Applying Machine Learning Techniques to Wireless Sensor Networks: Methods and Use Cases

Heta Desai

Abstract

Wireless sensor networks (WSNs) are used to monitor environments that often undergo rapid and unpredictable changes. These variations can stem from external influences or be introduced intentionally by system designers. To effectively respond to such dynamic conditions, WSNs frequently employ machine learning techniques, which reduce the need for frequent manual reconfiguration. Machine learning also contributes to efficient resource usage and helps extend the operational life of these networks.

This paper presents a comprehensive review of machine learning approaches applied to WSNs from 2002 to 2013. It examines how various algorithms have been used to tackle key challenges in WSNs and evaluates their strengths and limitations in relation to each problem.
Additionally, a comparative overview is provided to help WSN developers select appropriate machine learning methods tailored to their specific application needs.

Keywords—Wireless sensor networks, machine learning, data mining, security, localization, clustering, data aggregation, event detection, query processing, data integrity, fault detection, medium access control, compressive sensing.

1. Introduction

A Wireless Sensor Network (WSN) generally consists of numerous small, autonomous,
low-cost, and energy-efficient sensor nodes. These nodes are designed to collect data from their surrounding environment and collaborate to transmit this information to a centralized system, known as a base station or sink, for further analysis. Depending on the application, sensor nodes may be equipped with various types of sensors—such as thermal, acoustic, chemical, pressure, weather, or optical—enabling WSNs to support a wide range of powerful and diverse applications, each with unique requirements and characteristics.

Creating effective algorithms that work well across different WSN applications is a complex task. Designers must overcome key challenges such as data aggregation, ensuring data reliability, node localization, clustering, energy-efficient routing, event scheduling, fault detection, and maintaining network security.

Machine Learning (ML), originally introduced in the late 1950s as part of artificial intelligence (AI), has since evolved to focus on algorithms that are both computationally efficient and resilient. In the past decade, ML techniques have been widely applied across various domains for tasks like classification, regression, and density estimation. These applications span areas such as bioinformatics, speech recognition, spam filtering, computer vision, fraud detection, and digital advertising. The development of ML methods has been influenced by multiple disciplines, including statistics, mathematics, neuroscience, and computer science.

The core concept of machine learning is well captured by two foundational definitions:

1. The creation of computer models that simulate learning processes to enhance system performance and address the challenge of acquiring knowledge.

2. The use of computational techniques to boost machine performance by identifying and describing patterns within training data.

When applied to Wireless Sensor Networks (WSNs), these definitions highlight the potential of machine learning to leverage past data in order to improve how sensor networks perform specific tasks—without the need for manual reprogramming. Machine learning plays a vital role in WSNs for several key reasons:

1. **Adaptability in dynamic environments:** WSNs often operate in settings that change rapidly, such as nodes shifting position due to soil erosion or ocean currents. It is important to develop systems that can adapt to these unpredictable changes and maintain efficient operation.
2. **Support for exploratory applications:** WSNs are frequently deployed in hazardous or inaccessible locations like volcanic regions or polluted waters to gather new information. Since initial system designs may not account for all unexpected behavior, machine learning can offer adaptive solutions that recalibrate based on newly observed data.
3. **Modeling complex environments:** Many WSN deployments occur in scenarios where creating accurate mathematical models is difficult or infeasible. Even in cases where simple models exist, solving problems like routing can still require sophisticated algorithms. Machine learning offers efficient, low-complexity alternatives to traditional modeling.
4. **Insight extraction from large datasets:** WSNs generate significant amounts of data, but identifying meaningful correlations within this data is challenging. For example, applications must balance communication efficiency, energy use, and coverage. Machine learning can analyze sensor data to uncover important patterns and optimize sensor deployment to maximize data coverage.
5. **Integration with emerging technologies:** WSNs are increasingly integrated with modern technologies like cyber-physical systems (CPS), machine-to-machine (M2M) communication, and the Internet of Things (IoT). These systems aim to enable smarter decision-making and automation. Machine learning is crucial in this context, as it helps extract layered abstractions from data, enabling intelligent behavior with minimal human involvement.

Despite the many benefits of applying machine learning in wireless sensor networks (WSNs), there are several limitations and challenges that must be taken into account:

1. **Resource constraints:** WSNs are limited in terms of energy and computational power. Running machine learning algorithms to achieve high prediction accuracy or discover complex relationships in data can significantly deplete energy resources. Therefore, designers must balance the trade-off between the model's accuracy and its computational and energy demands. In some cases, offloading the learning process to centralized, more capable computing units might be a more practical solution.
2. **Need for large datasets:** Machine learning typically requires a substantial amount of training data to generalize well and achieve low error rates. Additionally, developers have limited control over how the algorithm formulates and interprets the acquired knowledge, which may impact the system's performance and reliability.

Over the last decade, there has been a growing interest in applying advanced machine learning methods to WSNs. For example, [11] provided a brief review of algorithms aimed at enhancing information processing and network efficiency. Similarly, [12] explored machine learning applications in wireless ad-hoc networks, while [13] examined the use of reinforcement learning, neural networks, and decision trees across various layers of WSN communication.

More focused studies have also emerged, like [14] and [15], which investigated effective outlier detection strategies—many of which use machine learning concepts. Furthermore, [16] delved into computational intelligence methods for solving typical WSN problems like data fusion, routing, task scheduling, optimal sensor placement, and localization. Computational intelligence, a subfield of machine learning, includes biologically-inspired techniques such as neural networks, fuzzy logic, and evolutionary algorithms [17].

Early surveys mainly emphasized reinforcement learning, neural networks, and decision trees due to their practical efficiency and theoretical appeal. However, this paper takes a broader approach by analyzing a diverse set of modern machine learning algorithms, categorizing them into supervised, unsupervised, and reinforcement learning types. What sets this work apart from earlier reviews is its focus on WSN-specific challenges, linking each algorithm to relevant application areas to better support the adoption of these techniques. Moreover, the paper not only summarizes past research but also provides actionable insights and guidance for WSN researchers and engineers aiming to explore new machine learning directions in future developments.

The structure of the remainder of this paper is as follows:

- **Section II** provides an overview of key machine learning algorithms and concepts that will be referenced throughout the paper. Basic examples are included to demonstrate how these techniques apply to wireless sensor networks (WSNs).
- **Section III** explores how machine learning has been used to solve core functional challenges in WSNs. These functional issues are critical to the network's primary operations and include areas such as routing, localization, clustering, data aggregation, query handling, and medium access control.
- **Section IV** focuses on machine learning approaches that address non-functional requirements in WSNs. These are aspects that influence the quality or enhance the performance of the network's core functions—such as security, quality of service (QoS), and data integrity. This section also highlights innovative applications of machine learning in specialized WSN scenarios.
- **Section V** discusses significant challenges and open questions that remain in applying machine learning to WSNs, pointing to opportunities for future research.
- **Section VI** concludes the paper with a summary and offers a comparative guide of effective machine learning paradigms to support ongoing and future research efforts in different WSN applications.

2. INTRODUCTION TO MACHINE LEARNING IN WIRELESS SENSOR NETWORKS

Sensor network designers often view machine learning as a set of tools and techniques used to develop predictive models. However, for machine learning specialists, the field is much broader, encompassing a wide range of underlying themes and theoretical frameworks. Gaining an understanding of these deeper concepts can be highly valuable for those aiming to apply machine learning within wireless sensor networks (WSNs). In various WSN applications, machine learning offers a high degree of flexibility and adaptability.

This section outlines key theoretical ideas and strategies for integrating machine learning into WSNs. Machine learning algorithms are generally classified based on the structure of the model they aim to create. The majority fall into one of three main categories: supervised learning, unsupervised learning, and reinforcement learning [18].

In **supervised learning**, algorithms are trained on labeled data, meaning both inputs and their corresponding outputs are known. The goal is to model the relationship between inputs, outputs, and system parameters. **Unsupervised learning**, on the other hand, operates without labeled outputs. Here, the primary aim is to group data into clusters based on similarities among the input samples.

Reinforcement learning represents a third category where an agent learns optimal behavior through interaction with its environment, receiving feedback in the form of rewards or penalties—essentially learning online.

Some algorithms don't fit neatly into these categories and instead combine elements from both supervised and unsupervised approaches. These are referred to as **semi-supervised learning** methods, designed to leverage the strengths of both techniques while addressing their limitations [19].

This section's purpose is to introduce readers to these foundational algorithm types, as they will be referenced throughout the paper. Examples are provided here to illustrate how machine learning can be applied in WSNs. However, detailed explanations will be omitted in Sections III and IV. Readers seeking a deeper dive into the theoretical background of machine learning can consult sources such as [18], [20], and the references cited within.

A. Supervised Learning

Supervised learning involves training a model using a labeled dataset—where both the inputs and their corresponding outputs are known. The goal is to construct a model that captures the relationship between these inputs, outputs, and any underlying system parameters. In wireless sensor networks (WSNs), supervised learning techniques are widely adopted to address various challenges. These include tasks such as **localization and target tracking** ([21]–[23]), **event detection and query handling** ([24]–[27]), **media access control** ([28]–[30]), **network security and intrusion detection** ([31]–[34]), and ensuring **quality of service (QoS), data integrity, and fault detection** ([35]–[37]).

1) K-nearest neighbor (k-NN):

The k-NN algorithm is a simple yet effective supervised learning technique that classifies an unknown data point (also called a query point) by examining the labels of its nearest neighbors. For instance, if a sensor node has missing data, k-NN can estimate the value using the average readings from nearby sensors within a defined range.

