are mostly designed for a specific network scale. Among different categories, AI-based techniques are shown to be more adaptable to various network sizes. This also indicates significance of AI-based approaches in adaptability.

All existing works have considered real-timeness as an important issue since an event detection approach should be real-time to detect events promptly.

Considering the aforementioned shortcomings, design of event detection methods can be improved in the following directions:

- Having discussed advantages of AI-based event detection, studies should be more towards AI-driven techniques.
- There is a need for studies which are mainly focused on event detection technique itself to address explicit requirements of event detection in WSNs such as accuracy, adaptability, and heterogeneity.
- Since WSNs are becoming more and more heterogeneous, event detection techniques need to better support heterogeneous sensor data.
- Event detection techniques designed for WSNs need to be robust to cope with inherent failure of sensors and network.
- Since events are not always known beforehand, machine learning techniques and in particular unsupervised learning approaches need to be developed, which do not require labeled data and can cope with unseen events.
- Event detection techniques designed for WSNs need to be computationally cheap. To this end, computational cost and space complexity of every approach should be calculated to give an indication about feasibility and applicability of the approach.
- Event detection techniques need to be adaptable to various network scales, context, and event types.
- Although useful, performance evaluation of event detection techniques for WSNs in simulation environment is not enough and these techniques need to be gauged against realistic assumptions and settings. To this end, they need to be implemented and evaluated on sensor nodes in a reasonable size network.

2.5 Conclusion

In this chapter we reviewed event detection techniques specifically designed for WSNs. We then proposed a taxonomy on the basis of the underlying event detection techniques used by the reviewed techniques and discussed their important features. We also presented guidelines and directions to follow for further development of event detection techniques in WSNs. We consider shortcomings of existing event detection techniques and the proposed guidelines to propose new event detection solutions, which meet WSNs specific requirements and advance the state of the art. In this section we conclude that event detection studies should direct more towards AI-based techniques. This is because AI-based techniques provide great promises for accurate event detection while are adaptable to various network scales, less computational intensive, and can handle heterogeneity of WSNs.

Chapter 3[†]

Data Analysis

All statements are true, if you are free to redefine their terms.

Thomas Sowell (1930) American economist, social theorist and political philosopher.

Abstract:

Data is an important ingredient of event detection process, since it is used on the one hand in the training phase and on the other hand in performance evaluation. In this chapter, we introduce and analyze three datasets containing sensory data from real experiments and find their most contributing features by employing expert knowledge and quantitative analysis. These datasets have three levels of difficulties i.e., easy: fire #1 (small number of event sources), medium: fire #2 (medium number of event sources), and complex: activity (highly overlapping dataset for activity recognition). In the next chapters these datasets will be used to evaluate our proposed event detection approaches.

3.1 Introduction

This chapter introduces and analyzes three sensor datasets that are used later in this thesis to evaluate our proposed event detection techniques. Two of these datasets are fire datasets (fire #1 and fire #2) and one of the datasets is an activity dataset. The motivation behind choosing these datasets are as follows:

- these datasets are from real experiments;
- the datasets exhibit three levels of difficulties. Fire #1 dataset is an easy dataset and contains small number of event sources. Fire #2 is semi-difficult dataset and contains medium number of event sources. Activity dataset is a complex dataset and contains highly overlapping features.

In this chapter, we aim to analyze these datasets to get an insight about their complexity and its effect on classification (detection) accuracy. Later on in this chapter, we also discuss

[††] This chapter is partially published with the title: "Fire data analysis and feature reduction using computational intelligence methods". In: Advances in Intelligent Decision Technologies - Proceedings of the Second KES International Symposium IDT 2010, 28-30 July 2010, Baltimore, Maryland, USA [108].

how these datasets are pre-processed using a feature selection procedure to determine which features contribute to the process of event detection most.

The rest of this chapter is organized as follows. Sections 3.2, 3.3, and 3.4 discuss and analyze fire #1, fire #2, and activity datasets, respectively. In Section 3.5, we discuss the feature selection procedure for all three datasets. Final conclusions are drawn in Section 3.6.

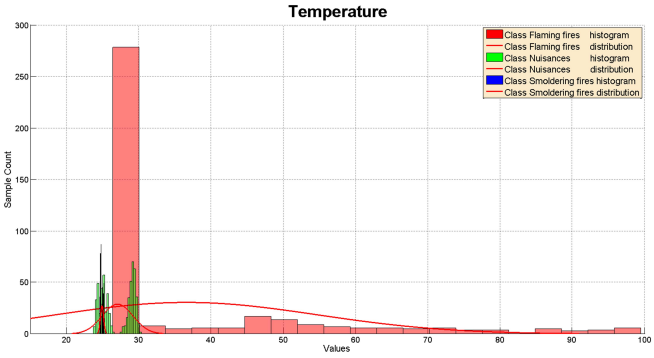
3.2 Fire #1 dataset

Fire #1 dataset is real dataset containing three types of fires, i.e., flaming fires (fires that produce massive heat), smoldering fires (fires that produce massive smoke) and nuisances (fire-like incidents such as smoking a cigarette or toasting a bread) [108]. The dataset is obtained from the National Institute of Standard and Technology (NIST) website [109] by merging two smoldering fire datasets (SDC31, SDC40), two flaming fire datasets (SDC10, SDC14) and two nuisance resource datasets (MHN06, MHN16). The dataset contains four features (sensor data types), i.e., Temperature (C), Ionization (1/m), Photoelectric (1/m) and CO (carbon monoxide, ppm) . The total number of data instances in this dataset is 1400. Table 3.1 shows data range of all features of fire #1 dataset.

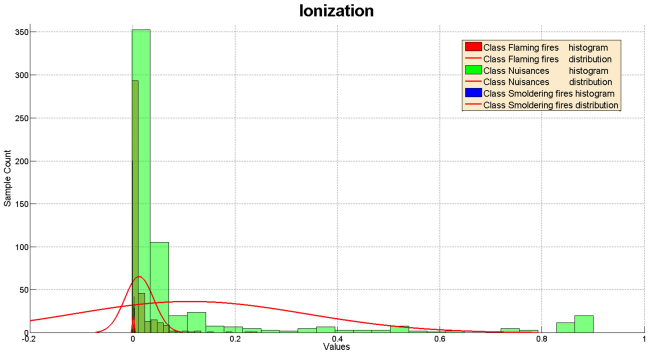
Table 3.1: Data range of features of fire #1 dataset.

	Temperature (C)			Ionization (1/m)			Photoelectric (1/m)			CO (ppm)		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
Flaming class	26.400	36.583	99.400	-0.000	0.0132	0.2434	-0.000	0.0124	0.0499	-5.590	68.539	351.0
Smoldering class	24.600	24.917	25.400	0.0000	0.0017	0.0044	-0.004	-0.001	0.0008	-25.40	-23.09	-22.20
Nuisance class	23.700	27.062	30.371	-0.001	0.1122	0.8990	-0.067	-0.001	0.231	-2.000	0.613	36.00
Overall	23.7	29.169	99.400	-0.001	0.0523	0.8990	-0.067	0.0025	0.2310	-25.4	13.245	351.0

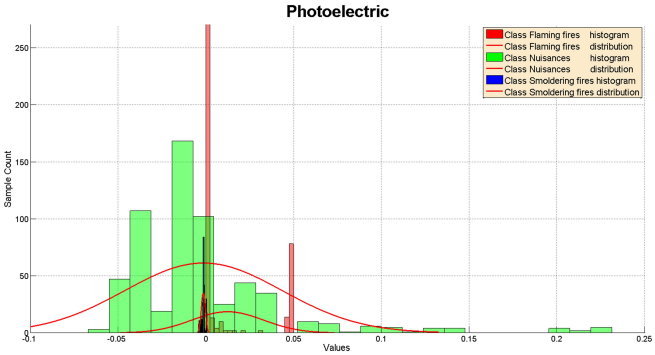
Based on Table 3.1, one can conclude that Temperature (TMP) and CO value ranges in flaming fire datasets are larger than smoldering and nuisances datasets. Additionally, Ionization and Photoelectric ranges for nuisance dataset is larger than the rest. Data distribution per feature for all three event (class) types, i.e., flaming fire, smoldering fire, and nuisance, of this dataset is depicted in Figure 3.1.


Temperature
Class Flaming fires histogram
Class Flaming fires distribution
Class Nuisances histogram
Class Nuisances distribution
Class Smoldering fires histogram
Class Smoldering fires distribution
Sample Count
Values

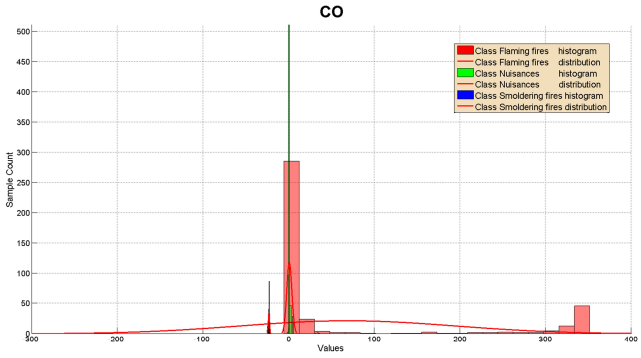
(a)


Ionization
Class Flaming fires histogram
Class Flaming fires distribution
Class Nuisances histogram
Class Nuisances distribution
Class Smoldering fires histogram
Class Smoldering fires distribution
Sample Count
Values

(b)


Photoelectric
Class Flaming fires histogram
Class Flaming fires distribution
Class Nuisances histogram
Class Nuisances distribution
Class Smoldering fires histogram
Class Smoldering fires distribution
Sample Count
Values

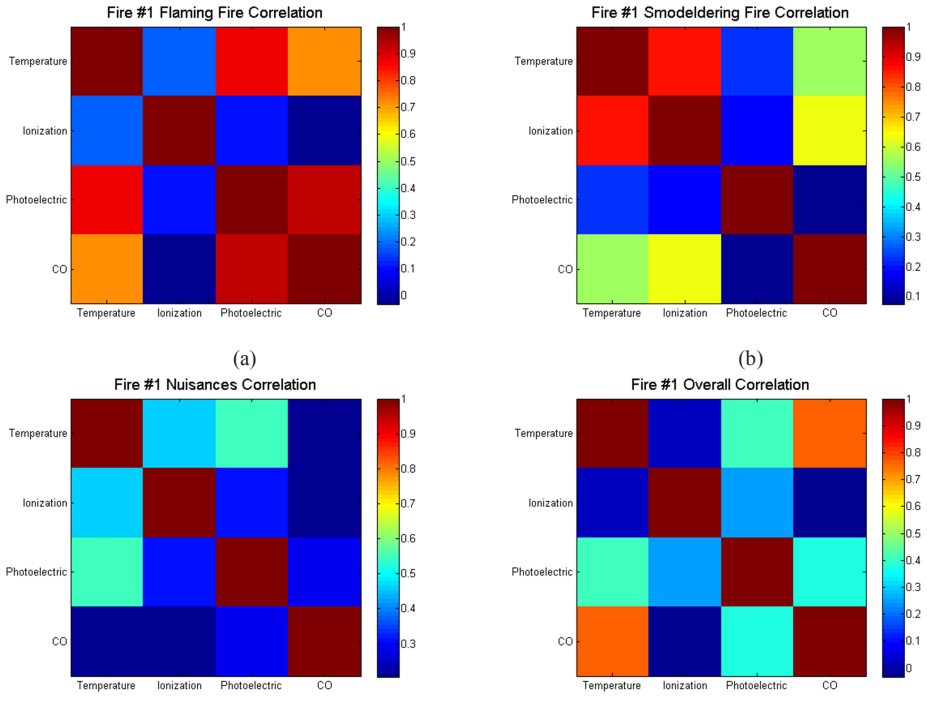
(c)



(d)

Figure 3.1: Data distribution for fire #1 dataset for (a) Temperature, (b) Ionization, (c) Photoelectric, and (d) CO.

It can be seen from Figure 3.1 that Temperature and CO have a wider distributions for flaming fire, while Ionization and Photoelectric have wider distributions for nuisances. Generally speaking, wider data distribution leads to lesser data overlap. Therefore, classification in a dataset with wider distribution is easier and more accurate.



(a) (b)

(c) (d)

Figure 3.2: Data correlation in fire #1 dataset for (a) Flaming fire class, (b) Smoldering fire class, (c) Nuisances class, and (d) Overall correlation between data in all classes.

Figure 3.2 shows data correlation of fire #1 dataset as well as the overall correlation. It can be seen that in case of flaming fire, all features, except CO and Ionization, are correlated. In case of smoldering fire, all features are correlated, apart from Photoelectric and CO. In case of nuisances, all features are correlated except CO, Temperature and Ionization. It can be seen from the overall correlation that Ionization has almost no correlation with CO and Temperature, while the other three features are correlated.

To investigate the contribution of each feature to the classification (event detection) process, that is how well an event can be detected using a single sensor data type, we consider all possible combinations of these four features and run our classifiers (the classifiers are introduced and discussed in Chapter 4) for 1000 times and report mean and standard deviations in Table 3.2-3.5. Absence of a feature (sensor) is indicated by '0' and presence of a feature (sensor) in the dataset is indicated by '1'.

Table 3.2: Sensor contribution in Fire #1 dataset using Naïve Bayes (NB) classifier.

TMP	ION	Photo	CO	Accuracy	Standard deviation
0	0	0	1	99.48	0.27
0	0	1	0	88.35	1.26
0	1	0	0	86.35	1.38
1	0	0	0	89.18	1.23
1	1	0	0	86.43	1.38
0	1	1	0	88.74	1.51
0	0	1	1	98.49	0.77
1	0	0	1	99.35	0.32
0	1	0	1	97.67	0.65
1	0	1	0	89.70	1.35
1	1	1	0	89.65	1.24
1	1	0	1	97.65	0.65
1	0	1	1	99.15	0.42
0	1	1	1	98.23	0.54
1	1	1	1	98.17	0.58

As it can be seen from Table 3.2, using NB classifier, the best classification result is achieved when only CO sensor is used. The worst result is achieved when only Ionization sensor is used.

Results of the same simulation using Feed Forward Neural Network (FFNN) classifier is reported in Table 3.3. It can be seen that the best classification results for FFNN is obtained when all features are available and the worst result is achieved when Temperature, Ionization, and Photoelectric sensors are used together.

Table 3.3: Sensor contribution in Fire #1 dataset using Feed Forward Neural Network (FFNN) classifier.

TMP	ION	Photo	CO	Accuracy	Standard deviation
0	0	0	1	93.81	1.15
0	0	1	0	86.07	2.64
0	1	0	0	53.46	2.08
1	0	0	0	87.86	2.68
1	1	0	0	48.78	6.89
0	1	1	0	47.21	2.24
0	0	1	1	97.71	0.68
1	0	0	1	96.74	0.81
0	1	0	1	98.42	0.60
1	0	1	0	51.01	6.76
1	1	1	0	45.53	4.29
1	1	0	1	98.56	0.46
1	0	1	1	97.55	0.73
0	1	1	1	98.55	0.52
1	1	1	1	98.61	0.45

Table 3.4 reports simulation results using SOM K-means classifier. It can be seen that the best result for SOM K-means is obtained when CO sensor is combined with Temperature sensor. The worst result is obtained when Ionization sensor is used alone.

Table 3.4: Sensor contribution in Fire #1 dataset using SOM K-means classifier.

TMP	ION	Photo	CO	Accuracy	Standard deviation
0	0	0	1	99.02	0.42
0	0	1	0	92.80	1.69
0	1	0	0	85.26	1.63
1	0	0	0	88.21	1.62
1	1	0	0	92.78	1.60
0	1	1	0	96.03	0.89
0	0	1	1	99.00	0.46
1	0	0	1	99.76	0.33
0	1	0	1	98.01	0.63
1	0	1	0	98.17	0.61
1	1	1	0	97.58	0.72
1	1	0	1	99.03	0.55
1	0	1	1	99.61	0.34
0	1	1	1	97.94	0.63
1	1	1	1	98.84	0.58

Table 3.5 reports contribution of each feature in the classification process using decision tree (DT) classifier. It can be seen that the best classification result is obtained when Ionization, Photoelectric, and CO sensors are used in combination, while the worst result is achieved when Temperature sensor is used alone.

Table 3.5: Sensor contribution in Fire #1 dataset using Decision Tree (DT) classifier.

TMP	ION	Photo	CO	Accuracy	Standard deviation
0	0	0	1	99.14	0.37
0	0	1	0	92.92	1.11
0	1	0	0	94.50	1.12
1	0	0	0	89.71	1.14
1	1	0	0	98.14	0.61
0	1	1	0	97.05	0.82
0	0	1	1	98.95	0.49
1	0	0	1	98.78	0.43
0	1	0	1	98.95	0.44
1	0	1	0	99.04	0.47
1	1	1	0	98.79	0.56
1	1	0	1	99.02	0.41
1	0	1	1	99.05	0.40
0	1	1	1	99.21	0.37
1	1	1	1	99.18	0.40

From Tables 3.2-3.5, one can draw the following conclusions:

- Availability of CO has the highest effect on classification accuracy. Using almost all classifiers when CO is unavailable, the classification accuracy drops considerably off.
- In contrary to what is generally believed, Temperature sensor is not the most contributing sensor for fire detection.
- Amount of contribution of each feature to the classification accuracy differs for different classifiers. Therefore, it is not easy to identify the best feature or feature combination for each classifier.

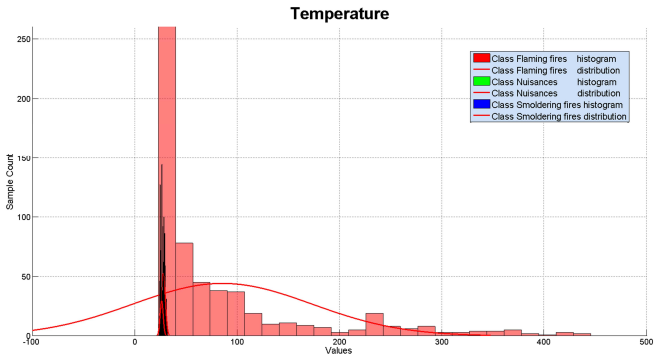
3.3 Fire #2 dataset

Fire #2 dataset is a bigger dataset than fire #1 and contains more data instances. Fire #2 dataset is also obtained from NIST [109] and has three classes, i.e., (i) flaming fire (SDC10, SDC14, SDC07, SDC05, SDC33), (ii) smoldering fire (SDC31, SDC40, SDC34) and (ii) nuisances (MH09, MHN06, MHN15). The total number of data instances in this dataset is 2506. This dataset also contains four features (sensor data types), i.e., Temperature (C), Ionization (1/m), Photoelectric (1/m) and CO (carbon monoxide, ppm). Table 3.6 shows data range of all features of fire #2 dataset.

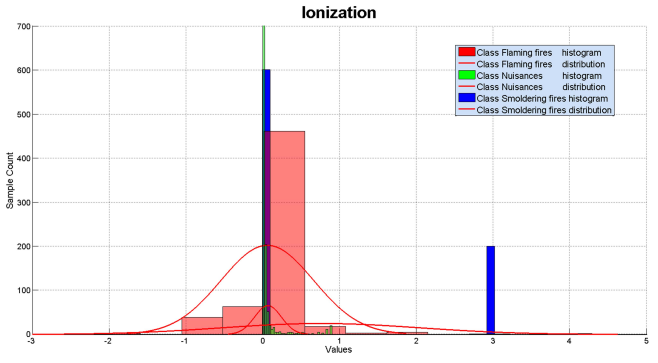
Table 3.6: Data range of all features of fire #2 dataset.

	Temperature (C)			Ionization (1/m)			Photoelectric (1/m)			CO (ppm)		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
Flaming class	22.880	85.520	445.00	-9.061	0.0580	4.2826	-0.055	0.1613	0.4710	0.0000	0.0001	0.000
Smoldering class	24.570	26.221	28.730	-0.006	0.7545	3.0161	-0.051	-0.013	0.0086	0.0000	0.0000	0.000
Nuisance class	23.700	27.859	32.000	-0.00	0.0666	0.8990	-100.0	-26.02	0.2310	-4.000	1.0072	36.00
Overall	22.880	41.050	445.00	-9.061	0.2841	4.2826	-100.0	-11.49	0.4710	-4.000	0.4462	36.00

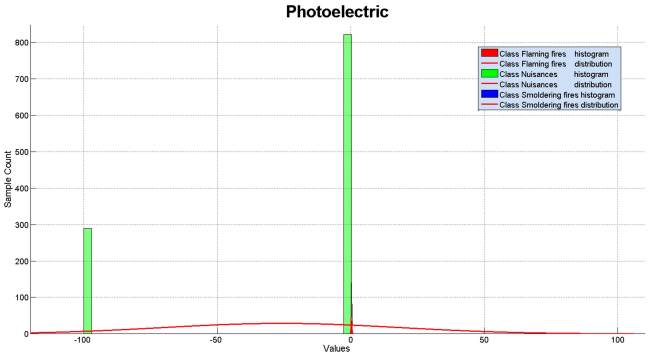
It can be seen from Table 3.6 that Temperature and Ionization sensor value ranges in flaming fire are larger than smoldering and nuisances. Additionally, Photoelectric and CO sensor ranges for nuisances dataset are the largest. Data distribution per feature for all three event (class) types, i.e., flaming fire, smoldering fire, and nuisance, of fire #2 dataset is depicted in Figure 3.3.



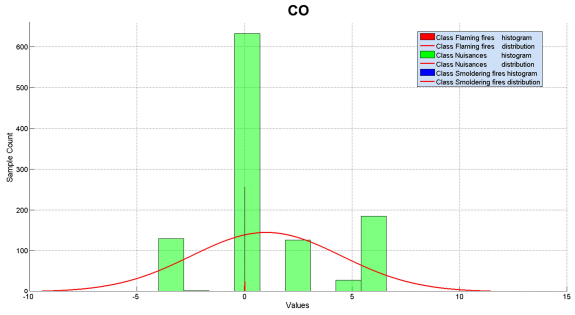
(a)



(b)



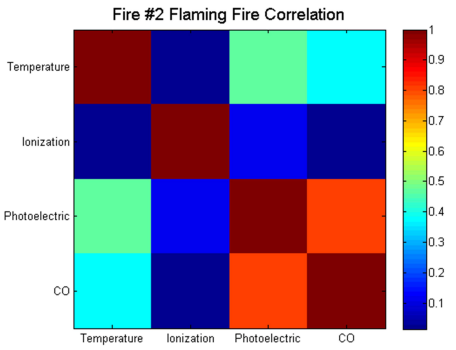
(c)



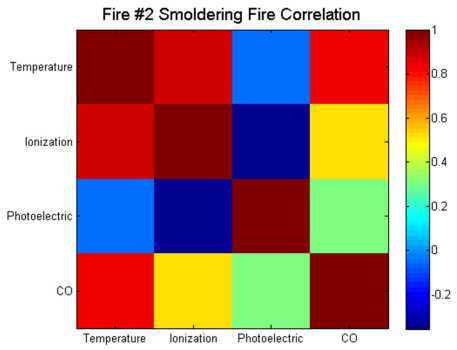
(d)

Figure 3.3: Fire #2 data distribution for (a) Temperature, (b) Ionization, (c) Photoelectric, and (d) CO sensors.

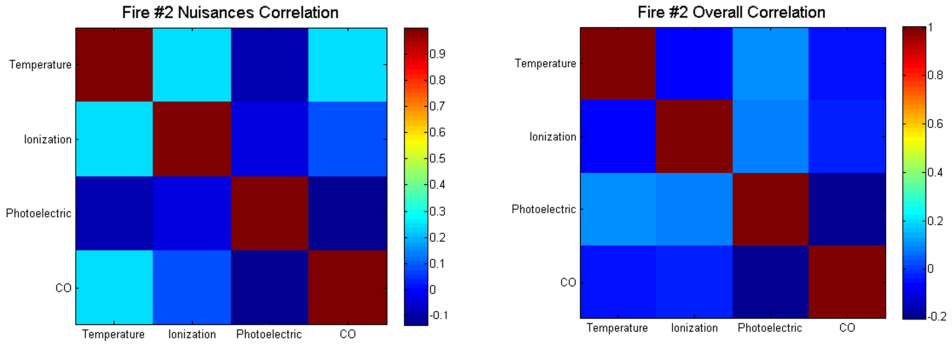
It can be seen from Figure 3.3 that Temperature and Ionization sensors have a wider data distribution for flaming fire, and Photoelectric and CO have the wider data distribution for nuisances. Generally speaking, wider data distribution leads to lesser data overlap. Therefore, classification in a dataset with wider distribution is easier and more accurate. Figure 3.4 demonstrates how fire #2 features are correlated.



(a)



(b)



(c) (d)

Figure 3.4: Data correlation in fire #2 dataset for (a) Flaming fire class, (b) Smoldering fire class, (c) Nuisances class, and (d) Overall correlation between data in all classes.

From Figure 4.3, one notes that features in flaming fire are correlated, except for Ionization with Temperature and Ionization with CO. Smoldering fire's features and nuisance's features are also correlated except for Ionization with Photoelectric. In overall correlation all features report a minimal correlation apart from CO with Temperature and CO with Ionization. To find out about contributions of each of these features to the classification (event detection) process, we consider all possible feature combinations and run our classifiers (the classifiers are introduced and discussed in Chapter 4) for 1000 times and report mean and standard deviations in Table 3.7-3.10. Absence of a feature (sensor) is indicated by '0' and presence of a feature (sensor) in the dataset is indicated by '1'.

Table 3.7: Sensor contribution in Fire #2 dataset using Naïve Bayes (NB) classifier.

TMP	ION	Photo	CO	Accuracy	Standard deviation
0	0	0	1	58.16	1.37
0	0	1	0	58.75	1.49
0	1	0	0	66.89	1.33
1	0	0	0	86.00	1.03
1	1	0	0	84.66	1.78
0	1	1	0	61.66	1.56
0	0	1	1	46.53	1.37
1	0	0	1	57.29	1.40
0	1	0	1	65.93	1.34
1	0	1	0	60.22	1.41
1	1	1	0	63.27	1.54
1	1	0	1	67.36	1.89
1	0	1	1	60.61	1.40
0	1	1	1	64.77	2.64
1	1	1	1	74.00	1.63

As it can be seen from Table 3.7, using NB classifier, the best classification result is achieved when all four features are present. The worst classification result is achieved when Photoelectric and CO are used in combination.

Table 3.8: Sensor contribution in Fire #2 dataset using Feed Forward Neural Network (FFNN) classifier.

TMP	ION	Photo	CO	Accuracy	Standard deviation
0	0	0	1	44.20	1.39
0	0	1	0	53.60	1.42
0	1	0	0	58.63	1.63
1	0	0	0	66.50	1.82
1	1	0	0	67.77	1.61
0	1	1	0	53.55	2.31
0	0	1	1	44.25	1.49
1	0	0	1	59.87	2.67
0	1	0	1	53.07	1.75
1	0	1	0	62.91	4.39
1	1	1	0	66.46	2.46
1	1	0	1	66.21	1.63
1	0	1	1	61.38	4.00
0	1	1	1	52.49	1.50
1	1	1	1	65.18	2.32

Table 3.8 reports classification result on fire #2 dataset utilizing Feed Forward Neural Network (FFNN). It can be seen from Table 3.8 that the best classification result using FFNN is obtained when only Temperature sensor is used, while combination of Photoelectric and CO leads to the worst classification result.

Table 3.9: Sensor contribution in Fire #2 dataset SOM K-means classifier.