There are various ways to identify the "nearest" sensors, with the **Euclidean distance** being a common and straightforward metric. Since k-NN focuses on nearby data points and involves low computational overhead, it is well-suited for distributed learning environments like WSNs, where resources are limited and sensor readings are often spatially correlated.

However, studies such as [38] have pointed out that k-NN's accuracy can suffer in **high-dimensional datasets** (more than 10–15 features), where distances between points tend to lose significance—the closest and farthest neighbors start to appear equally distant. In the context of WSNs, k-NN is especially useful for **query processing**, as demonstrated in works like [24] and [25].

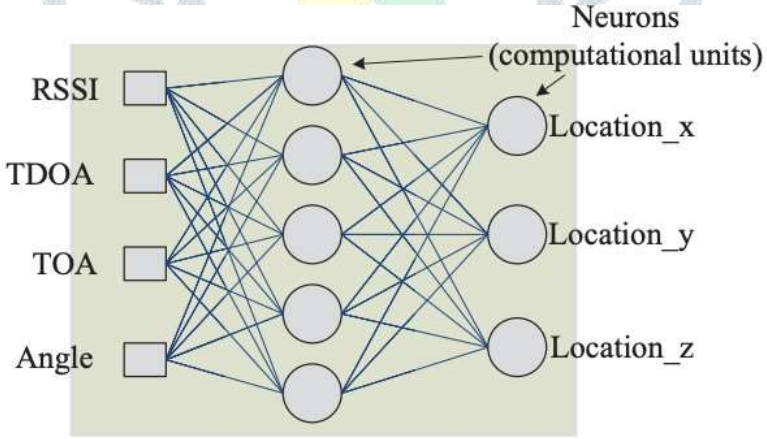

Fig. 1: Illustration example of node localization in WSNs in 3D space using supervised neural networks.

2) Decision Tree (DT):

A decision tree is a classification technique that predicts the label of a data point by guiding it through a tree-like structure. At each node of the tree, the input features are evaluated against specific conditions, eventually leading to a final classification. Decision trees have been widely applied in wireless sensor network (WSN) design due to their simplicity and effectiveness. For instance, DTs can be used to estimate link reliability by focusing on key parameters like **packet loss rate**, **corruption rate**, **mean time to failure (MTTF)**, and **mean time to restore (MTTR)**. However, one limitation is that DTs perform well primarily with **linearly separable data**, and

constructing an optimal decision tree is an **NP-complete** problem [40], making it computationally challenging in some cases.

3) Neural Networks (NNs):

Neural networks consist of layers of interconnected processing units (such as **perceptrons** or **radial basis functions**) and are capable of learning complex, non-linear relationships within data [9]. In WSNs, deploying NNs in a **distributed** fashion is still limited, mainly due to the **intensive computation** required for training and updating network weights, along with the **management complexity**. Despite this, **centralized implementations** of NNs can be very powerful, as they are able to simultaneously learn multiple decision functions and output mappings [41], making them ideal for addressing several network-related problems with a single unified model.

We examine the **sensor node localization** problem—identifying the physical coordinates of a node—as an example of how **neural networks** can be applied in wireless sensor networks (WSNs). Localization can rely on signal measurements such as **Received Signal Strength Indicator (RSSI)**, **Time of Arrival (TOA)**, and **Time Difference of Arrival (TDOA)**, which are transmitted from fixed-position **anchor nodes** [42]. These signal properties help estimate the unknown node's position. After a period of supervised learning, the neural network can predict the node's location as a set of **3D spatial coordinates**. Related neural network techniques include **Self-Organizing Maps (SOMs)**, also known as **Kohonen maps**, and **Learning Vector Quantization (LVQ)** [43].

Beyond estimating functions like position, neural networks are particularly useful for handling **large-scale, high-dimensional data**, helping in **tuning** and **dimensionality reduction** tasks [44].

4) Support Vector Machines (SVMs):

SVM is a supervised learning method that classifies data points based on a set of labeled examples [45]. In WSNs, SVMs can be applied to **detect malicious node behavior** by analyzing the **temporal** and **spatial** patterns in the data, helping to distinguish between normal and suspicious activities.

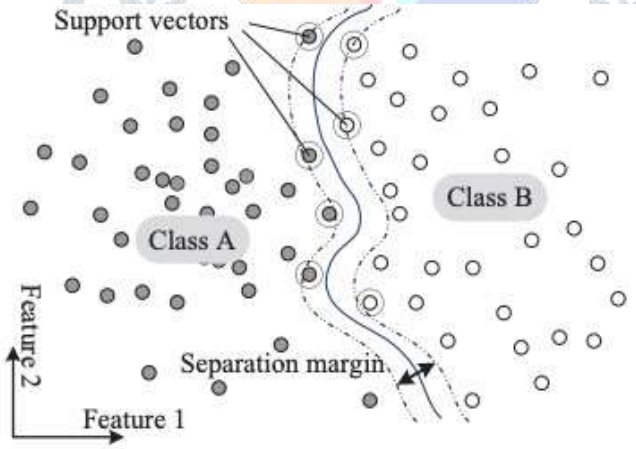

Fig. 2: An example of non-linear support vector machines.

To explain how it works, when observations from a WSN are represented as points in a **feature space**, a **Support Vector Machine (SVM)** algorithm splits this space into distinct regions using **margins**—which are as wide as possible—to separate different classes. Any new data point is then classified based on which side of the margin it falls, as shown in Figure 2. The process involves solving a **quadratic optimization problem** with **linear constraints**, effectively constructing a set of **hyperplanes**. This makes SVMs a strong alternative to **multi-layer neural networks**, which typically involve **non-convex** and **unconstrained optimization** problems [39].

In WSNs, **SVMs** have proven useful for applications like **security** ([33], [34], [46], [47], [48]) and **localization** ([49], [50], [51]). For an in-depth understanding of the underlying theory, refer to [45].

5) Bayesian Statistics:

Unlike many machine learning approaches, **Bayesian inference** can function effectively with **a relatively small set of training samples** [52]. It uses **probability distributions** to learn from uncertain data, helping avoid overfitting. The key concept is updating **prior knowledge** using new observations. Given observed data **D** and a parameter **θ**, the **posterior probability** is calculated as:

$$p(\theta|D) \propto p(\theta) \cdot p(D|\theta)$$

Where:

- **$p(\theta)$** is the prior probability of the parameter
- **$p(D|\theta)$** is the likelihood of the data given the parameter
- **$p(\theta|D)$** is the posterior probability after observing the data

In WSNs, Bayesian methods can be applied to **evaluate event consistency** (θ) using **incomplete or partial data** (D), relying on prior environmental knowledge. However, the reliance on such statistical knowledge limits how broadly Bayesian techniques can be used in WSNs. A related method is **Gaussian Process Regression (GPR)** [53].

B. Unsupervised Learning

In **unsupervised learning**, algorithms work **without labeled data**—meaning there are no predefined output values. The main goal is to **group data samples** based on their **similarity**, uncovering hidden patterns or clusters.

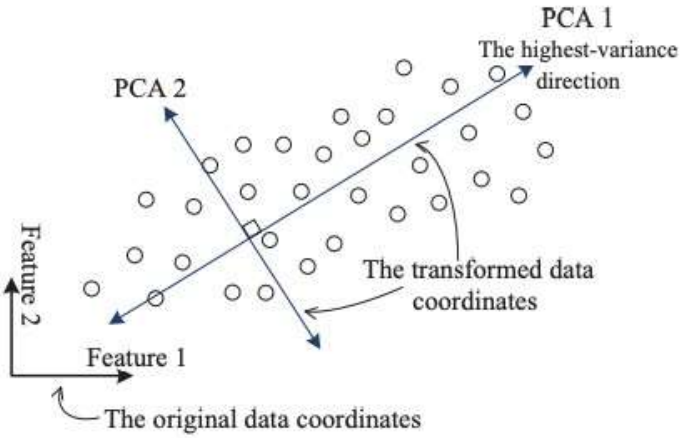

Fig. 3: A simple 2D visualization of the principal component analysis algorithm. It is important to note that the potential of the PCA algorithm is high mainly when dealing with high-dimensional data [62].

This approach is especially popular in WSN tasks like **node clustering** and **data aggregation** ([54]–[60]). These problems typically involve unstructured or unlabeled datasets, which makes unsupervised learning a natural and effective fit for such applications.

1. **K-means Clustering:** The **k-means algorithm** [61] is a popular **unsupervised learning** method used to group data into distinct clusters. It is commonly applied in sensor node clustering problems due to its **linear complexity** and **simple implementation**. The k-means process to solve the node clustering issue follows these steps:

- (a) Randomly select **k nodes** as initial centroids for the clusters.
- (b) Assign each node to the nearest centroid using a **distance function**.
- (c) Recalculate the centroids based on the current node assignments.
- (d) Repeat steps (b) and (c) until the clustering converges, i.e., when the sum of distances between nodes and centroids falls below a predefined threshold.

2. **Principal Component Analysis (PCA): PCA** is a **multivariate technique** used for **data compression** and **dimensionality reduction**, aiming to extract significant information from data and represent it in the form of new orthogonal variables called **principal components** [62]. As illustrated in Fig. 3, these principal components are arranged such that the first component captures the **highest variance** in the data, and subsequent components capture decreasing levels of variance. Thus, components with the least variance, which contain minimal information, can be discarded. For instance, PCA helps reduce the amount of data transmitted between sensor nodes by identifying a smaller set of uncorrelated linear combinations of the original readings. Additionally, PCA simplifies problem-solving by focusing on a few relevant factors in large-scale data problems (i.e., compressing large data into smaller, more manageable representations) [63]. A detailed explanation of PCA theory, including eigenvalues, eigenvectors, and covariance matrix analysis, is provided in [62].

C. Reinforcement Learning

Reinforcement learning allows an agent (e.g., a sensor node) to learn through interactions with its environment. The agent gradually learns to choose actions that maximize its long-term rewards based on its experiences. A widely recognized reinforcement learning method is
Q-learning [64]. As illustrated in Fig. 4, the agent updates its rewards after each action taken in a particular state. The expected total future reward, known as the **Q-value**, for performing a specific action ata_t in state sts_t, is calculated using Eq. (1).

$$Q(st+1, at+1) = Q(st, at) + \gamma\,(r(st, at) - Q(st, at))$$

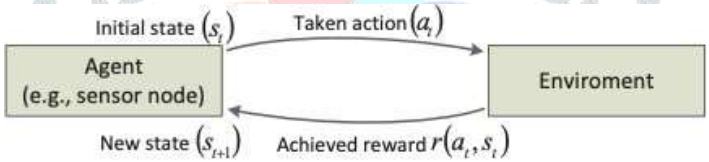

Fig. 4: A Visualization of the Q-learning method.

Here, r(st,at)r(s_t, a_t) represents the immediate reward obtained by performing action ata_t in state sts_t, and γ\gamma is the learning rate that controls the speed of learning, typically set between 0 and 1. This algorithm can be effectively implemented in a distributed system like WSNs, where each node aims to select actions that maximize its long-term rewards. Notably, Q-learning has been widely and efficiently applied to solve the WSN routing problem (e.g., [65], [66], [67], [68]).

3. FUNCTIONAL CHALLENGES

In the design of wireless sensor networks (WSNs), it is crucial to account for the power and memory limitations of sensor nodes, topology changes, communication link failures, and the need for decentralized management. Machine learning approaches have been effectively used to tackle various functional challenges in WSNs, such as energy-efficient routing, real-time query processing, event detection, localization, node clustering, and data aggregation.

A. Routing in WSNs

Designing an efficient routing protocol for WSNs requires addressing challenges like energy consumption, fault tolerance, scalability, and data coverage. Sensor nodes have limited processing power, memory, and bandwidth. Typically, the routing problem in WSNs is modeled as a graph G=(V,E)G = (V, E), where VV represents the nodes, and EE represents the bidirectional communication links between them. The routing problem can then be

described as finding the minimum cost path from a source node to all destination nodes using the available edges. This path forms a spanning tree T=(V,E)T = (V, E), with the source as the root node and the destination nodes as leaves. Solving this tree with optimal data aggregation is NP-hard, even when the complete network topology is known.

Machine learning enables a sensor network to learn from past experiences, make optimal routing decisions, and adapt to the dynamic environment. The advantages of using machine learning in routing include:

- The ability to learn optimal routing paths that save energy and extend the network's lifetime, even in dynamically changing environments.
- Simplifying complex routing problems by breaking them down into smaller sub-problems. In each sub-problem, nodes focus on their local neighbors, leading to low-cost, efficient, and real-time routing.
- Meeting quality of service (QoS) requirements in routing through relatively simple computational methods and classifiers.

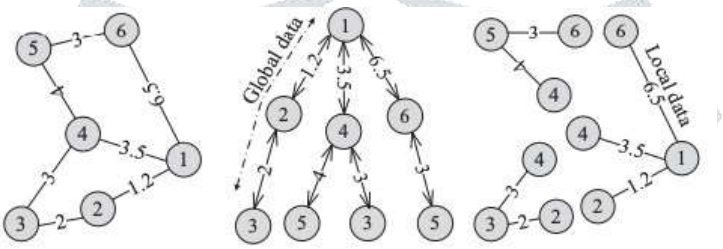

Fig. 5: An example of a sensor network routing problem using a graph along with each path routing cost, traditional spanning tree routing, and the generated sub-problems using machine learning that require only local communication to achieve optimal routing (i.e., require only single-hop neighborhood information exchange).

Figures 5a and 5b depict a basic sensor network routing problem using a graph and the traditional spanning tree routing algorithm, respectively. To determine the optimal routing paths, the network nodes must exchange routing information. In contrast, Figure 5c shows how machine learning simplifies the typical routing problem by focusing only on the information from neighboring nodes, which is used to predict the quality of the entire path. Each node independently carries out routing procedures to decide on channel assignments and optima transmission power. As discussed in this subsection, this approach provides near-optimal routing decisions with minimal computational complexity.

This subsection covers a variety of machine learning-based routing protocols developed for WSNs. Table I offers a summary and comparison of these protocols, with the "Scalability" column indicating the solutions' ability to route data in large-scale networks.

1. **Distributed Regression Framework**: In [69], Guestrin et al. introduced a general framework for sensor data modeling. This distributed framework relies on network nodes to fit a global function that matches their individual measurements. The nodes perform kernel linear regression using weighted components. Kernel functions map training samples into a feature space to simplify data manipulation (for an introduction to kernel methods, see [71], [72]). The framework takes advantage of the high correlation between readings from multiple sensors, minimizing communication overhead while detecting the structure of sensor data. These results represent a significant step towards developing a distributed learning framework for wireless networks using linear regression methods. The main benefits of this algorithm include good fitting results and minimal learning phase overhead. However, it is limited in its ability to

learn non-linear and complex functions.

TABLE I: Summary of wireless sensor network routing protocols that adopt machine learning paradigms.

ROUTING PROTOCOLS	TOPOLOGY	MACHINE LEARNING ALGORITHM(S)	OVERHEAD	SCALABILITY	DELAY	DISTRIBUTED / CENTRALIZED	QOS
Distributed regression [69]	Flat / multi-hop	kernel linear regression	Low	Limited	High	Distributed	No
SIR [70]	Flat / multi-hop	SOM	High	Limited	Moderate	Hybrid	Yes
Q-MAP multicast [65]	Flat /multi-hop	Q-learning	Low	Moderate	High	Distributed	No
RLGR [66]	Hierarchical / geographic routing	Q-learning	Low	Good	Low	Distributed	No
Q-Probabilistic [68]	Flat / geographic routing	Q-learning	Low	Limited	High	Distributed	Yes
FROMS [67]	Flat - multi-hop	Q-learning	High	Limited	Moderate	Distributed	No

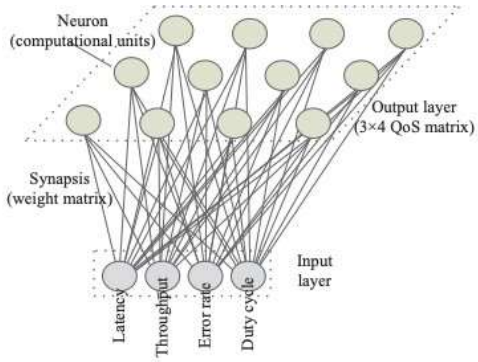

Fig. 6: The SOM construction of the SIR algorithm, where routing link is selected based on the multi-hop path QoS metrics (latency, throughput, error rate, and duty cycle) and the Dijkstra's algorithm [70].

2. **Data Routing using Self-Organizing Map (SOM)**: Barbancho et al. [70] introduced the "Sensor Intelligence Routing" (SIR) approach, which utilizes SOM unsupervised learning to identify optimal routing paths, as shown in Figure 6. SIR makes a slight modification to Dijkstra's algorithm to establish the network backbone and determine the shortest paths from the base station to each node in the network. During the route learning process, the second layer neurons compete to reserve the highest weights in the learning chain. As a result, the weights of the winning neuron and its neighboring neurons are updated to better align with the input patterns. This learning phase is computationally intensive due to the neural network generation task, so it should be carried out in a resourceful central station. However, the execution phase incurs no computational cost and can be performed directly on the network nodes. This hybrid approach, combining Dijkstra's algorithm with the SOM model, incorporates Quality of Service (QoS) requirements such as latency, throughput, packet error rate, and duty cycle when updating the neurons' weights. The main challenges in applying this algorithm include its complexity and the overhead of the learning phase, especially when the network topology or settings change.

3. **Routing Enhancement Using Reinforcement Learning (RL)**: In multicast routing, a node sends the same message to multiple receivers. Sun et al. [65] demonstrated how the Q-learning algorithm can enhance multicast routing in wireless ad hoc networks. The Q-MAP multicast routing algorithm is designed to ensure reliable resource allocation. In a mobile ad hoc network, nodes may have varying capabilities, and it is not feasible to maintain a global, up-to-date view of the entire network structure. The multicast routes are determined in two phases. The first phase, "Join Query Forward," discovers the optimal route and updates the Q-values (predictions of future rewards) in the Q-learning algorithm. The second phase, "Join Reply Backward," establishes the optimal path for multicast transmissions. Using Q-learning for multicast routing in mobile ad hoc networks helps reduce the overhead of route searching. However, energy efficiency is a critical requirement for WSNs, so Q-MAP needs to be adapted for WSNs, taking into account factors such as hierarchical and geographic routing.