TMP	ION	Photo	CO	Accuracy	Standard deviation
0	0	0	1	84.68	20.29
0	0	1	0	84.58	1.24
0	1	0	0	76.74	2.19
1	0	0	0	84.75	1.43
1	1	0	0	92.93	1.19
0	1	1	0	87.23	1.30
0	0	1	1	73.16	8.33
1	0	0	1	88.11	1.32
0	1	0	1	81.61	2.55
1	0	1	0	88.59	1.37
1	1	1	0	93.13	1.60
1	1	0	1	93.08	1.42
1	0	1	1	90.47	1.32
0	1	1	1	85.41	2.11
1	1	1	1	93.41	1.62

Table 3.9 reports classification on fire #2 dataset utilizing SOM K-means. It can be seen that the best classification result using SOM K-means is obtained when all four features are

available. The worst classification result is achieved when Ionization and CO are used in combination.

Table 3.10: Sensor contribution in Fire #2 dataset using Decision Tree (DT) classifier.

TMP	ION	Photo	CO	Accuracy	Standard deviation
0	0	0	1	95.79	0.64
0	0	1	0	85.35	1.26
0	1	0	0	90.68	1.02
1	0	0	0	93.55	0.81
1	1	0	0	98.50	0.42
0	1	1	0	96.02	0.78
0	0	1	1	96.72	0.65
1	0	0	1	99.13	0.35
0	1	0	1	99.09	0.43
1	0	1	0	95.78	0.73
1	1	1	0	98.48	0.45
1	1	0	1	99.58	0.28
1	0	1	1	98.98	0.36
0	1	1	1	99.27	0.40
1	1	1	1	99.56	0.29

Table 3.10 reports classification result on fire #2 dataset utilizing decision tree (DT). It can be seen that combination of Temperature, Ionization, and CO leads to the highest classification accuracy, while use of Photoelectric alone leads to the worst classification accuracy. From Table 3.2-3.5 one can draw the following conclusions:

- As the size of a dataset increases, so does its complexity. Fire #2 dataset is clearly a more complex dataset than fire #1 dataset.
- Fire #2 dataset has a different data distribution than fire #1 dataset. Having a different data distribution leads to different contribution of features to the classification process. Therefore, in fire #2 dataset Temperature and CO sensors both appear to be the most contributing sensors.
- The choice of the best contributing set of sensors is very dependent on the classifier used.

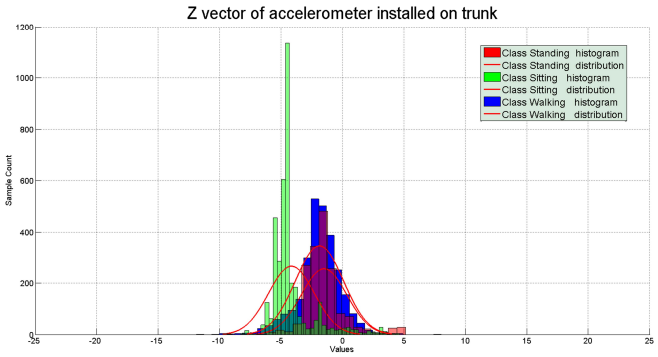
3.4 Activity dataset

Activity dataset is a dataset containing activities of a healthy person collected at the medical center of Enschede city (MST) [110]. This data was collected while the person was standing still, sitting, and walking. This dataset contains 30 features (collected by ten 3D sensors). Out of these 10 features, we select 4 most contributing features using the genetic algorithm (GA) [111]. In Section 3.5, we present the procedure of feature selection using GA. The four most contributing features being selected after applying the genetic algorithm are (i) ‘X’ vector of accelerometer installed on left foot, (ii) ‘Z’ vector of

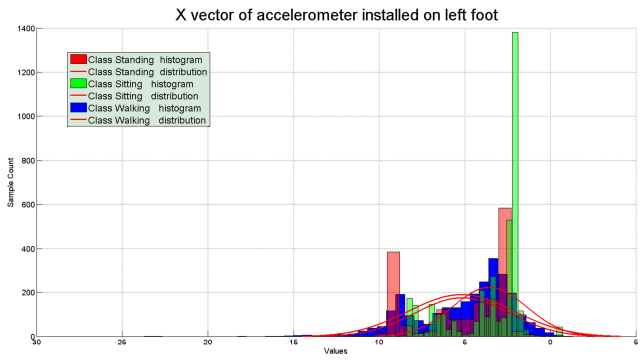
accelerometer installed on trunk, (iii) 'Y' vector of accelerometer installed on trunk, and (iv) 'Z' vector of gyroscope installed on the right foot. This dataset contains three classes (activities), i.e., (i) standing still, (ii) sitting, and (iii) walking. Table 3.11 shows data range of all four features of the activity dataset.

Table 3.11: Data range of activity dataset.

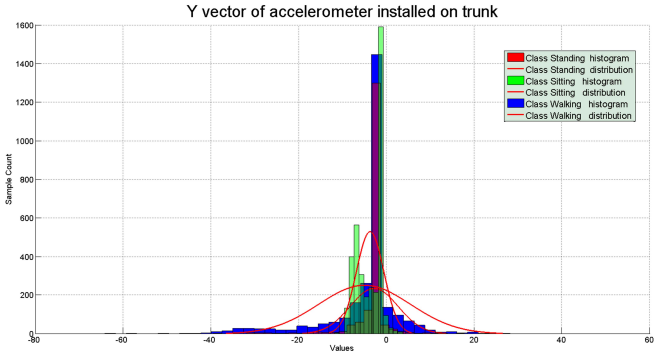
	Z-gyro-r-foot			Y-acc-trunk			Z-acc-trunk			X-acc-l-foot		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
Standing Class	-5.050	0.0259	6.4239	-49.02	-2.845	31.031	-24.54	-1.517	5.1276	-29.80	-5.181	0.573
Sitting Class	-9.233	-0.004	5.9959	-53.95	-3.756	14.771	-12.94	-4.169	7.5942	-19.40	-3.580	0.667
Walking Class	-8.871	-0.022	10.77	-78.76	-5.029	50.005	-12.52	-1.890	20.473	-26.99	-5.224	1.801
Overall	-9.233	-0.004	10.775	-78.76	-3.992	50.005	-24.54	-2.857	20.473	-29.80	-4.461	1.801



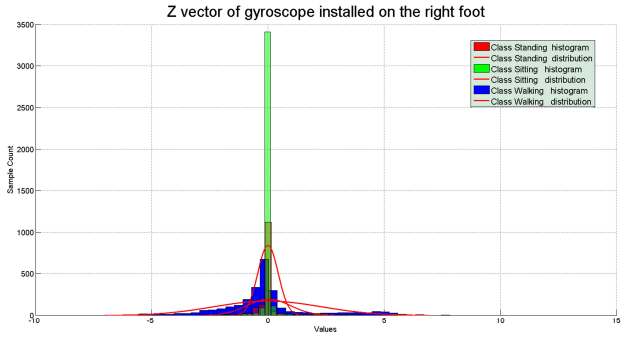
(a)



(b)



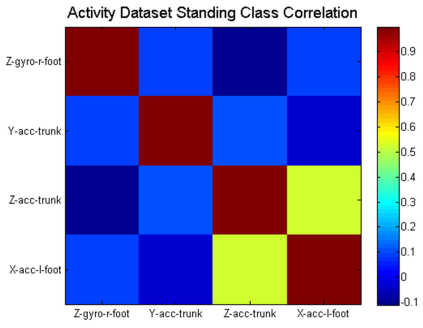
(c)



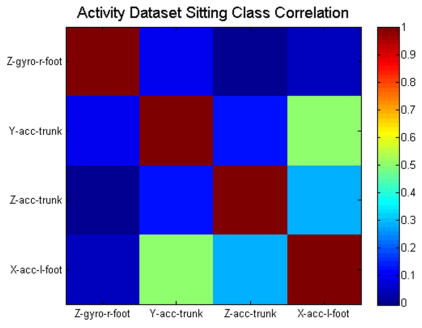
(d)

Figure 3.5: Data distribution for (a) 'Z' vector of accelerometer installed on trunk , (b) 'X' vector of accelerometer installed on left foot, (c) 'Y' vector of accelerometer installed on trunk, (d) 'Z' vector of gyroscope installed on the right foot features.

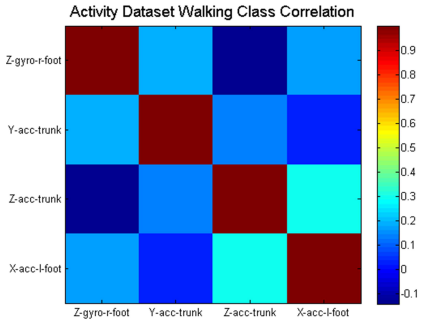
Figure 3.5 shows how activity data is distributed. It can be clearly seen from Figure 3.5 and Table 3.11 that the data range of all three classes and the their data distributions are quite similar. This high degree of overlap makes the classification (event detection) process difficult.



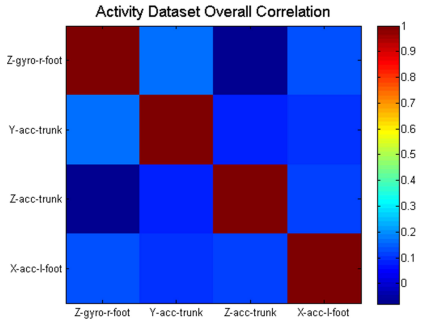
(a)



(b)



(c)



(d)

Figure 3.6: Data correlation in fire #2 dataset for (a) standing (b) sitting (c) walking and (d) overall correlation between data in all classes.

Figure 3.6 illustrates data correlation of the activity dataset. It can be seen that features in standing class are almost uncorrelated. The only exception is 'Z' vector of gyroscope installed on the right foot, which is correlated with the 'Z' vector of accelerometer installed on trunk. For the sitting class, 'Z' vector of gyroscope installed on the right foot is correlated with the 'Z' vector of accelerometer installed on trunk, and 'X' vector of accelerometer installed on left foot is correlated with the 'Y' vector of accelerometer installed on trunk. All features in the walking class are somewhat correlated except for the 'Y' vector of accelerometer installed on trunk, which is not correlated with 'X' vector of accelerometer installed on left foot. The overall correlation between features is quite minimal. Moreover, as expected, X' vector of accelerometer installed on the left foot shows almost no correlation with 'Z' vector of gyroscope installed on the right foot.

To investigate the contribution of each feature to the classification (event detection) process, we consider all possible combinations of the four most contributing features and run our classifiers (the classifiers are introduced and discussed in Chapter 4) for 1000 times and report mean and standard deviations in Table 3.12-3.16. Absence of a feature (sensor) is indicated by '0' and presence of a feature (sensor) in the dataset is indicated by '1'. In these tables (i) F1 is 'Z' vector of gyroscope installed on the right foot, (ii) F2 is 'X' vector of accelerometer installed on left foot, (iii) F3 is 'Z' vector of accelerometer installed on trunk, and (iv) F4 is 'Y' vector of accelerometer installed on trunk.

Table 3.12: Sensor contribution in Activity dataset using naïve Bayes classifier.

F1	F2	F3	F4	Accuracy	STANDARD DEVIATION
0	0	0	1	61.29	0.80
0	0	1	0	67.17	0.79
0	1	0	0	64.45	0.78
1	0	0	0	58.73	0.88
1	1	0	0	67.27	0.93
0	1	1	0	67.57	0.82
0	0	1	1	67.50	0.77
1	0	0	1	60.77	0.95
0	1	0	1	65.82	0.78
1	0	1	0	63.68	0.87
1	1	1	0	67.64	0.88
1	1	0	1	66.82	0.90
1	0	1	1	63.69	0.90
0	1	1	1	67.03	0.84
1	1	1	1	67.72	0.84

The best classification accuracy for the activity dataset, according to Table 3.12, is obtained when all four features are used, while the lowest classification accuracy is achieved when F1 is used alone.

Table 3.13: Sensor contribution in Activity dataset using Feed Forward Neural Network (FFNN) classifier.

F1	F2	F3	F4	Accuracy	STANDARD DEVIATION
0	0	0	1	54.83	2.26
0	0	1	0	47.29	1.43
0	1	0	0	50.76	1.43
1	0	0	0	62.51	7.96
1	1	0	0	51.42	2.31
0	1	1	0	50.49	1.14
0	0	1	1	48.73	3.86
1	0	0	1	55.63	4.25
0	1	0	1	49.52	1.61
1	0	1	0	51.66	5.01
1	1	1	0	50.89	3.82
1	1	0	1	52.17	3.83
1	0	1	1	50.52	4.83
0	1	1	1	49.32	1.23
1	1	1	1	55.67	7.79

Table 3.10 reports classification results on the activity dataset utilizing Feed Forward Neural Network (FFNN). According to Table 3.13, using a Feed Forward Neural Network on the activity dataset, the best classification result is obtained when four features are used in combination and the worst result is obtained when F3 alone is used.

Table 3.14: Sensor contribution in Activity dataset using SOM K-means classifier.

F1	F2	F3	F4	Accuracy	STANDARD DEVIATION
0	0	0	1	66.65	0.78
0	0	1	0	70.91	0.76
0	1	0	0	70.03	0.73
1	0	0	0	67.74	0.74
1	1	0	0	76.44	1.66
0	1	1	0	76.09	0.75
0	0	1	1	77.76	0.70
1	0	0	1	76.15	0.86
0	1	0	1	76.19	0.73
1	0	1	0	77.50	0.73
1	1	1	0	79.98	0.71
1	1	0	1	79.98	0.78
1	0	1	1	81.70	0.68
0	1	1	1	79.62	0.70
1	1	1	1	81.96	0.75

Classification results on the activity dataset applying SOM K-means is reported in Table 3.14. It can be seen that SOM K-means achieves its best accuracy when all four features are used in combination and the worst classification results is achieved when F4 is used alone.

Table 3.15: Sensor contribution in Activity dataset using Decision Tree (DT) classifier.

F1	F2	F3	F4	Accuracy	STANDARD DEVIATION
0	0	0	1	57.08	0.87
0	0	1	0	59.18	0.93
0	1	0	0	59.49	0.91
1	0	0	0	55.90	0.90
1	1	0	0	70.93	0.79
0	1	1	0	70.89	0.87
0	0	1	1	70.67	0.86
1	0	0	1	69.54	0.85
0	1	0	1	71.86	0.84
1	0	1	0	70.08	0.85
1	1	1	0	76.97	0.83
1	1	0	1	76.79	0.79
1	0	1	1	76.61	0.80
0	1	1	1	76.35	0.81
1	1	1	1	79.07	0.81

Table 3.15 reports classification results on the activity dataset using decision tree (DT) classifier. Decision tree reaches to its maximum accuracy when all four features are used in combination and the weakest classification result is achieved when F1 is used alone.

From Table 3.12-3.15 one can draw the following conclusions:

- The best classification accuracy on the activity dataset (using our classifiers) is obtained when all four features are used in combination.
- The features are relatively of the same importance.
- Amount of contributions of features to the event detection (classification) process is varied using various classifiers.

3.5 Feature Selection

In this section, we discuss how we identify the most contributing features of the two fire datasets and the activity dataset.

3.5.1 Feature selection for fire datasets

The basic idea of fire detection is similar to commercial products for fire detection. Many commercial products can only detect airborne smoke by using either Ionization sensors or Photoelectric sensors [112]. An alarm is then generated upon increase of the airborne smoke. The problem with such detection is nuisance sources such as a cigarette or a toasting bread [113, 114]. Consequently, many researchers agree on the fact that increasing fire detection accuracy requires more than one sensor along with an appropriate detecting algorithm [113-115].

To find out about the best set of features for fire detection, we have done an extensive research on previous fire detection projects to see how fires are detected accurately by fire management experts and published the research in [116]. We found out that fire can accurately be detected using a precise algorithm utilizing Temperature, Ionization, Photoelectric, and Carbon monoxide sensors.

Using the above sensors not only helps with detection flaming and smoldering fires, but also helps detection of nuisances (e.g., smoking a cigarette) which are very similar to real fires [113-115]. We obtained our fire datasets from NIST [109], which contains data collected from real fire experiments. The other available features in NIST dataset are:

- Time
- Carbon dioxide
- Smoke Obscuration
- Aspirated Smoke
- Sprinkler telltale pressure
- Mass of burning item

3.5.2 Feature selection for activity dataset

Activity dataset contains real data about activities performed by a healthy person collected in medical center of Enschede city (MST) [110]. Activities performed include (i) standing still, (ii) sitting, and (iii) walking. This dataset contains 30 features including:

- Features 1-3: x,y,z accelerometer on trunk
- Features 4-6: x,y,z gyroscope on trunk
- Features 7-9: x,y,z accelerometer on thigh
- Features 10-12: x,y,z gyroscope on thigh
- Features 13-15: x,y,z accelerometer on shank
- Features 16-18: x,y,z gyroscope on shank
- Features 19-21: x,y,z accelerometer on left foot
- Features 22-24: x,y,z gyroscope on left foot
- Features 25-27: x,y,z accelerometer on right foot
- Features 29-30: x,y,z gyroscope on right foot

The overall correlation of the activity dataset containing all 30 features is depicted in Figure 3.7.

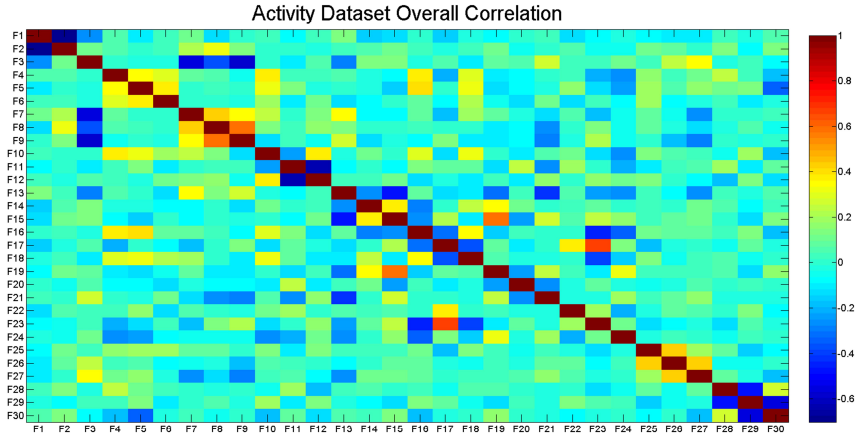


Figure 3.7: Overall correlation between features of activity dataset.

It can be clearly seen from Figure 3.7 that every three features are correlated because they are x,y,z vectors of the same sensor. To eliminate the less contributing features we use the genetic algorithm (GA) along with decision tree (DT) to search in solution space and to find the optimal features automatically. By using GA and DT we identify the four most contributing features, which are:

- ‘Z’ vector of gyroscope installed on the right foot (feature #30)
- ‘X’ vector of accelerometer installed on left foot (feature #19)
- ‘Z’ vector of accelerometer installed on trunk (feature #3)
- ‘Y’ vector of accelerometer installed on trunk (feature #2)

Next subsection discusses the feature selection method utilizing GA and DT in more details. To have another confirmation on the selected features, we use self-organizing map (SOM) and Kohonen training algorithm to train neural networks by all possible combinations of the 30 features. Figure 3.8 shows final results of our feature selection process using SOM.

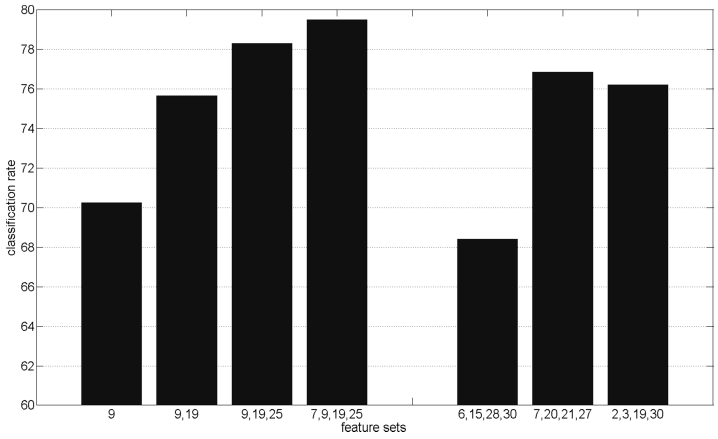


Figure 3.8: Feature selection results using Kohonen network.

Figure 3.8 shows that SOM chooses features number 7, 9, 19, 25 as features leading to the best classification accuracy (around 79% accuracy). However, as we have learned in previous sections the most contributing sensors are, in most circumstances, classifier dependent. Therefore, the lesson learned from Figure 3.8 is that our set of best features (2, 3, 19, 30) provides a classification accuracy (around 76%) very close the best result provided by SOM, and therefore, is not far from being the most contributing features. Further information on self-organizing maps and Kohonen training is available in Chapter 4.

3.5.2.1 Feature selection by genetic algorithm (GA)

Genetic algorithm (GA) is a population-based search heuristic algorithm that imitates natural evolution [117]. GA belongs to a larger class of evolutionary algorithms (EA) being used for optimization and search problems [118, 119]. In a genetic algorithm, a population of sequences (called chromosomes), encodes possible solutions (called individuals) and then evolves toward global best solution using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover [109, 110, 117, 118].

Using GA, in our feature selection procedure, we code feature numbers into a chromosome. Fitness of the chromosome is obtained by training a decision tree (DT) with the features available in the chromosome and testing the DT with unseen data. Then, detection accuracy (obtained by classification of unseen data) is stored as the fitness of the chromosome. 10% brilliant population always saved to keep the best solutions for next generation. Crossover rate is set to 50% and mutation rate 30%. Figure 3.9 shows chromosomes, fitness and crossover method.

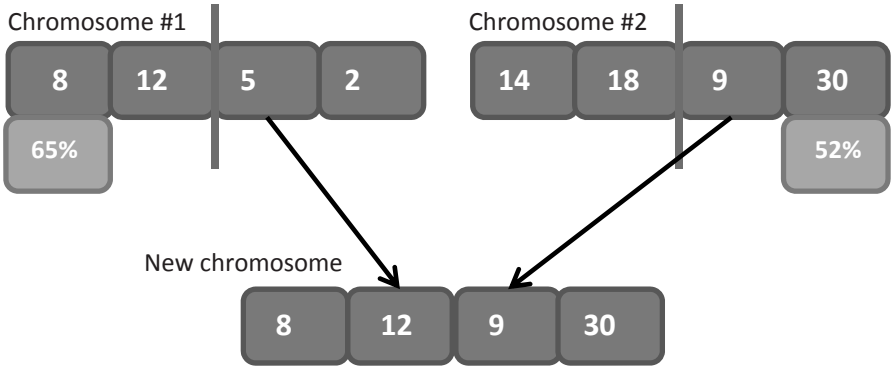


Figure 3.9: A graphic representation of genetic algorithm for our feature selection application.

Figure 3.9 shows two chromosomes while chromosome #1 encodes features number 8,12,5,2 having fitness value of 65% and chromosome #2 encodes features number 14,18,9,30 having fitness value of 52%. These two chromosomes make a crossover to create a new chromosome inheriting from both parents. The fitness of the new chromosome will be calculated using DT.

3.6 Conclusion

In this chapter, we analyzed three datasets to have an understanding about nature of data we will be using in this thesis for performance evaluation of our proposed event detection techniques. We explicitly looked into effects of data correlation, distribution and contribution of each feature to the classification process. We also discussed our feature selection procedure, using which the most contributing features are selected. The lesson learned in this chapter include:

- As discussed in Section 3.5, to find the most contributing features from a dataset, either expert knowledge should be used (as in case of our two fire datasets), or a quantitative research should be conducted (as in case of our activity dataset).
- Data and its features contribute differently to the classification process. Therefore, the best features or the best feature sets should be chosen by investigating various classifiers. As discussed in Sections 3.2-3.4, classifiers produce various classification accuracy on the same set of features.

- Single sensor classification cannot be a strong paradigm as the most contributing sensor differ in most circumstances for every classifier. As discussed in Section 3.2-3.4, generally the most contributing single feature is not the same when a different classifier is used.

Data with high degree of overlap (as in case of activity dataset) is more difficult to classify causing a classification result with lower accuracy.

Chapter 4[‡]

Online Supervised Event Detection

The man of science has learned to believe in justification, not by faith, but by verification.

Thomas H. Huxley (1825-95) English biologist.

Abstract:

As it has been discussed earlier in this thesis, event detection requires an algorithm to discriminate event data from non-event data. In this thesis we are in particular interested in use of artificial intelligence techniques for event detection in wireless sensor networks. However, machine learning and artificial intelligence approaches are traditionally developed for resource-rich computing devices and not for resource constrained sensor nodes. Using them on sensor nodes, therefore, requires an optimization. Therefore, in this chapter we optimize three machine learning approaches, i.e., feed forward neural networks, decision tree, and naïve Bayes classifiers to enable them to be executed on sensor nodes. Moreover, by combining self-organizing maps (SOM) and K-means clustering algorithms, we propose a new supervised machine learning approach, called SOM K-means, which is computationally light, resource friendly and capable of being executed in both local and fusion-based classification models. Furthermore, we analyze applicability of the aforementioned approaches for fast and accurate event detection using two classification models, i.e, local and fusion-based through simulation.

[‡] This chapter is the extension of the following papers:

- "Use of wireless sensor networks for distributed event detection in disaster management applications". International Journal of Space-Based and Situated Computing, 2012 [32].
- "Sensor Fusion-based Activity Recognition for Parkinson Patients". In: Sensor Fusion - Foundation and Applications. InTech, 2011 [33].
- "Fast and Accurate Residential Fire Detection Using Wireless Sensor Networks". Environmental Engineering and Management Journal 2010 [7].
- "Distributed Event Detection in Wireless Sensor Networks for Disaster Management". In: International Conference on Intelligent Networking and Collaborative Systems, INCoS 2010 [8].
- "Sensor Fusion-based Event Detection in Wireless Sensor Networks". In: SensorFusion 2009 [30].
- "Use of AI Techniques for Residential Fire Detection in Wireless Sensor Networks". In: AIAI 2009 [29].

4.1 Introduction

Supervised learning is the process of setting up an inferring function from specific number of labeled training data [120-122]. The labeled training data contains a set of training examples that stores input data (normally as a vector) and a desired output value. During the learning phase, a supervised learning algorithm analyzes the labeled training data to create the inferring function. The inferring function (also known as a classifier) calculates the correct output value (or degree of belongings) for all valid input data. The accuracy of the classifier can then be gauged by measuring classification correctness on a set of unseen data (so called test data) during a process called testing [120-122]. Calculating the accuracy of a classifier on a test dataset gives an insight on precision and accuracy of the classifier when the same data types appear in real world situations.

Since events generally exhibit a pattern [8, 40, 41], supervised learning techniques appear to be promising for accurate event detection in WSNs. This is because the pattern of event can be learnt during the training phase by the classifier. However, using classifiers in WSNs is by no means easy as they were originally designed for computationally powerful devices. Moreover, real-time classification is not well supported by most classifiers. Therefore, existing classifiers need modifications or optimizations before being applicable to WSNs. These modifications aim to make classifiers computationally inexpensive and fast to be executed on sensor nodes for online and (near) real-time event detection.

In this chapter we analyze applicability of four supervised algorithms for online event detection in WSNs and optimize them. These algorithms are based on our modifications of Decision Tree (DT), Naïve Bayes (NB), Feed Forward Neural Networks (FFNN), and K-means classifiers. We also present two classification models, in which these classifiers are executed, i.e. either on a single sensor node or a group of sensor nodes. We will show later in this chapter (in Sections 4.4 and 4.6) that involvement of more than one sensor node in the event detection process improves detection accuracy and fault tolerability. Having said this, it will also be shown (in Section 4.2) that local event detection on a single sensor node offers valuable advantages in terms of communication efficiency in small scale sparse wireless sensor networks, where there is no possibility or need for sensor node communication.