The Federal Communications Commission (FCC) has allocated the frequency range from 3.1 to 10.6 GHz (7,500 MHz) for unlicensed ultra-wideband (UWB) communication [73]. UWB is a technique used for transmitting large amounts of data over short distances by utilizing a broad range of frequency bands with relatively low power. Dong et al. [66] adopted a similar approach to that in [65] to improve geographic routing in UWB-equipped sensor networks. The "Reinforcement Learning-based Geographic Routing" (RLGR) protocol incorporates sensor node energy and delay as key factors for defining the reward function in the learning process. This hierarchical geographic routing utilizes UWB technology to detect node locations, with only cluster heads equipped with UWB devices. Each node maintains a simple look-up table to store information about its neighbors (such as their locations and energy levels) during network learning. This information is exchanged between nodes via brief “hello” messages to determine the best routing actions. One major advantage of using reinforcement learning in routing is that it does not require knowledge of the global network structure to find a good routing solution.

In [68], Arroyo-Valles et al. introduced the "Q-Probabilistic Routing" (Q-PR), an improved geographic routing protocol for WSNs that learns from past routing decisions (e.g., selecting the path with the highest delivery rate over time). Unlike RLGR [66], Q-PR offers enhanced Quality of Service (QoS) support. It takes into account message importance, expected delivery rate, and power constraints to determine the best routes using reinforcement learning and a Bayesian decision model. This algorithm performs on-line operation by discovering the next hop during the routing process. The Bayesian method is employed to decide which neighbor nodes should receive the packets, considering factors such as data importance, node profiles, and energy usage for transmission and reception.

Förster and Murphy [67] also proposed an improvement to routing in WSNs using reinforcement learning. Their approach, called "Feedback Routing for Optimizing Multiple Sinks in WSN" (FROMS), allows for efficient routing from multiple sources to multiple sinks. The primary advantage of FORMS is its ability to optimize routing paths in such complex scenarios. The
Q-values are initialized based on the hop counts to each node in the network, which are gathered through short “hello messages” exchanged between nodes during the initial stages of deployment. FORMS builds on the basic mechanism of RLGR [66], assuming that all nodes can directly communicate with their neighbors.

B. Clustering and Data Aggregation

A key drawback of reinforcement learning-based routing algorithms is their limited ability to predict future knowledge (i.e., they cannot look ahead). As a result, these algorithms may not be suitable for highly dynamic environments, as they require considerable time to learn optimal routes.

In large-scale, energy-constrained sensor networks, directly transmitting all data to the sink is inefficient [74]. A more efficient approach is to transmit the data to a local aggregator, known as a cluster head, which collects and aggregates data from all sensors within its cluster before sending it to the sink. This method typically leads to energy savings. Several studies have explored the optimal selection of cluster heads (i.e., the cluster head election process), as seen in [75], [76], [77]. A comparison and classification of traditional clustering algorithms can be found in [78].

Figure 7 illustrates cluster-based data aggregation in WSNs, where data is transmitted from sources to a base station. In this setup, some faulty nodes may exist, and these nodes need to be removed from the network. Faulty nodes could provide inaccurate readings, which may negatively impact the network's overall performance. Machine learning (ML) techniques enhance node clustering and data aggregation in the following ways:

- ML can be used to compress data locally at the cluster heads by effectively identifying similarities and dissimilarities (such as from faulty nodes) in the readings of different sensors.

- ML algorithms help efficiently select the cluster head, as the proper choice of cluster head can significantly reduce energy consumption and improve the network's longevity.

Table II compares different solutions for data aggregation and node clustering. The "Balancing energy consumption" column indicates whether the protocol distributes computationally intensive tasks across all nodes while considering the remaining energy levels. The "Topology aware" column shows whether full knowledge of the network's topology is required.

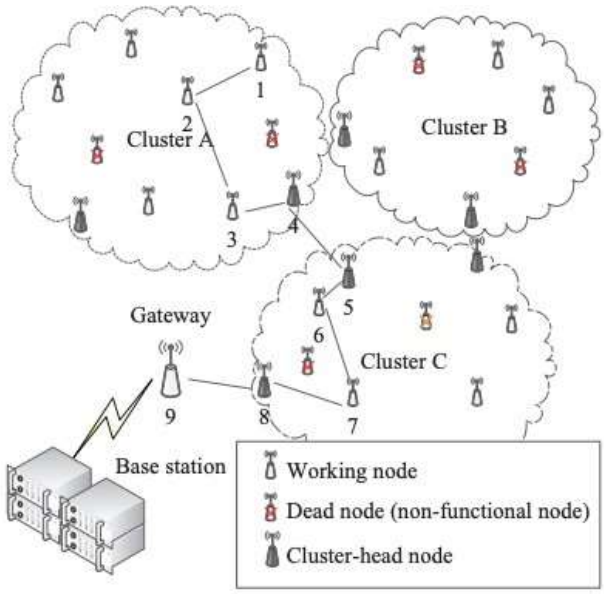

Fig. 7: Data aggregation example in a clustered architecture, where the nodes are marked as working, dead and cluster heads.

1. **Large-Scale Network Clustering Using Neural Networks**: Hongmei et al. [79] explored the creation of self-managed clusters through neural networks. This approach addresses the clustering challenge in large-scale networks with short transmission ranges, where centralized algorithms may not be efficient. However, when the transmission range is large, the performance of this algorithm approaches that of centralized algorithms in terms of efficiency and quality of service.

2. **Cluster Head Election Using Decision Trees**: Ahmed et al. [80] used a decision tree algorithm to tackle the cluster head election problem. This method processes input vectors through the decision tree, considering key factors like the distance to cluster centroids, battery levels, mobility, and vulnerability indicators. The results of their simulations show that this method improves cluster head selection performance compared to the “Low Energy Adaptive Clustering Hierarchy” (LEACH) [87] algorithm.

3. **Gaussian Process Models for Sensor Readings**: A Gaussian process (GP) is a combination of random variables characterized by mean and covariance functions. Ertin
[81] proposed a scheme for initializing probabilistic models of sensor readings using Gaussian process regression. Similarly, Kho et al. [82] extended Gaussian process regression to sample sensor data adaptively based on its significance. Focusing on energy consumption, [82] examined the trade-off between computational cost and solution optimality. In general, Gaussian process models are suited for problems with small training datasets (a few thousand samples or fewer) and for predicting smooth functions [53]. However, WSN designers must account for the high computational complexity of these methods when applied to large-scale networks.

TABLE II: Compassion of different machine learning-based data aggregation and node clustering mechanisms.

Mechanisms	Machine learning algorithm(s)	Complexity	Balancing energy consumption	Delay	Overhead	Topology aware
Large scale network clustering [79]	NNs	Moderate	Yes	High	Low	Yes
Cluster head election [80]	DT	Low	Yes	Low	Low	Yes
Gaussian process models for censored sensor readings [81]	GPR	Moderate	No	Moderate	Moderate	No
Adaptive sampling [82]		High	Yes	High	High	No
Clustering using SOM and sink distance [54]	SOM	Moderate	No	High	Moderate	Yes
Online data compression [83]	LVQ	High	No	High	High	Yes
Data acquisition using compressive sensing [55], [56]	PCA	High	Yes	High	High	Yes
Transmission reduction [57]		Moderate	No	High	High	Yes
Consensus-based distributed PCA [58]		Moderate	Yes	High	High	No
Lossy data compression [84]		Moderate	No	Moderate	High	Yes
Collaborative signal processing [60]	k-means	Low	Yes	Moderate	Moderate	No
Advanced surveillance systems [59]		Moderate	Yes	Low	Low	Yes
Role-free clustering [85]	Q-learning	Low	No	Low	Low	No
Decentralized learning for data latency [86]	RL	Moderate	Yes	Low	Low	No

4. **Data Aggregation Using Self-Organizing Map (SOM)**: The SOM algorithm is an unsupervised, competitive learning technique used to map high-dimensional data to lower dimensions. Lee et al. [54] introduced a new network architecture called "Cluster-based Self-Organizing Data Aggregation" (CODA). In this design, nodes can
classify aggregated data using the self-organizing algorithm. The neuron j∗j^*, which has a weight vector w(t)w(t) closest to the input vector x(t)x(t), is identified as the winning neuron.

$$j* = \arg\min j \, kxj(t) - wj(t)k, \; j = 1, ..., N$$

where N represents the number of neurons in the second layer. Further, the winning node and its neighbors are updated as follows:

$$wj(t+1) = wj(t) + h(t)(xj(t) - wj(t))$$

where w (t) and w (t + 1) represent the values of a neuron at time t and t+1, respectively. In addition, h (t) is the Gaussian neighborhood function given as:

$$h(t) = 1 \sqrt{2\pi\sigma} \exp - kj* - jk \, 2 \, 2\sigma \, 2(t) !$$

Using CODA for data aggregation will result in enhancing the quality of data, saving network energy, and reducing the network traffic.

Compression: While the methods mentioned above require full knowledge of the network topology, some algorithms do not have this limitation. For instance, Lin et al. [83] introduced "Adaptive Learning Vector Quantization" (ALVQ), a technique designed to retrieve compressed versions of sensor node readings accurately. By utilizing data correlation and historical patterns, ALVQ employs the LVQ learning algorithm to predict a code-book based on past training samples. This reduces the bandwidth needed for transmission and improves the accuracy of recovering the original readings from the compressed data.

A significant drawback of using LVQ for online data aggregation is that neurons far from the training samples (referred to as dead neurons) will never compete, which means they won't contribute to the process. Therefore, it's important to develop robust algorithms that can handle outliers. LVQ, however, is effective for representing large datasets with a few vectors [43].

6) Data Aggregation Using Principal Component Analysis (PCA): Two key algorithms used alongside PCA to improve data aggregation in WSNs are as follows:

- **Compressive Sensing (CS)**: CS replaces the traditional "sample then compress" method with "sample while compressing". It takes advantage of signal sparsity to recover the original signal from a smaller number of random measurements. An introductory guide to CS can be found in [88].
- **Expectation-Maximization (EM)**: This iterative algorithm consists of two steps: an expectation (E) step and a maximization (M) step. During the E-step, EM formulates a cost function based on the current expectations of system parameters. In the M-step, it recomputes parameters to minimize the estimation error of the cost function.

Masiero et al. [55], [56] developed a method for estimating distributed observations using only a few samples from a WSN. This approach utilizes PCA to create orthogonal components, which are then used by compressive sensing to reconstruct the original readings. This method is independent of the routing protocol, as it can estimate spatial and temporal data correlations.
Similarly, Rooshenas et al. [57] applied PCA to optimize the direct transmission of readings to the base station. PCA helps significantly reduce traffic by combining data from multiple nodes into fewer packets, and the process is carried out in intermediate nodes before forwarding the combined packets to the destination.

Equally significant, Macua et al. [58] introduced distributed consensus-based approaches for data compression utilizing PCA and maximum likelihood estimation of observed data. These methods include "Consensus-based Distributed PCA" (CB-DPCA), which explores the eigenvectors of local covariance matrices, and "Consensus-based EM Distributed PCA" (CB-EM-DPCA), which employs a distributed EM algorithm. Both methods use the consensus algorithm [90] to predict the data's probability distribution and calculate the global dominant eigenvectors via local communication (i.e., single-hop communications). CB-DPCA and
CB-EM-DPCA can be adjusted to balance approximation quality and communication costs by modifying the consensus round parameter. For instance, increasing the number of consensus rounds improves the algorithm's accuracy but also raises its computational demands.

Recently, Fenxiong et al. [84] addressed the data compression challenge using PCA by reducing data from high-dimensional to lower-dimensional space. Data is collected over time and transmitted from each node to its corresponding cluster head. At the cluster head, the data matrix is compressed to eliminate redundancy. This compression is achieved by excluding the principal components with the least variation.

The primary challenge of PCA-based data aggregation techniques is their high computational requirements. While these methods help increase throughput, they effectively manage the high dimensionality of collected data by retaining only the most relevant information (data dimensionality reduction).

7) Collaborative Data Processing Through K-means Algorithm: Li et al. [60] explored distributed detection and tracking of a single target in sensor networks. Their "Collaborative Signal Processing" (CSP) framework gathers information from the monitored environment and can track multiple targets using classification methods like Support Vector Machines (SVM) and k-nearest neighbors.

Traditional surveillance systems often face the challenge of managing large volumes of data collected from surveillance cameras. The need for complex computation and analysis adds to this burden, prompting the exploration of more practical solutions. In this context, Tseng et al.
[59] introduced the “Integrated Mobile Surveillance and Wireless Sensor System” (iMouse), which leverages powerful mobile sensors to enhance conventional surveillance systems. iMouse divides the monitored area into clusters using the k-means unsupervised learning algorithm, where each cluster is monitored by a single mobile sensor. While the use of k-means for data processing is appealing due to its simplicity and low complexity, it remains sensitive to outliers and the initial selection of seeds.

Förster and Murphy [85] proposed a new approach for wireless sensor network (WSN) clustering called "Role-Free Clustering with Q-Learning for Wireless Sensor Networks" (CLIQUE). Instead of relying on an election process, CLIQUE allows each node to evaluate its capability to act as a cluster-head node, using the Q-learning algorithm along with dynamic network parameters like energy levels.

Mihaylov et al. [86] addressed the issue of high data latency in sensor networks with random topologies by using reinforcement learning. In their approach, each node independently runs the learning algorithm to optimize data aggregation, eliminating the need for a central control station. This decentralized method reduces the transmission overhead and helps conserve node energy, thereby extending the network's lifespan.

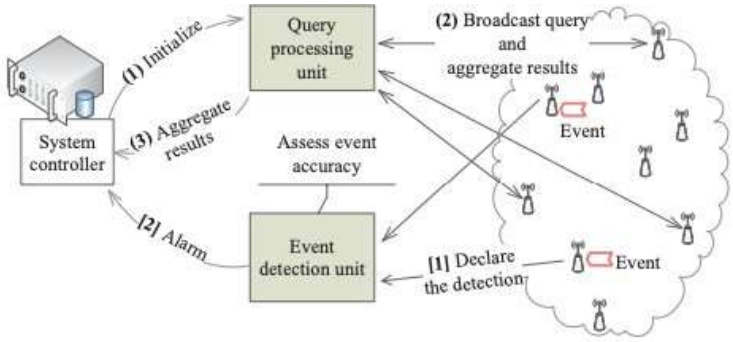

Fig. 8: Event detection and query processing enhancement using machine learning methods by assessing event validity and delimiting queried areas. System controller initiates query that is spread by the query processing unit to intended nodes. In contrast, events are detected by nodes to monitor specific signs within the monitored area.

C. Event Detection and Query Processing

Event detection and query processing are key functional requirements for large-scale sensor networks, necessitating reliable event scheduling and detection with minimal human intervention. In Wireless Sensor Networks (WSNs), monitoring can be categorized as
event-driven, continuous, or query-driven [6]. Figure 8 outlines the operations involved in event detection and query processing within WSNs. Machine learning offers effective solutions to restrict the query areas and evaluate event validity, improving event detection and query processing efficiency. The integration of machine learning provides several advantages:

- Learning algorithms help create efficient event detection systems that require minimal storage and computational resources. Additionally, they can assess the accuracy of events using simple classifiers.
- Machine learning also supports the development of optimized query processing methods for WSNs, allowing the identification of relevant search regions when a query is made, without overwhelming the entire network.

The design of advanced event detection and query processing techniques has gained considerable attention in the WSN research community. Traditional methods often rely on setting a fixed threshold value for the sensed phenomenon and notifying the system manager if violations occur. However, many modern WSN applications demand more complex event and query processing approaches beyond simple threshold definitions. One promising technique is the use of machine learning to develop sophisticated event detection and query processing solutions. Table III compares various machine learning-based approaches for event detection and query processing in WSNs.

TABLE III: Comparison of functional aspects of different machine learning-based event detection and query processing solutions for WSNs.

Approaches	Machine learning algorithm(s)	Data Delivery Models	Complexity	Characteristics
Event region detection [91]	Bayesian	Event-driven	Low	Fault-tolerant event region detection
Activity recognition [92]			Moderate	Real-time activity recognition
In-network query processing [24]	k-NN	Query-driven	Low	In-network query processing
Query processing in 3D space [25]			Moderate	Enhance 3D space query processing
Forest fire detection [26]	NNs	Event-driven	Moderate	Real-time and lightweight forest fire detection
Distributed event detection [27]	DT	Event-driven	Low	Disaster prevention system
Query optimization [93]	PCA	Query-driven	High	Query optimization and dimensionality reduction

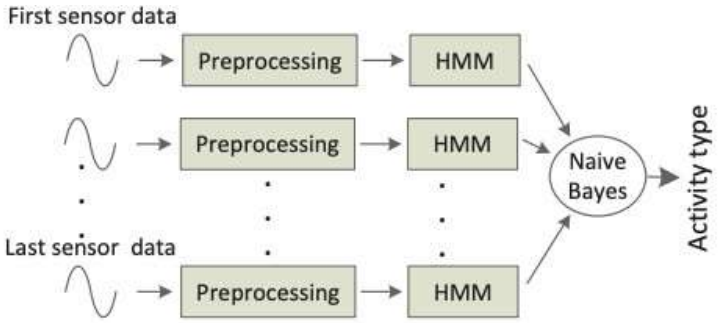

Fig. 9: Human activity recognition using the hidden Markov model and the naive Bayes classifier [92].

1. **Event recognition using Bayesian algorithms:** Krishnamachari and Iyengar [91] explored the use of Wireless Sensor Networks (WSNs) for detecting environmental phenomena in a distributed manner. In this approach, readings that exceed a specific threshold are considered faulty. The study employs decentralized Bayesian learning to detect up to 95% of faults, helping identify the event region. It is important to note that Chen et al. [94] made corrections to several errors in the distributed Bayesian algorithms proposed by Krishnamachari and Iyengar [91], which led to improved error and performance calculations. Additionally, Zappi et al. [92] introduced a real-time activity recognition approach using WSNs to accurately detect body gestures and motion. The system involves nodes placed across the body, which use accelerometer sensors to measure motion along three axes (positive, negative, and null). These measurements are then processed by a hidden Markov model (HMM) to predict the activity at each sensor. Sensor selection is based on their potential contribution to classifier accuracy, ensuring that the most informative sensors are chosen for gesture detection. The predictions from individual sensors are combined using a naive Bayes classifier to maximize the posterior probability, leading to the final gesture decision. The system's architecture is depicted in Fig. 9.
2. **Forest fire detection using neural networks:** WSNs have been widely employed in fire detection and rescue systems (see [95] and related references for system requirements and challenges). Additionally, WSN-based forest fire detection systems can outperform satellite-based solutions while being more cost-effective. Yu et al. [26] presented a
real-time forest fire detection scheme that utilizes a neural network. In this scheme, data processing is distributed to cluster heads, and only crucial information is aggregated for decision-making. While this approach is innovative and environmentally beneficial, it can be difficult for decision-makers to interpret the classification tasks and core functionality of such systems.