In what follows, the classification models are first introduced and followed by description of the three classifiers we optimize for event detection in WSNs. Next, by combining self-organizing maps (SOM) and K-means clustering algorithms, we propose a new supervised machine learning approach, called SOM K-means. Consequently, all four classifiers are gauged in terms of event detection accuracy and computational and space complexities. Finally, impacts of parameters of the classifiers are analyzed in detail and some conclusions are drawn.

4.2 Classification models

Regardless of which classifier is being used, two classification models may be used. These models in fact indicate the place where classifiers are executed. The first model is a local model, which involves only one sensor node at a time for detecting events. The second model is a fusion-based model engaging more than one sensor node to detect events.

4.2.1 Local classification model

The local classification model assumes that each sensor node performs classification individually without communicating and cooperating with others. Figure 4.1 illustrates the local classification model, which consists of (i) a number of sensors providing input to the classifier, (ii) the classifier, which is responsible for event detection, and (iii) classification output, which indicates type of occurred event.

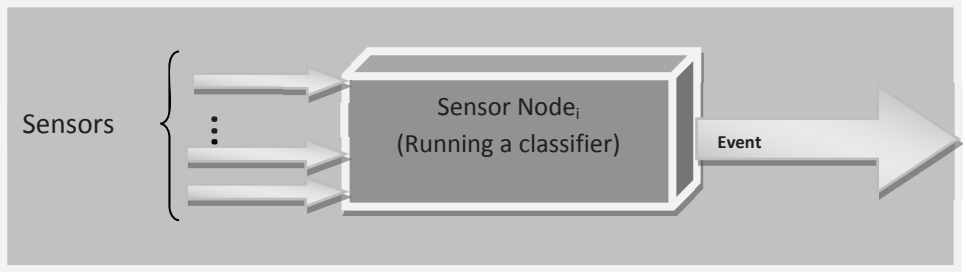


Figure 4.1: Local classification model.

4.2.2 Fusion-based classification model

It is well known that sensors, sensor nodes, and communication links in WSNs are not always reliable and their failure is a common practice. A fusion-based classification model tolerates individual sensor and sensor node failures and involves more than one sensor node in the classification process. The fusion-based approach is composed of the local approach and allows individual sensor nodes first classify and detect events on their own. Then, the classification results are all sent to a fuser node (also known as voter) to reach a consensus. Figure 4.2 illustrates the fusion-based classification model. One should note that as this model is independent of the type of classifiers, not all sensor nodes (including the fuser node) need to have the same classifiers.

Since we consider sensor nodes send their detected event as a singleton (yes/no or class type) to the data fuser, the communication between nodes and the voter is unidirectional. Additionally, the rate of sending data is based on rate of event occurrences, which is usually low. Having a unidirectional communication model with low data transfer makes the fusion-based classification model a distributed model with low communicational overhead.

As it can be seen from Figure 4.2, sensor nodes can be homogenous or heterogeneous in terms of their type of sensors. Classification on each sensor node is done independently. Only data fuser needs to wait for inputs from sensor nodes to fuse their decision about occurrence of an event and reach a consensus. In the case of sensor failure, which makes the sensor nodes unable to send their decision to the voter, fuser node considers that input as a "missed value" and continues its fusion task. Therefore, the classifier used on fuser node should be trained in a way to be able to handle the missed values.

Despite the advantages offered by the fusion-based classification model, its performance may be suboptimal when many faulty sensor nodes report wrong decisions to the fuser. In this case, the fuser cannot accurately decide about occurrence of an event (more information is available in [8, 32]).

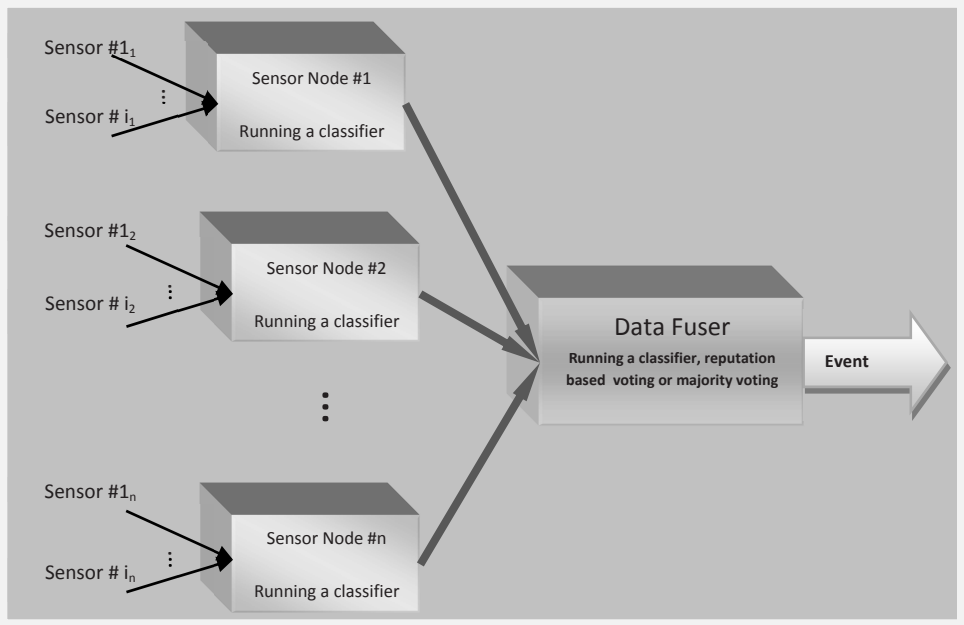


Figure 4.2: Fusion-based classification model.

4.3 Classifiers

Due to resource constraints of the wireless sensor nodes (such as memory, processing and communication), any classifier being executed in WSNs need to be resource friendly to be executed on sensor nodes. This makes existing classifiers not directly applicable to WSNs. To this end, we optimize Naïve Bayes, Feed Forward Neural Network, and Decision Tree classifiers and also present a new supervised learning technique, called SOM K-means, which is based on self-organizing maps (SOM) and K-means. As we will show in Section

4.6, there is always a trade-off between resource consumption (such as memory, processing and communication) and detection accuracy of the classifiers.

4.3.1 Decision tree

The motivation behind choosing decision tree (DT) as a classifier for event detection in WSNs is its simplicity, as it can be implemented as if-then-else rules on sensor nodes once it is created and has undergone the training phase.

Generally, a decision tree is a learning algorithm that uses tree-like graphs to model and evaluate discrete functions. The inputs to the tree may be either continuous or discrete but the outputs (the decisions) are discrete. Construction of a decision tree for classification requires a training phase. This training phase employs a set of data and a learning algorithm to find a minimum depth decision tree. The tree should contain the minimum required nodes (or minimum depth) to reduce time and memory complexities. Therefore, the training algorithm is usually a local search greedy algorithm to find an optimum decision tree. A typical graphical representation of a decision tree is shown in Figure 4.3. In a tree, the decision making process starts from the root of the decision tree and propagates down to the leaves [120, 121, 123].

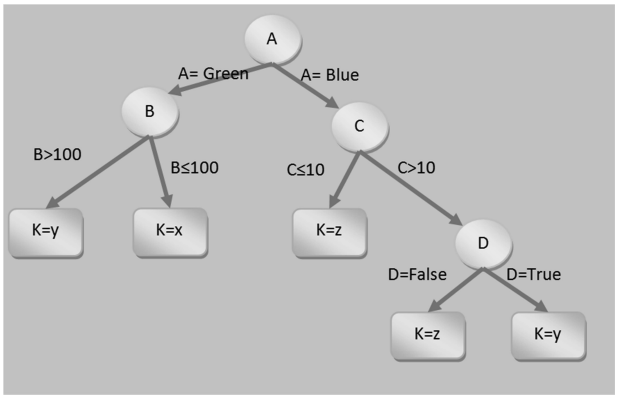


Figure 4.3: An example of a typical graphical representation of a decision tree.

A DT can be implemented either as a set of if-the-else rules or a tree data structure (using arrays or linked lists). If a DT is constructed offline in a computer, then programing it into sensor nodes by set of if-then-else rules enables sensor nodes to execute it.

4.3.2 Naïve Bayes classifier

A Naïve Bayes classifier uses Bayesian statistics and Bayes' theorem to find the probability of each instance belonging to a specific class. It is called Naïve because of its emphasis on independency of the assumptions [120]. To find the probability of belongingness of each

instant to a specific class, Eq. (4.1) can be used, which expresses the probability of an example $E = (x_1, x_2, ..., x_n)$ belonging to class c [124].

$$p(c \mid E) = \frac{p(E \mid c)p(c)}{p(E)} \qquad \text{Eq.(4.1)}$$

Naïve Bayes is algorithmically simple and computationally light. This allows Naïve Bayes to be easily programmed on sensor nodes. The most time and resource consuming part of running a Naïve Bayes classifier is computation of $P(E|c)$, as it holds data distribution for every classes [121]. This probability calculation is important because it has a direct effect on classification accuracy. In basic literatures of pattern recognition or machine learning, it is proposed that this probability can be estimated by a standard data distribution like Gaussian or Poisson [121]. To do a more accurate probability calculation, a histogram approach can also be used [7, 29, 30]. The histogram approach partitions data into several intervals and counts the data frequency within each interval. Frequency of repetition in each interval shows the probability of each instance belonging to that specific interval.

By calculating $P(E|c)$ offline and storing the histogram of data in a multi-dimensional array, the histogram of data can be stored in memory of a sensor node. Then for every incoming data, the probability of its belongingness to every class can be calculated and the most probable class can be assigned to.

4.3.3 Artificial neural networks

An artificial neural network (ANN) is a computational model based on organic neural networks. It consists of an interconnected collection of artificial neurons processing information using a connectionist method [120, 121, 123]. As a simple modeling of biological neural networks, feed forward neural network (FFNN) consists of one input layer, one or more hidden layers, and one output layer. Inputs are the data and the output is a predicted value (class) that input data is expected to belong to.

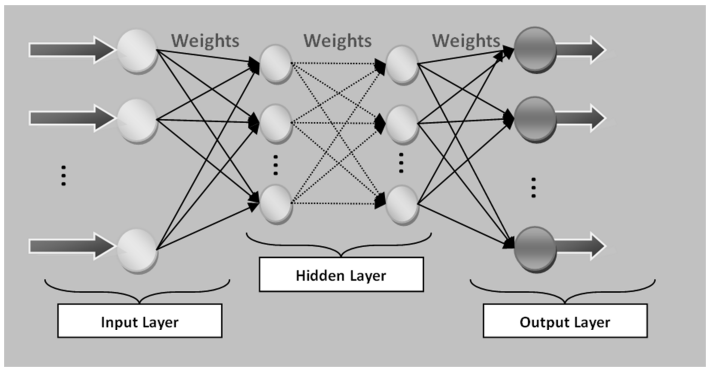


Figure 4.4: An example of a FFNN with three neurons in Input, two neurons in hidden layer, and one neuron in output.

To make an output from inputs of a FFNN, a set of weights should be found in advance. The process of finding these weights is called training phase. Once an FFNN is trained, the whole neural network can be represented as a formula. For example the FFNN illustrated in Figure 4.4 can be expressed as Eq. 4.2

$$Output = [W_{3,1} \times \sum_{j=1}^{3} (W_{1,j} \times I_j)] + [W_{3,2} \times \sum_{j=1}^{3} (W_{2,j} \times I_j)] \quad \text{Eq. (4.2)}$$

By finding the set of weights (in training phase) offline, and programming the representation formula of FFNN in sensor nodes, sensor nodes are able to execute FFNN classifier and classify events online. For implementing FFNN on sensor nodes, sensor nodes should store weights of FFNN in their data memory and arithmetic formula in their code memory. Then the inputs from input layer of FFNN are multiplied to the weights (in input to hidden layer, and hidden to output layer) and the output is calculated using a formula similar to Eq. 4.2.

4.3.4 SOM K-means

SOM K-means is our new proposed technique, combining self-organizing maps (SOM) and K-means clustering algorithm. Although SOM and K-means algorithms are both unsupervised clustering techniques, our SOM K-means is a supervised learning technique which like other supervised learning classifiers needs training phase. Next subsection shortly introduces SOM, while the section after goes into details of our proposed SOM K-means. Figure 4.6 shows flowchart of the SOM K-means.

4.3.4.1 Self-organizing map (SOM)

Self-Organizing Map (SOM) is an unsupervised learning clustering technique that belongs to neural network family. For this technique, similar to many clustering techniques, the number of clusters should be predefined in order to cluster all data into the predefined number of clusters. In addition to number of clusters, a shape also is required to be chosen (Figure 4.5) for SOM. This shape defines neighbors of each cluster center in a 2D space in form of hexagonal, grid or a random topology [125]. Figure 4.5 shows three topologies for SOM. When a cluster center is updated, its neighbors are also updated. Having a geometric shape of neighbors in 2D space makes SOM different from other clustering techniques.

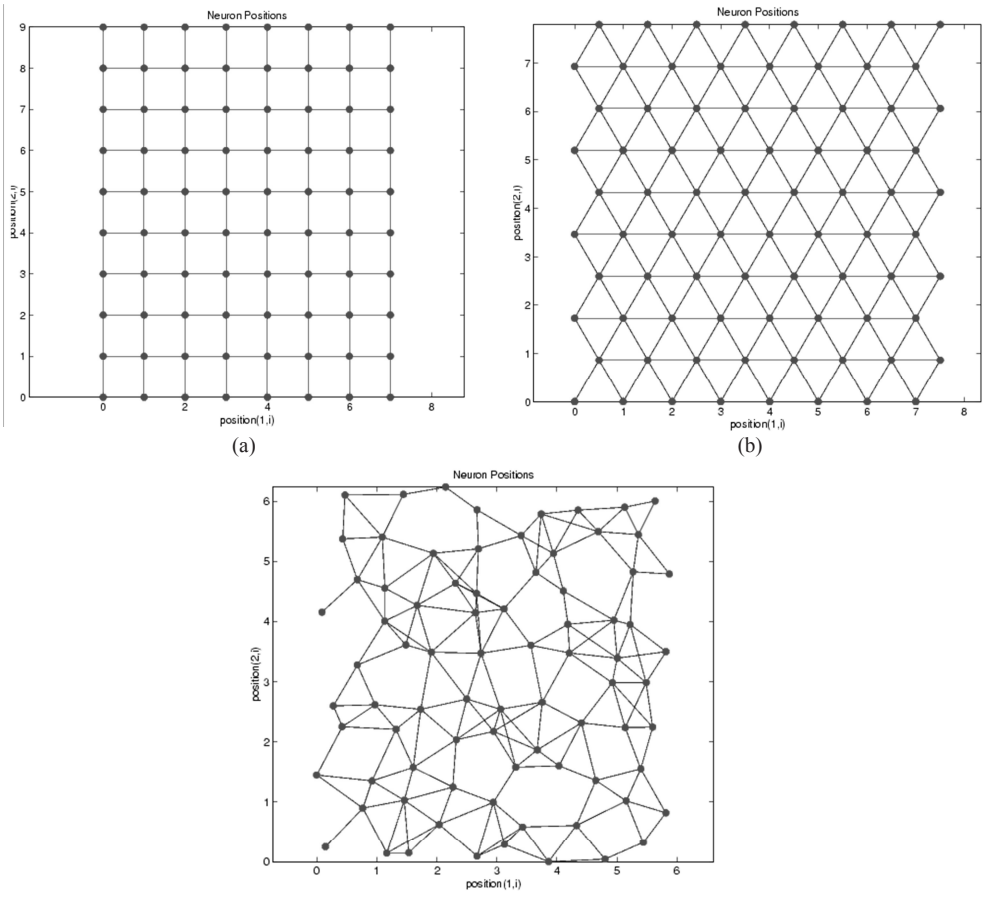


Figure 4.5: Examples of SOM topologies: (a) Grid topology (b) Hexagonal topology (c) Random topology.

The Kohonen algorithm is widely used as an algorithm for SOM clustering. In our proposed SOM K-means approach, SOM clustering utilizing Kohonen algorithm is conducted offline in training phase (Figure 4.6). The steps for Kohonen algorithm are as follows:

1) Initialize center of each cluster, $c_i (i = 1,2,..., n)$, randomly.

2) Read an input vector

3) Traverse each center of cluster

   a. Use Euclidean distance formula $= \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2}$ to find similarity between each cluster center and the input vector. x_{ik}, x_{jk} are the elements of two vectors X_i, X_j and n is the total number of the elements.

   b. Find the cluster center which has the shortest distance with the input vector. This cluster center is called the Best Matching Unit (BMU) or c_{BMU}.

4) Update cluster centers Cv of the BMU neighbors. This update is performed by pulling C_v closer to the input vector using Cv(t + 1) = Cv(t) + Θ(t)α(t)(D(t) – Cv(t)), Where, t is current iteration, λ is limit on time iteration, C_v is the selected neighbor, D is the input vector, Θ(t) is restraint due to distance from BMU and α(t) is learning restraint due to the time.

5) Increment *t* and repeat while $t < \lambda$.

The outputs of SOM is a set of $c_i (i = 1,2,..., n)$, which is a set of cluster centers. Additional information on SOMs can be found in [126-128].

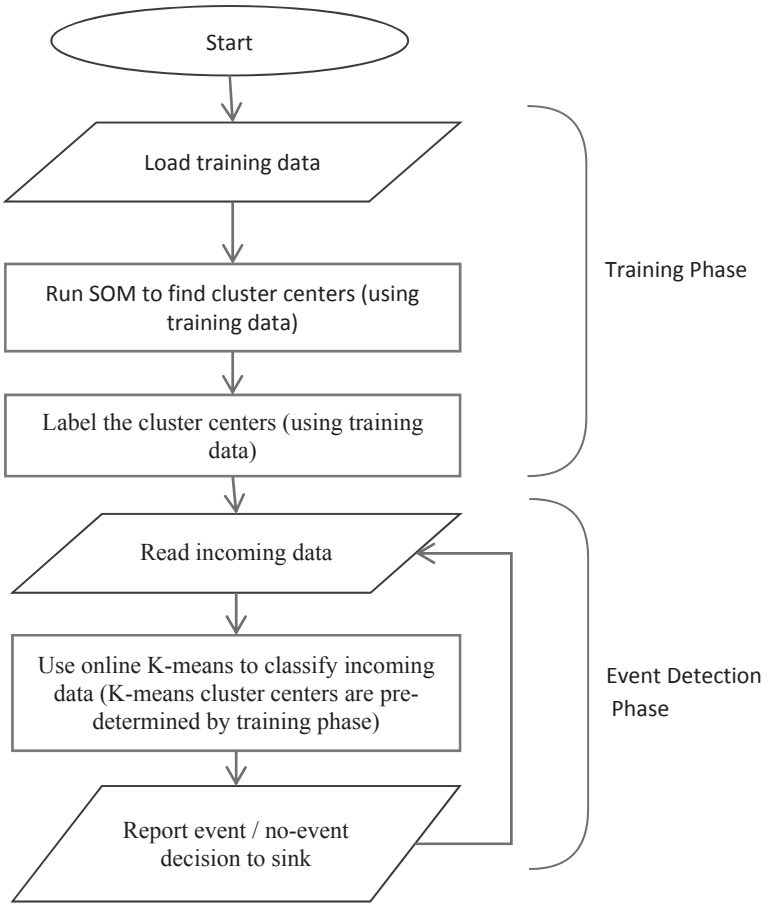


Figure 4.6: Flowchart of SOM K-means algorithm.

4.3.4.2 Training SOM K-means

In the training phase of SOM K-means, a self-organizing map (SOM) is used to cluster the training data into *k* clusters. When *k* number of cluster centers are created by SOM, they should be labeled. Labeling refers to the process of assigning class types to *k* cluster centers.

To label cluster centers, a set of training data is required. The data is classified into the cluster whose cluster center has the shortest distance with. The label of the training data is used to assign the class which this cluster center belongs to. For example, let us assume to have 3 classes and we want to label cluster A. We first check the training data in cluster A to determine which class the majority of data of cluster A belong to. The cluster A may

hold 70% data of class 1, 20% data of class 2, and 10% data of class 3. By having this information, the cluster A is labeled as class 1 (because majority cluster A's population have class 1 in their labels). Figure 4.7 shows an example of a synthetic dataset with 15 cluster centers (C1-C15) and 3 classes (in 3 different colors). This labeling is done by checking labels of the training data as mentioned earlier.

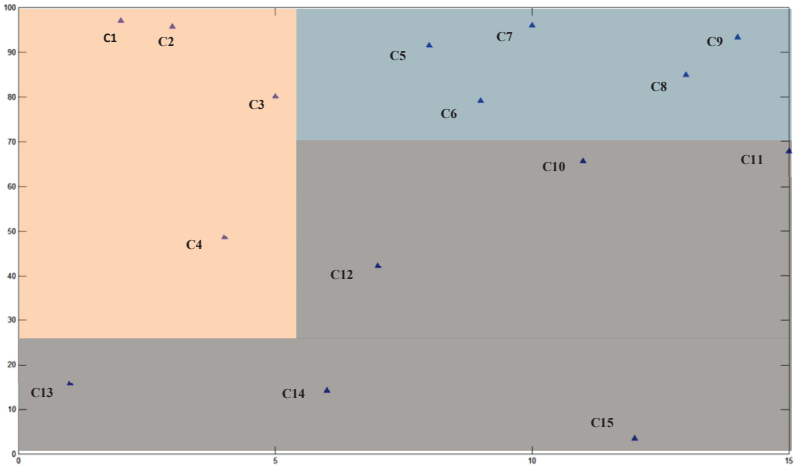


Figure 4.7: An example of labeled data with 15 cluster centers (C1-C15) and 3 classes (separated by 3 colored regions). The cluster centers (C1-C15) are found by SOM and then labeled in 3 classes (separated by 3 colored regions).

4.3.4.3 SOM K-means event detection (testing or classification phase)

Once the cluster centers are found in the training phase, these cluster centers are given to the K-means algorithm as fixed initial points to start clustering. When a new data arrives, Manhattan distance (Eq. 4.3) is calculated between the incoming data and all the cluster centers. This distance calculation indicates similarity between the incoming data and previously clustered data (as cluster centers). Using this distance measurement, the incoming data is classified as the class, whose cluster center has the shortest distance to the incoming data.

$$d_j = \sum_{i}^{m} |x_{i-} y_i| \qquad \text{Eq. (4.3)}$$

Where, j=(1,2,..k), d_j is distance to jth cluster center, m is number of features (dimensions), x_i is the i^{th} dimension of the data, and y_i is the center of cluster j in i^{th} dimension.

Implementing this algorithm on sensor nodes is straightforward. First the cluster centers are found in an offline manner. Then, the cluster centers are programmed into data memory of

a sensor node. In the code memory of the sensor node, K-means algorithm is implemented to check the shortest distance between incoming data and the cluster centers.

Laerhoven proposes a self-organizing map approach for online context detection using sensor data [129]. The self-organizing map is then augmented by a second layer labeled online K-means algorithm to improve the accuracy of the self-organizing map. However, our proposed SOM K-means approach is one supervised learning technique that incorporates SOM and K-means sequentially to train and classify data. By training SOM K-means in offline manner and classifying data in an online fashion, the proposed approach offers a faster data classification and detection of events.

4.4 Simulation results

To show the applicability of the aforementioned classifiers, a number of simulations are conducted and reported in this section. These simulations are conducted using Matlab® R2010b. The datasets used for performance evaluations are introduced and analyzed previously in Chapter 3. Each dataset is divided into $^2/_3$ training data and $^1/_3$ testing data. Table 4.1 and 4.2 report the results on Fire #1, Fire #2, and Activity datasets. All simulations are repeated 1000 times and the mean and standard deviations of detection accuracies for both local and fusion-based classification models are reported in Table 4.1, Figure 4.8, Table 4.2 and Figure 4.9.

Table 4.1: Event detection accuracy using local classification model.

Technique	Dataset	Accuracy	
		Mean	STD
Decision Tree	Fire #1	99.18%	0.40
	Fire #2	99.56%	0.29
	Activity	79.07%	0.81
FFNN	Fire #1	98.61%	0.45
	Fire #2	65.18%	2.32
	Activity	55.67%	7.79
Naïve Bayes	Fire #1	98.17%	0.58
	Fire #2	74.00%	1.63
	Activity	67.72%	0.84
SOM K-means	Fire #1	98.84%	0.58
	Fire #2	93.41%	1.62
	Activity	81.96%	0.75

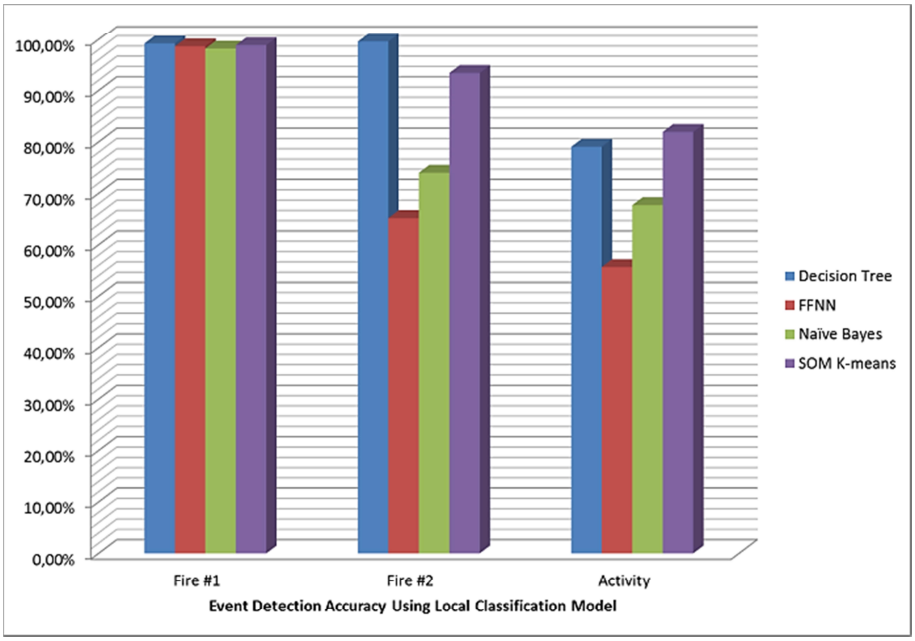


Figure 4.8: Event detection accuracy graph using local classification model.

Table 4.2: Event detection accuracy using fusion-based classification model.

Technique	Dataset	Accuracy	
		Mean	STD
Decision Tree	Fire #1	99.53%	0.26
	Fire #2	99.85%	0.14
	Activity	90.44%	0.65
FFNN	Fire #1	99.09%	0.85
	Fire #2	75.00%	4.35
	Activity	67.19%	2.13
Naïve Bayes	Fire #1	98.60%	0.44
	Fire #2	73.81%	1.32
	Activity	68.60%	0.81
SOM K-means	Fire #1	99.29%	0.37
	Fire #2	95.49%	0.95
	Activity	83.59%	0.62

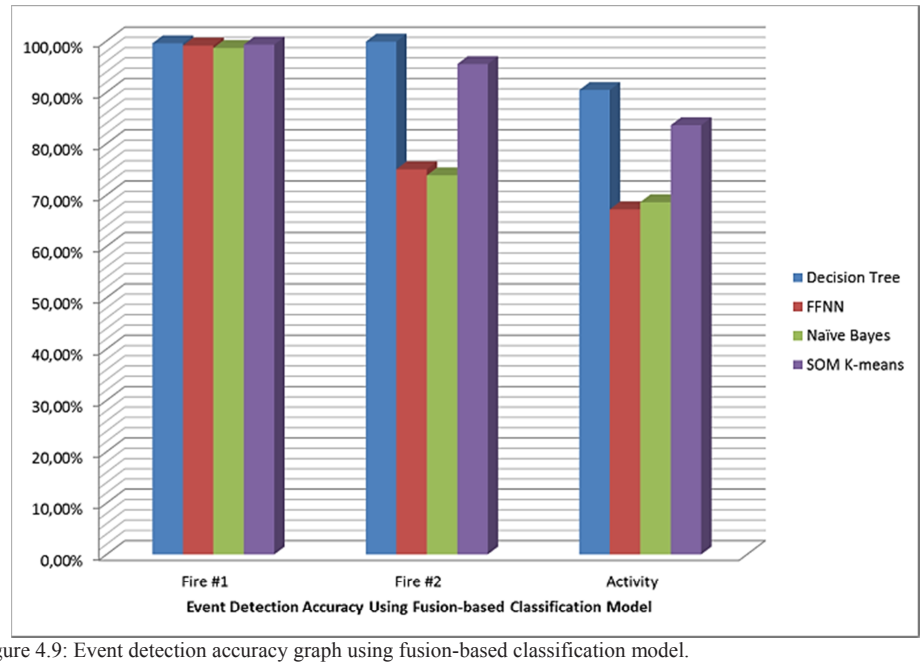


Figure 4.9: Event detection accuracy graph using fusion-based classification model.

From Table 4.1 and Table 4.2, it can be concluded that generally fusion-based classification approaches are more accurate than local classification approaches. This can be due to the fact that data is twice classified, once on the sensor nodes and once on the fuser node. Another conclusion is that there is no classifier and no classification model, which performs best for all datasets. For example, SOM K-means is the best for local detection of activity, while decision tree is the best for distributed event detection of activity. This makes it difficult to select one classifier for all situations.

4.5 Time and space complexity

To compare various classifiers not only their classification accuracy but also their time and space complexities should be taken into account. Time and space complexity give an insight about how much resources are required to gain such an accuracy. Since sensor nodes have code and data memories, which are separate from each other to store code and variables accordingly, by space complexity we mean data memory of sensor nodes (for storing variables).

As mentioned earlier, supervised learning algorithms require a training phase to build the classifier. Since the training phase is conducted only once, and it is conducted offline (usually on a PC and not on a sensor node), here we disregard time complexity of training phase and calculate time and space complexity of the classifiers after the training phase.

4.5.1 Decision tree - time and space complexity

4.5.1.1 Time complexity

To classify using a decision tree (DT) in a local classification model, nodes from root to leaves should be traversed. In the worst case required comparisons is equal to the depth of the tree. Eq. 4.4 shows time complexity of DT.

$$O(DT) = O(f) \qquad \text{Eq. (4.4)}$$

Where *f* is the depth of decision tree. For example, *f* for the fire #1 dataset is 4.

In the fusion-based classification model, n decision trees (n is the number of sensor nodes contributing to the event detection process) independently of each other perform the event detection task. Then, their local results are sent to the fuser, on which another decision tree performs the final classification. Eq. 4.5 represents the time complexity of this model using decision tree.

$$O(Distributed\ DT) = O(decision\ trees\ ) + O(voter) = O(f_1 + f_2 + \cdots + f_n) + O(f) = O(n \times f) + O(f) = O(n \times f) \qquad \text{Eq. (4.5)}$$

Where *f* is the depth of decision tree, *n* is the number of nodes in the network contributing to event detection. For example, *f* for the fire #1 dataset is 4, and in our practical implication (Chapter 6) *n* is *5*.

4.5.1.2 Space complexity

To store a DT in memory of sensor nodes, an array is required to keep all nodes of the DT. Therefore, the required amount of memory for DT is a function of nodes in DT as Eq. 4.6.

$$O(DT) = O(z\ ) \qquad \text{Eq. (4.6)}$$

Where *z* is the number of nodes in the decision tree. For example, *z* for the fire #1 dataset is 16.

4.5.2 Naïve Bayes - time and space complexity

4.5.2.1 Time complexity

Naïve Bayes (NB) uses a 3D array to store histogram of data and finds the most probable class that the data belongs to. The dimensions of the 3D array are equal to the number of classes, number of features, and number of intervals. To find the most probable class that incoming data belongs to, a search in the 3D array is required. As the number of and size of intervals are fixed, going to the right interval is a direct access; however, this direct access should be done for all classes and features. Therefore, the number of times that 3D array should be referred to is a function of number of classes and number of features. Eq. 4.7 shows time complexity of Naïve Bayes (NB) classifier in the local classification model.

$$O(NB) = O(m \times c) \quad \text{Eq. (4.7)}$$

Where m is the number of features (or sensors) and c is the number of classes. For example, m for the fire #1 dataset is 4 and c is 3.

In fusion-based classification model, n sensor nodes run NB classifier. Therefore the time complexity of the whole event detection process, as expressed in Eq. 4.8, is n times *O(NB)*.

$$O(Distributed\ NB) = \text{n} \times O(NB) = O(n \times m \times c) \quad \text{Eq. (4.8)}$$

Where m is the number of features (or sensors) and c is the number of classes, and n is the number sensor nodes contributing to the event detection. For example, m for the fire #1 dataset is 4, c is 3 and in our practical implication (Chapter 6), n is *5*.

4.5.2.2 Space complexity

As mentioned previously, a 3D array is required to store the histogram of data. This data histogram is the most spacious data structure of the NB classifier. Eq. 4.9 shows space complexity of NB classifier.

$$O(NB) = O(m \times c \times i) \quad \text{Eq. (4.9)}$$

Where m is the number of features (or sensors), c is the number of classes and i is the number of intervals. For example, m for the fire #1 dataset is 4, c is 3 and in our practical implication (Chapter 6), i is 180.

4.5.3 Feed forward neural network - time and space complexity

4.5.3.1 Time complexity

Time complexity of Feed Forward Neural Network (FFNN), as expressed in Eq. 4.10, is a function of input, output and neurons in the hidden layer.

$$O(FFNN) = O(m \times o \times h) \quad \text{Eq. (4.10)}$$

Where m is the number of features (input layer), o is the number of outputs and h is the number of neurons in the hidden layer.

In some cases that one output neuron is enough for FFNN to classify data (such as FFNN in our case), Eq. 4.10 alleviates to Eq. 4.11.

$$O(FFNN) = O(m \times h) \quad \text{Eq. (4.11)}$$

Where m is the number of features (input layer), and h is the number of neurons in the hidden layer. For example, m for the fire #1 dataset is 4, and in our practical implication (Chapter 6), h is 100.

In fusion-based classification model, n sensor nodes run FFNN classifiers. Therefore the time complexity of the whole system, as expressed in Eq. 4.12, is n times *O(FFNN)*.

$$O(Distributed\ FFNN) = \text{n} \times O(FFNN) = O(n \times m \times h) \quad \text{Eq. (4.12)}$$

Where n is the number of sensor nodes contributing to event detection, m is the number of features (input layer), and h is the number of neurons in the hidden layer. For example, m for the fire #1 dataset is 4, c is 3 and in our practical implication (Chapter 6), h is 100, and n is *5*.

4.5.3.2 Space complexity

The most spacious data structure of FFNN is the matrix that keeps FFNN weights. The FFNN weights are required for connecting the input layer to the hidden layer and the hidden layer to the output layer. Therefore, all these weights should be stored. Space complexity of FFNN classifier is expressed in Eq. 4.13. Since in this study number of neurons in output layer is 1, FFNN space complexity is mainly a function of input to hidden layer weights.

$$O(FFNN) = O((m \times h) + (h \times o)) = O(m \times h) \quad \text{Eq. (4.13)}$$

Where m is the number of features (input layer), o is the number of outputs and h is the number of neurons in the hidden layer. For example, m for the fire #1 dataset is 4, and in our practical implication (Chapter 6), h is 100.

4.5.4 SOM K-Means - time and space complexity

4.5.4.1 Time complexity

In SOM K-means algorithm, SOM is executed in the training phase and K-means is utilized for classification (event detection). As we do not consider the offline operation here, the time complexity of SOM K-means concerns the K-means part only. Eq. 4.14 shows the time complexity of SOM K-means, which is the part that shortest distance between incoming data and a cluster center is found. When the closest cluster center is found the label of the cluster center shows the class that the data belongs to.

$$O(SOM\ K-means) = O(m \times k) \quad \text{Eq. (4.14)}$$

Where m is the number of features (or sensors) and k is the number predefined clusters in k-means. For example, m for the fire #1 dataset is 4, and k is 3.

In fusion-based classification model, n sensor nodes run K-means classifier, which results in a time complexity expressed in Eq. 4.15.

$$O(SOM\ K-means) = O(n \times m \times k) \quad \text{Eq. (4.15)}$$

Where n is the number sensor nodes contributing to event detection, m is the number of features (or sensors) and k is the number predefined clusters in k-means. For example, m for the fire #1 dataset is 4, k is 3 and in our practical implication (Chapter 6), *n is 5*.

4.5.4.2 Space complexity

The most spacious data structure of SOM K-means is the matrix holding cluster centers in every dimension (the number of dimensions is the number of features). These cluster

centers are then used to determine when the input data are close to them as indicators of the classes the input data belongs to. Cluster centers are usually in multi-dimensional space and therefore, the space complexity of SOM K-means, as expressed in Eq. 4.16, becomes a function of cluster centers and number of dimensions or features.

$$O(\ SOM\ K-means) = O(m \times k\ ) \qquad \text{Eq. (4.16)}$$

Where *m* is the number of features (or sensors) and *k* is the number predefined clusters in k-means. For example, *m* for the fire #1 dataset is 4, and *k* is 3.

4.5.5 Discussion on time and space complexity

Table 4.3 summarizes time and space complexity of the four mentioned supervised learning classifiers. One conclusion that can be drawn from this table is that both time and space complexities of all these classifiers are polynomial. Therefore, they are plausible to be run by sensor nodes. Additionally, since they are polynomial order of one or two, they are capable of (near) real-time event detection. In Table 4.4 we quantify variables in Table 4.3 from our practical implication (Chapter 6) to give more insight on time complexity and space complexity of the approaches.

Decision about which are faster or less spacious in memory is not straightforward, because complexity of classifiers are functions of different variables (Table 4.3). However, it can be seen that DT is more likely to be the fastest, as its complexity is polynomial order one. To have more insight on how fast DT can be, we take values of parameters from our implementation (Chapter 6) and report them in Table 4.4. Consequently, one can conclude that DT is 3 times faster than NB and SOM K-means and 100 times faster than FFNN.

Table 4.3: Time complexity and space complexity summary.

Classifier	Time complexity	Space Complexity
DT	$O(f)$	$O(z\ )$
NB	$O(m \times c\ )$	$O(m \times c \times i\ )$
FFNN	$O(m \times h\ )$	$O(m \times h)$
SOM K-means	$O(m \times k\ )$	$O(m \times k\ )$

Where *f* is the depth of decision tree, *z* is the number of nodes in the decision tree, *m* is the number of features (or sensors), *c* is the number of classes, *i* is the number of intervals, *o* is the number of outputs, *h* is the number of neurons in the hidden layer, and *k* is the number predefined clusters in k-means.

Table 4.4: Summary of time and space complexity using values from implementation of the approaches on sensor nodes (Chapter 6).

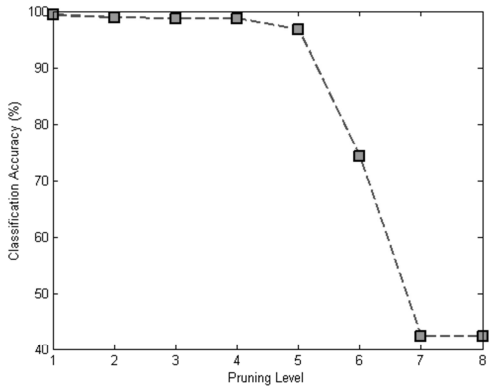
Classifier	Time complexity	Space Complexity
DT	$O(4)$	$O(16\ )$
NB	$O(4 \times 3\ )$	$O(4 \times 3 \times 180\ )$
FFNN	$O(4 \times 100\ )$	$O(4 \times 100)$
SOM K-means	$O(4 \times 3\ )$	$O(4 \times 3\ )$

4.6 Parameter study

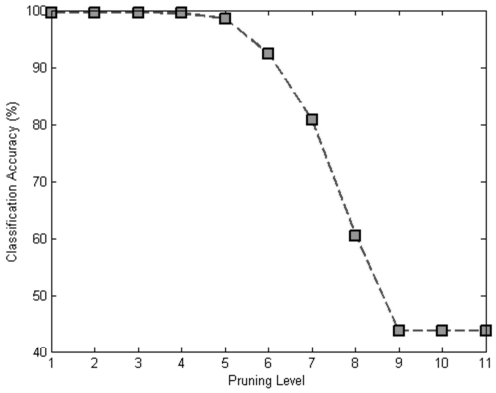
The internal parameters of these four classifiers have direct effect on the classification accuracy as well as time and space complexity. Therefore, the best parameter for each classifier should be chosen considering the tradeoff between classification accuracy and time and space complexity.

4.6.1 Decision tree - parameter study

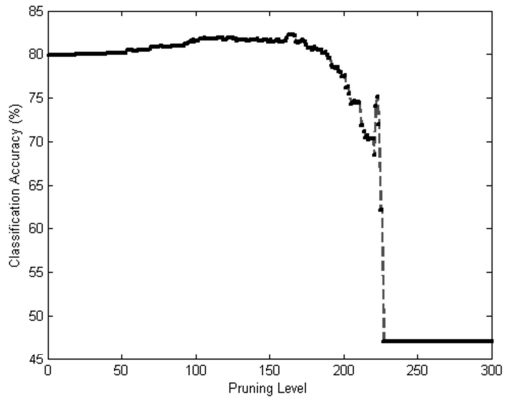
Once the decision tree is created for the purpose of event detection, it can be pruned. Pruning a decision tree reduces the number of nodes and alleviates time and space complexity. However, pruning may also reduce classification accuracy. Figure 4.10 shows the effect of pruning on classification accuracy for Fire #1, Fire #2, and Activity datasets.



(a)



(b)

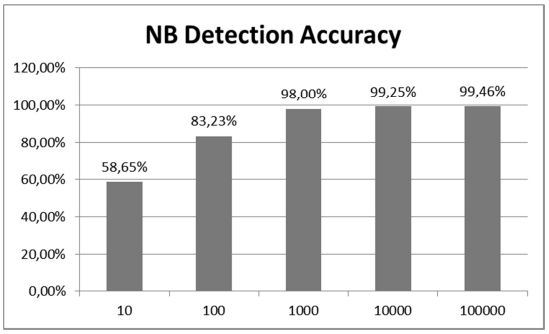


(c)

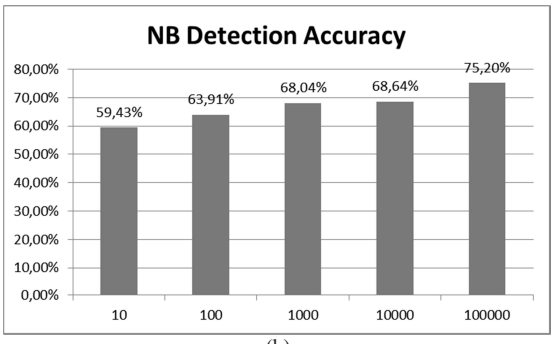
Figure 4.10: Effect of decision tree pruning on classification accuracy for (a) Fire #1, dataset (b), Fire #2 dataset, and (c) Activity dataset.

4.6.2 Naïve Bayes - parameter study

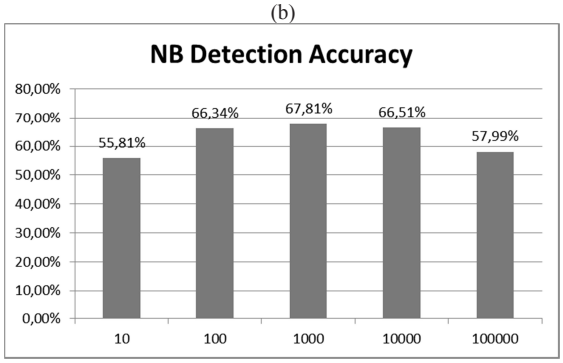
Naïve Bayes (NB) classifier works with histogram of data to find the most probable class that incoming data belongs to. The size of histogram is a function of number of features, number of classes and number of intervals. As the number of feature and classes are fixed and cannot be changed, the only variable parameter is number of intervals. By looking at Table 4.3 it can be seen that number of intervals has a direct effect on space complexity. Figure 4.11 shows the effect of number of intervals on event detection accuracy for Fire #1, Fire #2, and Activity datasets.



(a)



(b)



(c)

Figure 4.11: Effect of intervals number in accuracy of Naïve Bayes (NB) classifier for (a) Fire #1 dataset, (b) Fire #2 dataset, and (c) Activity dataset.

As Figure 4.11 shows, by increasing the number of intervals, the classification accuracy improves, while the space complexity for storing the histogram of data is increased. However, there can also be a turning point where interval growth causes accuracy reduction. This incident can be seen in Figure 4.11. (c), as increase in number of intervals 1000 intervals, leads to accuracy drop off.

4.6.3 Feed forward neural network - parameter study

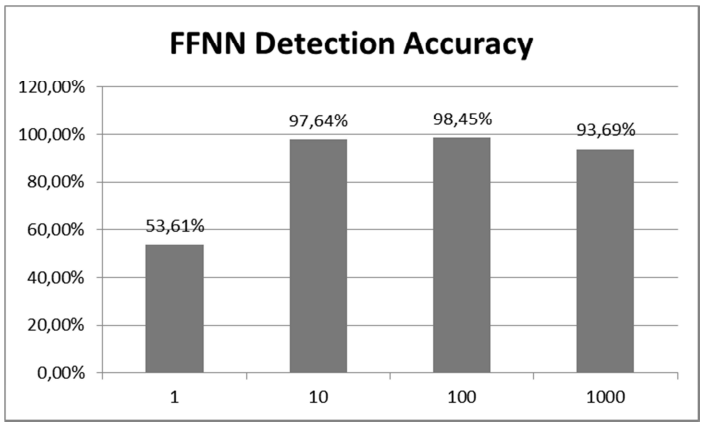
As shown in Table 4.3, number of neurons in Feed Forward Neural Network (FFNN) classifier plays an important role in its time and space complexity. The number of neurons in input and output layers is fixed. Therefore the only variable parameter is number of neurons in the hidden layer. The effect of number of neurons in the hidden layer on classification accuracy is shown in Figure. 4.12 for Fire #1, Fire #2, and Activity datasets.

Figure 4.12 shows that increase of neurons in the hidden layer improves classification accuracy. However, this improvement is not constant and at some point turns into accuracy reduction.

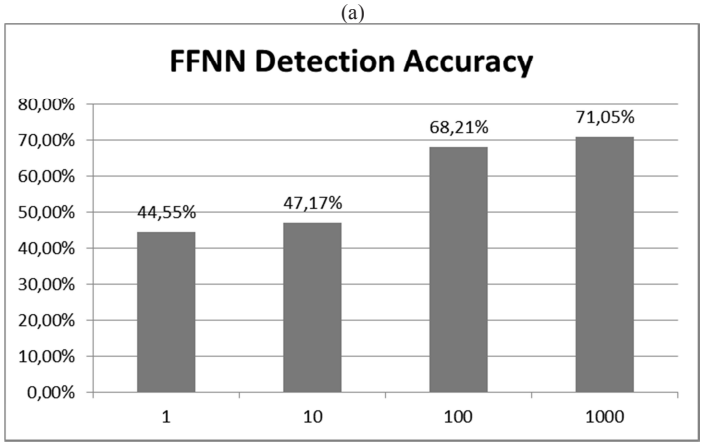
4.6.4 SOM K-means - parameter study

As shown in Table 4.3, the number of cluster centers has significant effect on time and space complexity of SOM K-means classifier. These cluster centers are in 2D space because they are created by a self-organizing map (SOM). Figure 4.13 shows how number of cluster center affects classification accuracy for Fire #1, Fire #2, and Activity datasets.

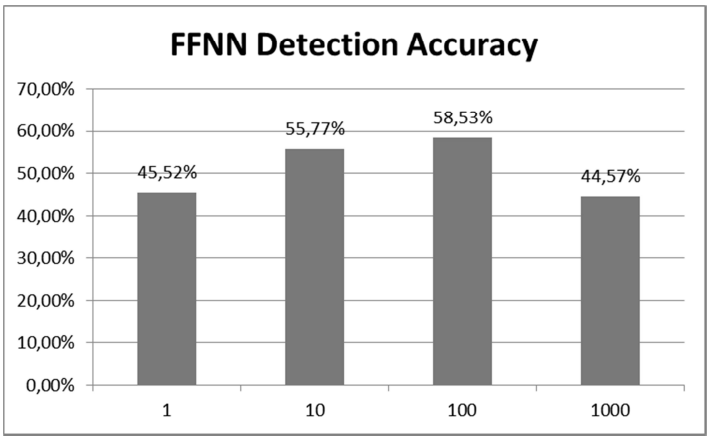
It can be seen from Figure 4.13 that increase of number of cluster center improves detection accuracy to some points. After that, either the detection accuracy remains unchanged (Figure 4.13 (b, c)) or drops off (Figure 4.13 (a)).



(a)

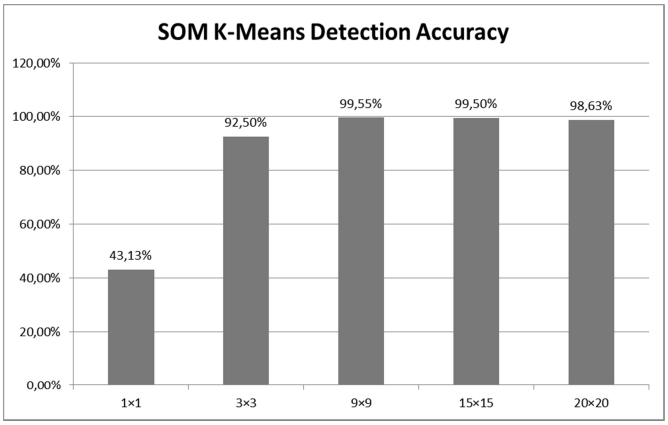


(b)

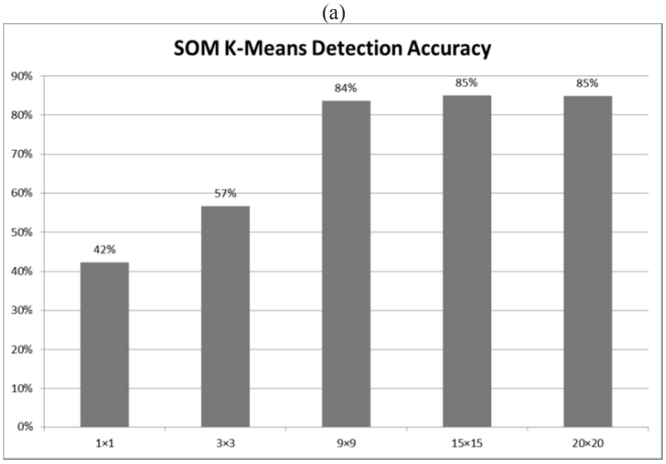


(c)

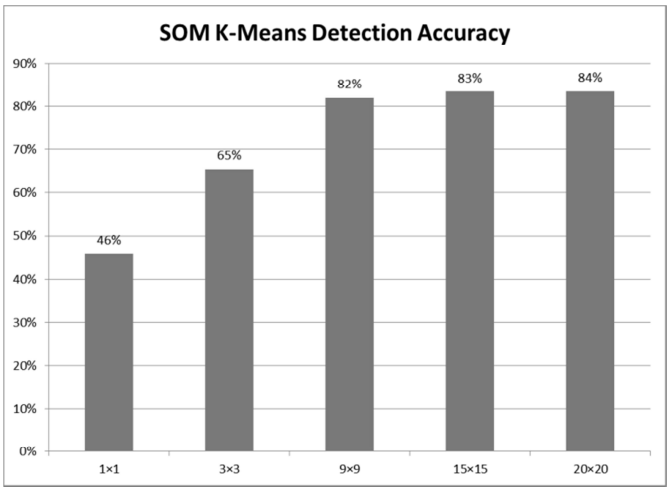
Figure 4.12: Effect of number of neurons in hidden layer in accuracy of Feed Forward Neural Network (FFNN) classifier for (a) Fire #1 dataset, (b) Fire #2 dataset, and (c) Activity dataset.



(a)



(b)



(c)

Figure 4.13: Effect of number of cluster centers in accuracy of SOM K-means classifier for (a) Fire #1, dataset (b) Fire #2 dataset, and (c) Activity dataset.

4.6.5 Fusion-based classification model - parameter study

Two parameters appear to be effective on classification accuracy of event detection using fusion-based classification model. The first parameter is the number of nodes contributing to the event detection process and the second is available sensors in the network. To analyze these effects, we perform a number of simulations on Fire #1 dataset in Matlab using decision tree as classifier on individual sensor nodes and majority voting on the fuser to reach a consensus (more information is available in [32]). We consider a number of heterogeneous network schemes to study these two parameters. Table 4.5 shows the effect of number of nodes and availability of sensors on classification accuracy.

One can conclude from results shown in Table 4.5 that event detection accuracy does not necessarily depend on number of sensor nodes contributing to the classification process, since the network with higher number of sensor nodes does not have the highest accuracy. This is also supported by locality of reference fact (as discussed in Chapter 1), which is an important element in event detection.

Therefore, distributed event detection accuracy is a direct function of most contributing sensor(s) present in the network. In Chapter 3 it is shown that for Fire #1 dataset, CO is the most contributing sensor. Therefore absence of this sensor can make a significant detection accuracy drop off (Table 4.5, scenario #3). Presence of the most contributing sensors help increase event detection accuracy.

Table 4.5: A heterogeneous network with different number of sensor nodes and sensor types along with event detection accuracy using decision tree

Scenario no.	Availability of sensors	Total number of sensor nodes	Accuracy
1	Node: TMP, ION, Photo, CO 1: ✓, ✓, ×, × 2: ×, ✓, ✓, × 3: ✓, ×, ✓, ✓ 4: ✓, ×, ×, ✓ 5: ✓, ×, ×, × 6: ×, ✓, ×, × 7: ×, ×, ×, ✓	7	96.35%
2	Node: TMP, ION, Photo, CO 1: ✓, ✓, ×, × 2: ×, ✓, ✓, × 3: ✓, ×, ✓, ✓	3	92.46%
3	Node: TMP, ION, Photo, CO 1: ✓, ×, ×, × 2: ×, ✓, ×, × 3: ×, ×, ✓, × 4: ✓, ✓, ×, × 5: ✓, ×, ✓, × 6: ×, ✓, ✓, × 7: ✓, ✓, ✓, ×	7	71.65%
4	Node: TMP, ION, Photo, CO 1: ✓, ×, ×, × 2: ×, ✓, ×, × 3: ×, ×, ✓, × 4: ✓, ✓, ×, × 5: ✓, ×, ✓, × 6: ×, ✓, ✓, × 7: ✓, ✓, ✓, × 8: ×, ×, ×, ✓	8	82.22%
5	Node: TMP, ION, Photo, CO 1-7: ✓, ×, ×, × 8: ×, ×, ×, ✓	8	93.43%
6	Node: TMP, ION, Photo, CO 1-5: ✓, ✓, ✓, ✓ 6-10: ✓, ×, ×, × 11-15: ×, ✓, ×, × 16-20: ×, ×, ✓, ×	20	98.95%
7	Node: TMP, ION, Photo, CO 1-10: ×, ×, ×, ✓ 11-100: ✓, ✓, ✓, ×	100	87.1%
8	Node: TMP, ION, Photo, CO 1-100: ✓, ✓, ✓, ×	100	90.63%
9	Node: TMP, ION, Photo, CO 1-2: ✓, ✓, ✓, ✓ 3-4: ×, ✓, ✓, ✓ 5-6: ✓, ×, ✓, ✓ 7-8: ✓, ✓, ×, ✓ 9-10: ✓, ✓, ✓, × 11-2: ×, ×, ✓, ✓ 13-14: ✓, ×, ×, ✓ 15-16: ✓, ✓, ×, × 17-18: ×, ✓, ✓, × 19-20: ✓, ×, ✓, × 21-22: ×, ✓, ×, ✓ 23-24: ×, ×, ×, ✓ 25-26: ✓, ×, ×, × 27-28: ×, ✓, ×, × 29-30: ×, ×, ✓, ×	30	98.85%
10	Node: TMP, ION, Photo, CO 1-50: ✓, ✓, ✓, ✓	50	99,05%

4.7 Conclusion

Since events usually exhibit patterns, use of machine learning techniques for event detection is appropriate. Once the event pattern is learnt by the supervised learning approaches during the training phase, events can be detected (classified) by these techniques (or classifiers) online.