3. **Query processing using k-nearest neighbors:** The k-nearest neighbor (k-NN) algorithm is an effective method for query processing in Wireless Sensor Networks (WSNs). For instance, Winter et al. [24] proposed an in-network query processing solution called the "K-NN Boundary Tree" (KBT) algorithm. In this approach, nodes that are aware of their location determine their k-NN search region when a query is received from the application manager. Jayaraman et al. [25] further expanded this concept with the "3D-KNN" scheme, which applies the k-NN algorithm to restrict the query region to include at least the k-nearest nodes within a 3D space. Additionally, they use signal-to-noise ratio (SNR) and distance measurements to refine the k-NN search. The main challenges of k-NN-based query processing are the large memory requirements to store collected data and the high processing delay in large-scale sensor networks.

4. **Distributed event detection for disaster management using decision trees:** Bahrepour et al. [27] developed a decision tree-based approach for event detection and recognition in sensor networks aimed at disaster prevention. This decentralized mechanism is particularly useful for fire detection in residential areas. The final event detection decision is made through a simple voting process, where the highest-reputation nodes cast the deciding vote.

5. **Query optimization using principal component analysis (PCA):** Malik et al. [93] improved traditional query processing in WSNs by utilizing data attributes and PCA to reduce processing overhead. PCA helps dynamically identify key attributes (dominant principal components) within a correlated data set. Figure 10 illustrates the four-step workflow of the proposed algorithm. In Step 1, a structured query language (SQL) request containing human-readable attributes is sent to the database management and optimization system. In Step 2, the system optimizes the query by extracting high-variance components from historical data using PCA. In Steps 3 and 4, the optimized query is sent to the WSN to collect sensory data. The original human-readable attributes can then be retrieved by reversing the PCA process. This approach results in a 25% improvement in energy savings for network nodes and achieves 93% accuracy.
However, the trade-off is a reduction in data accuracy, as some components are discarded. As such, this method may not be suitable for applications that require high precision and accuracy.

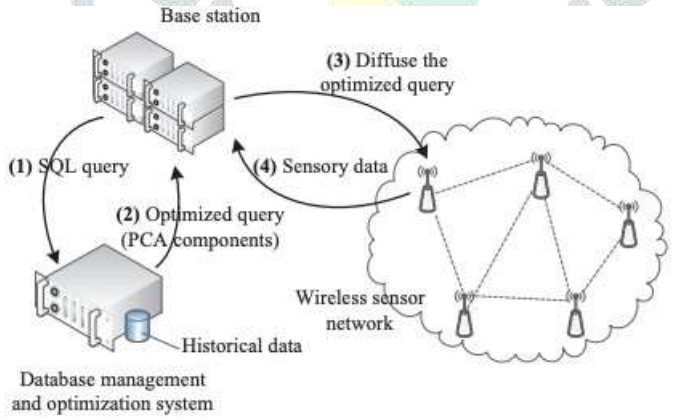

Fig. 10: Workflow of the query optimization and reduction system using PCA proposed in [93]

D. Localization and Objects Targeting

Localization refers to the process of identifying the geographical coordinates of nodes and components within a network. Knowing the position of sensor nodes is a crucial aspect of Wireless Sensor Networks (WSNs), as many of their operations depend on location information [96]. However, in large-scale systems, equipping every node with GPS hardware is often cost-prohibitive. Additionally, GPS signals may not be accessible in certain environments, such as indoors. In

many cases, relative positioning is adequate. Moreover, by knowing the exact positions of a few nodes (called anchor nodes), it is possible to convert the relative positions of the rest into absolute coordinates [97].

To improve the accuracy of proximity-based localization, additional data such as distance, angle, or a combination of both can be utilized. Distance measurements can be obtained using various methods, including Received Signal Strength Indicator (RSSI), Time of Arrival (TOA), and Time Difference of Arrival (TDOA). Additionally, signal angles can be estimated using compasses or specialized smart antennas [98]. A comprehensive overview of different
range-based localization techniques is available in [42].

Sensor nodes may shift from their original locations after deployment due to movement or external factors. Machine learning algorithms offer several advantages in the localization process, including:

- Estimating absolute positions from relative node locations using a limited number of anchor points, thereby eliminating the need for additional hardware for range measurements.
- In applications such as surveillance and target tracking, machine learning can help divide the monitored area into clusters, each representing a distinct location reference.

The discussion begins with defining common terms used in the WSN localization domain, as shown in Figure 11.

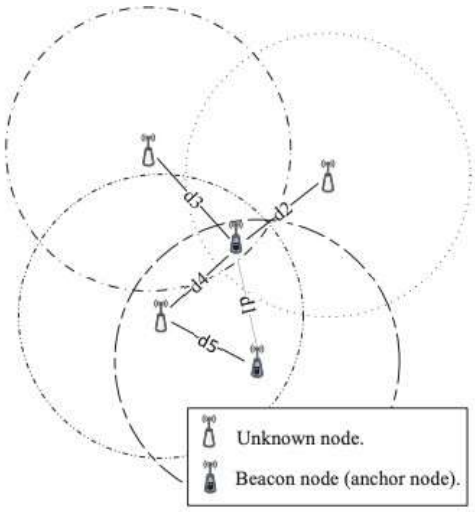

Fig. 11: Localization using few beacon nodes by utilizing machine learning algorithms and other signal strength indicators (reformulated from [96]).

- An **unknown node** refers to a node that is unable to determine its current geographical position.
- A **beacon node** (also known as an anchor node) is a node that knows its exact location, either through positioning hardware or by manual placement. These nodes typically serve as reference points to help estimate the positions of unknown nodes.
- **Received Signal Strength Indication (RSSI)** is a measure of the strength of a received signal, commonly used to assess transmission performance or approximate distance.

The following section explores key localization techniques in Wireless Sensor Networks (WSNs) that incorporate machine learning, with a summary provided in Table IV.

1. **Bayesian node localization:** Morelande et al. [21] proposed a localization method using Bayesian algorithms that relies on only a small number of anchor nodes. Their work focuses on improving a technique known as *progressive correction* [109], which involves sampling from likelihood distributions

to better approximate the posterior distribution. This Bayesian-based approach is particularly effective for large-scale networks comprising thousands of nodes. The strength of the Bayesian method lies in its ability to work with incomplete datasets by leveraging prior knowledge and probabilistic reasoning.

2. **Reliable location-aware activity recognition:** Lu and Fu [22] tackled the challenge of detecting both the location of sensors and the activities they monitor within smart home environments. These activities include actions such as answering a phone call, listening

TABLE IV: Summary of localization algorithms in WSNs that adopt machine learning concepts and their prime advantages. The column "Applications" specifies the targeted application(s) of the proposed solution (either general-purpose or a specific application).

Studies description	Machine learning algorithm(s)	Complexity	Distributed / centralized	Beacon / Beacon-less	Applications
Bayesian node localization [21]	Bayesian	Moderate	Centralized	Beacon	General-purpose
Location-aware activity recognition [22]		Moderate	Centralized	Beacon	Smart homes
Localization based on NNs [23]	NNs	High	Centralized	Beacon	General-purpose
Soft localization [99]		Moderate	Distributed	Beacon	General-purpose
Localization based on NNs [100]		High	Distributed	Beacon	General-purpose
Area localization [51]	SVM	Moderate	Distributed	Beacon	General-purpose
Localization using SVM [50]		Moderate	Distributed	Beacon	General-purpose
Localization using SVR [49]	SVR	Moderate	Distributed	Beacon	General-purpose
Target classification and information fusion [101]	DT	Low	Distributed	Beacon	Acoustic WSNs
Underwater surveillance system [102]		Moderate	Centralized	Beacon	Submarine surveillance
Sensor placements [103]	GP	Low	Distributed	Beacon	Node deployment
Spatial Gaussian process regression [104]	GPR	Moderate	Distributed	Beacon	Collective node motion
Localization in 2D space [105]	SOM	Moderate	Distributed	Beacon	Large scale WSNs
Localization using SOM [106]		Low	Centralized	Beacon-less	General-purpose
Distributed localization [107]		Moderate	Distributed	Beacon-less	General-purpose
Path determination [108]	RL	Low	Distributed	Beacon	Mobile nodes

to music, opening a refrigerator, or studying. In these settings, developers must consider both human behavior and environmental constraints to ensure seamless operation. The proposed system, called *Ambient Intelligence Compliant Object* (AICO), enhances interaction with household devices by enabling smarter functions like automated power management. AICO uses multiple naive Bayes classifiers to identify the resident's current position and detect faulty sensors, ensuring the system's reliability. Although the localization approach is strong, it is tailored to specific applications and requires the predefined selection of relevant activities. This limitation arises because the system's learning features are manually chosen based on the target domain. To address this issue in centralized architectures, the authors suggest exploring unsupervised learning techniques such as deep learning or non-negative matrix factorization [110] to enable automatic feature extraction.

3. **Neural network-based localization:** Shareef et al. [23] compared three types of neural networks for WSN localization—multi-layer perceptron (MLP), radial basis function (RBF), and recurrent neural networks (RNN). Among these, RBF networks produced the lowest localization error but required more computational resources. On the other hand, MLPs were the most resource-efficient in terms of processing and memory usage. Similarly, Yun et al. [99] proposed two sensor localization approaches using RSSI data from anchor nodes. The first approach combines fuzzy logic with a genetic algorithm, while the second utilizes a neural network to predict sensor locations based on RSSI inputs. In a comparable effort, Chagas et al. [100] also used RSSI-based neural networks for WSN localization.The key benefit of these neural network-based approaches is their ability to output continuous-valued coordinates, such as positions in 3D space. However, unlike probabilistic models like Bayesian methods, neural networks do not provide a measure of uncertainty in their predictions, making it harder for designers to assess the confidence level of estimated node positions or manage potential localization errors.