Supervised learning techniques (classifiers) are traditionally designed for resource-rich devices. Therefore, running them on sensor nodes with limited computational power and resources is a challenge. Modification of these techniques and/or design of new techniques is needed to make them suitable to run in an online, (near) real-time fashion.

To this end, we analyzed and optimized Decision Tree (DT), Naïve Bayes (NB), Feed Forward Neural Network (FFNN) in such a way that they fulfill WSN requirements in terms of resource and computational limitation. Additionally, we proposed SOM K-means as a new event detection method to detect events in WSNs.

We gauged applicability of these techniques in Matlab simulations to detect events hidden in three datasets using two classification model schemes, i.e., (i) local on an individual sensor node, and (ii) fusion-based on a group of nodes. To give an idea about resource consumption of these techniques, we also presented the time and space complexity.

The lessons learned include:

- Supervised learning techniques are shown to be promising for event detection in WSNs, as they are accurate and resource friendly.
- As a general guideline, high resource consumption of conventional supervised learning techniques can be reduced by (i) converting the classifier to a mathematical formula (or any other explicit form of business logic) which can be simply calculated (or evaluated) on the sensor nodes, and (ii) optimizing (or simplifying/replacing) the most time-consuming component of the formula in such a way that entire logic is not altered.
- Local classification model and fusion-based classification model can both detect events accurately. However the latter results in a higher event detection accuracy, since data are classified twice.
- Every classifier has certain parameters that directly affect detection accuracy as well as time and space complexities. The right parameters should be chosen considering the tradeoff between time complexity, space complexity, and detection accuracy.

- Scalability is not an important requirement for in-network event detection in WSNs, since event detection accuracy does not necessarily depend on number of sensor nodes. This strongly relates locality of reference as being a strong feature of events.
- Generally speaking, no classifier can detect all event types with high detection accuracy. The event detection accuracy of a classifier is therefore data-dependent.

Chapter 5[§]

Online Unsupervised Learning Event Detection

Absence of understanding does not warrant absence of existence.

Ibn Sina (Avicenna) (980-1037) Persian polymath.

Abstract:

Unsupervised learning techniques are promising for event detection in WSNs due to their capability to detect unknown events. The greatest problem with existing unsupervised techniques, however, is that they are basically designed for batch offline processing. Therefore, to be used for event detection in WSNs, conventional unsupervised techniques should be modified so that they are (i) less resource exhaustive and (ii) can be executed in an online and (near) real-time manner. To enable in-network unsupervised event detection, in this chapter we optimize the well-known K-means algorithm to be faster and less resource exhaustive and to be able to detect events without being trained by label data in a training phase.

5.1 Introduction

Unsupervised learning is a method of finding hidden structures in unlabeled data [120, 121, 123, 130]. This implies that unsupervised learning techniques do not need labeled data to construct an inferring function (or classifier) to separate (or classify) data. Clustering algorithms are one of the main approaches of unsupervised learning that work by grouping similar data into the same clusters [120, 130]. Another approach to unsupervised learning is 'blind signal separation using feature extraction techniques for dimensionality reduction' that finds hidden pattern of data by transforming data into a feature-degraded domain (e.g., PCA and ICA) [53]. Unsupervised learning techniques have already been used in applications such as machine perception, computer vision, natural language processing, search engines, medical diagnosis, bioinformatics, robot locomotion, and smart

[§] This chapter is partially published with the following titles:

- "Online Unsupervised Event Detection in Wireless Sensor Networks". In: Proceedings of the 7th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2011) [31].
- "How Wireless Sensor Networks Can Benefit from Brain Emotional Learning Based Intelligent Controller (BELBIC)". In: 2nd International Conference on Ambient Systems, Networks and Technologies (ANT-2011) [147].

environments [131-136]. There exist various clustering techniques including K-means [137], mixture models [138], hierarchical clustering [139], self-organizing map [103], and adaptive resonance theory [140]. K-means groups *n* observations into *k* clusters in such a way that each observation belongs to the cluster whose center (or its mean) is the closest [121, 137]. Mixture models use a probabilistic model to group all observations into clusters (or sub-observation) [120, 138]. Hierarchical clustering techniques make a hierarchy of data clusters using a recursive function [139, 141], while self-organizing map (SOM) is a topographic clustering technique based on a map showing nearby locations in the map as observations with similar properties [103, 120]. Adaptive resonance theory (ART) permits the number of clusters to change with the size of problem and allows monitoring the degree of resemblance between members of the same clusters by means of a user-defined constant called the vigilance parameter [53, 140].

Unsupervised learning techniques are interesting for event detection in WSNs because they can detect events whose patterns are not known beforehand. In other words, use of unsupervised learning methods allows to detect new type of events without availability of labeled data or training. The greatest problem with existing unsupervised techniques, however, is that they are basically designed for batch offline processing. In this thesis we aim to detect event online (in matter of seconds). Therefore, to be used for event detection in WSNs, conventional unsupervised techniques should be modified so that they are (i) less resource exhaustive and (ii) can be executed in an online and (near) real-time manner.

In this chapter, we propose unsupervised learning techniques, which enable sensor nodes to locally detect events. Our techniques are based on the K-means clustering algorithm, which we modify to be less resource extensive and able to perform online clustering. The motivation behind using K-means is its simplicity, which makes it a good candidate to be executed on resource constrained sensor nodes.

We revisit our processing model briefly in Section 5.2. In Section 5.3, we describe the standard K-means, while in Section 5.4 we explain our modifications to the standard K-means to make it suitable for execution on sensor nodes in an online manner. The simulation results in terms of event detection accuracy and computational and space complexities are presented in Section 5.5 and Section 5.6. Section 5.7 provides an overview of our various optimization attempts being applied to the proposed approach, while Section 5.8 concludes this chapter by drawing some conclusions.

5.2 Classification model

As mentioned in Chapter 4, regardless of the classifier being used, two classification or processing models can be utilized for event detection techniques in WSNs. These classification models are (i) local classification model and (ii) fusion-based classification model. In local classification model, the event detection approach is conducted by an

individual sensor node, while in fusion-based model, the event detection is conducted distributedly by a group of sensor nodes. Distributed event detection can offer advantages such as higher event detection accuracy and fault tolerability. However, local event detection can be equally valuable for small scaled sparse networks. More details on classification models can be found in Chapter 4.

5.3 Standard K-means algorithm

The term "K-means" was first used by James MacQueen in 1967 [142] for classification and analysis of multivariate observations. K-means is a partitioning method that divides data into *k* distinct clusters. K-means treats each observation in the dataset as an object that has a location in *m*-dimensional space (*m* is the number of features or sensors in this context). It groups data in such a way that objects within each cluster are as close to each other as possible, and as distant from objects in other clusters as possible. Each cluster is defined by its members and by its centroid (or cluster center). The centroid for each cluster is the mean value of the members within that cluster. K-means uses an iterative algorithm that minimizes the sum of distances from each object to its cluster center, for all clusters. In an offline batch processing fashion, this algorithm moves objects between clusters until the distance between members of clusters to their centers cannot be decreased further. The result is a set of clusters that are as compact and well-separated as possible [121].

The standard K-means algorithm performs the following steps:

1. Placing *K* points into the space representing the center of objects being clustered. These points represent initial cluster means.

2. Assigning each object to the cluster whose mean has the shortest Euclidean distance ($d = \sqrt{\sum_i^n (x_i - y_i)^2}$) to the object.

3. Recalculating the positions of the cluster centers after a new objects is assigned to the clusters using a regular averaging formula expressed in Eq. 5.1.

$$new\ cluster\ center_l = \frac{1}{n}\sum_{i}^{n} X_i \qquad \text{Eq. (5.1)}$$

Where, $l = (1,2,..,k)$, n is the total number of objects (cluster members) in the l^{th} cluster and X_i is the i^{th} object in the l^{th} cluster.

4. Repeating Steps 2 and 3 till whole data is assigned to K-clusters.

The convergence of the algorithm is discussed in [130]. Figure 5.1 shows an example of data clustering using the K-means algorithm (K=3) on our Fire #2 dataset and on temperature sensor data.

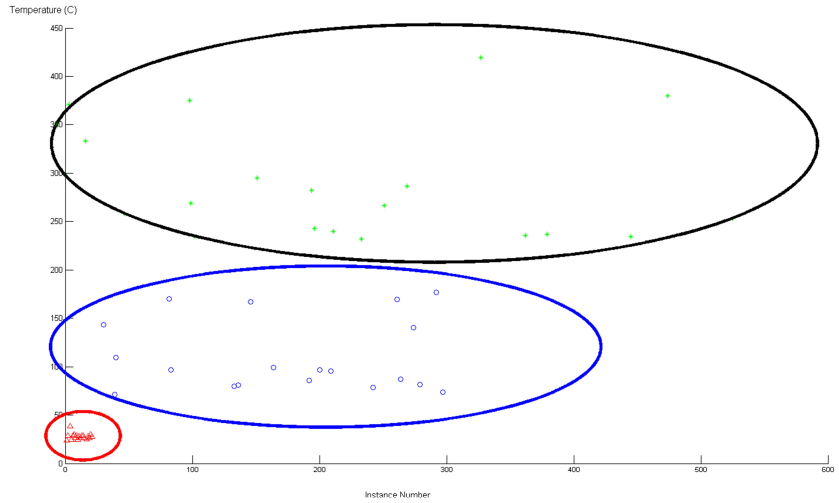


Figure 5.1: An example of data clustering using K-means algorithm (K=3) on Fire #2 dataset, temperature sensor data.

5.4 An online k-means algorithm

Our aim to modify the standard K-means algorithm is to lower down its time and space complexity. This in turn will lead to a simpler classifier to be used for fast event detection on resource limited wireless sensor nodes. The modification of K-means should have the same logic as standard K-means but consume less resources and be faster (to fit in our event detection application of WSNs). Therefore we pursue the following steps:

1. The first k data are assigned to the first k event classes. By doing this the first K-means (first k cluster centers) are created. We do not place k points in space randomly (as standard K-means) because we have no insight on data ranges at the beginning. Assigning the first k data to the first k event classes assures that cluster centers are not off from data ranges for every sensors.

2. A tracking table is made to keep track of the previously seen data in a compact form. The tracking table contains k columns, each representing one of the k clusters. In each column the center of the clusters as well as the number of populations within the respective clusters are stored. Figure 5.2 illustrates how the tracking table looks like. The tracking table is a replacement for storing whole data in memory in standard K-means algorithm, and therefore, conserves memory usage.

3. The Manhattan distance between the $(k+1)^{th}$ data instance (the next coming data) and k means (center of k clusters) is calculated using Eq. 5.2. Consequently the new data instance is assigned to the cluster whose mean has

the shortest distance to the data. A Manhattan distance is proposed to use because it does not need to calculate square root as Euclidean distance and, therefore, is faster.

$$d_j = \sum_{i}^{m} |x_i - y_i| \qquad \text{Eq. (5.2)}$$

Where, j=(1,2,..*k*), d_j is distance to jth cluster center, m is number of features (dimensions), x_i is the i^{th} dimension of the data, and y_i is the center of cluster j in i^{th} dimension.

4. After assigning the new data to one of the clusters, the tracking table should be updated so that the population of the assigned cluster is increased by one. Consequently, center of the cluster, which new data is assigned to, should be updated using Eq. 5.3. In fact, Eq. 5.3 is a weighted accumulative averaging function which updates cluster centers by giving a weight as big as the population of the cluster, before a new member joins, to the previous cluster center and involves the new data with a weight of 1, to calculate new cluster center. This step keeps cluster center updated by any incoming data.

$$y_j = \frac{(P_j \times OldCenter_j) + x_i}{P_j + 1} \qquad \text{Eq. (5.3)}$$

Where, j=(1,2,..*k*), y_j is jth cluster center, *m* is number of features (dimensions), x_i is the i^{th} dimension of the data (i=1,2,..,*m*), and P_j is population of jth cluster.

5. Steps 3 and 4 will be repeated until all data instances are assigned to the *k* clusters.

	Clusters			
	Cluster *1*	**Cluster *2***	**…**	**Cluster *K***
Centers	$<c_{1,1},c_{1,2},..,c_{1,m}>$	$<c_{2,1},c_{2,2},..,c_{2,m}>$	…	$<c_{k,1},c_{k,2},..,c_{k,m}>$
Population	P_1	P_2	…	P_k

Figure 5.2: The tracking table, which keeps track of previously seen data in a compact form.

The proposed algorithm has the same logic as the standard K-means algorithm and its convergence is similar to the standard K-means, which is proved in [31, 130].

The main differences between the our proposed k-means and existing online clustering techniques such as competitive learning [143] are:

- Our proposed k-means is basically designed to be computationally inexpensive and being fitted with requirements of WSNs, therefore, it uses a unique tracking table and cluster center updating formula.
- Unlike competitive learning, in our proposed K-means no neighborhood and topological relationship between cluster centers are defined. Therefore, the approach updates one cluster center at the time and saves some time in updating cluster centers.

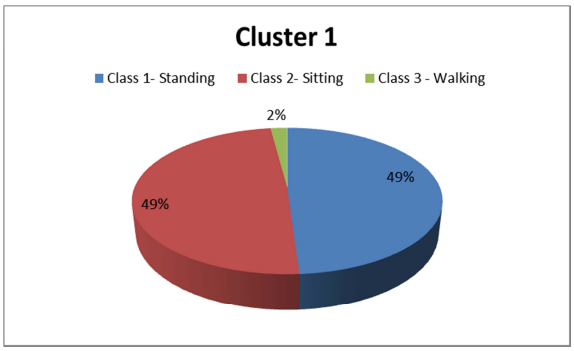
5.5 Performance evaluation and simulation results

Performance evaluation of clustering techniques for event detection applications is not straightforward because labeled data, to gauged the event detection (clustering) accuracy, is often unavailable. Even if labeled data is available, finding the right cluster to assign event types to is not trivial. Clustering techniques first group similar data together in a number of clusters and then need to semantically give a meaning to each cluster. This semantic indicates type of event each cluster represents. For example, let us consider a clustering technique used for activity detection. The idea is to detect three types of events (activities), namely, walking, sitting, and standing still. The clustering technique can group data into three clusters, but the semantic of these clusters is unknown. Use of confusion matrices [121] can assist in identification of type of cluster members and assigning the cluster to the class which has the highest population. The confusion matrix is simply calculated by summing up members of each classes (event types) within each cluster. Table 5.1 shows an example of such confusion matrix for three clusters and three classes utilizing the real activity data. As it can be seen in Table 5.1, cluster 1 has 75% members belonging to class 1 (standing), and 25% members belonging to class 3 (walking). The same calculation can also be done for the other two clusters. The first observation from the confusion matrix illustrated in Table 5.1 is that cluster 1 corresponds to class 1, because high percentage of its members belongs to class 1. Applying the same principle leads to the conclusion that cluster 2 and cluster 3 correspond to class 2 and 3, respectively. Another observation from this confusion matrix is that all three clusters contain some errors. This means that they mistakenly contain members of the other clusters (for example cluster 2 has 20.77% data of class 1 and 33.33% data of class 3). An ideal clustering situation is similar to have no misclassification in the way that 100% of a cluster members belong to one specific class.

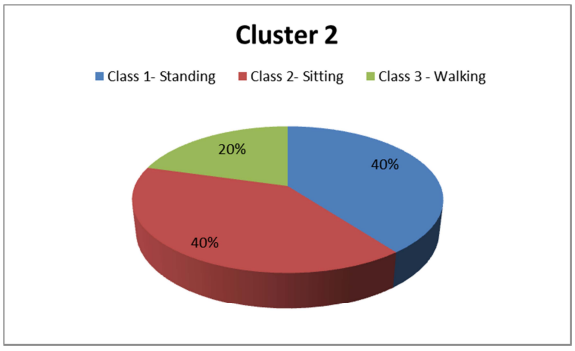
Table 5.1: An example of a confusion matrix utilizing real activity dataset.

	Class 1- Standing	Class 2- Sitting	Class 3- Walking
Cluster 1	75,00%	0,00%	25,00%
Cluster 2	20,77%	45,90%	33,33%
Cluster 3	0,00%	27,27%	72,73%

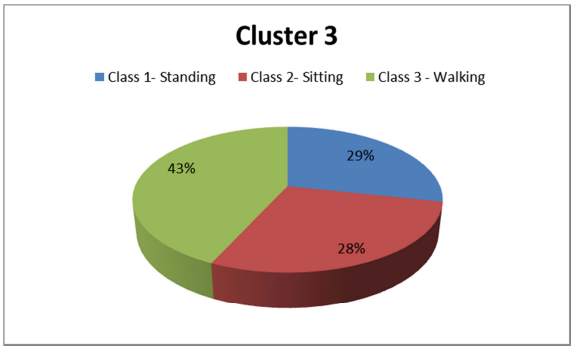
Inability to correctly cluster data is more common for complex datasets, in which population of an event type (a class) is scattered among various clusters. Figure 5.3 shows this problem in our activity dataset. As it can be seen in Figure 5.3 (a) and (b), class 1 (standing) has overlap with class 2 (sitting), as they have the same percentage of members in cluster 1 and cluster 2. In such situations it is hard to assign classes to clusters and cause in accuracy.



(a)



(b)



(c)

Figure 5.3: An example of poor clustering on the activity dataset presented in Chapter 3.

Two major reasons may be found for this problem, i.e., (i) complexity of the dataset and consequently high degree of overlap between classes so that no distinct clusters can be formed, and (ii) incapability of the classifier and its inapplicability for the given dataset. Since we cannot do much about the data complexity, our focus is on increasing the capability of the unsupervised learning techniques to better cluster data.

To gauge the accuracy of our technique we use three major metrics, namely detection rate, false alarm, and rand index. Detection rate represents correctly detected events, while false alarm represents the incorrectly detected events (misclassified events). Rand index (Eq. 5.4) is the general accuracy of data clustering showing similarity of clustering results with actual labeled data.

$$RI = \frac{TP + TN}{TP + TN + FP + FN} \qquad \text{Eq. (5.4)}$$

Where, *RI* is Rand Index ($0 \leq RI \leq 1$, 0 indicates no similarity and 1 indicates 100% similarity), TP is true positive, TN is true negative, FP is false positive and FN is false negative.

To calculate detection rate, false alarm, and rand index, we perform a number of simulations. These simulations are conducted using Matlab R2010b. The datasets were introduced and analyzed previously in Chapter 3. Table 5.2 and Table 5.3, and their graphical representation in Figures 5.4 and 5.5 present the performance evaluation of our proposed online K-means and standard K-means on Fire #1, Fire #2 and Activity datasets. All simulations are repeated for 1000 times and the mean values are reported.

Table 5.2: Detection rate and false alarm rate of standard and online K-means techniques on Fire #1, Fire #2, and Activity datasets.

	Standard K-means		Online K-means	
	Detection accuracy	False Alarm	Detection accuracy	False Alarm
Activity Dataset	50.91%	49.09%	48.58%	51.48%
Fire #1 Dataset	86.23%	13.77%	85.68%	14.32%
Fire #2 Dataset	56.80%	43.20%	57.50%	42.50%

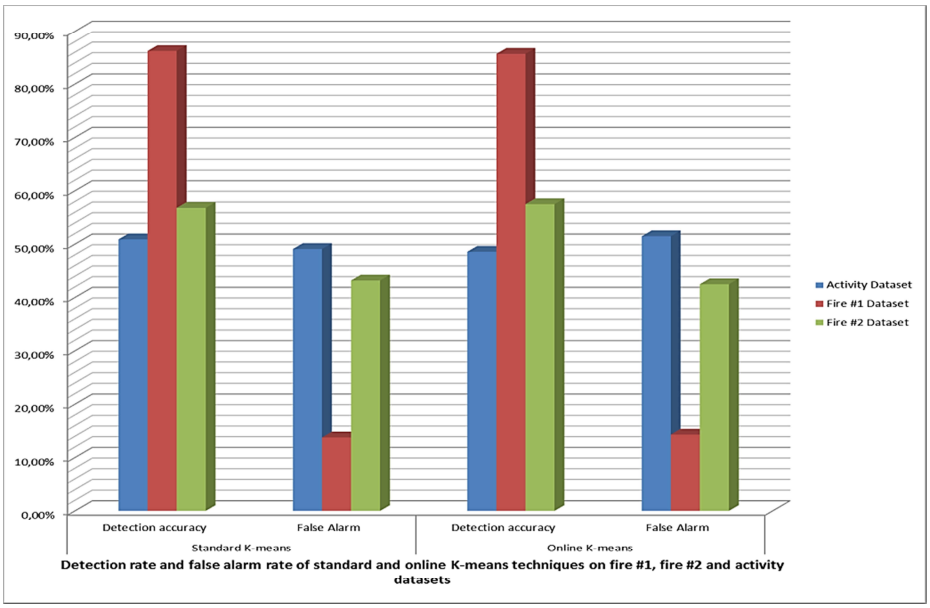


Figure 5.4: Detection rate and false alarm rate of standard and online K-means techniques on Fire #1, Fire #2, and Activity datasets.

Table 5.3: Rand Index of standard and online K-means techniques on Fire #1, Fire #2, and Activity datasets.

	Rand Index	
	Standard K-means	Online K-means
Activity Dataset	55.00%	51.96%
Fire #1 Dataset	94.82%	94.42%
Fire #2 Dataset	62.57%	63.47%

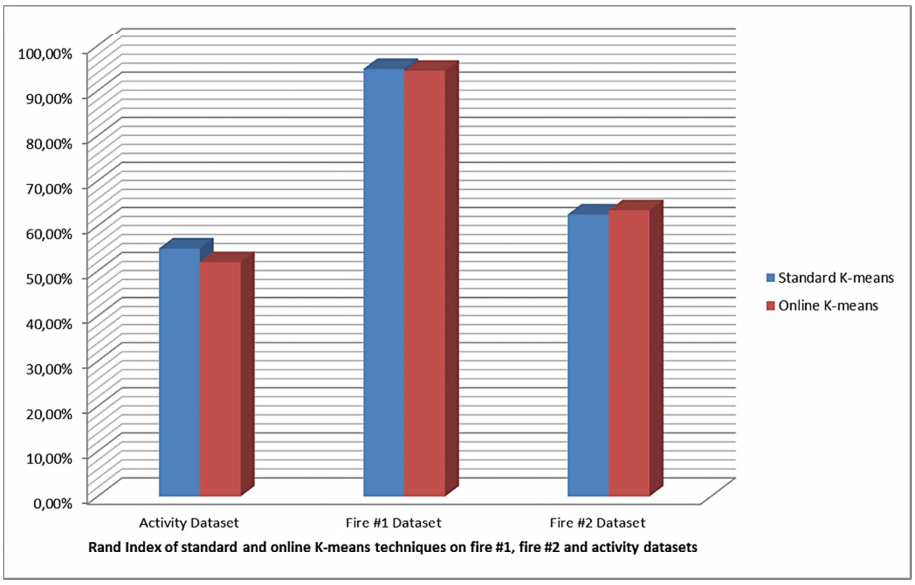


Figure 5.5: Rand Index of standard and online K-means techniques on Fire #1, Fire #2, and Activity datasets.

It can be observed that the difference between performance of our online K-means and the standard K-means is not high. On Activity and Fire #1 datasets, accuracy and false alarm of standard K-means are better. The rand index is also better. Accuracy, false alarm, and rand index of our online K-means, however, is better on Fire #2 dataset.

5.6 Computational and space complexity

Computational and space complexities are also key metrics for algorithms being developed for WSN applications as they give an insight on how resource exhaustive the algorithms are. In what follows we analyze computation and space complexity of the online and standard K-means algorithms.

5.6.1 Computational complexity

Computational complexities of the online and standard K-means in terms of the big-O notation are the same and are expressed using Eq. 5.5.

$$O(\text{K-means}) = O(k \times m) \qquad \text{Eq. (5.5)}$$

Where, k is number of cluster centers and m is number of features (sensors). For example, for fire #1 dataset, m is 4 and k is 3.

Equation 5.5 implies that when a new data instance arrives, its distance to all cluster centers is calculated in the multidimensional space and it is then assigned to the closest cluster. This part is the most time consuming part of the algorithm. Therefore, $(k \times m)$ comparisons are needed for each data instance. In standard K-means the distance

calculation is usually done using the Euclidean distance. Due to the fact that we have replaced the Euclidean distance with Manhattan distance, the distance calculation cost is considerably alleviated. To show this difference, we ran 100 time distance calculation on a HP laptop with Intel Dual Core 2,5 GHz CPU with 4 GB RAM. The average running time of distance calculation for each algorithm is reported in Table 5.4. It can be seen that calculation of the Manhattan distance is (1.44 times) faster than Euclidean. Although, Manhattan distance is faster, Euclidean distance usually results in a better accuracy (more details is available in [31]).

Table 5.4: Average time required to calculate distance functions.

Distance Measurement	Average Running Time
Manhattan distance	2.6033e-006 sec.
Euclidean distance	3.7490e-006 sec.

5.6.2 Space complexity

Standard K-means was developed for offline batch processing. Therefore, it needs to hold the entire dataset in memory. Since the online K-means is designed for sensor nodes, it is able to work with limited amount of data and therefore consumes less memory. Table 5.5 reports the space complexity of the online and the standard K-means algorithms in terms of the Big-O notation.

Table 5.5: Memory complexity of the modified K-means and the standard K-means algorithms.

Standard K-means	$f(n) = O[(n + k)m]$
Proposed algorithm	$f(n) = O(m \times k)$

Where , n is total data, m is dimension of data (number of features) and k is number of clusters (or classes). For example, for fire #1 dataset, m is 4, n is 1400, and k is 3.

It can be seen from Table 5.5 that for the standard K-means algorithm, k means (k cluster centers) and the entire data instances in an m dimensional space need to be stored inside the memory. The online K-means algorithm, however, requires only to keep the tracking table in memory. The tracking table is a table of k columns and m rows, where k is number of classes and m is number of features. Unlike the standard K-means, the space complexity of the online K-means algorithm is independent of number of inputs and requires therefore less memory.