4. **Localization using support vector machines (SVM):** SVMs have also gained popularity in WSN localization, particularly where it's impractical to equip every node with GPS. Yang et al. [51] introduced a mobile node localization system that uses SVMs in conjunction with connectivity information. The system begins by detecting node movement via radio frequency variations, such as changes in RSSI values. Once movement is identified, SVMs are used to estimate the new location. Likewise, Tran and Nguyen [50] proposed the LSVM (Localization Based on Support Vector Machines) method, which utilizes training data along with metrics like node connectivity and signal indicators to localize nodes efficiently. While LSVM enables fast and decentralized localization, it remains vulnerable to inaccuracies when the training data contains outliers.
5. **Localization using Support Vector Regression (SVR):**
   Due to resource constraints and the complexity of high-dimensional data, applying SVR in wireless sensor networks (WSNs) can be challenging. To address this, Kim et al. [49] proposed a lightweight SVR approach that splits the overall regression task into smaller, manageable sub-tasks. The network is divided into multiple sub-networks, with each handling a limited data set. These smaller sets are then processed by individual SVR sub-models. Finally, the results of these sub-models are integrated using a tailored ensemble technique. This method is not only computationally efficient but also resilient to noisy inputs, allowing it to converge to accurate predictions with minimal resource usage.
6. **Decision Tree-Based Localization:**
   Merhi et al. [101] introduced an acoustic target localization system for WSNs using decision tree learning. Their method relies on time difference of arrival (TDOA) metrics, organized in a spatially aware decision tree, to accurately determine the location of detected targets. They also developed the *Event-Based MAC* (EB-MAC) protocol, which enables real-time, event-driven localization in acoustic sensor networks. This system was implemented using the MicaZ board, which supports ZigBee 802.15.4 for personal area network communication.
   In underwater environments where GPS signals cannot effectively propagate, alternative localization strategies are required. Erdal et al. [102] addressed this by creating a submarine detection system for underwater surveillance. Here, sensor nodes—each attached to surface buoys via cables—are randomly deployed. These buoys collect sensor data and send it to a central unit, where a decision tree classifier is used to detect the presence of submarines based on the known locations of beacon nodes.
7. **Sensor Placement Using Gaussian Processes:**
   Krause et al. [103] developed an optimized strategy for sensor deployment in scenarios with spatially dependent data, such as temperature monitoring. Their approach incorporates a *lazy learning* method built upon Gaussian process models. Lazy learning defers heavy processing until a prediction is requested, storing training data for
   on-demand use. This technique ensures better handling of node failures and uncertainty in the model, improving the reliability and accuracy of sensor placement decisions.
8. **Distributed Node Motion Using Gaussian Process Regression:**
   A distributed protocol was designed to enable coordinated movement of sensor nodes. This method utilizes **Distributed Gaussian Process Regression (DGPR)** to estimate the most suitable positions for mobile nodes. Traditional Gaussian Process Regression (GPR) methods have a computational complexity of $O(N^3)$, where N denotes the number of data samples. To address this issue, a **sparse GPR** technique is used to reduce the computational burden. Each node independently runs the regression model using only the spatiotemporal data from its nearby nodes.
9. **Localization Using Self-Organizing Maps (SOM):**
   Paladina et al. [105] proposed a SOM-based localization solution tailored for WSNs with thousands of nodes. Each node implements a basic SOM model comprising a 3x3 input layer linked to two output neurons. The input consists of spatial data from eight surrounding anchor nodes, and once training is complete, the output neurons represent the node's 2D coordinates. A key limitation of this approach is the assumption that nodes are uniformly and evenly distributed throughout the area.

Giorgetti et al. [106] developed a localization method that relies solely on connectivity data and SOM, eliminating the need for absolute positioning or GPS. This approach is particularly resource-efficient, but it is centralized—requiring each node to forward its neighbor information to a central unit for building an adjacency matrix and calculating positions.

In contrast, Hu and Lee [107] introduced a SOM-based method that operates without anchor nodes and distributes the computational load across all nodes in the network. This decentralized model reduces data transmission demands and removes dependence on a central processing unit.

10. **Path Planning Using Reinforcement Learning:**
    Li et al. [108] proposed a reinforcement learning approach named **Dynamic Path determination of Mobile Beacons (DPMB)** for efficient, real-time positioning of sensor nodes in WSNs. A mobile beacon (MB), aware of its own coordinates during movement, is used to help determine the locations of nearby nodes. The method applies
    Q-learning, where the beacon's various locations represent different states. The objective is for every sensor in the network to eventually receive a location update from the MB. All processing is done within the beacon itself, conserving resources for the other nodes. However, being a centralized solution, the system's functionality depends entirely on the MB; if it fails, the entire process is compromised.

E. Medium Access Control (MAC)

In Wireless Sensor Networks (WSNs), multiple sensors work together to facilitate efficient data transmission. This creates unique challenges for designing Medium Access Control (MAC) protocols compared to traditional wireless networks, particularly in terms of **energy efficiency** and **latency** [112]. One critical factor is controlling the **duty cycle**—the proportion of time a sensor node remains active—to minimize power consumption. Therefore, MAC protocols must be tailored to optimize both data transmission and reception in sensor nodes. A detailed overview of MAC protocols tailored for WSNs can be found in [113].

In recent developments, **machine learning (ML)** techniques have been incorporated to enhance MAC protocol performance in WSNs. These advancements include:

- **Adaptive Duty Cycle Management:** ML can be employed to dynamically adjust a node's active time based on historical network traffic patterns. Nodes capable of predicting when others will finish their transmissions can switch to a low-power sleep mode during idle times and activate only when the channel is likely to be free. In WSNs, optimizing for energy savings and low latency often takes precedence over ensuring transmission fairness.

- **Security Enhancements:** ML methods can also be integrated with MAC protocols to improve **data security**, enabling the system to learn and detect irregular attack patterns over time. These ML-enhanced security features operate independently of the specific application layer.

Table V provides a comparative overview of the MAC protocols discussed. The "Synchronization" column denotes whether the protocol depends on external time synchronization, while "Adaptivity to changes" refers to how well the protocol manages network topology changes, such as node failures.

1) Bayesian Model-Based MAC Protocol:

Kim and Park [28] introduced a contention-based MAC protocol designed to manage when nodes are active or asleep in a more energy-efficient manner. Rather than continuously monitoring the communication channel, this method applies a **Bayesian statistical model** to predict when the channel is likely to be available, thus conserving energy. This protocol is primarily intended for **Carrier Sense Multiple Access (CSMA)**-based approaches, such

as **Sensor MAC (S-MAC)** [116] and **Timeout MAC (T-MAC)** [117].

2) Neural Network-Based MAC Protocols:

Time Division Multiple Access (TDMA) protocols organize network access by assigning periodic time slots to individual nodes. These protocols typically require a centralized unit to broadcast updated transmission schedules whenever the network topology changes. To address this, **Shen and Wang [29]** proposed a method that utilizes a **Fuzzy Hopfield Neural Network (FHNN)** to handle schedule broadcasting in TDMA-based systems. Their approach assigns time slots across the network nodes while optimizing the overall cycle length, avoiding transmission conflicts, and minimizing processing delays.

Similarly, **Kulkarni and Venayagamoorthy [30]** introduced a **CSMA-based MAC protocol** designed to protect WSNs from **Denial-of-Service (DoS)** attacks. These attacks flood the network with unnecessary traffic, exploiting WSNs' limited bandwidth and buffer capacities, which can hinder legitimate data delivery. Their proposed solution uses a neural network to analyze key network characteristics—like packet request rates and average packet wait times—to detect potential flooding. If the neural network detects abnormal activity above a predefined threshold, it disables MAC layer access **only at the affected nodes**, as the system is decentralized and localized in nature.

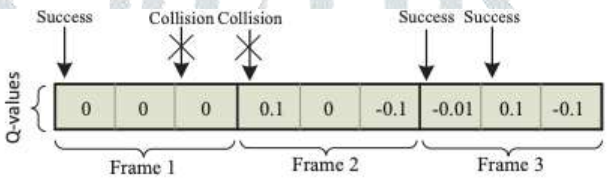

Fig. 12: An example of the Q-values of a node over three frames in a WSN that employs ALOHA-QIR to manage medium access [114].

3) Reinforcement Learning for Duty Cycle Optimization:

Liu and Elhanany [114] introduced **RL-MAC**, a MAC protocol for WSNs based on **Reinforcement Learning (RL)**. The protocol dynamically optimizes the **duty cycle**—the active time of a node—to reduce energy consumption and boost throughput. RL-MAC, similar to protocols like S-MAC [116] and T-MAC [117], operates on a synchronized, frame-based schedule. It intelligently adjusts parameters like slot duration, duty cycle, and active transmission periods in response to changing traffic loads and bandwidth conditions.

Likewise, **Chu et al. [112]** combined the principles of **slotted ALOHA** and **Q-learning** to develop a new protocol called **ALOHA-QIR (ALOHA and Q-Learning based MAC with Informed Receiving)**. This hybrid approach provides the benefits of simplicity, low overhead, and reduced collision probability. Nodes announce their future transmission slots during each frame, enabling neighboring nodes to enter sleep mode during unused periods. A **Q-value map** is maintained by each node to indicate its priority for slot usage. Nodes with higher Q-values are given transmission priority. These Q-values are updated over time using standard Q-learning techniques—starting at zero and adjusting based on transmission outcomes (rewarded with +1 for success, -1 for failure).