5.7 Other possible variations of the online K-means technique

Clustering techniques differ in terms of how they (i) find similarity (or distance) between new data instances and existing clusters and (ii) update clusters upon availability of new data instances. In standard K-means the distance is (usually) calculated using Euclidean distance and the cluster centers are updated by calculating the mean value of the current cluster members. The online K-means technique uses Manhattan distance and an

accumulative averaging technique is used for updating the cluster centers. Performance of online K-means may therefore be improved by changing the distance function or updating the cluster centers differently. In what follows we investigate a number of possibilities to do so.

5.7.1 Online K-means with BELBIC (brain emotional learning based intelligent controller)

Lucas et al. [42] have developed a computational model based on the limbic system in the mammalian brain. The model, called brain emotional based intelligent controller (BELBIC), uses the network model developed by Moren and Balkenius [43] to model emotional part of brain. The block diagram of BELBIC is illustrated in Figure 5.6. BELBIC has been employed as a feedback controller in control design problems [144]. Lucas and his colleagues use the term emotional learning for the emotional control process of the BELBIC model. In emotional learning, emotions are produced by the performance of the output and are used as a reinforcement mechanism to the learning process. The emotional learning algorithm has three distinctive properties compared with other learning methodologies [145]. Firstly, one can use very complicated definitions for emotional signal without increasing the computational complexity of algorithm or worrying about differentiability or renderability into recursive formulation problems. Secondly, the parameters can be adjusted in a simple intuitive way to obtain the best performance. Finally, the training is very fast and efficient [145].

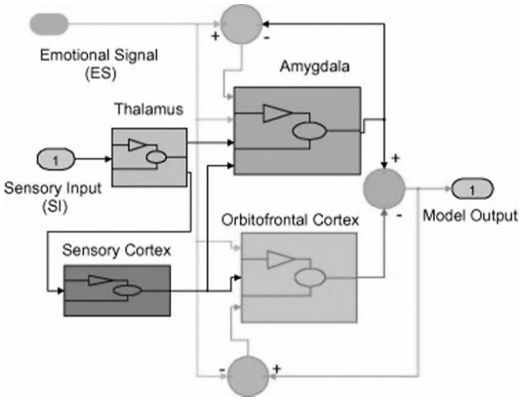


Figure 5.6: Block diagram of BELBIC [146].

BELBIC is an error reduction system that can be merged into any system to reduce the system error. We previously combined BELBIC together with neural networks in order to improve accuracy of neural networks [147]. Now we use BELBIC to update K-means cluster centers.

As mentioned earlier, BELBIC is an error reduction system. This system accepts error as its input and generates an output for lowering system error. For our online K-means algorithm, error is the distortion factor (distance between data and cluster center) that causes inaccuracy by making the clusters sparse. Our aim is to reduce distortion factor to make denser and better-shaped clusters. Finding the best updating formula, which integrates BELBIC output with the original cluster updating formula requires extensive research. We have explored many possibilities to integrate output of BELBIC with center updating formula and after performing extensive simulations have found that Eq. 5.6 is the best we could find to update the cluster centers.

$$new\ cluster\ center = \frac{[(E \times old\ cluster\ center) + X]}{1 + E} \quad \text{Eq. (5.6)}$$

Where E is output of BELBIC (Eq. 5.7) and X is the new cluster member.

$$E = \sum_i A_i - \sum_i O_i \quad \text{Eq. (5.7)}$$

Where i is the number of neurons in BELBIC and A is output of Amygdala (Eq. 5.8) and O is output of Orbitofrontal cortex (Eq. 5.9)

$$A_i = S_i \times V_i \quad \text{Eq. (5.8)}$$

Where S_i is error signal (distortion factor) and V_i is Primary Reward (Eq. 5.10)

$$O_i = S_i \times W_i \quad \text{Eq. (5.9)}$$

Where S_i is error signal (distortion factor) and W_i is output od Sensory cortex (Eq. 5.11)

$$\Delta V_i = \alpha\left(\max\left(0, S_i\left(Stress - \sum_j A_j\right)\right)\right) \quad \text{Eq. (5.10)}$$

Where, α is learning rate, S_i is error signal (distortion factor) and *Stress* is stress signal (Eq. 5.12)

$$\Delta W_i = \beta\left(S_i \sum_j (O_j - Stress)\right) \quad \text{Eq. (5.11)}$$

Where, β is learning rate, S_i is error signal (distortion factor) and *Stress* is stress signal (Eq. 5.12)

$$Stress = Error^2 \qquad \text{Eq. (5.12)}$$

Where, *Error* is the distortion rate, the Manhattan distance from new member of a cluster to its cluster center.

5.7.1.1 Simulation results

We simulated our proposed online K-means algorithm with BELBIC in Matlab and compared it with standard K-means and online K-means. Tables 5.6 and 5.7, and their graphical representation in Figures 5.7 and 5.8 report our simulation results on Fire #1 , Fire #2, and Activity datasets.

Table 5.6: Detection rate and false alarm rate of online K-means with and without BELBIC and standard K-means, on fire #1, fire #2and activity datasets.

	Online K-means with BELBIC		Online K-means		Standard K-means	
	Detection Rate	False Alarm	Detection Rate	False Alarm	Detection Rate	False Alarm
Fire #1 dataset	74.66%	25.34%	85.68%	14.32%	86.23%	13.77%
Fire #2 dataset	55,85%	44,15%	57.50%	42.50%	56.80%	43.20%
Activity dataset	47.54%	52.46%	48.58%	51.48%	50.91%	49.0942%

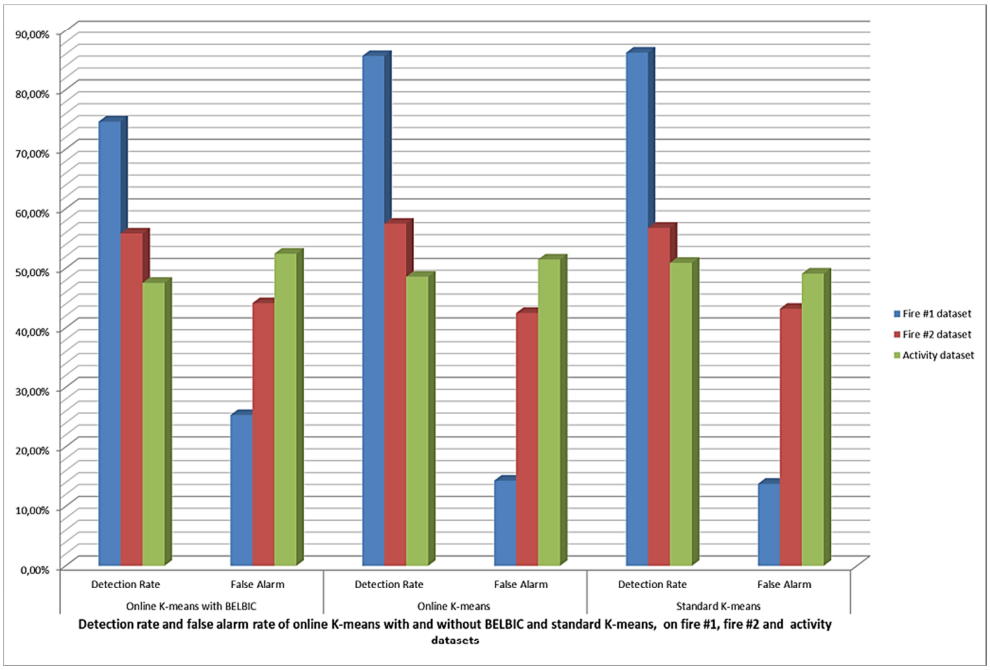


Figure 5.7: Detection rate and false alarm rate of online K-means with and without BELBIC and standard K-means on Fire #1, Fire #2, and Activity datasets.

Table 5.7: Rand Index of online K-means with and without BELBIC and standard K-means on Fire #1, Fire #2, and Activity datasets.

	Rand Index		
	Online K-means with BELBIC	Standard K-means	Online K-means
Activity Dataset	50.71%	55.00%	51.96%
Fire #1 Dataset	84.25%	94.82%	94.42%
Fire #2 Dataset	59.37%	62.57%	63.47%

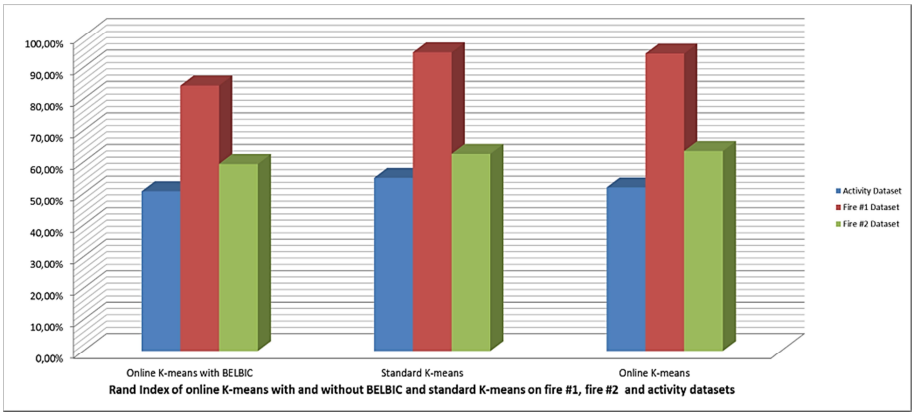


Figure 5.8: Rand Index of online K-means with and without BELBIC and standard K-means on Fire #1, Fire #2, and Activity datasets.

It can be seen that, online K-means with BELBIC has the lowest detection rate and the highest false alarm. This can be because of the best integration of BELBIC with K-means is unknown and needs further research. As mentioned earlier, BELBIC receives error signal and produces error reduction signal. This error reduction signal then can be integrated with the main systems in many ways [147]. We have tried several center updating equations and Eq. 5.6 led to our best accuracy. However, further investigation on the issue of center updating is required to improve the results.

5.7.2 Online nested K-means (K-means within K-means)

The idea behind nested K-means is to cluster data and then to do another clustering within each cluster. The motivation here is to re-cluster the misclassified clusters to improve the classification. Considering the example illustrated in Figure 5.3, the idea is to investigate whether the second round of classification on each cluster can lead to more distinct classes.

Algorithm of nested K-means is, therefore, running K-means one more time within each cluster. Number of final clusters is then K^2. For example, if K = 3 (grouping data into 3 clusters), by running nested K-means the number of clusters at the end is 9. The problem is now how to turn K^2 clusters to K clusters, because K-means should cluster data into K clusters and not more. To address this problem, we merge every 3 clusters whose cluster centers have the shortest distance to each other to end up with 3 clusters again.

5.7.2.1 Simulation results

We simulated our proposed online nested K-means algorithm in Matlab and compared it with standard K-means and online K-means. Tables 5.8 and Table 5.9 and their graphical representation in Figures 5.9 and 5.10 present our simulation results on the Fire #1 , Fire #2, and Activity datasets.

Table 5.8: Detection rate and false alarm rate of online nested K-means for Fire #1, Fire #2, and Activity datasets.

	Online nested K-means		Online K-means		Standard K-means	
	Detection Rate	False Alarm	Detection Rate	False Alarm	Detection Rate	False Alarm
Fire #1 dataset	70.50%	29.50%	85.68%	14.32%	86.23%	13.77%
Fire #2 dataset	54.31%	45.69%	57.50%	42.50%	56.80%	43.20%
Activity dataset	47.36%	52.64%	48.58%	51.48%	50.91%	49.0942%

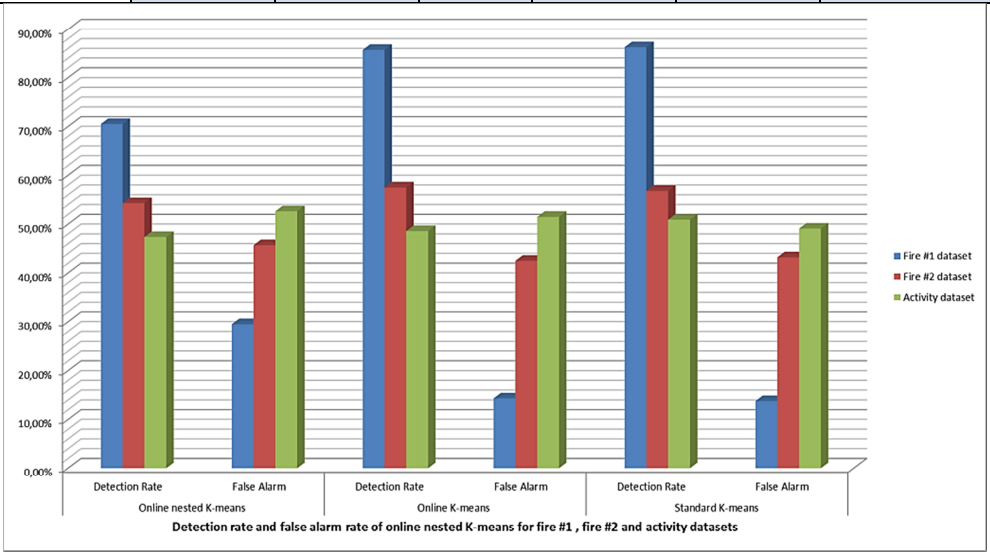


Figure 5.9: Detection rate and false alarm rate of online nested K-means for Fire #1, Fire #2, and Activity datasets.

Table 5.9: Rand Index of online nested K-means for Fire #1, Fire #2, and Activity datasets.

	Rand Index		
	Online nested K-means	Standard K-means	Online K-means
Activity Dataset	50.48%	55.00%	51.96%
Fire #1 Dataset	79.57%	94.82%	94.42%
Fire #2 Dataset	58.87%	62.57%	63.47%

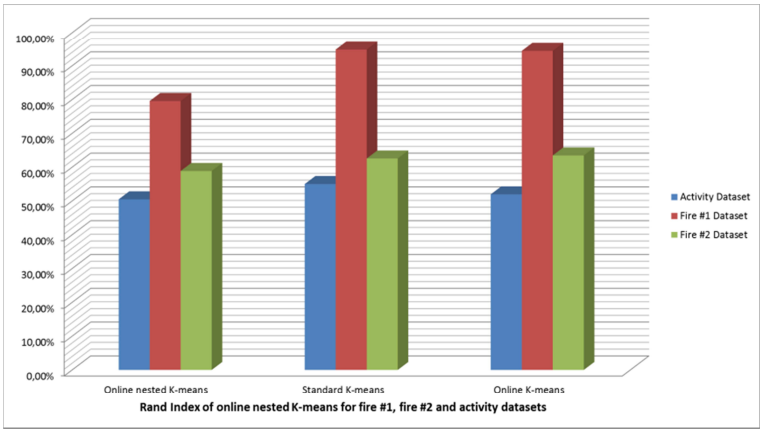


Figure 5.10: Rand Index of online nested K-means for Fire #1, Fire #2, and Activity datasets.

One may note that use of nested K-means also does not improve the results. This is because by doing the second round of clustering, error of the first round of clustering is propagated to the second round and makes the final result less accurate. This effect is more visible in simple datasets such as Fire #1, in which the accuracy drop off is higher than complex datasets such as Activity dataset. One can conclude here that simple datasets should not go through the second round of clustering.

5.7.3 Online merged K-means

The idea of online merged K-means is very similar to the nested K-means. We observed that by clustering data into K clusters and then re-clustering them into K^2 clusters, it is possible of better separate the data. However, merging K^2 clusters into K clusters is a crucial task in order to not impose more error. In online merged K-means we use an agglomerative algorithm to merge K^2 clusters into K clusters. An agglomerative algorithm is a bottom up clustering algorithm that initially considers each data as one separate cluster and then merges clusters recursively together to reach the number of desired clusters [141, 148, 149]. Therefore, K^2 distinct clusters are merged into K clusters in such a way that clusters closer to each other are merged together and make one bigger cluster.

5.7.3.1 Simulation results

We simulated our online merged K-means algorithm Matlab and compared it with standard K-means and online K-means. Table 5.10 and Table 5.11 and their graphical representation in Figures 5.11 and 5.12 present our simulation results on Fire #1, Fire #2, and Activity datasets.

Table 5.10: Detection rate and false alarm rate of online merged K-means for Fire #1, Fire #2, and Activity datasets.

	Online merged K-means		Online K-means		Standard K-means	
	Detection Rate	False Alarm	Detection Rate	False Alarm	Detection Rate	False Alarm
Fire #1 dataset	72.92%	27.08%	85.68%	14.32%	86.23%	13.77%
Fire #2 dataset	53,91%	46,09%	57.50%	42.50%	56.80%	43.20%
Activity dataset	47.88%	52.12%	48.58%	51.48%	50.91%	49.0942%

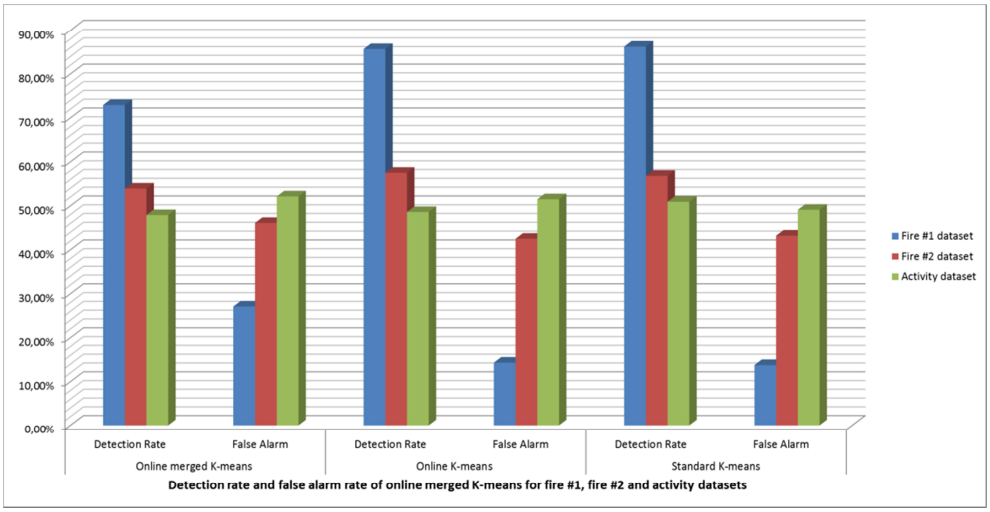


Figure 5.11: Detection rate and false alarm rate of online merged K-means for Fire #1, Fire #2, and Activity datasets.

Table 5.11: Rand Index of online merged K-means for Fire #1, Fire #2, and Activity datasets.

	Rand Index		
	Online merged K-means	Standard K-means	Online K-means
Activity Dataset	51.15%	55.00%	51.96%
Fire #1 Dataset	82.34%	94.82%	94.42%
Fire #2 Dataset	59.37%	62.57%	63.47%

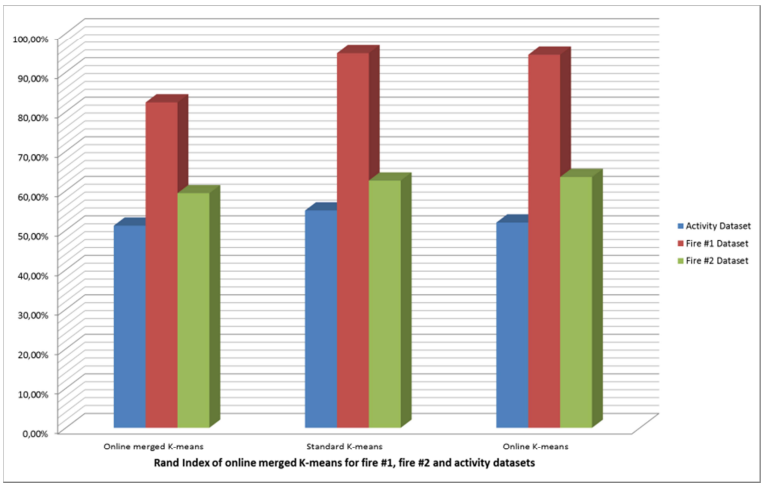


Figure 5.12: Rand Index of online merged K-means for Fire #1, Fire #2, and Activity datasets.

We observe that online merged K-means performs better than online nested K-means. However, there are still signs of error propagation, because the results are less accurate than standard K-means and online K-means. Therefore, further investigation is required for merging K^2 clusters into K clusters in order to increase the accuracy and to prevent error proliferation.

5.8 Conclusion

In this chapter, we propose local online unsupervised event detection techniques for WSNs. The well-known K-means clustering algorithm is the underlying technique of our proposed techniques. Motivation behind choosing K-means is its simplicity, which makes it a good candidate for being executed on sensor nodes. Time and space complexities of these algorithms are presented to show their applicability for WSNs. The lesson learned include:

- As expected, accuracy of unsupervised event detection is lower than supervised event detection (see Chapter 4).
- Unsupervised learning event detection techniques require availability of a semantic (event type) for each cluster to indicate the type of event being detected. Although our technique is able to detect events in an online manner, the semantic itself is found in an offline manner utilizing labeled data. To detect events by unsupervised event detection schemes in absence of labeled data, an expert system is needed to give the semantic to the clusters either online or offline (base on the requirements).
- Although a bit lower, event detection accuracy of our online K-means algorithm is very close to the standard K-means. We believe that by changing the cluster center updating formula and distance calculation, this accuracy can be improved even

more. In any case, the advantage offer by our proposed technique is its ability to detect events online and being optimized to be executed on sensor nodes.

- Increasing complexity of K-means algorithm (e.g., adding a BELBIC as Subsection 5.7.1) does not necessarily improve the accuracy of event detection. Increasing complexity of K-means can even cause accuracy drop off (as it is shown in Section 5.7).

Chapter 6

Practical Implications

As complexity rises, precise statements lose meaning and meaningful statements lose precision.

Lotfi A. Zadeh (1921) Father of fuzzy logic and founder of soft computing.

Abstract:

In Chapter 4 and 5, we discussed our proposed event detection approaches and demonstrated their applicability in simulation environment. Additionally, we calculated time and space complexities of the approaches in terms of big-O notation to demonstrate that they are resource friendly for sensor nodes. In this chapter, we analyze practical implications of our proposed event detection techniques and present their code size, execution time, and energy consumption footprints when they are implemented on TelosB sensor nodes.

6.1 Introduction

In the previous chapters, we presented our event detection approaches and evaluated them on three real datasets in Matlab simulation environment. Appropriateness of the proposed approaches are already proven through their reasonably high detection accuracies and low time and space complexities. Investigation of applicability of the proposed techniques will not be complete without evaluating their performance in a real setting and without their actual implementation on wireless sensor nodes. In this chapter, we therefore, describe our real implementation and evaluation.

Among many available types of wireless sensor node platforms, we choose TelosB sensor node for real implementation of our approaches. TelsoB is basically designed for low-power research development and lab experimentations. Additionally, it is programmed in nesC, i.e., a dialect of C language, and utilizes TinyOS, i.e. an open source embedded operating system. This open-source platform enables us to easily program our techniques on the sensor nodes.

In this chapter, we first briefly introduce TelosB sensor node platform and TinyOS operating systems in Sections 6.2 and Section 6.3, respectively. We then discuss our implementation procedure and set up in Section 6.4 and Section 6.5. In Section 6.6, we present a set of performance metrics to analyze applicability of our proposed techniques to

WSNs, while in Section 6.7 we report results of our performance evaluation. Section 6.8 concludes this chapter and presents lesson learned.

6.2 TelosB Platform

MEMSIC's TelosB is designed for low-power research development and wireless sensor network experimentation [150]. This sensor node comes in two versions, i.e., TPR2400 Processor Radio and TPR2420 Mote that are both open-source. The TelosB platform was developed and made available to the research community by UC Berkeley. The TPR2400 and TPR2420 package all the fundamentals for laboratory studies into a single platform containing USB programming capability, an IEEE 802.15.4 radio with integrated antenna, a low-power MCU with extended memory and the following features [150]:

- IEEE 802.15.4 compliant RF transceiver
- 2.4 to 2.4835 GHz, a globally compatible ISM band
- 250 kbps data rate
- Integrated Onboard Antenna
- 8 MHz TI MSP430 microcontroller with 10kB RAM
- Low current consumption
- 1MB flash for data logging
- Programming and data collection via USB
- Runs TinyOS 1.1.11 or higher

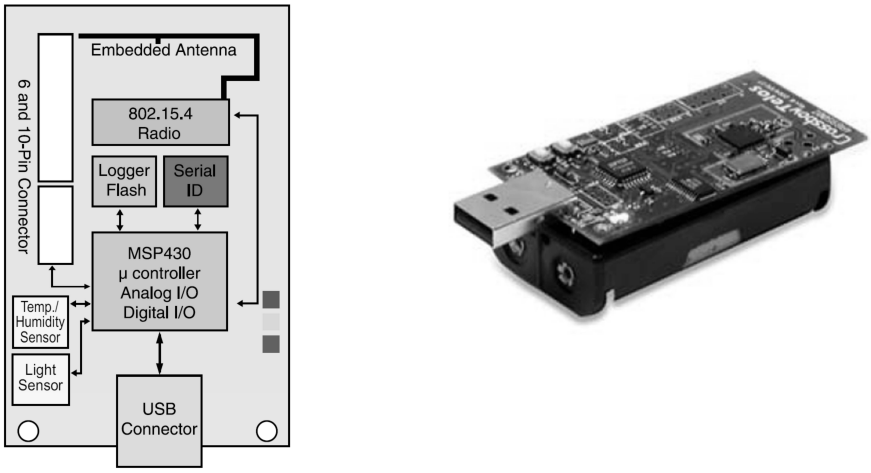


Figure 6.1: (a) Block diagram and (b) real look of MEMSIC's TelosB sensor node and (taken from [150]).

The TPR2420 comes with the integrated light, temperature and humidity sensors. Figure 6.1 (a) shows a block diagram of MEMSIC's TelosB sensor node while Figure 6.1 (b) depicts how it looks in real.

6.3 TinyOS

TinyOS is an open source, BSD-licensed operating system being designed for low-power wireless devices, such as sensor nodes [151, 152]. TinyOS is an embedded operating system, which was initially brought up through a collaboration between the UC Berkeley, Intel Research, and Crossbow Technology and has since grown to an international consortium, i.e., the TinyOS Alliance [153, 154].

TinyOS applications are developed in nesC that is optimized for the memory constraint sensor nodes. TinyOS programs are built from software components, some of which present hardware abstractions. The components are connected to each other by interfaces. TinyOS provides interfaces and components for common abstractions such as packet communication, routing, sensing, actuation and storage. TinyOS code is linked with program code and is compiled into a small binary file by a custom GNU toolchain [151, 153, 154].

6.4 Classifiers and dataset

We decide to implement our Feed Forward Neural Network (FFNN), Naïve Bayes (NB), Decision Tree (DT), and SOM K-means based event detection approaches with the fusion-based model. The motivation behind choosing the fusion-based model, in which sensor nodes communicate with each other, is to be able to get a complete footprint including communication overhead. We choose Fire #1 dataset to test these approaches. In fact, the type of dataset is not a key factor in this implementation, because our aim is to understand how much resources our techniques require to be executed by sensor nodes and what is their overall performance in terms of detection accuracy, time, and resource consumption. This will shed some light on how reliable and realistic our simulation results were. By changing the datasets all footprints, except event detection accuracy, will remain intact.

The following schemes are used in implementation of the fusion-based event detection techniques:

- FFNN: event detection is performed by FFNN classifiers distributedly and the data fuser also uses an FFNN classifier to fuse decisions and to make the final decision about occurrence of events. We consider an FFNN classifier with 100 neurons in its hidden layer (similar to simulations of Chapter 4) and Eq. 6.1 as the arithmetic formula of FFNN classifier:

$$E = tansig((\sum_{i=1}^{k} Z_i \times \left(tansig\left(\left(\sum_{j=1}^{m} X_j \times W_{j,i}\right) + B_i\right)\right)) + b_f) \quad \text{Eq. (6.1)}$$

Where, E is output of FFNN, k is number of hidden layers, Z is weight of edges from hidden layer to output layer, m is number of features (sensors) , X is input vector (sensor data) , W is weights of edges from input to hidden layer, B is bias in hidden layer, and b_f is bias in output layer.