While reinforcement learning-based methods offer appealing advantages—such as distributed control and low memory and processing demands—they can initially suffer from **high collision rates** during the early learning or exploration stages.

4) Adaptive MAC Layer:

In many modern applications, such as healthcare and assisted living systems, Wireless Sensor Networks (WSNs) are used to directly transmit collected data to users' mobile devices. This creates new design challenges related to the changing communication patterns and service demands over time. Sha et al. [115] addressed this issue at the MAC layer by proposing the "Self-Adapting MAC Layer" (SAML) design. SAML consists of two primary

components: the "Reconfigurable MAC Architecture" (RMA), which allows switching between different MAC protocols, and the MAC engine, which learns the most suitable MAC protocol based on current network conditions. The learning process utilizes a decision tree classifier, as shown in Fig. 13. The decision tree considers several features, including inter-packet interval (IPI), received signal strength indication (RSSI) statistics (mean and variance), application QoS (Quality of Service) requirements (such as reliability, energy consumption, and latency), packet delivery rate (PDR), and traffic patterns. Supported MAC protocols include Pure TDMA [118], Adaptive TDMA [119], Box-MAC [120], RI-MAC [121], and ZigBee [122]. While the SAML approach offers an adaptive MAC solution for dynamic environments, it also introduces additional complexity and cost into the system design.

References

[1] T. O. Ayodele, "Introduction to machine learning," in New Advances in Machine Learning. InTech, 2010.

[2] A. H. Duffy, "The "what" and "how" of learning in design," IEEE Expert, vol. 12, no. 3, pp. 71–76, 1997.

[3] P. Langley and H. A. Simon, "Applications of machine learning and rule induction," Communications of the ACM, vol. 38, no. 11, pp. 54– 64, 1995.

[4] L. Paradis and Q. Han, "A survey of fault management in wireless sensor networks," Journal of Network and Systems Management, vol. 15, no. 2, pp. 171–190, 2007.

[5] B. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in 22nd International Conference on Distributed Computing Systems Workshops, 2002, pp. 575– 578.

[6] J. Al-Karaki and A. Kamal, "Routing techniques in wireless sensor networks: A survey," IEEE Wireless Communications, vol. 11, no. 6, pp. 6–28, 2004.

[7] K. Romer and F. Mattern, "The design space of wireless sensor networks," IEEE Wireless Communications, vol. 11, no. 6, pp. 54–61, 2004.

[8] J. Wan, M. Chen, F. Xia, L. Di, and K. Zhou, "From machine-tomachine communications towards cyber-physical systems," Computer Science and Information Systems, vol. 10, pp. 1105–1128, 2013.

[9] Y. Bengio, "Learning deep architectures for AI," Foundations and Trends in Machine Learning, vol. 2, no. 1, pp. 1–127, 2009.

[10] A. G. Hoffmann, "General limitations on machine learning," pp. 345– 347, 1990.

[11] M. Di and E. M. Joo, "A survey of machine learning in wireless sensor netoworks from networking and application perspectives," in 6th International Conference on Information, Communications Signal Processing, 2007, pp. 1–5.

[12] A. Forster, "Machine learning techniques applied to wireless ad-hoc networks: Guide and survey," in 3rd International Conference on Intelligent Sensors, Sensor Networks and Information. IEEE, 2007, pp. 365–370.

[13] A. Förster and M. Amy L, Machine learning across the WSN layers. InTech, 2011.

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

[15] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," Artificial Intelligence Review, vol. 22, no. 2, pp. 85–126, 2004.

[16] R. Kulkarni, A. Förster, and G. Venayagamoorthy, "Computational intelligence in wireless sensor networks: A

survey," IEEE Communications Surveys & Tutorials, vol. 13, no. 1, pp. 68–96, 2011.

[17] S. Das, A. Abraham, and B. K. Panigrahi, Computational intelligence: Foundations, perspectives, and recent trends. John Wiley & Sons, Inc., 2010, pp. 1–37.

[18] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, Learning from data. AMLBook, 2012.

[19] O. Chapelle, B. Schlkopf, and A. Zien, Semi-supervised learning. MIT press Cambridge, 2006, vol. 2.

[20] S. Kulkarni, G. Lugosi, and S. Venkatesh, "Learning pattern classification-a survey," IEEE Transactions on Information Theory, vol. 44, no. 6, pp. 2178–2206, 1998.

[21] M. Morelande, B. Moran, and M. Brazil, "Bayesian node localisation in wireless sensor networks," in IEEE International Conference on Acoustics, Speech and Signal Processing, 2008, pp. 2545–2548.

[22] C.-H. Lu and L.-C. Fu, "Robust location-aware activity recognition using wireless sensor network in an attentive home," IEEE Transactions on Automation Science and Engineering, vol. 6, no. 4, pp. 598–609, 2009.

[23] A. Shareef, Y. Zhu, and M. Musavi, "Localization using neural networks in wireless sensor networks," in Proceedings of the 1st International Conference on Mobile Wireless Middleware, Operating Systems, and Applications, 2008, pp. 1–7.

[24] J. Winter, Y. Xu, and W.-C. Lee, "Energy efficient processing of k nearest neighbor queries in location-aware sensor networks," in 2nd International Conference on Mobile and Ubiquitous Systems: Networking and Services. IEEE, 2005, pp. 281–292.

[25] P. P. Jayaraman, A. Zaslavsky, and J. Delsing, "Intelligent processing of k-nearest neighbors queries using mobile data collectors in a location aware 3D wireless sensor network," in Trends in Applied Intelligent Systems. Springer, 2010, pp. 260–270.

[26] L. Yu, N. Wang, and X. Meng, "Real-time forest fire detection with wireless sensor networks," in International Conference on Wireless Communications, Networking and Mobile Computing, vol. 2, 2005, pp. 1214–1217.

[27] M. Bahrepour, N. Meratnia, M. Poel, Z. Taghikhaki, and P. J. Havinga, "Distributed event detection in wireless sensor networks for disaster management," in 2nd International Conference on Intelligent Networking and Collaborative Systems. IEEE, 2010, pp. 507–512.

[28] M. Kim and M.-G. Park, "Bayesian statistical modeling of system energy saving effectiveness for MAC protocols of wireless sensor networks," in Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, ser. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2009, vol. 209, pp. 233– 245.

[29] Y.-J. Shen and M.-S. Wang, "Broadcast scheduling in wireless sensor networks using fuzzy hopfield neural network," Expert Systems with Applications, vol. 34, no. 2, pp. 900 – 907, 2008.

[30] R. V. Kulkarni and G. K. Venayagamoorthy, "Neural network based secure media access control protocol for wireless sensor networks," in Proceedings of the 2009 International Joint Conference on Neural Networks, ser. IJCNN'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 3437–3444.

[31] D. Janakiram, V. Adi Mallikarjuna Reddy, and A. Phani Kumar, "Outlier detection in wireless sensor networks using Bayesian belief networks," in 1st International Conference on Communication System Software and Middleware. IEEE, 2006, pp. 1–6.

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

[33] S. Kaplantzis, A. Shilton, N. Mani, and Y. Sekercioglu, "Detecting selective forwarding attacks in wireless sensor networks using support vector machines," in 3rd International Conference on Intelligent Sensors, Sensor Networks and Information. IEEE, 2007, pp. 335–340.

[34] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. Bezdek, "Quarter sphere based distributed anomaly detection in wireless sensor networks," in International Conference on Communications, 2007, pp. 3864–3869.

[35] A. Snow, P. Rastogi, and G. Weckman, "Assessing dependability of wireless networks using neural networks," in Military Communications Conference. IEEE, 2005, pp. 2809–2815 Vol. 5.

[36] A. Moustapha and R. Selmic, "Wireless sensor network modeling using modified recurrent neural networks: Application to fault detection," IEEE Transactions on Instrumentation and Measurement, vol. 57, no. 5, pp. 981–988, 2008.

[37] Y. Wang, M. Martonosi, and L.-S. Peh, "Predicting link quality using supervised learning in wireless sensor networks," ACM SIGMOBILE Mobile Computing and Communications Review, vol. 11, no. 3, pp. 71–83, 2007.

[38] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?" in Database Theory. Springer, 1999, pp. 217–235.

[39] T. O. Ayodele, "Types of machine learning algorithms," in New Advances in Machine Learning. InTech, 2010.

[40] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," IEEE Transactions on Systems, Man and Cybernetics, vol. 21, no. 3, pp. 660–674, 1991.

[41] R. Lippmann, "An introduction to computing with neural nets," ASSP Magazine, IEEE, vol. 4, no. 2, pp. 4–22, 1987.

[42] W. Dargie and C. Poellabauer, Localization. John Wiley & Sons, Ltd, 2010, pp. 249–266.

[43] T. Kohonen, Self-organizing maps, ser. Springer Series in Information Sciences. Springer Berlin Heidelberg, 2001, vol. 30.

[44] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, vol. 313, no. 5786, pp. 504–507, 2006.

[45] I. Steinwart and A. Christmann, Support vector machines. Springer, 2008.

[46] Z. Yang, N. Meratnia, and P. Havinga, "An online outlier detection technique for wireless sensor networks using unsupervised quarter-sphere support vector machine," in International Conference on Intelligent Sensors, Sensor Networks and Information Processing. IEEE, 2008,
pp. 151–156.