- NB: event detection is performed by NB classifiers distributedly and the data fuser also uses a NB classifier to fuse decisions and to make the final decision about occurrence of events. We consider 180 intervals (similar to simulations of Chapter 4) for NB classifiers.
- DT: event detection is performed by DT classifiers distributedly. The data fuser uses majority voting to fuse decisions and to make the final decision about occurrence of events. Depth of DT is set to 4 for fire detection (similar to simulations of Chapter 4).
- SOM K-means: event detection is performed by SOM K-means distributedly. The data fuser uses majority voting to fuse decisions and to make the final decision about occurrence of events. SOM K-means uses a 3-by-3 self-organizing map (similar to simulations of Chapter 4).

The reasons behind using majority voting on the data fuser for DT and SOM K-means classifiers are:

- In our previous studies we investigated DT (as event detector technique) and reputation theory (as data fuser) for fusion-based event detection [8, 33] and found out this combination to be promising. Reputation theory, in certain situations (when all of the sensor nodes are identical) turns into majority voting. Therefore, having an implemented version of reputation-based data fusion is useful for further studies.
- Majority voting is simple and can also be accurate in various applications. It is handy to have a footprint of a simple voting technique for further research and investigation.

6.5 Experimental set up

In our experimental set up, we connect one sensor node, as a gateway, to a laptop through a USB connection. The gateway sends Fire #1 dataset (see Chapter 3) to four sensor nodes for fusion-based event detection. The result of event detection by four sensor nodes is then transmitted to a data fuser. The data fuser fuses the results and sends the final decision about occurrence of an event back to the gateway. Upon receipt of the final decision, the gateway transfers the result to the laptop through the USB connection. Figure 6.2 shows how we set up the sensor nodes for the experiment.

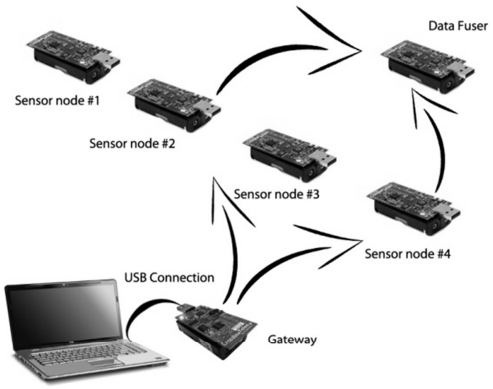


Figure 6.2: Experimental set up in the lab for testing our event detection approaches.

We developed an application in to be executed on a Linux laptop to transfer records of the dataset one by one to the gateway. When the gateway receives each record from the USB port, it broadcasts it to the sensor nodes. The gateway is also responsible to collect event detection results from the data fuser and to transfer them to the laptop. Figure 6.3 shows a snapshot of the application's GUI running on the laptop.

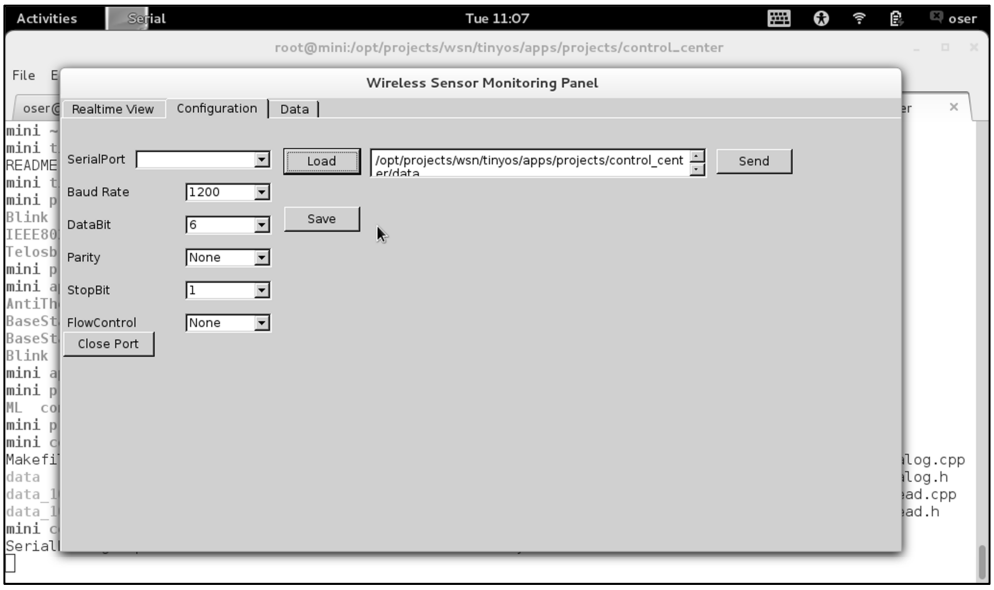


Figure 6.3: GUI of the designed application for connecting the laptop to a gateway node.

The channel access protocol for sensor nodes is Carrier Sense Multiple Access with collision detection (CSMA/CD). In CSMA/CD protocol, each device checks the medium to see whether it is idle or busy. If the medium is idle, the device begins to transmit its first frame. If another device has tried to send its frame at the same time, a collision occurs and

both frames are discarded. Each device then has to wait for a random amount of time and to retry until successfully transmits its frame [155].

Our application on the laptop side logs every transmissions by storing information such as end-to-end delay and the event decision of the data fuser.

The following considerations are applied to our experimental set up:

- If a packet is lost somewhere in the system, the whole system waits for the lost packet for 5 seconds. If the packet is not found, the same data is sent again.
- As illustrated in Figure 6.4, 16 Bytes are used to store data (suitable for sending up to four floating point data).
- The data packets from each sensor node are distinguished by their header. An ID of 0 is given to the gateway, 1 to the sensor node 1, 2 to the sensor node 2, 3 to the sensor node 3, 4 to the sensor node 4, and 5 to the data fuser. ID size is 1 Byte.
- The type of data packet, i.e., data, event detection from sensor nodes, event detection result of the data fuser, is recognizable by a 'type' field in the header.

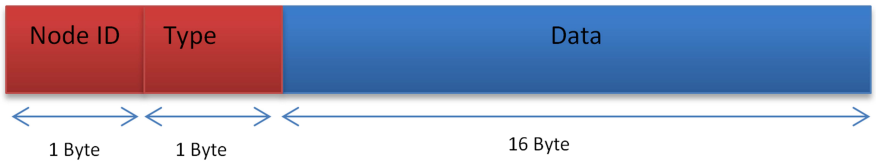


Figure 6.4: Sample packet that contains a header section (red color) and data section (blue color).

6.6 Metrics

To get the footprint of the fusion-based approaches we calculate the following metrics:

- **Processing time on sensor node:** amount of time spent by a sensor node to run a classifier to determine locally whether an event has occurred.
- **System time:** amount of time spent for the whole setup to have the event detection result reported to the laptop.
- **Code size:** size of code on a sensor node's flash memory.
- **Data size:** amount of data stored on RAM of a sensor node. Data size is calculated by summing up the space being occupied by the variables and arrays.
- **Average communication time for one time data transmission:** average time required for a packet to be sent by a sensor node (sender) and received by another sensor node (receiver).
- **Average total communication time:** Average amount of time expressed in system time which is spent for communication.

- **Processing energy cost:** energy required by a classifier for each time run (Eq. 6.2). This formula is specific to TelosB sensor nodes derived from its datasheet [150, 156].

$$Processing\ Energy\ Cost\ (nW) = Processing\ time\ (ms) \times 1.8\ (mA) \times 3000\ (mV) \quad \text{Eq. (6.2)}$$

- **Data sending energy cost:** energy that is required to send a packet (Eq. 6.3). This formula is specific to TelosB sensor nodes, where there is no other networking overhead, derived from its datasheet [150, 156].

$$Data\ Sending\ Energy\ Cost\ (\mu W) = \frac{Packet\ size\ (Bytes)}{(250\ \times 10^3)/8} \times 17.4\ (mA) \times 3000(mV) \quad \text{Eq. (6.3)}$$

- **Data receiving energy cost:** energy required for receiving a packet (Eq. 6.4). This formula is specific to TelosB sensor nodes, where there is no other networking overhead, derived from its datasheet [150, 156].

$$Data\ Receiving\ Energy\ Cost\ (\mu W) = \frac{Packet\ size\ (Bytes)}{(250\ \times 10^3)/8} \times 18.8\ (mA) \times 3000(mV) \quad \text{Eq. (6.4)}$$

6.7 Footprints and empirical results

In the previous sections we have described the lab set up and metrics to calculate footprints of the fusion-based approaches. Table 6.1 summarized processing time of a sensor node and the whole system for the event detection techniques.

Table 6.1: Footprint on processing time

	Processing time on a sensor node (in milliseconds)			System time (in milliseconds)		
	Min	Avg	Max	Min	Avg	Max
FFNN	1376	1453.60	1502	3021	3161.80	3314
NB	11	14.69	16	293	408.06	633
DT	Algorithm: 1	Algorithm: 1.91	Algorithm: 3	268	385.06	502
	Data fuser: 1	Data fuser: 1.03	Data fuser: 2			
SOM K-Means	Algorithm: 7	Algorithm: 8.02	Algorithm: 9	274	388.50	519
	Data fuser: 1	Data fuser: 1.02	Data fuser: 2			

Table 6.1 reports time required for execution of the approaches on sensor nodes for analyzing 600 data instances (except FFNN that is tested by 20 instances of data that we will discuss this issue later in this section) and reporting their min, max and average. The reason the same approach takes different time for every data instance is because of interrupts to CPU. TinyOS is a single task operating system but it is interruptible (by sources such as clock and receiver) [6, 150, 154]. Therefore, occurrence of interrupts,

specially by receivers, cause some time delay in processing task of classifier. According to Table 6.1, DT has the least processing time on a sensor node and as such is the fastest classifier while FFNN is the slowest. Additionally, we can see that majority voting technique is a fast data fusion technique for sensor nodes. The reason that FFNN is slow is the complicated arithmetic operation, including two times *tansig* calculation (Eq. 6.1), it has to do. Regarding the system time, we can see that most of the time is spent on communication rather than on data processing. This is because the system time is combination of processing time (on sensor nodes and on data fuser) and communication time between nodes. From Table 6.1 one can notice that processing time on system level is somehow insignificant, therefore, the system time is mostly spent on communication. This fact can also be witnessed in Table 6.5, which shows time required for communication is greater than processing time on sensor nodes.

Table 6.2 summarizes the energy consumption for classification task running on single sensor nodes (being a normal sensor node or the data fuser), as well as for data transition.

Table 6.2: Footprint on energy consumption.

	FFNN	NB	DT	SOM K-means
Average processing energy cost in a sensor node (in milliwatt)	7,84944 (mW)	0,07930278 (mW)	Sensor node: 0,01026216 (mW)	Sensor node: 0,0432945 (mW)
			Data fuser: 0,005562 (mW)	Data fuser: 0,00549018 (mW)
Communication energy cost (milliwatt)	0,0300672 (mW) per sending one time data (for all the classifiers) 0,0324864 (mW) per receiving one time data (for all the classifiers) 0,1876608 (mW) per onetime event detection in our experimental set up (for all the classifiers)			

According to Table 6.2, required energy for running a classifier is a function of time the classifier takes from CPU. This is also shown in Eq. 6.2. Therefore, majority voting and DT require less energy and FFNN requires more energy to be run on a sensor node. However, data transfer is not a function of classifier. Since the packet size for all classifiers is the same, sending and receiving this packet consumes equal energy for all classifiers.

Table 6.3 compares code size, data size, and detection accuracy of our implemented event detection approaches. The obtained accuracy in implementation is the same as Matlab simulation when data and classifier (and its parameters such as weights in FFNN) are the same.

Table 6.3: Footprints on memory consumption and detection accuracy.

	Code size	Data size	accuracy
FFNN	23032 Bytes	1640 Bytes	99.83%
NB	28238 Bytes	8687 Bytes	92.07%
DT	17900 Bytes	28 Bytes 50 Bytes (data fuser)	98.60%
SOM K-means	17662 Bytes	194 Bytes (sensor node) 50 Bytes (data fuser)	94.00%

From Table 6.3 one can conclude that NB takes more space for code and data. SOM K-means has the least code size and DT requires the least memory space for storing data (because of its set of if-then-else rules). Justifying detection accuracy is somewhat complicated. This is because of following reasons:

- All classifiers are tested by 600 data instances and their average accuracies are reported. However, FFNN is tested by 20 data instances. This is because in our experimental set up, the driver in Linux for connecting the gateway to the laptop stop working after sending 20 packages. This bug happens because FFNN takes long time for data processing. We could not solve this problem because it resides in the OEM driver software.
- Event detection accuracy is always data-dependent. A best working classifier for Fire #1 dataset is never best working classifier for all other kinds of datasets (for more details readers are referred to Chapter 4).

Table 6.4 reports metrics for communication time for sending a 18 Bytes data packet, as shown in Figure 6.4, from a node to another as well as the average time consumed in the system for communication (based on the aforementioned experimental set up) .

Table 6.4: Footprints on communication time.

	Average communication time for one time data transmission	Average total communication time for the experimental set up
FFNN	46.9 (ms)	234.6 (ms)
NB	71.7 (ms)	358.7 (ms)
DT	72.3 (ms)	361.6 (ms)
SOM K-means	71.1 (ms)	355.3 (ms)

As mentioned earlier, communication time and energy consumption are not dependent on the type of the classifier and are the same for all classifiers. Table 6.4 shows that the communication time is almost the same for different classifiers. However, values reported by FFNN are different than others. As mentioned earlier, this is because in our experimental set up, the driver in Linux for connecting the gateway to the laptop stop working after sending 20 packages. We could not solve this problem because it resides in the OEM driver software.

6.8 Discussion and conclusion

We implemented our Feed Forward Neural Network (FFNN), Naïve Bayes (NB), Decision Tree (DT), and SOM K-means based event detection approaches with the fusion-based model and report their implementation footprints in this chapter. This chapter provides extra evidences on applicability of our approaches to WSNs. It also provides metrics to show how time and resource exhaustive these algorithms are. The lessons learned in this chapter include:

- As it can be seen from Table 6.1, FFNN is heavy to be executed on TelosB. This is because we use *tansig* in the formula (Eq. 6.1) which is a complicated arithmetic function. By removing *tansig* or lowering down the number of neurons in the hidden layer, we can speed up execution of FFNN on sensor nodes. However, speeding up the FFNN can also cause accuracy drop off (see Chapter 4 for parameters study of FFNN).
- As it can be seen from Table 6.3, the memory footprint of NB and FFNN are higher than the rest because NB and FFNN require to hold data distribution and to store weights in memory, respectively.
- As it can be seen from Table 6.3, decision trees and SOM K-means have the lowest code and data footprints for execution. This is due to their simplicity, which leads to less space requirement.
- DT is the fastest algorithm for fire detection in terms of both processing time and total communication time (with average required processing time of 1.9 ms [see Table 6.1.] and average total communication time of 361 ms [see Table 6.4]). Using DT, in average 156 events can be detected in a minute (or 2.6 events in a second).
- Assuming that each AA batteries of sensor nodes is 1500 mAh, 1.5V and sensing by sensors (temperature and relative humidity, for example) costs 3 mW [157], detecting events by DT costs 0.01 mW (Table 6.2) and sending data costs 0.03 mW (Table 6.2). By only doing these operations nonstop, in an idle situation where there is no other networking overhead, batteries will last for 1480 hours or 61 days on a sensor node.

Chapter 7

Conclusions

If you justify one mistake with one thousand reasons, you will end up with one thousand and one mistakes!

Ibn Sina (Avicenna) (980-1037) Persian polymath.

Abstract:

This chapter concludes the thesis by presenting an overview of the thesis contributions, discussing the research achievements, and highlighting the lessons learned. At the end of this chapter we outline future directions of this research.

7.1 Thesis overview

Event detection is a significant functionality of WSNs, as it can manage huge amount of data generated by sensor nodes, and has the ability to alert when and where events of interest occur. This thesis tackles the problem of time critical event detection in WSN by proposing fast, accurate, in-network event detection techniques using artificial intelligence (AI). We summarize the content of the thesis as follows:

Chapter 1 discusses the motivation and significance of event detection functionality of WSNs in details. Motivations for conducting in-network event detection in WSNs is twofold: (i) efficiently managing data produced by sensor nodes in such a way that data resolution is preserved and the amount of communications between sensor nodes are minimized, and (ii) accurately and fast detecting events of interest. Generally, event detection should be conducted within a certain (pre-defined) timeframe. Event detection applications may pose time critical or best effort time constraints on event detection process. This thesis tackles the problem of time critical event detection in WSNs by utilizing artificial intelligence (AI) techniques. The motivation behind using AI algorithms and in-network event detection is their potential to overcome existing challenges and concerns in WSNs in terms of dealing with unreliability and resource constraints of sensor nodes as well as adaptability and heterogeneity.

Chapter 2 reviews existing event detection approaches designed for WSNs, analyzes, and classifies them based on their underlying detection techniques. Event detection techniques are classified in four main classes, i.e., (i) threshold-based, (ii) pattern matching based, (iii) model-based, and (iv) AI-based. This chapter concludes that AI-based approaches are more

prominent for event detection functionality of WSNs, and based on shortcomings of current studies, it presents a set of guidelines and future directions for design and development of event detection techniques for WSNs.

Chapter 3 introduces the datasets we use in this thesis for performance evaluation and training of classifiers. These datasets contain sensory data from real-experiments and have three levels of difficulties, i.e., easy: fire #1 (small number of event sources), medium: fire #2 (medium number of event sources), and complex: activity (highly overlapping dataset for activity recognition. This chapter brings into attention that having a large number of sensor types (features) in a dataset is not necessarily helpful and a set of most contributing features needs to be identified in this case. Feature extraction process removes less significant features and leads to a faster and more accurate event detection.

Chapter 4 elaborates our proposed AI-based supervised event detection approaches. These approaches are optimized version of decision trees, feed forward neural networks and naïve Bayes, and a new algorithm, called SOM K-means. This chapter also highlights the fact that internal parameters of these approaches have direct effect on the classification accuracy as well as time and space complexity. Therefore, the best parameter for each classifier should be chosen considering the tradeoff between classification accuracy and time and space complexity. Lastly, performance of the proposed approached is evaluated on the three datasets presented in Chapter 3 in Matlab® simulation environment.

Chapter 5 explains our proposed AI-based unsupervised event detection approaches. To enable in-network unsupervised event detection, we optimize the well-known K-means algorithm to be faster and less resource exhaustive and to be able to detect events without being trained by label data in a training phase. We evaluate performance of the proposed approached on the three datasets presented in Chapter 3 in Matlab® simulation environment. Additionally, in this chapter we conclude that there is still a long way towards achieving an accurate and fully unsupervised event detection for WSNs. There are a number of reasons for this including the fact that (i) not knowing both number of events and their patterns beforehand makes unsupervised learning approaches complicated, and (ii) finding the semantic of newly detected event autonomously is by no means straightforward.

Chapter 6 reports practical implication of implementing our supervised event detection techniques on TelosB sensor nodes and provides footprints in terms of code and memory size as well time and energy consumption required for data processing and communication. This chapter provides extra evidences on applicability of our proposed approaches to be executed on sensor nodes.

7.2 Research achievements

Our research achievements can be classified into two main categories:

- *Achievements related to analysis and optimization of AI-based algorithms for event detection in wireless sensor networks:*
    - Advancing AI research by applying it in the domain of event detection in WSNs and optimizing AI techniques for sensor nodes while fulfilling requirements of WSNs and meeting concerns such as dealing with unreliability of sensor nodes, heterogeneity, and adaptability.
    - Analyzing applicability of AI-based approaches for WSNs by providing metrics such as detection accuracy, time and space complexity on three datasets.
    - Proposing two in-network classification models for running event detection algorithms to gain the maximum accuracy while meeting challenges and concerns of WSNs.
    - Proposing a new AI-based algorithm for event detection in WSNs.
    - Analysis of practical implication of the proposed approaches to go beyond simulation environment and demonstrate the applicability of the proposed approaches for WSNs.
- *Achievements related to data selection and feature extraction.*
    - Explaining the need and presenting an approach to identify the most contributing features for event detection in WSNs to speed up the process of event detection and make the event detection more accurate.

7.3 Conclusion and lessons learned

Important lessons we have learned during this research can be summarized as the following:

- *Using artificial intelligence (AI) approaches is promising for event detection in WSNs*. This promise can be evaluated by metrics such as event detection accuracy, time and space complexity and energy consumption of event detection approaches. In chapters 4-5 we presented our AI-based approaches and demonstrated their promise for event detection in MATLAB® simulation environment. Chapter 6 provides extra evidences on their applicability to WSNs by reporting footprints of the approaches being implemented on TelosB sensor nodes.

- *Optimizing of AI approaches for resource constrained sensor nodes is challenging, but possible.* As mentioned through this thesis, AI-based approaches were originally designed for computer systems where there are plenty of resources available. Sensor nodes, however, are restricted in their resources. Therefore, the

standard AI approaches need to be optimized for sensor nodes. The optimization aims to alleviate time and space complexity of the approaches and to make them possible to be executed on sensor nodes.

- *Event detection approaches require a classification model as a carrier determine where final classification needs to be performed.* Detecting events locally by an individual sensor node or by a group of sensor nodes require a processing model, which determine how and where the final decision about occurrence of an event is made. In general, fusion-based event detection approaches provide more accurate event detection, because data is processed two times, one time on sensor nodes and one time on the data fuser.

- *There is no universally accepted best event detection approach which works best in all circumstances.* Accuracy of event detection is somewhat data-dependent. For example, A well-working event detection approach that can detect fires with 99% may not necessarily detect activities with the same accuracy. Additionally, there are some features involved in every event detection techniques (e.g., time, space complexity or energy consumption) that must be taken into consideration for choosing the right event detection approach. For example, an event detection approach which is accurate in many applications, may not be fast enough or it may takes much more space (memory) on sensor nodes. Therefore, there is always a trade-off between detection accuracy and resource consumption, which needs to be taken into account and analyzed well before an event detection technique is chosen for an application.

- *There is still a long way to be able to detect unknown events accurately in WSNs.* The advantage of using unsupervised learning techniques is that they do not need training phase and do not need labeled data, while they can find unknown events. However, as we have shown unsupervised event detections are not as accurate as supervised event detections in general. Before being fully automated, unsupervised event detection techniques still require today a number a few human intervention to find the required parameters of unsupervised learning techniques and to give "the right" semantic to events.

- *There are no universally accepted best features (sensor types) contributing most to the event detection process.* The idea of using one single sensor for event detection is not practical, as the most contributing sensor is algorithm dependent in many situations (e.g., temperature may be the most contributing sensor for fire detection using decision trees, while CO is the most contributing sensor for fire detection using neural networks). Additionally, failure of the most contributing sensor leads to failure of the event detection process. Therefore, a set of most contributing

sensors is needed to be chosen using either human expert knowledge or intelligent feature extraction methods such as genetic algorithm.

- *Finding the right dataset for event detection is very important and yet challenging.* Designing event detection approaches requires a set of data for evaluation and possibly training. While datasets play a key role in this regard, finding the right dataset is challenging and in most cases impossible. Collecting own data is also possible but time consuming. Therefore, it is best if such data is collected and made available for academic use.

7.4 Future research directions

Research presented in this thesis can be continued and extended in the following three directions:

1. *Detection of events by unsupervised learning approaches.* Supervised learning approaches are established in this thesis which are mature enough to detect even complex events. However, unsupervised learning approaches are still immature for event detection in WSNs. An unsupervised learning technique that can detect unknown events accurately and fast and is able to give semantic to events autonomously is not out there yet. This is a challenging and yet useful research topic on its own.
2. *Adaptive event detection approaches.* Supervised learning approaches can be improved by making them more adaptive to environment and context. There are some parameters in supervised learning approaches (e.g., weights in artificial neural networks) that can be adaptive when (i) the same type of events need to be detected in another environment, or (ii) a set of different events need to be detected in the same or another environment. Designing an expert system which can make supervised event detection adaptive to aforementioned situations is another futures direction of this study.
3. *Developing event detection approaches which provide meta information related to evolution of event.* Event detection approaches can detect events when and where they happen, however, as they are now they do not provide meta information related to evolution of events and when events become normal again. Real-time analysis and detection of event evolution is another interesting continuation of the research presented in this thesis.

References:

[1] S. A. Borbash, "Design considerations in wireless sensor networks," University of Maryland, 2004.

[2] F. L. Lewis, "Wireless sensor networks," *Smart Environments: Technologies, Protocols, and Applications,* pp. 11-46, 2004.

[3] D. Chen and P. K. Varshney, "QoS support in wireless sensor networks: A survey," 2004.

[4] S. Tilak*, et al.*, "A taxonomy of wireless micro-sensor network models," *ACM SIGMOBILE Mobile Computing and Communications Review,* vol. 6, pp. 28-36, 2002.

[5] G. Werner-Allen*, et al.*, "Fidelity and yield in a volcano monitoring sensor network," presented at the 7th symposium on Operating systems design and implementation (OSDI '06), 2006.

[6] R. Marin-Perianu*, et al.*, *Managing Chronic Conditions Using Wireless Sensor Networks*. Amsterdam: IOS Press, 2012.

[7] M. Bahrepour*, et al.*, "Fast and Accurate Residential Fire Detection Using Wireless Sensor Networks," *Environmental Engineering and Management Journal,* vol. 9, pp. 215-221, 2010.

[8] M. Bahrepour*, et al.*, "Distributed Event Detection in Wireless Sensor Networks for Disaster Management," presented at the International Conference on Intelligent Networking and Collaborative Systems, INCoS 2010, Thessaloniki, Greece, 2010.

[9] J. Hill*, et al.*, "System architecture directions for networked sensors," *ACM Sigplan Notices,* vol. 35, pp. 93-104, 2000.

[10] J. A. Stankovic, "Wireless sensor networks," *Computer,* vol. 41, pp. 92-95, 2008.

[11] M. I. Brownfield, "Energy-efficient wireless sensor network MAC protocol," Virginia Polytechnic Institute and State University, 2006.

[12] D. Braginsky and D. Estrin, "Rumor routing algorthim for sensor networks," 2002, pp. 22-31.

[13] J. C. Pomerol, "Artificial intelligence and human decision making," *European Journal of Operational Research,* vol. 99, pp. 3-25, 1997.

[14] D. D. Hawley*, et al.*, "Artificial neural systems: A new tool for financial decision-making," *Financial Analysts Journal,* pp. 63-72, 1990.

[15] M. L. Gargano and B. G. Raggad, "Data mining-a powerful information creating tool," *OCLC Systems & Services,* vol. 15, pp. 81-90, 1999.

[16] I. Matzkevich and B. Abramson, "Decision analytic networks in artificial intelligence," *Management Science,* vol. 41, pp. 1-22, 1995.

[17] D. Waltz, "Artificial Intelligence: realizing the ultimate promises of computing," NEC Research Institute and the Computing Research Association1996.

[18] (15-10-2012). *INTELLIPATH*. Available: http://biocare.net/products/instrumentation/intellipath-flx-automated-slide-stainer/

[19] M. M. Waldrop, *Man-Made Minds: The Promise of Artificial Intelligence*: New York: Walker and Company, 1987.

[20] R. Harkins. (2009, 15-10-2012). *The Promises of Artificial Intelligence*. Available: http://the-american-catholic.com/2009/01/16/the-promises-of-artificial-intelligence/

[21] S. Okdem and D. Karaboga, "Routing in wireless sensor networks using ant colony optimization," 2006, pp. 401-404.

[22] R. GhasemAghaei*, et al.*, "Ant colony-based reinforcement learning algorithm for routing in wireless sensor networks," 2007, pp. 1-6.

[23] Y. Sun*, et al.*, "An ant-colony optimization based service aware routing algorithm for multimedia sensor networks," *Dianzi Xuebao(Acta Electronica Sinica),* vol. 35, pp. 705-711, 2007.

[24] X. Hui*, et al.*, "A novel routing protocol in wireless sensor networks based on ant colony optimization," 2009, pp. 646-649.

[25] Y. Zhang*, et al.*, "Adaptive and online one-class support vector machine-based outlier detection techniques for wireless sensor networks," 2009, pp. 990-995.

[26] S. Rajasegarar*, et al.*, "Quarter sphere based distributed anomaly detection in wireless sensor networks," 2007, pp. 3864-3869.

[27] M. Xie*, et al.*, "Anomaly detection in wireless sensor networks: A survey," *Journal of Network and Computer Applications,* vol. 34, pp. 1302-1325, 2011.

[28] Y. Zhang*, et al.*, "Outlier detection techniques for wireless sensor networks: A survey," *Communications Surveys & Tutorials, IEEE,* vol. 12, pp. 159-170, 2010.

[29] M. Bahrepour*, et al.*, "Use of AI Techniques for Residential Fire Detection in Wireless Sensor Networks," presented at the AIAI 2009, Greece, 2009.

[30] M. Bahrepour*, et al.*, "Sensor Fusion-based Event Detection in Wireless Sensor Networks," presented at the SensorFusion, Toronto, Canada, 2009.

[31] M. Bahrepour*, et al.*, "Online Unsupervised Event Detection in Wireless Sensor Networks," presented at the Proceedings of the 7th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2011), Adelaide, Australia, 2011.

[32] M. Bahrepour*, et al.*, "Use of Wireless Sensor Networks for Distributed Event Detection in Disaster Management Applications," *International Journal of Space-Based and Situated Computing,* vol. 2, pp. 58-69, 2012.

[33] M. Bahrepour*, et al.*, "Sensor Fusion-based Activity Recognition for Parkinson Patients," in *Sensor Fusion - Foundation and Applications*, ed InTech, 2011, pp. 171-190.

[34] M. Marin-Perianu and P. Havinga, "D-FLER–a distributed fuzzy logic engine for rule-based wireless sensor networks," *Ubiquitous Computing Systems,* pp. 86-101, 2007.

[35] R. S. Michalski and G. Tecuci, *Machine Learning: A Multistrategy Approach, Volume IV*: Morgan Kaufmann, 1994.

[36] Wikipedia. (26-9-2012). *Machine learning - Wikipedia, the free encyclopedia*. Available: http://en.wikipedia.org/wiki/Machine_learning

[37] Longman. (1-6-2012). *Longman Dictionary of Contemporary English Advanced Learner's Dictionary, "Event"*. Available: http://www.ldoceonline.com/dictionary/event

[38] Oxford-Dictionaries. (1-6-2012). *Oxford Online Dictionary, "Event"*. Available: http://oxforddictionaries.com/definition/event?q=event

[39] Merriam-Webster. (1-6-2012). *Merriam-Webster Dictionary Online, "Event"*. Available: http://www.merriam-webster.com/dictionary/event

[40] J. Gupchup*, et al.*, "Model-Based Event Detection in Wireless Sensor Networks," presented at the Data Sharing and Interoperability on the World Wide Web (DSI 2007), 2007.

[41] A. Hudaya*, et al.*, "A distributed event detection scheme for wireless sensor networks," in *7th International Conference on Advances in Mobile Computing and Multimedia*, 2009, pp. 295-299.

[42] C. Lucas*, et al.*, ", Introducing belbic: Brain emotional learning based intelligent controller," *Intelligent Automation & Soft Computing,* vol. 10, pp. 11–22, 2004.

[43] J. Moren and C. Balkenius, "A computational model of emotional learning in the amygdala," presented at the 6th Int. Conf. on the Simulation of Adaptive Behaviour, 2000.

[44] I. F. Akyildiz*, et al.*, "Wirelesssensornetworks: a survey," *Computer Networks,* vol. 38, 2002.

[45] S. Vardhan*, et al.*, "Wireless integrated network sensors (WINS): distributed in situ sensing for mission and flight systems," presented at the IEEE Aerospace Conference, 2000.

[46] F. Martincic and L. Schwiebert, "Distributed Event Detection in Sensor Networks," presented at the Systems and Networks Communications (ICSNC '06), 2006.

[47] C. T. Vu*, et al.*, "Composite Event Detection in Wireless Sensor Networks," in *Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE Internationa*, 2007.

[48] A. V. U. P. Kumar*, et al.*, "Distributed Collaboration for Event Detection in Wireless Sensor Networks," in *Proceedings of the Third International Workshop on Middleware for Pervasive and Ad-hoc Computing*, France, 2005, pp. 1-5.

[49] D. Janakiram, *et al.*, "COMiS: Component oriented middleware for sensor networks," presented at the IEEE Workshop on Local Area and Metropolitan Networks (LANMAN), 2005.

[50] T. S. N. Museum. (30-7-2012). *The Sensor Network Museumtm - Tmote Sky*. Available: http://www.snm.ethz.ch/Projects/TmoteSky

[51] M. Baqer and A. I. Khan, "Event Detection in Wireless Sensor Networks Using a Decentralised Pattern Matching Algorithm," *White Paper,* 2008.

[52] C. Zhang, *et al.*, "Unspecific Event Detection in Wireless Sensor Networks," presented at the International Conference on Communication Software and Networks, Chengdu, Sichuan, China, 2009.

[53] NICTA. (26-8-2012). *Castalia: A simulator for WSNs*. Available: http://castalia.npc.nicta.com.au/

[54] A. Khelil, *et al.*, "MWM: A Map-based World Model for Wireless Sensor Networks," presented at the Autonomics, Turin, Italy 2008.

[55] M. Li, *et al.*, "Non-Threshold based Event Detection for 3D Environment Monitoring in Sensor Networks," presented at the Proceedings of the 27th International Conference on Distributed Computing Systems, 2007.

[56] I. Solis and K. Obraczka, "Efficient continuous mapping in sensor networks using isolines," presented at the MobiQuitous 2005, 2005.

[57] G. Jin and S. Nittel, "NED: An Efficient Noise-Tolerant Event and Event Boundary Detection Algorithm in Wireless Sensor Networks," in *7th International Conference on Mobile Data Management*, 2006.

[58] C. V. Easwaran, "An Efficient In-Network Event Detection Algorithm for Wireless Sensor Nodes " in *Novel Algorithms and Techniques In Telecommunications, Automation and Industrial Electronics*, ed: Springer Netherlands, 2008, pp. 318-322.

[59] J. Yin, *et al.*, "Spatio-temporal event detection using dynamic conditional random fields," in *International Joint Conference On Artificial Intelligence*, Pasadena, California, USA, 2009.

[60] G. Wittenburg, *et al.*, "A system for distributed event detection in wireless sensor networks," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2010.

[61] J. Gupchup, *et al.*, "Model-Based Event Detection in Wireless Sensor Networks," presented at the *Workshop on Data Sharing and Interoperability on the World-Wide Sensor Web (DSI)*, 2007.

[62] M. Zoumboulakis and G. Roussos, "Escalation: Complex Event Detection in Wireless Sensor Networks " in *Smart Sensing and Context*. vol. 4793, ed: Springer Berlin / Heidelberg, 2007, pp. 270-285.

[63] F. Martincic and L. Schwiebert, "Distributed Event Detection in Sensor Networks," presented at the Systems and Networks Communications, ICSNC '06., Tahiti 2006.

[64] N. Dziengel, *et al.*, "Towards distributed event detection in wireless sensor networks," presented at the *4th IEEE/ACM International Conference on Distributed Computing in Sensor Systems (DCOSS'08)*, Greece, 2008.

[65] T. M. Mitchell. (1999) Machine learning and data mining. *Communications of the ACM*. 30-36.

[66] Y. Guo, *et al.*, "EDA:Event-oriented Data Aggregation in Sensor Networks," in *IPCCC2009*, Phoenix, Arizona, USA, 2009.

[67] T. Kieu-Xuan and I. Koo, "A collaborative event detection scheme using fuzzy logic in clustered wireless sensor networks," *International Journal of Electronics and Communications (AEÜ),* vol. 65, pp. 485-488, 2011.

[68] H. Malazi, *et al.*, "FED: Fuzzy Event Detection model for Wireless Sensor Networks," *International Journal of Wireless & Mobile Networks,* vol. 3, pp. 29-45, 2012.

[69] K. Kapitanova, *et al.*, "Event Detection in Wireless Sensor Networks – Can Fuzzy Values Be Accurate?," *AD HOC NETWORKS, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering,* vol. 49, 2010.

[70] N. Wang and X. Meng, "Real-time forest fire detection with wireless sensor networks," presented at the Wireless Communications, Networking and Mobile Computing, 2005.

[71] B. Krishnamachari and S. Iyengar, "Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," *IEEE Transactions on Computers* vol. 53, pp. 241-250, 2004.

[72] M. Moradi, *et al.*, "A new method for detection of a distributed event in wireless sensor networks," presented at the Electrical Engineering (ICEE), 2011 2011.

[73] G. A. Carpenter and S. Grossberg, *Adaptive resonance theory (ART)*: MIT Press Cambridge, 1998.

[74] A. Kulakov and D. Davcev, "Tracking of unusual events in wireless sensor networks based on artificial neural-networks algorithms," presented at the Information Technology: Coding and Computing, 2005. ITCC 2005, 2005.

[75] C. E. Looa, *et al.*, "Intrusion Detection for Routing Attacks in Sensor Networks," *International Journal of Distributed Sensor Networks,* vol. 2, pp. 313-332, 2006.

[76] S. Rajasegarar, *et al.*, "Distributed Anomaly Detection in Wireless Sensor Networks," presented at the Communication systems, 2006. ICCS 2006. 10th IEEE Singapore 2006.

[77] G. I. Webb, *et al.*, "Not so naive Bayes: Aggregating one-dependence estimators," *Machine Learning,* vol. 58, pp. 5-24, 2005.

[78] M. T. Hagan*, et al.*, *Neural network design*: PWS Pub, 1996.

[79] M. Martinez-Arroyo and L. E. Sucar, "Learning an optimal naive Bayes classifier," 2006, pp. 1236-1239.

[80] N. Friedman*, et al.*, "Bayesian network classifiers," *Machine Learning,* vol. 29, pp. 131-163, 1997.

[81] T. Chard, "Self-learning for a Bayesian knowledge base: how long does it take for the machine to educate itself," *Meth Inform Med,* vol. 26, pp. 185-8, 1987.

[82] D. B. Leake, "Case-based reasoning," *The knowledge engineering review,* vol. 9, pp. 61-64, 1994.

[83] J. R. Quinlan, "Induction of decision trees," *Machine Learning,* vol. 1, pp. 81-106, 1986.

[84] S. Muggleton, "Inductive logic programming," *New generation computing,* vol. 8, pp. 295-318, 1991.

[85] S. Choi, "Gaussian Process Regression."

[86] S. Seo*, et al.*, "Gaussian process regression: Active data selection and test point rejection," 2000, pp. 241-246 vol. 3.

[87] Y. Pachepsky*, et al.*, "Use of soil penetration resistance and group method of data handling to improve soil water retention estimates," *Soil and Tillage Research,* vol. 49, pp. 117-126, 1998.

[88] K. S. Narendra and M. A. L. Thathachar, *Learning automata: an introduction*: Prentice-Hall, Inc., 1989.

[89] M. Krini, "Learning Vector Quantization."

[90] A. Sato and K. Yamada, "Generalized learning vector quantization," *Advances in neural information processing systems,* pp. 423-429, 1996.

[91] D. W. Aha, *Lazy learning*: Kluwer Academic Publishers, 1997.

[92] D. W. Aha*, et al.*, "Instance-based learning algorithms," *Machine Learning,* vol. 6, pp. 37-66, 1991.

[93] B. Li*, et al.*, "An improved k-nearest neighbor algorithm for text categorization," *Arxiv preprint cs/0306099,* 2003.

[94] R. Skousen, "An overview of analogical modeling," *Analogical modeling: An exemplar-based approach to language,* pp. 11-26, 2002.

[95] A. D. Pila and M. C. Monard, "Rules Induced by Symbolic Machine Learning Algorithms Using Rough Sets Reducts for Selecting Features: An Empirical Comparison with Other Filters."

[96] M. A. Hearst*, et al.*, "Support vector machines," *Intelligent Systems and their Applications, IEEE,* vol. 13, pp. 18-28, 1998.

[97] L. Breiman, "Random forests," *Machine Learning,* vol. 45, pp. 5-32, 2001.

[98] E. Frank and M. Hall, "A simple approach to ordinal classification," *Machine Learning: ECML 2001,* pp. 145-156, 2001.

[99] N. R. Draper and H. Smith, "Applied regression analysis (wiley series in probability and statistics)," 1998.

[100] B. Al-Salim and M. Abdoli, "Data Mining for Decision Support of the Quality Improvement Process," *AMCIS 2005 Proceedings,* p. 115, 2005.

[101] D. Agarwal, "A comparative study of artificial neural networks and info fuzzy networks on their use in software testing," 2004.

[102] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics),* vol. 28, pp. 100-108, 1979.

[103] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE,* vol. 78, pp. 1464-1480, 1990.

[104] V. Elanayar and Y. C. Shin, "Radial basis function neural network for approximation and estimation of nonlinear stochastic dynamic systems," *Neural Networks, IEEE Transactions on,* vol. 5, pp. 594-603, 1994.

[105] R. Gray, "Vector quantization," *ASSP Magazine, IEEE,* vol. 1, pp. 4-29, 1984.

[106] C. Bishop*, et al.*, "GTM: A principled alternative to the self-organizing map," *Artificial Neural Networks—ICANN 96,* pp. 165-170, 1996.

[107] N. Tishby*, et al.*, "The information bottleneck method," *Arxiv preprint physics/0004057,* 2000.

[108] M. Bahrepour*, et al.*, "Fire data analysis and feature reduction using computational intelligence methods," presented at the Advances in Intelligent Decision Technologies - Proceedings of the Second KES International Symposium IDT 2010, Baltimore, Maryland, USA, 2010.

[109] J. Stender, *Genetic Algorithms in Optimisation, Simulation and Modelling*: IOS Press; First Edition edition, 1994.

[110] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," ed: Addison-Wesley Professional; 1 edition, 1989.

[111] B. H. Liu and H. Motoda, *Feature Extraction, Construction and Selection: A Data Mining Perspective*: Springer, 1998.

[112] M. Brain. (2000). *How Smoke Detectors Work* Available: http://home.howstuffworks.com/smoke1.htm

[113] J. A. Milke, "Using Multiple Sensors for Discriminating Fire Detection," in *Fire Suppression and Detection Research Application Symposium*, 1999, pp. 150-164.

[114] D. T. Gottuk*, et al.*, "Advanced fire detection using multi-signature alarm algorithms," *Fire Safety Journal,* vol. 37, pp. 381-394 2002.

[115] J. A. Milke and T. J. McAvoy, "Analysis of signature patterns for discriminating fire detection with multiple sensors," *Fire Technology,* vol. 31, pp. 120-136, May, 1995 1995.

[116] M. Bahrepour*, et al.*, "Automatic Fire Detection: A Survey from Wireless Sensor Network Perspective," Centre for Telematics and Information Technology University of Twente, Enschede2008

[117] J. H. Holland, "Genetic Algorithms," *Scientific American,* 1992.

[118] D. A. Coley, *Introduction to Genetic Algorithms for Scientists and Engineers*: Wspc; Har/Dskt edition, 1999.

[119] Wikipedia. (20-8-2012). *Genetic algorithm*.

[120] T. M. Mitchell, *Machine Learning*: McGraw-Hill Science/Engineering/Math, 1997.

[121] E. Alpaydin, *Introduction to Machine Learning*: The MIT Press, 2009.

[122] N. J. Nilsson. (2010, 14-05-2012). *Introduction to Machine Learning*. Available: http://ai.stanford.edu/~nilsson

[123] S. Marsland, *Machine Learning: An Algorithmic Perspective*: Chapman and Hall/CRC; 1 edition 2009.

[124] H. Zhang, "The Optimality of Naive Bayes," in *Seventeenth Florida Artificial Intelligence Research Society Conference*, 2004, pp. 562–567.

[125] I. O. Software. (2004). *Self Organizing Maps Overview*. Available: http://www.improvedoutcomes.com/docs/WebSiteDocs/SOM/Overview_of_Self-Organizing_Maps_SOMs_.htm

[126] K. Gurney, *An Introduction to Neural Networks*: CRC Press, 1997.

[127] M. B. Howard Demuth, Martin Hagan, *Neural Network Toolbox, For Use with MATLAB*: The MathWorks, 2006.

[128] L. J. Madan M. Gupta, Noriyaso Homma, *Static and Dynamic Neural Networks, from Fundamentals to Advanced Theory*: IEEE Press, 2003.

[129] K. Van Laerhoven, "Combining the self-organizing map and k-means clustering for on-line classification of sensor data," *Artificial Neural Networks—ICANN 2001,* pp. 464-469, 2001.

[130] P. E. H. Richard O. Duda, David G. Stork, *Pattern Classification (2nd Edition)* Wiley-Interscience; 2 edition 2000.

[131] A. Ng. (8-7-2012). *The Concept of Unsupervised Learning*. Available: http://www.academicearth.org/lectures/the-concept-of-unsupervised-learning

[132] K. Barlett, "Application of unsupervised learning methods to an author separation task," Faculty of Rensselaer Polytechnic Institute, Troy, New York, July 2008.

[133] C.-H. Leea and H.-C. Yangb, "Construction of supervised and unsupervisedlearning systems for multilingual text categorization," *Expert Systems with Applications,* vol. 36, 2007.

[134] R. L. King*, et al.*, "Using unsupervisedlearning for feature detection in a coal mine roof," *Engineering Applications of Artificial Intelligence,* vol. 6, 2003.

[135] K. A. J. Doherty*, et al.*, "Unsupervisedlearning with normalised data and non-Euclidean norms," *Applied Soft Computing,* vol. 7, 2004.

[136] A. R. Abas, "Unsupervisedlearning of mixture models based on swarm intelligence and neural networks with optimal completion using " *Egyptian Informatics Journal,* vol. In Press, 2012.

[137] J. A. Hartigan, *Clustering Algorithms*: John Wiley & Sons, Inc. New York, NY, USA, 1975.

[138] G. McLachlan and K. E. Basford, *Mixture Models*: NewYork: Marcel Dekker, 1987.

[139] S. C. Johnson, "Hierarchical clustering schemes," *PSYCHOMETRIKA,* vol. 32, 1967.

[140] G. A. Carpenter and S. Grossberg, "Adaptive Resonance Theory: Neural Network Architectures for self-organizing pattern recognition," presented at the Parallel Processing in Neural Systems and Computers (10th Cybernetics DGK), North Holland, Amsterdam, 1990.

[141] G. S. Linoff and M. J. Berry, *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management*: Wiley Computer Publishing; 3 edition (April 12, 2011), 2011.

[142] J. B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations," in *5th Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, 1967, pp. 281–297.

[143] D. E. Rumelhart and J. L. McClelland, "Parallel distributed processing: explorations in the microstructure of cognition. Volume 1. Foundations," 1986.

[144] A. R. Mehrabian*, et al.*, "Design of an aerospace launch vehicle autopilot based on optimized emotional learning algorithm," *Cybernetics and Systems,* vol. 39, pp. 284–303, 2008.

[145] T. Babaie*, et al.*, "Prediction of solar conditions by emotional learning," *Intelligent Data Analysis,* vol. 10, pp. 583–597, 2006.

[146] B. M. Dehkordi*, et al.*, "Sensorless speed control of switched reluctance motor using brain emotional learning based intelligent controller," *Energy Conversion and Management,* vol. 52, 2011.

[147] T. E. Kalayci*, et al.*, "How Wireless Sensor Networks Can Benefit from Brain Emotional Learning Based Intelligent Controller (BELBIC)," presented at the 2nd International Conference on Ambient Systems, Networks and Technologies (ANT-2011), Niagara Falls, Canada, 2011.

[148] G. Govaert, *Data Analysis*: Wiley-ISTE; 1 edition (October 5, 2009), 2009.

[149] C. D. Manning*, et al.* (2008, 11-7-2012). *Introduction to Information Retrieval*. Available: http://nlp.stanford.edu/IR-book/html/htmledition/hierarchical-agglomerative-clustering-1.html

[150] CrossbowTechnology. (5-8-2012). *TelosB Datasheet*. Available: http://bullseye.xbow.com:81/Products/productdetails.aspx?sid=252

[151] P. Levis and D. Gay, *TinyOS Programming* Cambridge University Press; 1 edition, 2009.

[152] T. D. Team. *TinyOS Home Page*. Available: http://tinyos.net/

[153] Wikipedia. TinyOS [Online]. Available: http://en.wikipedia.org/wiki/TinyOS

[154] TinyOS. *TinyOS Documentation Wiki*. Available: http://docs.tinyos.net/tinywiki/index.php/Main_Page

[155] H. Wu and Y. Pan, *Medium Access Control in Wireless Networks*: Nova Publishers, 2008.

[156] TexasInstruments. (6-8-2012). *ZigBee/802.15.4 Modules 2483.5MHz 250Kbps 48-Pin VQFN EP T/R*. Available: http://avnetexpress.avnet.com/store/em/EMController/ZigBee/Texas-Instruments/CC2420-RTR1/_/R-4242369/A-4242369/An-0?action=part&catalogId=500201&langId=-1&storeId=500201

[157] SensirionInc, "Datasheet SHT1x (SHT10, SHT11, SHT15)," ed: SENSIRION AG, Switzerland, 2011.

[158] Mitchell, Tom Michael. The discipline of machine learning. Carnegie Mellon University, School of Computer Science, Machine Learning Department, 2006.

Publications:

Journals

- Bahrepour, M. and Meratnia, N. and Poel, M. and Taghikhaki, Z. and Havinga, P.J.M. (2012) **Use of wireless sensor networks for distributed event detection in disaster management applications**. International Journal of Space-Based and Situated Computing, 2 (1). pp. 58-69. ISSN 2044-4893
- Bahrepour, M. and Meratnia, N. and Havinga, P.J.M. (2010) **Fast and Accurate Residential Fire Detection Using Wireless Sensor Networks**. Environmental Engineering and Management Journal, 9 (2). pp. 215-221. ISSN 1582-9596

Book Chapter

- Bahrepour, M. and Meratnia, N. and Taghikhaki, Z. and Havinga, P.J.M. (2011) **Sensor Fusion-based Activity Recognition for Parkinson Patients**. In: Sensor Fusion - Foundation and Applications. InTech, pp. 171-190. ISBN 978-953-307-446-7

Peer reviewed conferences and workshops

- Bahrepour, M. and Meratnia, N. and Havinga, P.J.M. (2011) **Online Unsupervised Event Detection in Wireless Sensor Networks**. In: Proceedings of the 7th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2011), 6-9 Dec 2011, Adelaide, Australia. pp. 306-311. IEEE Computer Society. ISBN 978-1-4577-0673-8
- Kalayci, T.E. and Bahrepour, M. and Meratnia, N. and Havinga, P.J.M. (2011) **How Wireless Sensor Networks Can Benefit from Brain Emotional Learning Based Intelligent Controller (BELBIC)**. In: 2nd International Conference on Ambient Systems, Networks and Technologies (ANT-2011), 19-21 Sep 2011, Niagara Falls, Canada. pp. 216-223. Procedia Computer Science 5. Elsevier. ISSN 1877-0509
- Zwartjes, G.J. and Bahrepour, M. and Havinga, P.J.M. and Hurink, J.L. and Smit, G.J.M. (2011) **On the Effects of Input Unreliability on Classifion Algorithms**. In: 8th International ICST Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services, MobiQuitous 2011, 6-9 Dec 2011, Copenhagen, Denmark. pp. 126-137. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST) 104. Springer Verlag. ISSN 1867-8211 ISBN 978-3-642-30972-4

- Bahrepour, M. and Meratnia, N. and Poel, M. and Taghikhaki, Z. and Havinga, P.J.M. (2010) **Distributed Event Detection in Wireless Sensor Networks for Disaster Management**. In: International Conference on Intelligent Networking and Collaborative Systems, INCoS 2010, 24-26 Nov 2010, Thessaloniki, Greece. pp. 507-512. IEEE Computer Society. ISBN 978-0-7695-4278-2
- Bahrepour, M. and van der Zwaag, B.J. and Meratnia, N. and Havinga, P.J.M. (2010) **Fire data analysis and feature reduction using computational intelligence methods**. In: Advances in Intelligent Decision Technologies - Proceedings of the Second KES International Symposium IDT 2010, 28-30 July 2010, Baltimore, Maryland, USA. pp. 289-298. Smart Innovation, Systems and Technologies 4. Springer-Verlag. ISSN 2190-3018 ISBN 978-3-642-14615-2
- Bahrepour, M. and Meratnia, N. and Havinga, P.J.M. (2009) **Sensor Fusion-based Event Detection in Wireless Sensor Networks**. In: SensorFusion, 13-16 July 2009, Toronto, Canada. pp. 1-8. IEEE. ISBN 978-963-9799-59-2
- Bahrepour, M. and Meratnia, N. and Havinga, P.J.M. (2009) **Use of AI Techniques for Residential Fire Detection in Wireless Sensor Networks**. In: AIAI 2009 Workshop Proceedings, 23-25 April 2009, Greece. pp. 311-321. ceur-ws.org. ISSN 1613-0073
- Bahrepour, M. and Zhang, Yang and Meratnia, N. and Havinga, P.J.M. (2009**) Use of Event Detection Approaches for Outlier Detection in Wireless Sensor Networks**. In: Proceedings of Symposium on Theoretical and Practical Aspects of Large-scale Wireless Sensor Networks, The 5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing 2009 (ISSNIP 2009), 7-10 Dec 2009, Melbourne, Australia. pp. 439-444. IEEE Press. ISBN 978-1-4244-3518-0
- Bahrepour, M. and Mahdipour, E. and Cheloi, R. and Yaghoobi, M. (2008) **SUPER-SAPSO: A New SA-Based PSO Algorithm**. In: Applications of Soft Computing: From Theory to Praxis, 10-28 Nov 2008. pp. 423-430. Advances in Itelligent and Soft Computing 58. Springer Verlag. ISSN 1867-5662 ISBN 978-3-540-89618-0
- Davarynejad, M. and Sedghi, S. and Bahrepour, M. and Ahn, C.W. and Akbarzadeh, M. and Coello Coello, C. A. (2008) **Detecting Hidden Information from Watermarked Signal using Granulation Based Fitness Approximation**. In: Applications of soft Computing: From Theory to Praxis, 10-28 Nov 2008. pp. 463-472. Advances in Itelligent and Soft Computing 58. Springer Verlag. ISSN 1867-5662 ISBN 978-3-540-89618-0

Technical report

- Bahrepour, M. and Meratnia, N. and Havinga, P.J.M. (2008) **Automatic Fire Detection: A Survey from Wireless Sensor Network Perspective**. Technical Report TR-CTIT-08-73, Centre for Telematics and Information Technology University of Twente, Enschede. ISSN 1381-3625